„DIE CODE-STADT IST IMMER IM UMBAU.“

Gespräch mit
Dr. Johannes Bohnet,
CEO des Software-
Process-Mining-
Pioniers Seerene.

DIALOG: Herr Dr. Bohnet, Sie sprechen davon, dass Softwareorganisationen zu Softwarefabriken werden müssen. Was steckt hinter dieser Analogie und wo sind ihre Grenzen?

JB: Softwaresysteme sind heute das Fundament von zentralen Geschäfts- und Produktionsprozessen in jeder Industrie. Wenn man eine städtebauliche Analogie nutzt, dann sind Softwaresysteme große gewachsene Städte, die den Ablauf des Business ermöglichen. Diese Software-Städte aufzubauen, sie immer wieder um zusätzliche Funktionalitäten zu erweitern und die Code-Gebäude vor dem Verfall zu schützen, ist allerdings keine triviale Aufgabe. Das liegt zum einen an der Größenordnung, zum anderen an der Komplexität und Abstraktheit. Wir können Softwaresysteme nicht anschauen, wir können sie mit unseren Sinnen nicht erfassen. Aber würden wir den Source Code ausdrucken, würde sich das Papier so hoch stapeln, wie alle Wolkenkratzer Manhattans. Diese enormen Dimensionen sind selbst für Menschen, die unmittelbar mit Software zu tun haben, kaum greifbar. Solche Softwarestrukturen – große Kernbankensysteme oder eine Plattform wie Netflix – lassen sich nur erschaffen, wenn man Softwareentwicklung als Ingenieurwissenschaft betreibt. Die Softwareentwicklung hat allerdings noch nicht den Reifegrad einer klassischen Ingenieurwissenschaft, denn die gesamte Disziplin ist noch ziemlich jung.

Viel wichtiger ist aber die dynamische Natur der Software, die nie fertig ist – genauso wie eine Stadt. Die Code-Stadt ist immer im Umbau. Die Methodologien und Ansätze verändern sich, die Systematiken entwickeln sich weiter, es wird permanent gearbeitet. Was dabei bislang immer fehlt, ist die Transparenz.

Man kann sich Software-Engineering ein Stückweit wie den Turmbau zu Babel vorstellen. So treffen etwa Management-Logiken, deren Sprache Deadlines, Geschwindigkeit und Budgets sind, auf eine technologische Perspektive, die sich an Frameworks oder Modulen orientiert. Softwareentwicklung ist eine Herausforderung geradezu biblischen Ausmaßes und das ist aus meiner Sicht ein wesentlicher Grund, warum wir als Software-Ingenieurwissenschaft noch nicht da sind, wo wir sein könnten. Hinzu kommt, dass der Turm auch noch unsichtbar ist – und die Baustelle nicht darauf angelegt ist, jemals fertigzuwerden.

DIALOG: Wie kann man sich eine Fabrik vorstellen, deren Produkt nicht fertig werden kann, und was bedeutet das für die Steuerung und Effizienz von Softwarefabriken?

JB: Beim Software-Engineering gibt es eigentlich kein Greenfield. Das Produkt, an dem gearbeitet wird, verbleibt immer in der Fabrik. Es wird kontinuierlich weiterentwickelt, angepasst, erweitert. Das ist ein signifikanter Unterschied zur Hardwareproduktion – und ein Treiber für Komplexität und Intransparenz. Es gibt keine Dokumentation über diesen evolutionären Prozess, außer das Produkt selbst. Der Code ist die Dokumentation. Genau daraus erwächst auch die häufige Notwendigkeit für sehr aufwendiges Reverse Engineering, wobei Softwareentwickler die in Software eingebaute Geschäftslogik über das Lesen von Code nachzuvollziehen versuchen.

DIALOG: Lässt sich überhaupt analytisch über einen Prozess sprechen, der eine solche Komplexität und Abstraktheit aufweist?

JB: Global verteilte, dynamisch wachsende Software – das ist ja nicht nur die Software selbst. Es sind Tausende, vielleicht

Zehntausende von Menschen, die kontinuierlich daran arbeiten. Ich glaube, man hat nur eine Chance, das mit dem menschlichen Verstand zu erfassen, wenn man Metaphern verwendet. Deswegen verwenden wir die Metapher der Stadt, um greifbar zu machen, was in Jahren oder auch Jahrzehnten an Source Code entstanden ist, mit Code Units als Gebäuden. Wir zeigen eine hierarchische Modulstruktur, die sich durch Stadtbezirke, Häusergruppen und einzelne Gebäude darstellen lässt. Diese Symbolik erlaubt es, eine gemeinsame Sprache zu etablieren, in der Management und Technik sich austauschen können. Wieso ist mein Gebäude so hoch und rot? Warum brauche ich mehr Zeit oder Budget, um mein Gebäude zu modernisieren? Dadurch lassen sich abstrakte Themen wieder kommunizieren, objektivieren und in notwendige Entscheidungsprozesse einbringen.

DIALOG: Wie kann man vor diesem Hintergrund die Entwicklung hin zu einer echten Ingenieurwissenschaft unterstützen?

JB: Die Softwarefabriken sind heute sehr gut ausgerüstet. Die Entwickler haben Compiler, es gibt Ticketmanagementsysteme, Continuous Integration und vieles mehr. Es fehlt nicht an exzellenten Werkzeugen für die Produktion und da ist nicht mehr viel Raum für Optimierung. Was allerdings fehlt, ist die Anwendung von Methodiken, die in der Hardwareproduktion eingesetzt werden. Dort ist es Standard, dass der Durchfluss gemessen wird, dass jeder einzelne Teilproduktionsschritt erfasst wird, dass sich Aussagen über Qualität und Effizienz treffen lassen und vieles mehr.

Einen solchen Leitstand oder auch einen „Digital Boardroom“, wie wir es nennen, brauchen wir auch für die Softwareproduktion. Heute ist es für das Management kaum möglich, einen wirklichen Einblick in die Softwarefabrik zu gewinnen – ganz unabhängig davon, welche Methodologien und Programmiersprachen verwendet werden oder ob es um die Produktion eines Kernbankensystems geht, einer digitalen Plattform oder einer Software für smarte Wasserhähne. Wir brauchen die durchgängige Transparenz von der operativen bis hin zur strategischen Ebene, um Effizienzverluste zu erkennen und einen zahlengestützten, kontinuierlichen Verbesserungsprozess initiieren zu können. Dieses Vorgehen, bei dem ein Digital Boardroom aufgebaut wird und faktenbasierte Steuerungsprozesse initiiert werden, lässt sich auch als „Software Process Mining“ bezeichnen.

DIALOG: Worin unterscheidet sich Software

Process Mining vom herkömmlichen Process Mining?

JB: Process Mining adressiert Prozesse, die per se schon da sind, die greifbar sind, hinter denen konkrete Artefakte oder physische Elemente stehen. Es funktioniert beeindruckend gut auf der Business-Ebene. Aber im Unterbau, auf der Ebene der Softwarefabriken, da greifen klassische Process-Mining-Lösungen nicht. Da ist heute eine Leerstelle, in die das Software Process Mining hineinstößt.

Dabei geht es zunächst darum, den Gegenstandsbereich überhaupt greifbar, darstellbar, kategorisierbar zu machen, Zielkonflikte und Verkantungen zu visualisieren. Die Basis zu schaffen, auf der die Prozesse nachvollzogen werden und ein Optimierungsansatz ablaufen kann. Dafür ist tiefes Domänenwissen erforderlich, das über die herkömmlichen Process-Mining-Systeme nicht zugänglich ist.

DIALOG: Wie ausgeprägt ist das Verständnis für das Thema auf der Business-Seite?

JB: Entscheider verstehen sehr genau, dass Softwarekompetenz existenziell ist. Und sie wissen auch, welche Konsequenzen es hat, wenn ihre Softwarefabriken nicht als Enabler der Transformation fungieren und die Softwareentwicklung nicht mit den Business-Anforderungen mithalten kann. Software wird künftig allgegenwärtig sein, sie durchdringt schon heute einen Großteil der produzierten Hardware und wird selbst in so komplexen und mit enorm hohen Eintrittshürden versehenen Industrien wie Automobilbau zum Unterscheidungsmerkmal. Deshalb wird in den Unternehmen intensiv an agilen Methoden gearbeitet, an der Frage wie man eine Softwareorganisation optimal aufstellt. Dafür wird sehr viel interne Expertise und externe Beratungskompetenz mobilisiert.

Das Problem ist nur, sobald eine Softwareorganisation aufgestellt ist, wird sie typischerweise sich selbst überlassen. Sie wird zur Blackbox. Man kann nicht nachvollziehen, warum Deadlines gerissen werden, Fehler passieren oder an den Bedürfnissen vorbei entwickelt wird. Man hat keinen Einblick, man kann die Prozesse nicht präzise beschreiben und messen. Man weiß nicht, ob man wirklich besser wird. Vielleicht wird man schneller, man hat vielleicht das Gefühl, innovativer zu sein. Aber wird man besser, effizienter? Das ist exakt die Leerstelle, von der ich gesprochen hatte. Und es gibt auf der Business-Seite enormes Interesse, das zu ändern.

DIALOG: Lässt sich denn auf der Grundlage von Software Process Mining Analytics auch die Effizienz der Prozesse in Softwarefabriken berechnen?

JB: Die Messung der Produktivität ist schwierig. Wir können heute kein sinnvolles Verhältnis zwischen Input, der Zeit,

die Entwickler investieren, und der Menge des erzeugten Business Value berechnen. Noch nicht. Das Problem ist, wie man berechnen kann, wie viele Business-Euro durch 100 neue Codezeilen entstanden sind. Wir beschäftigen uns natürlich intensiv mit dem Thema, ebenso das Hasso-Plattner-Institut, mit dem wir eng verbunden sind, aber auch einige andere Organisationen weltweit. Eine gute Nachricht ist aber, dass wir trotzdem Kenngrößen wie die Effizienz berechnen können. Ich kann nachvollziehen, wohin dieses kostbare Gut, die Zeit der Softwareentwickler, hinfließt, und in welche Themen. Ich kann nachvollziehen, wie viel dieser Zeit verloren geht und wie viel für die Schaffung von Business Value, von neuen Funktionalitäten genutzt werden kann.

DIALOG: Welche Erkenntnisse lassen sich dabei typischerweise gewinnen?

JB: Etwa die Erkenntnis, dass rund 80% der Ressourcen in der Softwarefabrik verpuffen. Das bedeutet, dass nur zwei von zehn investierten Euro wirklich Business-Wert generieren. Solche Relationen wären in der Hardwareproduktion unvorstellbar. Es ist extrem viel Sand im Getriebe, es knirscht an allen Ecken und Enden. Jeder, der Softwareentwicklung verantwortet, weiß das, fühlt das, hat aber kaum eine Möglichkeit zu sehen, wo genau in dieser unsichtbaren, dynamischen und verteilten Struktur der Sand ist, was zu tun ist, um diese Situation zu ändern.

Wir greifen da erneut auf die Metapher der Stadt und die entsprechende Visualisierung zurück. Jeder Entwickler kennt Code-Bausteine, die man lieber nicht anfassen will: marode Gebäude, bei denen die Fassade bröckelt, wenn man etwas anbauen will. Das sind Folgen der Altlasten, der „technischen Schuld“. Aber man muss natürlich da ran und muss dabei sehr vorsichtig und langsam vorgehen, was sehr viel Zeit erfordert. Solche Strukturen, die natürlich Effizienzkiller sind, können wir zeigen und dafür sorgen, dass darüber disziplinübergreifend gesprochen werden kann.

DIALOG: Allerdings kann man – um in der Metapher zu bleiben – ja nichts daran ändern, dass diese Gebäude nun mal in der Stadt stehen. Sie sind in die Prozesse und Strukturen tief integriert und müssen gepflegt werden – ob effizient oder nicht. Was nutzt mir dann die Transparenz?

JB: Ja, die Gebäude machen schon Sinn. Es ist allerdings so, dass die Code-Stadt ohne Analytics nicht überschaubar ist. Und deshalb herrscht oftmals ein extremer Projektdruck auf die IT von der Business-Seite. Es sollen laufend neue Features und Funktionalitäten erzeugt werden. Um diesen Anforderungen gerecht zu werden, müssen die Entwickler häufig an die alten, maroden Gebäude der Code-Stadt ran.

Und es ist essenziell, dass sie dabei die Komplexitäten, die Wechselwirkungen, insbesondere auch die technischen Schulden, die dabei entstehen, der Business-Seite erklären können.

Denn diese Schulden führen einerseits zu teils gravierenden Risiken, die man in das System einbaut, und andererseits zu Zinszahlungen. Diese beiden Aspekte muss man transparent machen. Und das ist etwas, wofür die reine Code-Analyse nicht ausreicht. Denn in einer Fabrik kann man nicht nur die Vorprodukte und Produktionsverfahren betrachten. Es geht auch um die Menschen, die in dieser Fabrik arbeiten und ihre Zeit für bestimmte Themen investieren. Man braucht die Möglichkeit zu sehen, wo die technische Schuld dringend abgebaut werden muss, weil die Zinszahlungen gigantisch hochgeschwungen sind.

Wenn auf der Business-Seite erkannt wird, dass die Entwicklerteams vielleicht 5% ihrer Zeit dafür verwenden, den bestellten Balkon anzubauen, und 95% darauf, die Fassade zu stabilisieren, die auf der anderen Gebäudeseite herunterbröckelt, dann hat man sehr viel erreicht – man hat die sehr spezielle fachliche Perspektive der Entwickler in eine Sprache übersetzt, die an betriebswirtschaftliche Prozesse anschlussfähig ist und vice versa. Und damit lässt sich eine in der Softwareentwicklung absolut kritische Bruchstelle zwischen der Business-Seite und den Technik-Experten überbrücken.

DIALOG: Existiert diese Bruchstelle auch in agilen Setups, wo die Kommunikation enger und kurzzyklischer ist?

JB: Ja. Und ich würde sogar sagen, dass gerade dort der Einsatz von Analytics dringend notwendig ist. Denn in agilen Strukturen, erst recht bei großen Scaled-Agile-Organisationen, lässt sich kaum nachvollziehen, wie viel Entwicklerzeit für welche Themen investiert wird. Agile Teams sind wie kleine Schnellboote. Man kann jedes einzelne Boot betrachten, aber man sieht die Flotte nicht mehr. Es fehlt die übergreifende Perspektive, die sicherstellt, dass nicht nur jedes Boot für sich, sondern die ganze Flotte immer besser wird. Das fehlt in der agilen Welt. Das kann man mit Analytics wieder erreichen.

Beim Einsatz von Analytics sind allerdings zwei Aspekte sehr wichtig. Erstens darf die Einführung der Analytics-Apparatur nicht dazu führen, dass operative Prozesse verzögert werden. Denn die Softwareorganisationen stehen bereits unter hoher Belastung. Die Idee von Software Process Mining darf nicht dadurch diskreditiert werden, dass es diesen Druck erhöht. Zweitens ist eine sehr klare und transparente Kommunikation absolut kritisch. Software Process Mining darf nicht als Überwachung in „Big-Brother-Mannier“ wahrgenommen werden, sondern als ein Katalysator, um die Sprachverwirrung aufzulösen, als eine Chance, endlich zu zeigen, welche Komplexitäten, Risiken, Aufwände und Zeitverluste bei bestimmten Anforderungen entstehen und welche Faktoren in Entscheidungsprozesse auf der Business-Seite einfließen müssen. Es geht darum, endlich Licht in die Code-Stadt zu bringen.