Umgang mit technischen Schulden in Großunternehmen

Maic Siemering, 2023

Agenda

1. Grundlagen
2. Methodik
3. Verwandte Arbeiten
4. Umsetzung
5. Ergebnisse
6. Offene Fragen

Technische Schulden

⇨ Aus der Finanz entlehnte Methaper ([1] Ward Cunningham, 1992)

> A little debt speeds development so long as it is paid back promptly with a rewrite. [...] The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt.

⇨ Zwei Arten: `bewusst` & `unbewusst` ([2] S. McConnell, 2008)

⇨ Kategorisierung: `reckless/prudent` & `deliberate/inadvertent` ([3] M. Fowler, 2009)

⇨ Ontology/ Typisierung: Towards an Ontology of Terms on Technical Debt ([4] N. Rios 2014)

Definition ([5] Dagstuhl Seminar 2016)

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

Definition ([5] Dagstuhl Seminar 2016)

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

❗ Ausgewerteten Daten wiedersprechen dieser explizieten Limitierung.

Unterkategorien aus Literatur Review

A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools ([6] V. Lenarduzzi et al., 2021)

- ⇨ Requirements TD
- ⇨ Architectural TD
- ⇨ Design TD
- ⇨ Code TD
- ⇨ Test TD
- ⇨ Build TD
- ⇨ Documentation TD
- ⇨ Infrastructure TD
- ⇨ Versioning TD
- ⇨ Defect TD

Unterkategorien aus Literatur Review

A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools ([6] Valentina Lenarduzzi et al., 2021)

- ⇨ Requirements TD
- ⇨ Architectural TD
- ⇨ Design TD
- ⇨ Code TD
- ⇨ Test TD
- ⇨ Build TD
- ⇨ Documentation TD
- ⇨ Infrastructure TD
- ⇨ Versioning TD
- ⇨ Defect TD

❗ Eher umstritten, dass Bugs als technische Schuld gesehen werden [13]

Groß Unternehmen

> Großunternehmen haben über 249 Beschäftigte oder über 50 Millionen Euro Umsatz. [7]

Charakteristika [8]:

- Umfangreiche Planung
- Hochgradige sachbezogene Arbeitsteilung
- Ferne zum Betriebsgeschehen
- Komplexe Organisationsstruktur
- Hoher Formalisierungsgrad

Motivation

Einfluss des organisatorischen Umfeldes von Groß Unternehmen auf den Umgang mit technischen Schulden.

Grounded Theory Methodology

Qualitative Forschungsmethode

Iteratives Vorgehen

- ⇨ Daten sammeln
- ⇨ Offenes Kodieren
- ⇨ Axiales Kodieren
- ⇨ Selectives Kodieren
- ⇨ Theoretical Sampling

Aussagen verwandter Arbeiten

1. An Enterprise Perspective on Technical Debt [9]
    - Nonintentional debt typically was much more problematic than the intentional debt.
    - Most details regarding decisions were only available through "tribal memory,"
    - Neither a channel nor a common vocabulary to express the costs of incurring a technical debt to non-technical stakeholders
2. Umgang mit technischen Schulden in Startups [10]
    - Abgleich mit Theorien folgt

Unternehmen

- Deutsches Großunternehmen
- Globale Produktionsstandorte
- Betrachtete Software Entwicklung in Deutschland
- Organisation mit mehreren tausend Mitarbeitern

Team Arbeitsmodel

- Agile Arbeitsweise; an Extreme Programming oder Scrum angelehnt
  - Pairing, Pre-IPM & IPM/ Sprint Planning, Review
  - Retrospektive (nicht Teil der Daten)

Daten Sammlung I

- Begleitung des `Team S`
  - Aufgabe: *Enablement* von Projekt Teams
  - Kein externer Product Owner (PO)
  - Selective Audioaufnahmen
- Interviews mit `Teams E, W, N, D`
  - Aufgabe: Software Entwicklung für einen Fachbereich
  - PO aus dem Fachbereich
  - Audioaufnahmen
  - Zugriff auf Backlog und Architecture Decision Records (ADR)

Daten Sammlung II

- Hospitation für jeweils einen Tag in `Team E & W`
    - Gedächtnisprotokoll des Authors

Team Eigenschaften

Team	Projektzeit	Anzahl Entwickler	Besonderheit	PO
`S`	8 Jahre	3-8		
`E`	4 Jahre	5	4 Teams	❎
`W`	10 Jahre	5-15	4 Migrationen, vorher extern entwickelt	❎
`N`	2 Jahre	6		❎
`D`	2 Jahre	5		❎

Ähnlichkeiten:

✚ interne Entwickler

✚ agiles Vorgehen

Limitierungen:

— keine externe Entwicklung

— Daten beschränkt auf einen Teil der Organisation

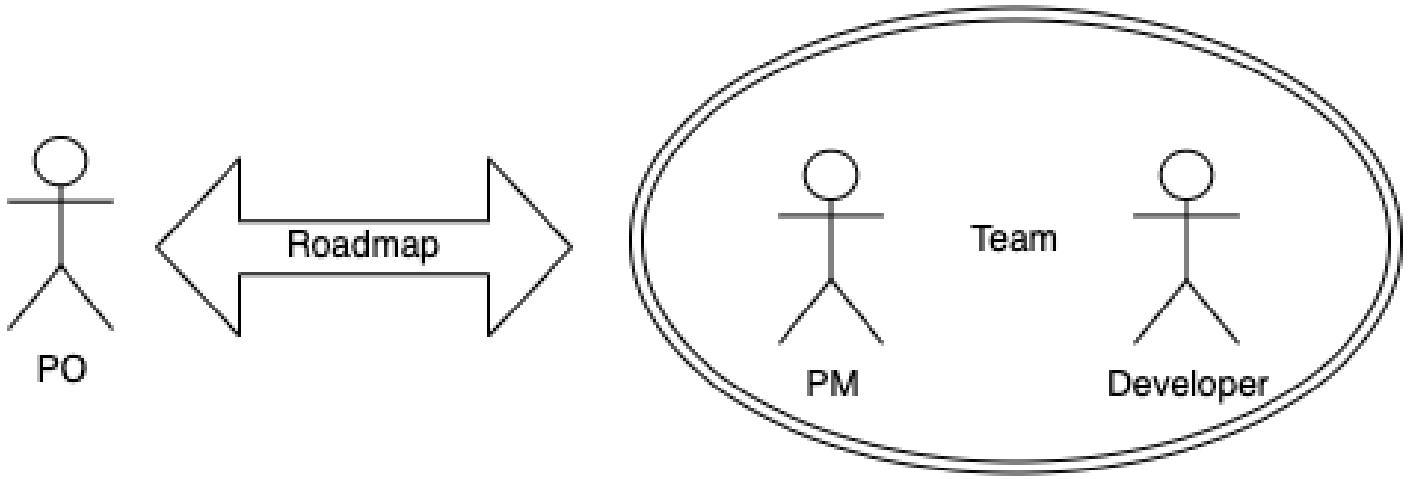
Interview Partner

Name	Rolle	Erfahrung
SL	Entwickler	15
SM	Entwickler	5
SK	Entwickler	5
ST	Entwickler	10
SH	Entwickler	15

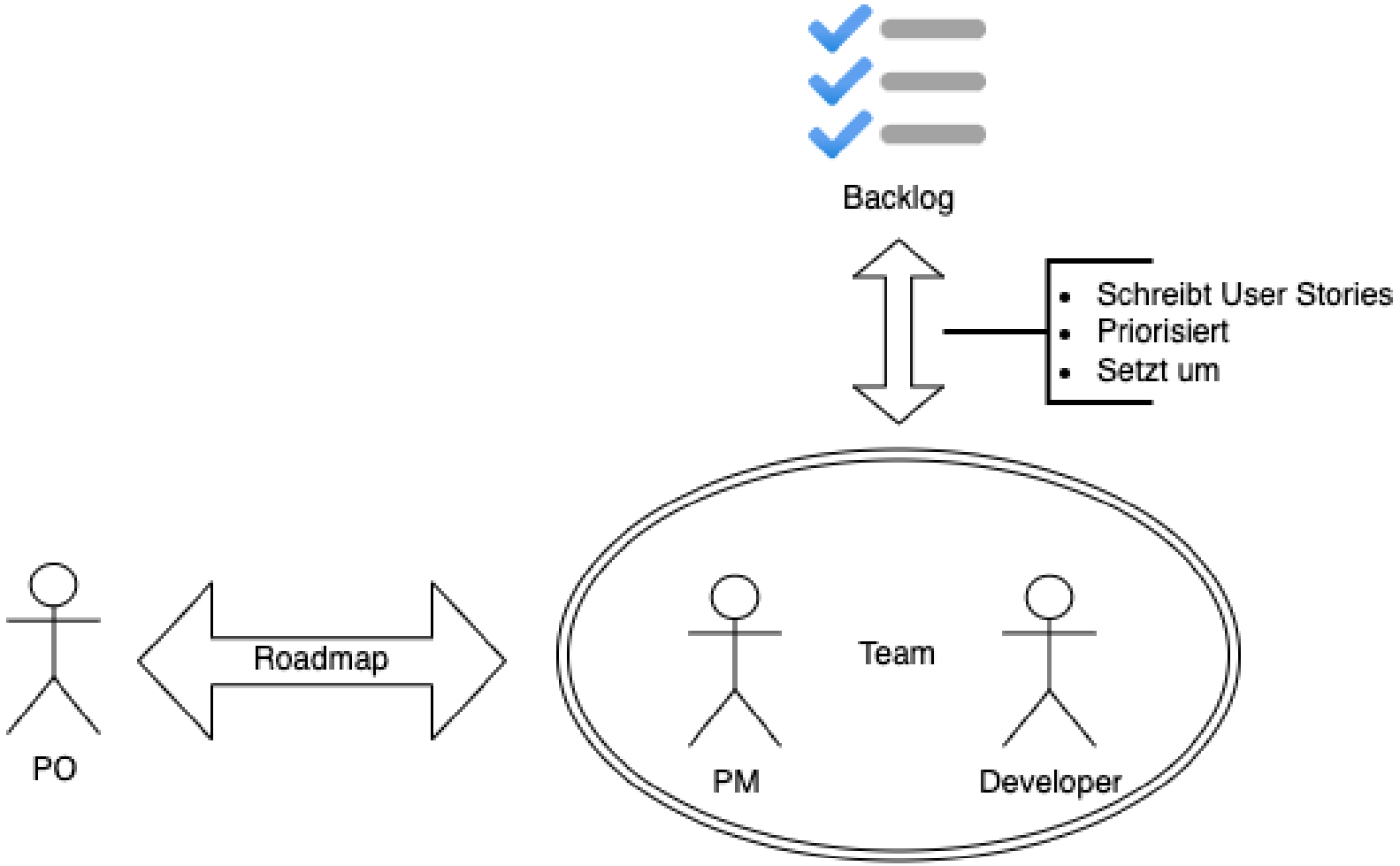
Name	Rolle	Erfahrung
EN	Entwickler	X
ES	Entwickler	15
EF	Entwickler	5
WS	Entwickler	8
NT	Produkt Manager	7
NS	Entwickler	8
DJ	Entwickler	X

Ergebnisse

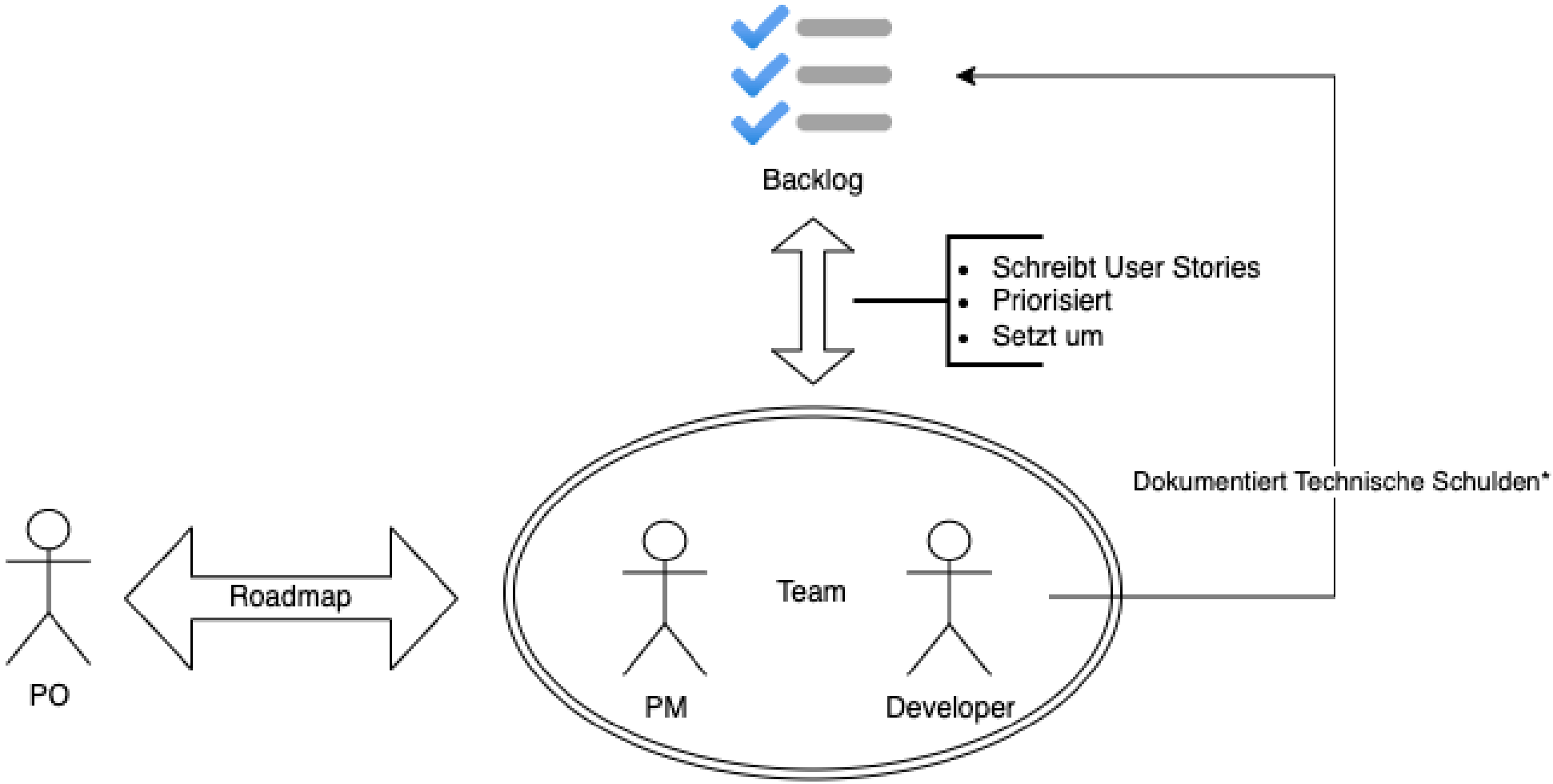
Grundsetup



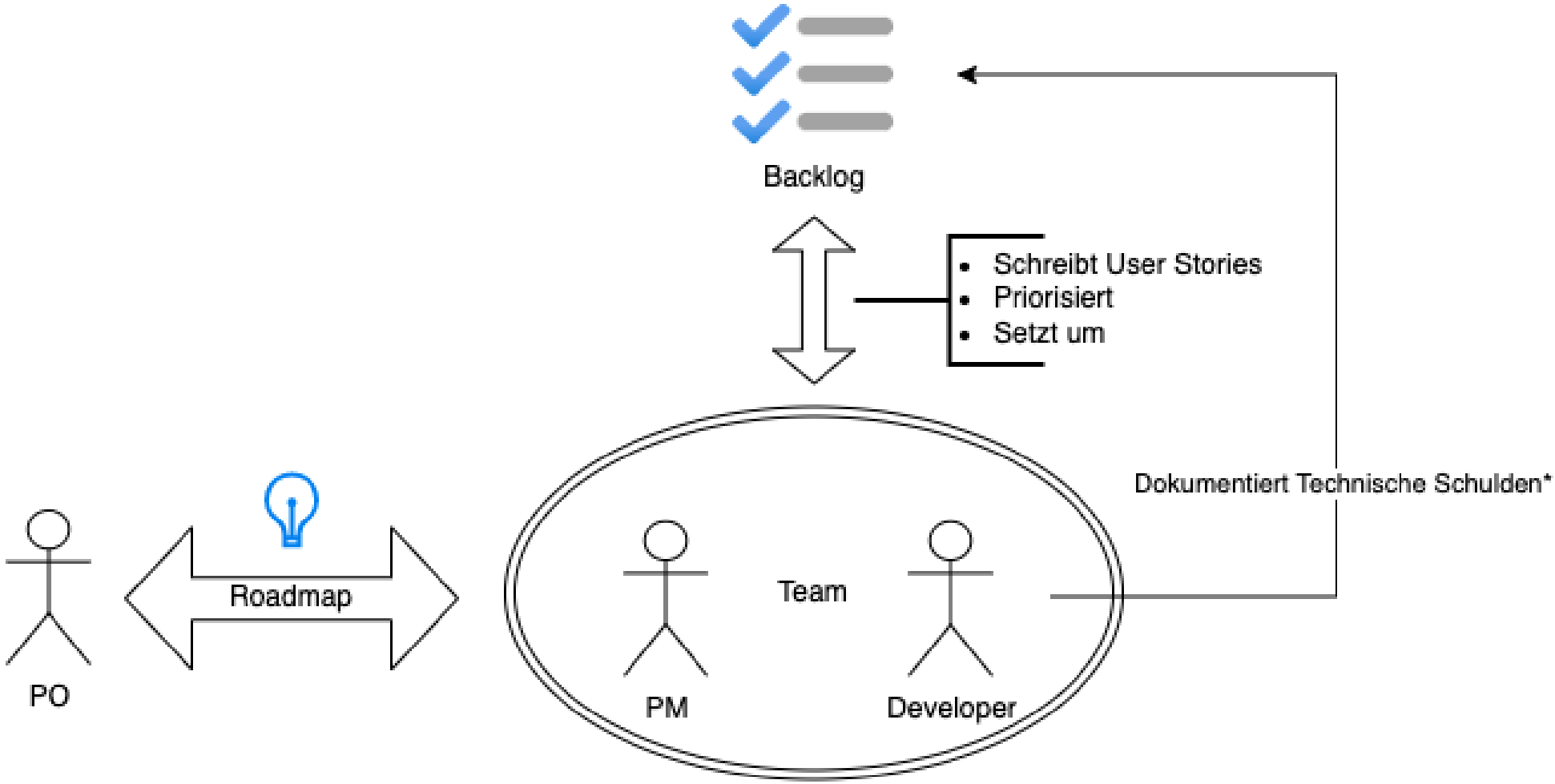
Vorgehen im Team



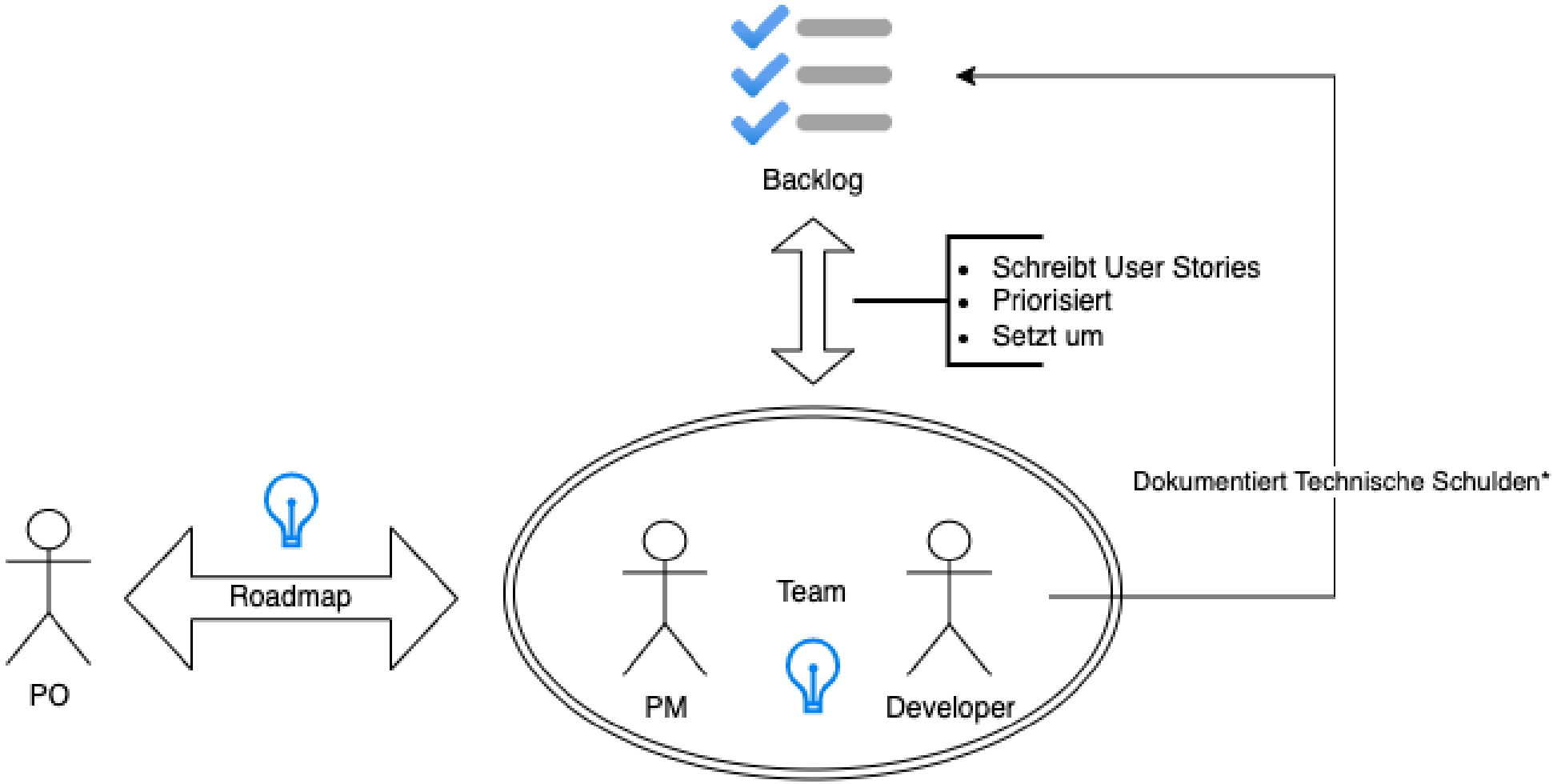
Technische Schulden



PO ↔ Team

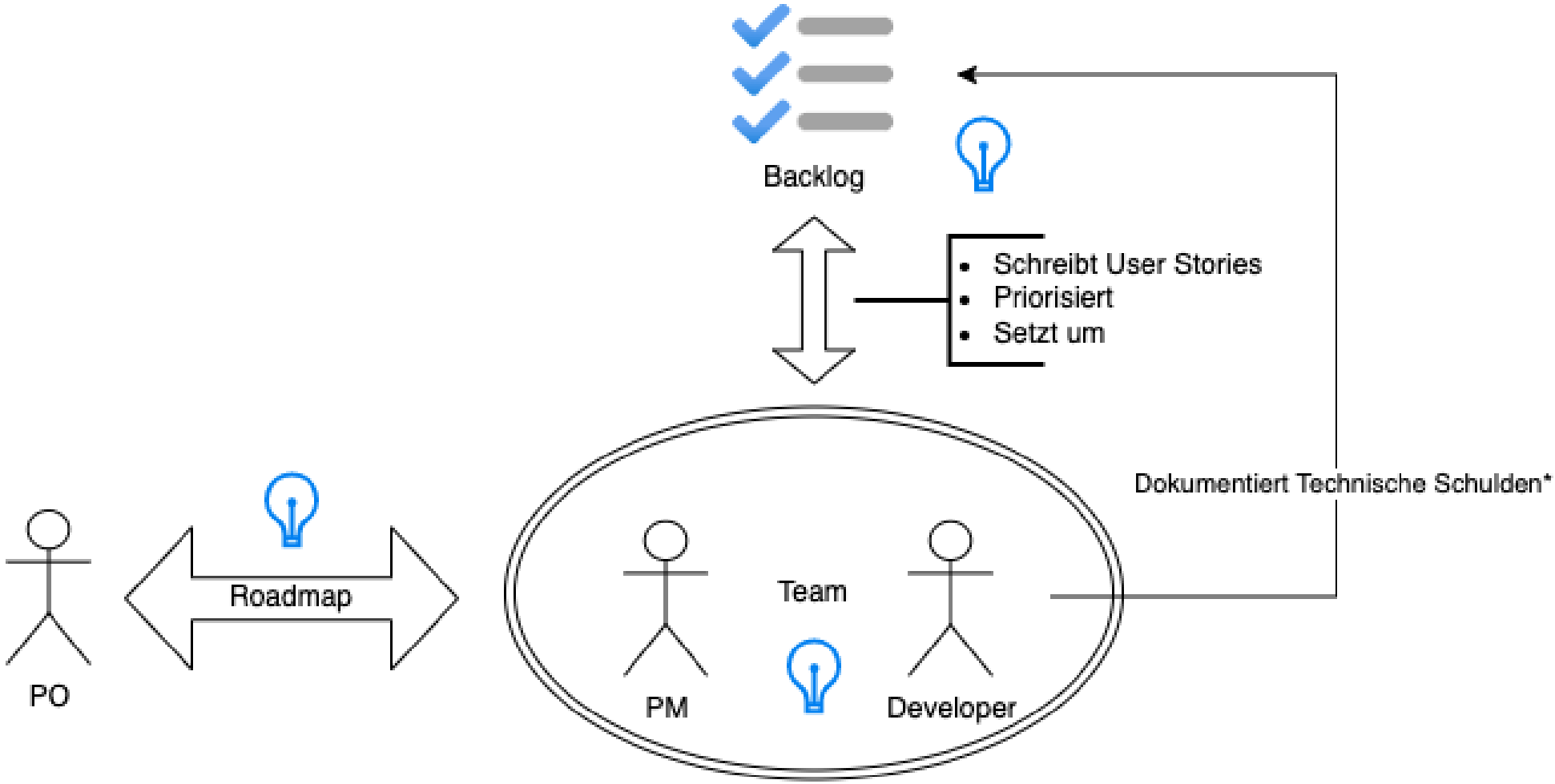

Backlog
Schreibt User Stories
Priorisiert
Setzt um
Dokumentiert Technische Schulden*
Roadmap
PO
Team
PM
Developer

PM ↔ Entwickler

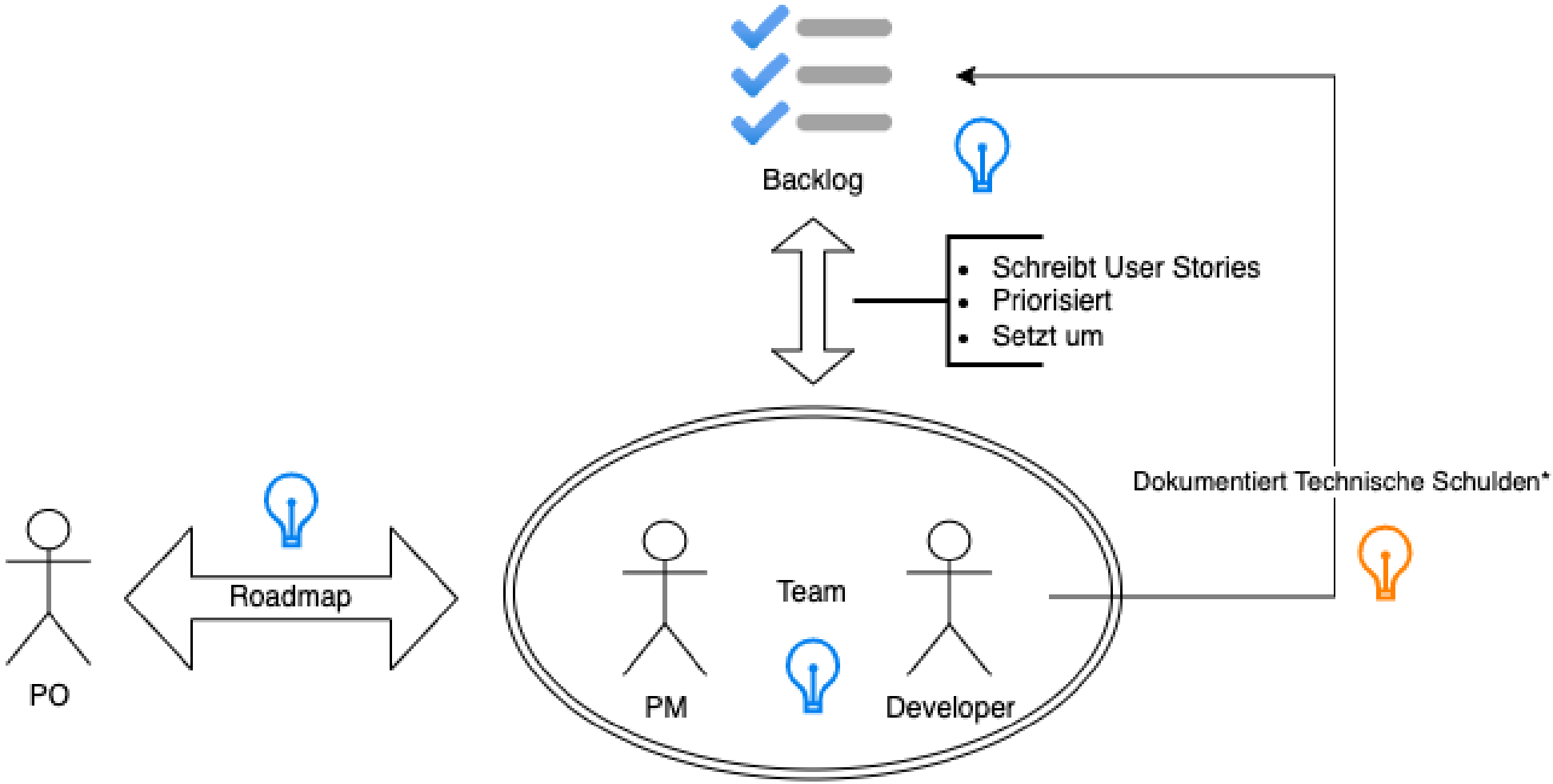


draft

Backlog


Backlog
Schreibt User Stories
Priorisiert
Setzt um
Dokumentiert Technische Schulden*
Roadmap
PO
Team
PM
Developer

🤔 Was wird als technische Schuld wahrgenommen?



Auszug aus einem Interview

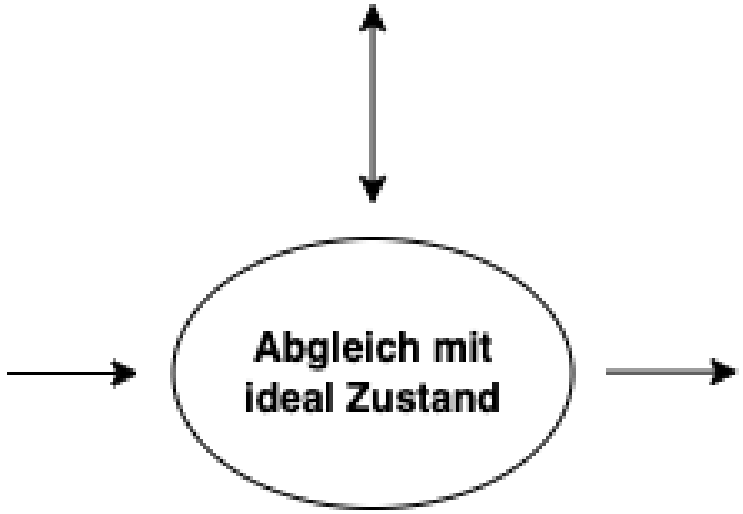
> Wie definierst du technische Schulden?
>
> "... Einfach nur der aktuelle Zustand, der sich unterscheidet von einem Idealzustand, den man heute auf einer grünen Wiese mit dem aktuellen Wissen haben könnte. Das würde ich als technische Schulden bezeichnen." - SL

Codes

- Ist Zustand
- Differenz
- Ideal Zustand
- Grüne Wiese
- Aktuelles Wissen

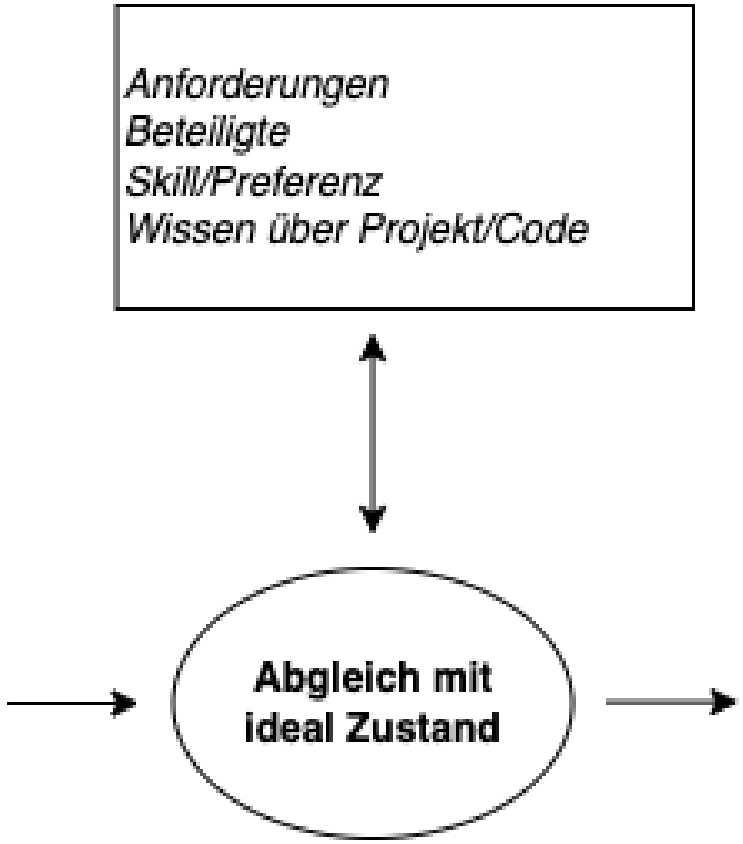
⇨ **Differenz zwischen Ist und Ideal Zustand**

Konzept: Ideal Zustand



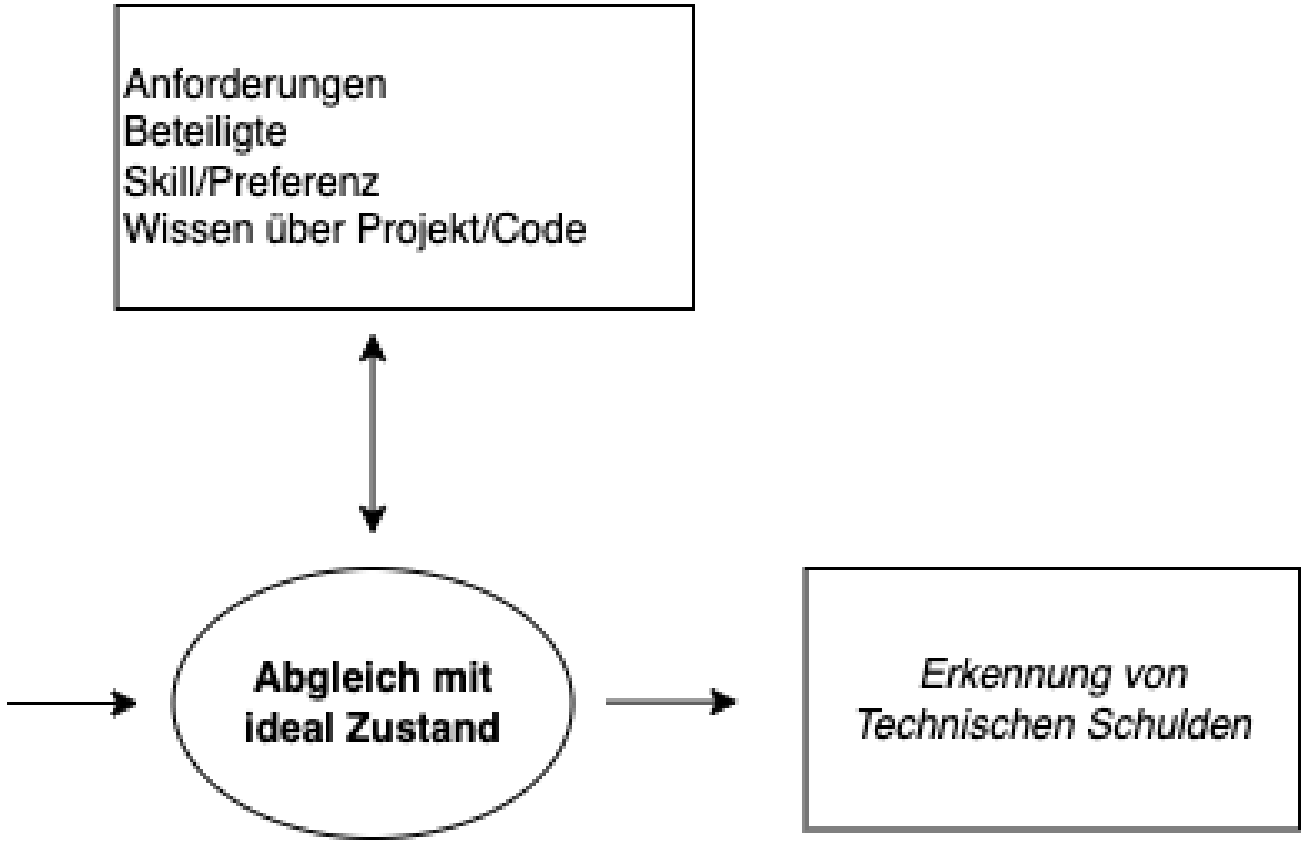
Kernkategorie:
Ideal Zustand

Ideal Zustand - Kontext



Kernkategorie:
Ideal Zustand

Ideal Zustand - Auswirkung



Kernkategorie:
Ideal Zustand

Bedingungen/Einfluss I

> Im Service X hat der `Entwickler A` Clean Code quasi zelebriert. Jetzt wo er aber nicht mehr da ist, hat sich `Entwickler B` ran gesetzt und hat an sich selbst gezweifeld, ob er einfach nicht gut genug ist, um den Code zu verstehen. Aber ich und `Entwickler C` haben genau das gleiche Problem, das ist einfach nicht lesbar.

⇨ Persönlicher Stil, Wissen über Code Basis

> Wir hatten gerade alles fertig, eine schöne State Machine, Retry, alles. Da kam die Nachricht, es ist deprecated.

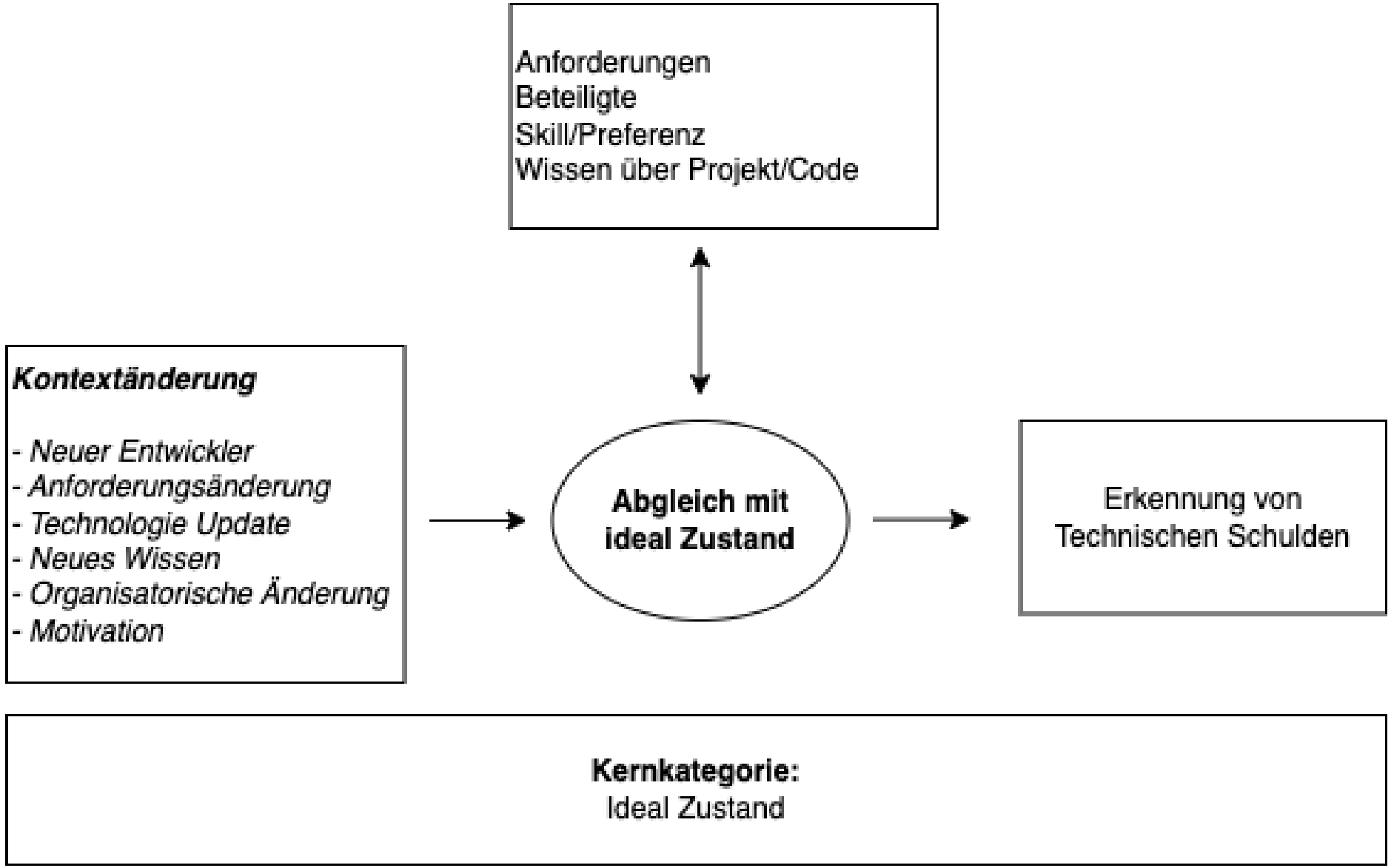
⇨ Technologie Änderung, Neues Wissen

Bedingungen/Einfluss II

> Team S übernimmt eine Code Basis und reduziert die Nutzung des `Pattern U`, schreibt Code teilweise gänzlich neu.
> Team W übernimmt eine Code Basis und refactored richtung `Pattern U`

⇨ Persönliche Preferenz, Wissen über Code Basis

Ideal Zustand - Bedingungen



Blog posts

- [11] Don't Call It Tech Debt, A. Wilkinson, 2022

  > While there is broad agreement about what constitutes good code, the more detailed you get the more it becomes about personal opinions.

- [12] My 20 Year Career is Technical Debt or Deprecated, Matt Watson, 2023

  > My entire career is now technical debt, or the code has been deprecated.

Weitere Auffälligkeiten I

Identifikation

- ⇨ Aktuell bearbeiteter Code
- ⇨ Hotspots

Qualifikation

- ⇨ Gefühl, kaum Messungen

Wirkung

- ⇨ beeinträchtigt Maintanance
- ⇨ beeinträchtigt Qualität
- ⇨ beeinträchtigt Weiterentwicklung
- ⇨ Sicherheitsprobleme
- ⇨ höhere Kosten
- ⇨ Unmut im Team*

Weitere Auffälligkeiten II

- Definition von TS immer unterschiedlich
- Kein etablierter Prozess
- Führungsebene wenig technischer Hintergrund, `technische Schulden` teils unbekannt
- Not invented here -> Massiven Einfluss auf die Wahrnehmung technischer Schulden
- PO & PM vertrauen zu den Entwickler*Innen wichtig
- Passen zu Aussagen zu [9]

Beispielhafter Abgleich mit Thesen aus [10]

- 1.3 unklaren Anforderungen und daraus resultierenden suboptimalen Designentscheidungen -> Wahl der Datenbank
- 1.4 neue Entwickler hinzu, werden diese Schulden und evtl. daraus resultierende Probleme oder zeitliche Mehraufwände deutlich sichtbar. -> andere Sicht
- 5.1 ausgebildete Entwickler*innen intrinsische Motivation -> konnte auch hier gefunden werden
- 5.2 demotivation -> wurde beobachtet, führte sogar zu einem Neustart
- Empfehlung: Personal halten -> Massiver Einfluss auf Ideal Zustand

Diskussion

- Neuer Bereich des Unternehmens (<500 Mitarbeiter, <10 Jahre)
- Limitiert auf 4 Teams
- Interviews beinhalten sicher Voreingenommenheit (Seniorität)
- Workflow von internen Entwicklern

Nächste Schritte

- Feedback abwarten
- Abgleich mit Aussagen aus [9]
- Abgleich der Empfehlungen aus [10]
- Suchen weiterer Daten für das Kernkonzept

Danke

Quellen

[1]W. Cunningham, “The WyCash portfolio management system,” ACM SIGPLAN OOPS Messenger, vol. 4, no. 2, pp. 29–30, Apr. 1993, doi: https://doi.org/10.1145/157710.157715.

[2]S. McConnell, “Managing Technical Debt,” White Paper, Construx Software, 2008. Accessed: Jun. 28, 2023. [Online]. Available: https://www.construx.com/uploadedfiles/resources/whitepapers/Managing Technical Debt.pdf

[3]M. Fowler, “bliki: TechnicalDebtQuadrant,” martinfowler.com, Oct. 14, 2009. https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

[4]N. S. R. Alves, L. F. Ribeiro, V. Caires, T. S. Mendes, and R. O. Spinola, “Towards an Ontology of Terms on Technical Debt,” 2014 Sixth International Workshop on Managing Technical Debt, Sep. 2014, doi: https://doi.org/10.1109/mtd.2014.9.

[5]P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing Technical Debt in Software Engineering,” Report from Dagstuhl Seminar, p. 16162, Apr. 2016, doi: https://doi.org/10.4230/DagRep.6.4.110.

[6]V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, and F. Arcelli Fontana, “A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools,” Journal of Systems and Software, vol. 171, p. 110827, Jan. 2021, doi: https://doi.org/10.1016/j.jss.2020.110827.

[7]Statistisches Bundesamt, “Kleine und mittlere Unternehmen,” Statistisches Bundesamt. https://www.destatis.de/DE/Themen/Branchen-Unternehmen/Unternehmen/Kleine-Unternehmen-Mittlere-Unternehmen/_inhalt.html (accessed Jun. 28, 2023).

[8]Haufe, “BGM in Großunternehmen/Konzernen / 2.2 Qualitative Charakteristika von Konzernen | Haufe ...,” Haufe.de News und Fachwissen. https://www.haufe.de/personal/haufe-personal-office-platin/bgm-in-grossunternehmenkonzernen-22-qualitative-charakteristika-von-konzernen_idesk_PI42323_HI9363130.html (accessed Jun. 28, 2023).

[9]T. Klinger, P. Tarr, P. Wagstrom, and C. Williams, “An enterprise perspective on technical debt,” Proceeding of the 2nd working on Managing technical debt - MTD ’11, 2011, doi: https://doi.org/10.1145/1985362.1985371.

[10]L. Ververs, “Umgang mit technischen Schulden im Startup-Umfeld,” MSc Thesis, Freien Universit¨at Berlin, 2022.

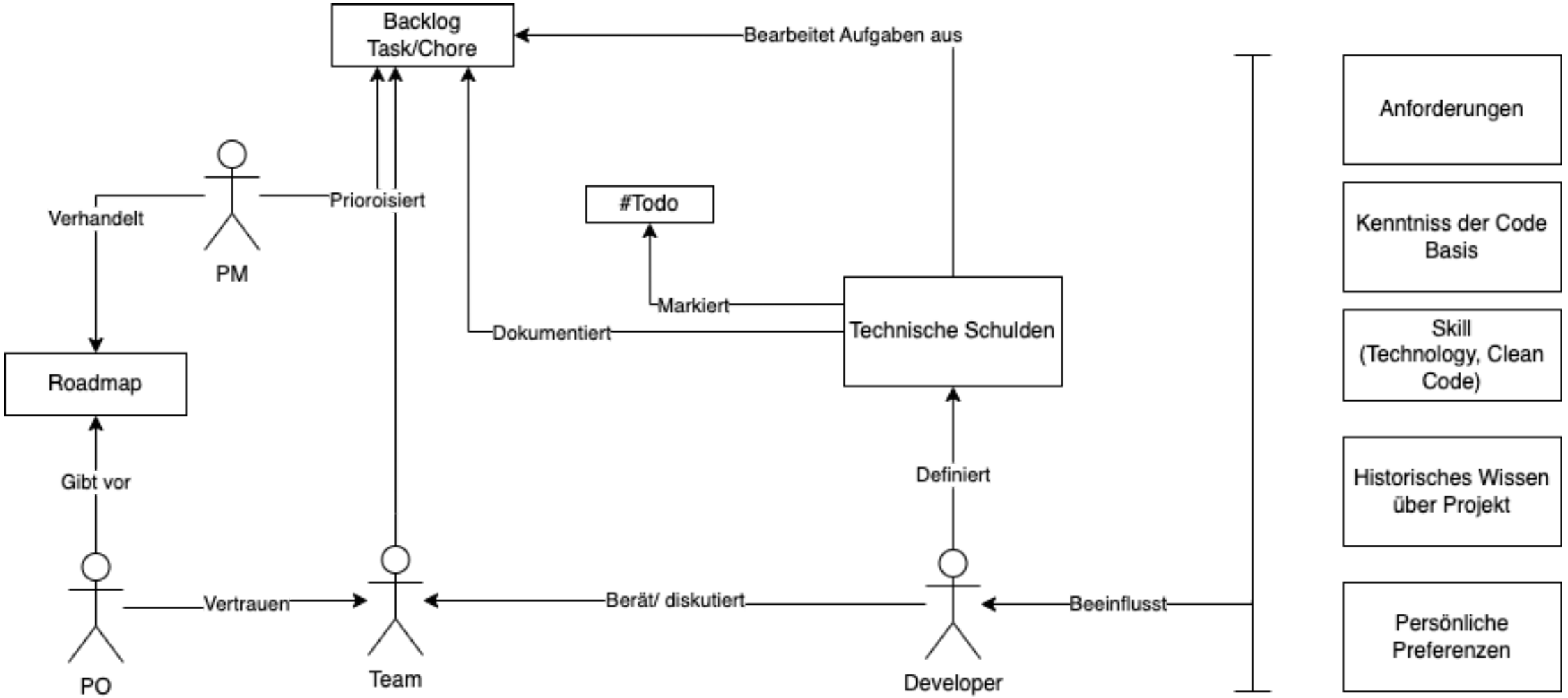
[11]A. Wilkinson, “Don’t Call It Tech Debt · Andrew Wilkinson,” www.theandrewwilkinson.com, Feb. 03, 2022. https://www.theandrewwilkinson.com/2022/02/03/dont-call-it-tech-debt/ (accessed Jun. 28, 2023).

[12]M. Watson, “My 20 Year Career is Technical Debt or Deprecated,” blog.visionarycto.com, May 15, 2023. https://blog.visionarycto.com/p/my-20-year-career-is-technical-debt (accessed Jun. 28, 2023).

[13]D. Rumney, “Bugs Are Not Technical Debt,” dancrumb.com, Jul. 08, 2016. https://dancrumb.com/engineering/agile/2016/07/08/bugs-are-not-technical-debt.html (accessed Jun. 28, 2023).

Backup

Team Model (Interviews)



draft