mehr zum thema:
www.lpzradar.de
http://portal.acm.org/citation.cfm?id=642249

METHODEN UND GRUNDLAGEN DER WERTEBASIERTEN SOFTWAREENTWICKLUNG

Unter wertebasierter Softwareentwicklung versteht man Methoden, Modelle und Messgrößen, mit denen Manager, Entwickler und Anwender von Software verschiedene Zielkonflikte im Softwareprozess auf wirtschaftlicher Basis bewerten und auflösen können. In diesem Artikel stellen wir einige dieser Ansätze vor und zeigen, in welchen Situationen diese von Nutzen sein können.

Hin und wieder kommt es vor, dass ein IT-Leiter – sei es aus eigener Motivation oder auf Nachfrage seiner Vorstandskollegen – die Frage stellt: „Welchen Wert stellt denn dieses oder jenes Informationssystem für unser Unternehmen dar?" Die gefragten IT-Mitarbeiter antworten dann üblicherweise in der Art „Das ist eine schwierige Frage" oder „Das hängt ganz davon ab". Einerseits haben die Gefragten natürlich recht – doch andererseits sind solche Antworten nicht zufriedenstellend, wann man die mit den Informationssystemen verbundenen Investitionssummen betrachtet.

Für die IT-Community ist es nun einmal nach wie vor schwierig, den Wert einer Investition in Soft- und Hardware zu ermitteln. Und bei dieser Wertbestimmung kommt es – wie bei jeder Wertbestimmung – zunächst darauf an, genau festzulegen, welcher Gegenstand die Basis der Betrachtung darstellt. Anschließend muss bestimmt werden, aus wessen Blickwinkel der Wert des Gegenstands ermittelt werden soll. Beides ist im Bereich der Software schwierig, da sich Softwareinvestition nur schwerlich genau voneinander abgrenzen lassen. Wenn man ein System im Kontext einer Organisation betrachtet, wie geht man dann mit den Schnittstellen zu anderen Systemen um? Zählen Hard- und Middleware mit? Kann die betrachtete Anwendung ohne andere Anwendungen existieren? Wozu zählen Kosten, die für Schnittstellen zu anderen Systemen aufgewendet werden müssen? Und wessen Blickwinkel müssen bei der Wertbetrachtung berücksichtigt werden?

Wertebasierte Softwareentwicklung (*Value-Based Software-Engineering*) adressiert genau diese Fragestellungen, die sich zur Entscheidungsunterstützung auf eine Wertbestimmung von Softwaresystemen beziehen. Sie macht es sich zum Ziel, Entscheidungen zu erleichtern, die unter konkurrierenden (manchmal auch widersprüchlichen) Kriterien in unsicheren Kontexten gefällt werden müssen. Da Software nun einmal viele Stakeholder hat – seien es Entwickler, Benutzer oder Manager – ist es unerlässlich, zur Entscheidungsunterstützung Modelle und Messgrößen zu entwickeln, mit denen alle Stakeholder die dem Softwareprozess inhärenten Zielkonflikte, z. B. Qualität vs. Kosten oder Funktionalität vs. Zeitplan, auf wirtschaftlicher Basis bewerten und auflösen können.

Um exemplarisch einen Überblick über das Thema wertebasierte Softwareentwicklung zu geben, stellen wir in diesem Artikel eine Reihe bewährter Vorgehensweisen (*Best Practices*) dazu vor. Diese haben das Potenzial, bei überschaubarem Aufwand die Abwägung zwischen Entscheidungsvarianten leichter zu machen, systematisch Stakeholder zu identifizieren und einzubinden sowie eine Projektsteuerung und -kontrolle zu ermöglichen, die deutlich über das Zählen von Codezeilen oder Manntagen hinausgeht.

Dazu fassen wir aus dem aktuellen Buch „Value-Based Software-Engineering" (vgl. [Bif06]) das Kapitel „Value-Based Software-Engineering: Seven Key Elements and Ethical Considerations" zusammen, da dieses eine Reihe von konkreten Ausgangspunkten enthält, mit denen Unternehmen das Thema wertebasierte Softwareentwicklung ganz praktisch auf ihre Agenda setzen und ins Tagesgeschäft integrieren können. Diese Methoden umspannen den Lebenszyklus eines Softwareprojekts und erleichtern eben darum die Abwägung der Zielkonflikte, um durch eine sinnvolle Fokussierung wertbasierte Anwendungsentwicklung betreiben zu können.

die autoren

Sören Blom (E-Mail: blom@ebus.informatik.uni-leipzig.de) ist wissenschaftlicher Mitarbeiter an der Professur für Angewandte Telematik/e-Business an der Uni Leipzig.

Prof. Dr. Volker Gruhn (E-Mail: gruhn@ebus.informatik.uni-leipzig.de) ist Inhaber der Stiftungsprofessur der Deutschen Telekom AG für Angewandte Telematik/e-Business an der Uni Leipzig.

André Köhler (E-Mail: koehler@ebus.informatik.uni-leipzig.de) ist wissenschaftlicher Mitarbeiter an der Professur für Angewandte Telematik/e-Business an der Uni Leipzig.

Clemens Schäfer (E-Mail: schaefer@it-factum.de) ist Geschäftsführer der it factum GmbH in Garching bei München und wissenschaftlicher Mitarbeiter an der Professur für Angewandte Telematik/e-Business an der Uni Leipzig.

Softwaretrends im IT-Radar
Die Professur für Angewandte Telematik / e-Business an der Universität Leipzig bietet mit dem LPZ IT-Radar einen Informationsservice für IT-Entscheider an. Regelmäßig werden dazu neue Trends vorgestellt und bewertet. Der Service ist frei verfügbar und kann unter www.lpzradar.de abgerufen werden. Im LPZ IT-Radar finden Sie auch weiterführende Informationen zum Thema wertebasierte Softwareentwicklung.

Idee der wertebasierten Softwareentwicklung

Ideen und Verfahren sind oft mit Köpfen verbunden. Und so wird die wertebasierte Softwareentwicklung oft mit dem Namen Barry Boehm in Verbindung gebracht.

Nachdem Barry Boehm und Kevin Sullivan im Jahr 2000 eine erste Forschungsagenda zum Thema wertebasierte Softwareentwicklung vorgeschlagen hatten (vgl. [Boe00]), fand dieses Thema breitere Beachtung in der akademischen Community. In den vergangenen Jahren sind mittlerweile in diesem Kontext beachtliche wissenschaftliche Aktivitäten und Ergebnisse zu verzeichnen, die nun zu praxisverwertbaren Ergebnissen geführt haben – und damit auch den Transfer in die nichtakademische Welt ermöglichen.

Wesentliche Erkenntnis bei der wertebasierten Softwareentwicklung ist, dass ein Softwareprojekt nicht in voneinander unabhängige Phasen mit reinem Business-Bezug am Anfang des Projekts und reinem IT-Bezug im späteren Verlauf aufgeteilt werden darf, sondern dass nur eine ganzheitliche Betrachtungsweise es ermöglicht, den Fokus frühzeitig auf mögliche Konflikte der verschiedenen Stakeholder zu legen, diese zu verdeutlichen und konstruktiv aufzulösen.

In seinem Übersichtskapitel in [Boe00] führt Boehm die folgenden Elemente der „Value-Based Software Engineering Agenda" auf:

- Unter dem Oberbegriff *Value-Based Requirements-Engineering* lassen sich Prinzipien und Praktiken zusammenfassen, mit denen erfolgskritischen Stakeholder eines Systems bestimmt und die Werte, die diese Stakeholder in dem System sehen, ermittelt werden. Das geschieht, um diese Werte anschließend im gegenseitigen Einvernehmen miteinander abzustimmen und so zu einer Menge von Zielvorgaben für das zu entwickelnde System zu gelangen. Dazu bedient man sich Verfahren zur Priorisierung von Releases und Anforderungen, Analysetechniken für Business-Cases sowie Vorgehensweisen zur systematischen Identifikation von Stakeholdern und zur Verhandlung von Anforderungen.
- *Value-Based Architecting* macht es sich zum Ziel, die Vorgaben der Stakeholder für ein System durch architektonische Entscheidungen möglichst optimal zu erfüllen. Dazu benutzt man neben der bekannten Methode *Architecture Trade-off Analysis* (vgl. [Cle02]) zum Beispiel auch Ansätze zur konkurrierenden Entwicklung von Software oder Multiattribut-Entwurfs- und Entscheidungsunterstützungsverfahren.
- So wie man beim *Value-Based Architecting* versucht, die zwischen allen Stakeholdern abgestimmten Vorgaben und Ziele für ein System in die Architektur einfließen zu lassen, so macht es sich das *Value-Based Design and Development* zum Ziel, diesen Stakeholder-Einfluss auch während Entwurf und Implementierung der Software sicherzustellen. Als Beispiele seien an dieser Stelle *Traceability*-Techniken oder agile Methoden mit ihrer Anwender-Einbeziehung genannt.
- Bei *Value-Based Verification and Validation* geht es schließlich um Methoden und Verfahren, mit denen man nach der Implementierung eines Softwaresystems feststellen kann, ob die mit den Stakeholdern ausgehandelten Zielvorgaben auch tatsächlich erreicht wurden. Zudem gibt es Ansätze, die Verifikations- und Validierungsaufwände als Investition zu betrachten, um durch einen Abgleich mit den Stakeholder-Zielen einen möglichst optimalen Einsatz der Ressourcen zur Qualitätssicherung zu erreichen.
- Unter der Bezeichnung *Value-Based Planning and Control* werden traditionelle Verfahren zur Planung und Steuerung von Kosten, Zeitplänen und Produkten so erweitert, dass die Stakeholder-Interessen systematisch mit einbezogen werden können.
- Das *Value-Based Risk-Management* vereinigt Prinzipien zur Identifikation, Analyse, Priorisierung und Eindämmung von Risiken im Entwicklungsprozess. Hierzu existiert eine Reihe von risikobasierten und agilen Verfahren.
- Im Rahmen des *Value-Based Quality Management* kümmert man sich darum, Qualitätsfaktoren in einem Softwareprojekt mit Rücksicht auf die unterschiedlichen Wertvorstellungen der einzelnen Stakeholder optimal zu priorisieren.
- Unter den Begriff *Value-based People Management* fallen Teambildung für die Stakeholder und Erwartungsmanagement. Damit wird sicherstellt, dass die Wertvorstellungen aller Stakeholder während des gesamten Projekts berücksichtigt werden.

Diese Übersicht verdeutlicht, dass es eine Vielzahl von Bereichen gibt, in denen eine wertebasierte Unterstützung des Softwareprozesses möglich ist. Im Folgenden werden nun einige *Best Practices* vorgestellt, die als illustrierende Beispiele dienen mögen.

Best Practices erlauben Wertorientierung im Softwareprozess

Entsprechend den zuvor genannten Bereichen zeigen die folgenden Beispiele, wie Werteorientierung nun tatsächlich in einen konkreten Softwareprozess eingeführt werden kann.

Best Practice 1:
Analyse der tatsächlichen Vorteile

In seinem Buch „The Information Paradox" (vgl. [Tho99]) stellt John Thorp fest, dass Softwareprojekte oftmals an dem – wie er es nennt – „Field of Dreams"-Syndrom scheitern: In einem amerikanischen Film errichtet ein Bauer irgendwo im mittleren Westen der USA auf seiner Farm ein Baseballfeld in der irrigen Annahme, dass dadurch berühmte Baseballspieler zu einem Besuch oder Spiel veranlasst werden. Übertragen auf Software bedeutet dies: Baue die Software und der Nutzen wird sich schon einstellen.

Mit dem Werkzeug „Analyse der tatsächlichen Vorteile" versucht man, den Überblick über das Verhältnis von Investment und Nutzen zu behalten. Eine *Results-Chain*-Analyse verdeutlicht dabei den Zusammenhang zwischen so genannten Initiativen (diese verbrauchen Ressourcen), Beiträgen (nicht die erstellten Systeme, sondern deren Auswirkungen),

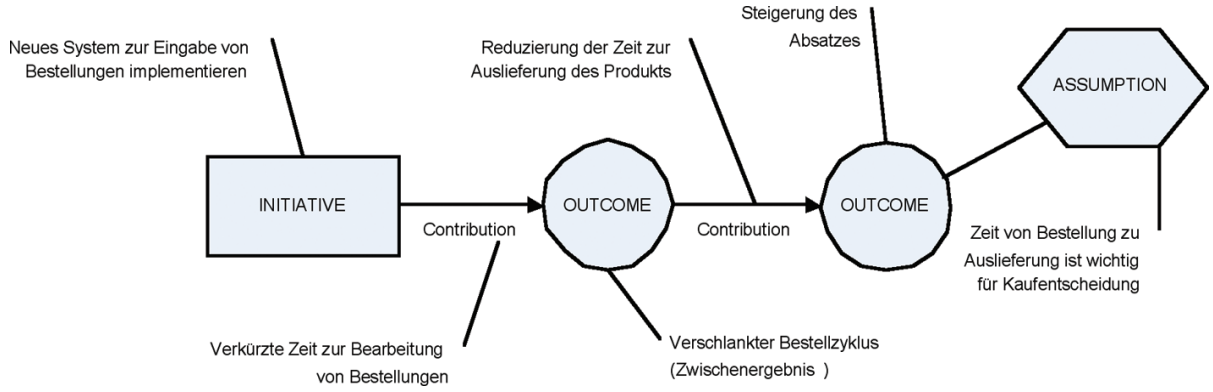

Abb. 1: *Results-Chain-Analyse (Quelle: [Bif06]).*

Ergebnissen (führen zu weiteren Beiträgen oder Mehrwert) und Annahmen.

Die systematische Durchführung dieser *Results-Chain*-Analyse (**wie in Abb. 1 dargestellt**) für verschiedene Aspekte deckt z.B. auf, ob Initiativen fehlen (also Features vergessen wurden) oder ob erfolgskritische Stakeholder nicht berücksichtigt wurden.

Best Practice 2: **Ausarbeitung und Abstimmung von Stakeholder-Interessen**

Die „Analyse der tatsächlichen Vorteile" verschafft unter anderem einen Überblick über wichtige Stakeholder im Softwareprojekt. Da jedoch nicht alle Interessen der einzelnen Stakeholder quantifizier- und vergleichbar sind, wird die Abwägung zwischen einzelnen Features erschwert.

Bei der zweiten Best Practice werden daher die Ergebnisse aus der *Results-Chain*-Analyse für Interviews der Stakeholder genutzt. Die Ergebnisse werden in ein so genanntes Spinnennetz-Diagramm (**siehe Abb. 2**) eingetragen, das entstehende Widersprüche intuitiv visualisiert. Damit wird für alle Stakeholder deutlich, wo widersprüchliche und konkurrierende Ziele im Projekt aufeinanderprallen. Etwas Verhandlungsgeschick erlaubt es, für die Konflikte *Win-Win*-Vereinbarungen zu finden. Wichtig ist es, die Ergebnisse in einem gemeinsamen Visionsdokument festzuhalten. Dieses umfasst eine High-Level Projekt- und eine Ergebnisbeschreibung, eine Liste der erfolgskritischen Stakeholder, ein Blockdiagramm des Systems (Geltungsbereich und Grenzen) sowie eine Auflistung der wesentlichen Projektrahmenbedingungen.

Best Practice 3:

Business-Case-Analyse

Business-Cases werden auch heute für jedes Projekt gerechnet, oftmals so lange, bis das Projekt *sich* rechnet. Ziel dieser Best Practice ist, durch ein strukturiertes Vorgehen zu besseren Business-Cases zu kommen und damit vor allem zu einem besseren Verständnis der wirtschaftlichen Implikationen.

Grundlegend sollte ein initialer Business-Case zügig erstellt werden, indem er z. B. aus einem vergleichbaren Projekt abgeleitet wird. In diesem werden die relativen finanziellen Kosten, der Nutzen und letztendlich der ROI über die Dauer des gesamten Lebenszyklus des zu bauenden Systems ermittelt. Wichtig ist hierbei die Betrachtung eines diskontierten Geldflusses.

Verschiedene Optionen zur Realisierung des Systems sollten im Business-Case miteinander verglichen werden. Dazu wird für die unterschiedlichen Optionen der Verlauf des ROI über die Zeit ermittelt und zusätzlich werden qualitative Verbesserungen, Unsicherheiten und Risiken der betrachteten Optionen aufgelistet. Wird diese Bewertung iterativ mit den Stakeholdern durchgeführt, ergibt sich der positive Zusatznutzen, dass bei diesen eine gemeinsame Einschätzung der Optionen sowie der wirtschaftlichen Implikationen erzielt wird.

Best Practice 4:

Kontinuierliches Management von Risiken und Alternativen

Obwohl naheliegend, werden sie in der Projektpraxis häufig übersehen: die Risiken. Auch bei der Anwendung der hier vorgestellten Best Practices gilt: Die Unsicherheiten in der *Results-Chain*-Analyse und dem Business-Case sind Risikoquellen.

Die wertebasierte Softwareentwicklung sieht daher vor, solche Risiken durch gelebte projektbegleitende Maßnahmen zu eliminieren. Solche Maßnahmen können Prototyping oder Nutzerumfragen sowie eine *COTS*-Evaluierung (*Commercial of the Shelf*) sein und führen in der Regel zur Entwicklung von Ausweichplänen. Im Rahmen der wertebasierten Softwareentwicklung wird zudem vorgeschlagen, eine Fokusperson zur Überwachung von Technologien und Märkten zu installieren, die sich der Identifikation von neuen Risiken und Chancen verschreibt, diese in die Projekte einbringt und damit sowohl der sicheren Projektabwicklung als auch der Rolle der IT als treibende Kraft Rechnung trägt.

Best Practice 5:

Konkurrierendes System- and Software-Engineering

Die fünfte Best Practice dürfte für viele IT-Entscheider als bereits umgesetzt gelten: Die zunehmende Veränderungsgeschwindigkeit in der Informationstechnologie verlangt nach agilen Methoden zur Entwicklung von Software. Das bedeutet, dass sequenzielle Vorgehensweisen zunehmend zum Risiko werden. Schließlich arbeiten Software- und Systementwickler parallel, wenn sie Anforderungen, Architekturen, Lebenszyklus-Pläne usw. entwickeln. Abhilfe schafft hier die Anwendung passender Prozesse, wie z. B. das Spiralmodell oder der „Rational Unified Process" (RUP).

Best Practice 6:

Projektüberwachung und -steuerung

Wenn es darum geht, den Projektfortschritt zu überwachen und zu steuern, wird üblicherweise in *CMM (Capability Maturity Model)* und *CMMI (Capability Maturity Model Integration)* das *Earned Value Management* vorgeschlagen, mit dessen Hilfe ersichtlich wird, wie gut ein Projektverlauf mit der ursprünglichen Planung

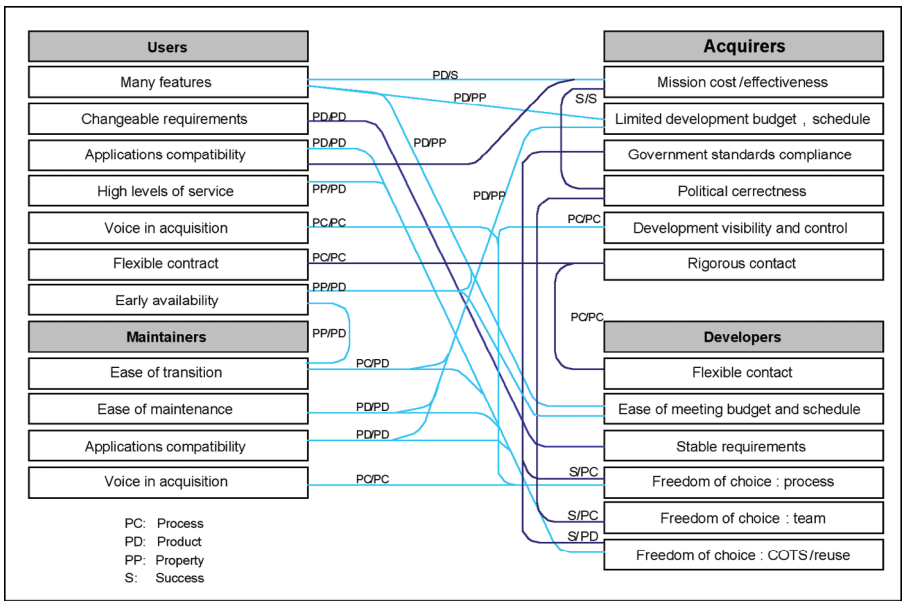

Abb. 2: Spinnennetz-Diagramm (Quelle: [Bif06]).

übereinstimmt. Eine solche Bewertung wird jedoch immer schwieriger, je häufiger sich die Planung ändert. Und noch problematischer ist, dass die Ergebnisse nichts über den Wertbeitrag des Projekts für Organisation bzw. Unternehmen aussagen.

Best Practice 6 schlägt daher vor, den Business-Case sowie die realisierten Stakeholder-Vorteile zu überwachen und zur Steuerung heranzuziehen. Dies geschieht durch ein kontinuierliches Überprüfen der Gültigkeit von Annahmen, die in der *Results-Chain*-Analyse gemacht wurden, sowie durch eine kontinuierliche Überprüfen der aufgelaufenen Kosten. Diese Ergebnisse werden verwendet, um Korrektivmaßnahmen festzulegen. Zusätzlich schlägt die Methode vor, die gewonnenen Erfahrungen zu sammeln und für andere Projekte zur Verfügung zu stellen.

Best Practice 7:
Veränderung als Chance

Die letzte Best Practice in dieser Aufzählung ist mehr ein Anstoß als eine konkrete Maßnahme: Begreifen Sie Änderungen als Chance, nicht als Risiko! Änderungsanforderungen werden immer existieren und mengenmäßig sogar zunehmen. Für verschiedene Typen von Änderungen sollten auch verschiedene Vorgehensweisen eingesetzt werden.

Eine praktische Verinnerlichung dieser Erkenntnis kann zum Beispiel bedeuten, in kleineren Projekten auf agile Methoden zu setzen. Bei größeren Projekten kann es sinnvoll sein, die häufigen Quellen für Anforderungsänderungen zu lokalisieren und das System dann entsprechend so zu modularisieren, dass Änderungen aus dieser Quelle möglichst nur lokale Auswirkungen haben.

Fazit

Die wertebasierte Softwareentwicklung enthält viele sinnvolle Methoden. Einige von diesen sind mittlerweile zwar State-of-the-Art, andere sind jedoch neu. Alles in allem sind die vorgeschlagenen Methoden mit wenig (wenn überhaupt) Zusatzaufwand umzusetzen und versprechen dabei eine wesentliche Verbesserung des Softwareprozesses und intensivieren die Zusammenarbeit aller Stakeholder eines Softwareprojekts. Allein aus diesem Grund ist es lohnenswert, die Methoden – wo nötig – in den individuellen Softwareprozess zu integrieren. ■

Literatur

[Boe00] B. Boehm, K. Sullivan, Software Economics: A Roadmap, in: A. Finkelstein (Hrsg.), Proc. of the Conf. on The Future of Software Economics, ACM Press, New York 2000

[Bif06] S. Biffl et al., Value-Based Software Engineering, Springer, 2006

[Cle02] P. Clements, R. Kazman, M. Klein, Evaluating Software Architecture: Methods and Case Studies, Addison-Wesley, 2002

[Tho99] J. Thorp, The Information Paradox, McGraw-Hill, 1999