

Kognitions-orientierte Softwareentwicklung

Ein Erfahrungsbericht

Martin Rösch, mr@roesch.com

17. März 2014

Die Aufforderung zu diesem Kurzbeitrag beginnt mit dem Satz „Auch Software altert!“. Doch dieses Altern ist eine subjektive Wahrnehmung, die man auch anders – handlungsorientierter – sehen kann:

Wechsel der Perspektive

- Software altert doch nicht. Denn sie verändert sich nicht.
- Was sich in Wirklichkeit verändert, ist die Kognitive Distanz zwischen unseren Erwartungen / Anforderungen und dem, was die Software tatsächlich leistet.
- Denn unsere Erwartungen / Anforderungen ändern sich ständig (1-4% pro Monat).
- Und wenn wir unsere Software nicht schnell genug anpassen, dann eilen unsere Erwartungen / Anforderungen ihr immer weiter voraus.
- Bis wir denken: sie ist veraltet.

Kognitive Distanz → Null!

Der Begriff „Kognitive Distanz“ stammt ursprünglich aus der Lernforschung und bezeichnete die Abstände zwischen dem Wissen von Lehrenden und Lernenden. Doch wir können ihn auch auf die Software-Entwicklung anwenden. Dann kann er z.B. den Abstand bezeichnen zwischen dem heutigen Wissen um den Soll-Zustand einer Software und dem Wissen, das in die Software tatsächlich hineinprogrammiert worden ist.

Im Idealfall ist diese Kognitive Distanz zu jedem Zeitpunkt Null. Dann ist die Software immer top-aktuell. Und da meine Erfahrungen zeigen, dass wir dieses Ziel erreichen können, empfehle ich, es auch ganz ausdrücklich anzustreben. Nach meinen Erfahrungen müssen wir hierfür zwei Unter-Ziele erreichen:

1. Sehr wenig Fehler (Null-Fehler-Strategie). Denn jeder Fehler erhöht die kognitive Distanz.
2. Sehr kurze Zykluszeiten (Live-Entwicklung). Denn die Anforderungen ändern sich ständig (s.o.).

Null-Fehler-Strategie

Da die Informatiker von heute gelernt haben, dass Software immer Fehler haben müsse (Halteproblem), ist ihnen nur schwer nahezubringen, eine Null-Fehler-Strategie für die Software-Entwicklung zu akzeptieren. Ich will es dennoch versuchen. Weil es in der Praxis sehr gut funktioniert.

Live-Entwicklung

Die heute bekannten Zykluszeiten der Software-Entwicklung betragen regelmäßig mehrere Monate. Mit Agil, Continuous Integration, Continuous Delivery und DevOps gelingt es gelegentlich, einzelne Teile des Entwicklungsprozesses zu beschleunigen. Doch was wir für Kognitive Distanz → Null! brauchen, sind Zykluszeiten von Stunden, maximal Tagen. Und auch dies ist gar nicht so schwer wie es vielleicht scheint.

Wie eine Null-Fehler-Strategie funktioniert

Nehmen Sie einmal an, alle Anforderungen an eine Software wären bekannt, jede einzelne Anforderung wäre messbar, und sie könnten alle diese Messungen in Millisekunden durchführen. Dann hätten Sie zu jedem Zeitpunkt die Sicherheit, dass Ihre Software alles kann, was sie soll. Hierfür braucht man folgende ...

Wie Live-Entwicklung funktioniert

...

Leider kann ich diese Seite heute nicht vervollständigen. Wenn das Thema interessant erscheint, will ich dies aber gerne tun und ggf. auch auf dem Workshop Rede und Antwort stehen.

So viel vorneweg:

1. Die Erfahrungen stammen aus mehreren Projekten, darunter auch aus dem Risikomanagement einer Bank.
2. Nach meiner Meinung sollten Kognitionswissenschaft und Informatik näher zusammenrücken.
3. Die Industrie fordert diese Art von Ergebnissen mittlerweile:

<http://www.cio.de/strategien/2895781/>

Mit freundlichem Gruß

Martin Rösch

mr@roesch.com