Advertorial

Warum es in der Softwareentwicklung heute vor allem auf Offenheit und Erfahrung ankommt

„Wir sind keine Codeschubser!“

Interview mit Richard Attermeyer und Sven Bernhardt

Alle sprechen von Low Code, Continuous Delivery und Containertechnologien. In der Softwareentwicklung scheint es kaum noch um Programmierung zu gehen. DevOps ist das neue Zauberwort. Aber können wir Software wirklich am Fließband produzieren? Bauen Maschinen besseren Code als wir? Was geschieht mit den Entwickler:innen? Brauchen wir sie überhaupt noch? Und was machen wir mit Methoden wie Scrum, Kanban oder Software Craftsmanship? Wie passen diese in eine automatisierte Produktionsumgebung?

Richard Attermeyer und Sven Bernhardt beraten und managen seit vielen Jahren kleine, große und sehr große Softwareprojekte. Sie kennen die Sorgen der Kunden, aber auch den Ärger der Entwickler. Von ihnen wollten wir wissen, wie sie professionelle Softwareentwicklung und DevOps in der Praxis erleben. Was läuft gut? Was überhaupt nicht? Und was sollte sich unbedingt ändern?

Die gute alte Zeit

OBJEKTspektrum: Manche sprechen von der „guten alten Zeit“, wenn sie an Softwareprojekte zurückdenken, bei denen sie wochenlang mit zwei Kollegen an einem einzelnen Feature getüftelt haben. Sobald das fehlerfrei lief, wurde der Code zum Test freigegeben und das nächste Feature wartete. Wie die Softwarekomponente betrieben wurde, war kein Problem der Entwickler, darum kümmerte sich der Betrieb. Die Aufgaben waren klar abgesteckt. Heute müssen Security- und Konfigurationsanforderungen von Anfang an mitbedacht werden. Aber haben wir als Entwickler oder Softwarearchitekten wirklich Lust, uns mit solchen Fragen zu beschäftigen? Und können wir das überhaupt?

Sven Bernhardt: Die Zeiten, in denen man sich als Entwickler nur um die richtige Programmiersprache, ein passendes Framework und eine stabile Datenbank kümmerte, sind lange vorbei. Sich nur mit der Entwicklung beschäftigen und den Betriebsgedanken nicht mitdenken geht heute nicht mehr. Heute ist testgetriebene Entwicklung Standard. Man schreibt also Tests, gegen die man entwickelt.

Richard Attermeyer: Die Zyklen, in denen wir Software entwickeln, haben sich geändert. Ziel ist es, das Produkt schneller in den Betrieb zu bekommen, denn nur dort generiert es Mehrwert. Also genau dann, wenn es benutzt wird. Und auch erst dann bekommen wir ein echtes Feedback über die Qualität der Software. Hier zeigt sich dann sehr schnell der Unterschied zwischen „Working Software“ und „Professional Software“.

Sven Bernhardt: Wir haben verschiedene Thesen aufgestellt, wie und was sich in der Softwareentwicklung ändern wird. Eine wichtige These ist, dass wir eine DevOps-Kultur brauchen, um Professional Software zu entwickeln. D. h. Fehler sind dazu da, sich und das Produkt weiterzuentwickeln und auch mal Experimente zu wagen. Früh erkennen, Anpassungen vornehmen und im Team gemeinsam Lösungen entwickeln – das ist für uns die Grundvoraussetzung für eine erfolgreiche Entwicklung. Und das Team sollte sich immer fragen: Wofür wird das Produkt gebaut, und wird es in realen Situationen überleben? Nur dann geht man die richtigen Dinge an.

Richard Attermeyer: Ein anderes Prinzip lautet „Automatisiere (fast) alles“. Automatisierung hilft, komplexe Tools zu handhaben. Bewährte Lösungen helfen bei der Sicherstellung von Qualität. Wir nutzen leider immer noch zu selten Synergieeffekte zwischen den Projekten. Skripte, Policy Pipelines und von Experten vorkonfigurierte Plattformen senken die Fehlerquote.

Turmbauer zu Babel

OBJEKTspektrum: Als der Turm von Babel geplant wurde, waren die Erbauer hochmotiviert. Sie wollten mit diesem Projekt den Himmel erreichen. Doch ihre Teams machten ihnen einen Strich durch die Rechnung: Jedes brabbelte in seiner eigenen Sprache und brachte seine Fähigkeiten nicht mit den anderen zusammen. Das Ende kennen wir. Könnte diese Geschichte auch auf DevOps und große, verteilte Teams zutreffen? Wie bekommen wir die unterschiedlichen Fähigkeiten, Erfahrungen und Sichtweisen unter einen Hut? Und Effizienz in unsere Arbeit?

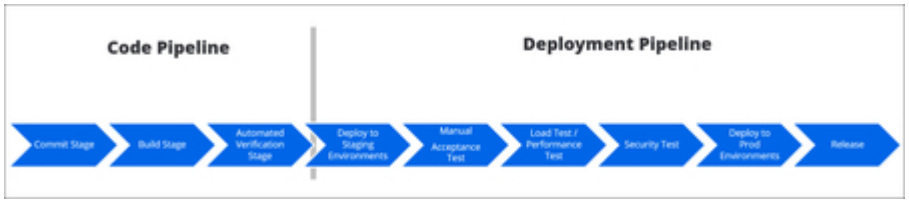

Abb. 1: Shift Left

Richard Attermeyer: Eine unserer Thesen heißt „Shift Left“. Um zu verstehen, was wir damit meinen, muss man die Development-Pipeline vor Augen haben. Auf der linken Seite gibt es die Code-Pipeline und auf der rechten die Deployment-Pipeline. Dort, wo in der Vergangenheit viele Aufgaben später, teilweise sogar manuell, angegangen wurden, wandern einige davon heute weiter nach links. Der Entwickler nimmt so auch verschiedene Rollen ein. Wer spezialisiert ist, kann eine Beraterrolle einnehmen.

Sven Bernhardt: Die Idee hinter „Shift Left“ ist es, recht früh Qualität ins Produkt einzubauen („Build quality in“), diese über die Pipeline zu messen und transparent zu machen. Security Tests werden heute beispielsweise viel früher und vor allem automatisiert durchgeführt. Das Scannen von Docker Images und Dependencies sorgt dafür, dass zum Zeitpunkt der Auslieferung keine bekannten kritischen Dinge in der Codebasis auftauchen. Eigene Fehler kann dies leider nicht verhindern. Aber da bekannte Fehler von Angreifern als erstes ausprobiert werden, hilft uns das weiter. Hier kann eine statische Codeanalyse helfen, typische Fehlermuster zu erkennen.

Richard Attermeyer: Als Entwickler müssen wir mehr Verantwortung übernehmen. Wenn wir zum Beispiel ein Trading-Portal entwickeln, würden wir es dann unserem besten Freund empfehlen? Oder, wenn wir es mit der Steuerung von Bremsen in Zügen zu tun haben, würden wir dann mit unseren Kindern in diesem Zug fahren? Wir sollten uns bewusst machen, dass Software heutzutage fast in allem enthalten ist. Und diese Software wurde fast immer von einem Team entwickelt. Wie ist das Team vorgegangen? Hat es funktionierenden Code entwickelt? Dann sollte uns das nicht ausreichen. Funktionierender Code und Working Software sind gut für Demos und PoCs, doch wir wollen das Vertrauen der Anwender gewinnen. Und Vertrauen gewinnen wir nur, wenn wir auf Qualität setzen.

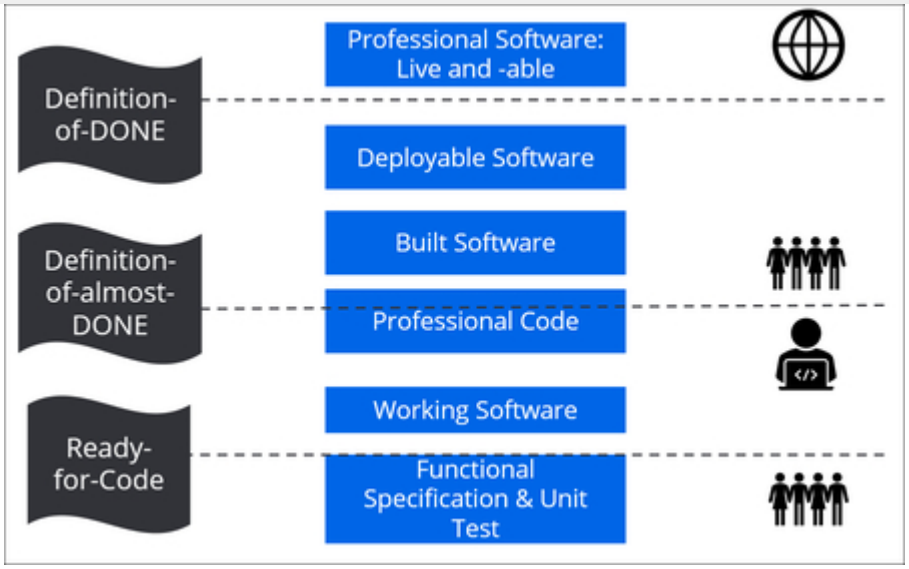

Abb. 2: Professional Software entwickeln

Sven Bernhardt: Das kann ich nur unterstreichen: Professional Software schaut nicht nur auf Features. Teams, die Professional Software erstellen, schauen vor allem auf Qualität und auf nichtfunktionale Aspekte der Software-Entwicklung. Sie stellen sich Fragen wie: Ist die Software testbar, nachvollziehbar, betreibbar, wiederherstellbar, skalierbar, überwachbar und so weiter. Und überall sollte die Antwort ja lauten. Das, was wir anstreben sollten, kennen wir seit längerem als Software Craftmanship. Und in heutigen Zeiten von DevOps ist die Bedeutung noch einmal gestiegen.

Richard Attermeyer: Ein weiterer Aspekt ist der Betrieb der Produkte, die wir entwickeln. Wenn wir ein Auto kaufen, wollen wir ja auch, dass es lange funktioniert und die Wartung einfach ist. Ich kann mit meinem Auto in verschiedene Fachwerkstätten fahren und alle wissen, was zu tun ist. Für meinen Geldbeutel wäre es außerdem sinnvoll, wenn die Werkstatt nicht immer das komplette System austauschen muss, wenn was nicht läuft. Bei unseren Applikationen sollten wir den gleichen Anspruch haben. Verständlichkeit und einfache Wartung heben die Qualität, erhöhen die Laufzeit und tragen so zur Zufriedenheit der Kunden und Anwender bei.

Zwei Welten

OBJEKTspektrum: Cloud versus On-Prem, Monolith versus Microservices, Agil versus Wasserfall – hier scheinen Welten aufeinander zu treffen. Gibt es zwei Welten? Und was ist, wenn mein Kunde, die „alte Welt“ gar nicht überwinden möchte? Oder koste es, was es wolle, in die neue Welt möchte? Welche Möglichkeiten haben wir, wenn es darum geht, Technologien und Verfahren im Hauruck-Verfahren zu modernisieren, sanfte Übergänge zu schaffen oder hybride Szenarien zu ermöglichen?

Sven Bernhardt: Wir sind überzeugt, wenn man einige Prinzipien befolgt, dann verbaut man sich nicht den Weg. Wenn wir neue Applikationen entwickeln, sollten wir immer den Cloud-Native-Ansatz fahren.

Richard Attermeyer: Das heißt jedoch nicht, dass diese Applikation auch in der (Public-)Cloud betrieben werden muss.

Sven Bernhardt: Genau. Im Grunde will ich eine Applikation bauen, die die Vorteile von Cloud optimal ausnutzen kann. Dabei geht es um Skalierbarkeit oder auch die Fähigkeit, die Anwendung auf unterschiedlichen Umgebungen auszuführen. Dafür nutzen wir Container-basierte Ansätze. Egal welche Plattform ich wähle, sie muss nur den Container-Betrieb ermöglichen. Und auch bestimmte Prinzipien helfen bei der Cloud-nativen Entwicklung. Zum Beispiel „API Design-first“. Das Prinzip könnte man mit einem Wireframe bei der UI-Entwicklung vergleichen: Bevor ich eine Zeile Code schreibe, mache ich mir unter UXGesichtspunkten Gedanken über den Aufbau und die Struktur der Oberfläche. Mit „API Design-first“ kann ich bei der Entwicklung von APIs ähnlich vorgehen: Wir beschreiben die Ressourcen, erstellen ein Mock-up und können so sehr schnell potenzielle APIKonsumenten für ein Feedback mit einbeziehen. So können wir Business APIs effizient und zielgerichtet entwickeln. Kurze Feedbackzyklen und kollaboratives Vorgehen helfen uns dabei.

Abb. 3: Teams bündeln verschiedene Schwerpunkte und Fähigkeiten

Richard Attermeyer: Was wir uns auch bewusst machen sollten, ist: Es gibt keine Fullstack-Entwickler. Professional Software zu bauen, ist eine Teamaufgabe. Der Zoo der Tools ist gigantisch und wächst fast täglich. Alle zu beherrschen, ist unmöglich. Deshalb brauchen wir Entwickler, die sich schnell in neue Tools reindenken können. Basis dafür sind ein Verständnis der Konzepte und Muster hinter den Werkzeugen und ein tiefes Wissen in einem bestimmten Bereich, sei es eine Programmiersprache oder eine Datenbanktechnologie oder ...

Sven Bernhardt: Experte zu werden, bedarf Zeit, viel Zeit. Und ein Team sollte die notwendigen Experten integrieren. Wir wissen, das ist nicht immer möglich. Dennoch plädieren wir dafür, dass das Team zumindest Zugriff auf das Expertenwissen hat, um bestimmte Fehler zu vermeiden und die Einarbeitungszeit zu verkürzen.

Fazit

OBJEKTspektrum: Wie würden Sie das Gesagte zusammenfassen? Was sollten wir aus diesem Artikel mitnehmen? Bitte geben Sie zum Abschluss ein kurzes Statement ab.

Richard Attermeyer: Die Softwareentwicklung hat sich schon immer verändert. Doch diesmal geht es nicht um eine neue Programmiersprache oder ein neues Framework. Jetzt geht es um ein neues Selbstverständnis, um Teambuilding und am Ende auch um Automatisierung und Qualität. Wir sind keine Codeschubser, sondern Teil eines großen Ganzen, das wir mitgestalten und für das wir auch mitverantwortlich sind.

Sven Bernhardt: Ja, richtig. Und „lebenslanges Lernen“ wird hier anfassbar. Wir brauchen echte Teamarbeit, um die komplexen Anforderungen zu erfüllen, und die Bereitschaft, miteinander zu teilen und uns weiterzuentwickeln. Mit Prinzipien wie „Shift Left“ oder „API Design-first“ können wir unsere Arbeitsweise an die neue Situation anpassen und Professional Software entwickeln. Mit Cloud-native-Anwendungen und containerbasierten Architekturen schaffen wir veränderungsfähige Systeme für unsere Kunden und rüsten sie für die Digitalisierung.

Richard Attermeyer

ist bei OPITZ CONSULTING als Senior Manager Business & IT Innovation unter anderem für das Architecture Board zuständig. Er beschäftigt sich seit Jahren mit flexiblen Systemarchitekturen und deren Wechselwirkungen mit Entwicklungsprozessen und Unternehmens- und Managementkulturen. Hot Topics: CI/CD, Technologiemanagement, flexible Systemarchitekturen und DevOps.

E-Mail: richard.attermeyer(at)opitz-consulting.com

Sven Bernhardt

fungiert bei OPITZ CONSULTING als Chief Architect und Integration Evangelist. Er konzipiert und implementiert zukunftsorientierte, robuste Anwendungen. Sven Bernhardt arbeitet in verschiedenen Projekten, die in den Bereichen Cloud, Microservices und API Management angesiedelt sind. Dabei ist er an der Entwicklung und Konzeption von Best Practices in Bezug auf moderne Lösungsarchitekturen beteiligt. (http://omesa.io)

E-Mail: sven.bernhardt(at)opitz-consulting.com