

Effiziente Managementinformation bei agilen Vorgehensweisen – ein Widerspruch?

Herwig Mayr

Fakultät für Informatik,
Kommunikation und Medien Hagenberg
Fachhochschule Oberösterreich
Softwarepark 11, A-4232 Hagenberg, Österreich
Tel. +43 7236 3888 2021, Fax +43 7236 3888 99
E-Mail: Herwig.Mayr@fh-hagenberg.at

Kurzfassung

Eine beträchtliche Anzahl von Praktiken aus agilen Vorgehensmethoden hat sich in der Realität bewährt und mittlerweile Eingang in die allgemeine Softwareentwicklungspraxis gefunden. Beispiele dafür sind die enge Kundeneinbindung, die stark iterativ-inkrementelle Entwicklung oder die effiziente Werkzeugnutzung. Allerdings stehen Aspekte wie Managementinformation und Prozessverbesserung bei agilem Vorgehen nicht im primären Fokus und werden daher gerade bei weniger erfahrenen Entwicklern oft vernachlässigt. Dieser Beitrag zeigt, dass Prozessdaten und Management-Kennzahlen auch bei agilen Vorgehensmethoden effizient gewonnen werden können und durchaus zur (Entwickler-seitigen) Prozessverbesserung als auch zur Steigerung des Kundenwerts („Value-based Software Engineering“) herangezogen werden können. Anhand des „Adaptierten Twin-Peaks-Modells“ werden praktische Beispiele für entsprechende Techniken und Metriken gegeben.

Schlüsselwörter: Software Engineering, Requirements Management, Wertgenerierung, VBSE, Twin Peaks Modell

1 Motivation: Value-Based Software Engineering

Wertgesteuerte Softwareentwicklung („Value-Based Software Engineering, VBSE“) kann definiert werden als die Schaffung eines möglichst hohen Kundenwerts während der Entwicklung eines Softwareprodukts, dessen Anforderungen zu Beginn eher allgemein und unklar sind (frei nach [Boehm, 2005]). Werterzeugung („value generation“) hat sich in letzter Zeit zu einem der Top-Themen im Bereich der Softwareentwicklungsprozesse entwickelt, und zwar aus drei Gründen:

1. Klassische Fortschrittmessung (basierend auf „entwickelten Einheiten“, wie z.B. Codezeilen oder definierten Klassen/Objekten) funktioniert nur für Projekte mit stabilen Anforderungen und einem Wertesystem, das sich im Laufe des Projekts nicht ändert.
2. Businesswerte ändern sich heutzutage sehr schnell und manchmal sehr überraschend. Fortschrittmessung durch „entwickelte Features“ kann daher oft durch die sich ändernde

Bedeutung bzw. Wichtigkeit einzelner Features in die Irre führen (vgl. Software für aktuelle Smartphones/PDAs).

3. Die Gruppe der Stakeholder, die ihren Wert im Rahmen eines Projekts laufend gesichert haben möchte, – und dadurch häufig Kundenwerte verschieben – wird laufend größer und stärker diversifiziert.

Konsequenter Weise müssen die klassische *designgesteuerte* Entwicklung und die agile *testgesteuerte* Entwicklung Platz machen für eine *anforderungsgesteuerte* Entwicklung, bei der der Kundenwert einzelner Anforderungen/Features den primären Steuerungsaspekt darstellt [Boehm und Huang, 2003]. Darüber hinaus gibt nicht länger die (gewichtete) Anzahl von Anforderungen die Richtung der Entwicklung eines Projekts vor bzw. stellt ein geeignetes Fortschrittsmaß dar; es ist der *Kundenwert*, der ausschlaggebend ist. Für den modernen Softwareprojekt-Manager ist daher das Monitoring des Kundenwerts der gerade in Entwicklung befindlichen Features eines Produkts eine Schlüsselaufgabe geworden.

In den nächsten Abschnitten stellen wir das *Twin Peaks Modell* vor, das von uns adaptiert wurde, um ein geeignetes Modell für ein derartiges Projektmonitoring darzustellen.

2 Iterativ-inkrementelle Entwicklung und das (Adaptierte) Twin Peaks Modell

In der modernen Softwareentwicklung ist es essenziell, Anforderungen von der Anforderungsanalyse über das Design in den Code und zu den zugeordneten Testfällen verfolgen zu können [Grünbacher, 2006]. Dies kann nur mit geeigneten Werkzeugen geschehen.

Zusätzlich muss es möglich sein, die Anforderungen mit einem Kundenwert zu gewichten, der sich im Laufe der Zeit ändern kann. Als Entwickler (bzw. Projektleiter) muss man daher laufend die aktuell realisierte Menge an Funktionalitäten (und den Kundenwert der zugehörigen Anforderungen) im Auge behalten, sowie auch den Wert der noch zu realisierenden Funktionalität, um deren Realisierung geeignet priorisieren zu können.

Die (positive) Akzeptanz sich laufend ändernder Anforderungen hat zur modernen Ansicht eines iterativ-inkrementellen Softwareentwicklungsprozesses geführt (siehe [Mayr, 2005] für Details). Dieser Prozess besteht aus drei ineinander verschachtelten Schleifen, die die Anforderungsänderungen, Produktinkremente und Entwicklungsschritte darstellen (vgl. Abbildung 1).

Nuseibeh hat in 2001 das Konzept des „Twin Peaks Modells“ für ein in Entwicklung befindliches Softwareprodukt eingeführt, bei dem ein „Peak“ die ständig wachsende Anzahl von Anforderungen im Laufe der Zeit darstellt, und der zweite „Peak“ die Architektur der entwickelten Software bzw. deren steigende Verflochtenheit mit den bislang realisierten Anforderungen, vgl. [Nuseibeh, 2001]. Der Hauptzweck dieses Modells ist, den ständig wachsenden Grad der Implementierungsabhängigkeit eines in Entwicklung befindlichen Softwareprodukts darzustellen sowie die laufend schrumpfenden Möglichkeiten, Spezifikationsänderungen effizient einzubauen.

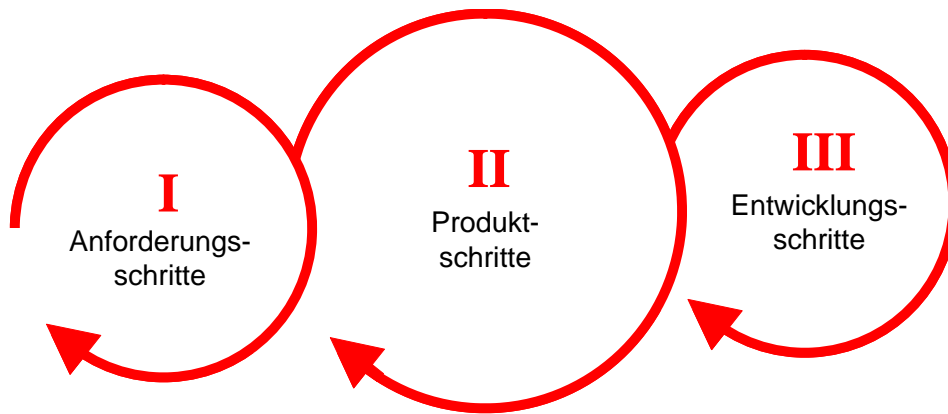


Abb. 1: Moderner iterativ-inkrementeller Softwareentwicklungsprozess

Wir haben das Basiskonzept von Nuseibeh erweitert, um die steigende Anzahl der Funktionalitäten, die im Lauf eines Softwareprojekts spezifiziert werden, sowie deren Kundenwert darzustellen, und diesen den Verlauf der Softwareentwicklung gegenüberzustellen. In unserem so genannten *Adaptierten Twin Peaks Modell* definieren wir zusätzlich die Überlappung der beiden Peaks als das Produkt selbst, d.h. den Teil der entwickelten Software, der Kundenanforderungen erfüllt (vgl. Abbildung 2). Diese Sicht korrespondiert sehr gut mit den drei verschachtelten Iterationen des iterativ-inkrementellen Softwareentwicklungsprozesses, wie Abbildung 2 darstellt.

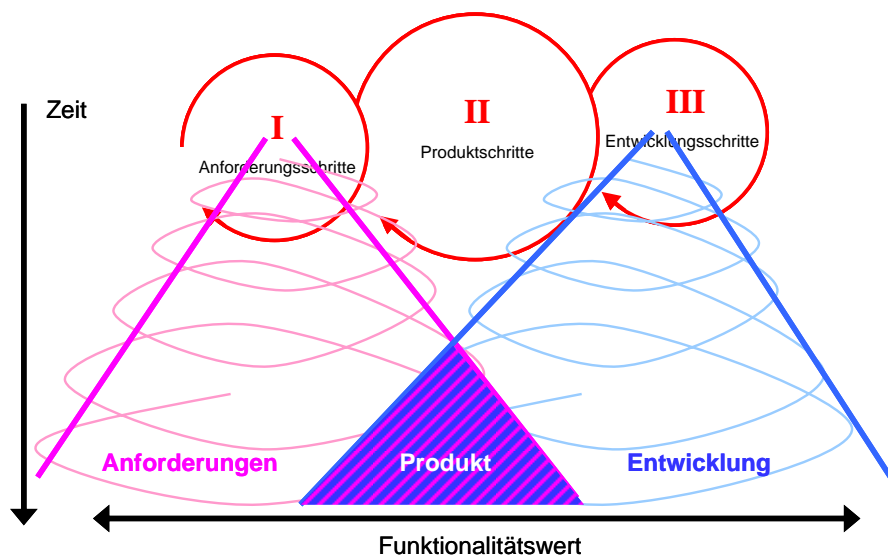


Abb. 2: Herleitung des (Adaptierten) Twin Peaks Modells aus dem iterativ-inkrementellen Softwareentwicklungsprozess

3 Definition einer geeigneten Metrik für den Funktionalitätswert

Wenn die Wertemetrik für die individuellen Anforderungen geeignet gewählt wird, erlaubt die Visualisierung des Adaptierten Twin Peaks Modells eine gute Abschätzung des Verhältnisses

zwischen Kundenwert und Entwicklungsaufwand zu jeder beliebigen Zeit während einer Softwareentwicklung.

Unser Adaptiertes Twin Peaks Modell stellt die geforderte und realisierte Funktionalität auf der X-Achse dar (gewichtet mit dem Kundenwert), und den Verlauf der Entwicklungszeit auf der Y-Achse. Abbildung 3 zeigt, wie das adaptierte Twin Peaks Modell zur Identifikation (und Interpretation) des aktuellen Projektstatus verwendet werden kann.

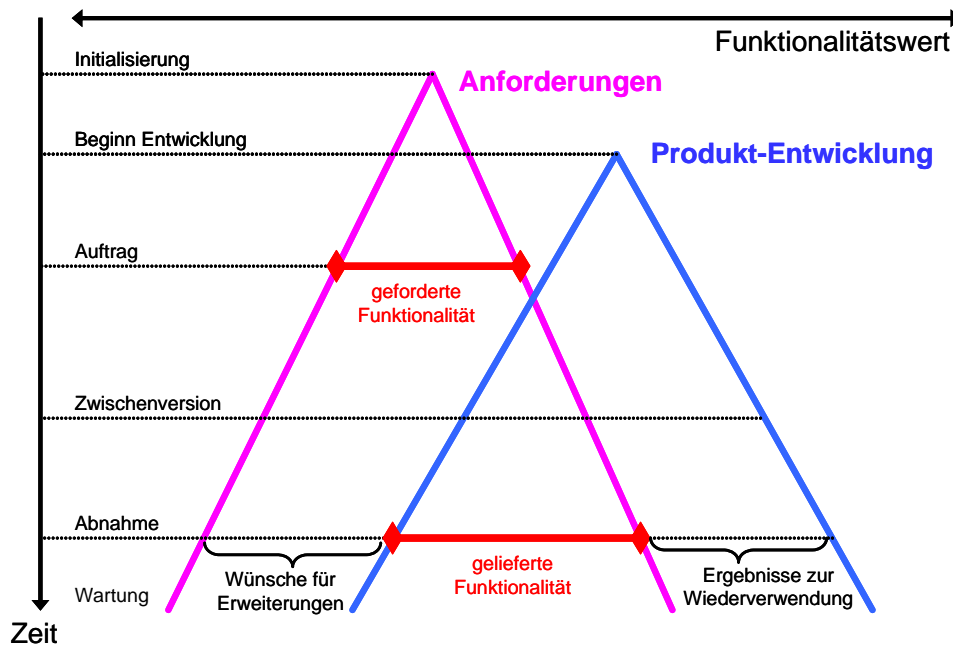


Abb. 3: Bedeutung des Funktionalitätswerts (gesamt)

Um eine sinnvolle Interpretation zu ermöglichen, muss eine geeignete Metrik für den Funktionalitätswert definiert werden. Diese Metrik muss in Betracht ziehen, dass der geschätzte Businesswert eines Produktfeatures primär durch den Kunden festgelegt werden kann („educated guess“), und mit dem (durch die Entwickler) geschätzten (und später tatsächlich erbrachten) Entwicklungsaufwand in Relation gebracht werden muss.

Derzeit verwenden wir folgende Metrik für den *Funktionalitätswert (FV)* für unsere Softwareentwicklungsprojekte:

$$FV = \sum \frac{FVE_i^2}{CFE_i}, \quad (1)$$

wobei *FVE* den *geschätzten Funktionalitätswert (functionality value estimate)* darstellt, der durch den Businesswert für den Kunden festgelegt wird (ein Integer-Wert im Bereich von 1 bis 10), und *CFE* den *aktuellen Funktionalitätsaufwand (current functionality effort)*, spezifiziert in Arbeitsstunden ([h]; siehe folgenden Absatz). Der Summenoperator addiert die Werte über alle Anforderungen *i*.

Der aktuelle Funktionalitätsaufwand *CFE (current functionality effort)* wird für jede bereits spezifizierte Anforderung auf folgende Weise berechnet:

$$CFE = \sum_j RTE_j + RRE, \quad (2)$$

wobei **RTE** den realisierten Aufwandaufwand (*realized task effort*) für jede Aufgabe *j* darstellt, gemessen in [h], der aus den auf diese Aufgabe gebuchten Arbeitsstunden durch die Entwickler resultiert. **RRE** umfasst den restlichen Anforderungsaufwand (*remaining requirement effort*), der durch den Produktverantwortlichen im Entwicklungsteam abgeschätzt wird (der Initialwert wird häufig vom Produktverantwortlichen gemeinsam mit dem Kunden abgeschätzt). **RRE** deckt also denjenigen Teil einer Anforderung ab, der bereits bekannt ist, aber noch nicht durch bereits spezifizierte Aufgaben abgedeckt ist, und wird ebenfalls in [h] angegeben.

Es ist wichtig anzumerken, dass Funktionalität, die durch die Entwickler als intern erforderlich (vgl. dazu Abschnitt 5) eingestuft wurde (z.B. um die Wiederverwendbarkeit des Produkts zu erhöhen), aber noch nicht – durch Aufgaben – realisiert wurde, nicht im Werteschema unseres Modells aufscheint, da sie keinen Businesswert für den aktuellen Kunden darstellt, sondern nur einen möglichen zukünftigen internen Wert für den Entwickler.

4 Analyse des Funktionalitätswerts im Verlauf der Zeit

Wenn man die Änderungen des Funktionalitätswert im Lauf der Zeit analysiert, ermöglichen dessen Evaluierung zu bestimmten Meilensteinen sowie Vergleiche in bestimmten Zeitintervallen das Sammeln von Managementinformation und die Ableitung von Projektfortschrittsmaßen. Abbildung 4 illustriert die Klassifikation des Funktionalitätswerts in fünf Segmente (gekennzeichnet mit A bis E), die die Bestimmung von Verhältnissen zueinander für Analysezwecke ermöglichen:

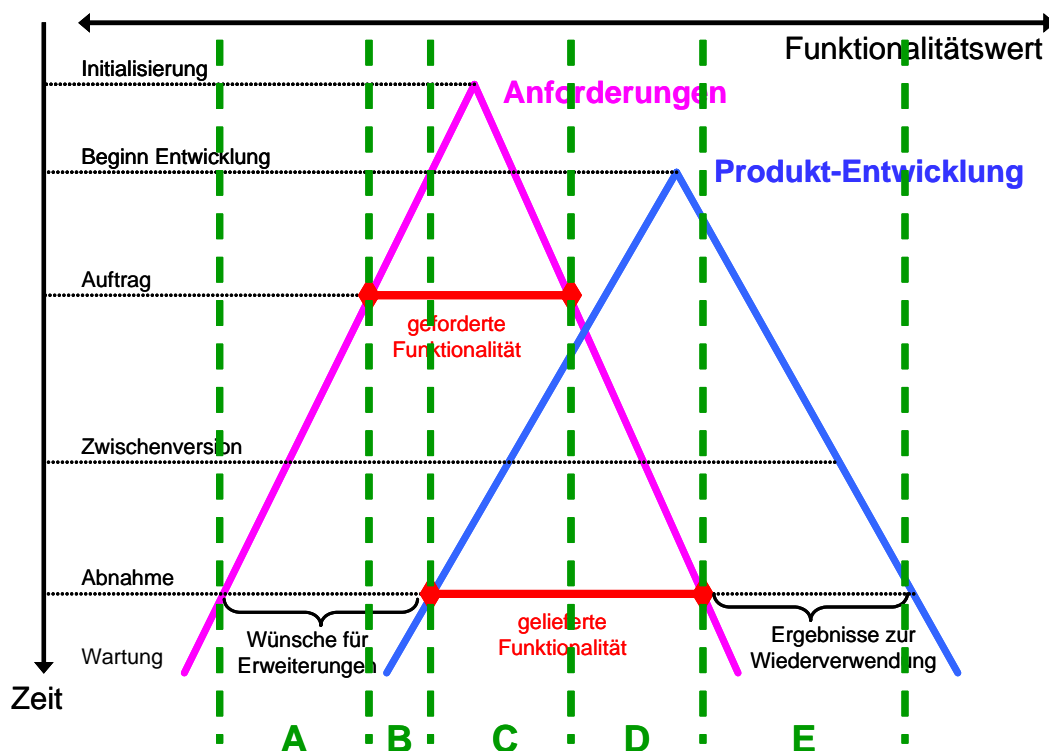


Abb. 4: Bedeutung des Funktionalitätswerts im Verlauf der Zeit

- A. vom Kunden geforderte Funktionalität *nach* Vertragsabschluss, die *nicht* realisiert wurde,
- B. vom Kunden geforderte Funktionalität *im* Vertrag, die *nicht* realisiert wurde,
- C. vom Kunden geforderte Funktionalität *im* Vertrag, die realisiert wurde,
- D. vom Kunden geforderte Funktionalität *nach* Vertragsabschluss, die realisiert wurde,
- E. vom Kunden *nicht* geforderte Funktionalität, die realisiert wurde.

Unter Verwendung der Länge (= des Funktionalitätswerts) dieser fünf Segmente können die folgenden Maße jederzeit während eines Projekts aus dem Adaptierten Twin Peaks Modell abgeleitet werden:

- $[C:(B+C)]$: Status der *Verifikation* (vom Kunden geforderte Funktionalität),
- $[(C+D):(C+D+E)]$: Status der *Validierung* (vom Kunden nutzbare Funktionalität),
- $[E:(C+D+E)]$: prozentueller Anteil der *Entwicklung zur Wiederverwendung*,
- $[(C+D):(B+C)]$: gelieferte Funktionalität versus bezahlte Funktionalität (“*Erfolg*”),
- $[(B+C):(C+D+E)]$: bezahlte Funktionalität versus entwickelte Funktionalität (“*Wirtschaftlichkeit*”),
- $[(C+D):(A+B)]$: gelieferte Funktionalität versus offene Wünsche (“*Kundenzufriedenheit*”).

Die Überwachung dieser Verhältnisse kann dem Projektmanager helfen, laufend den Projektstatus einzuschätzen und Probleme rechtzeitig zu erkennen, seien es ungenügender Entwicklungsfortschritt oder überproportionales Ansteigen der Kundenwünsche. Auf diese Weise kann unser Adaptiertes Twin Peaks Modell als ein schlankes, effizientes Managementinformations-Werkzeug für iterativ-inkrementelle („agile“) Softwareentwicklungsprozesse eingesetzt werden.

Derzeit wird im Rahmen einer Bachelorarbeit eine Optimierung der Bewertungsmetrik für den Funktionalitätswert untersucht und die Praxistauglichkeit des Systems (gerade im Hinblick auf effiziente Bedienung) sowie der angegebenen Maße anhand von Fallbeispielen im Studienbetrieb (Studienprojekte) und in unserer F&E-Abteilung erprobt.

5 Ermittlung des Management-Overhead

Wir definieren den Management-Overhead als das Verhältnis zwischen internen und externen (bzw. den gesamten) Aufwänden in einem Projekt. Diesen explizit auszuweisen erscheint uns wichtig, um den Overhead selbst als notwendig und begründbar und seine Höhe als akzeptabel darzustellen.

Entscheidend ist dabei, für jedes Projekt (besser noch projektübergreifend) ein geeignetes Einteilungsschema zu definieren (was als Overhead gewertet wird) und dieses konsistent anzuwenden (vgl. [Haager, 2006] für einen diesbezüglichen Vorschlagskatalog unsererseits). Wir betrachten sämtliche interne Aufgaben, also Aufgaben, die der Auftraggeber nicht unmittelbar bezahlen würde, als Management-Overhead. Dies inkludiert organisatorische Aufgaben (z.B. Entwicklerbesprechungen durchführen) wie auch Softwareentwicklungen, die dem Kunden nicht direkt ausweisbaren Nutzen bringen (z.B. Generalisierung des Codes für bessere Wiederverwendbarkeit oder Code-Refactoring).

Unter dieser Betrachtungsweise haben wir die Erfahrung gemacht, dass bei den von uns betrachteten Projekten der Management-Overhead typischer Weise immer über 20% des Gesamtaufwands liegt und – speziell bei sehr kurzen Iterationszyklen – bis zu 40% ausmachen kann! Daraus kann für die weitere Projektverlaufszeit eine Prognose für den Management-Overhead erstellt werden, die eine Grundlage für die verplanbaren Aufwände in den nächsten Iterationen darstellt. Wir unterscheiden bei dieser Prognose zwischen realem, tendenziellem und erwartetem Management-Overhead, je nach Eingang des bislang aufgetretenen Overhead und des bereits abschätzbaren in der Projektrestlaufzeit (siehe Abbildung 5 aus [Haager, 2006]).

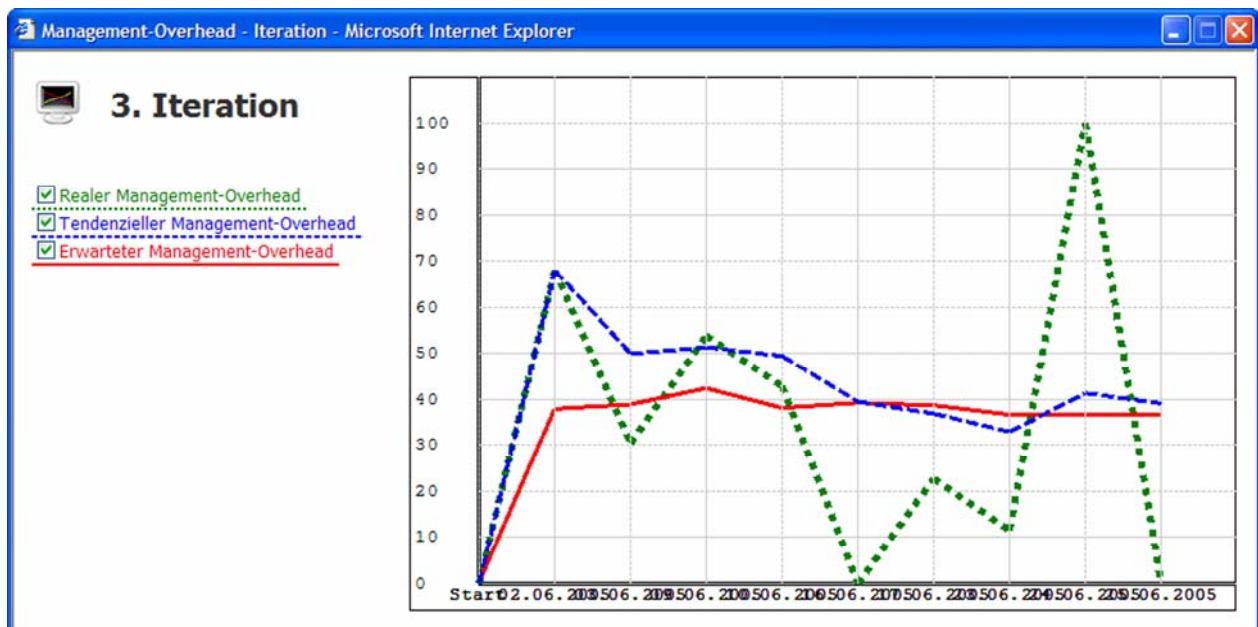


Abb. 5: Varianten des Management-Overhead – Beispiel

Neben der besseren Begründbarkeit von Management-Overhead und der „ehrlicheren“ Angabe von Werten ergeben sich durch einen klar definierten, explizit ausgewiesenen Management-Overhead weitere Vorteile:

- leichteres Erkennen von „overmanaged“ Projekten, Iterationen, etc.,
- präziser ermittelbare Aufschläge für interne Aufwände,
- genauere Kontrolle der getätigten Aktionen aus Managementsicht,
- bessere Reflexion von vergangenen Projektphasen bzw. Vergleich von Projektphasen und Projekten.

Aus dem Adaptierten Twin Peaks Modell kann der Management-Overhead (in der Art, wie wir ihn definieren) aus der Einteilung zwischen externen Aufgaben (im Anforderungs-Peak enthalten) und internen Aufgaben (nur im Produktentwicklungs-Peak enthalten) abgelesen werden, wobei zur Berechnung nur die tatsächlich geleisteten Aufwände (bzw. geschätzten Werte für zukünftige Aufwände) herangezogen werden und der Kundenwert keinen Eingang findet.

6 Nutzen des Adaptierten Twin Peaks Modells für die wertgesteuerte Softwareentwicklung (Value-Based Software Engineering)

Wenn das Adaptierte Twin Peaks Modell aus den Projektdaten mit einem geeigneten Werkzeug generiert und laufend aktuell gehalten wird, so kann es den Projektleiter bei einem sehr entscheidenden Schritt des modernen, iterativ-inkrementellen („agilen“) Softwareentwicklungsprozesses unterstützen: *der Auswahl der geeignetsten und für den Kunden wertigsten Anforderungen zur Realisierung in der nächsten Iteration!* Eine geeignete Auswahl der jeweils nächsten Anforderungen, die in Funktionalität umgesetzt werden sollen, ist eine der Schlüsselaufgaben moderner „agiler“ Entwicklung. Häufig stellt diese eine primäre Quelle für Projektverzögerungen (oder das völlige Scheitern) dar, insbesondere wenn die Auswahl aus Entwicklersicht erfolgt, ohne auf ein geeignetes Werteschema aus Kundensicht zurückzugreifen (vgl. [Boehm und Turner, 2003]).

Ein anderer Aspekt, der von einer koordinierten, prioritätsgesteuerten Anforderungsauswahl profitiert, ist die testgesteuerte Entwicklung (*test-driven development*). Nur wenn Anforderungen kontinuierlich auf Grund des Kundenwerts ausgewählt werden (unter Verwendung eines wohl definierten Vorgehens und Modells!), können geeignete Testfälle für diese Anforderungen rechtzeitig erstellt werden. Mit diesen Testfällen kann dann die Entwicklung in die richtige Richtung gesteuert werden, i.e. die Entwickler erstellen (und testen!) diejenige Funktionalität, die aktuell den maximalen Wertzuwachs für den Kunden bringt (siehe Abbildung 6). Testgesteuerte Entwicklung ohne entsprechendes Anforderungsmanagement ist eine Farce. [Mayr, 2006]

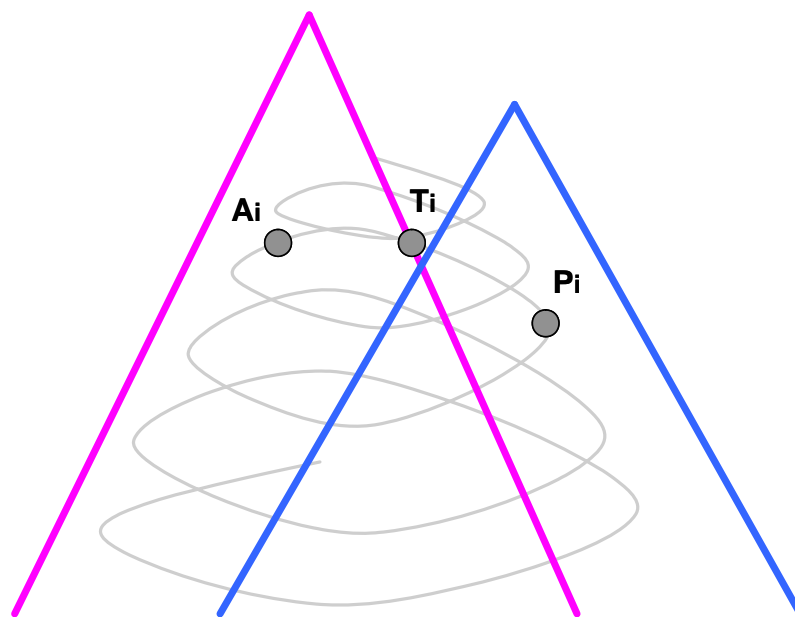


Abb. 6: Ableiten von Testfällen aus Anforderungen für die testgesteuerte Entwicklung (**A** = Anforderungen, **T** = Tests, **P** = Produktfunktionalitäten)

7 Szenarios und Werkzeuge

Das Adaptierte Twin Peaks Modell kann dazu verwendet werden, verschiedenste Arten der Vorgehensweise bei der Softwareentwicklung zu visualisieren und die in Abschnitt 4 erwähnten Maße zu generieren sowie deren Veränderung zu verfolgen. Drei (einfache) Beispiele sind in Abbildung 7 dargestellt.

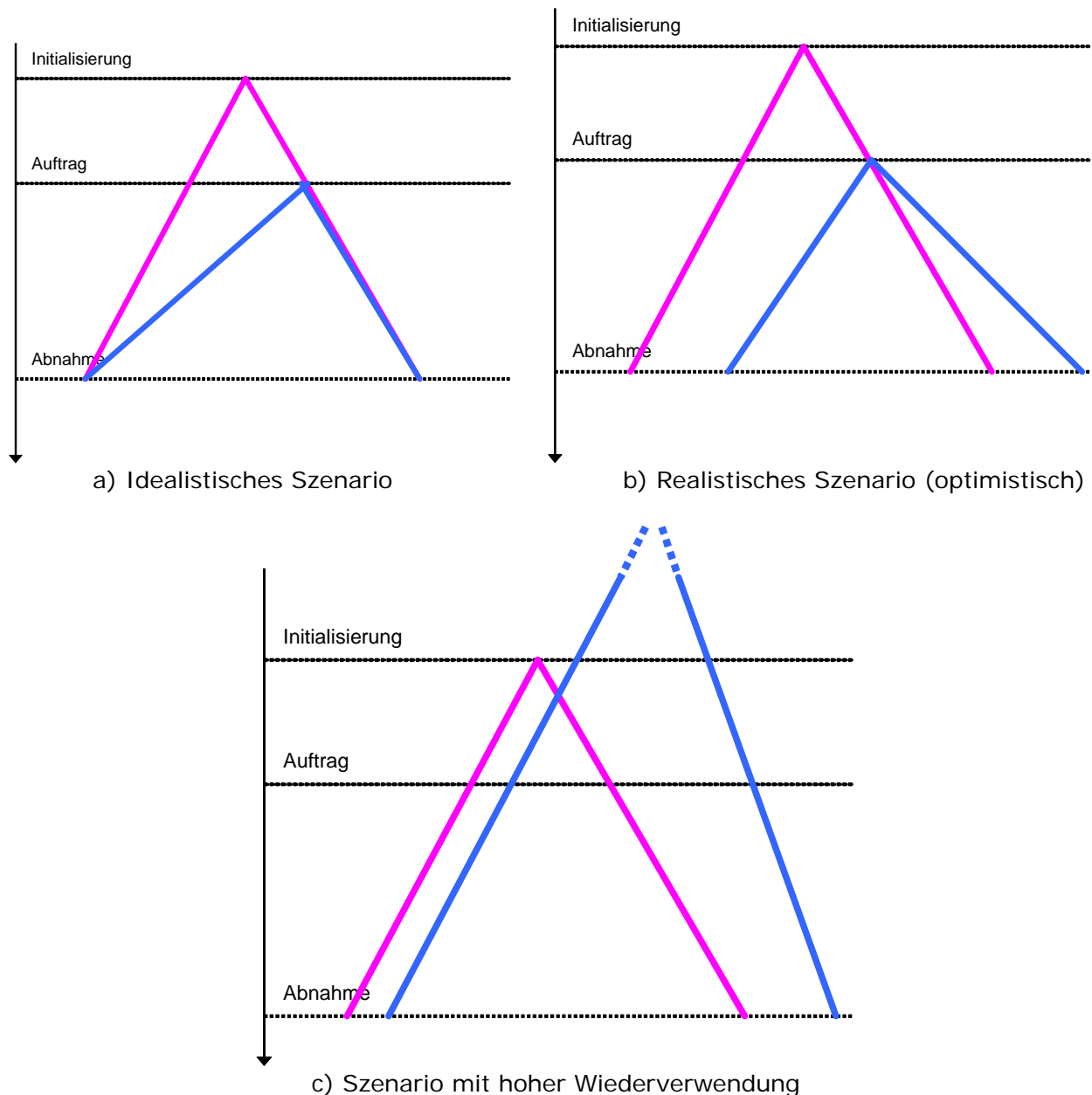


Abb. 7: (Adaptiertes) Twin Peaks Modell: Szenarios

- ein sehr idealistisches Szenario, in dem ein klar spezifiziertes Produkt von Grund auf entwickelt wird, ohne jeglichen Overhead,
- ein eher realistisches Szenario einer Produktentwicklung von Grund auf, bei dem Overhead anfällt (der hoffentlich zu zumindest teilweise wieder verwendbaren Komponenten

führt); selbst in diesem Fall ist die dargestellte Überlappung der beiden Peaks von Anfang an (und damit eine Verfügbarkeit eines Zwischenprodukts für den Kunden von der ersten Iteration an) sehr unwahrscheinlich,

- c. eine stark auf Wiederverwendung basierende Entwicklung, die die Ausrichtung der Kundenanforderungen auf Funktionalität ermöglicht, die bereits in wieder verwendbaren Komponenten verfügbar ist (natürlich in vernünftigem Rahmen).

Um die „Twin Peaks Charakteristik“ eines Projekts rasch visualisieren zu können, haben wir ein einfaches, derzeit MS Office-basiertes Werkzeug entwickelt, das die Daten aus einem Projektmanagement-/überwachungstool übernimmt und eine aktuelle, Excel-basierte Visualisierung erstellt (siehe Abbildung 8 für ein Beispiel bzw. [Mayr, 2007] für Details).

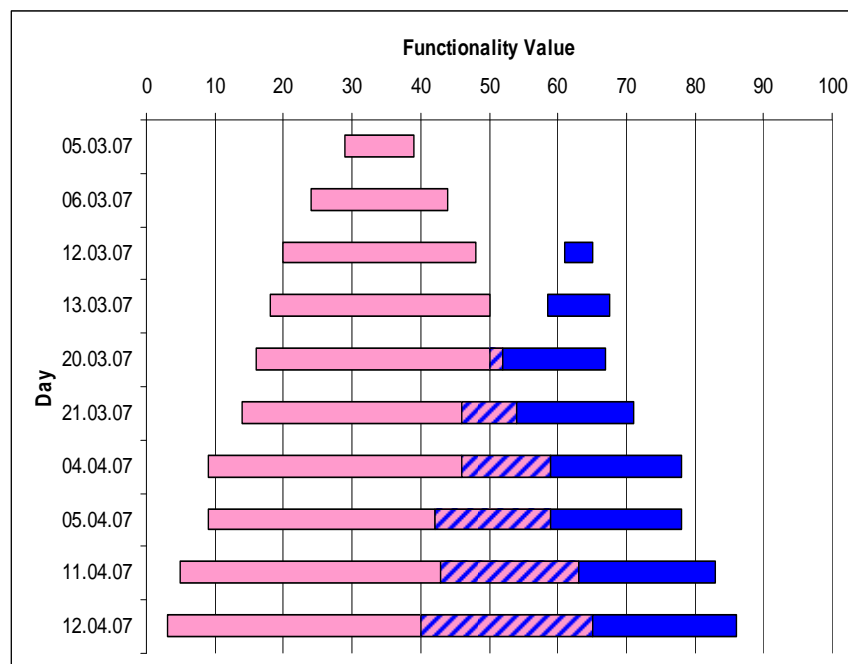


Abb. 8: Adaptiertes Twin Peaks Modell: Einfaches Monitoring-Werkzeug zur Managementinformation

Nach unserer Erfahrung stellt das Adaptierte Twin Peaks Modell ein exzellentes Werkzeug dar zur Visualisierung eines modernen, iterativ-inkrementell abgewickelten Softwareprojekts. Die Hauptvorteile sind:

- Es erlaubt die Auswahl der für den Kunden wichtigsten (wertigsten) Anforderungen zur Realisierung in jeder Iteration.
- Testgesteuerte Entwicklung kann (auch aus Kundensicht!) sinnvoll eingesetzt werden.
- Es hilft dem Projektleiter rasch festzustellen, ob ein Projekt aus dem Rahmen läuft (seien es zu häufige Anforderungsänderungen durch den Kunden, die Entwicklung falscher – oder falsch priorisierter – Funktionalität durch die Entwickler etc.).

- Es ist ein exzellentes Modell zur Visualisierung der „Projektkultur“ und könnte dadurch zur Verbesserung der Professionalität in der Softwareentwicklung im Laufe der Zeit beitragen.
- Das Adaptierte Twin Peaks Modell ermöglicht ein laufendes Assessment des Projektstatus (Messwerte!). Es stellt daher ein schlankes, effizientes Managementinformations-Werkzeug für die iterativ-inkrementelle Entwicklung dar.

Mit dem Adaptierten Twin Peaks Modell haben wir bislang positive Erfahrungen gesammelt sowohl bei Studenten im Rahmen von Projektgruppen an unserer Fachhochschule als auch bei Entwicklungsteams in unserer Forschungsabteilung und (in kleinerem Rahmen) in der Wirtschaft. Dies motiviert uns, unser Modell besser in moderne Entwicklungsumgebungen bzw. Projektmanagement-Werkzeuge einzubinden. Weiters werden wir Varianten der Funktionalitätswerte-Metrik untersuchen. Diese Aufgaben stellen unser nächstes Arbeitspaket dar, um eine bessere Modell- und Werkzeugunterstützung für die moderne iterativ-inkrementelle Softwareentwicklung auch aus der Sicht der Managementinformation zu bieten.

Dank

Der Autor möchte Paul Grünbacher von der Johannes Kepler Universität Linz und Rudolf Ramler vom SCCH Hagenberg für die vielen fruchtbringenden Diskussionen zum Thema der wertgesteuerten Softwareentwicklung danken.

Literatur

- [Boehm, 2005] B.W. Boehm. *Value-Based Software Engineering: Overview and Agenda*. Technischer Bericht USC-CSE-2005-504, University of Southern California, USA, 2005.
- [Boehm und Turner, 2003] B.W. Boehm und R. Turner. *Balancing Agile and Plan-Driven Methods: A Guide for the Perplexed*. Addison Wesley, 2003.
- [Boehm und Huang, 2003] B.W. Boehm and L.G. Huang. Value-Based Software Engineering: A Case Study. *IEEE Computer*, 36(3):33-41, 2003.
- [Grünbacher, 2006] P. Grünbacher. *Requirements Engineering for Web Applications*. In G. Kappel et al., Hrsg., *Web Engineering*, Ch. 2, pp. 23-38, Wiley, 2006.
- [Haager, 2006] F. Haager. *Feststellung von Management-Overhead und individuellen Leistungsdaten in Team-Projekten*. Bachelorarbeit, Studiengang Software Engineering, Fachhochschule OÖ Hagenberg, A-4232 Hagenberg, 2006.
- [Mayr, 2007] H. Mayr. *Value-Based Software Engineering Using the Adapted Twin Peaks Model*. In A. Bruzzone et al., Hrsg., *Proc. Intl. Mediterranean Modeling Multiconference I3M'07*, 4.-6. Oktober 2007, Bergeggi, Italien, Universität Genua 2007.
- [Mayr, 2006] H. Mayr. *Web Project Management*. In G. Kappel et al., Hrsg., *Web Engineering*, Kap. 9, S. 171-196, Wiley, 2006.
- [Mayr, 2005] H. Mayr. *Projekt Engineering – Ingenieurmäßige Softwareentwicklung in Projektgruppen*. 2. Auflage, Fachbuchverlag Leipzig / Hanser, 2005.
- [Nuseibeh, 2001] B. Nuseibeh. Weaving Together Requirements and Architectures. *IEEE Computer*, 34(3):115-117, 2001.

Über den Autor



Dipl.-Ing. Dr. Herwig Mayr ist Professor für Software Engineering an der Fachhochschule Oberösterreich in Hagenberg und leitet den übergeordneten Fachbereich für Projekt Engineering und Softwareprozesse. Er besitzt ein Doktorat für Technische Wissenschaften (Informatik) der Johannes Kepler Universität Linz und ist neben seiner Lehr- und Forschungstätigkeit seit mehr als 20 Jahren als selbstständiger IT-Berater tätig. Erreichbar ist er per E-Mail unter Herwig.Mayr@fh-hagenberg.at.