Fragen zu «No Silver Bullet» von Elias Büchel

Was meint der Autor mit der Aussage «there is no Siver Bullet»?

Eine «Silver Bullet» bezieht sich auf einen Ansatz, eine essenzielle Komplexität und/oder Aufwand zu verringern um schlussendlich massive Entwicklungszeit und Kosten sparen zu können. Dies ist jedoch laut der aussage «no silver bullet» nicht möglich. Mit «silver bullet» ist damit die Lösung zum problem gemeint.

Beschreiben sie kurz den Unterschied zwischen «Accidental complexity» und «Esential complexity» anhand eines konkreten Beispiels.

«Esential complexity» bezieht sich auf die Komplexität eines Problems, welche nicht vermeidbar ist.

«Accidental complexity» bezieht sich auf die Komplexität, welche beim lösen eines Problems entsteht. Es sind Schwierigkeiten die man sich beim Entwickeln einbaut.(nicht der optimale Weg)

Man entwickelt somit ein Programm, dass gewissen Anforderungen gerecht werden soll. Die optimale Implementierung stellt die essenzielle Komplexität dar. Die über die optimale hinaus entstandene Komplexität beschreibt man als «accidental compplexity».

Was ist für den Autor die Schwierigkeit beim Entwickeln von Software? Was ist relativ einfach?

Die essenzielle Komplexität einer Problemstellung lässt sich nicht verringern/vermeiden. Ausserdem erhöht sich die Komplexität exponentiell zur Grösse des Programms. Dies wird unteranderem noch durch das arbeiten in Teams verstärkt, da Unstimmigkeiten/Unwissen auftreten.

Das einfache ist, dass heutzutage für beinahe jedes Problem, bereits ein vergleichbarer Lösungsansatz im Internet existiert.

Weshalb ist Software Entwicklung von der Natur her schwieriger als Ingenieursarbeiten die auf physikalischen Prinzipien beruhen?

Strukturen/Vorgehensweisen lassen sich, wenn man es mit es mit physikalischen Prinzipien vergleicht, schwer Vorstellen. Bei Physikalischem kann man meist Verknüpfungen zu existierenden Objekten machen.

Weshalb muss sich Software ständig verändern?

Die Hardware hat sich in der Vergangenheit rapide verändert. Somit stiegen auch die Anforderungen in der Software, da Prozesse aus der Software immer schneller auf der Hardware laufen. Dies lässt komplexere zu, was von dem Markt auch gefordert wird.

Weshalb konnte in der Vergangenheit trotz der Komplexität der Software die Entwicklungsproduktivität gesteigert werden?

Sowohl die starken Hardwareeinschränkungen, spezielle Programmiersprachen, wie auch die fehlende zeit um an den Maschinen zu arbeiten, machte es den Entwicklern schwierig, die Produktivität zu

steigern. All dass hat sich jedoch über die Zeit stark verändert. Jeder kann heutzutage von zuhause Software erstellen und diese testen.

Was sind für Fred Brooks vielversprechende Möglichkeiten um die Produktivität in der Softwareentwicklung zu steigern?

Expert Systems:

Durch das einheitliche definieren von Regeln für ein System, legt man diese gleichzeitig für Schnittstellen, Tests und Überprüfung von Anpassungen fest. So kann man Entwicklungszeit und Kosten einsparen.

Incremental Development:

Das wachsen des Projekt, erlaubt einfache Rückverfolgung und führt zu frühzeitigem Prototyping. Die Struktur des Programms bildet sich frühzeitig ab, was Anpassungen der Struktur für bewältigbar macht. Ausserdem kann man die Komplexität für jede Erweiterung gering halten, obwohl die Komplexität des gesamten Programms steigt.

Great Designers:

Hervoragende Designer können im Verglich zu «normalen» sowohl hochwertigere Designs als auch diese mit weniger Zeitaufwand bewältigen. Da dies die Basis für die weiteren Entwicklungsschritte ist, kann es einen enormen Unterschied ausmachen. Um mit diesem potenziellen Punkt weiter zu kommen, muss man bei der Bildung anfangen wie auch und die Leute gezielt schulen.

Warum ist “Rapid Prototyping ” so wichtig?

Nichts ist so schwierig, wie die detaillierten technische Anforderungen festzulegen. Dies beinhaltet unteranderem die Schnittstellen zu Menschen, Maschinen und anderen Softwaresystemen. Meistens wissen die Kunden nicht genau was sie wollen. Somit kann man mit einem «schnellen Prototyp» erstellen, welcher die Schnittstelen simuliert und grundlegende Funktionalitäten implementiert. Dieser Prototyp soll dem Kunden dienen, die Konsistenz und Benutzerfreundlichkeit zu testen.