Sonderbeilage zu Ausgabe 01/2014

www.OBJEKTspektrum.de

OBJEKTspektrum

IT-Management und Software-Engineering

G 6540F

editorial

20 Jahre OBJEKTspektrum: Wegbegleiter des Software-Engineering

Liebe Leserinnen und Leser,

ich möchte Sie einladen, mir auf dem Weg, auf den sich der Verlag vor 20 Jahren aufgemacht hat, zu folgen. Die rein thematische Analyse überlasse ich den hervorragenden, fachkundigen Autoren dieser Jubiläumsausgabe.

Als auf der OOP 1994 von Rick Friedman, dem damaligen Geschäftsführer, und Dr. Frances Paulisch, erste Chefredakteurin des OBJEKTspektrum, eine Euro-Palette mit der ersten Ausgabe feierlich enthüllt wurde und die Hefte verteilt wurden, war es das erste deutschsprachige Entwicklermagazin für den Bereich der *Objektorientierten Programmierung (OOP)*. Die erste Ausgabe zeigte deutlich den Einfluss des US-amerikanischen Mutterhauses (damals: SIGS Conferences, New York) mit allein sechs übersetzten Artikeln aus den USA. OOP war damals ein völlig neues Programmierparadigma, um Software zu entwickeln. Und OBJEKTspektrum hatte mit dem „Journal of Object-Oriented Programming" eine ältere „US-Schwester", die auch bei der Kontaktanbahnung mit potenziellen amerikanischen Autoren für die Chefredakteurin sehr hilfreich war.

OBJEKTspektrum stellte sich dann aber schnell auf eigene „Autoren-Füße" und so finden Sie heute Artikel von amerikanischen Autoren eher als Ergänzung zu den deutsch-sprachigen Inhalten denn als „tragende Säulen" der Zeitschrift.

So wie sich die Branche der Softwareentwicklung veränderte, so veränderte auch OBJEKTspektrum im Laufe der Jahre sein Gesicht. Die ersten Ausgaben wirken aus heutiger Sicht ein wenig psychedelisch in der Anmutung, aber das entsprach dem damaligen Zeitgeist: Der Softwareentwickler sah sich eher als „Künstler des Codes" denn als klassischer „Software-Ingenieur".

Im Zuge der Weiterentwicklung des Fachgebiets verbreiteten sich a) die Einführung der eigenständigen Informatikstudiengänge und b) die Nutzung des Internets für firmenübergreifende Prozesse, aber c) auch der Austausch von Informationen mittels E-Mail. Was lag also in dieser Zeit näher, als OBJEKTspektrum ein Gesicht zu geben, welches dieses reflektierte: OBJEKTspektrum erhielt ein Cover im Browser-Layout, denn Web-Services waren der neue Trend. Damit entfernten wir uns auf Dauer immer stärker von der „Schönheit der Lines-of-Code", hin zum Software-Engineering.

Und OBJEKTspektrum marschierte munter voran. Die Beiträge veränderten sich, passend zum Aufstieg der Ursprungsleserschaft vom Programmier-Level zu dem der Softwarearchitekten und Projektmanager. Die Wechsel in der Chefredaktion – von Frances Paulisch über Jens Coldewey bis hin zu Thorsten Janning – setzten diese zunehmende Orientierung der Softwareentwicklung auf den Business-Case fort.

So wie unsere Leser nutzen auch wir eine Kollaborationsplattform zur Herstellung der Ausgaben – vom ersten redaktionellen Kontakt, über die Reviews bis hin zur Übergabe der Daten an die Druckmaschine. Wir bieten OBJEKTspektrum sowohl im Print- und im Internetformat als auch als App an und bedienen damit alle Kommunikationskanäle, die unsere Leser nutzen.

Mit dem 20-jährigen Jubiläum und dem neuen Layout reflektieren wir wieder, was Sie uns als Leser kommunizieren (siehe auch die Glückwünsche auf S. 7) mit einem neuen Layout: Wir konzentrieren uns auf das Wesentliche, geben Artikeln den Raum, auch einmal in die Tiefe zu gehen, und tragen damit zur „Entschleunigung" bei, wie Sie als Leser dankbar kommentieren.

Auf der Verlagsseite möchte ich die Gelegenheit nutzen und danke der aktuellen Redaktion, Dr. Thorsten Janning, Chefredakteur, Kirsten Waldheim, Schlussredakteurin seit den Kindertagen des OBJEKTspektrum, und unserem Grafiker Sven Lesemann, der alle unsere Ideen und Vorschläge in ein konkretes Layout gegossen hat. Wir sind als Verlag stolz, dass wir, als scheinbar wirtschaftlich verlockende Alternativen präsentiert wurden, wie die Umstellung des Abonnement-Konzepts auf „Controlled Circulation", nicht auf diesen Zug aufgesprungen sind. Wir waren uns immer gewiss, dass Sie als Leser die Neutralität des OBJEKTspektrum schätzen und bereit sind, dafür einen angemessenen finanziellen Beitrag zu zahlen.

Und somit endet diese kurze Wegbeschreibung auch ***bei der wichtigsten Person: Ihnen, unserem Leser***. Ihnen gilt unser allergrößter Dank – für Ihre teilweise bis zu 20-jährige Treue, der wir uns jeden Tag verpflichtet fühlen. Wir möchten Sie auch weitere 20 Jahre mit unseren Beiträgen informieren, das für Sie Relevante aus der Menge der unwichtigen Informationen filtern und mit den veröffentlichten Beiträgen zum Nachdenken anregen.

Wir sehen uns – spätestens in der nächsten Ausgabe des OBJEKTspektrum.

Susanne Herl
Redaktions- und Herstellungsleitung

PS: Wir freuen uns sehr auf Ihre Kommentare zum neuen Layout des Magazins unter fachartikel@objektspektrum.de.

In die Zukunft kann sehen, wer ihren Anblick erträgt

Was bringt die Zukunft wirklich? Diese Frage wird oft beantwortet, meist ganz falsch – und der wahre Prophet muss sich dann noch lange Zeit verspotten lassen. Es gibt eine andere Frage, die die Menschen viel mehr umtreibt: Was erhoffe oder erwarte ich von der Zukunft? Wie schön könnte sie sein? Kurz: Wie wünsche ich mir die Zukunft?

Ich wünsche mir eine glückliche Familie, Wohlstand, Freizeit, Wellness, Gesundheit, Arbeitsplätze für alle, Urlaub überall hin und ein bedingungsloses Grundeinkommen. Die Menschen sollen wieder kooperativ sein, nicht miteinander kämpfen, Frieden halten und zu ethischen Werten zurückkehren. Und im ICE soll Internet verfügbar sein.

Die meisten Überlegungen zur Zukunft vermischen diese beiden Fragen. Prognose und Wunsch werden mental durcheinander gebracht. Wir spüren zum Beispiel (das ist eine Prognose), dass das Internet eine Menge Arbeit automatisiert und entsprechend Arbeitsplätze vernichtet. Das wollen wir aber nicht (Wunsch). Wir suchen dann unwillkürlich nach einer anderen Zukunft – meist wollen wir die reine Prognose nicht zur Kenntnis nehmen. „Es wird so schlimm nicht werden, es wird uns schon etwas einfallen." Das tut es aber nicht, wenn wir nur warten, dass uns etwas einfällt. Es wäre besser, wir nähmen eine Prognose ernst und würden wirklich aktiv, um die negativen Folgen der Zukunft schon heute einzudämmen versuchen.

Wenn ich selbst bei Interviews Prognosen abgebe, werde ich oft angeklagt. „Was – Sie wollen eine solche schreckliche Zukunft?" Ich erkläre dann zum x-ten Mal, dass ich nur vorhersage, was kommt – nicht, was ich persönlich *wünsche*. Aber das ist wohl so – wenn der Arzt jemandem sagt, er habe Krebs und werde bald sterben, hält dieser den Arzt für einen bösen und rücksichtslosen Menschen. Es ist ja auch üblich, den Überbringer schlechter Nachrichten zu bestrafen. Also in diesem Sinne: Ich versuche eine Prognose.

IT und Netze als langsam wachsende Strukturen

Denken Sie an die Erfindung der Waschmaschine zurück. Sie ist ein singuläres Gerät, der das Problem des Waschens in vier Programmen löst: Kaltwäsche, Schonwäsche, Warmwäsche und Kochwäsche. Wer etwas mit der Maschine waschen will, muss zuerst

(foto: www.fotolia.de)

im Wäschestück nachschauen, zu welcher Klasse dieses gehört – meist steht da: 30, 60 oder 90 Grad. Als die Waschmaschine erfunden wurde, gab es diese Unterscheidung noch nicht. Für die Waschmaschinen-Programme aber brauchte man eine Standardisierung. Dazu musste man die Wäsche praktisch neu erfinden – die Textilwirtschaft entwickelte neue Stoffe und Verfahren. Das war viel, viel mehr Arbeit als die alleinige Erfindung der Waschmaschine. Im Kern: Die Waschmaschine erforderte eine komplett neue Infrastruktur rund um Stoffe. Wenige Jahre später wurden Trockner modern. Auch sie erzwangen eine neuerliche Umstellung der gesamten Infrastruktur. Genau so etwas vollzieht sich heute rund um den Computer und das Netz, so wie beim Waschen und Trocknen. Da das Aufbauen neuer Strukturen lange dauert und immer mal wieder Akzeptanzbarrieren überwunden werden müssen, ist der Zeitpunkt schwer vorauszusagen, wann genau sich etwas Neues etabliert. Qualitativ aber kann man schon in die Zukunft schauen, finde ich. Ein Blick dorthin gefällig?

Das Internet der Dinge

In meiner allerersten Kolumne, die ich im Jahr 1998 schrieb, gibt es schon ein kleines Kapitel mit diesem Titel: „Das Internet der Dinge" (vgl. [Due99]). Das war damals denkerisch schon da. Jetzt sind wir 15 Jahre weiter und es gibt immer noch keine richtige Vernetzung der Dinge. Die Infrastruktur stimmt noch nicht. Aber es tut sich etwas: Samsung stattet gerade einige Elektrogeräte mit einem Android-Bedienfeld aus (auch mit iOS 7 erhältlich). Die Programme der Spüler, Waschmaschinen, Kühlschränke und der Kameras von Samsung bedienen wir nun wie eine App auf dem Smartphone, wo wir die App natürlich auch installieren und die Haushaltsgeräte von überall her steuern: „Licht an im Kühlschrank! Innenkamera an! Ich muss schauen, ob Schlagsahne da ist."

Ich glaube, dass sich aus den Betriebssystemen für Smartphones langsam solche für alle Geräte im Haus entwickeln und sich alles langsam zum Internet der Dinge auswächst. Es ist nicht so sehr die Idee, die etwas voranbringt, sondern eine Infrastruktur „von unten", die langsam ein ganz neues Technologiefeld erzeugt. So, wie die Textilien neu erfunden werden mussten, werden nun alle Geräte des Alltags mit so etwas wie „Android" versehen und mit unserem Leben enger als bisher verbunden. Und was sagen Sie zu dem Ganzen? „Wer braucht denn so etwas?"

Selbstfahrende Autos

Auch Autos brauchen eigentlich ein solches Betriebssystem. Diese Entwicklung ist absehbar. Hersteller von Luxusautos bepa-

cken die Autos schon länger mit Sensoren, damit das Auto den Stau oder die Parklücke vor sich erkennt und dem Fahrer hilft und ihn nicht nur warnt. Das sind Mini-Innovationen, die das Auto angenehmer machen. Der weitreichendere Versuch wäre, das Auto mit Kameras auszustatten und das Sichtfeld auf ein Display zu übertragen, sodass man das Auto vom Bildschirm aus steuern kann. Wo dabei der Bildschirm steht, ist technisch egal, also dann am besten in Bangalore oder in Vietnam, wo dann der Fahrer das Auto über das Internet lenkt – zu einem Zehntel des hierzulande üblichen Fernfahrerlohns. Das ist eine „industrialisierende" Innovation, nicht eine der „Commodity" der Luxushersteller. Wir wünschen uns den Luxus der Einpark-Sensoren, aber nicht das Abschaffen der LKW-Fahrer hierzulande.

Gehen wir noch einen Schritt weiter: Google testet schon lange selbstfahrende Autos und hat vor ein paar Wochen eine Viertelmilliarde in Anteile an der „Taxifirma" Uber Technologies gesteckt, die über eine App ein Taxi punktgenau zum Anforderer schickt. Diese Taxis sind teurer als normale Taxis, zum Teil sehr viel teurer, wenn Taxis knapp sind – „Pricing wie an einer Tankstelle". Was wird Google daraus machen? Bei unserer Anforderung per Smartphone kommen natürlich bald selbstfahrende Autos und keine Taxifahrer mehr. Und zum Schluss – das ist jetzt meine nüchterne Prognose – verbieten wir den Besitz eigener Autos ganz und stellen allen Transport auf selbstfahrende Taxis um. Jeder von uns kauft dann 24.000 km Flatrate im Jahr – und alles wird gut.

Warum ist das so toll? Mein Auto parkt meistens. Ich habe ausgerechnet, dass es am Tag zu 7 Prozent der Zeit fährt, sonst steht es herum. Ich schließe logisch: Zwischen 5 und vielleicht 20 Prozent der Autos fahren jemals gleichzeitig auf der Straße, mindestens 80 Prozent parken – viele stehen auf Bezahlparkplätzen und kosten auch dann, wenn sie nicht fahren. Wenn man das Google-Modell als Modell für den gesamten Verkehr nimmt, kann man also 80 Prozent der Autos einsparen. Dazu spart man alle Innenstadt-Parkplätze, alle Parkhäuser und alle Garagen an den Häusern. Alte Menschen können auf dem Dorf in ihrer Wohnung weiterleben – sie sind ohne Auto nicht länger hilflos. Das selbstfahrende Taxi bringt sie zum Einkaufen, zum Arzt und zum Theater. Die Lebensqualität für Senioren steigt erheblich. Millionäre zahlen den zehnfachen Preis für das Taxi und bekommen im System der nur noch selbstfahrenden Autos immer Vorfahrt – das lässt sich programmieren.

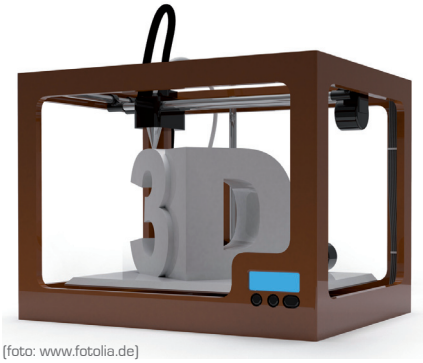
Fazit: Uns winkt eine wundervolle Welt mit viel weniger Autos – ohne Staus (die selbstfahrenden Autos regeln das besser als wir), ohne Verkehrstote, ohne Betrunkene am Steuer und ohne Probleme im Alter. Man muss „nur" eine riesige IT-Infrastruktur für die Verkehrsregelung aufziehen. Leider kostet es fast alle Arbeitsplätze im Automobilbau, da man viel weniger Autos bauen muss, die dann auch so ziemlich gleich aussehen können, eben wie fahrende Kabinen mit Internet und Schreibtisch drin – oder mit Spielplatz. Ich weiß, 30 Prozent der deutschen Arbeitsplätze hängen am Auto. „Herr Dueck, und das wollen Sie?" – „Hilfe, es ist eine Prognose!" Und ich frage, warum keine Autofirma, sondern Google das anfängt, wieder einmal Google, warum nicht *wir*?

Wir haben Angst vor den Konsequenzen. Wir könnten doch aber die Infrastruktur für die Gesamtverkehrsregelung selbstfahrender Autos auf Kiel legen? Und erst einmal nach Singapur verkaufen, wo es schon fast verboten ist, mit dem Auto in der Innenstadt herumzufahren?

3D-Drucken

Die billigsten 3D-Drucker kann man schon bei Amazon bestellen. Für 1.000 Euro können Sie mit 3D-Drucken anfangen. Sie können „Plastik drucken", also etwa Happy Hippos oder Osterhasen. Mit Hilfe einer digitalisierenden Scanner-Software oder Ihren NSA-Daten vom letzten USA-Flug können Sie sich selbst als Plastikpuppe ausdrucken lassen – in 3D. Das gibt ein fabelhaftes Geschenk für Ihren Lebenspartner.

Neulich erzählte mir jemand, dass es schon eine große Nachfrage von Schwangeren gäbe, die sich alle paar Monate als Skulptur ausdrucken lassen – und dass sie dann das Baby, Kind, den Teenager jedes Jahr vorbeibringen und neu ausdrucken, sodass sie statt des langweiligen Fotobandes mit Babybildern einfach das 3D-Baby in allen Jahrgängen auf dem Kaminsims oder als Gartenzwerg draußen stehen haben. Da alles digitalisiert ist, kann das Baby immer neu ausgedruckt und verschenkt werden, wenn es im Laufe der Jahre dann bessere HD-Farbdrucker gibt.



(foto: www.fotolia.de)

Ich sehe Ihnen im Geiste beim Lesen zu. Jetzt lachen Sie! Nicht ernsthaft? Doch. Meinetwegen etwas Ernsthaftes: Man kann heute sehr gut mit dem Metall Titan drucken, zum Beispiel Zahnimplantate oder neue Hüftknochen. Sehr teure Titandrucker lohnen sich heute schon für Zahnlabore und Kliniken.

Bei weiteren Anwendungen wird es schwieriger, es ist wie bei der Erfindung der Waschmaschine. Wir brauchen jetzt eine jahrelange, jahrzehntelange Grundlagenforschung in geeigneten Druckmaterialien. Der Drucker ist gar nicht so sehr der Engpass. Die Forschung wird immer mehr Anwendungen eröffnen.

Auf der anderen Seite sterben wieder wahnsinnig viele Arbeitsplätze, weil man jetzt alle eher selten gebrauchten Ersatzteile oder Maschinenteile einfach „dumm ausdruckt", ganz ohne Ingenieure zu bemühen. Es gibt auch schon superkomplizierte Teile an Autos, die man gar nicht traditionell herstellen, sondern nur ausdrucken kann, zum Beispiel mathematisch optimierte Kühler, die wie ein System von Gehirnzellen aussehen – irre kompliziert, aber das ist dem Drucker egal.

Für die Produktion im Gussverfahren werden Formen für die zu gießenden Teile hergestellt – na, die druckt man ebenfalls – usw. Auch hier kommt eine ganz neue Welt auf uns zu. Wer kümmert sich in Deutschland darum? Gerade heute, ich schaue beim Schreiben dieses Artikels auf die Kurse in New York, werden die Aktien der deutschen Voxeljet AG erstnotiert. Die Firma macht gerade etwa 11 Millionen Euro Umsatz mit rund 50 Mitarbeitern und wird heute Nachmittag bestimmt in der Gegend von 200 Millionen Dollar Gesamtwert an der Börse bewertet[1]. Überall sind Chancen auf großes Geld, eben zum Beispiel neue Industrien der Verkehrsregelung selbstfahrender Autos und der Materialforschung beim Drucken – wer geht mit? Ich leide wirklich darunter, dass die deutsche Phantasie immer nur so weit geht, sich den

[1] Nachtrag beim Korrigieren dieses Artikels: Die Aktien wurden zu 13 Dollar eingeführt und standen beim Börsenschluss bei 28,80 Dollar, also einem Gesamtwert von fast 500 Millionen Dollar!

breiten Wegfall klassischer Arbeitsplätze vorzustellen, und sich damit vor der Zukunft eher fürchtet.

Elitearbeiter oder Slum

Diese Furcht vor dem Wegfall von Arbeitsplätzen darf doch nicht lähmen – sie muss anspornen, sich „endlich zu bewegen“. Denn die schlimme Wahrheit ist, dass die naive Sicht, „dass viele Arbeitsplätze wegfallen“, die Dramatik der ganzen Angelegenheit gar nicht erfasst. Ich behaupte: IT und Internet erzeugen (wie im Beispiel Auto und 3D-Drucken) in allen Bereichen (Verlage, Banken, Energieverteilung usw.) immer mächtigere und wirksamere Infrastrukturen, die einen immer größeren Teil unseres Arbeitslebens betreffen. Damit wird das Internet zu so etwas wie einem „Gesellschaftsbetriebssystem“.

Ich hole argumentativ etwas aus: Ein Betriebssystem verwaltet die Ressourcen des Systems und stellt die erforderlichen Grundanforderungen zur Verfügung, die die größeren Anwendungen brauchen. Das Allgemeine und Einfachere wird im Betriebssystem abgehandelt, das Spezielle oder Komplexe in eigenen Anwendungsprogrammen. Wir sagen zum Beispiel: „SAP läuft auf Windows.“

Lassen Sie uns dieses Bild des Betriebssystems und der Anwendung auf Berufe übertragen. In vielen Berufen gibt es Routineaufgaben, die zunehmend von Computern übernommen werden. Bankservices kann man als Kunde weitgehend selbst erledigen, einfache Krankheiten lassen sich so gut „ergoogeln“, dass der Arzt auch nicht besser helfen kann, eine einfache Rechtsberatung lässt sich im Internet finden. Kurzum: Alles Einfache eines Berufes wird vom Internet zur Verfügung gestellt und oft auch gleich erledigt. Der Mensch am Arbeitsplatz muss nun nur noch das Komplexe und Schwierige selbst abarbeiten. Dazu muss er telefonieren, regeln, verhandeln, erörtern, überzeugen, verkaufen, entscheiden, verantworten, managen oder anleiten – denn genau das ist das Komplexe, das uns das „Betriebssystem“ nicht abnehmen kann. Dazu kommt, dass der Kunde Premium-Dienstleistungen erwartet, die er sich dank Googeln im Internet richtig gut vorstellen kann. Der Kunde ruft nicht einfach mal einen „Malermeister“, um die Tapeten zu erneuern. Er verlangt „Innendesign“. Der Kunde ist nicht mehr mit einem Gärtner zufrieden, er möchte Naturgestaltung für Ganzjahresblühen. IT-Kunden kaufen keine Einzelteile aus Hardware und Software zusammen, sie wollen „Lösungen“.
Das hat Konsequenzen:

- Die fachlichen Anforderungen steigen so stark, dass man von einem Upgrade der Berufe sprechen kann.
- Die Anforderungen an den Service und die Projekte steigen – jeder Mitarbeiter muss jetzt managen, führen, verkaufen, verhandeln usw.
- Nicht gut qualifizierte Mitarbeiter sind bei höher komplexen Aufgaben nicht nur ganz ungeeignet, sondern ihre Fehler (bei Projekten oder Beratung) machen in der Regel „mehr Verlust als ihr Gehalt“. Fehler im Komplexen sind nicht einfach nur verlorene Zeit oder ein Nachbesserungsfall, sondern eine Gesamtkatastrophe („Berliner Flughafen“).
- Schwächer oder auch mittelmäßig Qualifizierte haben keine Berechtigung mehr, weil sie Schaden anrichten – sie können nicht mehr dazu eingesetzt werden, einfache Arbeiten zu erledigen, weil es keine einfachen Arbeiten mehr gibt (die werden ja vom Betriebssystem übernommen).

Folglich braucht man nur noch superqualifizierte Fachleute mit allen Soft Skills, wie man heute sagt. Der Innendesigner designt uns eine Wohnung, dann kommen Mindestlohn-Arbeiter aus einem Handwerkerpool oder aus dem Ausland und tapezieren. Der Naturgestalter plant alles ganz genau, erstellt Skizzen für den Garten – und dann kommen wieder Mindestlohn-Arbeiter zum Graben und Einpflanzen. Das Mittlere oder heute Normale verschwindet. Internet und Computer spalten alle Berufe in einen hochqualifizierten Teil auf, dazu in einen zweiten, der im Internet von selbst oder vom Kunden erledigt wird, und in einen dritten, der körperliches Tun verlangt, was der Computer noch nicht kann (Roboter demnächst schon). Das alles führt zu einer Spaltung der Gesellschaft in „Hochqualifizierte und Mindestlohn-Arbeiter“ oder – schmissig gesagt – in Elite und Slum.

Diese Entwicklung ist schon weit fortgeschritten: Wir sehen die zunehmende Armut, wir können nachrechnen, dass Mindestverdienende dereinst nie und nimmer von der Rente leben können, die dann vielleicht 60 Prozent des letzten Gehaltes beträgt.

Warum tun wir nichts? Und vor allem: Was wollen wir tun? Das Internet und der Computer formen die Basis der eigentlichen Wissensgesellschaft – wir Menschen sollen auf dieser Basis arbeiten, müssen also von der Bildung her auf ein höheres Niveau kommen, so wie es schon einmal im 17. Jahrhundert mit der Einführung der Schulpflicht geschah. Wir brauchen dringend eine neue Vorstellung des in der Zukunft arbeitenden Menschen – wir müssen Abschied nehmen vom Facharbeiter und Akademiker, für die das Schulsystem entworfen wurde.

So sonnenklar es ist, dass wir das Niveau anheben müssen: Politiker sind allenfalls bereit, wieder einmal noch eine Milliarde „in die Bildung zu pumpen“ und damit vielleicht die quantitativ Not zu lindern. An der Vorstellung unserer Zukunft und des in ihr gewünschten Menschen arbeitet niemand. Viele regen sich eher auf: „Soll jetzt jeder studieren? Jeder managen und verkaufen?“ Und ich wende ein: „Wer nicht so weit kommt, bekommt allenfalls Arbeit, die mit Mindestlohn bezahlt wird.“ „Und was machen wir mit den vielen Menschen, die nicht gebildet sind?“ Da frage ich: „Fertigkeiten und Bildung sind doch wundervoll, warum sträubt ihr euch?“

Unsere Zukunft entscheidet sich daran, dass wir mindestens so tüchtig und gerne *lernen* wie wir fleißig arbeiten – daran, dass wir nicht nur bereitwillig Überstunden für den Chef leisten, sondern einfach uns selbst entwickeln – für unsere eigene Zukunft als Upgrade-Mensch. ‖

Literatur

[Due99] G. Dueck, Rundum Business Intelligence!, Teile 1-3, in: Informatik Spektrum, Band 22, 1999

der Autor

‖ Gunter Dueck
(dueck@omnisophie.com)
war Mathematikprofessor in Bielefeld und dann 24 Jahre bei der IBM Deutschland, zuletzt als Chief Technology Officer. Er schrieb zahlreiche Bücher im Bereich Management/Philosophie/Psychologie/Bildung. Auch 2014 wird er als Referent bei der OOP dabei sein.

Auszug aus den vielen Glückwünschen zum Jubiläum

OBJEKTspektrum

Prof. Dr. Andreas Rausch, Technische Universität Clausthal

„Danke für 20 Jahre Interessantes und Neues rund um das Software Engineering. Mir hat es immer Spaß gemacht, als Leser, als Autor und als Mitglied der Redaktion. Weiter so."

Frank Pientka, Materna GmbH

„Wie ein guter Fotograf nicht nur verschiedene Objektive einsetzt, ist der kreative Einsatz von Blende und Zeit wichtig. Deswegen müsste das OBJEKTspektrum eigentlich OBJEKTIVspektrum heißen, da es bisher gut verstanden hat, die verschiedenen Aspekte der Softwareentwicklung aus unterschiedlichen Perspektiven und mit der nötigen Schärfe und Belichtung zu betrachten. Deswegen wünsche ich weiterhin ein „Gut Licht!"

Morten Burmeister

„OBJEKTspektrum wird 20 Jahre alt und ich sende freudige Glückwünsche. Die Leser bezirzt es mit Aktualität, Relevanz und dem sonst oft vermissten Blick über den Tellerrand."

Bernd Rücker, camuda services GmbH

„Das OBJEKTspektrum hat mich bereits während des Studiums begleitet und flattert seitdem regelmäßig ins Haus – auch in Zeiten von Blogs und Google schätze ich immer noch die gut geschriebenen Beiträge und einen interessanten Mix aus Themengebieten, von Architektur über Technologie hin zu Vorgehensmodellen. Weiter so!"

Martin Schimak, plexiti GmbH

„Das OBJEKTspektrum gibt uns mitten in einer oft auf 140 Zeichen limitierten Zeit jede Menge Raum für ungekürzte Gedanken. Das finde ich gerade als Fan von Twitter & Co. genial – und wünsche mir weitere 20 Jahre Zukunft für Fachinformation mit Tiefgang."

Christopher Schulz, Syracom Consulting AG

„Liebes OBJEKTspektrum, ein herzliches „Happy Birthday" zum Aufbruch in die dritte Dekade. Mögen die kommenden 10 IT-Jahre, die Du in Deinen Beiträgen auf Schritt und Tritt begleitest, ebenso spannend und abwechslungsreich wie die vergangenen sein."

Jens Coldewey, Principal Consultant bei der improuv GmbH und Chefredakteur des OBJEKTspektrum von 2009 bis 2011

„Das OBJEKTspektrum kombiniert seit 20 Jahren für mich den Qualitätsanspruch wissenschaftlicher Publikationen mit dem praktischen Bezug, der nur möglich ist, wenn Praktiker für Praktiker schreiben. Was für die Autoren oft anstrengend und manchmal auch frustrierend ist, kommt aber dem Leser zugute. 20 Jahre lang die führende Fachzeitschrift für Software-Engineering und -Architektur zu sein, kommt eben nicht von ungefähr."

Matthias Bohlen, Matthias Bohlen IT-Beratung

„Herzlichen Glückwunsch an das Team vom OBJEKTspektrum – schon 20 Jahre damit unterwegs und immer aktuell, auf der Höhe der Zeit und dem seriösen Anspruch gerecht geblieben! Weiter so, auf die nächsten 20!"

Carsten Sydow, Swiss Life

„Für mich ist diese Zeitschrift die Beste im Bereich Softwareengineering auf dem Markt.

Sie stellt für mich auch eine ideale Ergänzung zum Besuch der OOP dar."

Prof. Dr. Gerhard Wanner, Fachhochschule Stuttgart

„Das OBJEKTspektrum hat mich große Teile meiner Berufslaufbahn begleitet und dort immer wieder wichtige Impulse gegeben und für Denkanstöße gesorgt. Mit dem OBJEKTspektrum verbinde ich ausgewiesene Expertise, wertvolle Anregungen und tiefgehendes Know-how.

Herzlichen Glückwunsch zum Jubiläum und viel Erfolg für die nächsten 20 Jahre!"

Dr. Frances Paulisch, Leiterin der Siemens Software Initiative und Chefredakteurin in der „Jugendphase" des OBJEKTspektrum (1994–2009)

„Besonders in der schnelllebigen Welt der Software ist es extrem wichtig, auf dem Laufenden zu sein. OBJEKTspektrum bietet einen sehr effektiven Weg, State-of-the-Art zu bleiben, und ist außerdem eine bewährte Quelle für neue Ideen, sodass man seine Wissensbasis auch verbreitern kann."

Christian Heckler, IKB Deutsche Industriebank AG

„Für einen Softwareentwickler ist es nicht leicht, den Überblick im Dschungel neuer Trends, Methoden und Produkte zu behalten.

In diesem Dschungel war das OBJEKTspektrum stets ein Leuchtturm für mich. Dafür vielen Dank und herzliche Glückwünsche zum Jubiläum!"

Standards, offene Standards und Open-Source

Wie bewältigen wir die großen anstehenden Herausforderungen in der Softwareindustrie? Was braucht es, um die Plattformen der Zukunft zu beschreiben oder sogar zu implementieren? Dieser Frage will ich in dem Artikel nachgehen.

In der Historie ist der Wettstreit zwischen De-facto-Standards und Standardisierungsbestrebungen ein stets wiederkehrendes Muster in der Informationstechnologie. Während die so genannten De-facto-Standards mitunter auf der Arbeit eines Unternehmens oder sogar eines Teams beruhen, werden Standards meist kollaborativ erarbeitet. De-Facto-Standards entstehen oft auf der Basis einer Implementierung (Windows, SNA, TCP/IP, Arpanet), während Standardisierungen meist auf einer theoretische Basis von einem Gremium – industriegebunden oder öffentlich – beruhen, in der Verfahrensweisen oder Schnittstellen beschrieben werden.

Standards sind die traditionellen Vorgehensweisen, um gemeinsam technische Lösungen zu beschreiben. Diese Methodik hat in der Vergangenheit maßgeblich zu Fortschritten in der Informatik geführt. In der Gegenwart gibt es einen Trend zu „Standardization by Implementation": Entweder wird die Standardisierung durch parallele Referenzimplementierungen beschleunigt oder es werden sogar Standards nachrangig – basierend auf bestehenden Implementierungen – definiert. Hier spielt die Implementierung in Open-Source aus vielen Gründen eine besondere Rolle.

Standards und offene Standards

Standards können von kleinen Industrieorganisationen getrieben werden oder aber von größeren und großen übergreifenden Konsortien (beispielsweise w3c, OASIS und OMG IEEE). So ist zum Beispiel der Autosar-Standard nur relevant für die Automobilindustrie, während die Spezifikation von HTML industrieübergreifend vom w3c gelenkt wird.

Standardisierung hat in der Elektronik und Informatik in den letzten Jahrzehnten zu bedeutenden Fortschritten im Bereich der Interoperabilität geführt oder diese sogar erst ermöglicht.

Erfahrungsgemäß bringen Standardisierungen aber auch eine Reihe von Problemen mit sich: So bilden Standards häufig nicht die bestmöglichen Lösungen ab, sondern die in politischen Gremien möglichen Kompromisse und haben somit oft viele Schwachstellen. Oder Standards werden „proprietär" erarbeitet: Ihre Erarbeitung und (kommerzielle) Benutzung sind einer bestimmten Gruppe vorbehalten oder durch Mitgliedschaften eingeschränkt. Neue Versionen werden nach einem wasserfallähnlichen Vorgehen erarbeitet und nur in langen Abständen veröffentlicht.

In den 90er Jahren war ich beispielsweise an mehreren Standardisierungsgremien im Bereich „Financial Frameworks" im Windows-Umfeld beteiligt (z. B. WOSA-XFS). Für mich war die Arbeit in solchen Gremien verwirrend: Wie können wir theoretisch über das Zusammenspiel verschiedener Komponenten und Geräte diskutieren, ohne zu wissen, ob unsere Gedanken richtig sind und wie sie praktisch funktionieren?

Während die Teilnehmer der anderen Unternehmen weiter dem theoretischen Ansatz folgten, konnten wir in unserem Unternehmen das Geld zu einer „Überprüfung" unserer Gedanken durch eine prototypische Implementierung aufbringen, die wir parallel zu den Meetings betrieben. Das Ergebnis war frappierend: Schon nach kurzer Zeit hatten wir nicht nur den Standard im wahrsten Sinne des Wortes begriffen, sondern wir fingen an, die Standardisierung zu treiben, indem wir funktionsfähigen Code vorlegten, der unsere Ideen unterstützte.

Wir konnten die Diskussionen extrem verkürzen, unsere eigenen Ideen und Gedanken besser positionieren und später auch unser Produkt viel schneller auf den Markt bringen. So lernte ich eine interessante Lektion, der ich später noch oft begegnen sollte. Im Gegensatz zu den Standards haben offene Standards den Vorteil, dass es wie bei Open-Source keine Einschränkungen bei der Verwendung des Standards gibt (kommerziell oder nicht-kommerziell): keine Einschränkungen durch Patente, keine Einschränkungen durch andere Technologien oder aber Einschränkungen in der Veröffentlichung (*Intentional Secrets*) hindern daran, den offenen Standard einfach einmal auszuprobieren. Das ist eine wichtige Voraussetzung dafür, dass eine Implementierung des Standards von vielen durchgeführt und so eine Überprüfung schnell und kostengünstig durchgeführt werden kann (vgl. [Ope], [Wik]).

Als zwei konträre Beispiele möchte ich hier wiederum Autosar (vgl. [Aut13]) auf der proprietären Seite und OSGi (vgl. [OSG13]) auf der offenen Seite anführen. In Unterhaltungen mit Autosar-Teilnehmern hörte ich oft, dass die langen Zykluszeiten und der geschlossene Review-Prozess zu Problemen führten. Ebenso verbesserte sich die Tooling-Seite erst durch die Kollaboration von vielen in der Artop-Gemeinschaft (vgl. [Art]). Immer noch werden nach meiner Einschätzung die Verbreitung und damit die Kommerzialisierung dieses Standards durch seine Einschränkung auf die Verwendung in der Automobilindustrie behindert.

Solche Einschränkungen fanden sich nicht bei der Spezifikation von OSGi. So hat sich OSGi über verschiedene Industriedomänen verbreiten können (IT, Home Automation, IoT usw.) und ist zu einem interessanten Gegenpol der früher von Sun Microsystems geführten JCP geworden.

Offene Innovation und Open-Source

Standards beschreiben im Allgemeinen nachgängig ein Format oder eine Technologie, selten jedoch eine aktuelle Innovati-

Literatur & Links

[Art] Artop (AUTOSAR Tool Platform), siehe: http://www.artop.org/
[Aut13] Autosar, Homepage, 2013, siehe: http://www.autosar.org/
[Che06] H.W. Chesbrough, Open Innovation: The New Imperative for Creating and Profiting from Technology, Harvard Business Press, 2006
[Ope] Open Source Initiative, Open Standards Requirement for Software, siehe: http://opensource.org/osr
[m2m] The Eclipse Foundation, m2m.exlipse.org, siehe: http://m2m.eclipse.org
[OAS13] OASIS, 2013, siehe: https://www.oasis-open.org/
[OSG13] OSGi Alliance, Homepage, 2013, siehe: http://www.osgi.org/Main/HomePage
[Pah] Paho, siehe: http://www.eclipse.org/paho/
[Wik] Wikipedia, Offener Standard, siehe: http://de.wikipedia.org/wiki/Offener_Standard

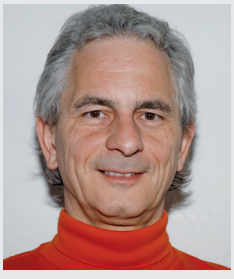
on. Innovationen werden jedoch oft direkt als Implementierungen oder Programme zusammen mit entsprechenden Veröffentlichungen vorgestellt. Da sich aus solchen Innovationen häufig ein Wettbewerbsvorteil für das Unternehmen ergibt, werden selbige vielfach mit Patenten oder proprietären Lizenzen geschützt. Wie aber schon Henry W. Chesbrough in seinem beachtenswerten Buch „Open Innovation: The New Imperative for Creating and Profiting from Technology" (vgl. [Che06]) erläutert, benötigt unsere komplexer gewordene Welt nicht nur offene Plattformen, sondern oft auch quelloffene Implementierungen, um eine Innovation allgemein verfügbar zu machen. Als Beispiel aus dem Eclipse-Umfeld können hier die Rich-Client-Plattform, das Plug-In-Development-Tool oder auch die Open-Source-nahe SWT-Plattform angeführt werden. Während jede für sich genommen eine kleine Innovation darstellt, haben sie doch insgesamt für die Entwicklung eines großen Eco-Systems gesorgt. Offensichtlich hat hier die Entwicklung in Open-Source auch für De-facto-Standards gesorgt und einen neuen Markt definiert, ohne dass ein formaler Standard definiert wurde. Durch die Offenheit der Implementierung gab es kaum Eintrittsbarrieren für neue Marktteilnehmer und die Verfügbarkeit „ohne Grenzen" führte zu einer rasanten Adaption.

Ausblick

Standards werden in der Informationstechnologie weiterhin eine große Rolle spielen. Ich hoffe, dass Standardisierungsgremien immer häufiger offene Standardisierungen bevorzugen. Dies führt nach meiner Einschätzung zu Zeitgewinnen, Kostenersparnissen und Qualitätsverbesserung. Weiterhin vermute ich, dass Parallelentwicklungen eines Standards in Open-Source zu weiteren Verbesserungen im Bereich Zeit, Qualität und Kosten führen sowie auch zu einer schnelleren Marktausbreitung. Wie wir bei OSGi sehen, gibt es auch hier die Koexistenz zwischen offenen (Felix, Equinox) und proprietären Implementierungen (z. B. Firma ProSyst).

Für die großen Aufgaben der Zukunft wünsche ich mir einen Ansatz wie den der DARPA in den 60er Jahren, der uns letztendlich das Internet beschert hat. Das Internet der Dinge (*Internet of Things – IoT*) oder auch Industry 4.0 können nicht durch Standards erschlossen werden. Hier braucht man (eine) solide und innovative Open-Source-Plattform(en), die für Experimente und Produktideen bereitsteht. Die Eclipse Foundation hat hierzu im letzten Jahr die M2M Working Group (vgl. [m2m]) gegründet, in der viele Unternehmen ihre Projekte beitragen, um so die Zukunft des IoT mit zu gestalten. Ein herausragendes Beispiel ist das Protokoll MQTT, das als quelloffene Implementierung (vgl. [Pah]) schon vorliegt und weiterentwickelt wird und das bis 2014 bei OASIS (vgl. [OAS13]) als offener Standard vorliegen soll. ||

der Autor



|| Ralph Müller
(ralph.mueller@eclipse.org)
ist Direktor der Eclipse Foundation in Europa. Mit weltweit rund 170 Mitgliedern mit mehr als 200 Projekten und 900 Entwicklern ist Eclipse eine der erfolgreichsten Open-Source-Gemeinschaften.

Die Entwicklung der OMG-Standards

Vor 20 Jahren veröffentlichte das OBJEKTspektrum in seiner allerersten Ausgabe einen Artikel über die Relevanz der OMG-Standards und der daraus resultierenden anwendungsnahen Lösungen. In diesen längst vergangenen Tagen umfassten die OMG-Standards die CORBA-Spezifikation (Common Object Request Broker Architecture) und dazugehörige Services. Eine Fokussierung auf Modellierung und Standards für vertikale Märkte lagen noch in weiter Zukunft. Auch die Standards für Echtzeit und eingebettete Systeme befanden sich damals noch in einer frühen Entstehungsphase. CORBA hatte und hat immer noch einen immensen Einfluss auf das, was sich inzwischen zu einem florierenden Middleware-Markt entwickelt hat. OBJEKTspektrum hat diese Bedeutung früh erkannt und schon frühzeitig darüber berichtet.

Als internationales, offenes und gemeinnütziges Konsortium zur Entwicklung von Standards für die Computerbranche steht die Object Management Group (OMG) für wegweisende Richtlinien und stellt sicher, dass die von der OMG anerkannten Standards breite Unterstützung in der Industrie erfahren und alltagstaugliche Anwendungen hervorbringen. Die Grundregel, dass jede Organisation nur genau eine Stimme innerhalb des Konsortiums hat, stellt sicher, dass der Standardisierungsprozess nicht durch einzelne Großunternehmen dominiert und einseitig beeinflusst werden kann. Ebenso sorgt eine „Anti-Shelfware"-Regel dafür, dass ein Standard nur verabschiedet werden kann, wenn mindestens ein realistischer Plan für seine Implementierung existiert. Dies war für CORBA recht offensichtlich der Fall und trifft ebenso auf jeden anderen der circa 1.000 bislang verabschiedeten OMG-Standards zu.

Alle OMG-Standards durchlaufen einen strengen Anerkennungsprozess. Einige Standards starten mit einem *Request For Information (RFI)*, der von einer OMG-Arbeitsgruppe (*Task Force*) ausgeschrieben wird, um ausreichende und objektive Informationen über ein Themengebiet zu erfassen. Die Rückmeldungen hierauf werden von der Arbeitsgruppe ausgewertet.

In einem darauf folgenden optionalen Schritt wird ein *Request For Proposal (RFP)* ausgeschrieben, auf den Mitgliedsorganisationen mit Beiträgen zur zukünftigen Spezifikation – *Letter of Intent" (LOI)* – reagieren können. Interessierte Mitglieder können diese Einreichungen anschließend lesen und kommentieren. Auf Basis dieser Inputs wird der zukünftige Standard immer wieder überarbeitet, bis man ihn als reif genug für eine Anerkennung einstuft. Was folgt, ist eine Reihe von Abstimmungen:

- Die Arbeitsgruppe stimmt darüber ab, ob man den Standard beim technischen Komitee einreicht.
- Das Architektur-Gremium stimmt seinerseits über eine Anerkennung ab.
- Das technische Komitee stimmt über eine Anerkennungsempfehlung an das Direktoren-Gremium der OMG (*Board of Directors – BOD)* ab.
- Das BOD-Untergremium für Geschäftsangelegenheiten berichtet dem BOD über das Ergebnis einer Umfrage innerhalb des Untergremiums.
- Wenn mindestens eine positive Rückmeldung zu dieser Umfrage eingegangen ist, stimmt das BOD über die Anerkennung des Standards ab.

Wurde dieser Prozess erfolgreich abgeschlossen, wird die Spezifikation zu einem offiziellen OMG-Standard. Gleichzeitig wird ein separater, ähnlich strukturierter Pflege- und Weiterentwicklungszyklus gestartet, der sicherstellt, dass alle OMG Standards aktuell und relevant bleiben.

Zurück ins Jahr 1994: Als das OBJEKTspektrum seinen ersten Auftritt hatte, feierte die OMG ihr fünfjähriges Bestehen. Der Kern des Konsortiums konzentrierte sich in dieser Zeit auf die Entwicklung von Standards für verteilte Objekte mit einem besonderen Schwerpunkt auf seine Vorzeige-Spezifikation CORBA. Sie erlaubt es Softwarekomponenten zusammenzuarbeiten, auch wenn sie in unterschiedlichen Programmiersprachen geschrieben und auf unterschiedlichen Plattformen betrieben werden. Es sollte noch ein bis zwei weitere Jahre dauern, bevor die OMG begann, sich ebenso stark um die Definition von Entwicklungsmethoden zu kümmern.

Diese Veränderung führte zu der Standardisierung einer der wohl am meisten verwendeten OMG-Spezifikationen: der *Unified Modeling Language (UML)* sowie der ihr zu Grunde liegenden *Meta-Object-Facility (MOF)*. Als vorherrschende Modellierungssprache führte die UML auch zur Entwicklung einer ganzen Familie von Sprachen:

- Für die Systementwicklung entstand die *OMG Systems Modeling Language (SysML)*.
- Für die Geschäftsprozess-Analyse wurde die *OMG Business Process Modeling Notation (BPMN)* entwickelt.
- Es entstanden Sprachen für Unternehmensarchitekturen, wie die *OMG Unified Profiles für DoDAF & MODAF (UPDM)* und die *Service-oriented Architecture Modeling Language (SoaML)*.

Im ersten Jahrzehnt, nachdem die UML von der OMG standardisiert wurde, entwickelte sich der globale Markt für objektorientierte Werkzeuge von einem Jahresumsatz von 22.500 Euro auf 3 Millionen Euro. Mit der UML können Softwareentwickler sicherstellen, dass die Funktionalität ihrer Software vollständig und korrekt ist, dass Benutzeranforderungen erfüllt sind und dass der Softwareentwurf die gestellten Anforderungen an Skalierbarkeit, Robustheit, Sicherheit, Erweiterbarkeit und andere Charakteristika erfüllt.

Seit 1997 hat die OMG ihre Aktivitäten auf die Standardisierung für vertikale Märkte konzentriert. Die Fokussierung auf verschiedene vertikale Märkte hat es ermöglicht, die entstandenen OMG-Standards für den jeweiligen Industriezweig anzupassen. Heute sind 85 Prozent aller OMG-Aktivitäten auf über zwei Dutzend vertikale Märkte ausgerichtet, wozu auch viele neue Entwicklungen gehören, wie zum Beispiel:

- Die Finanzbranche mit Standards wie der *Financial Industry Business Ontology (FIBO):* Die globale Finanzkrise 2008 hat zu diversen internationalen Gesetzen zur Finanzaufsicht und Erfassung von Finanzdaten geführt, wie die Dodd-Frank-Gesetze in den Vereinigten Staaten. Getrieben durch daraus entstehende Anforderungen, ist der FIBO-Standard eine industrielle Initiative zur Spezifikation von Begriffen, Definitionen und Synonymen der Finanzbranche mittels Möglichkeiten des semantischen Webs und weitverbreiteter OMG-Modellierungsstandards.
- Die Versicherungsbranche mit dem

In der allerersten Ausgabe des OBJEKTspektrum gab es einen Artikel über die Relevanz der OMG-Standards und der daraus resultierenden anwendungsnahen Lösungen.

Standard *Property and Casualty (P&C) Information Model:* Als Referenzmodell adressiert das P&C-Information Model die Bedürfnisse für Sach- und Haftpflichtversicherungen in den Vereinigten Staaten. Basierend auf den Prinzipien der *OMG Model Driven Architecture* und existierenden P&C-Branchenstandards, wie dem IBM *Insurance Application Architecture Framework*, etabliert die P&C-Spezifikation ein Branchenglossar sowie Methoden und definiert Fachbereiche für den Markt.

- Die Automobilindustrie mit dem Standard *Dependability Assurance Framework for Safety-Sensitive Consumer Device*, früher bekannt als *Consumer Device Safety Standard:* Da die Haftungsbedingungen immer kritischer werden, müssen Automobilhersteller eine durchgängige Verlässlichkeit ihrer Fahrzeuge während der Entwicklung und des Betriebs sicherstellen. Der Standard wird ein Prozessmodell für die simultane Entwicklung von eingebetteten Steuergeräten und zugehörigen Zuverlässigkeitsszenarien umfassen.

Am Anfang dieses Jahrtausends begann die OMG damit, sich auf nicht-standardisierungsbezogene Aktivitäten zu erweitern, um tiefgreifender mit verschiedenen Technologien und Branchen zu arbeiten. 2007 wurde das *SOA*-Konsortium (*Service-Oriented Architecture*) als Partnerschaft vieler Fortune 200 Unternehmen gegründet. Als SOA-Interessensgemeinschaft setzt sich das Konsortium aus Anwendern, Anbietern und Technologieherstellern zusammen, die alle das Ziel verfolgen, eine globale Verbreitung von SOA bis zum Jahr 2010 zu ermöglichen. Tausende nahmen an Treffen des SOA-Konsortiums teil, folgten Experten-Podcasts, haben Whitepapers und Präsentationen zu dem Thema heruntergeladen und steuerten Fallstudien zu einem SOA-Fallstudien-Wettbewerb bei, der gemeinsam vom SOA-Konsortium und dem CIO-Magazin ausgerichtet wurde. Sie abonnierten den SOA Insights Blog und besuchten Vortragsreihen des SOA-Konsortiums im Rahmen von führenden Industrieveranstaltungen.

Während das Ziel des Konsortiums – die Verbreitung von SOA – als erfolgreich erreicht verbucht werden konnte, entwickelte sich das Konsortium selbst weiter. Um seinen Fokus auf die Optimierung von Geschäftsprozessen zu erweitern, entstand das BPM/SOA-Konsortium unter dem Schirm der *Business Ecology Initiative (BEI)*: eine Managementinitiative, um Unternehmen reaktionsfähiger, effektiver, nachhaltiger und sicherer innerhalb einer komplexen und vernetzten Welt zu machen. Als Grundidee betrachtet die BEI eine Organisation als einen lebendigen Organismus, in dem alle Aspekte des Managements, Betriebs und der Prozesse miteinander interagieren und zueinander in Beziehung stehen.

Engagiert für die Verbreitung effektiver, anpassbarerer, nachhaltiger und sicherer Methoden innerhalb und zwischen Organisationen, bietet die BEI Führung, Ausbildung und Förderung von Praktiken der Geschäftsökologie und eine Plattform für die Entwicklung und den Austausch von Fallstudien und Praktiken zwischen Branchen- und Technologieführern.

Als das BPM/SOA-Konsortium der BEI beitrat, tat es das auf einer Ebene mit GCIO, CyberSecurity, Event Processing, und weiteren Interessensgruppe mit klaren Fachbereichen unter dem Schirm der BEI.

In den letzten Jahren hat die OMG weitere nicht-standardisierungsbezogene Organisationen gegründet, wie zum Beispiel das *Cloud Standards Customer Council (CSCC)*, eine Anwender-Interessensgemeinschaft mit dem Ziel, erfolgreiche Cloud-Einführungen zu fördern und sich um Standardisierungs-, Sicherheits- und Interoperabilitätsfragen im Rahmen einer Cloud-Adaption zu kümmern. Darüber hinaus gründete die OMG das *Consortium for IT Software Quality (CISQ)* – eine Gruppe von IT-Führern mit dem Ziel, einen Standard von berechenbaren Metriken für die Messung von Softwarequalität und -größe zu entwickeln.

CSCC und CISQ werden nicht die letzten nicht-standardisierungsbezogenen Bewegungen der OMG bleiben. Weitere spannende Gruppen werden folgen, die in alle möglichen Bereiche von Technologie und Management investieren und Informationen dazu bereitstellen. Folgen Sie daher diesem Umfeld: Mit Sicherheit wird das OBJEKTspektrum zu den ersten gehören, die über die Organisationen sowie ihre Bedeutung berichten.

Die OMG hat sich in den letzten zwei Jahrzehnten von ihrem Kernfokus auf objektorientierten Standards, wie CORBA, hin zu anpassbaren Standards für die Bedürfnisse von vertikalen Märkten entwickelt und betrachtet nun das größere Gesamtbild, in dem sich Standards bewegen. Bei dieser enormen Weiterentwicklung des OMG-Fokus in den ersten 20 Jahren wird es spannend zu sehen, wie sich der Fokus in den nächsten 20 Jahren zur Anpassung an sich ändernde Technologien weiterentwickeln wird.

Gratulation an das OBJEKTspektrum zu den ersten 20 erfolgreichen Jahren! Wir bei der OMG freuen uns darauf, auch die nächsten 20 Jahre mit Ihnen bei der Einführung von OMG-Standards in den DACH-Ländern zusammenzuarbeiten.

Besuchen Sie die OMG-Website unter **`www.omg.org`** und entdecken Sie, woran die OMG-Mitglieder arbeiten. Lesen Sie die vollständigen OMG-Spezifikationen und erfahren Sie, wie Sie selbst Mitglied werden können. ||

Übersetzung aus dem Englischen: Benjamin Klatt (klatt@fzi.de), wissenschaftlicher Mitarbeiter am FZI Forschungszentrum Informatik.

der Autor

|| Dr. Richard Mark Soley
(soley@omg.org)
ist Chairman und Chief Executive Officer der Object Management Group (OMG). Er begann 1989 bei der OMG als Technical Director und leitete die Entwicklung des weltweit führenden Standardisierungsprozesses und der CORBA-Spezifikation. Später leitete er die Ausweitung in vertikale Marktstandards, die zur ersten UML führte.

20 Jahre Objektorientierung

Was ist aus den Versprechungen vor 20 Jahren geworden, die wir uns mit der Einführung der Objektorientierung gegeben haben? Wo stehen wir jetzt und wo geht es hin?

Erinnern Sie sich noch an die 90er Jahre? Es war die Zeit der Methodenblüte: Peter Coad, Ed Yourdon, Jim Odell, Sally Shlaer, Stephen Mellor, Rebecca Wirfs-Brock und viele andere, vor allem natürlich die „Amigos" Grady Booch, Jim Rumbaugh und Ivar Jacobsen, hatten ihre Methoden (meistens waren es eher Notationen) publiziert. Einige basierten bereits auf der damals noch weniger verbreiteten objektorientierten Programmierung. Obwohl die eigentlich gar nicht mehr so neu war und die meisten wesentlichen Elemente und Konzepte Anfang der 70er Jahre im Xerox PARC und der dortigen Entwicklung der Sprache Smalltalk vorhanden waren. Und zu Hause zog die Objektorientierung mit Turbo Pascal 5.5 ein.

Abb. 1: Gehört der Vergangenheit an: Das Ausdruckens des vollständigen Programmcodes auf endlosen Papierbergen.

(foto: www.fotolia.de)

Objektorientierung als Anthroposophie

Kaum hatte ich die so genannte strukturierte Programmierung kennengelernt, konnte ich mich in lästerlicher Weise von den Spaghetti-Programmierern abgrenzen, zu denen ich gerade selbst noch gezählt hatte. Doch die Trennung von Daten und Funktionen ließ vor allem größere Programme schnell unübersichtlich werden, was Programmieren zu einer heroischen Leistung machte. Die Objektorientierung versorgte uns dann mit der Hoffnung und dem Glauben, jetzt viel „natürlicher" zu programmieren. Die damals sehr faszinierende Idee war, die Objekte so, wie wir sie in der realen Welt erleben, modellhaft nachzubauen: also so, dass Objekte direkt miteinander kommunizieren, sich Nachrichten senden und die Objekte autonome Einheiten bilden – mit eigener Identität, mit internen Daten, mit nach außen explizit offen gelegtem Protokoll. Und dann noch diese Möglichkeiten zur Vererbung, also auch eine Übertragung von natürlichen, biologischen Prinzipien auf die Programmierung.

Durch das Konzept der Polymorphie konnten wir demonstrieren, dass der Sender einer Nachricht nicht wissen kann und muss, wie der Empfänger diese verarbeitet, was sich ein wenig wie zwischenmenschliche Kommunikation anfühlte. Für die *Early Adopter* waren diese Neuerungen spektakulär genug, um Nachfolgenden das große Wort Paradigmenwechsel vorzuhalten (vgl. [Oes13]) – ohne zu merken, wie löchrig deren eigene Adaption war, wie Ralf Westphal dies gerade mit der Herausarbeitung des feinen Unterschiedes zwischen Methodenaufruf und Nachrichtenaustausch in seinem Beitrag „Biologiestunde" (vgl. [Wes13]) beschrieben hat.

Paradigmenwechsel spart Papier

Ein Versprechen war also, dass jetzt alles irgendwie natürlicher, einfacher und mächtiger wird. Und was ist tatsächlich daraus geworden? Die Grundidee, dass autonome kooperierende Objekte, die sich noch nicht einmal direkt kennen müssen, miteinander geordnet kommunizieren können, ist zwar prinzipiell möglich, aber sicherlich nicht bestimmend für das, was wir aus der Objektorientierung gemacht haben.

Die meines Erachtens größte praktische Veränderung war das Aussterben des „Listings", also des Ausdruckens des vollständigen Programmcodes. Die Objektorientierung sorgte für eine andere Form der Anordnung und Verteilung des Codes. Vorher wurden mit leistungsfähigen Nadeldruckern hunderte und nicht selten tausende Seiten Code auf zickzackgefaltetem Endlospapier ausgedruckt und mühevoll durchstöbert, um den Programmablauf nachzuvollziehen, weil man nicht genau sagen konnte, wo sich der jeweils relevante Code befand. Durch die Objektorientierung kam es zu einer Strukturierung des Codes, die das Auffinden und Nachvollziehen eines möglichen Programmablaufs erheblich vereinfachte – obwohl und gerade, weil der Code viel kleinteiliger war.

Damit hatte sich auch der Dokumentationsbedarf für den Code geändert. In der vor-objektorientierten Zeit haben wir in Kommentaren das Zusammenwirken der willkürlich verstreuten und angeordneten Codesequenzen erklärt. Heute erklärt sich der Code im Kleinen vielmehr von selbst, unterstützt durch Testfälle, Verweise auf Entwurfsmuster und einiges mehr. Und im Großen nennen und erklären wir eher Architekturprinzipien und -entscheidungen.

Entwurfsmuster markieren ebenfalls einen Übergang: Irgendwie sind es Regeln und konkrete Handlungsanleitungen oder wurden als solche verstanden. Wie Kochrezepte wurden sie zu Katalogen zusammengefasst. Als Entwickler muss ich zwar die Entscheidung treffen, ob und welches Entwurfsmuster für eine Situation richtig ist, danach folge ich der Anleitung aber einfach. Andererseits sind Entwurfsmuster auch schon als Prinzip oder Idee beschrieben worden, also nur als prinzipieller Lösungsweg, deren konkrete Ausprägung und Umsetzung

ich als Entwickler selbst herausfinden und entscheiden muss.

Abstrakter Denken

Damit wären wir bei der meines Erachtens größten intellektuellen Errungenschaft der Objektorientierung: Sie hat uns zu erheblich höheren Abstraktionen befähigt und dadurch zur Entwicklung deutlich komplexerer Anwendungen. Dieser Nutzen ist so essenziell, dass Objektorientierung eine so selbstverständliche Grundlage ist, dass wir sie fast gar nicht mehr bewusst bemerken.

Durch die Abstraktionen überblicken wir nicht mehr alle Details. Wir sehen und kennen nicht mehr den kompletten Code, sondern nur noch die im aktuellen Kontext relevanten Segmente und Bedeutungen. Es geht mehr darum, was gerade passt und dass es funktioniert, als dass die Gesamtheit konzeptionell konsistent ist. Mit der Objektorientierung sind dann auch Analyse- und Designmethoden und -notationen populär geworden, allen voran die *Unified Modeling Language (UML)*.

Zwischen 1995 (UML 0.9) und 2005 (UML 2.0) gab es noch viele praxisrelevante Änderungen. Seit Oktober 2013 liegt die Version 2.5 vor, wobei die Änderungen der letzten Jahre hauptsächlich hinter den Kulissen stattfanden. Ein paar grundlegende Konzepte und Notationselemente der UML, zum Beispiel das Klassendiagramm, gehören zum allgemeinen Handwerkszeug, wobei die häufigsten Modellierungswerkzeuge Papier und Stift sind. (Ich erinnere mich, dass die Firma oose Anfang der 2000er Jahre zum Spaß eine Anzeige im OBJEKTspektrum geschaltet hatte, in der wir einen Bleistift anpriesen, der schon die nächste UML-Version unterstützte.) Die Anzahl der gegenwärtigen Absolventen von UML-, SysML- und BPMN-Zertifizierungen und das Interesse an Anforderungs- und Modellierungsmethoden belegen deren anhaltende Relevanz.

Neben dem „Was“ wurde dann aber auch

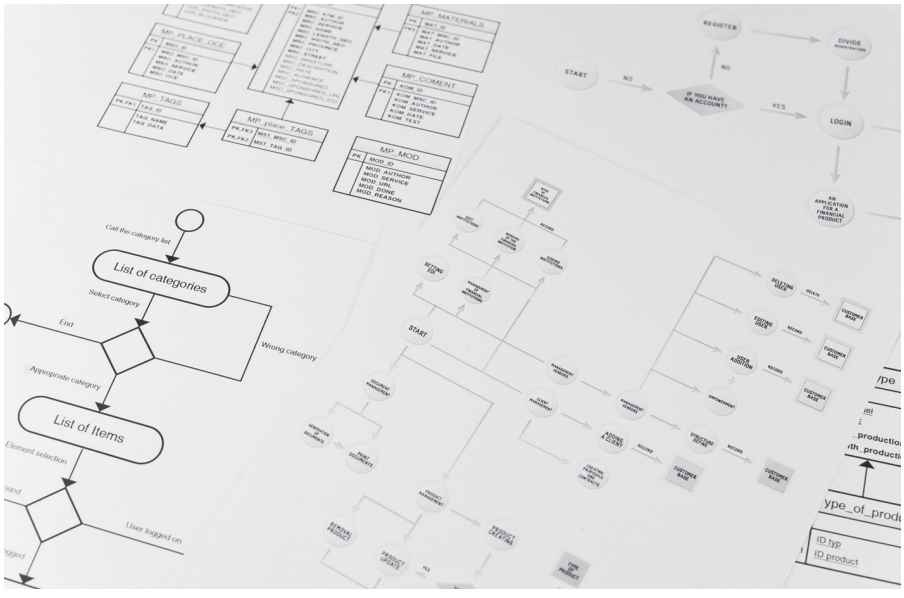


Abb. 2: Mit der Objektorientierung begann auch der Siegeszug der „Unified Modeling Language“ (UML). (foto: www.fotolia.de)

das „Wie“ immer wichtiger. Hier haben *eXtreme Programming* und nachfolgend die agile Bewegung sowie Scrum Impulse gegeben. Diese Bewegungen führten uns vom „Was“ zum „Wie“: Wie wollen wir entwickeln und arbeiten?

Wo stehen wir heute? Die Frage nach dem „Wie“ ist immer noch sehr präsent, allein schon deswegen, weil Agilität zwar zum Mainstream gehört und ein Standardvorgehen ist – aber eben noch keine vollständige Durchdringung und Selbstverständlichkeit erlangt hat, wie vielleicht die Objektorientierung. Das lässt sich daran erkennen, dass wir über Agilität und Scrum noch immer reden.

Meta-Denken

Mittlerweile geht es nicht mehr um das „Was“ und auch nicht mehr allein um das „Wie“, sondern mehr um das „Warum“. Immer häufiger enden Diskurse bei der Frage, aus welchen Überzeugungen heraus wir entscheiden und handeln und welchen Werten wir folgen. Prinzipien erfordern, anders als Regeln und Kochrezepte, viele eigene Entscheidungen. Vor allem, wenn sie sich widersprechen: Stabilität versus Anpassbarkeit, Abhängigkeit versus lose Kopplung, schnelle Lösung versus Sicherheit etc. Diese neue Evolutionsstufe der Softwareentwicklung fordert andere kommunikative und technische Fähigkeiten: Sie führt uns ebenso zu *Soft Skills* (Moderation, Kommunikation, *Facilitation* usw.) wie zur *Craftsmanship*-Bewegung. Wir müssen uns darüber verständigen, welche Denkmodelle, Überzeugungen und Werte uns leiten, weil das über die Wahl der passenden Abstraktionen und über die Wahl des „Wie“ entscheidet.

Nicht ganz zufällig erhalten Typologien und Quadrantenmodelle gerade mehr Aufmerksamkeit. Wir beziehen die zweite Ordnungsebene in unseren Alltag ein. Wollen wir eher *probieren* (weil wir denken, das Problem ist komplex) oder eher *analysieren* (weil wir denken, das Problem sei kompliziert)? Wir beginnen, Denkmodelle und Überzeugungen auszuhandeln, so wie wir vor 20 Jahren ausgehandelt haben, ob ein Rechteck eine Unterklasse von Quadrat ist oder umgekehrt. Wir entscheiden, wie wir denken wollen und welche Realität wir kreieren möchten. Dann bedienen wir uns bei allem, was wir schon davor kannten: von Kanban bis zur Objektorientierung. ||

Literatur

[Oes13] B. Oestereich, Objektorientierte Softwareentwicklung – von der Analyse bis zur Spezifikation, Oldenbourg-Verlag, 1. Auflage 1995 (in der mittlerweile 11. Auflage von 2013 (mit Koautor Axel Scheithauer) heißt das Buch „Analyse und Design mit der UML 2.5“)

[Wes13] R. Westphal, Biologiestunde, in: dotnetpro 11/2013

der Autor

|| Bernd Oestereich
(bernd.oestereich@oose.de),
Gründer der oose Innovative Informatik, war einmal OO- und UML-Pionier und -Experte und ist heute Autor und Sprecher zu innovativen Organisations- und Führungsthemen.

Agil seit mehr als 20 und in mehr als 20 Jahren

Um die Jahrtausendwende gab es den Begriff „Agile Softwareentwicklung“ noch nicht. Er entstand erst 2001 mit dem Agilen Manifest. Und in 20 Jahren? Was wird dann in aller Munde sein, zumindest bei den Softwareentwicklern (und dann vielleicht auch mehr Entwicklerinnen als heute)? Wird es die Softwareentwicklung überhaupt noch als bedeutende Disziplin geben?

Eingeführt wurde der Begriff „Software“ vom Statistiker John W. Turkey erst im Jahre 1958. Damals gab es auch noch jede Menge Schriftsetzer (mein Vater war einer), die an beeindruckenden Linotype-Setzautomaten arbeiteten. Mit Lochkarten und Lochstreifen gesteuerte Systeme waren damals schon lange gebräuchlich, etwa für Webmaschinen und Drehorgeln. Physische Lochkartonware also, aber keine Software.

Wie schnell sich doch alles ändert

Der Begriff „agil“ erscheint bereits in den 50er Jahren, acht Jahre vor dem Begriff „Software“, bei Talcott Parsons, einem zusammen mit Niklas Luhmann wichtigen Vertreter des soziologischen Systemtheorie, als Akronym für vier überlebenswichtige Funktionen von Systemen (vgl. [Par51]), die vier Teilsysteme (Verhaltenssystem bis kulturelles System) begründen:

- *A: Adaption (Anpassung):* Fähigkeit, auf Veränderungen in der Umwelt zu reagieren -> Verhaltenssystem.
- *G: Goal Attainment (Zielverfolgung):* Fähigkeit, Ziele zu verfolgen -> Persönliches System.
- *I: Integration (Eingliederung):* Fähigkeit, Zusammenhalt und Einschluss herzustellen -> Soziales System.
- *L: Latenz (Aufrechterhaltung):* Fähigkeit zur Erhaltung der Identität des Systems, obwohl alles stetig im Wandel ist -> Kulturelles System.

1970 wies Winston W. Royce in dem bei IEEE publizierten Artikel „Management of Large Software Systems“ darauf hin, dass ein „wasserfallartiges“ Vorgehen ungeeignet sei, und empfahl stattdessen ein iteratives Vorgehen mit sehr vielen Rückkopplungsschleifen unter Einbeziehung auch des Benutzers. Allerdings verfolgte er – dem Zeitgeist entsprechend – die Idee, für die Softwareentwicklung einen definierten und keinen sich erst ad hoc und navigierend ergebenden Prozess zu beschreiben, so wie das übrigens auch heute noch Scrum mit seiner klaren Abfolge von Ereignissen und Artefakten pro Iteration (= Sprint) tut.

Damals, ab 1970, gab es auch schon Vorgehensmethoden und Praktiken wie *Rapid Application Development* von Dan Gielan (in den 80er Jahren, bis 1991 formalisiert von James Martin, vgl. [Mar91]) und ebenfalls *Evolutionary Systems Delivery (Evo)* von Tom Gilb, heute von ihm *Evolutionary Project Management* genannt (vgl. [Gil]). Das Spiralmodell von Barry W. Boehm (vgl. [Boe88]), (**siehe Abbildung 1**) folgte 1986 und die Crystal-Methodenfamilie (vgl. [Coc08]) von Alistair Cockburn ab 1990.

Jeff Sutherland, John Scumniotales und Jeff McKenna entwickelten 1993 ein schlankes teambasiertes inkrementell-adaptives Vorgehen bei Easel Corporation und nannten es *Scrum*, inspiriert vom 1986 im Harvard Business Review publizierten Artikel „New New Product Development Game“ der Autoren Hirotaka Takeuchi und Ikujiro Nonaka (vgl. [Tak86]) – sehr lesenswert übrigens auch noch heute, insbesondere der letzte Abschnitt „Managerial implications“. Das war also vor nun bereits 20 Jahren.

Bei Ken Schwabers – heute noch existierender – Firma Advanced Development Methods (ADM) wurde in den frühen 90er Jahren ein ähnliches Vorgehen wie Sutherlands Scrum praktiziert und 1995 stellte

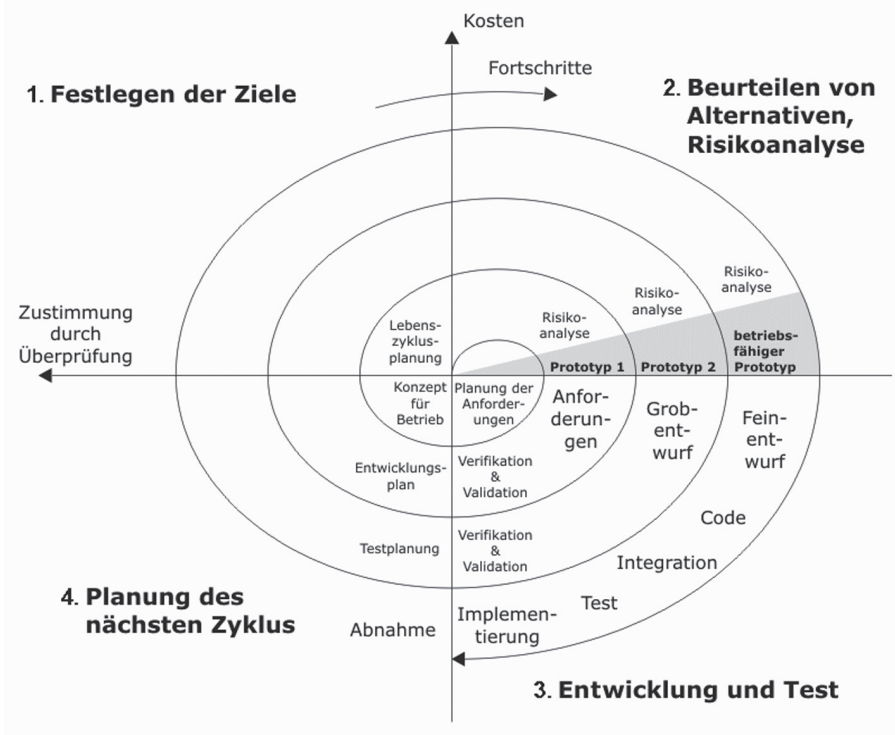


Abb. 1: Spiralmodell von Barry W. Boehm (GNU Free Documentation License).

Schwaber „Scrum“ in einem Workshop der OOPSLA‘95 (mit Jeff Sutherland als einem der Organisatoren) vor. Ken Schwabers Papier von damals ist auch aus heutiger Sicht aufschlussreich. Weiter unten komme ich darauf zu sprechen.
1996 verwendeten Kent Beck, Don Wells, Ward Cunningham und Ron Jeffries bei Chrysler erstmals *eXtreme Programming (XP)* (vgl. [Wel99]), wobei dieses Projekt leider abgebrochen wurde (vgl. [Wik]). Dennoch: Nach seiner Publikation 1999 fand XP rasante Verbreitung. Danach folgten Schlag auf Schlag (vgl. [Kor13-a]:

- 1996: *PRINCE2* (Nachfolger von PRINCE) als allgemeine inkrementell-adaptive Projektmanagement-Methode.
- 1997: Erste Publizierung von *DSDM*.
- 1997: Das *Feature Driven Development (FDD)* von Jeff De Luca.
- 1999: Das *Adaptive Software Development*, entwickelt von Jim Highsmith und Sam Bayer, ausgehend von RAD.

Jetzt war die Zeit reif für einen großen Wurf: Am 12. Februar 2001 trafen sich 17 Vertreter solcher – damals „leichtgewichtig“ genannter – Methoden und Vorgehensweisen in einem Skiressort in den Rocky Mountains mit dem Ziel, das sie Verbindende kompakt zu formulieren und dafür einen auch für Topmanager attraktiven Titel zu finden. Das Ergebnis war das heute bestens bekannte „Agile Manifest“ (vgl. [Bec01]).
Es ist jedoch nicht die bleibende Referenz für „agil“. Alistair Cockburn, einer dieser 17 Personen, erklärte in einer Diskussion am 16. September 2013, dass es ein Schnappschuss dieser 17 Leute aus der damals aktuellen Softwareentwicklungsszene heraus sei. Heute würden diese oder andere Exponenten es anders formulieren.
Etwa ab Mitte 2005 entwickelte sich Scrum rasch zur heute bekanntesten agilen Vorgehensweise der Softwareentwicklung auf Teamebene und fand dort mehr und mehr Einzug, auch bei großen Software-Entwicklungsfirmen. Dabei wurde zunehmend bewusst, dass ein Vorgehen, das auf den folgenden Faktoren basiert, ein oft allzu vereinfachendes, idealisierendes und starres Korsett darstellt und viele Problemzonen der Produktentwicklung und des Projektmanagements unbehandelt lässt (vgl. [Kor13-b]).

- Ein einzelnes kleines und über alle Fachkompetenzen verfügendes („interdisziplinäres“) Team.
- Fix getaktete „Sprints“ mit einer Dauer von jeweils nur 2 bis 4 Wochen, die jeweils praktisch nutzbare Teilprodukte zu liefern haben.
- Nur ein „Product Owner“ als einzige gegenüber dem Team alle Produktanforderungen vertretende und verbindlich entscheidende Person.
- Eine bereits vorhandene erste Version der das gewünschte Endergebnis beschreibenden Liste von Anforderungen (dem „Product Backlog“).

Zudem stellte und stellt sich heute immer wieder heraus, dass die mit der Einführung von Scrum verbundenen Verantwortlichkeits- und Organisationsänderungen insbesondere bei etablierten Unternehmen einen allzu großen und für viele Betroffenen in seiner Radikalität als nicht unbedingt erforderlich gesehenen Schritt bedeuten. Und es wurde deutlich, dass bei einer agilen Vorgehensweise auf Teamebene auch eine ihr entsprechende – also hoch adaptive – teamübergreifende Abstimmung auf Projekt-, Programm- und Portfolio-Ebene nötig wird.
Die Problematik starrer Sprints und der oft überfordernden Organisationsänderungen wird durch das erst seit 2010 breiter bekannte, von David J. Anderson für die Softwareentwicklung angepasste *Kanban* entschärft (vgl. [And11], [Sha13]). Es verbreitet sich derzeit rasch und wird auch zusammen mit *Scrum* zu *Scrumban* kombiniert.
Zur team- und projektübergreifenden agilen Abstimmung der projektunabhängigen fortlaufenden Produktentwicklung auf Programm- und Portfolio-Ebene tritt derzeit das *Scaled Agile Framework for Enterprises (SAFe)* von Dean Leffingwell stark in Erscheinung. Er hat es erstmals 2010 als Buch (vgl. [Lef10]) und als „Big Picture“ (vgl. [Sca13]) im Internet veröffentlicht. Für das agile Management von Projekten gewinnen DSDM und PRINCE2 als Rahmen für die Arbeit einzelner agiler Entwicklungsteams an Bedeutung.

Back to the Roots

Heute, 18 Jahre nach der OOPSLA‘95, an der Ken Schwaber sein Positionspapier „Scrum Development Process“ im „Business Object Design and Implementation Workshop“ (mit Jeff Sutherland und anderen als Organisatoren) präsentierte, ist es interessant, sein damaliges Paper nochmals zu lesen (vgl. [Sch95-a]).
Gleich zu Beginn des Papiers fallen mir diese zwei Sätze auf:

New axioms in systems development are:

- *Small teams of competent individuals, working within a constrained space that they own and control, will significantly outperform larger development groups*
- *Product development in an object-oriented environment requires a highly flexible, adaptive development process.*

Der erste Satz *(Small teams of competent individuals)* ist heute immer noch zentral. Wobei „competent individuals“ nach wie vor sicher der Schlüsselfaktor ist: Nicht ein amorphes, transdisziplinäres, selbstorganisierendes Team steht im Zentrum, sondern zentral sind „competent individuals“, die als Team arbeiten. Dieser Aspekt des Individuums scheint mir heute angesichts der Idealisierung der Teamarbeit manchmal unterbewertet zu sein.
Der zweite Satz *(Product development in an object-oriented environment)* ist bemerkenswert: OO wird hier als der eigentliche Treiber für einen „highly flexible, adaptive development process“ gesehen, und nicht die Unplanbarkeit und hohe Volatilität der Anforderungen, mit der heute die Notwendigkeit von Agilität vor allem begründet wird. Und noch interessanter ist ein weiterer Satz:

- *Object Oriented technology provides the basis for the Scrum methodology. Objects, or product features, offer a discrete and manageable environment. Procedural code, with its many and intertwined interfaces, is inappropriate for the Scrum methodology.*

Diese Formulierung sollte auch heute noch in allen Scrum-Guides erscheinen. Damit hätten wir uns viel Frustration erspart beim Versuch, Scrum auch in historisch gewachsenen, vom prozeduralen Denken dominierten IT-Landschaften mit ihren undurchsichtigen Schnittstellen einsetzen zu wollen, in welchen neue bzw. geänderte Funktionalität einen oft mehrwöchigen Integrationsprozess ins Gesamtsystem erfordert.
Und es wäre klarer, dass ein inkrementell-adaptives Vorgehen nur bei Produkten funktioniert, die aus Teilen (Objekten, Features) bestehen, die eigenständig und unabhängig voneinander handhabbar sind *(Objects, or product features, offer a discrete and manageable environment)*.

Erstaunt bin ich auch, wenn ich das lese:

- *Scrum is concerned with the management, enhancement and maintenance of an existing product […] Scrum is not concerned with new or re-engineered systems development efforts.*

Der Grundmechanismus von Scrum – welcher sich seit 1995 nicht verändert hat – zielt also nicht auf die (in der Regel als zeitlich begrenzte Projekte abgewickelte) Entwicklung neuer Systeme oder ihrer Rundum-Erneuerung, sondern auf die fortlaufende Weiterentwicklung bzw. Wartung von bestehenden Systemen als – gemäß Alistair Cockburn – „open-ended" oder „infinite games". Deshalb also fehlen in Scrum bis heute auch alle Rahmenvorgaben, z. B. für das Initialisieren einer Produktentwicklung bzw. eines Projekts, Rahmenvorgaben für die Erstellung eines initialen Backlogs oder für die frühzeitige Risikobewertung oder für die Erarbeitung einer initialen Rahmenarchitektur. Deshalb war und ist Scrum auch kein Projektmanagement-Framework, sondern kann – bei Projekten – nur im Rahmen eines PM-Frameworks, wie z. B. PRINCE2 oder DSDM, eingesetzt werden.
Interessantist auch, dass im „Ur-Scrum" der „Release" im Vordergrund steht (nicht seine einzelnen Sprints) und Ur-Scrum sich auf die Arbeit an jeweils einem Release in diesen drei Phasen (**siehe auch Abbildung 2**) konzentriert:

Scrum has three phases:

- *Planning: Definition of a new release based on currently known backlog.*
- *Development: Development of the new release functionality, with constant respect to the variables of time, requirements, quality, and competition. Interaction with these variables defines the end of development cycles.*
- *Closure: Preparation for release, including final documentation, pre-release staged testing, and release.*

Nur in der Phase „Development" erscheint beim Ur-Scrum der Begriff „Sprints" in dieser Art:

Development:

- *Meeting with teams to review release plans.*
- *Distribution, review and adjustment of product standards.*
- *Multiple sprints, until the product is deemed to be ready for distribution.*

Ergebnisse eines Sprints im Ur-Scrum sind integrierte „packets" als „demonstrable version of changes and how they implement backlog requirements". Sie sind jedoch nicht „ready for distribution". Erst der Release liefert ein „Product ready for distribution".
Das passt interessanterweise recht gut zum Konzept der Releases des *Scaled Agile Framework (SAFe)*: Erst die Releases liefern gemäß SAFe die „Potentially Shippable Increments". Die integrierten Ergebnisse der einzelnen Sprints innerhalb eines Releases werden gemäß SAFe in teamübergreifenden „System Demos" vorgeführt, stellen aber keine bereits benutzbaren Produktinkremente dar.
Zu den Phasen „Planning" und „Closure" ist in [Wes10]) zu lesen: „*[…] consist of defined processes, where all processes, inputs and outputs are well defined. The knowledge of how to do these processes is explicit. The flow is linear, with some iterations in the planning phase*". Erst das „Developmernt" ist hochgradig adaptiv: „*The Sprint phase is an empirical process […] Sprints are nonlinear and flexible.*"
Bezüglich „Rollen" nennt das Ur-Scrum von 1995 nur diese zwei Arten von Teams:

- *Management: Led by the Product Manager, it defines initial content and timing of the release, then manages their evolution as the project progresses and variables manifest. Management deals with backlog and issues.*
- *Development teams: Development teams are small, with each containing equal numbers of developers, documenters and quality control staff. Multiple teams of between three and six people may be used. Each is assigned a set of packets (or objects), including all backlog items related to each packet.*

Der „Product Owner" erscheint also bereits im Ur-Scrum, nicht aber primär als Einzelperson, sondern als „Product Manager", der ein Managementteam leitet, das sich als Team um das Backlog kümmert.
Im Ur-Scrum erscheint noch keine dem „Scrum Master" entsprechende Rolle – möglicherweise deshalb, weil das Ur-Scrum von „Small teams of competent individuals, working within a constrained space that they own and control" ausgeht, die so eine Rolle nicht (mehr) benötigen.

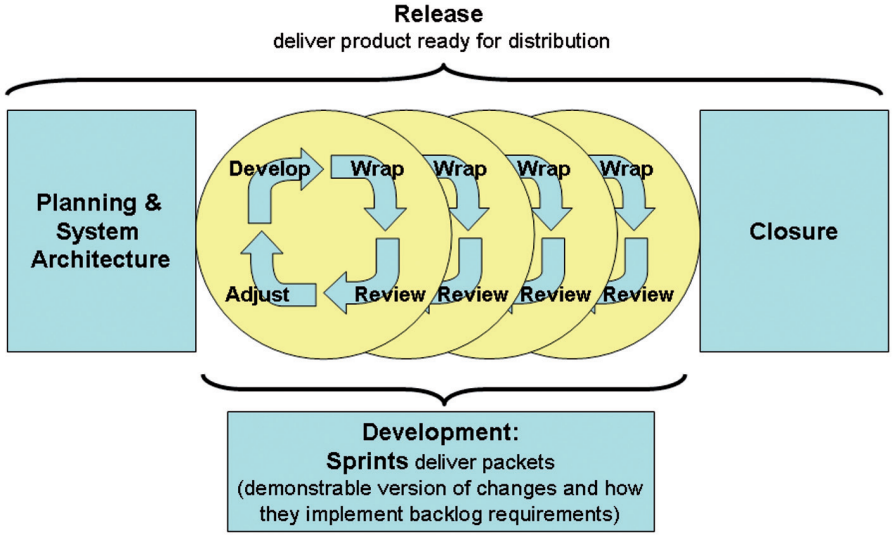


Abb. 2: SCRUM-Methodologie gemäß OOPSLA'95 (in Anlehnung an Abb. 6 in [Wes10]).

Die Zukunft gehört der agilen Agilität

Die Vielfalt der ab etwa 1970 entstandenen Vorgehensweisen, die dem iterativen, inkrementellen und adaptiven Prinzip folgen, sowie das rasche hype-artige Auftauchen und Verschwinden etlicher Methoden und Rahmenwerke sowie die noch sehr jungen Entwicklungen wie Kanban und SAFe legen den Schluss nahe, dass es nicht „die eine" und „für alles richtige" Vorgehensweise gibt. Stattdessen gibt es ganz unterschiedliche firmen- und produktspezifische Ausformungen agiler Vorgehensweisen – wobei sich viele davon als „Scrum-Vorgehen"

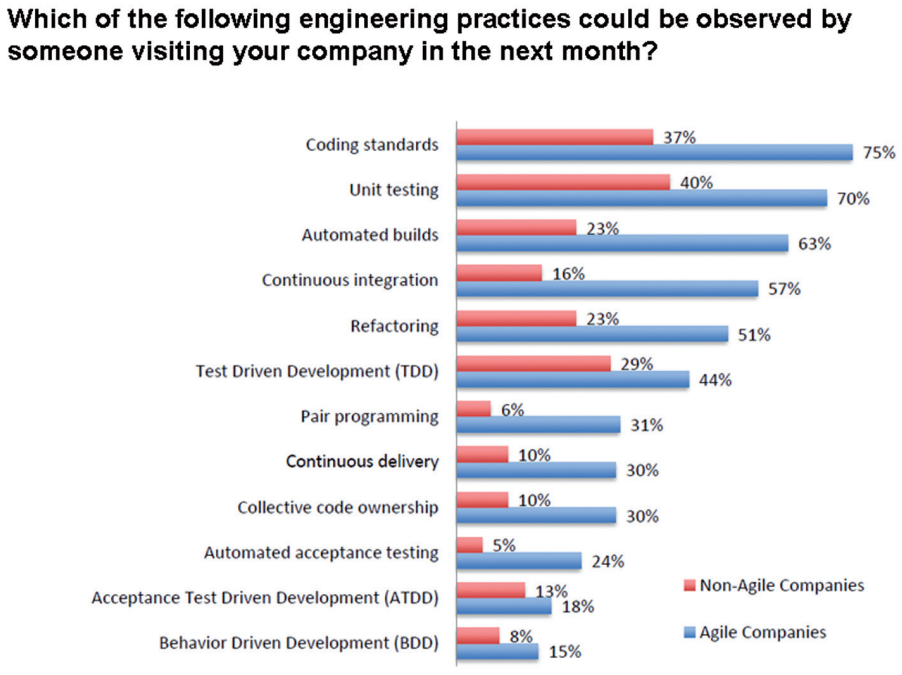


Abb. 3: Aktuell genutzte Engineering-Praktiken (Swiss Agile Study 2012, vgl. [Kro12]).

bezeichnen, weil sie einige Elemente von Scrum (die notabene oft von XP stammen) nutzen.

Dieser real existierende agile Umgang mit agilen Vorgehensmodellen wurde 2010 im Forrester Report „Agile Development: Mainstream Adoption Has Changed Agility“ (vgl. [Wes10]) als gängige Praxis in unterschiedlichsten Unternehmen beschrieben. Dort ist nachzulesen, dass sogar die am weitesten fortgeschrittenen Anwender agiler Methoden aus unterschiedlichen Vorgehensmodellen die für sie passenden Techniken heraus pflücken und sie mit anderen kombinieren. Die einzelnen Techniken nämlich (z.B. „Daily Standup“ und „Continuous Integration”) können unabhängig voneinander eingesetzt werden. Ein einzelnes agiles Vorgehensmodell könne nicht alles abdecken. Das zeigen auch die Ergebnisse der Swiss Agile Study 2012 (**siehe Abbildungen 3 und 4**).

Des Weiteren ist im Forrester Report zu lesen, dass die Kombination unterschiedlicher agiler Praktiken sehr gut funktioniert, um auch innerhalb längerer Release-Zyklen agil zu arbeiten, um den Anforderungen von Governance und Compliance Rechnung zu tragen und um mehr als nur die Lieferung funktionierender Software im Auge zu haben.

Wenn die von den jeweiligen Aufgaben und Herausforderungen abhängige freie Kombination diverser agiler Praktiken und Methoden durch die Teams selbst so verbreitet und auch erfolgreich ist, gibt es aber auch keine Notwendigkeit für ein alle Teams übergeordnetes agiles Vorgehen.

Es geht doch eher darum, die Teams zu befähigen, die jeweils passende Kombination diverser agiler Praktiken und Methoden selbst bestimmen zu können.

Und zwar so, dass den Teams der Rahmen dafür gesetzt wird, „*was*” sie zu leisten haben und dasa sie das „*Wie*” selbst bestimmen können. Wenn Teams von außen das Befolgen bestimmter Vorgehensweisen für das „Wie“ innerhalb des als „Was“ gesetzten Rahmens vorgeschrieben wird, dann ist das ein Widerspruch zum agilen Mindset.

Das Agile Management XX.00

Gehört die Zukunft den „agilen Organisationen“ – ganz unabhängig von der agilen Softwareentwicklung? Kommt also das „Management XX.00“?

Das hängt davon ab, was unter „agil“ verstanden wird. Unternehmen umfassen neben der – allenfalls inkrementell-adaptiv gestaltbaren – Entwicklung und Realisierung von Produkten auch viele Prozesse ohne Produktcharakter, wie etwa das Personalwesen. Unter „agil“ wird man daher nicht primär das inkrementell-adaptive Vorgehen der Produktentwicklung, sondern diverse post-tayloristische, hierarchie- und autoritätsfreie und selbststeuernd-kollaborative Prinzipien und Überzeugungen zu verstehen haben. All das sind jedoch in der Organisationsentwicklung schon lange bekannte und diskutierte Ideen. Einige davon wurden – allerdings in vergleichsweise wenigen Fällen – erfolgreich umgesetzt. Vielfach zitierte Beispiele sind unter anderem SEMCO (vgl. [Sem95]) und die Morning Star Company (vgl. [Mor]) sowie etliche Unternehmen, die den „Beta Codex” umzusetzen versuchen.

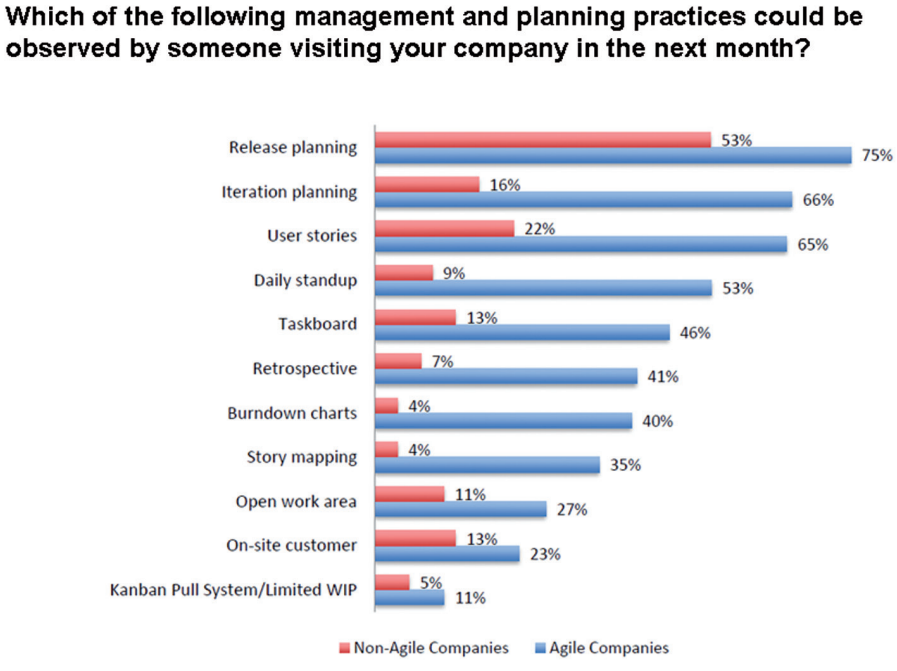


Abb. 4: Aktuell genutzte Management- u. Planungspraktiken (Swiss Agile Study 2012, vgl. [Kro12]).

Dass es angesichts all dieser seit Jahrzehnten existierender Ideen immer noch so wenige in dieser Art tickende Unternehmen gibt, liegt am damit verbundenen zu verändernden Denken und Handeln, beginnend bei der Unternehmensspitze, das durch Folgendes gekennzeichnet ist:

- Das „alles im Griff haben" wird ersetzt durch ein „vertrauensvolles Loslassen". Also: „Kontrolle ist gut, Vertrauen ist besser."
- Es herrscht die Überzeugung vor, dass ein Unternehmen erst durch die fortlaufende Konversation aller das Unternehmen umfassenden Mitarbeiter und Stakeholder entsteht, nicht durch von oben verkündete Visionen oder verordnete Strukturen.

Das zu leisten, ist sehr anspruchsvoll. Vielleicht gelingt es bis in 20 Jahren. ‖

der Autor

‖ Dr. Hans-Peter Korn
(hans-peter@korn.ch)
unterstützt als Coach und Berater der KORN AG größere Unternehmen dabei, die ihnen angemessene Form einer „agilen Agilität" zu entwickeln. Er ist Herausgeber der Bücher „Solution-Focused Management" und „Agiles IT-Management in großen Unternehmen" sowie Autor diverser Fachartikel.

Literatur & Links

[And11] D.J. Anderson, Kanban: Evolutionäres Change Management für IT-Organisationen, dpunkt.verlag 2011

[Bec01] K. Beck et al., Manifesto for Agile Software Development, siehe: **http://agilemanifesto.org/**

[Boe88] B.W. Boehm, A Spiral Model of Software Development and Enhancement, in: IEEE Computer, Vol. 21, Mai 1988, S. 61-72, siehe: **http://de.wikipedia.org/wiki/Spiralmodell**

[Coc08] A. Cockburn, Just-in-time methodology construction, 2008, siehe: **http://alistair.cockburn.us/Just-in-time+methodology+construction<**

[Gil] T. Gilb, K. Gilb, siehe: **http://www.gilb.com//Methods**

[Kor13-a] H.-P. Korn, Das „agile" Vorgehen: Neuer Wein in alte Schläuche – oder ein „Déjà-vu"? in: Series of the Gesellschaft für Informatik (GI), Volume P-224, 2013

[Kor13-b] H.-P. Korn, Hans-Peter (Hrsg.), Agiles IT-Management in großen Unternehmen, Symposion Publishing, 2013

[Kro12] M. Kropp, A. Meier, Swiss Agile Study 2012, Fachhochschulen Zürich und Nordwestschweiz, siehe:**www.swissagilestudy.ch**

[Lef10] D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs and the Enterprise, Addison-Wesley 2010

[Mal98] F. Malick, Der Mythos vom Team, in: managerSeminar Heft 33, Oktober 1998

[Mar91] J. Martin, Rapid Application Development, Macmillan USA 1991

[Mor] The Morning Star Company, siehe: **http://morningstarco.com/index.cgi?Page=Self-Management**

[Par51] T. Parsons, The social system, University of California Libraries 1951, siehe: **http://de.wikipedia.org/wiki/AGIL-Schema**

[Sca13] Scaled Agile Framework, 2013, siehe: **http://scaledagileframework.com/**

[Sch95-a] K. Schwaber, Scrum Development Process. Position Paper at OOPSLA '95, siehe: **http://www.jeffsutherland.org/oopsla/schwaber.html**

[Sch95-b] K. Schwaber, SCRUM Development Process (erstellt kurz nach der Präsentation an der OOPSLA'95), siehe: **http://agilix.nl/resources/scrum_OOPSLA_95.pdf**

[Sem95] R. Semler, Maverick: The Success Story Behind the World's Most Unusual Workplace. Grand Central Publishing; Reprint edition, April 1995

[Sha13] M. Shalloway, SAFe Kanban, 2013, siehe: **http://www.netobjectives.com/blogs/safe-kanban#!**

[Tak86] H. Takeuchi, I. Nonaka, The New New Product Development Game, in: Harvard Business Review, 1986, siehe: **http://hbr.org/1986/01/the-new-new-product-development-game/**

[Wel99] D. Wells, Extreme Programming: A gentle introduction, 1999, siehe: **www.extremeprogramming.org**

[Wes10] D. West, T. Grant, M. Gerush, D. D'Silva, Agile Development: Mainstream Adoption Has Changed Agility – Trends In Real-World Adoption Of Agile Methods, Forrester Report January 2010

[Wik] Wikipedia, Chrysler Comprehensive Compensation System, siehe: **http://de.wikipedia.org/wiki/Chrysler_Comprehensive_Compensation_System**

Über die Entwicklung von Entwicklungsorganisationen

Viel wurde in den letzten 20 Jahren über Vorgehensmodelle und Prozessmodelle geredet. Die Entwicklungsorganisation selbst wurde allerdings nie in ihrem Kern verändert. Mit wachsender Größe werden hierarchische Organisationen aufgebaut. Oft finden sich auch Matrixorganisationen mit Projektmanagement-Prozessen. Gibt es in den nächsten 20 Jahren für Entwicklungsorganisationen eine erfolgreiche Alternative dazu? Gerade durch die Impulse aus der agilen Welt wird intensiv darüber diskutiert. Aber können agile Prinzipien auch auf große Organisationen angewendet werden?

Zielsetzungen moderner Entwicklungsorganisationen

Softwareentwicklungsorganisationen stehen heute aus mehreren Richtungen unter Druck, wenn sie langfristig erfolgreich, also auch wirtschaftlich erfolgreich arbeiten wollen. Voraussetzung für diesen Erfolg ist natürlich ein Produkt, dessen Funktionalität und Gestaltung die Bedürfnisse im Markt trifft. Aber auch auf der Seite der Entwicklung können wir durch eine geeignete Organisation wesentlich zum Erfolg oder Misserfolg beitragen.

Zum einen muss das heute im Personalmarkt umworbene und relativ teure Personal möglichst effizient und effektiv eingesetzt werden. Gerade interne IT-Dienstleister arbeiten heute noch oft auf der Basis von Zeit und Material, sodass sie ihren Erfolg an der möglichst hohen Auslastung ihrer Entwickler messen.

Auf der anderen Seite wird *Time-to-Market* ein zunehmend wichtiger Erfolgsfaktor für Softwareorganisationen. Flexibilität und Reaktionsfähigkeit sind damit zu wichtigen Eigenschaften einer modernen Organisation geworden.

Schließlich spielt für den langfristigen Erfolg die Qualität der entstehenden Software eine entscheidende Rolle. Viele klassische Maßnahmen zur Qualitätssicherung kommen in Entwicklungsorganisationen unter Preis- und Zeitdruck aber immer wieder unter die Räder, weil sie eher reaktiv sind und zusätzlichen Aufwand bedeuten.

Don Reinertsen stellt in seinem Buch „The Principles of Product Development Flow“ (vgl. [Rei09]) das Ziel „Verkürzung der durchschnittlichen Verweildauer einer Anforderung im Entwicklungsprozess“ heraus. Wenn die Verweildauer kürzer ist, hat das nicht nur den Vorteil einer schnelleren Antwort auf Kundenbedürfnisse. Die kürzere Verweildauer führt auch dazu, dass wir mehr Anforderungen befriedigen können und dass die Effektivität und Effizienz der Organisation steigen. Und schließlich verbessert sich dadurch sogar die Qualität. Liegezeiten im System bedeuten nämlich immer „Verwesungszeiten“, weil sich währenddessen der Kontext, also die Schnittstellen, ändern können. Die Optimierung des Durchflusses ist deswegen eine Maßnahme, die alle wichtigen Ziele unserer Entwicklungsorganisation unterstützt.

Das Gerüst moderner Entwicklungsorganisationen

Die Betrachtung der Durchflussgeschwindigkeit lenkt unser Augenmerk direkt auf die Prozesse in unserer Entwicklungsorganisation. Es gibt unterschiedlichste Prozessmodelle für die Softwareentwicklung. Das *Capability Maturity Model Integration (CMMI)* (vgl. [Kne07]) beschreibt Reifegrade für eine Reihe von Prozessen. Dabei gibt es diese Prozesskategorien:

- Prozessmanagement
- Projektmanagement
- Engineering
- Support

Engineering umfasst die Kern-Entwicklungsprozesse von der Spezifikation über die Entwicklung bis zum Test. Im Projektmanagement werden die eher organisatorischen Prozesse Anforderungsmanagement, Planung, Risikomanagement und Projektmanagement als solches zusammengefasst.

Auch agile Ansätze beschreiben vergleichbare Prozesskategorien. Dean Leffingwell zum Beispiel unterscheidet die Prozessbereiche Portfoliomanagement, Programmmanagement und Entwicklung (**siehe www.scaledagileframework.com und Abbildung 1**). Er schließt damit den Support aus seiner Betrachtung aus.

Das Programmmanagement ist vergleichbar mit dem Projektmanagement für eine Produktorganisation, die Entwicklung auf Teamebene mit den CMMI-Engineering-Prozessen. Außerdem hebt er das Portfoliomanagement als strategisches Anforderungsmanagement aus dem Programmmanagement heraus. Schließlich ist für ihn der Ansatz von Don Reinertsen eine wesentliche methodische Grundlage zur Prozessverbesserung. Dies ist also das Gegenstück zum CMMI-Prozessmanagement.

Nicht die hierarchische Organisation, sondern die matrixartig dazu aufgespannten Prozesse bilden mit ihren Regelwerken für große Organisationen das Gerüst, das sehr viele Entwickler in vielen Entwicklungsteams zusammenhält. In Prozessen werden Rollen, Verantwortlichkeiten, Aufgaben und Kennzahlen beschrieben. Für Prozesse werden Praktiken vorgeschlagen und Werkzeuge, die den Prozess stützen. Aber gerade in der agilen Comunity sind solche Prozessansätze durchaus umstritten.

Die Problematik mit dem Teamschnitt

Die Prozesse des Projekt- oder Programmmanagements sind umso aufwändiger, je komplexer die Abhängigkeiten zwischen den Aufgaben der beteiligten Teams sind. Eine in Hinblick auf die entstehenden Abhängigkeiten zwischen Teams gewichtige Entscheidung ist deswegen, nach welchen Kriterien die Teams geschnitten werden.

Gerade in agil geprägten Organisationen wird die Strukturierung nach nutzerorientierten Funktionen (*Feature-Teams*, die komplette User-Storys und nicht nur Komponenten davon realisieren können) präferiert. Eine vollständige Gliederung in Feature-Teams minimiert natürlich die Abhängigkeiten zwischen Teams im Entwicklungsprozess, weil ja die Features als Liefer-

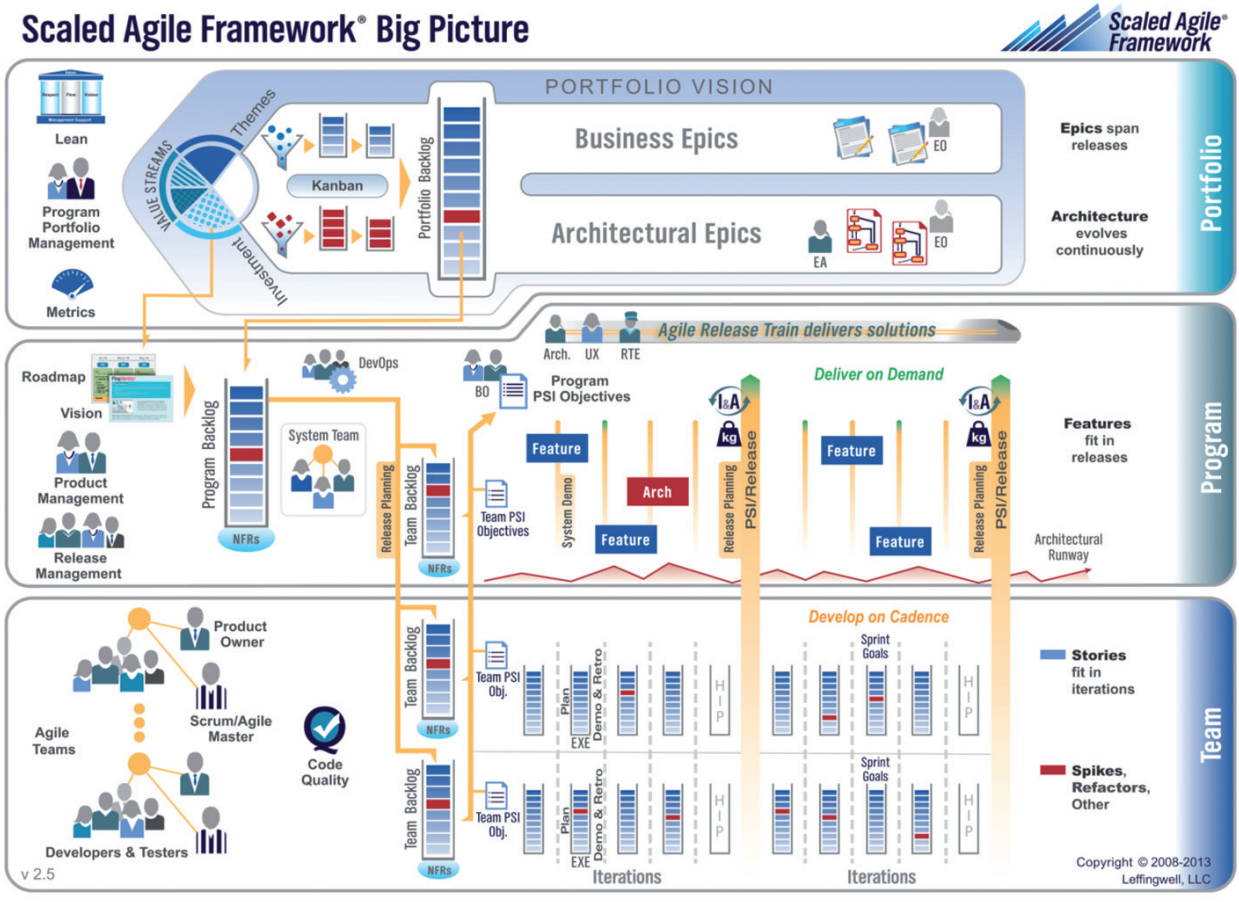


Abb. 1: Das Big Picture des Scaled Agile Framework.

einheiten eindeutig den Teams zugeordnet werden können.

In Organisationen mit einer ausgeprägten Architektur-Kultur wird aber durchaus erfolgreich eine in Schichten strukturierte Architektur als Blaupause für den Teamschnitt genommen. Gerade in sehr komplexen Systemen entstehen deswegen oft Teams für infrastrukturelle Standardkomponenten, wie Frameworks für GUIs und Persistenz, und Teams, die fachliche Basiskomponenten verantworten.

Abbildung 2 zeigt, dass gerade in den unteren Architekturschichten oft Standardlösungen verfügbar sind. Sie sind stabil, vielseitig einsetzbar und bieten hohe Qualität zu niedrigen Preisen, da sie vielfach genutzt werden können.

Die oberen Schichten sind dagegen sowohl für Softwareprodukt-Anbieter als auch für Individuallösungen entscheidend bezüglich der Funktionalität und Qualität verfügbarer Features. Hier entscheidet sich der Wettbewerb, hier muss die Entwicklungsorganisation schnell auf Bedürfnisse reagieren. Damit greifen alle Vorteile, die wir aus der Bildung von Feature-Teams ziehen können.

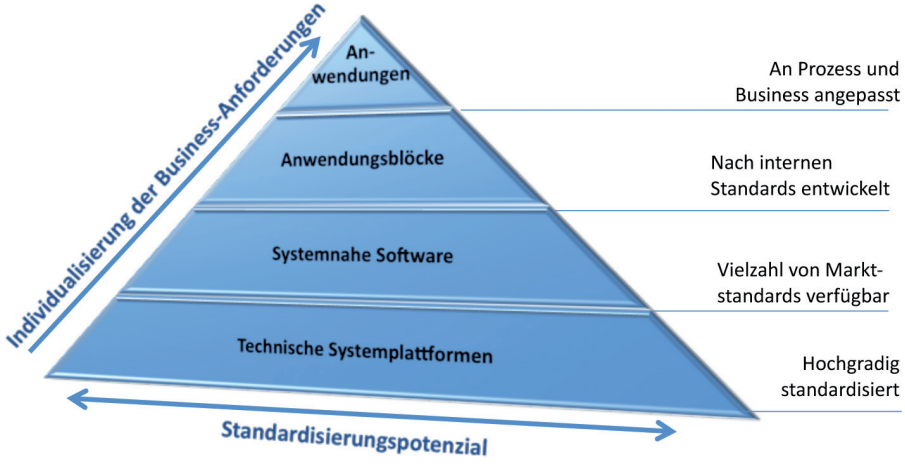


Abb. 2: Standardisierbarkeit von Architekturebenen.

Die Vision: Abkehr von großen Organisationen

Mary Poppendieck beschreibt in ihrem Artikel „Bevor es Management gab“ (vgl. [Pop13]) eindrucksvoll, dass Organisationen von mehr als 150 Menschen nicht so mehr effektiv arbeiten wie kleinere Organisationen, insbesondere dann nicht, wenn innovative und kreative Aufgaben eine große Rolle spielen. Sie leitet dies im Wesentlichen daraus ab, dass bei solchen Aufgaben die Kommunikation zwischen allen Beteiligten nur noch begrenzt möglich ist, was wohl in erster Linie auf kognitive Grenzen des Menschen zurückzuführen ist. Sie zeigt aber auch Beispiele aus der industriellen Praxis auf, wo große Unternehmen wie Apple oder Gore durch die Bildung wirtschaftlich und organisatorisch eigenständiger Einheiten von 100 bis 150 Leuten extrem erfolgreich agieren und den Aufbau lähmender Management-Zwischenschichten im Konzern vermeiden.

Interessanterweise kommt Dean Leffingwell bei der Definition seines Agile Release

Trains auf der Basis von Best Practices auch auf eine maximale Größe von 100 bis 150 Menschen, die innerhalb eines Trains die notwendige intensive Abstimmung ihrer Abhängigkeiten sinnvoll beherrschen können (**siehe Abbildung 1**). Über die Synchronisation mehrerer Release Trains äußert sich Leffingwell aber nicht wirklich.

Wenn wir deswegen davon ausgehen, dass wir mit 100 bis 150 Leuten eine Obergrenze für die sinnvolle Koordination und Kommunikation von Entwicklungsteams haben, auf der anderen Seite aber Anwendungslösungen heute häufig komplexer sind, als dass sie von so kleinen Gruppen bearbeitet werden können, müssen wir eine organisatorische Lösung finden, diese Gruppen zu managen.

Greifen wir die Idee von Mary Poppendieck auf, um das Management von Organisationen von 1.000 und mehr Menschen zu erklären, indem wir sie auf wirtschaftlich eigenständig handelnde Einheiten herunter brechen, die nur 100 bis 150 Menschen umfassen. Als Struktur dient uns die oben eingeführte Komponentenverantwortlichkeit. In Anlehnung an Leffingwells *SAFe (Scaled Agile Framework)* nennen wir im Folgenden diese für Komponenten verantwortliche Einheiten *Release Trains*, auch wenn wir keinen zwingenden Bezug zu den SAFe-Prozessen innerhalb der Trains herstellen wollen.

Die Trains einer großen Organisation bilden mit ihren angebotenen Services ein Wertschöpfungsnetz. Die Verwendung der Services eines Trains durch einen anderen Train begründet eine Auftraggeber-Auftragnehmer-Beziehung zwischen den beiden. Jeder Train trägt zur Wertschöpfung des Gesamtsystems bei und ist genau aus diesem Grund auch Teil dieses Systems. Für die Zusammenarbeit solcher Wirtschaftseinheiten gibt es also ein bewährtes System: den Markt.

In der idealen Vision einer marktorientierten Organisation treffen die für die einzelnen Trains verantwortlichen Train-Manager deswegen autonome Entscheidungen, welche Services anderer Trains verwendet werden sollten. Die Entscheidung beruht in dieser Idealvorstellung im Wesentlichen auf der Optimierung der eigenen Wertschöpfung. Das bedeutet, dass Services solcher anderen Trains verwendet werden, die zu möglichst niedrigen Kosten einen besonders hohen Nutzen versprechen.

Die Marktmechanismen bestehen in unserem Wertschöpfungsnetz also vor allem darin, dass ein Train-Verantwortlicher entscheiden muss, welche Services anderer Trains er nutzt, um seinen eigenen Services zu guter Qualität und zu niedrigen Kosten, d.h. mit hohem Wertschöpfungsanteil, bereitzustellen.

Ein Produktverantwortlicher für ein Anwendungssystem für Mittelständler steht beispielsweise vor der Entscheidung, das System auf Basis eines bislang eingesetzten 4GL-Werkzeugs zu realisieren oder Java-basierte Open-Source-Infrastrukturkomponenten zu verwenden. In dieser Entscheidung ist er tatsächlich autonom. Das heißt, er muss keiner mit Hoheitsrechten ausgestatteten IT- oder Architektureinheit folgen, sondern kann sich aufgrund einer eigenständigen Kosten-Nutzen-Analyse gegen das vorgeschlagene Framework entscheiden.

Trotzdem ist die Vorstellung, dass allein Marktmechanismen eine Anwendungslandschaft quasi automatisch optimieren, in der Praxis nur schwer haltbar. Durch die autonomen Entscheidungen einzelner Anwendungs- oder Train-Verantwortlicher werden jeweils in der lokalen Umgebung sehr gute Entscheidungen über die Verwendungen anderer Trains garantiert. In einer komplexen Architektur kann die lokale und vielleicht auch kurzfristige Optimierung von Architekturentscheidungen global betrachtet aber doch falsch oder zumindest unvollständig sein.

Eine wesentliche Rolle bei solchen Überlegungen spielt, ob und wie häufig ein Service voraussichtlich eingesetzt werden wird. Wird er in vielen Anwendungen eingesetzt, schafft er dort überall seinen Wertbeitrag und rechtfertigt damit eventuell hohe Erstellungs- und Wartungskosten. Wird eine Komponente nur in einer oder wenigen Anwendungen verwendet, muss die Erstellung und Wartung sehr viel günstiger sein.

Beim erstmaligen Design und Einsatz von Services eines Trains ist aber normalerweise nicht oder nur schwer abschätzbar, wie oft der Service noch eingesetzt werden wird und auf wie viele Kunden sich daher die Kosten für Entwicklung und Wartung verteilen lassen. Der Preis für den Service ist aber wieder genau von der Anzahl der Einsätze abhängig, und mit dem Preis natürlich auch der Wertbeitrag des verwendenden Service. Ebenso kann der Wert eines Service davon abhängig sein, dass er möglichst umfassend verwendet wird.

Deswegen kann das Management einer großen Organisation in den Markt regulierend eingreifen und die Verwendung strategischer Services subventionieren. Eine solche Subvention kann beispielsweise wirkungsvoll die Einführung neuer Technologien fördern, bevor diese sich im internen Markt eigenständig wirtschaftlich behaupten können.

Zusammenfassung

Wirklich große Organisationen können mit bekannten Prozessmodellen wie CMMI oder SAFe nur bis zu einer Größe von 100 bis 150 effektiv wirken, da die Softwareentwicklung nur auf der Basis einer relativ intensiven Kommunikation erfolgreich sein kann.

Über die Grenzen dieser Einheiten hinaus können Mechanismen eines regulierten Marktes mit ihren impliziten Kommunikationskanälen entlang der Auftragnehmer-Auftraggeber-Schnittstellen des Wertschöpfungsnetzes für die notwendige flexible Handlungs- und Entscheidungsstruktur sorgen.

Wahrscheinlich ist es nur eine Frage der Zeit und des nächsten wirklich prominenten Beispiels, dass ein solches Organisationsmodell die heutigen großen Entwicklungsorganisationen revolutionieren kann. ||

Literatur

[Kne07] R. Kneuper, CMMI, dpunkt.verlag (3.Auflage), 2007

[Pop13] M. Poppendieck, Bevor es Management gab: Warum die Dinosaurier der operativen Steuerung aussterben müssen, in: OBJEKTspektrum 1/2013

[Rei09] D.G. Reinertsen, The Principles of Product Development Flow: Second Generation Lean Product Development, Celeritas Publishing 2009

der Autor

|| Dr. Thorsten Janning
ist Management-Berater bei der KEGON AG in Wiesbaden und seit 2011 Chefredakteur des OBJEKTspektrum.

Quo vadis OO?

Die Entstehung des OBJEKTspektrum ist untrennbar mit einem Hype-Thema der 80er und 90er Jahre verbunden: der Objektorientierung. Über Zukunft und Vergangenheit der OO sprachen wir mit den IT-Meinungsführern Gunter Dueck, Wolfgang Keller und Bernd Oestereich.

Gunter Dueck,
Mathematiker, Wirtschaftsvordenker, Autor weltanschaulich-philosophischer Bücher und gefragter Redner, unter anderem bei der OOP 2014.

Wolfgang Keller,
freiberuflicher IT-Interimsmanager und Redaktionsmitglied des OBJEKTspektrum.

Bernd Oestereich,
OO-Pionier sowie Autor und Sprecher zu innovativen Organisations- und Führungsthemen.

Das OBJEKTspektrum wird 20 Jahre alt. Der Name war ja ursprünglich geprägt durch das Thema Objektorientierung – damals ein ähnliches Hype-Thema wie heute die Agilität. Wie haben Sie die Entwicklung der Objektorientierung in diesen 20 Jahren erlebt und wie schätzen sie die Bedeutung für die heutige Praxis in der Softwareentwicklung ein?

Gunter Dueck: Ich bin ja selbst gar nicht so sehr mit Softwareentwicklung befasst gewesen, allenfalls als Manager einer kleinen, feinen Optimierungsroutinen-Bibliothek meiner Abteilung aus den 90er Jahren. Wenn ich das alles aus einer gewissen Entfernung betrachte, fallen mir die Parallelen zu dem Wandel im Management auf. In den letzten 20 Jahren hat man „Lean Management", „Re-Engineering", „Prozessorientierung", Modularisierung" und „Oursourcing an Zulieferer" betrieben. So wie man bei OOP Objekte miteinander kommunizieren lässt, so arbeiten in der Realität Netze von autonomen Zulieferern zusammen, die nur über Input und Output spezifiziert werden. Es ist schade, dass die Welten der IT und des Managements so wenig Erfahrungen austauschen.

Wolfgang Keller: Objektorientierung habe ich als das „tägliche Brot der Softwareentwicklung erlebt" – jedenfalls für betriebliche Informationssysteme und User Interfaces. Nachdem ich vor allem mit betrieblichen Informationssystemen zu tun habe, war und ist Objektorientierung das Paradigma, in dem ich meistens denke und arbeite. Mit anderen Methoden zu arbeiten, ist in diesen Feldern oft schlicht unproduktiv.

Welche anderen Themen oder Technologien der letzten 20 Jahre hatten Ihrer Einschätzung nach eine ähnlich große Bedeutung für die heutige Anwendungspraxis?

Wolfgang Keller: Das kommt immer auch auf das Anwendungsgebiet an. Die Anwendung von Computern hat sich immer weiter ausgebreitet. Für Big Data benötigt man heute Algorithmen, die nur am Rande objektorientiert sind. Es verbreiten sich Methoden für parallele Programmierung und parallele Algorithmen, zum Beispiel für Suchmaschinen und Web-2.0-Anwendungen. Auch für funktionale Programmierung gibt es Nischen und Anwendungsgebiete und auch die Aspektorientierung wird praktisch eingesetzt. Als Softwareingenieur sollte man viele Lösungsmuster und -techniken kennen und nicht versuchen, mit einer Methode alle Probleme zu lösen.

Gunter Dueck: Management. Und noch eine Stufe tiefer: Die konsequente Anwendung der behavioristischen Psychologie des Stimulus-Response auf das Management, das sich nun über Incentives („Stimulus") und gewünschten Menschenoutput („Response") organisiert – das ist alles sehr nahe an den Ideen der IT. Ich bin darüber schon lange betrübt – über diese Sicht des programmierbaren Menschen, die uns heute schon oft zu Service-Robotern im Niedriglohn-Bereich gemacht hat. Management sollte heute wieder innovativ und agil werden. Das wird genauso wie in der IT diskutiert.

Lassen Sie uns einen Blick in die Zukunft werfen. Welche Themen oder welche Technologien werden die nächsten 20 Jahre wesentlich prägen?

Bernd Oestereich: Die Menschheit baut gerade ein gigantisches, global verteiltes Netzwerk, angetrieben durch eine ganze Reihe technischer und organisatorischer Entwicklungen, wie beispielsweise Internet of Things (IoT), Cloud-Computing, Industrie 4.0 und vernetzte Produktionssysteme, cyber-physische Systeme und autonome Objekte, generative Fertigungstechnologien und einiges mehr. Die zunehmende Vernetzung ermöglicht und fordert neue Geschäftsmodelle, neue Prozesse und neue Formen der Zusammenarbeit und Organisation. Es gibt kaum eine Disziplin, die nicht von dieser Revolution betroffen ist: Ad-hoc-Geschäftsprozess- und Anforderungsgestaltung, Systemarchitektur, dezentrale Steuerbarkeit, mobile Kommunikation, Echtzeitfähigkeit, Sensorik, eingebettete Systeme, Kontextsensitivität, User-Experience, Projektmanagement, agile Entwicklungsmethoden, Arbeitsorganisation, Selbstorganisation, unternehmensübergreifende interdisziplinäre Kooperation usw.
Deswegen beschäftigen wir uns bei oose mit so vielen verschiedenen Lösungsansätzen aus diesem Kontext: Adaptive Case Ma-

nagement, agiles Projektmanagement, Software-Craftsmanship, agile Architekturen, funktionale Architekturen, agiles BPMN, dynamische Organisationen, Design Thinking, Innovationskultur, UX – alle diese Ansätze gehen in neuer Weise mit Komplexität um. Entsprechend wird uns auch die Frage antreiben, wie diese verschiedenen Disziplinen in diesem revolutionären Kontext konstruktiv und wegweisend zusammenspielen. Die entstehende Komplexität und Dynamik, der Umgang mit dem Unplanbaren und Unvorhersehbaren sowie der Verlust kausaler Zusammenhänge sind dabei die größten Herausforderungen, für die wir viele neue Ideen, Praktiken, Methoden und Theorien (mit-)entwickeln und anbieten.

Gunter Dueck: Es werden immer vernetztere und integriertere Infrastrukturen entstehen – wie man die wohl handhabbar strukturiert, sodass sie einen Wandel aushalten? Zum Beispiel: Automatischer Verkehr mit lauter selbstfahrenden Autos? Ich befürchte auch eine elektronische Aufrüstungsspirale – es geht doch derzeit wohl nicht bloß um Spionage, sondern um das Üben elektronischer Kriege?

Wolfgang Keller: Aus meiner Sicht werden eher die Anwendungsgebiete die Sprachen und Methoden treiben als umgekehrt ein Programmierparadigma die Anwendungen. Bei den Anwendungen sind weitere Miniaturisierungen absehbar. Die Hardwareentwicklung wird weiter viele neue Anwendungen möglich machen. So genannter Smart-Dust wird schon entwickelt und in 20 Jahren sicher verfügbar und ubiquitär sein. Ebenso Mini-Drohnen oder direkte Körper/Maschine-Schnittstellen. Das Internet der Dinge wird sich weiter ausbreiten – und damit leider vermutlich auch eine nie dagewesene Überwachung und weitere militärische Anwendungen. Die Programmierparadigmen dafür sollten weitgehend vorhanden sein. Im Gebiet virtueller Realität und räumlicher Darstellung ist sicher noch viel zu tun. Ebenso auf dem Gebiet von Cyborgs (Mensch/Maschine-Kopplung).

Die Sicht auf die Softwareentwicklung schwankt ja immer zwischen dem Streben nach Anerkennung als Ingenieur-Disziplin und der Betonung des kreativen Elements. Welche der beiden Positionen vertreten Sie eher? Und wie wird sich die öffentliche Diskussion in den nächsten 20 Jahren entwickeln?

Wolfgang Keller: Man kann nur beide Positionen vertreten – abhängig vom Anwendungsgebiet. Im Bereich betrieblicher Informationssysteme oder zum Beispiel auch auf Gebieten wie Smart Metering oder Steuerung von Infrastrukturen werden Compliance, IT-Sicherheit und Zuverlässig eine immer dominantere Rolle spielen. Der Grad der Verwundbarkeit, der dort heute schon vorhanden ist, ist eigentlich nicht tolerabel – und er wächst derzeit noch stetig. Für das jeweilige „Next Big Thing" wird es auch immer Visionen und Kreativität geben müssen, wenn für das jeweils zu entwickelnde Objekt noch keine langfristig bewährten Technologien zur Verfügung stehen. Es wird also immer beides geben: Engineering und Innovation.

Bernd Oestereich: Obwohl ich selbst Diplom-Ingenieur bin, arbeite ich genauso selbstverständlich auch als Kreativer. Deswegen halte ich die Frage für falsch. Es geht nicht mehr darum, die Aufmerksamkeit auf die Angrenzung von Ingenieuren und Kreativen zu legen, sondern um die Integration: Beide bringen unterschiedliche und gleichermaßen relevante Kompetenzen mit. Die Frage könnte meines Erachtens eher lauten: Welche anderen Disziplinen und Fähigkeiten sind noch zu integrieren, um in Entwicklungsorganisationen die passende Komplexität herzustellen?

Gunter Dueck: Ich vertrete eigentlich lieber nichts. Ich weise immer wieder daraufhin, dass es eine analytische Plansicht auf die Dinge gibt, dann eine kreativ-künstlerische des ersten Erschaffens und schließlich eine unternehmerische des Tatendranges im Praktischen. Ich selbst bin mehr Künstler, aber die drei Perspektiven bestehen allezeit nebeneinander und denken sich dann separat „Wasserfall-Modelle", „Disruptive Innovation" oder „Agilität" aus. Je nach der Stabilität der Weltlage „gewinnt" bei der Massendiskussion mal dies, mal das. Ich will gar nicht, dass etwas „gewinnt". Ich möchte, dass jeder „emphatisch" alle Sichten kennt und die zweckmäßige im gegebenen Augenblick einnimmt. ||

impressum

Verlag: SIGS DATACOM GmbH
Lindlaustr. 2c, D-53842 Troisdorf
Tel.: +49 (0) 2241/2341-100, Fax: +49 (0) 2241/2341-199
www.sigs-datacom.de · E-Mail: info@sigs-datacom.de

Chefredakteur: Dr. Thorsten Janning
Lindlaustr. 2c, D-53842 Troisdorf
Tel.: +49 (0) 2241/2341-100, Fax: +49 (0) 2241/2341-199
E-Mail: fachartikel@objektspektrum.de

Mitglieder der Redaktion:
Wolfgang Keller, object architects
Dr. Hans-Peter Korn, Korn AG
Dr. Johannes Mainusch, Otto GmbH & Co. KG
Prof. Dr. Andreas Rausch, Technische Universität Clausthal
Martin Schimak, plexiti GmbH
Rainer Singvogel, msg systems ag
Prof. Dr. Gerhard Wanner, Hochschule für Technik Stuttgart

Honorary Editorial Member: Dr. Frances Paulisch

Newsredakteur (Aus der Szene): Ulrich Schmitz
E-Mail: Ulrich.Schmitz@sigs-datacom.de

Schlussredakteurin: Kirsten Waldheim
E-Mail: schlussredaktion@objektspektrum.de

Freie Mitarbeiterin: Elke Niedermair, E-Mail: e.a.n@gmx.de

Manuskripteinsendungen: Manuskripte werden von der Redaktion – ausschließlich in elektronischer Form – gerne angenommen. Mit der Einsendung von Manuskripten gibt der Verfasser die Zustimmung zum Abdruck, zur Vervielfältigung auf Datenträgern und zur Übersetzung. Weitere Informationen und Honorarfestlegungen finden Sie in unseren Autorenrichtlinien: www.sigs-datacom.de/fachzeitschriften/objektspektrum/hinweise-fuer-autoren/richtlinien.html.

Verlagsleitung: Günter Fuhrmeister

Redaktions- und Herstellungsleitung Zeitschriften:
Susanne Herl, Tel.: +49 (0) 2241/2341-550,
E-Mail: Susanne.Herl@sigs-datacom.de

Leserservice: objektspektrum@sigs-datacom.de

Anzeigen: Brigitta und Karl Reinhart
Ostring 5h, D-85630 Grasbrunn,
Tel.: +49 (0) 89/464729, Fax: +49 (0) 89/463815,
E-Mail: Karl.Reinhart@sigs-datacom.de

Druck: F&W Mediencenter GmbH
Holzhauser Feld 2 · D-83361 Kienberg · Tel.: +49 (0) 8628/9884-52
E-Mail: anzeigendaten@objektspektrum.de

Abonnenten-Service: IPS Datenservice GmbH, Postfach 13 31,
D-53335 Meckenheim, Tel.: +49 (0) 22 25/70 85-374
Fax: +49 (0) 22 25/70 85-376, Patrick König, Markus Preis
E-Mail: aboservice@sigs-datacom.de

Erscheinungsweise: zweimonatlich

Bezugspreise:
Einzelverkaufspreis: € 9,90 (D), € 10,90 (A), sfr 18,20 (CH)
Jahresabonnement Deutschland: € 56,40 inkl. Versandkosten
Jahresabonnement Europa: € 62,40 inkl. Versandkosten
Jahresabonnement außerhalb Europas: € 72,30 inkl. Versandkosten
Studentenabo: € 50,80 inkl. Versandkosten

Bankverbindung:
Postbank München, Konto-Nr. 560 737 801, BLZ 700 100 80

Lieferung an Handel:
Verlagsunion KG, Postfach 5707,
D-65047 Wiesbaden, Tel.: +49 (0) 61 23/620-0

Layout und Produktion: leuchtfeuer-agentur · Sven Lesemann
Meisenweg 2 · D-51467 Bergisch Gladbach,
www.leuchtfeuer-agentur.de · E-Mail: Lesemann@leuchtfeuer-agentur.de

Coverbild: www.fotolia.de

Urheberrecht: Für Programme, die als Beispiele veröffentlicht werden, kann der Herausgeber weder Gewähr noch irgendwelche Haftung übernehmen. Aus der Veröffentlichung kann nicht geschlossen werden, dass die beschriebenen Lösungen oder verwendeten Bezeichnungen frei von gewerblichen Schutzrechten sind. Die Erwähnung oder Beurteilung von Produkten stellt keine irgendwie geartete Empfehlung dar. Für die mit Namen oder Signatur gekennzeichneten Beiträge übernehmen der Verlag und die Redaktion lediglich die presserechtliche Verantwortung.

© 2013 SIGS DATACOM GmbH
ISSN 0945-0491

Die Zeitschrift wird mit chlorfreiem Papier hergestellt.