Gamification für Software Reengineering

Dr. Baris Güldali
S&N CQM
Zukunftsmeile 2, 33102 Paderborn

Dr.-Ing. Dehla Sokenou
WPS – Workplace Solutions
Hans-Henny-Jahnn-Weg 29, 22085 Hamburg

Zusammenfassung

Vor einer Software-Migration oder Reengineering steht oft die Frage, ob die Software 1-zu-1 migriert werden soll oder ob die Migration auch für eine Modernisierung genutzt werden kann. Dabei fehlt Softwareentwicklungsteams oft die Zeit und manchmal auch die notwendige Expertise, das System zu refraktorieren bzw. überarbeiten und technische Schulden zu beseitigen. Wir stellen spielerische Methoden für die genannten Probleme vor, die auf unseren Erfahrungen mit anderen Gamification-Ansätzen basieren, die wir hier auf den Bereich der Software-Reengineering angewendet haben.

Einleitung

In unserem früheren Artikel „Architektur nicht versenken“ [1] haben wir Szenarien definiert, wie Software-Entwickler-Teams Gamification nutzen können, um Probleme mit der Software-Architektur in den Griff zu bekommen und ihre Architektur kontinuierlich zu verbessern. Als wichtige Richtlinie gilt: *Erkenne zuerst das Problem, dann plane die Lösung, dann setze sie um*.

Probleme im Software Reengineering und in der Software-Migration sind ähnlicher Natur und haben ihre Ursachen oft in der Software-Architektur. Oft ist die Architektur mit der Zeit gewachsen und wird – wenn überhaupt – von einigen wenigen Team-Mitgliedern durchschaut, die am längsten dabei sind.

Ein Team, das ihre Software modernisieren oder in eine neue Technologie oder in eine neue Plattform migrieren möchte, muss den Ist-Stand der Architektur verstehen, den Einfluss der Reengineering-Maßnahmen bewerten und Entscheidungen für zukünftige Architektur-Eigenschaften treffen. Wir sind der Meinung, dass diese Entscheidungen im Team gemeinschaftlich getroffen werden sollten, um eine bessere Identifikation im Team mit den anstehenden Aktivitäten zu schaffen. Gamification kann dabei die Kommunikation und den systematischen Austausch fördern.

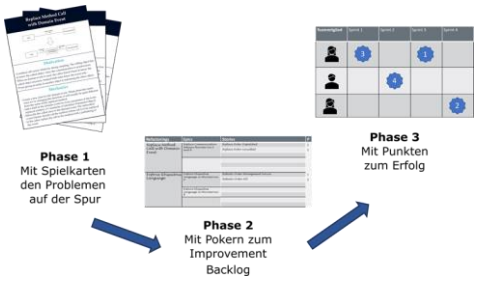

Abb. 1: Drei Phasen des Spiels

Der Gamification-Ansatz

Der Begriff *Gamification* meint den Einsatz von Elementen aus Spielen in einem nicht spielerischen Kontext [2]. Innerhalb des Arbeitsalltags sollen spielerische Elemente helfen, Prozesse zu optimieren, Alltagstätigkeiten zu erleichtern und Produkte zu verbessern. Das Ziel ist also, einen direkten, positiven Einfluss auf das Arbeitsergebnis zu haben. Unser Ansatz von Gamification für Software Reengineering besteht aus drei Phasen:

1) Mit Spielkarten den Problemen auf der Spur
2) Mit Pokern zum Improvement Backlog
3) Mit Punkten zum Erfolg

In der Phase 1 setzen wir einen Spielkartenset, das ähnlich wie die Karten beim Risk Storming [3] das Team bei der Problemanalyse über die Architektur und der Lösungsfindung leitet. Die Spielkarten beschreiben die Domain-Driven Refactorings, die von Henning Schwentner [4] zusammengetragen wurden. Sie umfassen jeweils eine Motivation – also das zu lösende Problem – und schlagen eine Lösung vor, die auf den Best-Practices aus dem Domain-Driven Design basiert (siehe Abb. 2).

Einige für Software Reengineering relevante Refactorings finden sich in den folgenden Spielkarten wieder:

- Extract Bounded Context: Mit Hilfe dieser Karte kann das Team erkennen, dass sich die Fachlichkeit nicht adäquat im System abbildet. Mehrere unterschiedliche Karten adressieren dieses Problem. Als Lösung schlägt dieses

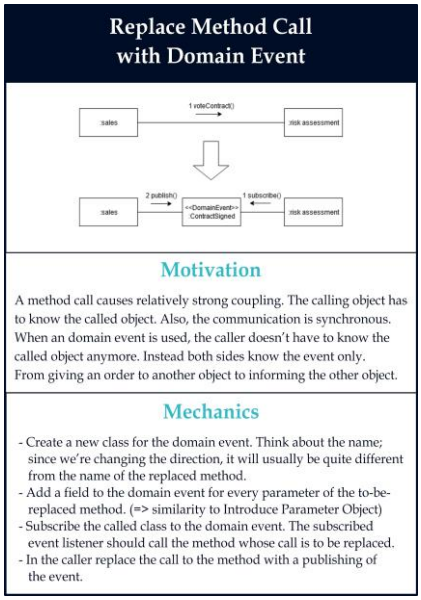

Abb. 2: Spielkarte aus den Domain-Driven Refactorings

Refactoring das Herausschälen eines Bounded Contexts aus dem Monolithen vor.

● Enforce Ubiquitous Language: Das bestehende Team ist an die technischen Begriffe im Code gewöhnt, dass die Teammitglieder die Diskrepanz zur Fachlichkeit nicht mehr erkennen. Das neue Teammitglied ist aber immer unsicher, an welche Stellen im Code auf die angeforderten Änderungen von der Fachseite durchzuführen sind. Dieses Refactoring erfordert das Umbenennen im Code, aber auch in der Datenbank anhand der fachlichen Begriffe zur besseren Lesbarkeit und zum besseren Verständnis des Codes.

● Move Logic from Service to Entity: Dieses Refactoring erfordert, dass die Logik so weit wie möglich in den Entities gekapselt werden soll.

● Replace Method Call with Domain Event: Anhand dieser Karte kann das Team feststellen, dass die Kopplung zwischen den Microservices viel zu eng ist, da diese sich gegenseitig direkt aufrufen. Eine mögliche Lösung wäre stattdessen Domain-Events zu schicken, um die Microservices unabhängiger voneinander weiterentwickeln zu können (vgl. Abb. 2).

Die Team-Mitglieder ziehen aus dem Spielkartenset Karten heraus und diskutieren über die beschriebenen Refactorings und ihre Priorisierung. Wo steht die eigene Architektur im Vergleich zu dem Refactorings? Welche Reengineering- oder Modernisierungs-Ziele können mit welchen Refactorings erreicht werden? Was ist der Einfluss eines ausgewählten Refactorings auf die anstehenden Entwicklungsarbeiten? Aus diesen Diskussionen entsteht in der zweiten Phase ein Improvement-Backlog.

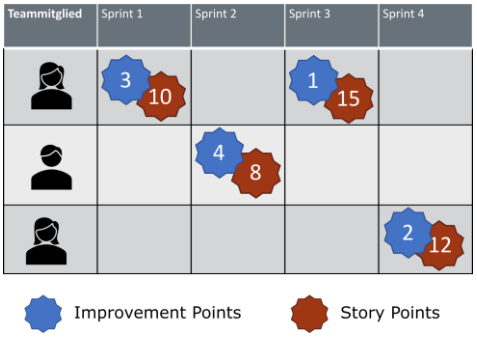

Abb. 3: Auswertung von Improvement- und Story-Points im Team

Nach unserer Erfahrung findet Software Reengineering oft parallel zum laufenden Betrieb und Weiterentwicklung statt. Daher ist es unausweichlich, die Refactorings in das tägliche Projektgeschäft zu integrieren. Je nach Organisation eines Reengineering-Projektes kann ein dediziertes Improvement Backlog erstellt werden oder die Refactorings werden in das vorhandene Backlog integriert.

Um ein gesundes Gleichgewicht zwischen den Refactorings und der Weiterentwicklung zu erzielen und ihre Abhängigkeiten zu berücksichtigen, stimmen die Teammitglieder die entsprechenden Entwicklungs-Aktivitäten während der Planungs-Meetings ab. Das Team macht daraus ein Challenge, wer wie viel zum Refactoring und zur Weiterentwicklung beigetragen hat. Bei Review-Meetings wird reflektiert, wie viele Improvements und Stories die Teammitglieder geschafft haben.

Das Team definiert, wie der gute Beitrag zur Verbesserung und Weiterentwicklung belohnt werden soll. Neben individuellen Belohnungen ist auch die Team-Belohnung eine gute Wahl zur Stärkung des Team-Spirits.

Literatur

[1] D. Sokenou, B. Güldali: Architektur nicht(!) versenken - Software-Architektur spielerisch evaluieren und verbessern, Java Magazin Whitepaper Architektur, 2024

[2] S. Deterding, D. Dixon, R. Khaled, L. Nacke. From game design elements to gamefulness: defining "gamification". MindTrek '11: Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, September 2011.

[3] Dehla Sokenou, Baris Güldali. Gamification in der Qualitätssicherung – Teil 3: Schach dem Risiko! JavaMagazin, 7.2023, S. 92-97. https://entwickler.de/agile/gamification-risiko-qualitaetssicherung

[4] Henning Schwentner. Domain-Driven Refactorings. https://hschwentner.io/domain-driven-refactorings/