Interview mit Dirk Lehmann

DevOps @ SAP – Trampelpfade zum Feature

IT Spektrum sprach mit Dirk Lehmann, DevOps-Evangelist und Experte für Continuous Delivery (CD) bei SAP.

Johannes Mainusch: Dirk, schön, dich nach deinem Vortrag auf der „Continuous Lifecycle"-Konferenz in Mannheim wiederzusehen. Du bist der DevOps-Guy bei SAP in Walldorf. Wie lange schon?

Dirk Lehmann: Ich bin seit knapp 23 Jahren bei SAP. Vorher war ich bei IBM, wo ich mein duales Studium absolvierte. Schon während meines Studiums hatte ich von SAP gehört, wusste jedoch nur wenig über deren Angebot an Unternehmenssoftware. Eine Berufsmesse dort im Jahr 2000/2001 änderte das. Zu dieser Zeit begann SAP, sich intensiver mit Non-ABAP-Programmiersprachen wie Java und JavaScript zu beschäftigen. Ich stieß auf einen Messestand, der sich mit Knowledge-Management befasste und nach Java-Experten suchte – genau mein Interessengebiet zu der Zeit.

Die Knowledge-Manager waren die Einzigen, die im Jahr 2001 Java auf dem Zettel hatten?

Zumindest auf der Berufsmesse, fast die gesamte restliche SAP programmierte damals ABAP, das steht für Advanced Business Application Programming und ist die traditionelle hauseigene Sprache in Walldorf.

2000/2001, also vor der großen Dotcom-Blase, kamst du zu einer großen Firma in Waldorf, mit Tennisplätzen vor den Büros.

Ich habe mich relativ gut mit den Kollegen an dem erwähnten Stand unterhalten und mir auch alles vor Ort angeschaut. Richtig, vor dem Schulungsgebäude gibt es Tennisplätze. Das alles hat mir zugesagt, und so habe ich vor meinem Studienende meinen Arbeitsvertrag unterschrieben. Ich glaube, damals arbeiteten bei SAP etwa 20 bis 25 Tausend Menschen, heute über 100 Tausend.

Damals waren bei SAP die Begriffe R3 und ABAP zentral. Eine ganz eigene Informatik-Welt …

R3 ist eine Art Laufzeit- und Entwicklungsumgebung, auf der Programme in ABAP laufen. Wer aus der Java-Welt kommt, kann sich schwer vorstellen, dass man in der Laufzeitumgebung auch coden und somit das System verändern kann, also in das SAP-Business-Coding eingreifen und es mit eigenen Funktionen erweitern kann.

Du kamst also mit einer modernen Programmiersprache in eine Welt, die bereits erfolgreich ist, aber technologisch isoliert erscheint, eine eigene Programmiersprache, Plattform und Beraterkaste hat …

Eine meiner ersten Aufgaben war das Management des Build-Prozesses. Dies war in einer Ausgründung, die sich um Portale kümmerte und primär in Java entwickelte. Wir nutzten als Build-Tool Apache-Ant, das XML zur Beschreibung des Build-Prozesses nutzt. Diese XML-Datei, etwa 2 bis 3 Tausend Zeilen lang, beschrieb den Prozess, wie Übersetzungen ins SAP-System hochgeladen und wieder runtergeladen wurden, sowie den Bau und die Tests des Portals.
Eines meiner ersten Probleme war, dass Code-Änderungen manchmal nur auf den Maschinen einzelner Entwickler kompiliert wurden. Deshalb habe ich früh einen CI/CD-Ansatz implementiert, den ich jedoch nicht bewusst als solchen benannte. Ich verwendete Perl- und Bash-Skripte, um einen CI/CD-Server namens „Bob-the-Builder" zu entwickeln.

Wie hast du die einzelnen Entwickler davon überzeugt, dass ein zentraler Build-Server besser ist, als alles lokal zu erstellen?

Das war gar nicht so einfach. Zunächst war Bob-the-Builder auch nur auf einem Desktop-Rechner implementiert. Immer, wenn es zu Fehlern beim Build kam, verschickte Bob-the-Builder E-Mails an die Entwickler, deren Sourcecode fehlerhaft war. Es gab dann häufig Entwickler, die bei mir in der Tür standen und sagten: Auf meinem lokalen Rechner läuft das aber. Es dauerte, bis wir das System stabil hatten, und es dauerte auch, bis wir uns darauf geeinigt hatten, dass das, was Bob-the-Builder sagt, zählt! Alles andere zählt nicht. Irgendwann wurde dieser Desktop-Rechner dann sehr warm und hatte immer mehr zu tun. Daher verlagerten wir das Ganze in einen Serverraum auf leistungsfähigere Server. So wurden wir eine der Komponenten des damaligen Produkts SAP NetWeaver.

Ich habe im Herbst in Mannheim auf der Continuous Lifecycle deinen Vortrag gehört, in dem du deine Rolle als eine Art technischer Product Owner eines DevOps-Teams beschreibst. Wenn ich DevOps höre, denke ich an Teams mit etwa sieben Personen. Wie groß ist euer DevOps-Team?

330 Personen (da wir dabei die Externen mitzählen, fluktuiert das ein wenig). Wir sind nicht ein Team, wir sind eine Organisation. Es ist etwas schwer zu

beschreiben, was mein aktueller Job ist. Laut Personalabteilung ist der Jobtitel Chief-Development-Experte, aber meine eigentliche Jobrolle ergibt sich aus dem SAFe-Framework, dort gibt es eine Rolle, die nennt sich Solution-Manager. Das ist meine Rolle, und der Solution-Manager soll den Ende-zu-Ende-Überblick haben und versuchen, die Entwicklungsumgebung für SAP-Entwickler zu verbessern.

Für wie viele Menschen macht ihr das?

Wir machen die Entwicklungsumgebung für alle Entwicklerinnen und Entwickler bei der SAP, das sind ungefähr 36.000 Menschen, die in Forschung und Entwicklung tätig sind. Und zwar hauptsächlich für die Non-ABAP-Entwickler, das sind ungefähr 18.000. Wie vorhin schon angesprochen, haben die ABAP-Entwickler ihre Entwicklungsumgebung im SAP-System und werden zum Großteil mit der Software-Logistik aus dem SAP-System versorgt. Aber einige Teile nutzen auch unsere Umgebung, etwa, wenn es um Release-Dokumentationen geht und andere Sachen eher administrativer Art.

Wie kann ich mir so eine Entwicklungsumgebung vorstellen? Kann ich da als Entwickler in meinem Lieblingseditor VS-Code einfach darauf los programmieren?

Bei uns im Non-ABAP-Bereich ist es freigestellt, welche Editoren verwendet werden. Es gibt einige Plug-ins für Qualitätsanalyse, Security-Scans usw. Für Code-Versionierung haben wir hauptsächlich GitHub im Einsatz und dann eine Phalanx von ungefähr 70 Tools, zum Beispiel für Security-Scanning, Test-Validierungen, Dokumentation der Export-Compliance, Dokumentation von Release-Entscheidungen und vieles mehr, was zur Softwareentwicklung und Auslieferung dazu gehört.

Einen KI-Assistent, der mir beim Coden hilft oder mir Fragen beantworten kann, sicher auch, oder?

Ja, KI kommt natürlich aktuell auch vor. Die Plattform, die wir entwickeln, umfasst den CI/CD-Teil. Ich plane meine Software, halte mein Backlog in Tools wie Jira, fange an zu coden, checke in eine Code-Versionierung ein, und dann laufen CI/CD-Orchestrierungen ab. Diese überprüfen Änderungen, führen Code-Scans, Tests und Security-Scans durch und speichern Artefakte. Am Ende treffen wir eine Release-Entscheidung und die Verantwortung geht an die Runtime-Plattform über. Dort sind Funktionen wie Observability und Alerting. Es gibt Übergabepunkte, um sicherzustellen, dass die Artefakte von uns stammen und offiziell gebaut wurden, bevor sie an die SAP-Business-Technology-Plattform übergeben werden.

Ist das nur für interne SAP-Entwickler oder kann das auch von Beratern und Partnern genutzt werden?

In der Vergangenheit haben wir einen Plattformansatz für interne und externe Entwickler probiert, was zu Skalierungsproblemen führte. Unterschiedliche Anforderungen und Skalierungsfaktoren kollidierten. Heute haben wir eine interne Plattform namens Hyperspace, während externe Kunden einen eigenen CI/CD-Service auf der SAP-Cloud nutzen. Wir managen die Überschneidungen teilweise über Inner Source. Ein wichtiger Teil ist die Pipeline-Templates-Bibliothek „Projekt Piper“, die auch Open Source ist. Diese Library wird von internen und externen Entwicklern genutzt, um Systeme anzusteuern und Tests durchzuführen. Grundsätzlich sind die Produktbereiche getrennt, um die verschiedenen Anforderungen zu erfüllen. Zum Beispiel werden unsere Softwareprodukte in 180 Ländern vertrieben und in 33 Sprachen übersetzt, was einzigartige Anforderungen an die interne Entwicklung stellt, welche die externe Entwicklung so nicht hat.

Das heißt, intern müsst ihr immer alles weltweit compliant machen, also beispielsweise Templates in 33 Sprachen hinterlegen, während solche Anforderungen in einem speziellen, etwa nur deutschen Kundenprojekt gar nicht erfüllt werden müssen.

Genau. Es gibt fast 300 Compliance-Anforderungen, die die SAP-Produkte vor ihrem Release nachweisen und dokumentieren müssen. Das sind Dinge wie Security, Performance, User Experience, Functional Correctness. In Summe, glaube ich, 13 Kategorien. Und daraus entstehen dann rund 300 Anforderungen, wo Teams nachweisen müssen, dass sie funktionale Tests durchgeführt und Security-Maßnahmen ergriffen haben, Design-Richtlinien gelesen haben usw. Oder dass Exportrichtlinien eingehalten werden. Durch all diese Dinge sind die Anforderungen an unser internes System enorm und viel umfangreicher und komplizierter als in normalen SAP-Kundenprojekten.

Dirk Lehmann

kam als Java-Experte zu SAP.
Derzeit als Chief-Development-Experte Teil des internen CI/CD-Produktmanagement-Teams, das eine interne Entwicklerplattform für über 30.000 Ingenieure entwickelt
www.linkedin.com/in/dirk-lehmann-7712bb125/

Ich stelle mir das ziemlich schwierig vor, bei euch zu entwickeln. Als mittelmäßig begabter Informatiker müsste ich erst viel kennenlernen. Es klingt, als ob man durch zahlreiche Hindernisse springen muss, bevor ein Feature das Licht der Welt erblickt. Wie geht ihr damit um?

Das ist die große Masterfrage: „Wie macht man das einfacher?“ Wie viele größere Unternehmungen suchen wir unser Heil im Platform-Engineering. Wir bieten eine Vielzahl von Tools an, von denen nur eine Handvoll obligatorisch ist. Teams haben jedoch auch die Freiheit, zusätzliche Open-Source-Tools zu nutzen. Wenn zu viele Compliance-Anforderungen gestellt werden, leidet die Entwicklung darunter, da sie mehr Zeit mit internen Angelegenheiten verbringen als mit der Wertschöpfung für Kunden. Unsere Hyperspace-Plattform bietet daher Tooling mit garantierter Qualität und vereinfacht die Compliance. Und die allermeisten davon sind zur freiwilligen Nutzung. Ein Ansatz, den wir verwenden, ist der „Golden Path“, oder auch „Paved Road“, der sicherstellt, dass Teams effizient arbeiten können, ohne von zu vielen Anforderungen überwältigt zu werden.

Wie ein Märchen – der goldene Pfad …

Spotify hat das Konzept des „Golden Path“ veröffentlicht, das nun in verschiedenen Varianten bekannt ist. Bei uns

nennen wir das auch die „Paved Road“. Das Prinzip bleibt jedoch dasselbe: den Entwicklerteams klare Richtlinien geben, wie bestimmte Tools und Prozesse zusammenarbeiten sollen. Dabei sind unsere Werkzeuge freiwillig von den Teams zu nutzen, denn sie sind nicht für alle Situationen geeignet. Dennoch können etwa 80 Prozent der Teams davon profitieren, indem sie gemeinsame Vorgehensweisen für Sicherheitsscans, Backlog-Management, Repository-Strategie und Build-Prozesse erhalten. Diese Guidelines erleichtern die Einhaltung von Compliance-Anforderungen und den Entwicklungsprozess, indem sie klare Wege aufzeigen, wie Software gebaut und veröffentlicht werden kann.

Zur freiwilligen Nutzung?

Es ist wichtig, Plattformen nicht verpflichtend zu machen, da dies die Innovationsfreudigkeit beeinträchtigen kann. Wenn eine Plattform nur auf eine Technologie beschränkt ist und neue Trends auftauchen, können Innovationen unterdrückt werden. Es ist eine Herausforderung, eine Kultur zu schaffen, die fundierte Entscheidungen darüber trifft, wann es angebracht ist, von den etablierten Pfaden abzuweichen und wann nicht. Einseitige Innovation aus Eitelkeit kann zu Misserfolgen führen, wie wir in der SAP erlebt haben.

Gerade wenn du Innovation machst, bist du nicht immer auf dem goldenen Pfad unterwegs.

Genau, du machst was, was noch keiner gemacht hat, und eine „Paved Road“ oder ein „Golden Path“ sagt halt im Namen, das sind Trampelpfade, die sind schon andere gegangen, für die ist das genau das richtige Rezept. Aber im innovativen Bereich machst du ja Dinge zum ersten Mal, da mag es dann nicht immer schon einen Trampelpfad geben …

Wie ist bei euch das Verhältnis Kommunikation zu Coding? Also, wie viel musst du reden, bevor du etwas Neues erstellen oder eine Verbesserung machen kannst?

Kommunikation ist entscheidend, besonders aus Sicht der Solution- und Produktmanager in unserem Team. Anfangs haben wir die interne Kommunikation vernachlässigt und uns hauptsächlich auf die Kommunikation mit unseren internen Entwicklungsteams konzentriert. Wir haben jedoch festgestellt, dass viele Teams nicht ausreichend informiert waren über Änderungen und neue Entwicklungen. Deshalb setzen wir jetzt auf neue Kommunikationswege, wie die Entwicklung von Personas und Zukunftsvisionen durch Comicstrips. Diese Ansätze helfen dabei, eine klare Richtung vorzugeben und sicherzustellen, dass alle Teams über unsere langfristigen Ziele informiert sind, auch wenn es ständig Änderungen gibt.

Wie lange macht ihr das jetzt so?

Das Produktmanagement-Team, in dem ich jetzt dabei bin, gibt es seit gut zweieinhalb Jahren, die Teams, die diese ganzen Tools betrieben haben, gab es vorher schon, die waren vorher in der SAP verteilt.

Okay, 70 Tools in zweieinhalb Jahren wäre ja auch ein erstaunlicher Speed …

Das ist wie so oft historisch gewachsen. Vor knapp drei Jahren haben wir angefangen, das neu zu strukturieren und zu harmonisieren. Allein den Support und den Betrieb in einer Einheit zu übernehmen, ist schon eine Herkulesaufgabe. Das ist die erste Hauptaufgabe, mit der man anfängt, wenn man so einen heterogenen Zoo an Tools zusammenführt.

Da habt ihr doch bestimmt auch eine Menge Legacy im Einsatz? Was ist denn beispielsweise eure älteste Java-Version, die ihr in einem Tool im Einsatz habt? Und was ist die modernste?

Das weiß ich aus dem Stegreif gar nicht. Das modernste ist sicherlich immer latest greatest, das sogenannte bleeding edge, da bin ich mir ziemlich sicher. Aber wir haben auch Tools, die noch aus der Zeit kommen, als ich hier angefangen habe. Deshalb wird auch dieser Zoo immer größer, denn hinter diesen Tools stehen ja Produkte der SAP, und die haben per se schon einen langen Lebenszyklus. Mit den Produkten wird dann beim Endkunden Software erstellt, die oft für Dekaden im Einsatz ist. Und die Kunden erwarten auch einen jahrzehntelangen Support.

Als Physiker stelle ich mir Softwareentwicklung immer wie den Urknall vor. Wenn die Explosion noch ganz jung ist, passiert im zeitlichen Verlauf unfassbar viel. Und dann dehnt sich das aus und alles wird kälter. Und je mehr sich Software ausdehnt und je größer der Phasenraum ist, desto schwieriger wird es, alle Teile zu bearbeiten, weil einfach die Energie fehlt.

Dazu gibt es von meinem Namensvetter Clemens Lehman – nicht verwandt – the law of software evolution. Der hat mehrere Softwaregesetze aufgestellt. Ich glaube, das erste ist in etwa: „Software, die sich nicht ändert, wird irgendwann irrelevant“, und das zweite ist: „Jede Software, die sich ändert, wird komplexer“. Daher sage ich Entwicklerteams, Software ändern ist eigentlich keine gute Idee. Macht das nur, wenn es wirklich sein muss. Und wenn das einen Mehrwert für den Kunden bringt. Alles andere, nur auf gut Glück Software verändern, macht die Software komplexer. Komplexer in der Wartung, komplexer im Betrieb. Und in allen anderen Dimensionen nimmt diese Komplexität zu, wie du so schön gesagt hast, mit diesem Urknall-Bild.

Jetzt passiert etwas Wunderbares am Ende des Interviews, die gute Fee kommt vorbei und gewährt dir einen Wunsch, der geht in genau einem Jahr in Erfüllung, was wünscht du dir?

Glück und Gesundheit und Weltfrieden haben sich bestimmt schon andere Interviewpartner von der Fee gewünscht, daher wähle ich etwas, um unsere Plattform voranzubringen. Ich glaube, dass wir noch besser den Nutzen dieses Plattformansatzes, sowohl in der Organisation, also in dieser Hyperspace-Organisation, diesen ca. 300 Leuten, als auch innerhalb der SAP bewerben können. Wie du vorhin schon gesagt hast, es ist eine große Organisation. Daher wünsche ich mir Einfachheit. So wie in einem Start-up mit 20 Personen, da schreist du einmal über den Tisch und hast die richtige Person erwischt. Diese Einfachheit wünsche ich mir …

Viel Erfolg beim einfach machen, und vielen Dank für das Interview, Dirk!

Das Interview führte …

Dr. Johannes Mainusch
(johannes.mainusch@kommitment.works)
Berater für Unternehmen, die Bedarf im Bereich IT, Architektur und agiles Management haben. Dr. Mainusch ist seit 2012 Mitglied der IT Spektrum-Redaktion.