

SCHNELLER, KOSTEN- GÜNSTIGER, SCHLANKER: PROZESSOPTIMIERUNG MIT WERTSTROMANALYSE

Wasserfall, V-Modell, Agilität, RUP, XP und Scrum. Die Diskussionen und Grabenkämpfe in der Softwareentwicklung um den richtigen und wahren Prozess sind lang und tief. Unabhängig davon, welchen Prozess Ihre Entwicklungsorganisation lebt: Wissen Sie, welche Entwicklungsaktivitäten und – artefakte wirklich wertschöpfend sind? Kennen Sie das wirtschaftliche Optimierungspotenzial Ihres Softwareentwicklungsprozesses? Der Artikel untersucht den Einsatz einer bewährten Methode zur schnellen und kostengünstigen Ermittlung von Prozessoptimierungspotenzialen in der Softwareentwicklung: die Wertstromanalyse. Die Anwendung dieser Methode erlaubt es, Entwicklungszeiten zu reduzieren, Kosten zu senken und damit die Rentabilität der Softwareentwicklung zu erhöhen.

Wertströme und die Prinzipien schlanker Softwareentwicklung

Wertströme

Unter einem Wertstrom versteht man alle Aktivitäten (sowohl wertschöpfend als auch nicht-wertschöpfend), die dazu dienen, ein Produkt herzustellen und an den Kunden auszuliefern. **Abbildung 1** illustriert diesen Zusammenhang: Die Kundenbedürfnisse werden durch Entwicklungs- und Produktionsaktivitäten in ein Produkt umgewandelt, das der Kunde zur Befriedigung seiner Bedürfnisse käuflich erwirbt.

Im Rahmen dieses Artikels lege ich den Schwerpunkt auf den Softwareentwicklungsstrom, der alle Aktivitäten vom Produktkonzept bis zum Produktionsstart, also von der Anforderungserfassung bis hin zum Systemtest, umfasst.

Schlanke Softwareentwicklung

Schlanke Softwareentwicklung wendet bewährte Methoden der schlanken Produktentwicklung auf die Entwicklung von Software an. Sie lässt sich als neuester Trend in der Softwareprozess-Evolution beschreiben. Dies veranschaulicht **Abbildung 2**. (Die Zuordnung der Trends zu Jahreszahlen erhebt keinen Anspruch auf Genauigkeit. Beachten Sie auch, dass Trends länger existieren können als in der Abbildung dargestellt.)

Schlanke Softwareentwicklung ist gekennzeichnet durch eine radikale Kundenorientierung sowie die Eliminierung von Verschwendung – und damit einer deutlichen Reduzierung von Entwicklungszeit und -kosten. Um dies zu erreichen, werden fünf zentrale Prinzipien angewandt (vgl. [Wom96]):

- **Value:** Das Softwareprodukt ist ganz an den Kundenbedürfnissen ausgerichtet

der autor



Roman Pichler
(E-Mail: roman.pichler@siemens.com)
unterstützt als Senior Consultant der Siemens AG Siemens-Bereiche bei der Optimierung ihrer Entwicklungsprozesse.

tet und schafft genau den Wert, den der Kunde wünscht – und nicht etwa das, was das Marketing oder die Softwareentwicklung als erstrebenswert betrachtet.

- **Value Stream:** Alle Softwareentwicklungsschritte sind wertschöpfend. Stellen Sie sich vor, Sie wären Ihr eigener Kunde. Wären Sie bereit, für jede Entwicklungsaktivität zu bezahlen?
- **Flow:** Alle Entwicklungsschritte – von der Anforderungsermittlung über die Architekturerstellung und den Systemtest – fließen kontinuierlich. Das bedeutet die Eliminierung von Wartezeiten zwischen Aktivitäten sowie die Beseitigung unnötigen Überarbeitens und Nachbesserns der Arbeitsergebnisse.
- **Pull:** Der Kunde ist in der Lage, das Softwareprodukt dann zu erhalten, wenn er es benötigt. Für die Softwareentwicklung bedeutet das, schnell auf Kundenwünsche reagieren zu können und den Bestand an Anforderungen minimal zu halten gemäß dem Motto: „Je mehr Anforderungen vorgehalten werden, desto weniger wahrscheinlich ist es, dass tatsächlich eine Anforderung ein Kundenbedürfnis darstellt.“
- **Perfection:** Der Softwareentwicklungsprozess wird kontinuierlich verbessert, beispielsweise durch die Eliminierung von nicht-wertschöpfenden Aktivitäten, Wartezeiten und unnötigen Nacharbeiten. Das Ziel schlanker Softwareentwicklung ist es also, dem perfekten Prozess möglichst nahe zu kommen: einem Prozess, der Wert für Kunden frei von jeglicher Verschwendung schafft.

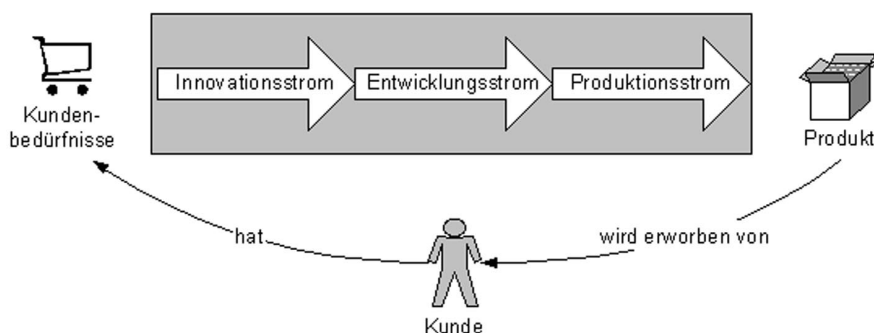


Abb. 1: Kunde, Produkt und ausgewählte Wertströme

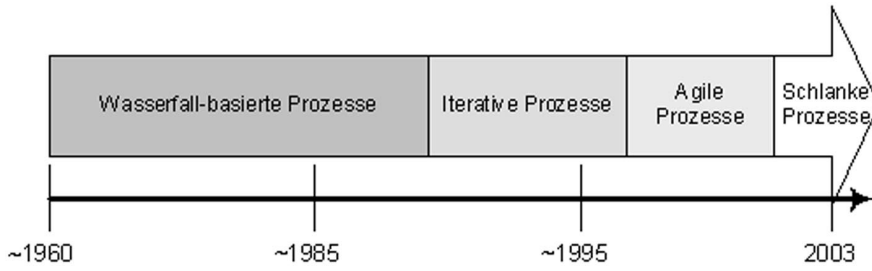


Abb. 2: Softwareprozesstrends

Nutzen der Wertstromanalyse

Die Wertstromanalyse ist eine zentrale Methode schlanker Produktentwicklung. Sie hilft Ihnen, Ihre Wertschöpfungskette und Ihren Wertfluss zu optimieren und damit die schlanken Prinzipien *Value Stream* und *Flow* in Ihrer Organisation umzusetzen. Dabei ist es zunächst nebensächlich, ob Sie einen wasserfall-basierten oder iterativ-inkrementellen Softwareentwicklungsprozess verwenden. Vorausgesetzt wird allerdings, dass Ihre Prozesse wiederholbar sind¹⁾.

Der volle Nutzen schlanker Produktentwicklung erschließt sich Ihnen allerdings erst dann, wenn Sie konsequent alle fünf Prinzipien umsetzen und über die Softwareentwicklung hinaus schlankes Denken in Ihrem Unternehmen etablieren.

Wertstromanalyse in der Softwareentwicklung

Im Folgenden stelle ich die Anwendung der Wertstromanalyse in der Softwareentwicklung vor. Zunächst erfasse ich den Ist-Zustand des Entwicklungsprozesses und

erstelle dann den Soll-Zustand. In einem dritten Schritt leite ich geeignete Maßnahmen zur Überführung des Ist- in den Soll-Zustand und damit zur Prozessoptimierung ab.

Der erste Schritt:

Ermittlung des Ist-Zustands

Eine Wertstromanalyse beschreibt zuallererst den real gelebten Softwareentwicklungsprozess in Form eines Wertstromplans (vgl. [Rot99] und [Wom]). Auch wenn Sie über einen wohl dokumentierten Prozess verfügen, lohnt sich die Mühe: Gelebter und dokumentierter Prozess divergieren häufig. Sprechen Sie daher mit allen an der Softwareentwicklung Beteiligten an ihren Arbeitsplätzen und nehmen Sie alle ausgeführten Aktivitäten auf, von der Kundenanforderungsermittlung bis zum Systemtest. Eine ganzheitliche Betrachtung des Entwicklungsstromes ist unerlässlich, um Verschwendung adäquat zu identifizieren und die richtigen Maßnahmen zu ihrer Beseitigung zu ergreifen.

Wählen Sie ein aktuelles, repräsentatives Entwicklungsprojekt aus. Bestehen Sie darauf, dass alle wichtigen *Stakeholder* der Prozessoptimierung gemeinsam mit Ihnen die Interviews führen und den Wertstrom

der Software aufzeichnen. Dies stellt sicher, dass alle *Stakeholder* ein gemeinsames Verständnis des Wertstroms entwickeln.

Beginnen Sie Ihre Reise entlang des Wertstroms dort, wo die Software ausgeliefert wird, und arbeiten sich dann schrittweise stromaufwärts bis zur Anforderungserfassung vor. Dies ermöglicht es Ihnen, eine Kundenperspektive einzunehmen und die erzielten Ergebnisse mit den jeweiligen Arbeitsschritten gezielt zu betrachten. Ermitteln Sie sowohl den Materialfluss (also den Weg, den Anforderungen, Spezifikationen, Designs, Code etc. nehmen), als auch den Informationsfluss. Letzterer dient der Ermittlung der Koordination der Entwicklungsaktivitäten. Fragen Sie die an der Entwicklung Beteiligten, woher Sie wissen, wann welche Arbeiten auszuführen sind.

Achten Sie bei der Erfassung der Aktivitäten auf Verschwendung: auf Wartezeiten zwischen den Aktivitäten, mögliche Überlastung von Mitarbeitern sowie die Menge an Verbesserungsarbeiten, die anfallen. Wartezeiten können beispielsweise durch Kapazitätsengpässe, Managemententscheidungen oder Ressourcenallokation verursacht werden (vgl. [Smi98]). Die Überlastung von Mitarbeitern manifestiert sich z. B. durch nachgelagerte Wartezeiten oder die parallele Abarbeitung vieler verschiedener Aufgaben. Verbesserungsarbeiten sind beispielsweise das Überarbeiten einer Anforderungsbeschreibung, die Beseitigung von Defekten oder auch bedingt Refaktorisierungsarbeiten. Befindet sich Ihr Entwicklungsprojekt in einer Phase der Konzeption/Exploration, in der Architektur und Design stabilisiert werden, so ist mit laufenden Codeanpassungen zu rechnen. Häufiges Refaktorisieren ist hier

¹⁾ Dies entspricht einem Prozessreifegrad gemäß dem „Capability Maturity Model“ (CMM) von Level 2 oder höher.

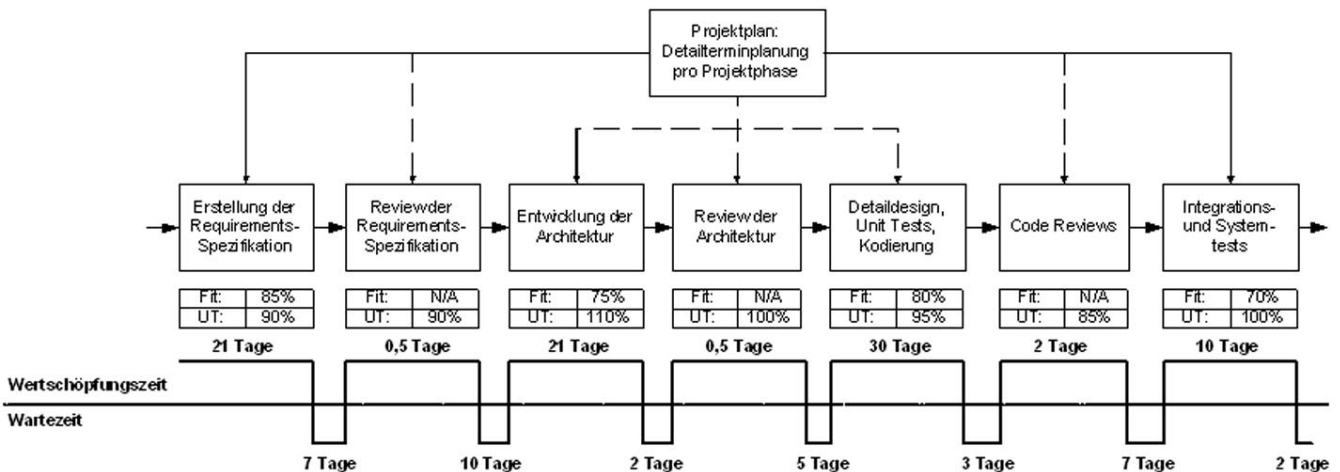


Abb. 3: Wertstromplan des Ist-Zustands



normal und nützlich. Wurde jedoch eine Architektur etabliert, so sollte auch die Anzahl der Nacharbeiten und der Umfang von Refaktorisierungsmaßnahmen sinken. Als Faustregel gilt: Je weiter Sie stromabwärts Verbesserungsarbeiten finden, desto größer ist das Verbesserungspotenzial einzuschätzen.

Notieren Sie die Wertstromaktivitäten auf einem Blatt Papier zusammen mit Ihren Beobachtungen. So entsteht ein Wertstromplan, der den Ist-Zustand repräsentiert. Berücksichtigen Sie dabei stets das Ziel des Wertstromplans: Verschwendung jeglicher Art offen zu legen. Ein Beispiel finden Sie in **Abbildung 3**.

Der Wertstromplan in **Abbildung 3** stellt den Entwicklungsartefakt- und Informationsfluss dar. Ersterer identifiziert die Wertschöpfungs- und Wartezeiten sowie die Auslastung der Mitarbeiter und die Korrektheitsrate der erstellten Arbeitsergebnisse. Unter Wertschöpfungszeit verstehen wir die produktiv für die Softwareentwicklung investierte Zeit. Wartezeiten treten beispielsweise zwischen dem Review der Anforderungsspezifikation und dem Beginn der Architekturentwicklungsarbeiten auf. Ein Grund für die Wartezeit könnten Schwierigkeiten sein, die benötigten Architekten dem Projekt schnell zur Verfügung zu stellen. Vergleichen wir Wertschöpfungs- und Wartezeit in **Abbildung 3**, so fällt auf, dass fast 30% der betrachteten Entwicklungszeit nicht produktiv genutzt wird. Eliminieren wir die Wartezeiten, so können wir die Entwicklungszeit deutlich reduzieren.

Die Abkürzung *UT* (für *Utilization*) im Wertstromplan in **Abbildung 3** zeigt die Auslastung der Mitarbeiter an. So sind beispielsweise die mit der Architekturerstellung beauftragten Mitarbeiter deutlich überlastet. Ein Grund hierfür könnte sein, dass nicht genügend Architekten verfügbar sind bzw. die Architekten ihre Zeit zwischen verschiedenen Projekten aufteilen. Die Überlastung von Mitarbeitern führt zu längeren Entwicklungszeiten. Eliminieren wir Überlastungen, so können wir die Entwicklungszeit weiter reduzieren.

Neben der Mitarbeiterauslastung identifiziert der Wertstromplan auch die Korrektheitsrate *FIT* der erzielten Arbeitsergebnisse. *FIT* beschreibt den Prozentsatz der Arbeitsergebnisse, die ohne Nachbesserungen in nachgelagerte Arbeitsschritte einfließen können. In dem Wertstromplan werden Korrektheitsraten

durch Review-Aktivitäten ermittelt. Beispielsweise werden im Architektur-Review nur 75% der Architektur als adäquat empfunden, 25% müssen damit überarbeitet werden. Durch eine Erhöhung der Korrektheitsrate und die Reduktion von Defekten sparen wir Kosten und Zeit – und erhöhen die Qualität der Arbeitsergebnisse.

Die einzelnen Aktivitäten und Arbeitsschritte werden durch den Projektplan des Softwareentwicklungsprojekts gesteuert. Der Wertstromplan in **Abbildung 3** verdeutlicht dies durch die Darstellung des Informationsflusses: Der Projektplan teilt den Mitarbeitern mit, wann welche Aktivitäten auszuführen sind. Prozessschritte werden also zentral geplant und gesteuert.

Der zweite Schritt:

Ermittlung des Soll-Zustands

Nachdem Sie den Ist-Wertstrom Ihrer Softwareentwicklung untersucht und Verschwendung offen gelegt haben, sind Sie nun in der Lage den optimalen Zielprozess für Ihre Softwareentwicklung zu ermitteln.

Zuvor sollten Sie sich allerdings über Ihre geschäftlichen Ziele und Rentabilitätstreiber im Klaren sein: Welche Art von Verschwendung ist für Ihr Geschäft am problematischsten? Erreichen Sie eine Gewinnsteigerung primär durch eine Senkung der Entwicklungskosten? Oder ist es für Sie zielführender, Ihre Entwicklungszeiten zu reduzieren? Wenn Sie noch unschlüssig sind, empfehlen wir Ihnen, zunächst ein Profitmodell auf der Basis einer Gewinn- und Verlustrechnung zu entwickeln und die Auswirkung der Verringerung von Entwicklungszeit bzw. -kosten auf Ihre Rentabilität durchzurechnen (eine gute Hilfestellung hierzu finden Sie in [Smi98] und [Rei97]). Die Bestimmung der Auswirkungen von Verbesserungsmaßnahmen auf Ihre Rentabilität ist unabdingbar für eine solide Priorisierung von Verbesserungsmaßnahmen im dritten Schritt unserer Wertstromanalyse.

Entwerfen Sie nun einen neuen Wertstromplan, der den Soll-Zustand Ihres Softwareentwicklungsprozesses repräsentiert. Ihr Ziel sollte sein, möglichst alle Quellen von Verschwendung zu eliminieren und die Prozessschritte so zu gestalten, dass alle Aktivitäten kontinuierlich fließen; dies gilt insbesondere für diejenigen Aktivitäten, die auf dem kritischen ▶

1/3
b + m

Pfad Ihres Projekts liegen²⁾. Überprüfen Sie dabei sorgsam, ob die einzelnen Aktivitäten Ihres Ist-Wertstroms wertschöpfend sind.

Abhängig davon, wie Sie Ihre Rentabilität am besten steigern können, sollten Sie entweder Ihren existierenden Prozess schrittweise verbessern (*kaizen*, aus dem Japanischen) oder sich von Ihrem alten Prozess trennen und einen neuen einführen (*kaikaku*, dito). Ist die Einführung eines neuen Prozesses für Sie zielführend, so sollten Sie erwägen, inwieweit die Verwendung eines existierenden Prozessmodells, wie beispielsweise des *Rational Unified Process (RUP)* (vgl. [Kru00]) oder *Extreme Programming (XP)* (vgl. [Bec00]), für Sie geeignet ist. Allerdings sollten Sie auch bei der Anpassung von RUP und XP die Ziele schlanker Softwareentwicklung verfolgen, um Ihre Rentabilität zu maximieren.

Nehmen wir im Folgenden an, Sie führen einen neuen, auf Ihre Organisation zugeschnittenen Prozess ein. Ihr Soll-Wertstromplan könnte beispielsweise so wie in **Abbildung 4** dargestellt aussehen. Der dort gezeigte Wertstrom wurde im Vergleich zum Ist-Wertstromplan in **Abbildung 3** spürbar verschlankt: Integrations- und Systemtests wurden beispielsweise in Architekturerstellung- und Konstruktionsaktivitäten parallelisiert. Wartezeiten wurden weitestgehend eliminiert – anstelle von 36 Tagen betragen diese nun einen Tag. Die Entwicklungszeit wurde von 85 auf 62 Tage reduziert. Ihre Softwareentwicklung wurde also um 27% beschleunigt. Der Informationsfluss wird nicht länger ausschließlich zentral in Form des Projektplans gesteuert, sondern durch kurze tägliche Statusrunden (*Stand-up Meetings* nach [Bee98]) unterstützt.

Der dritte Schritt:

Ableitung von Maßnahmen

Um von Ihrem Ist-Wertstrom zu dem ermittelten Soll-Prozess zu gelangen, leiten Sie nun geeignete Maßnahmen ab. Beziehen Sie bei der Maßnahmenauswahl alle wichtigen *Stakeholder* und Informationslieferanten mit ein. So stellen Sie

sicher, dass Sie keine guten Ideen und Anregungen übersehen und sich die Akzeptanz der Mitarbeiter sichern, die von den Prozessveränderungen am stärksten betroffen sind. Ich empfehle Ihnen, sich bei der Ableitung von Maßnahmen Anregungen z. B. bei [Col03], [Smi98] und [Rei97] zu holen.

Die nachfolgend aufgeführten Maßnahmen haben Beispielcharakter und dienen der Illustration. Meist existieren mehrere Lösungsalternativen, um ein Element Ihres Soll-Wertplans zu realisieren. Ermitteln und bewerten Sie alle Alternativen sorgsam. Denken Sie daran, dass sich die Maßnahmen stets an Ihren Geschäftszielen orientieren und dabei Ihren spezifischen Organisationskontext berücksichtigen sollten.

- **Bildung eines dedizierten interdisziplinären Projektteams:** Eine Verbesserung des Aktivitätsflusses und eine Reduzierung der Entwicklungszeit erreichen Sie beispielsweise durch die Einbeziehung aller *Stakeholder* in die Beschreibung der Anforderungen. Binden Sie neben Marketingvertretern auch Kunden, Entwicklung, Fertigung, Qualitätssicherung, Vertrieb und Service von Beginn an in die Erstellung der Anforderungsspezifikation ein. Ziel sollte dabei sein sicherzustellen, dass alle an der Entwicklung Beteiligten eine klare und gemeinsame Vorstellung des Produkts entwickeln. Dies erlaubt es Ihnen, die Anforderungsspezifikation weniger umfangreich zu beschreiben, flexibler auf Änderungswünsche zu reagieren und auf zeitintensive Reviews und Abstimmungsrunden zu verzichten. Stellen Sie außerdem sicher, dass alle wichtigen Projektmitarbeiter dem Projekt während seiner Gesamtlaufzeit gänzlich zur Verfügung stehen. Überlastung von Mitarbeitern und Zeitverluste beim Allokieren von Ressourcen sind oftmals teurer als eine suboptimale Auslastung.
- **Parallelisierung von Entwicklungs- und Testaktivitäten:** Ist Ihr Testteam im Dauerstress und häufig überlastet? Testteams haben bei einer sequenziellen Softwareentwicklung die undankbare Aufgabe, einen riesigen Berg von Integrations- und Systemtests in möglichst kurzer Zeit abzuarbeiten. Dies führt in der Regel zu Kapazitätsengpässen und damit zu Wartezeiten. Sie vermeiden eine Überlastung Ihres

Testteams, eliminieren Wartezeiten und reduzieren Entwicklungszeit, indem Sie Applikationsbestandteile kontinuierlich an den Test übergeben und Entwicklungs- und den Testaktivitäten überlappen. Zusätzlich erhalten Ihre Entwickler schneller und häufiger Rückmeldung über die Qualität ihrer Arbeit. Defekte können so zeitnäher, schneller und kostengünstiger beseitigt werden. Wichtige Erfolgsfaktoren für eine Parallelisierung von Entwicklungs- und Testaktivitäten sind eine adäquate Softwarearchitektur, eine frühzeitige und regelmäßige Stabilisierung von Softwarebausteinen (z. B. durch Unit-Tests) sowie eine gute Zusammenarbeit von Entwicklern und Testern.

- **Verbesserung des Informationsflusses:** Sie verbessern den Informationsfluss, indem Sie die betroffenen Mitarbeiter aktiv in die Projektplanung einbeziehen und zusätzlich den Informationsaustausch zwischen den Projektmitgliedern fördern, beispielsweise durch kurze, zielorientierte, tägliche *Stand-up Meetings*. Wenn Ihnen der Zeitaufwand für tägliche Besprechungen zu hoch erscheint, empfehle ich Ihnen: Berechnen Sie die Kosten, die Ihnen durch Informationsverlust oder -verzögerung entstehen können. Wenn Sie *Stand-up Meetings* abhalten, sollten Sie das Ziel der Besprechungen klar vermitteln und die Meetings möglichst kurz halten. Als Faustregel gilt: Die Dauer sollte maximal 15 Minuten betragen. Tägliche *Stand-up Meetings* eignen sich vorzüglich, um Rückmeldung über den Projektstatus zu erhalten, neue Risiken zu identifizieren und die Überlastung von Mitarbeitern zu erkennen. Visualisieren Sie zusätzlich den Projektfortschritt und bringen Sie die Visualisierung so an, dass sie für alle Projektmitarbeiter sichtbar ist. Achten Sie darauf, dass die Visualisierung des Projektfortschritts stets aktuell ist, z.B. indem Sie die Projektmitglieder bitten, den Status Ihrer Arbeitspakete fortlaufend zu aktualisieren.

Berechnen Sie die Auswirkung der ermittelten Maßnahmen auf die Rentabilität Ihrer Organisation und priorisieren Sie die Maßnahmen entsprechend. Machen Sie sich auch die Kosten für die Umsetzung der Maßnahmen bewusst und stellen Sie

²⁾ Der kritische Pfad entspricht der Sequenz von Aktivitäten, die die kürzestmögliche Dauer Ihres Projektes darstellt. Reduzieren Sie die Dauer einer Aktivität auf dem kritischen Pfad, so verkürzt sich auch die Projektlaufzeit.



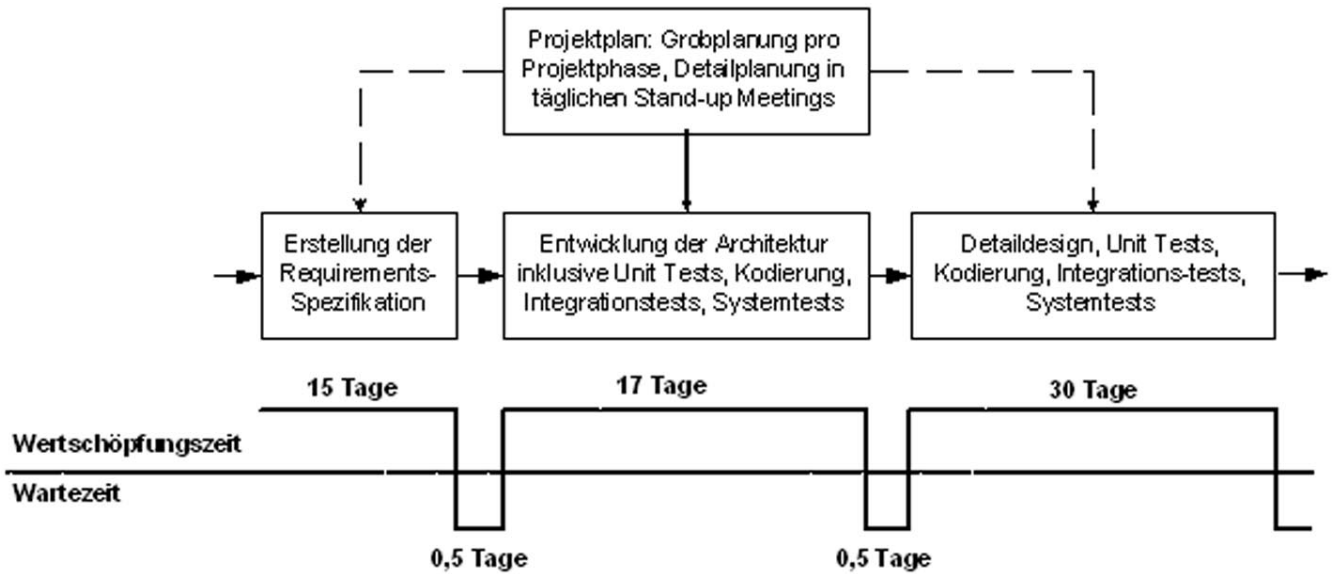


Abb. 4: Wertstromplan des Soll-Zustands

eine maßnahmenbezogene Kosten/Nutzen-Betrachtung an.

Sichern Sie sich die volle Unterstützung Ihres Managements und setzen Sie die Maßnahmen mit hoher Priorität zügig und konsequent um. Als Faustregel gilt: Erste positive Auswirkungen der Prozessoptimierung sollten sich nach circa drei Monaten einstellen. Ist dies nicht der Fall, so sollten Sie die Prozessoptimierungsmaßnahmen aussetzen und Ursachenforschung betreiben: Wurden die Maßnahmen nicht adäquat umgesetzt? Oder sind die Maßnahmen selbst inadäquat?

Der vierte Schritt:

kontinuierliche Verbesserung

Die Eliminierung nicht-wertschöpfender Aktivitäten und die Einführung eines kontinuierlichen Aktivitätsflusses sind die ersten Schritte auf dem Weg zu einem perfekten Softwareentwicklungsprozess: einem Prozess der mehr Wert für den Kunden mit weniger Ressourcen und frei von jeglicher Verschwendung schafft und somit die Rentabilität erhöht.

Um Ihren Wertstrom vor Gewinzzunahme zu schützen und weiter zu verschlanken, sollten Sie es wie mit jeder Schlankheitskur halten: Haben Sie erst einmal Ihre Ernährung umgestellt und den Soll-Wertstrom realisiert, müssen Sie ständig darauf achten, nicht wieder zuzunehmen, Ihre Diät weiter anzupassen und zu optimieren.

Um dies zu erreichen, sollten Sie eine Kultur der kontinuierlichen Verbesserung und Veränderung in Ihrer Organisation etablieren, die das kritische Hinterfragen aller Aktivitäten bezüglich ihrer Wertschöpfung fördert.

Zusammenfassung

Wertstromanalysen helfen Ihnen, Verschwendung bei der Softwareentwicklung zu eliminieren, einen kontinuierlichen Aktivitätsfluss herbeizuführen und die Rentabilität Ihrer Organisation zu steigern. Schließlich verfolgen alle kommerziellen Softwareentwicklungsprojekte ein Ziel: möglichst viel Gewinn zu erwirtschaften. ■

Literatur & Links

- [Bec00] K. Beck, Extreme Programming Explained, Addison-Wesley 2000
- [Bee98] M. Beedle, M. Devos, Y. Sharon, K. Schwaber, J. Sutherland, SCRUM: An extension pattern language for hyperproductive software development, in: Proc. of Pattern Language of Programs (PloP), 1998
- [Col03] J. Coldewey, M. Poppendieck, Fastenkur für den Prozess: „Lean Development“ (Teil 1), in: OBJEKTSpektrum 3/03
- [Kru00] P. Kruchten, The Rational Unified Process: An Introduction (2. Ausgabe), Addison-Wesley 2000
- [Rei97] D.G. Reinertsen, Managing the Design Factory. A Product Developer's Toolkit, Free Press 1997
- [Rot99] M. Rother, J. Shook, Learning to See, Version 1.2, Lean Enterprises Inst. 1999
- [Smi98] P.G. Smith, D.G. Reinertsen, Developing Products in Half the Time (2. Ausgabe), Wiley 1998
- [Wom96] J.P. Womack, D.T. Jones, Lean Thinking, Touchstone Books, 1996
- [Wom] J. Womack (with S. Withers), The NAM Articles (siehe: <http://www.lean.org/Lean/Community/Resources/ThinkersCorner.cfm>)

1/6
gebit