Probleme der Softwareentwicklung in der Wissenschaft

Präsentation zum Seminar:

Softwareentwicklung in der Wissenschaft

Renko Buhr
01.07.2013

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Zugrunde liegende Paper

- Segal & Morris, 2008: Devellopingg Scientific Software, IEEE Software pp. 18-20
- Segal, 2008: Models of scientific sortware development. IN: SECSE08, International Workshop on Software Engineering in Computational Science and Engineering, Leipzig, Germany (2008), pp. 1-5.

- McManua & Wood-Harper, 2007: Software engineering: a quality management perspective. The TQM Magazine Vol 19, No. 4, 2007, pp. 315-327.

Gliederung

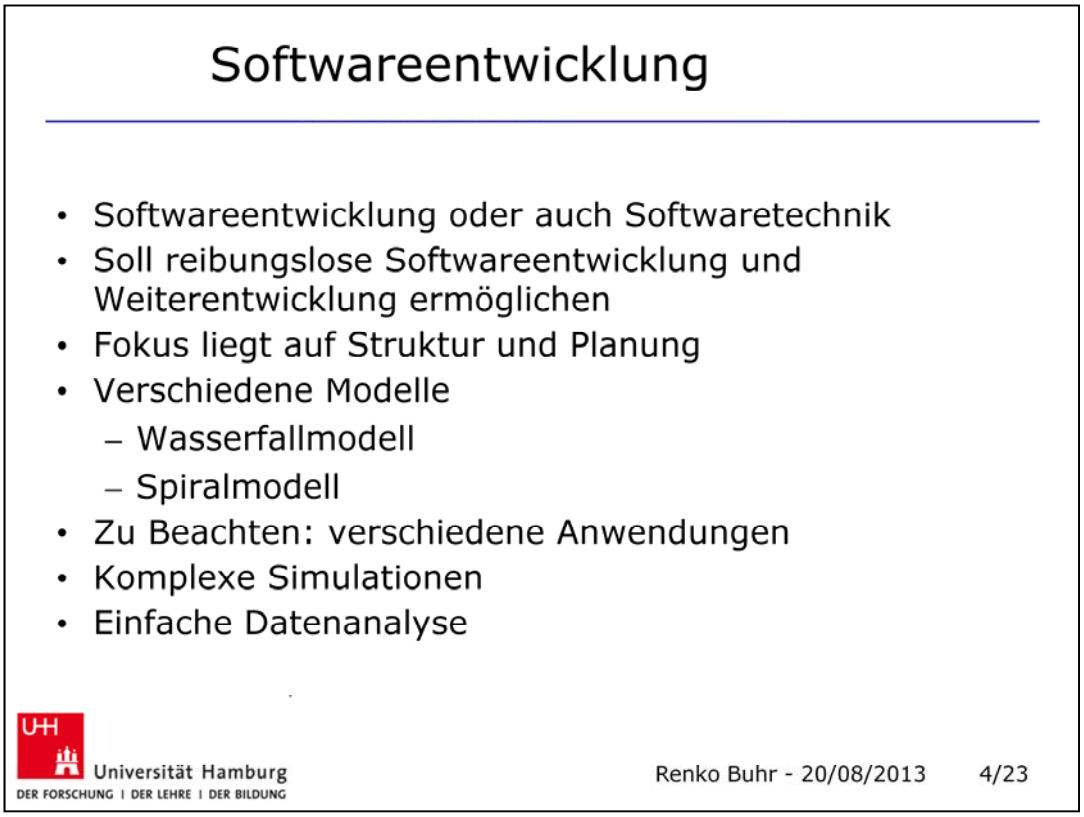
- Was ist Softwareentwicklung ?
- Software in der Wissenschaft
- Vorgehen Wissenschaftler
- Zusammenarbeit Wissenschaftler- Programmierer
- Qualitätsmanagement
- Zusammenfassung
- Fazit

Was ist Softwareentwicklung ?

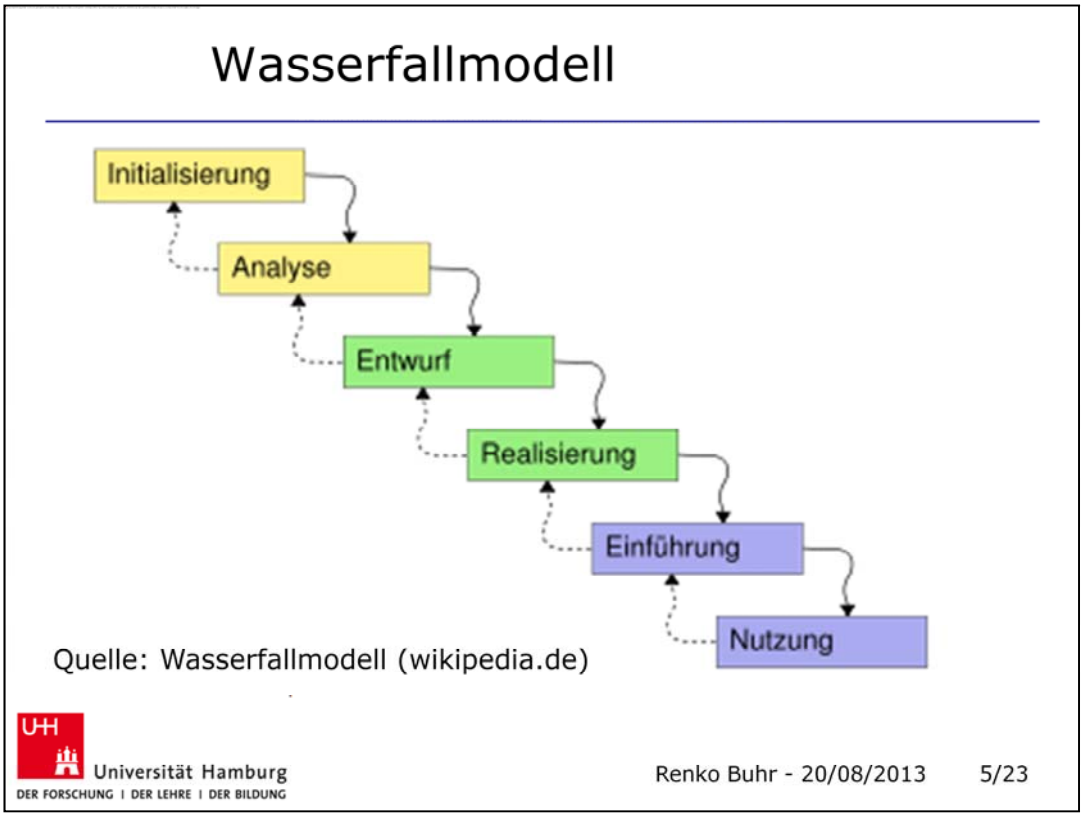
„Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen."

Balzert, S.36

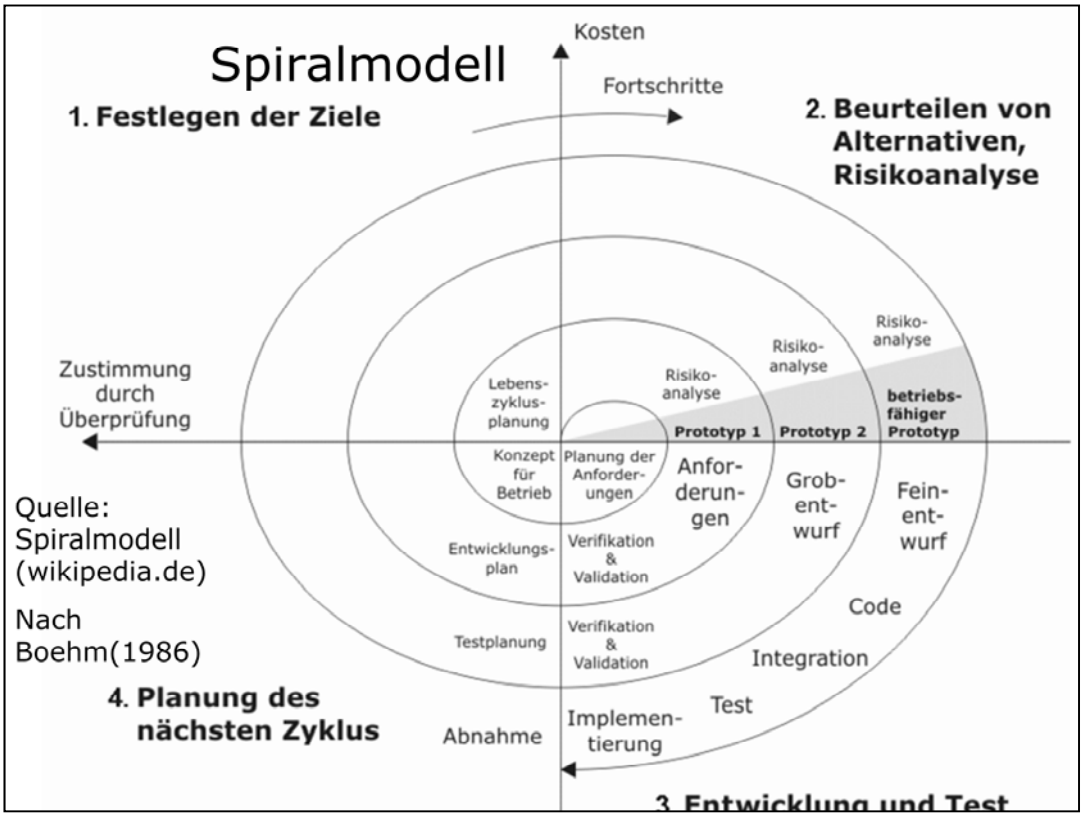
Quelle: Softwaretechnik (wikipedia.de)



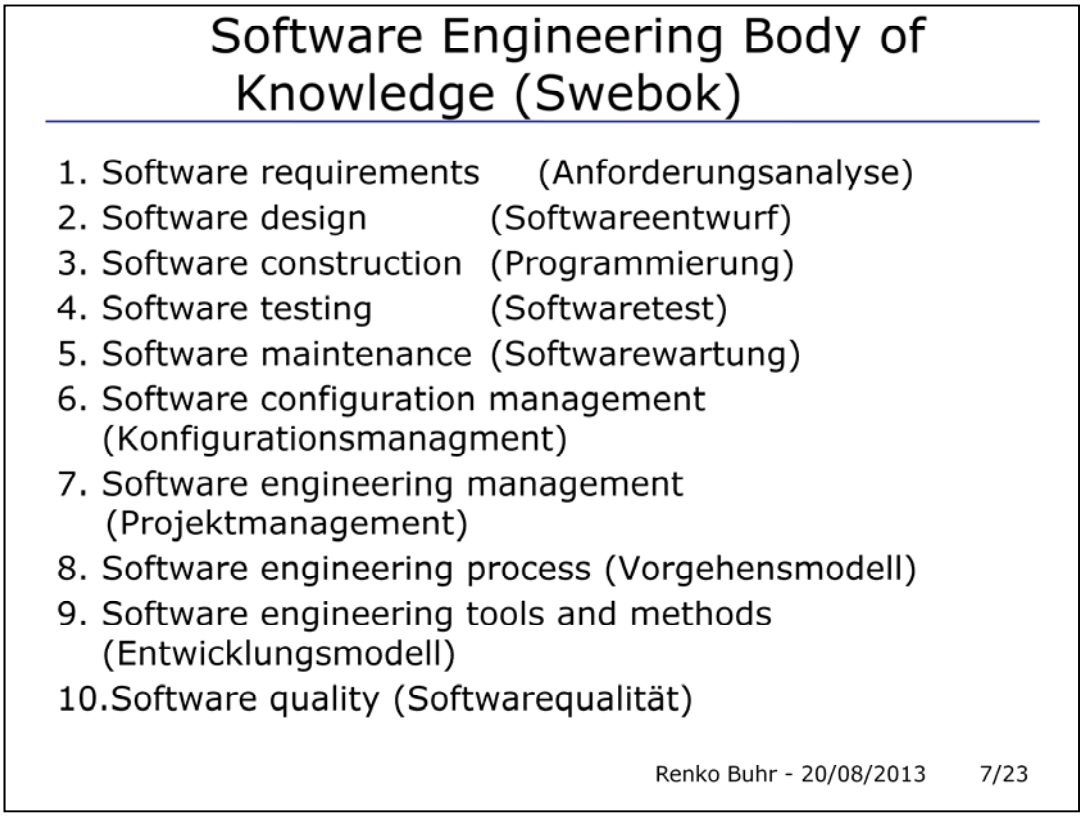
Softwareentwicklung oder Softwaretechnik beschreibt die systematische Anwendung verschiedener Vorgehensweisen und Methoden um einen reibungslosen Ablauf zu gewährleisten. Dabei liegt der Fokus weniger auf das schnelle Erreichen des Zielproduktes, sondern auf einer guten Struktur und Planung. Dabei gibt es verschiedene Modelle, Je nach Anwendung können verschiedene Modelle zum Einsatz kommen.



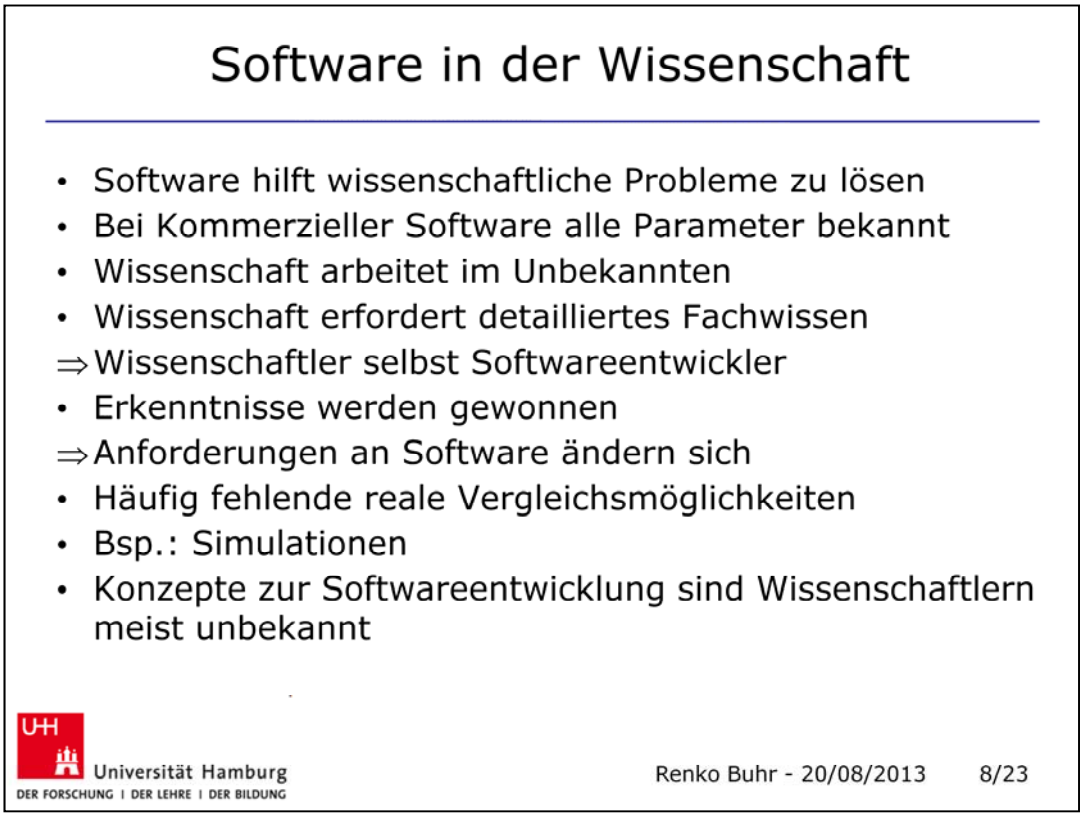
Wasserfallmodelle beschreiben die iterative Entwicklung einer Software, dabei wird immer ein Schritt nach dem anderen durchgeführt und erst wenn das Ergebnis zufrieden stellend ist zur nächsten Stufe übergegangen. Die einzelnen Stufen werden auch als Kaskaden bezeichnet. Die Rückschritte bilden eine Erweiterung dar, die es ermöglicht Fehler, welche erst in einer späteren Kaskade auftreten, zu beheben. Das Modell hat Vorteile durch die klaren Abgrenzungen der Kaskaden und die dadurch vereinfachte Kontrolle. Sie eignet sich besonders gut bei stabilen Anforderungen und klaren zeitlichen und finanziellen Abschätzungen. Darin liegt aber auch das größte Problem, wenn ein Problem anders gelagert ist. Dies ist in der Wissenschaft meist der Fall. Außerdem ist eine klare Abgrenzung auch bei einfachen Problemstellungen häufig nicht möglich.



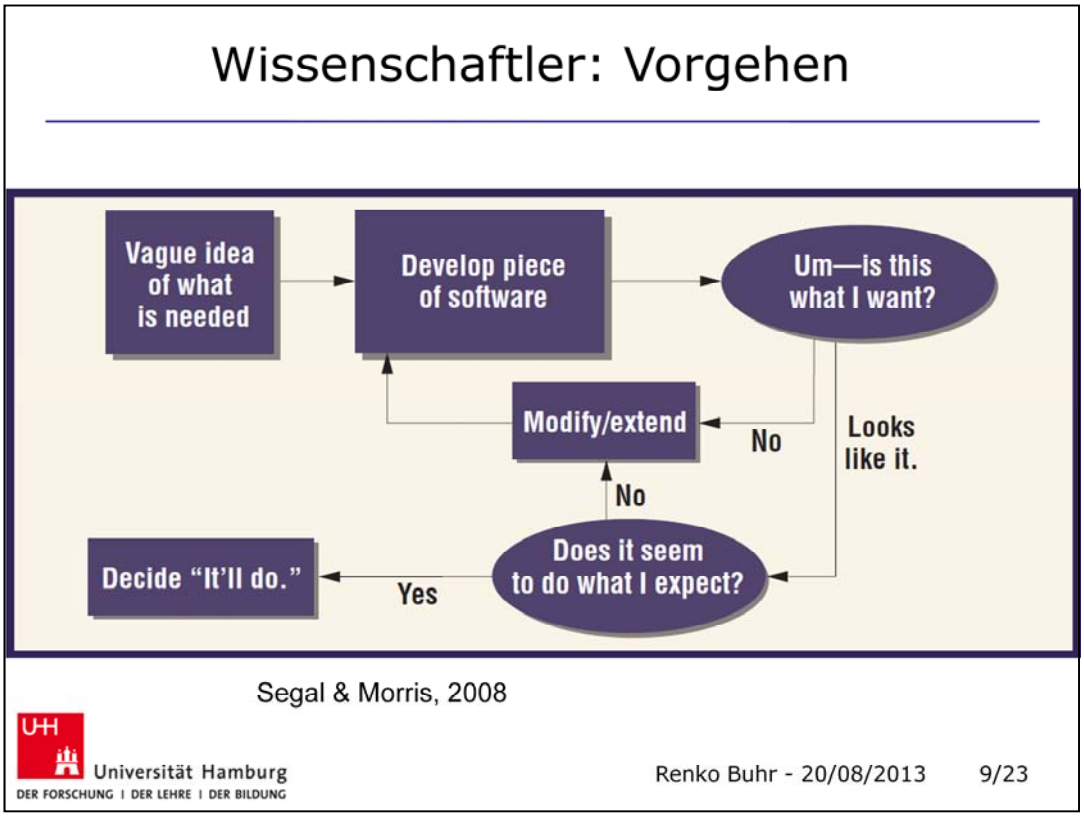
Im Gegensatz zum Wasserfallmodell, geht das Spiralmodell immer 4 wichtige Schritte durch. (Festlegung von Zielen, Analyse, Entwicklung/Test, Planung des nächsten Schrittes. Zentral ist dabei die Risikoabschätzung. Das Ziel ist die größten Risiken sukzessive abzubauen um eine Programm zu entwickeln. Davon hängt der Erfolg und nicht Erfolg des Projektes ab. Die 4 Stufen werden so oft wiederholt bis das Projekt beendet ist, ob nun erfolgreich oder nicht.



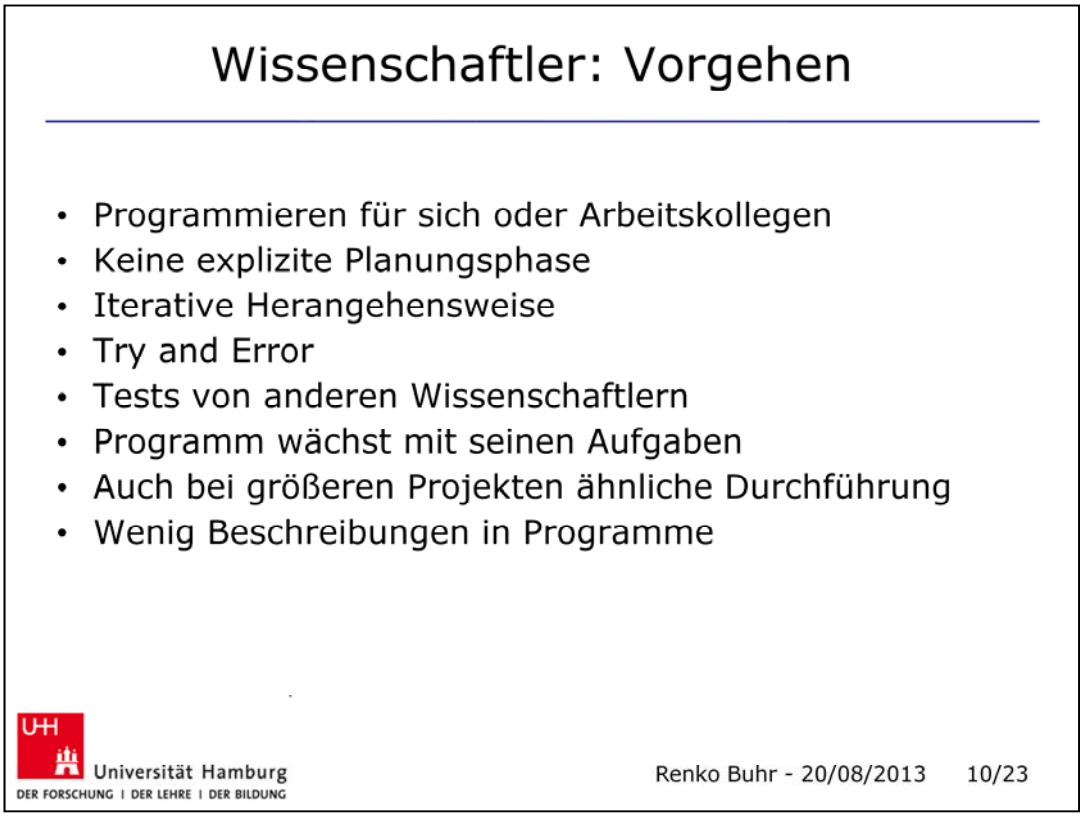
Der „Guide to Swebok Software Engineering Body of Knowledge“ (Swebok) ist ein Produkt der IEEE Computer Society (IEEE: **Institute of Electrical and Electronics Engineers**). Der Guide beschreibt welche verschiedenen Dinge bei der Entwicklung einer Software beachtet werden sollten, um eine möglichst reibungslose Entwicklung zu gewährleisten. Dabei stehen die Anforderungen an das Programm an erster Stelle. Die ersten 5 Punkte fallen auch in ein chronologisches Muster (vgl. Wasserfallmodell). Weiterhin folgen organisatorische Angelegenheiten, welche essenziell für eine erfolgreiche Entwicklung sinnvoll sind. Es bildet (eingeschränkt) den Standart in der Softwaretechnik.



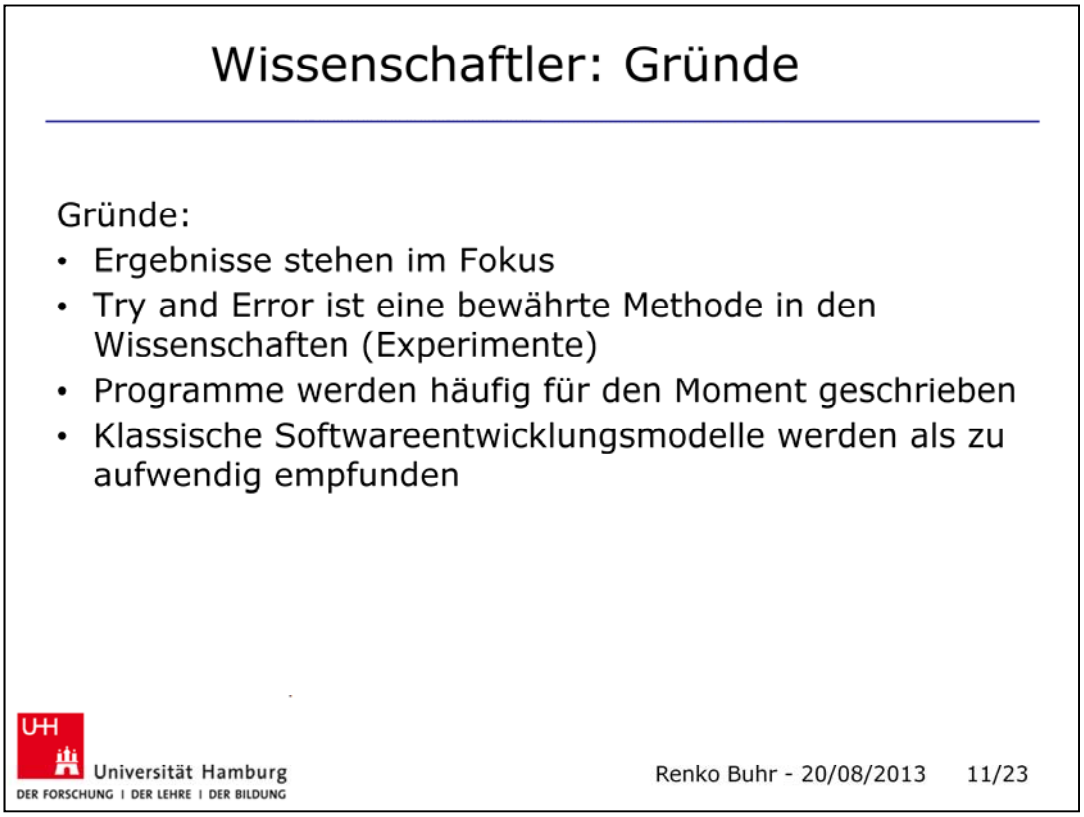
Während bei kommerzieller Software die Zielsetzung einer Software von Beginn an definiert ist und der Softwareentwickler alle Ausgangsparameter kennt, steht der Wissenschaftler steht immer vor einem wissenschaftlichen Problem welches er Lösen möchte. Bestimmte Rahmenbedingungen des Problems sind bekannt, aber viele andere wiederum nicht. Um herauszufinden was relevant ist und was nicht, benötigt der Wissenschaftler viel Fachwissen. Durch seine Ausbildung und Erfahrung in der Wissenschaft kann er viele Fragen durch Intuition beantworten. Dies kann ein Softwareentwickler nicht. Daher ist es notwendig, dass der Wissenschaftler selbst Software enwickeln. Außerdem ändern sich wissenschaftliche Fragestellen im Laufe der Zeit. So können beispielsweise Erkenntnisse den Ausgangspunkt verändern oder das zu behandelnde Problem erweitert werden. Dadurch ändern sich die Anforderungen an die Software. In der Wissenschaft fehlen häufig solide Vergleichsmöglichkeiten für die Ergebnisse entwickelter Software. Da Dinge erforscht werden, allerdings kaum gemessene Vergleichsmöglichkeiten vorhanden sind. Ein Vergleich von Programmcodes ist ebenfalls schwierig, da Wissenschaftlern selbst Lehrbeispiele aus der Softwareentwicklung gänzlich unbekannt sind.



Diese Abbildung zeigt die intuitive Herangehensweise des Wissenschaftlers mit einem besonderen Fokus auf den Beginn eines Projektes. Während des Softwareentwickler vor Beginn eines Projektes Zielsetzung etc festlegen, fängt der Wissenschaftler einfach an und entscheidet dann, ob eine weitere Entwicklung sinnvoll ist..

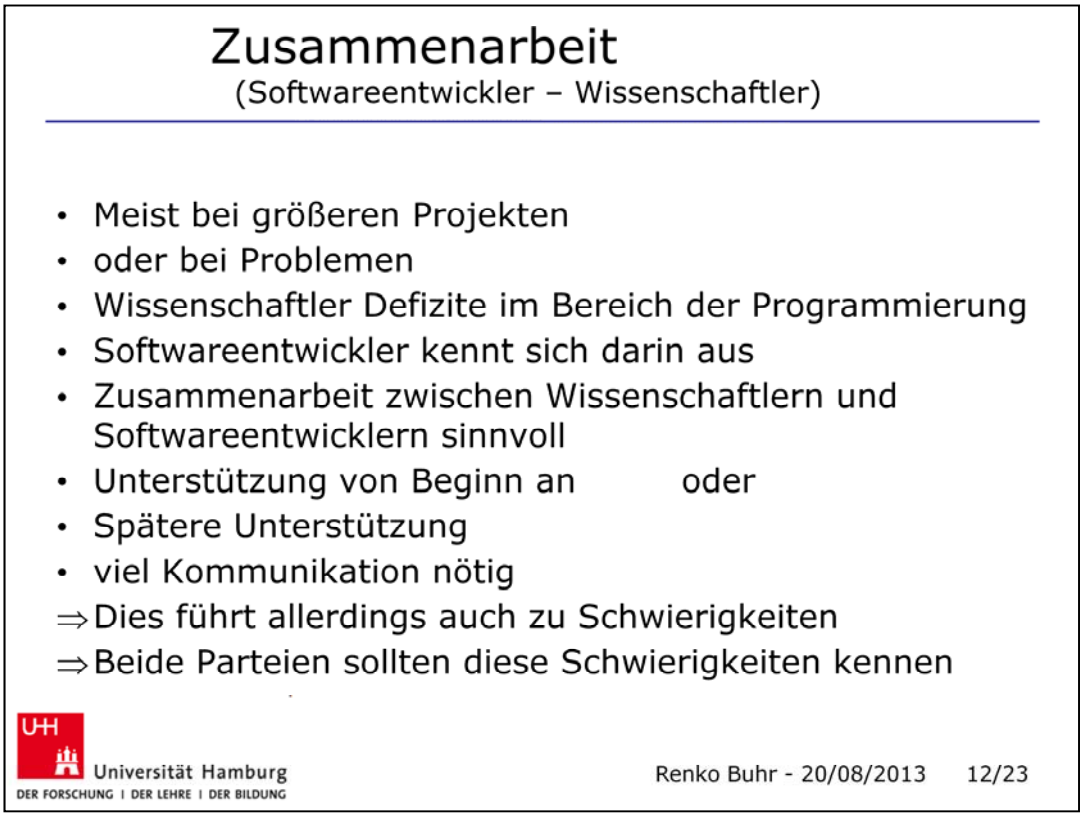


Bei größeren Projekten, werden meist nur die eigenen Erfahrungen genutzt, sowie der Austausch mit anderen Forschern bei Problemen gesucht. Wenn ein Programm ein gewisses Level erreicht hat wird es anderen Wissenschaftlern mit der Aussicht auf Rückmeldung gegeben. Diese „Bug-Report“ enthalten aber auch Erweiterungsvorschläge, welche dann wiederum implementiert werden können. Auf Softwareentwickler wird nur selten zurückgegriffen, da diese zunächst eine Einarbeitungszeit in den wissenschaftlichen Hintergrun benötigen würden.

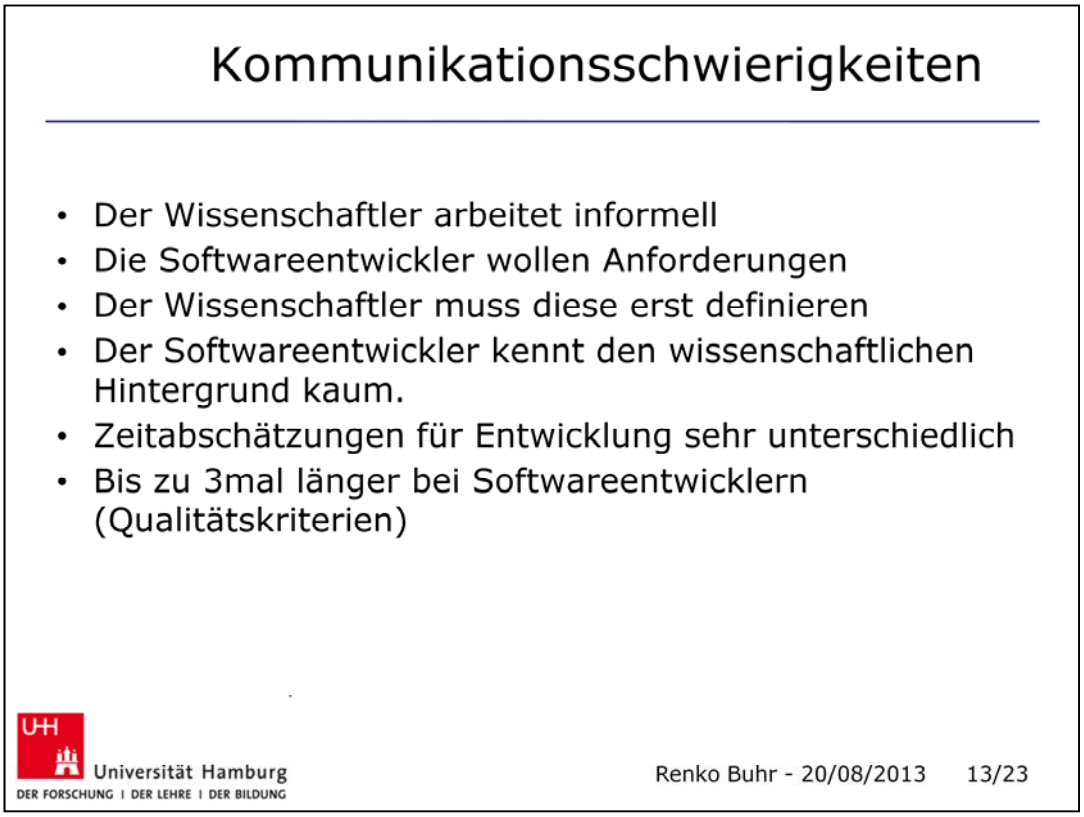


Der Wissenschaftler geht erst einmal von funktionsfähigen Programmen aus. Lässt sie dann laufen und wenn das Ergebnis anders ist als erwartet ist, wird eine Fehler gesucht.

Klassische Softwareentwicklungsmodelle gehen meist von Software aus, welche zunächst entwickelt und später auch wieder verwendet und gewartet werden soll. Dies ist in der Wissenschaft haben häufig nicht gegeben, der größte Anteil der Programme wird lediglich für konkrete Anwendungen geschrieben und meist nur in einem engen Zeitraum intensiv genutzt. Die Lebenszeit ist gering. Es gibt allerdings auch gegenteilige Software, welche in der Wissenschaft eine sehr lange Lebensdauer hat. Beispielsweise hat sich die grundlegende Struktur der Klimamodelle nicht verändert.



Die Zusammenarbeit ist nahe liegend, da auf beiden Seiten Defizite herrschen. Ein Wissenschaftler kann kleinere Aufgabe meist problemlos alleine oder mit anderen wissenschaftlichen Kollegen lösen. Ihm hilft hier die wissenschaftliche Intuition. Ein Softwareentwickler könnte dies nicht. Allerdings sind seine Kenntnisse bei größeren Projekten von enormer Wichtigkeit, da so doppelt effizient gearbeitet werden kann. Zum einen kann er technische Fehler im Programm feststellen, die zu einer unnötig langen Rechenzeit führen, zum Anderen kann der Entwicklungsprozess selbst effizienter gestaltet werden.

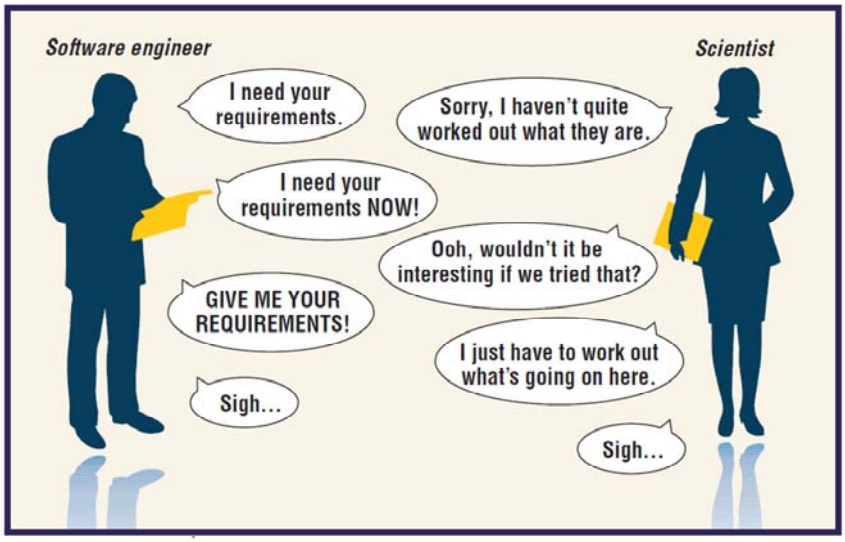


In diesem Zusammenhang bezieht sich das informelle Arbeiten auf das nicht korrekte Ausfüllen bestimmter Unterlagen zur Beschreibung von Programmteilen o.ä. Der Wissenschaftler ist hier eher die direkte Kommunikation gewohnt und ist nicht präzise genug.

Der Softwareentwickler kennt den wissenschaftlichen Hintergrund nicht, im Gegensatz zu einem wissenschaftlichen Programmierer. Spricht man mit letzterem, kann man das Problem lediglich kurz fassen und er weiß was zu tun ist. Er kann intuitiv fehlende Voraussetzungen hinzufügen. Dies kann man von einem Softwareentwickler nicht erwarten.

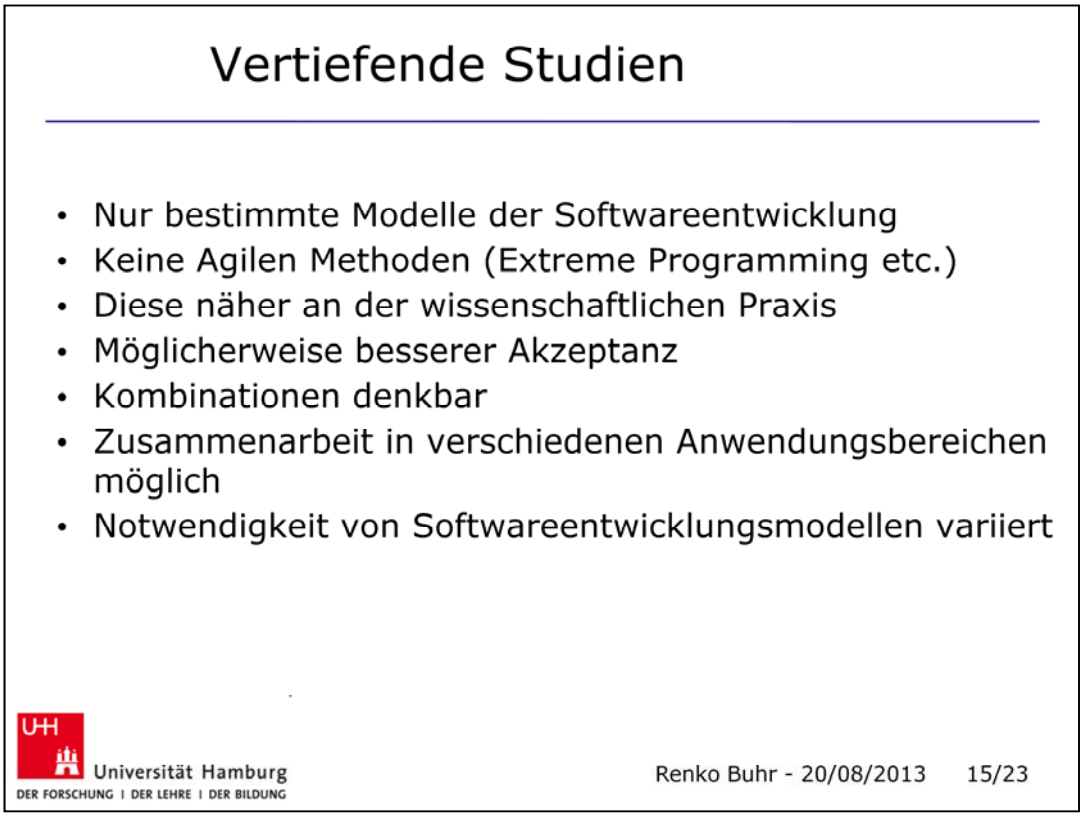
Zeitabschätzungen: Wissenschaftliche Programmierer müssen sich normalerweise keine Gedanken über Tests, Systemsicherheit oder Parallelsoftware machen. Der Softwareentwickler plant dies aber ein, daher sind seine Abschätzungen für den Projektzeitraum deutlich größer.

Kommunikationsschwierigkeiten

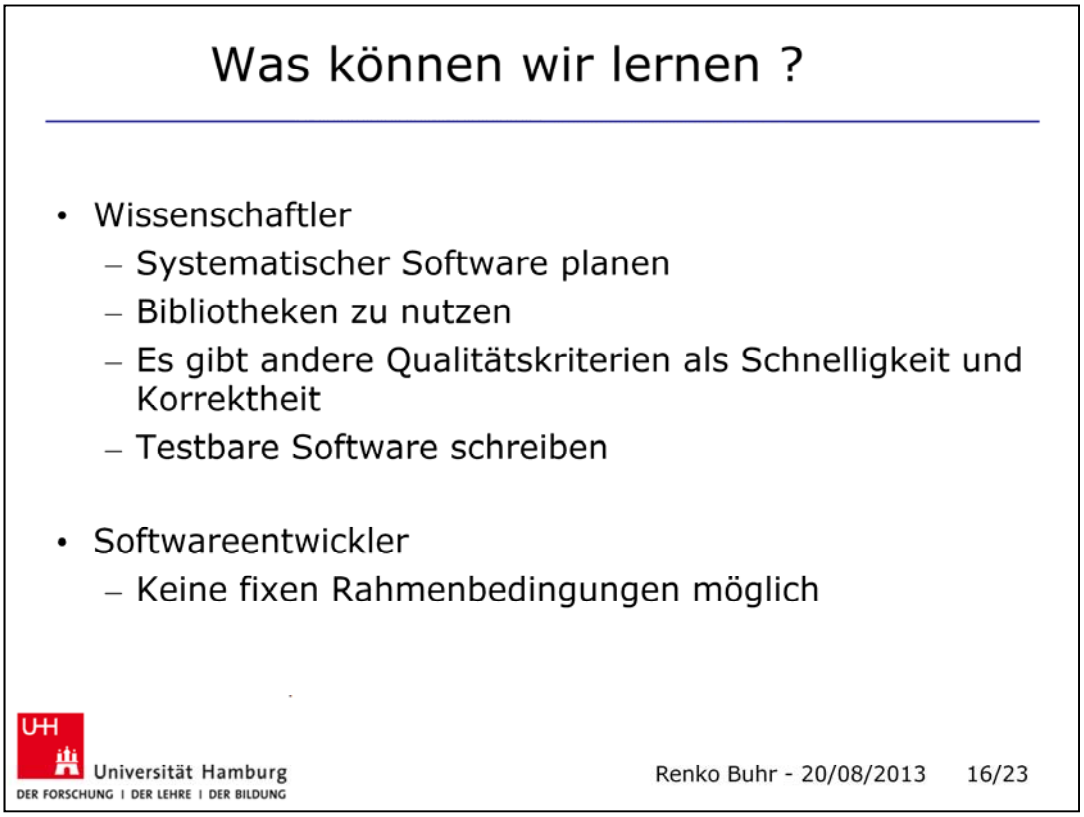


Segal & Morris, 2008

Diese Abbildung ist selbsterklärend. Während der Softwareentwicklung die Anforderungen für das Programm benötigt, kann die Wissenschaftlerin diese nicht liefern. Wenn sich beide Personen keine Kenntnis über das andere Feld haben, können hier nur Missverständnisse entstehen.



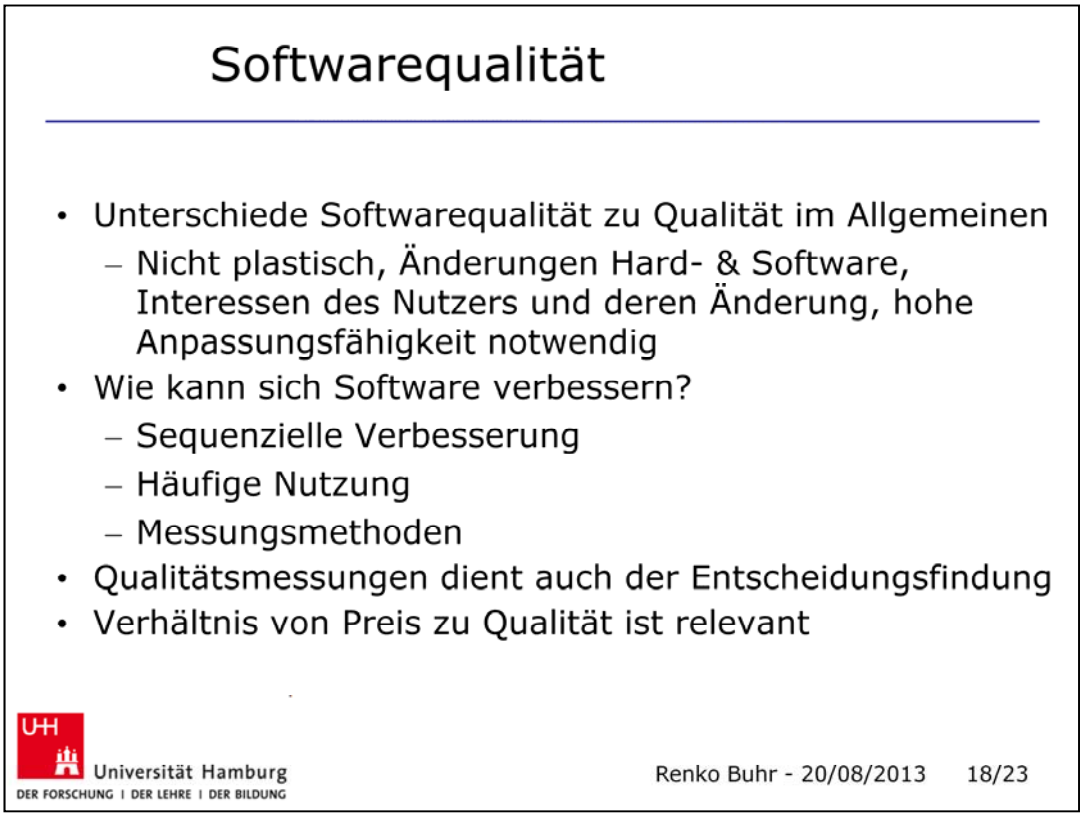
In dieser Studie berücksichtigte Entwicklungsmodelle vergleichen die iterative Herangehensweise der Wissenschaftler mit den klassischen Methoden der Softwareentwickler. Neuere, teilweise umstrittene Methoden, wie etwa Extreme Programming würden nicht berücksichtigt. Allerdings gibt es hier Studien, welche den Schluss nahe legen, dass diese Methoden der Herangehensweise der Wissenschaftler näher kommen und somit von diesen eher angenommen werden würden. Extreme Programming stellt ähnlich wie der Wissenschaftler das Ergebnis in den Vordergrund. Extreme Programming scheint im Bereich der Anforderungen nützlich zu sein, während es beim Software-Unterbau sinnvoll ist klassische Methoden zu verwenden.



Der Wissenschaftler muss vor allem bereit sein, Erkenntnisse und Ergebnisse anderer zu nutzen. Seien es Entwicklungsmodelle, welche sich über Jahrzehnte in der Softwareentwicklung etabliert haben, oder Softwareprodukte, wie Bibliotheken, stärker zu nutzen. Letzteres zum Einen um eigene Arbeit zu sparen, aber vor allem um auch andere Programmierstile kennen zu lernen und somit auch Probleme die hier auftreten zu erkennen und selber zu vermeiden. Wichtig ist auch, dass es ein breites Spektrum an Qualitätskriterien für Software gibt, welche über die mit direkter wissenschaftlicher Relevanz hinausgehen. Nicht zuletzt sollte und kann der Wissenschaftler lernen, seine Software so zu schreiben, das sie testbar ist, dies macht sie meist auch verständlicher für Außenstehende.

Der Softwareentwickler muss erkennen, dass Standardmodelle in der Wissenschaft unter Umständen nicht anwendbar sind und eine individuelle Vorgehensweise entwickelt werden muss.

Qualitätsmanagment im Allgemeinen beurteilt den Nutzen eines Produktes bzw. einer Dienstleistung in all seinen Facetten.



Die Softwarequalität unterscheidet sich von der Produktqualität, dadurch, dass eine Software ein digitales Produkt ist und somit keine physische Existenz hat. Außerdem ändern sich verschiedene Bedingungen für Software schnell. Hier ist zum Einen die Soft und Hardware gemeint, zum Anderen die Interessen des Nutzer. Hinzu kommt dass sich diese Interessen häufig und auch schnell ändern können. Zuletzt soll aus Nutzersicht die Software möglichst flexibel funktionieren, damit der Nutzer sie auch benutzen kann.

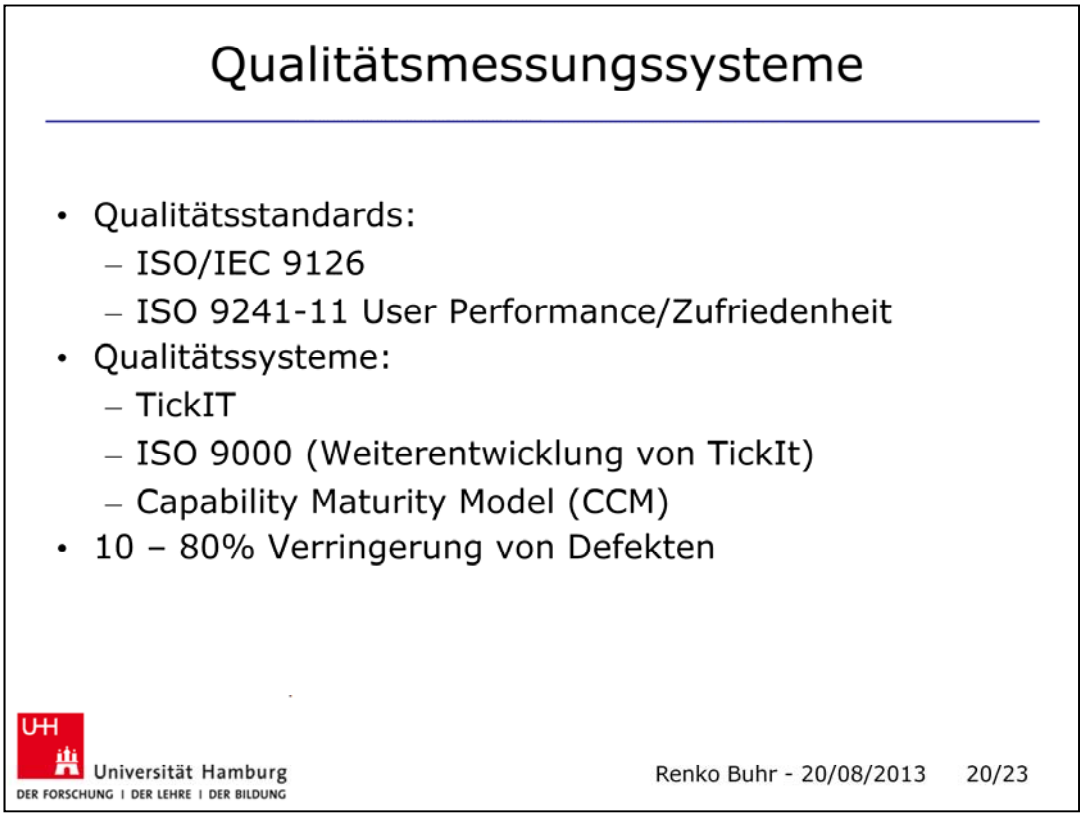
Software kann sich auf verschiedene Weisen verbessern. Die erste Möglichkeit ist die sequenzielle Verbesserung. Dabei geht man das Programm nach und nach durch, und versucht einzelne Teile klar und deutlich zu schreiben und dabei auch Verbesserungen durch zu führen. Die Zielsetzung ist von Beginn an, das Programm zu Verbessern. Die zweite Möglichkeit besteht in der häufigen Nutzung eine Software, insbesondere durch verschiedene Benutzer. Das Programm wird so häufig angepasst und bessere Versionen ersetzen dann die vorherige. Dieses Prinzip ist grob vergleichbar mit natürlicher Selektion in der Natur. Die dritte Möglichkeit ist verschiedenen Tools zu nutzen und Messungen durchzuführen um direkt Schwachstellen im Programm zu finden. Diese können dann gezielt verbessert werden. Hier gibt es standardisierte Verfahren, welche Software nach verschiedenen Gesichtspunkten untersuchen. Diese Verfahren können dann von Nutzer angewendet werden. Dadurch wird die Software zuverlässiger und der Nutzer kann bessere Entscheidungen treffen.

Auch wenn die bestmögliche Qualität wünschenswert ist, ist auch hier Wirtschaftlichkeit ein wichtiger Faktor, da die Qualitätsmessungsverfahren entweder von Fachleuten durchgeführt werden, oder aber die Software je nach Umfang teurer ist, muss hier der Nutzer entscheiden, wie viel Qualität er sich leisten kann.

Qualitätskategorien:

- Erlernbarkeit
- Benutzerfreundlichkeit [*]
- Effizienz [*]
- Fehlererkennung
- Wiederherstellbarkeit
- Installationsaufwand
- Datensicherheit
- Funktionalität [*]
- Zuverlässigkeit [*]
- Portabilität [*]
- Systemsicherheit [*]
- ...

[*]= ISO/IEC 9126



TickIT war die erste Software, welche nach strukturierten Qualitätsstandart, Programme bewertet hat (Fenton, 1991). Über die Jahre sind weitere Systeme entstanden. Die bekanntesten sind ISO 9000 und das Capability Maturity Model (CCM).

Zusammenfassung

- Es gibt Softwareentwicklungsmechanismen
- Weitgehend unbekannt in Wissenschaft
- Dennoch wird Software entwickelt
- Fachwissen ist essenziell
- Wissenschaftler müssen selbst Software entwickeln
- Systementwickler kennen sich nicht im Fachgebiet aus
- Bei größeren Projekten/ Problemen der Wissenschaftler
  => Zusammenarbeit sinnvoll
- Grundkenntnisse im anderen Feld notwendig
- Gegenseitige Erwartungen häufig zu hoch

- Fazit: beide Gruppen sollten voneinander lernen

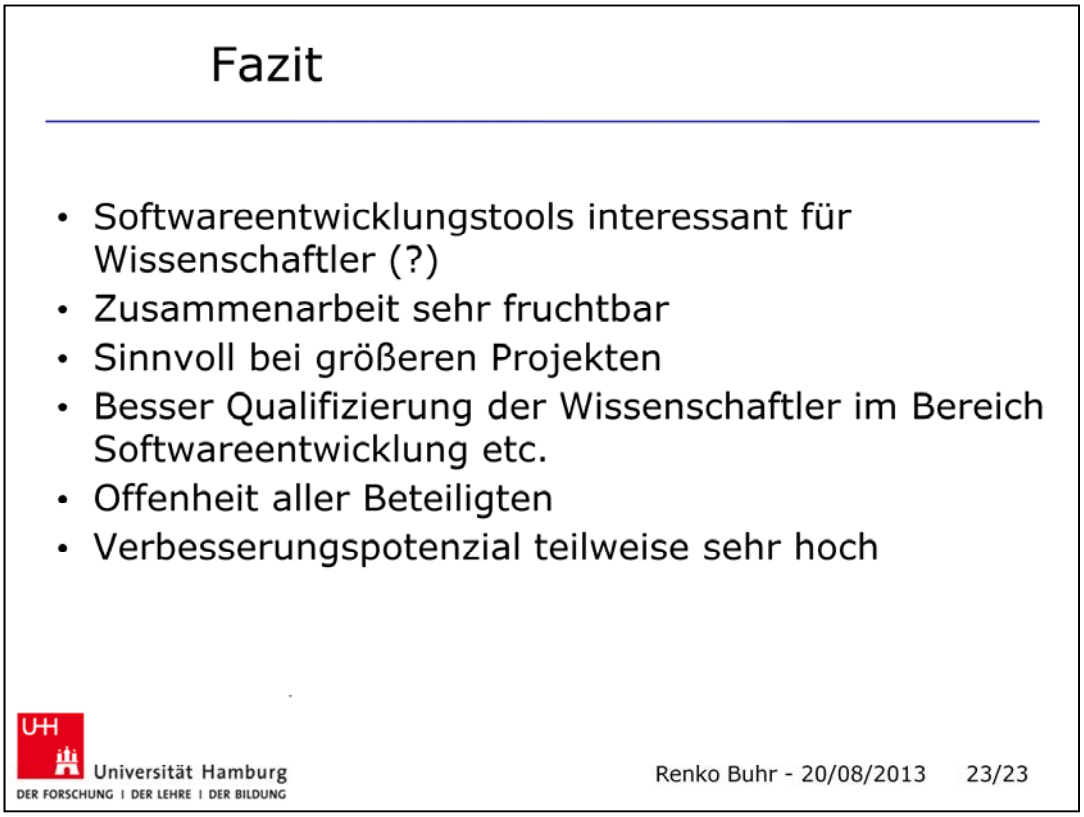
Zusammenfassung

- Software kann häufig verbessert werden
- Bessere Software – Bessere Entscheidungen
- Viele verschiedene Kriterien
- Standards wurden eingeführt
- Systematische Software zur Qualitätsmessung

- Keine Qualitätsmessungen in Wissenschaft

⇒Nutzbar auch in der Wissenschaft ?

Fazit: Besonders wenn Software eine gewisse Komplexität erreicht ist der Einsatz von Qualitätssoftware unabdingbar. Bei einfachen Programmen ist sie allerdings unnötig.



Softwareentwicklungstools sind in jedem Fall interessant für Wissenschaftler, ob sie allerdings sinnvoll sind, wird in der Studie nicht abschließend geklärt. Die Tatsache das sie einfach nicht genutzt werden reicht hier nicht aus, aber die Modelle sind für Projekte ausgelegt, die sich deutlich von wissenschaftlichen Projekten unterscheiden. Systematisch vorzugehen ist sinnvoll, aber der richtige Weg ist möglicherweise ganz anders. Zu Prüfen ist, ob Methoden wie Extreme Programming besser passen.

Besonders fruchtbar ist die Zusammenarbeit für Wissenschaftler