Kurzvortrag der Pecha-Kucha-Session auf dem Java Forum Stuttgart 2017 am 06.07.2017

Philosophy screws it all up!

Wie Erkenntnistheorie unsere Softwareentwicklung beeinflusst und wie wir damit umgehen können.

Stichworte: Fundamentale Probleme, Softwareentwicklung, Erkenntnistheorie

Abstract:

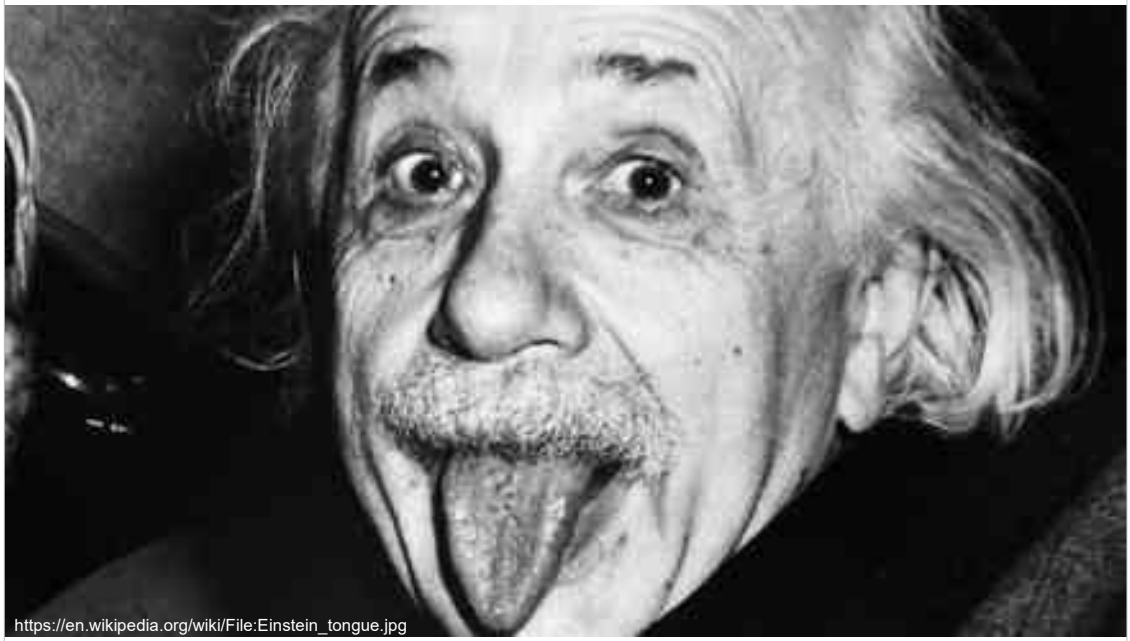
Softwareentwicklungsprojekte scheitern ab und an weder aus technischen, noch fachlichen – ja nicht einmal aus politischen Gründen. Sind anderweitige Gründe im Spiel, versenkt diese Art von Softwareprojektabbrüchen meist auch noch extrem viel Geld. Es ist unheimlich, dass wir diese Fälle nicht wirklich begründen können und daher vor allem nicht herausbekommen, woran es wohl wirklich gelegen hat. Es müssen somit noch andere, grundsätzliche Aspekte eine Rolle spielen, welche nicht sofort offensichtlich sind.

In diesem Pecha-Kucha-Vortrag ergründe ich mit Hilfe der philosophischen Disziplin der Erkenntnistheorie – der Theorie des Wissens – einige der nicht augenscheinlichen, jedoch immer vorhandenen Probleme der Softwareentwicklung. Die vorgestellten „erkenntnistheoretischen Dilemmata“ begegnen uns immer, wenn wir es mit Wissen zu tun haben. Softwareentwicklung ist die Implementierung von Wissen in Code. Daher sind auch wir – ob wir es wollen oder nicht – immer von diesen Dilemmata betroffen.

Es gibt jedoch Hoffnung: Sind diese Grundprobleme erst einmal bekannt, können wir sie nutzen, um unsere eigene Arbeit – unser Vorgehen, unserer eingesetzten Methoden und Praktiken in der Softwareentwicklung – nach objektiven Kriterien zu beurteilen und zu verbessern. Konkret zeige ich, wie sich die normative Wissenschaft der Erkenntnistheorie ganz praktisch als „Messwerkzeug“ verwenden lässt. Dazu stelle ich ihre Einflussnahme auf softwaretechnische Praktiken anhand von Praxisbeispielen vor. Meine Kernziele sind es, die Dilemmata kurz und prägnant darzustellen, handhabbar zu machen und praxisorientierte Lösungsansätze zum Umgang mit ihnen aufzuzeigen.

Über den Referenten:

Markus Harrer arbeitet seit über zehn Jahren in der Softwareentwicklung an Hochschule, Forschungsinstitut und Softwareunternehmen. Seine Spezialgebiete sind Clean Code, Softwaresanierung und Software Analytics. Derzeit entwickelt er Unternehmensanwendungen in Java und hilft durch die Analyse von Softwareartefakten, nachvollziehbar Verbesserungsmaßnahmen für Softwareprodukte zu erarbeiten. Privat erforscht er einige fundamentale Probleme der Softwareentwicklung und versucht, diese mit Hilfe von Erkenntnistheorie und Verhaltensökonomik zu veranschaulichen.



„Die Definition des Wahnsinns ist, immer dasselbe zu tun und ein anderes Ergebnis zu erwarten." – Albert Einstein

Willkommen in der Softwareentwicklung, wo wir seit über 50 Jahren denken, dass der Einsatz der neuesten Technologie all unsere alten Probleme lösen wird.

Und doch scheitern unsere Projekte spektakulärer als je zuvor.

Wir denken dann oft es liegt an der Technik oder am Vorgehen, dass wir einsetzen.

Aber es ist viel schlimmer:

Die Probleme, mit denen wir kämpfen, haben auch **philosophische Qualität**.

D. h. diese Probleme können nicht gelöst werden – sie sind Dilemmas.

Eine philosophische Disziplin, die **uns** besonders betrifft, ist...

… die Erkenntnistheorie – Die Theorie des Wissens.

Sie beschäftigt sich vor allem mit zwei Fragen:

„Was ist Wissen?“ und, *viel wichtiger*,
„Gibt es überhaupt so etwas wie Wissen?“

Auch die Erkenntnistheorie zeigt uns keine Lösungen, sondern nur wieder verschiedene Dilemmas, welche im Umgang mit Wissen entstehen.

Und gerade wir Softwareentwickler sind besonders von diesen erkenntnistheoretischen Dilemmas betroffen – wir gießen **Wissen** in Code.

Wir können aber die Dilemmas nutzen, um unser eigenes Vorgehen – unsere Handlungen und Arbeitsweisen zu reflektieren und stetig zu verbessern.

Was sind diese Dilemmas?

Ein Beispiel: Nehmen wir einmal an, wir gehen in ein Unternehmen und wollen dort einen realen Geschäftsprozess in ein Modell abbilden.

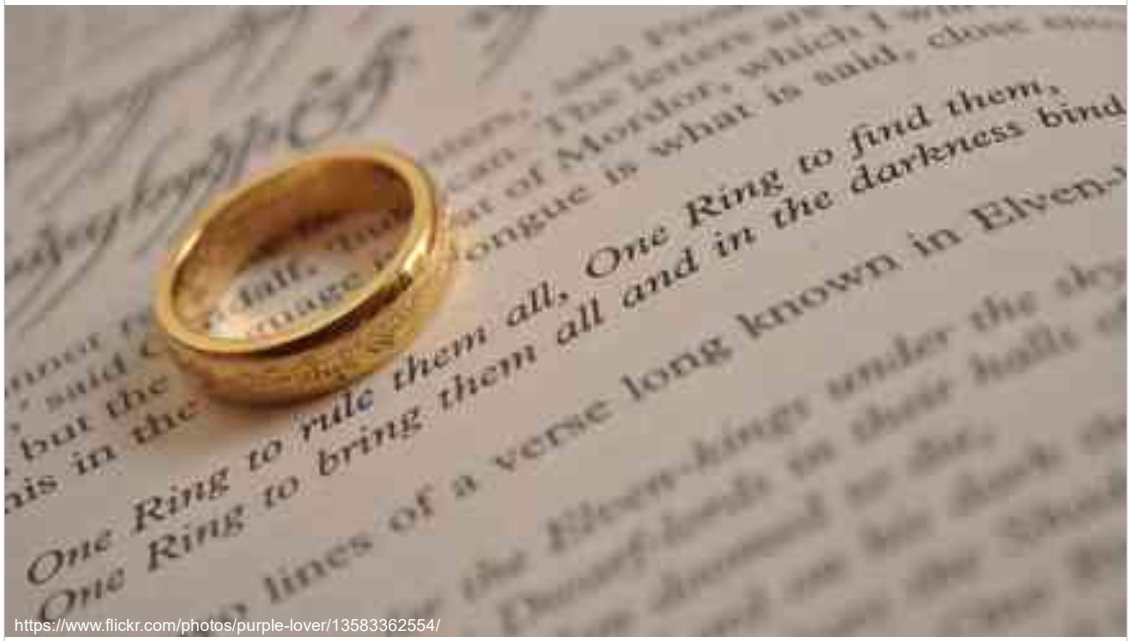
- Wo fangen wir an?
- Wo startet der Prozess?
- Wo hört er auf?
- Was gehört dazu?
- Was lassen wir weg?
- Wie nehmen wir den Geschäftsprozess auf?
- Ist das Mittel zur Aufnahme das Richtige?

Die Dilemmas der Abbildung stellen genau diese Fragen...

...und sagen uns, dass wir immer nur temporäre Beobachtungen vornehmen – gefiltert durch unsere Wahrnehmungsfähigkeiten – und dass nur für einen mehr oder weniger zufälligen Ausschnitt.

Sie sagen auch, dass eine 1:1 Beziehung zwischen der Realität (was das auch immer ist) und Modell nicht existieren kann.

Wir sehen nie alles.



Aber dennoch tun wir oft so, als ob dem nicht so sei.

Wir versuchen z. B. immer noch
unternehmensweite Datenmodelle zu schaffen:

„Ein Modell, um sie zu knechten, sie heimzusuchen, ins Dunkel zu treiben und ewig zu binden“ – und sei es als Maven Dependency.

Wir beachten hier aber nicht die
Perspektivengebundenheit des Wissens:


https://meditacionesdeldia.com/2014/09/25/vemos-lo-que-vemos-no-lo-que-es/

Wissen gilt immer nur in einen bestimmten Kontext – und dieser Kontext verändert sich relativ zu Zeit, Ort, Kultur und Personen.

Wenn wir also ein Modell erstellen, kann dies nur aus einer bestimmten Perspektive heraus passieren – in einem bestimmten Kontext – nie global für alles Mögliche.

Aber wie kommen wir überhaupt zu Wissen?

Wir fragen z. B. einen Mitarbeiter, der sich mit einem bestimmten Thema auskennt.

Was passiert, wenn wir ihn befragen?

Wir werden mit den Dilemmas der gegenseitigen Beeinflussung konfrontiert.

Wir erhalten seine Antworten oder sein Wissen „unter Beobachtung“.

Umgekehrt werden wir durch den Mitarbeiter beobachtet und selbst beeinflusst.

Wir interagieren also direkt mit dem Gegenstand, dem Subjekt, das wir eigentlich nur objektiv beobachten wollten.

Unser Wissen über die Begriffe und Abläufe passt sich somit gegenseitig an.

Das Ganze können wir noch ein wenig weitertreiben...

...wenn wir in Gruppen interagieren.

Hier unterliegen wir sehr stark dem Dilemma der Kohärenzbildung:

Wir bilden uns unsere eigene Welt aus dem für uns richtigem Wissen. Das heißt aber gleichzeitig, wir weisen fundamental anderes Wissen ab.

Aber wenn genau hier die eigentlichen Wünsche des Kunden stecken, ...

...ist das eine tickende Zeitbombe, die früher oder später losgeht und Projekte katastrophal scheitern lässt – und niemand hat es kommen sehen!

Verdammt viele unlösbare Probleme. Aber was können wir jetzt machen?

Zuerst einmal hilft es anzuerkennen, dass die Geschäftswelt da draußen dreckig und chaotisch ist – wir haben keine Chance sie 1:1 zu modellieren

Und: Es gibt nichts absolut Richtiges oder Falsches – auch bei Entscheidungen!

Das zu akzeptieren, entspannt die ganze Sache erst einmal.

Wir können aber auch die Erkenntnistheorie direkt und ganz konkret in unsere Arbeit mit aufnehmen:

Wir können z. B. akzeptieren, dass es unterschiedliche Sichten auf die Dinge im Unternehmen gibt und diese Vielfältigkeit in unsere Software unterstützen.

Kein ultimatives Framework oder globales Datenmodell mehr, sondern getrennte, spezialisierte Bereiche, in denen Wissen im jeweiligen Kontext aufleben kann.

Ja, das ist verdammt viel Arbeit!

Um das zu schaffen, müssen wir vereinfachen und alles abschneiden, was wir nicht unbedingt brauchen.

Je weniger wir machen, desto weniger können wir falsch liegen.

Nicht alles mögliche selbst implementieren, sondern nur das, was das Kerngeschäft eines Unternehmens wirklich unterstützt.

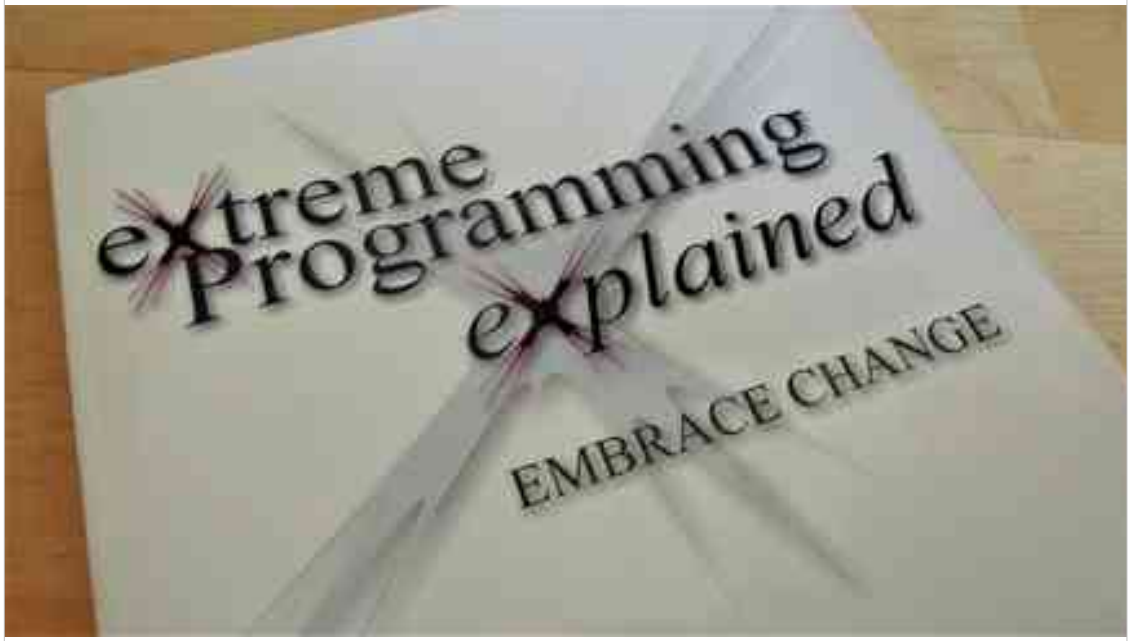
Das erfordert auch Mut zur Kommunikation und Transparenz:

Wir müssen zugeben, wenn für uns etwas nicht klar genug ist. Aber wir müssen ansprechen, wenn wir denken, dass der Kunde sich selbst nicht im Klaren über seine Wünsche oder sein Geschäft ist.

Wir können auch unsere eigenen Annahmen oder Hypothesen mittels Zahlen, Daten Fakten validieren oder widerlegen – und damit unser Bauchgefühl – das was wir glauben zu Wissen – explizit machen.

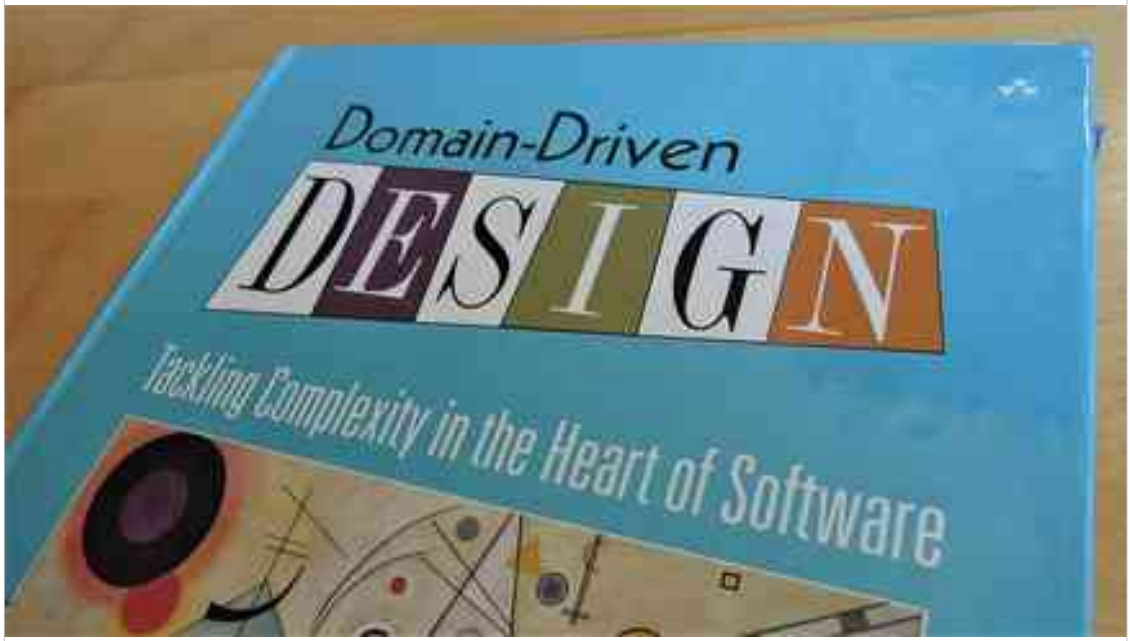
Wir müssen Software schrittweise, induktiv an das Wissen annähern – und sind uns dabei aber immer bewusst, dass es eine ultimative Softwarelösung nicht geben kann.

Und das alles immer mit dem Blick auf die erkenntnistheoretische Dilemmas, um unser eigenes Vorgehen – unsere Handlungen und Arbeitsweisen – zu reflektieren und stetig zu verbessern.



Und wir haben ja sogar schon ein paar Praktiken gefunden, welche sich bisher noch nicht als komplett falsch herausgestellt haben.

Z. B. Extreme Programming, welches sich mittels Transparenz und einem iterativen Vorgehen schrittweise an akzeptabel Lösungen annähert – die enge Zusammenarbeit mit den Wissensträgern aktiv forciert und auf Einfachheit abzielt



Und Domain Driven Design,
welches explizit versucht, verschiedene Begriffswelten durch eine gemeinsame Ubiquitous Language zu vereinen und kontextabhängige Modelle in Bounded Contexts als ein strategisches Mittel zur Softwaremodellierung vorschlägt.

Ich hoffe, dass wir Softwareentwickler zukünftig weiter solche Praktiken schaffen, die die Dilemmas der Erkenntnistheorie bereits Out-of-the-Box beachten.

Ich denke, dass wir dann zwar augenscheinlich immer dasselbe tun, aber doch stetig bessere Ergebnisse erhalten.

Vielen Dank!