mehr zum thema:
www.microsoft.com/germany/visualstudio/
www.hood-group.com/dienstleistungen/agile-methoden/practices/

NAVIGATIONSBESTECK FÜR WORKFLOWS: TEIL I: ARBEITSABLÄUFE IN DER SOFTWAREENTWICKLUNG OPTIMIEREN

Produktentwicklung ist teuer. Gerade in wirtschaftlich schwierigen Zeiten müssen Unternehmen auch hier die Kosten senken. Gerne greift man dann zur Rasenmäher-Methode. Doch diese unterscheidet nicht zwischen nützlichen Halmen und Unkraut. Eine intelligente Alternative ist die Fokussierung auf wertschöpfende Tätigkeiten. „Value-oriented Practices" (v-o-p) liefern das Handwerkszeug zur transparenten und an Wertbeiträgen ausgerichteten Gestaltung von Prozessen. Auch komplexe Arbeitsabläufe in der Softwareentwicklung lassen sich damit nachhaltig verstehen, verschlanken und automatisieren. Im ersten Teil des Artikels erläutern wir, wie v-o-p eine Brücke zwischen Geschäftsprozessen und der täglichen Arbeit in der Entwicklung schlägt.

Erinnern Sie sich noch an Ihren letzten Restaurantbesuch und an den Kellner, der Sie bediente? Wenn Sie seine Tätigkeit beschreiben sollten, dann können Sie es mit der herkömmlichen Geschäftsprozessmodellierung versuchen, also einer ablauforientierten Methode. Ein mögliches Ergebnis zeigt Abbildung 1. Diese Beispiellösung beschreibt die einzelnen Arbeitsabläufe chronologisch. Jeder Kellner kann seine Arbeit entsprechend organisieren. Seine Aktivitäten sind stets gleich. Es gibt nur wenige Randbedingungen, die eine Änderung dieses Ablaufs erfordern – beispielsweise, wenn sich Gäste übrig gebliebene Speisen zum Mitnehmen einpacken lassen. Wie sieht es dagegen mit dem Arbeitsablauf des Kochs aus? Mit der ablauforientierten Methode kommen wir zu einer Lösung, wie sie in Abbildung 2 dargestellt ist.

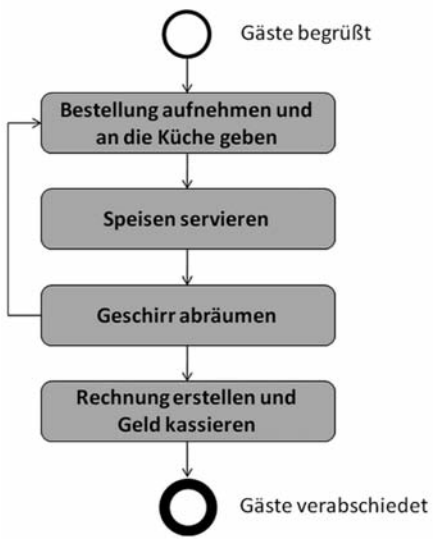

Abb. 1: Was tut ein Kellner?

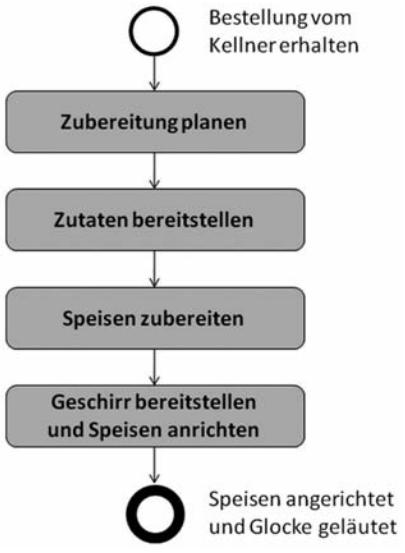

Abb. 2: Was tut ein Koch?

Kreative Arbeitsabläufe

Im Unterschied zu Abbildung 1 stellt dieses Schaubild nicht wie beim Kellner den genauen chronologischen Arbeitsablauf dar. Vielmehr verbergen sich hinter den einzelnen Schritten vielschichtige Tätigkeiten: Vom Zutaten Abmessen, Zerkleinern und Mischen, über das Kochen, Braten, Backen, Würzen und Abschmecken – bis hin zu den letzten Finessen beim Anrichten der Speisen. Schon die Zahl und die Beschaffenheit der verwendeten Utensilien beeindruckt.

Die genaue Abfolge der Zubereitung richtet sich nach der Reihenfolge der Bestellungen durch die Gäste. Dadurch entstehen für die Arbeitsschritte der einzelnen Rezepte verschiedene Synergien. Zum Beispiel werden – je nach Bestelleingang – bestimmte Zutaten für mehrere Rezepte auf einmal geschnitten, gekocht, gebraten usw. Der Workflow des Kochs ergibt sich damit aus einer optimierten Abfolge von Arbeitsschritten für gleichzeitig zubereitete Speisen. Die Optimierung kann erst erfolgen, nachdem die Bestellungen eingegangen sind. Zusätzlich sollen die Gerichte für jeden Tisch möglichst gleichzeitig fertig sein. Je nach Bestelleingang – und damit mehrmals im Laufe eines Tages bzw. Abends – ändert der Koch die Reihenfolge und Organisation seiner Tätigkeiten. Welche Bestellungen wann eingehen, lässt sich nicht vorhersagen, und damit erübrigt sich eine allgemeingültige Beschreibung der chronologischen Arbeitsabläufe. Dennoch ist eine konkrete Prozessbeschreibung nützlich, um etwa Schnittstellen zu benachbarten Teilprozessen festzulegen. Bevor wir auf passende Methoden eingehen, greifen wir noch einmal das Beispiel des Kellners im betriebswirtschaftlichen Umfeld auf.

Im Rechnungswesen finden wir stabile, durch Buchführungsvorschriften weitgehend geregelte Workflows. Hier eine typische Sequenz:

die autoren

Jens Donig
(E-Mail: Jens.Donig@HOOD-Group.com)
ist Senior Consultant für Systems-Engineering bei der HOOD Group. Seine Schwerpunkte liegen im Bereich Softwareentwicklungsprozesse und Requirements-Engineering.

Dr. Martin Künzle
(E-Mail: Martin.Kuenzle@siemens.com)
verantwortet das Kompetenzfeld „TFS Services" bei der evosoft GmbH und unterstützt die Einführung von integriertem ALM im Siemens-Konzern.

First Class Citizens (FCCs) (ursprünglich ein Begriff aus dem Bereich Programmier-Paradigmen) sind quasie „Bürger erster Klasse" und bezeichnen die Elemente einer Notation (Sprache), mit denen man erkennt, welchem Paradigma die Notation gehorcht. Hier ein paar Beispiele – übertragen auf Notationen der Geschäftsprozessmodellierung:

- ARIS-Methode – Ereignisgesteuerte Prozesskette (EPK): FCCs: Ereignisse, Funktionen
- UML – Aktivitätsdiagramm: FCCs: Aktionen, Aktivitäten
- BPMN – Geschäftsprozessdiagramm: FCCs: Ereignisse, Aktivitäten, Gateways

Kasten 1: *Beispiele für FCCs.*

- Rechnungseingang,
- Kontierung als Verbindlichkeit
- Auslösen des Zahlungsvorgangs
- Buchung als Zahlungsausgang
- Kontierung auf Kostenstelle

Die Abarbeitungsreihenfolge ist fix, es gibt keinen Spielraum für eine Parallelisierung. Für solche Vorgänge eignen sich ablauforientierte Beschreibungsmethoden wie *Ereignisgesteuerte Prozessketten (EPK)* (siehe Kasten 1). Das Beispiel des Kochs lässt sich im technischen Bereich auf die Entwicklung von Software übertragen. Hier stoßen wir auf hochgradig vernetzte Tätigkeiten, die durch komplexe Abhängigkeiten und Nebenläufigkeiten geprägt sind. Dazu kommen vielschichtige und variable Rahmenbedingungen, wie beispielsweise:

- Erfahrungen und Kompetenzen des Projektteams
- Entwicklungsumgebung und IT-Infrastruktur
- Änderungen im Zeitplan und der verfügbaren Ressourcen
- Neuer Anforderungen während der Projektlaufzeit

Die Wertschöpfung im Projektalltag wird durch die Abläufe und den Automatisierungsgrad im *Application Lifecycle Management (ALM)* geprägt (siehe Kasten 2). Bisher fehlte die Methodik, solche semi-strukturierten Geschäftsprozesse für Wissensarbeiter angemessen darzustellen. Häufig beschränkt sich die Prozessmodellierung auf eine Abfolge grobgranularer Produktionsschritte nach dem *EVA*-Muster (*Eingabe, Verarbeitung, Ausgabe*). ALM-Umgebungen, wie z. B. „Microsoft Team Foundation Server" oder „IBM Rational Team Concert" – bilden aber viel feiner aufgelöste Workflows ab, bei denen sämtliche fachlichen Abhängigkeiten berücksichtigt werden müssen.

In diesem Artikel stellen wir die *Value-oriented Practices (v-o-p)* vor, mit denen sich nicht-sequenzielle Abläufe gestalten und visualisieren lassen. v-o-p stellt die Wertschöpfung in den Mittelpunkt und berücksichtigt Abhängigkeiten, ohne die zeitliche Reihenfolge der Arbeitsabläufe festzuschreiben. In einem Softwareentwicklungprojekt der Healthcare-Industrie wurde v-o-p bereits erfolgreich eingesetzt. Im Folgenden beschreiben wir die v-o-p-Methode und unsere bisher damit gesammelten Erfahrungen.

Wertschöpfung in der Softwareentwicklung

Moderne Werkzeuge ermöglichen eine weitgehende Automatisierung und Verflechtung von Arbeitsabläufen im ALM. Eine besondere Herausforderung liegt in der präzisen und gleichzeitig flexiblen Gestaltung dieser Arbeitsabläufe, sodass agile, iterative, inkrementelle und weitere Vorgehensmuster in der Entwicklung unterstützt werden und stets eine hohe Ergebnisqualität sichergestellt werden kann. Für die optimale Wertschöpfung im Tagesgeschäft (Mikroprozesse) ist das passende Design der Prozessvorlagen entscheidend. Damit wird bestimmt, ob deren effiziente Verwendung möglich ist. Im Folgenden sind die Elemente einer Prozessvorlage dargestellt (vgl. auch [MSVS]):

- Work-Item-Typen (WIT)
- WIT-Beziehungen
- WIT-Felder
- WIT-Workflows
- WIT-Regeln
- WIT-Formulare
- Abfragen

Application Lifecycle Management (ALM) zielt auf eine Verzahnung von Anwendungsentwicklung und Anwendungsbetrieb. Dabei deckt ALM den gesamten Lebenszyklus einer Applikation ab: von der Idee bis zur Ablösung.

Kasten 2: *ALM-Definition.*

- Arbeitsmappen
- Berichte
- Dashboards
- Anleitungen

Bei der Gestaltung von Prozessvorlagen muss beispielsweise darauf geachtet werden, dass Felder und Workflows leicht verständlich benannt sind. Redundante, mehrdeutige und widersprüchliche Felder und Feldwerte müssen vermieden werden. Das gleiche gilt für Zustände und Zustandsübergänge der Workflows. Die Anwender arbeiten meist täglich mit diesen Prozessvorlagen. Je mehr sich die Nutzer dabei auf ihre fachlichen Aufgaben konzentrieren können und beispielsweise nicht rätseln müssen, welche Bedeutung welche Feldwerte haben, desto effizienter fallen die Arbeitsergebnisse aus.

Die semantische Lücke zwischen Makro- und Mikroprozessen

Makroprozesse stellen grundsätzliche Geschäftsabläufe dar. Sie beschreiben sehr allgemein (makroskopisch), welche Aktivitäten wann zu welchen Ergebnissen führen müssen. Eine Aktivität fasst dabei komplexe Entwicklungstätigkeiten zusammen, beispielsweise „Define System Requirements" oder „Plan Iteration". Die Mikroprozesse, also die täglichen Arbeitsabläufe, werden in ALM-Systemen durch Prozessvorlagen abgebildet. Diese Prozessvorlagen beschreiben die ALM-Artefakte mit ihren Eigenschaften (Feldern) und ihrer Dynamik (Workflows). Typische Domänenelemente sind „Anforderung", „Softwarefehler" oder „Testfall". Problematisch ist der fehlende Bezug zwischen den ablauforientierten Makroprozessen und den entsprechenden Prozessvorlagen. Aus den Makroprozessen allein lässt sich nicht systematisch ableiten, wann und warum welche Prozessvorlagen notwendig sind. Gründe dafür sind unterschiedlich verdichtete Informationen (makro vs. mikro) und verschiedene Sichten (ablauforientiert vs. datenorientiert). Beides führt zu der semantischen Lücke, die in Abbildung 3 dargestellt ist. Die Verknüpfung von Makro- und Mikroprozessen ist eine wichtige Voraussetzung, damit Entwicklungsaktivitäten systematisch optimiert werden können.

Die Schwäche der ablauforientierten Sicht

Viele in der Praxis verwendete Notationen für die Modellierung von *Geschäftspro-*

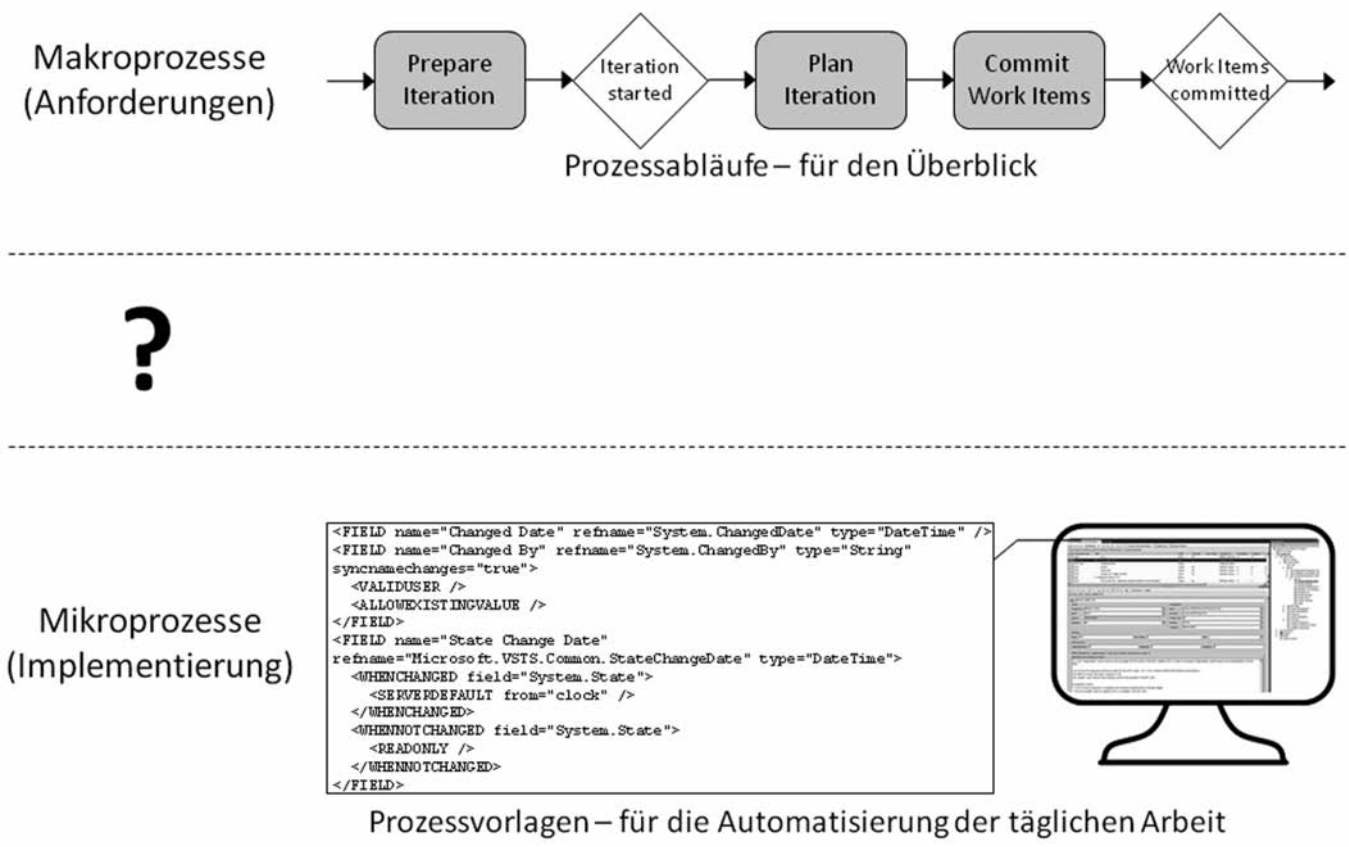

Abb. 3: Semantische Lücke zwischen Makro- und Mikroprozess.

zessen (GP) sind ablauforientiert. Als *First Class Citizens (FCC)* begegnen uns Aktivitäten, Ereignisse und Aktionen, die in zeitlicher Abfolge verknüpft sind (siehe Kasten 1). Mängel bei der Informationsbereitstellung und im Informationsfluss können damit identifiziert werden. Geschäftsprozesse werden in Teilprozesse bis hin zu Elementarprozessen zerlegt (vgl. [Sche01]). Je detaillierter die Beschreibung ausfällt, desto schwieriger und willkürlicher ist die Festlegung der zeitlichen Sequenz. Jede Konkretisierung geht zu Lasten der Allgemeingültigkeit oder der Korrektheit des Prozessmodells. Vorgehensmuster mit vielen Nebenläufigkeiten und hoher zeitlicher Flexibilität können nicht übersichtlich dargestellt werden. Ablauforientierte Prozessdarstellungen eignen sich also nur eingeschränkt für wenig strukturierte und facettenreiche Workflows wie im ALM.

Neue Prinzipien und Sichtenwechsel

Das Ziel der wertschöpfungsbasierten Optimierung und Harmonisierung der Entwicklung wird dadurch einen entscheidenden Schritt vorangebracht, dass die semantische Lücke zwischen Makro- und Mikroprozessen geschlossen wird. Damit wird die Wertschöpfung von der Sicht der Geschäftsprozesse bis zu deren Umsetzung in den Entwicklungsumgebungen nachvollziehbar. Neben der inhaltlichen Konkretisierung der Makroprozesse muss ein anderes Prinzip der Informationsmodellierung genutzt werden. Man benötigt ein Prinzip, das ein modulares Prozessdesign ermöglicht, das die Gestaltung der Prozessvorlagen unterstützt und mit volatilen, also mit beweglichen, schwankenden, unbeständigen, unstetigen, und wechselhaften Vorgehensweisen umgehen kann. Da sowohl Verhalten als auch Datenstrukturen für das Design einer Prozessvorlage relevant sind, bietet eine objektorientiere Sichtweise zu Konkretisierung der Makroprozesse wichtige Vorteile.

Eine Methode der Prozessbeschreibung mit diesem Ansatz sind die so genannten *Essential Practices* von Ivar Jacobson (vgl. [Jac09]). Er entwickelte die *Essential Practices* (siehe Kasten 3) als Mittel zur pragmatischen Darstellung der Zusammenarbeit in der Softwareentwicklung.

Wir haben diese Methode erweitert. Eine explizite Analyse der beschriebenen und gelebten Prozesse sowie die Ausrichtung der Methode an der Wertschöpfung führten zu einem ökonomischen Ansatz, den v-o-p. Die Erweiterungen ermöglichen die Ableitung des Designs für die Prozessvorlagen.

Value-oriented Practices

In Abbildung 4 sind die Elemente der v-o-p dargestellt. Ein zentrales Element ist das Artefakt bzw. *Artifact*. Die Artefakte sind Ergebnisse, die durch die Aktivitäten in beliebiger Abfolge erstellt werden. Eine Typisierung der Artefakte erfolgt über die *Artifact Types*. Beispiele hierfür sind die Artefakt-Typen „Anforderung" oder „Testfall". Für eine detaillierte Beschreibung der Artefakte werden verschiedene Reifestufen verwendet, die so genannten *Maturity Levels*. Eine Reifestufe eines Artefakts definiert sich aus der Wertschöpfung, die durch diese Stufe erreicht wird. Es werden damit nur wertschöpfungsrelevante Reifestufen beschrieben. Reifestufen, in denen beispielsweise formale Kriterien wie ein Qualitäts-Audit und/oder eine Freigabe erreicht werden, sind für die v-o-p nicht primär von Bedeutung. Artefakte haben in der Regel mehrere Reifestufen, die im Lebenszyklus, dem so genannten *Artifact*

„A practice provides a way to systematically address a particular aspect of a problem. It has clear beginning and end and includes verification, a clear goal and a way of measuring success. A practice needs to be repeatable and easily updated."

Kasten 3: Definition des Begriffs „Practice" (aus [Jac06]).

Lifecycle, in einer sachlogischen Reihenfolge beschrieben sind.

Die Aktivitäten, so genannte *Business Activities*, sind für das Erreichen der Reifestufen der Artefakte notwendig. Sie werden so lange ausgeführt, bis die definierte Reifestufe des Artefakts erreicht ist. Mit diesem Muster ist der Anfang (Artefakt mit Reifestufe x-1) und das Ende (Artefakt mit Reifestufe x) genau definiert. Eine zwingende Beschreibung des zeitlichen Ablaufs der Aktivitäten ist nicht notwendig. Der Erfolg der Aktivitäten kann exakt an der erreichten Reifestufe des Artefakts gemessen werden. Gleichartige Aktivitäten können auch über verschiedene Practices hinweg zu so genannten *Process Execution* gruppiert werden.

Die für die Ausführung der Aktivitäten notwendigen Kompetenzen werden in den *Competence Levels* beschrieben. Es gibt fünf additive Kompetenzstufen je Kompetenzfeld. Die Kompetenzfelder (*Competence Fields*) strukturieren zusammen gehörige Kompetenzen. Mehrere Artefakte mit unterschiedlichen Reifestufen werden über Wertestufen (Meilensteine), so genannte *Value Stages*, synchronisiert. Das heißt, es wird definiert, welche Ergebnisse in welcher Reife zu einer Wertestufe vorliegen müssen. Die Wertestufen repräsentieren damit eine projektorientierte Sicht, die den Projektfortschritt an Hand der erreichten Produktreife misst. Die Wertestufen werden durch eine Wertschöpfungskette (*Value Stream*) in eine sinnvolle logische Reihenfolge gebracht. Die einzelnen Elemente der v-o-p sind so strukturiert, dass unternehmensspezifische Abläufe immer mit Blick auf die resultierende Wertschöpfung gestaltet werden. Mit der intrinsischen Wertorientierung erweitern die v-o-p Jacobsons Konzept von Practices als modularen Prozessbaustein. Abbildung 5 stellt das Prinzip der Wertorientierung mit dem *Strukturierten Entity-Relationship-Modell (SERM)* der v-o-p dar. Die abgebildeten Existenzabhängigkeiten verdeutlichen die Wichtigkeit der „Artifact Lifecycles" mit den „Maturity Levels" im Kontext der „Value Stages" im „Value Stream".

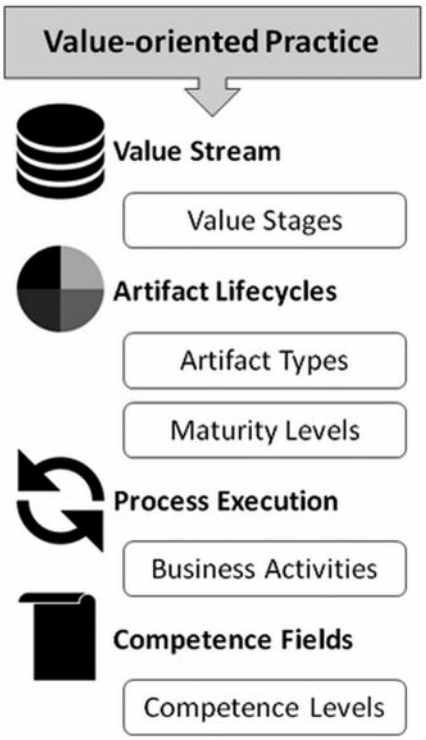

Abb. 4: Elemente der v-o-p.

Geschäftsprozessanalyse mit objektorientierten Methoden

Eine Analyse der Makroprozesse, also der beschriebenen Geschäftsprozesse, die das gewollte Vorgehen in der Entwicklung beschreiben, ist der erste wichtige Aspekt der v-o-p. Bei dieser Analyse stehen die Ergebnisse im Fokus, die auf der Makroebene entsprechend grob definiert sind. Häufig sind solche Ergebnisse umfangreiche Spezifikationen wie zum Beispiel:

- System-Anforderungsspezifikation
- Software-Anforderungsspezifikation
- Architektur-Spezifikation
- Komponentendesign-Spezifikation
- Quellcode
- Systemtest-Spezifikation
- Integrationstest-Spezifikation

Solche Ergebnisse setzen sich aus verschiedenen Artefakten zusammen, die bei der Analyse identifiziert werden. Moderne ALM-Systeme greifen auf zentrale Repositorys zu und organisieren Artefakte objektorientiert. Eine dokumentenzentrierte Sicht auf die Entwicklungsergebnisse ist dann nicht mehr vorrangig. Spezifikationen sind vielmehr Container, die eine definierte Menge von Artefakten mit einer bestimmten Reife beinhalten. Die Spezifikationen

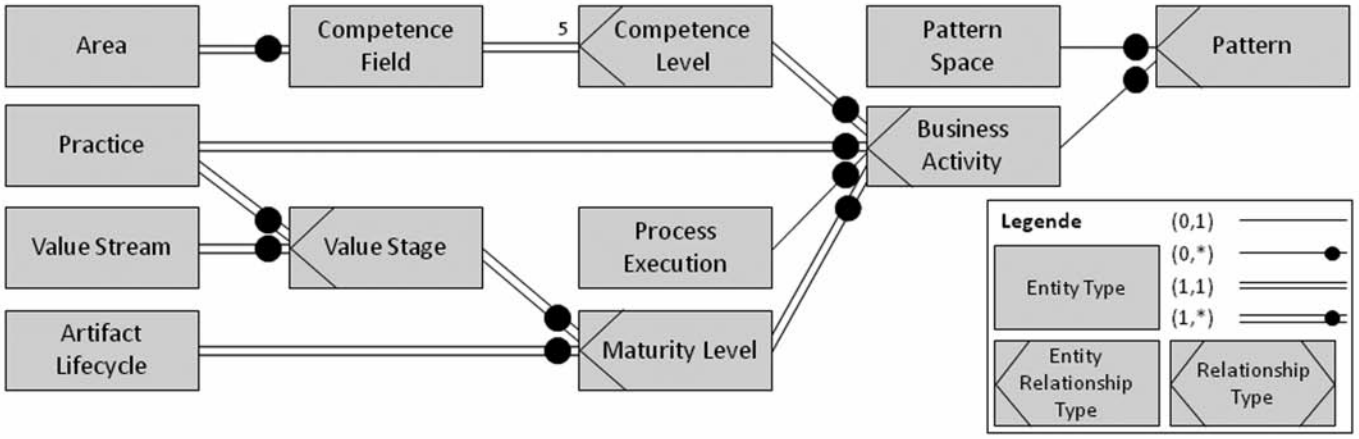

Abb. 5: Strukturiertes Entity-Relationship-Modell für v-o-p.

bilden Konfigurationsmengen von Artefakten. Sie werden zur Weiterverwendung versioniert und beispielsweise zum Nachweis regulatorischer Anforderungen oder für vertragliche Zwecke wie Ausschreibungen oder Angebote genutzt. Mit der objektorientierten Sicht auf die Bestandteile der Ergebnisse wird der Blick auf die wertschöpfenden Inhalte geschärft, da einem Objekt (Artefakt) die ihn bestimmenden Attribute (*Maturity Levels*) und Methoden (*Business Activities*) zugeordnet sind. Informationen über die Eigenschaften des Containers „Spezifikation" treten in den Hintergrund. So sind Dokumenten-Freigaben aus Sicht der Wertschöpfung am eigentlichen Produkt weniger relevant, weil hier im engeren Sinne kein Produktwert entsteht, sondern im besten Fall eine Produktreife festgestellt wird.

Identifizierung essenzieller Reifestufen

Artefakte und ihre Reifestufen stehen im Mittelpunkt der Analyse der Wertschöpfungskette im Entwicklungsprozess. Ein Projektbeispiel soll dies verdeutlichen, in dem einige Elemente aus Scrum adaptiert wurden, aber keine *1:1*-Umsetzung erfolgte. Für eine v-o-p müssen alle relevanten Ergebnisse und deren Artefakte analysiert werden. Der Scope der Analyse wird durch einen klar abgegrenzten Themenbereich bestimmt, den die v-o-p adressiert. In unserem Beispiel wurde für ein an Scrum angelehntes Vorgehensmodell unter anderem das Thema „Backlog-Management" für eine Practice identifiziert. Aus diesem Kontext ergibt sich das Ergebnis „Release-Backlog". Für das „Release-Backlog" wird unter anderen der Artefakt-Typ „Softwareanforderung" identifiziert. In diesem Artikel legen wir die Definition für Anforderung aus Kasten 4 zu Grunde. Für diesen Artefakt-Typ soll der vollständige Lebenszyklus untersucht werden. Das Thema „Backlog-Management" tritt zunächst in den Hintergrund und es stellen sich folgende Fragen:

- Welchen Lebenszyklus hat eine Softwareanforderung?
- Welchen Beitrag leistet die Softwareanforderung in der Wertschöpfung?
- Wie verteilt sich dieser Beitrag auf dessen Lebenszyklus?
- Welche wertschöpfende Reifestufen können identifiziert werden?

Aus der Beantwortung dieser Fragen ergibt sich in unserem Beispiel der essenzielle Lebenszyklus für eine Softwareanforderung (siehe Abbildung 6). Die Reifestufen bilden jeweils eine Wertschöpfungsstufe ab. So ist die Reifestufe „Defined" die erste inhaltliche Beschreibung der Erfordernisse an die zu entwickelnde Software. Die Reifestufe „Assigned & Committed" bedingt eine weitere Wertsteigerung der Softwareanforderung. Aus Sicht der Wertschöpfung reicht alleine die Zuordnung („Assigned") der Softwareanforderung zu einer Iteration und zu einem Team nicht aus. Erst mit dem *Commitment* entsteht ein Mehrwert. Es ist die verbindlichen Zusage zur Realisierung der Softwareanforderung im nächsten Sprint (vgl. [Schw06). Bei der agilen Entwicklung werden Softwareanforderungen gerne aus Sicht des Anwenders als User-Storys beschrieben. Idealerweise sollte der Realisierungsumfang je Softwareanforderung im Bereich einiger Personentage liegen. Gegebenenfalls wird die Softwareanforderung in Teilanforderungen (*Slices*) dieser Größenordnung aufgeteilt. Ein *Commitment* erfolgt durch die Personen, die die Umsetzung ausführen. Deshalb erfüllt die Softwareanforderung implizit eine Reihe von Qualitätskriterien (vgl. [Hoo05]).

Damit werden die Reife und der Wertzuwachs im Lebenszyklus der Softwareanforderung inhärent nachgewiesen. Folgende Qualitätskriterien müssen für ein *Commitment* unter anderem erfüllt sein:

- eindeutig
- identifizierbar
- notwendig
- realisierbar
- atomar
- nachweisbar

Welche Bedeutungen die Qualitätskriterien haben, zeigen die Definitionen im Kasten 5. Im Lebenszyklus der Softwareanforderung ist die nächste Reifestufe „Completed & Accepted" erreicht, wenn die implementierte Anforderung vom Stakeholder akzeptiert wurde. Bei Scrum erfolgt dies am Ende eines Sprints. Die letzte Reifestufe „Verified" besagt, dass die Implementierung auf Systemebene erfolgreich getestet wurde. Sobald alle Anforderungen verifiziert sind, beginnt die Validierungs- und Freigabephase des Software-Release.

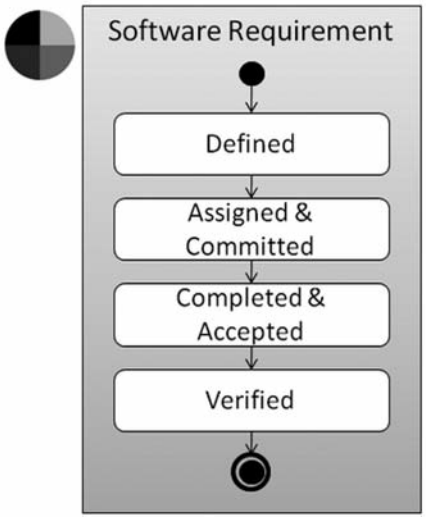

Abb. 6: *Essenzieller Lebenszyklus einer Softwareanforderung.*

Verbesserung der Makroprozesse

Hier ein Auszug aus dem Fragenkatalog für die Analyse von Makroprozessen:

- Sind die Geschäftsprozesse generisch beschrieben?
- Sind Start und Ende aller Aktivitäten definiert?
- Ist das Zusammenspiel verschiedener Aktivitäten beschrieben?
- Ist die Wertschöpfung jeder Aktivität nachvollziehbar?
- Sind die Geschäftsregeln vollständig definiert?

Die Reflexion und Beantwortung dieser Fragen gibt Hinweise auf Schwächen im untersuchten Makroprozess. Die Analyse führt zu definierten Verbesserungsmaßnahmen, deren Umsetzung die Prozessqualität steigert. Typische Schwächen sind fehlende Begriffsdefinitionen und ungenau oder unvollständig beschriebene Inhalte. Diese Defizite lassen sich durch bewusst eindeutige und präzise Darstellungen beheben. Dabei soll der Makroprozess nicht mit Informationen überladen werden. Vielmehr sollen die Inhalte auf gleich bleibendem Abstraktionsniveau in definierter Terminologie beschrieben werden, um dem Leser einen schnellen Überblick und eine eindeutige Orientierung zu verschaffen.

> „A condition or capability needed by a user to solve a problem or achieve an objective (ISO/IEC 24665:2009 Systems and software engineering vocabulary)."

Kasten 4: *Definition des Begriffs „Requirement" (aus [SEVO]).*

Zusammenfassung

Die v-o-p-Methode beschreibt Prozesse in modularer, auf das Wesentliche reduzierter Form. Dabei entsteht ein auf das Unternehmen zugeschnittener Prozessbaukasten, mit dem sich die Wertschöpfungskette im ALM optimieren lässt. Mit v-o-p schließen wir die Lücke zwischen Makro- und Mikroprozess (siehe Abbildung 7). Bei Siemens Healthcare konnten wir diverse Verbesserungsmöglichkeiten in der Definition der Sollprozesse identifizieren, beispielsweise bei der Modellierung von Abhängigkeiten. Im zweiten Teil dieses Artikels werden wir ein Beispiel aus dem agilen Backlog-Management mit Scrum untersuchen. Anschließend zeigen wir, welchen Beitrag v-o-p zur Automatisierung von ALM-Prozessen leistet. ■

Literatur & Links

[Hoo05] C. Hood, R. Wiebel, Optimieren von Requirements Management & Engineering – Mit dem HOOD Capability Model, Springer-Verlag 2005

[Jac06] I. Jacobson, P. Wei Ng, I. Spence, Enough of Processes – Lets do Practices Part II Time for Practices, 2006, siehe: www.drdobbs.com/embedded/198800543

[Jac09] I. Jacobson, R. Hauber, Practices als Alternative zu vollständigen Prozessen, in: OBJEKTspektrum 03/2009

[MSVS] Microsoft Visual Studio, Optimieren von Teamprojekten mit Prozessvorlagen, siehe: http://msdn.microsoft.com/de-de/magazine/dd221 363.aspx

[Sche01] A.-W. Scheer, Grundlegende Methodik von ARIS, Handbuch IDS Scheer AG, 2001

[Schw06] K. Schwaber, Agiles Projektmanagement mit Scrum, Microsoft Press, 2006

[SEVO] SEVOCAB: Software and Systems Engineering Vocabulary, IEEE Computer Society and ISO/IEC JTC 1/SC6 siehe: http://pascal.computer.org/sev_display/index.action (Suchbegriff: Requirement)

[Stö10] F. Stöckel, Hilfe zur Formulierung von Anforderungen, 2009, siehe: http://www.hood-group.com/fileadmin/user_upload/downloads/de/Brick01_Flyer_FormulierungAnf_v1.3.pdf

- *Eindeutig:* Eine Anforderung ist eindeutig formuliert, wenn sie genau eine Interpretation durch die erwarteten Leser zulässt.
- *Identifizierbar:* Eine Anforderung muss eindeutig als Anforderung erkennbar sein und einen Identifikator (ID) besitzen.
- *Notwendig:* Der Grund für die Existenz der Anforderung ist nachvollziehbar (z. B. in Form eines Stakeholders oder einer übergeordneten Anforderung).
- *Realisierbar:* Eine Anforderung kann im Rahmen des Projekts (Zeit, Kosten usw.) erfüllt werden.
- *Atomar:* Die Anforderung lässt sich nicht sinnvoll in weitere Anforderungen zerlegen.
- *Nachweisbar:* Es gibt eine im Projektrahmen anwendbare Möglichkeit nachzuweisen, dass das System die Anforderung erfüllt.

Kasten 5: Definition Qualitätskriterien von Anforderungen (vgl. [Stö10]).

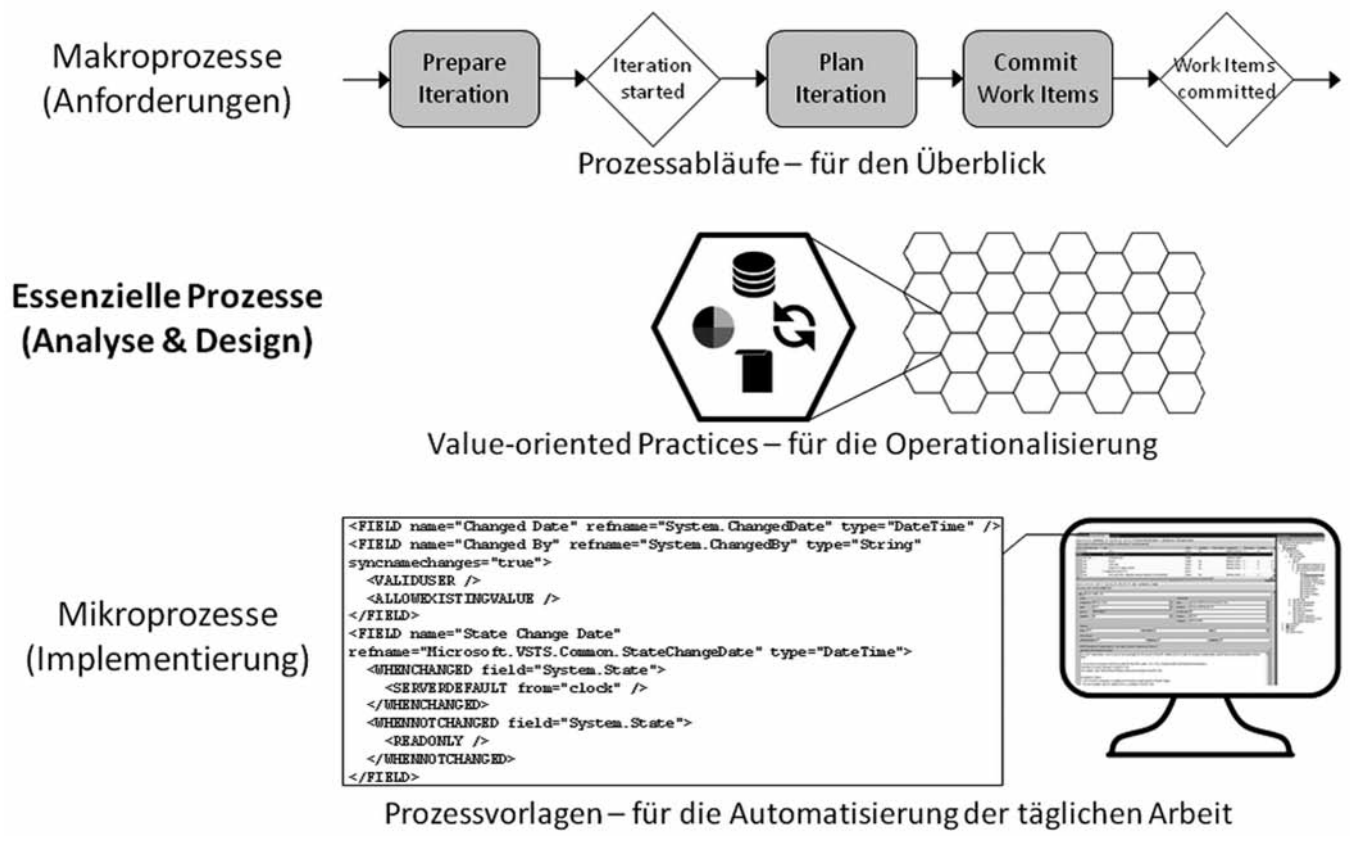

Abb. 7: Durchgängigkeit von Makro- zu Mikroprozessen.