Technologische und struktruelle Drift und Innovation steuern

Verantwortlich: Stefan Hoffmann

Hintergrund / Problem

Über die Zeit und durch parallele Arbeit entstehen statt einer Anwendungsstruktur und einem optimal limitierten Technologieset ein breiteres Spektrum an Lösungsvarianten und Technologien, die im Einsatz sind.

Die Folgen für die Abteilung:

- Durch das Vergessen bestehender Lösungen entsteht mangelnde Wiederverwendung.
- Durch mangelhafte Dokumentation und Wissensvermittlung entsteht mangelnde Wiederverwendung.
- Durch wechselnde Technologie-Entscheidungen der Abteilung erhöht sich die Menge des notwendigen Wissens für die Arbeit an allen Unternehmenslösungen. Dadurch entsteht Lernaufwand und Orientierungsaufwand.
- Durch die fortschreitende Drift sinkt die Zuverlässigkeit von Zeitplänen, weil dadurch Entwickler mit höherer Sicherheit Dinge im Code finden, für die sie erstmal etwas Zeit brauchen, weil sie etwas lernen, aktualisieren oder umschreiben müssen.

Insgesamt sinkt dadurch auf längere Sicht die Performance der Abteilung.

Ausgangssituation

- Wir haben keine Übersicht über den Gesamtzustand unserer Codebase und welche Anwendungen gepflegt werden müssten.
- Wir wissen nicht wie viel Zeit wir bereits zu viel investieren und auch wie man diese messen könnte. (Obwohl wir feststellen, dass wir oft D erreichen wollen, aber dafür erstmal C, B und A aus dem Weg räumen müssen. Wir können also nicht direkt am Problem arbeiten.)

- Wir haben keine Planzeiten und wenig konkrete Methoden, die wir einsetzen können, um der Drift und dem Wissensverlust entgegenzuwirken bzw. technologische Verbesserungen in allen Anwendungen gleichermaßen zu verteilen.

Zielsituation

- Wir haben eine Liste der konkreten Dinge, die wir in unseren Lösungen haben möchten und auch von denen, die wir nicht in Lösungen haben möchten. Diese Liste ist immer aktuell.
    - Optimalerweise haben wir automatisierte Werkzeuge, die uns auf Abweichungen aufmerksam machen und uns eine Übersicht über die gesamte Codebase erstellen.
    - Vielleicht haben wir sogar Werkzeuge, die bestimmte Migrationsaufgaben automatisieren können?
- Wir benötigen für die zielgerichtete Bearbeitung eine Möglichkeit den Umfang der Herausforderungen zu schätzen.
    - Auch hier könnte Automatisierung helfen, aber nur bei Dingen, die so kontrollierbar sind.
    - Checklisten könnten ebenfalls helfen, müssten aber in der Qualitätssicherung verwendet werden.
- Wir benötigen eine managementseitige Klärung wann und wieviel Aktualisierung wir in unserer Arbeit unterbringen können. Diese Geschwindigkeit der Korrekturarbeit sollte mittelfristig größer sein als die Geschwindigkeit der Drift.

Sofortmaßnahmen

- Besprechung, Erinnerung daran, dass bewusste Drift/Innovation dokumentiert und besprochen werden müssen.

Ursachenanalyse

Die Ursachen für die Drift sind:

- Vergessen
    - Begrenzung menschliche Kapazität, man kann sich nur so viel merken oder auch nur so vielen Dingen gleichzeitig bewusst sein.
- Mangelhafte Dokumentation und Wissensvermittlung
    - Durch Vernachlässigung (bewusstes Vergessen)
        - Zeitdruck
        - Keine Organisation der Abzahlung der Herausforderung
    - Durch schlechte Werkzeuge (schlechte Auffindbarkeit der Herausforderungen)
    - Durch Dezentralisierung ("ich wusste nicht wo ich suchen sollte")
    - Durch mangelhaften Prozess ("ich bin gar nicht auf die Idee gekommen zu suchen")
- Generelle Weiterentwicklung intern während der Arbeit
    - Neu aufkommende Nutzungsmuster der Benutzer machen andere technische Umsetzung attraktiv.
    - Schaffung neuen Codes zur komfortableren Lösung eines bestimmten Problems
    - Schaffung neuer Standards (z.B. Core-Bibliotheken, Frontend-Framework)
- Generelle Herausforderungen der Abteilung
    - z.B. um Talente in einem bestimmten Bereich anzulocken müssen als attraktiv geltende Technologien gewählt werden.
- Entscheidungen von Technologieanbietern (extern)
    - Externe Technologieanbieter verfolgen ihre eigenen Marktziele die sich positiv oder negativ auswirken können.
        - Abhängig von der Strategie des Technologieanbieters

Von Drift sprechen wir hierbei generell, weil hier eine entsprechende Bewegung stattfindet. Sozusagen eine Nicht-Funktionale, da nicht an unserem Business orientierte, Verschlickung.

Generell findet Drift statt. Es geht nicht darum Drift zu unterbinden, sondern die negativen Auswirkungen zu kontrollieren und einzudämmen.

Problemlösung / Gegenmaßnahmen

- Wir müssen unseren Soll-Zustand besser dokumentieren.
  - Aktualisierung der Standard-Checkliste, ggf. Ausweitung der Liste, weil wir durchaus kleingliedriger sein möchten.
    - Kann man Regelungen vielleicht von vornherein in einer Art allow / deny Code verfassen, welche dann durch einen Interpreter ausgeführt werden?
- Wir sollten unseren Soll-Zustand automatisiert prüfbar machen.
  - Schaffung eines Tools / von mehreren Tools
- Wir müssen dafür sorgen, dass der Zustand der Code-Base regelmäßig geprüft wird und dass die Ergebnisse zu uns kommen, wie andere Prüfergebnisse auch.
  - Wir benötigen einen regelmäßigen Prozess, der uns Auskunft über unsere Code-Base gibt.
  - Einbau einer kontinuierlichen Kontrolle in unseren Build-Prozess?
  - Automatisierung der Aktualiserung von Abhängigkeiten

Erfolgsmessung

Standardisierung und Know-How-Transfer