Kap des guten Codes

Interview mit den Gründern von Cape Of Good Code

Die deutsche Firma Cape Of Good Code wurde vor nicht mehr als einem Jahr gegründet. Sie beschäftigt sich mit der Analyse von Softwarearchitekturen und beweist, dass es auch hierzulande möglich ist, innovative Ideen kommerziell umzusetzen. JavaSPEKTRUM hat sich mit den beiden Gründern Egon Wuchner und Konstantin Sokolov unterhalten.

? Als Erstes würde ich gerne den Kontext herstellen. Warum und wann habt ihr euer Unternehmen gegründet?

Wir haben im August 2018 gegründet. Davor haben wir uns bei Siemens Corporate Technology mehrere Jahre mit Software-Qualitätsanalysen beschäftigt. Wir waren unzufrieden mit dem Status quo der Analysen, vor allem mit der Aussagekraft der Ergebnisse, dem fehlenden Actionable Knowledge. Wir haben also eigene Analysen konzipiert und bei Siemens angewendet. Nach einiger Zeit haben wir das weitere Potenzial dieser Analyseart gesehen, wofür wir aber in dem Unternehmen kein weiteres Budget erwerben konnten. Daher haben wir uns entschieden, alles neu, anders, besser, skalierbarer und auf eigenes Risiko zu machen.

? Wer arbeitet in eurer Firma und was ist eure kurz-, mittel- und langfristige Vision? Wo wollt ihr hin?

Wir sind zu dritt und werden 2020 weitere Mitarbeiter einstellen. Unsere Vision für 2019 war es, mit den ersten Ständen unseres Tools auf eigenen Beinen mittels Consulting zu stehen. Das haben wir geschafft.

»Wir bieten einen Orientierungspunkt, wie ein Leuchtturm weisen wir unseren Kunden die Richtung«

Ein Slogan von uns war: „everything you wanted to know about your code, but were afraid to ask". Wir wussten nicht, wie wahr das sein würde, dass Entwickler und Manager sich nicht trauen zu fragen beziehungsweise in dem bequemen Glauben verweilen, das Wesentliche über ihren Code und dessen Qualität zu wissen. Der Manager glaubt, dass der CTO es weiß, der CTO, dass der Architekt es weiß usw. Daher hatten wir nicht damit gerechnet, wie schwer uns die Akquise der ersten Kunden fallen würde. Denn mit unseren Analysen treten wir allen Projektbeteiligten indirekt auf die Füße. Wir machen die Leute damit nicht glücklich.

Das sieht im Rahmen unserer IT/SW-Due-Diligence-Projekte genau anders aus. Käufer wollen eine Bewertung der Software-Assets und sind darüber sehr froh.

Mittelfristig bis 2021 wollen wir unsere Toolkette schrittweise als SaaS/On-premise anbieten, sodass es für alle Business- und Software-Stakeholder zu einer besseren Entscheidungsfindung bezüglich Nutzung, Aufwand und Qualität der Features im Code kommt. Wir wollen mit unseren Lösungen dazu beitragen, dass alle Stakeholder besser zusammen und weniger gegeneinander arbeiten.

Unser fernes Ziel ist eine Symbiose von KI und menschlicher Intelligenz: Entwickeln bleibt eine kognitive Leistung, aber unsere Analysen und Prognosen sollen entscheidende Hinweise geben, sogar soweit, dass es zu automatisierten Verbesserungsvorschlägen kommt, also zur Synthese von neuem und vorgeschlagenem Code.

? Softwarearchitekturen sind die wichtigste Komponente in Softwareentwicklungsprojekten. Dementsprechend gibt es bereits einige Werkzeuge zur Architekturanalyse. Was macht ihr anders als die anderen?

»Wir wollen mit unseren Lösungen dazu beitragen, dass alle Stakeholder besser zusammen und weniger gegeneinander arbeiten«

Aktuelle Analysetools untersuchen bestimmte Best Practices und Code-Smells auf unterster Ebene, sowie nachweislich wenig aussagekräftige Metriken wie die zyklomatische Komplexität. Der Architekturaspekt beschränkt sich auf Abhängigkeitsanalysen. In beiden Fällen kommen dabei meistens zu viele undifferenzierte und oftmals für die Produktivität irrelevante Findings heraus. Was bringt es, sagen wir, ein Modul auf Utility-Ebene zu verbessern, obwohl es seit zwei Jahren nicht verändert wurde und ansonsten gut funktioniert?

Das mit der Utility-Ebene war ein triviales Beispiel. Es ist aber so, dass das Bauchgefühl kein guter Ratgeber ist, wenn es darum geht zu entscheiden, welche Module wichtig für die Produktivität sind und welche nicht.

Es ist auch schwer, das nur mittels der Auswertung des aktuellen Codes zu bestimmen. Am besten kriegt man so etwas raus, wenn man *beobachtet*, wie sich eine Architektur über die Zeit verhält. Eine Architektur ist ja immer bloß ein Lösungsversuch – wie gut dieser ist, lässt sich schwer a priori bestimmen. Die entscheidende Frage ist doch, wie leicht, ob mit vertretbarem kognitiven Aufwand und ohne neue Fehler, sich eine Software um neue Features erweitern lässt. Genau diesen Aspekt vermissen wir bei den aktuellen Architekturanalysen. Dazu braucht es aber mehr Datenquellen als nur den Code.

Mit unserem Analysetool DETANGLE analysieren wir die Historie des Codes im Code-Repository. Wir stellen die Verbindung zum Issue-Tracker her, wo Funktionalität und Bugs als Tickets erfasst werden. Wir messen dabei nicht Code-, sondern Feature-Modularität, also wie gekoppelt Features im Code zueinander implemen-

Software und Mathematik begeistern uns. Wir wollen zeigen, wie man Software und KI innovativ und mit Nutzen wiederum auf Software anwendet.

Dipl.-Math. Egon Wuchner verfügt über 20 Jahre Erfahrung als Softwareentwickler, -architekt und Projektleiter.

Dipl.-Ing. Konstantin Sokolov verfügt über 10 Jahre Erfahrung als Softwareentwickler und -architekt.

tiert wurden, oder wie die Kohäsion eines Features über den Code hinweg aussieht. Wir sprechen dabei von Feature-Qualitätsschulden.

? Euer Produkt heißt DETANGLE. Kann ich mir darunter vorstellen, dass euer Werkzeug „versteckte" Architekturinformation sichtbar macht?

Ja. „Tangle" heißt Knäuel, den wir zu entwirren helfen, sodass die geschäftsrelevanten Features einem DETANGLE, einer Entkoppelung, unterworfen werden.

Wir machen qualitativ mittels Visualisierungen und quantitativ mittels Metriken die Modularität des Codes in Bezug auf Features sichtbar. Welche Codeanteile müssen immer wieder angefasst werden, um neue Features hinzuzufügen? Diese Information über alle Features und die Codebasis hinweg ist für den Menschen nicht ersichtlich und wird von anderen Werkzeugen nicht sichtbar gemacht. Die Messung dieser Information bedarf einer neuen Art von Metriken.

Dass das Entkoppeln sich lohnt und sich positiv auf den Feature-Durchsatz auswirkt, kann man folgendermaßen nachvollziehen: Diverse Untersuchungen zeigen, dass Entwickler 50 bis 80 Prozent der Zeit mit dem Lesen/Verstehen von Code verbringen.

"Clean Code" by Robert Martin

Bob enters the module.
He scrolls down to the function needing change.
He pauses, considering his options.
Oh, he's scrolling up to the top of the module to check the initialization of a variable.
Now he scrolls back down and begins to type.
Ooops, he's erasing what he typed!
He types it again.
He erases it again!
He types half of something else but then erases that!
He scrolls down to another function that calls the function he's changing to see how it is called.
He scrolls back up and types the same code he just erased.
He pauses.
He pops up another window and looks at a subclass. Is that function overridden?

Abb. 1: Auszug aus „Clean Code" von Robert Martin

Da fällt uns ein Auszug aus Robert Martins Buch ein (s. Abb. 1), wo er den typischen Codierungsablauf beschreibt. Die Umgebung und Implikationen von neuem Code müssen verstanden werden. Zeit und unerwünschte Seiteneffekte können dann signifikant gespart werden, wenn der kognitive Aufwand zum Verstehen des Codes reduziert werden kann. Dies ist dann der Fall, wenn ein Modul klar definierte Verantwortlichkeiten hat und nur zu wenigen verwandten Features beiträgt.

Übrigens, um den Leseanteil beim Schreiben von Code zu berücksichtigen, haben wir auch eine eigene Aufwandsmessung eingeführt, die auf den Änderungen des Codes basiert: für das gesamte System, für Module oder pro Feature/Bug. Auch gelöschter Code bedeutet Aufwand, denn man muss erst verstehen, was man löscht.

? Wie würdet ihr DETANGLE in ein, zwei Sätzen beschreiben?

DETANGLE ist eine Analyse und KI-basierte Prognose erfolgskritischer Daten auf der geschäftsrelevanten Grundlage der Features (deren Entwicklungsaufwand, Fehlerdichte und Qualität im Code) statt reiner Codeanalysen oder subjektiver Schätzungen der Entwickler, Manager und Kunden.

? Was genau leistet es? Welche Information und Sichten liefert es dem Nutzer? In welchen Prozessschritten der Entwicklung kommt es wie zum Einsatz?

Es gibt bei uns Views und Boards. Views geben einen Überblick, während Boards Bewertungen als Entscheidungshilfen darstellen.

Views geben beispielsweise Auskunft über die Größe des Systems, den geleisteten Entwicklungsaufwand, die Aufteilung dieser Werte über Module, Features und Bugs und deren Verlauf über die Zeit. Damit kann man zum Beispiel erkennen, ob der Aufwand für Features größer als der zum Bugfixing ist. Auch für Refactorings lässt sich sehen, wann diese Aufwände erbracht wurden und ob sie zum Beispiel zu geringeren Bugfixing-Aufwänden geführt haben. Die entscheidende Frage ist: Leistet man im Schnitt mehr Aufwand zur Featureentwicklung als für Verbesserungen und Bugfixing.

»Das Bauchgefühl ist kein guter Ratgeber«

Die Boards messen Feature-Qualitätsschulden und den Impact von Bugs. Sie zeigen auf, für welche Module hohe Qualitätsschulden anfallen und ob diese kritische Schwellenwerte erreicht haben. Es lässt sich erkennen, ob diese Qualitätsschulden nicht nur die Erweiterbarkeit mit neuen Features einschränken, sondern bereits zu mehr Bugs führen.

Ein weiteres Board stellt eine Aufwandsschätzung zur Verbesserung kritischer Module dem prognostizierten Wartungsaufwand für zu erwartendes Bugfixing gegenüber.

Es gibt mehrere Prozessschritte für den Einsatz von DETANGLE. Bei jeder Releaseplanung, wenn Verbesserungsmaßnahmen geplant werden oder zum Iterationsende zu Monitoring-Zwecken für Architekten. Auch als Teil von Continuous Integration kann DETANGLE mit jedem Commit getriggert werden.

? Welche Arten von Nutzern adressiert DETANGLE – Architekten, Entwickler oder auch Entscheidungsträger?

DETANGLE adressiert Architekten und Entscheidungsträger. Entwickler sind eher an Ergebnissen von Codeanalysen interessiert, um Bad Smells und Programmierfehler zu erkennen.

Entscheidungsträger und Kunden verstehen wenig von Code. Aber sie müssen entscheiden, wofür das Budget verwendet wird. Dabei über Features in der Domäne des Kunden zu sprechen, hilft ungemein. Ein Manager/Kunde kann leicht folgen, wenn man von Features spricht. Und er kann sich an den Werten der Feature-Modularität orientieren, ohne das Konzept im Einzelnen zu verstehen. Mit Zahlen zu Aufwand und Qualität von Features, datenbasierten Aufwandsschätzungen und Wartungsprognosen kann er hantieren, um weitere Entscheidungen zu treffen.

»Es geht darum, das vorhandene Budget möglichst effektiv einzusetzen«

? Auf eurer Webseite beschreibt ihr als Vorteile verbesserte Qualität, Einsichten über Beobachten beziehungsweise Monitoring und bessere Planung. Könnt ihr bitte erläutern, wie DETANGLE diese Ziele erreicht?

Wie erwähnt, betreiben wir ein Monitoring der Aufwände pro Feature. Wir können identifizieren, wann Aufwände für Bugfixing diejenigen für Features übersteigen. Des Weiteren trifft die Paretoregel zu: 80 Prozent der Aufwände werden für 20 Prozent der Features aufgebracht. Da lohnt es sich, diese 20 Prozent genauer zu betrachten. Lohnt sich weiterer Aufwand überhaupt?

Budget für Qualitätsverbesserungen wird immer recht begrenzt sein. Mit DETANGLE ist es möglich, dieses Budget für die Verbesserung des Codes aufzuwenden, der zu geschäftsrelevanten Features beiträgt. Es geht darum, das vorhandene Budget möglichst effektiv einzusetzen.

»Mittels hoher Feature-Qualitätsschulden kann man Folgefehler prognostizieren und den kommenden Wartungsaufwand in Form von Bugfixes vorhersagen«

Die Planung dieser Verbesserungen versus weiterer Features unterstützen wir, indem wir Aufwandsschätzung für die Verbesserung und eine Wartungsprognose aufstellen, die man zur besseren Entscheidungsfindung heranzieht.

? Ihr habt auch einige Whitepapers erstellt, etwa über das Spannungsfeld zwischen Features und Qualität. Ihr gebt Anwendern Möglichkeiten an die Hand, entsprechende Entscheidungen leichter zu treffen, etwa welche Features in ein Produkt kommen sollten und welche nicht. Könnt ihr erläutern, wie DETANGLE bei diesen Entscheidungen hilft?

Wir messen mit DETANGLE unter anderem Feature-Qualitätsschulden. Basierend darauf können Kunden, Manager und Entwickler zusammen entscheiden, ob und wo die Qualität entlang des Geschäftsnutzens der Features verbessert werden muss.

Bei gefragten Features mit hohen Schulden muss man die Feature-Modularität verbessern, denn es ist mit Erweiterungen dieser Features zu rechnen. Oder weist der Code, wo ich neue Features hinzufügen werde, hohe Feature-Qualitätsschulden auf, so lohnt sich nur dort eine Verbesserung. Sind die Werte in dem Codebereich noch nicht kritisch, kann man weiter Features hinzufügen.

? DETANGLE verspricht Unterstützung für die Sicherstellung funktionaler Sicherheit. Wie genau erreicht ihr das? Über Nachverfolgbarkeit?

Das ist ein Aspekt, dem wir noch nicht in der nötigen Tiefe nachgehen konnten, aber wo wir ein Anwendungsfeld für DETANGLE sehen. Es leistet ja mehr als nur die Nachverfolgbarkeit von Funktionalität. Mittels Feature-Kopplung ließe sich auch die Rückkoppelungsfreiheit messen. Das wäre die Ausgangsthese, die man zusammen mit dem TÜV und einem Unternehmen in der Branche in einem Forschungsprojekt untersuchen könnte.

? Ebenso soll DETANGLE beim Management von Risiken helfen. Was versteht ihr konkret darunter und wie hilft mir euer Tool dabei?

Neben Features und Qualität gibt es in Softwareprojekten noch weitere Schlüsselspieler, nämlich Menschen – und ihr Wissen. Es geht um die Risiken der Wissensverteilung in Teams. Wo bestehen Wissensinseln, wo Wissensklumpen? Wer sind die Koordinatoren im Team (Key-People)? Wie verhält es sich mit dem Wissens-Ramp-up bei neuen Mitarbeitern? Optimalvorstellung wäre: Neue Mitarbeiter fangen mit Bugfixing an, gehen zur Features über, dann zu Refactorings und werden irgendwann Koordinatoren.

Die Frage bezüglich Wissensverteilung ist, wie viele Entwickler optimalerweise direkt zusammenarbeiten sollten. Wenn es zu wenige sind (z. B. nur einer), dann verliert man das Know-how, falls sie das Projekt verlassen, und muss einen hohen Einarbeitungsaufwand in Kauf nehmen. Wenn es zu viele sind, dann kann es zum Mehraufwand durch Kommunikations-Overhead kommen.

Es gibt in diesem Zusammenhang das „Diffusion of Responsibility"-Phänomen. Wenn an eine Unfallstelle gleichzeitig zu viele Menschen ankommen, dann sinkt die Wahrscheinlichkeit, dass zielgerichtet geholfen wird, weil jeder davon ausgeht, dass der andere es macht. Wenn also zu viele Entwickler an einem Modul arbeiten, fühlt sich am Ende niemand mehr für die Qualität verantwortlich.

Es zeigt sich, dass für Entwickler ähnliche Modularitätsprinzipien gelten wie für Features. Die „Committer-Kopplung" ist eine Metrik, die wir in diesem Zusammenhang erfassen. Außerdem visualisieren wir die Wissensverteilung, womit man schon qualitativ viel erkennen kann.

Wie viele Entwickler sollten optimalerweise an einem Modul arbeiten? Eine exakte Zahl ist schwer zu nennen. Aber unsere Untersuchungen und auch andere empirische Erkenntnisse zeigen, dass diese im Bereich von 2 liegt. Da kommt uns ein Foto in den Sinn, das gut dazu passt. Es stammt von einem Talk vom Logitech-CEO Bracken P. Darrell auf der TNW Conference 2019. (s. Abb. 2). Es verweist auf die erfolgreichen echten beziehungsweise fiktiven Paare Jobs/Wozniak und Holmes/Watson.

Abb. 2: Logitech-CEO Bracken P. Darrell auf der TNW Conference 2019

Eine anderer Anwendungsfall zielt auf die Abschätzung von Risiken beim Verlust von Mitarbeitern oder die Bestimmung von kritischen Mitarbeitern bei Firmenkäufen oder Team-Umstrukturierungen.

Gibt es typische Fehler oder Probleme, die ihr bei Softwaresystemen auffällig häufig antrefft?

Da wäre zunächst bei der Prozessqualität anzusetzen. Mittlerweile erfassen immer mehr Projekte Anforderungen und Bugs als Tickets und assoziieren Commits mit einer Ticket-ID. Die allerwenigsten erfassen aber technische Arbeiten, wie Refactorings, als Tickets. Damit ist man nicht in der Lage, den Effekt dieser Arbeiten zu erkennen, zum Beispiel ob es anschließend weniger Bugs gibt.

Diese technischen Arbeiten können nicht nur positive Auswirkung haben. Beispielsweise bei Django, dem Webframework, gibt es den Tickettyp „Cleanup/Optimization", unter dem technische Arbeiten erfasst werden. Diese Arbeiten haben dazu geführt, dass mit 2017 die Feature-Qualitätsschulden niedriger wurden, das heißt, neue Features sind weniger gekoppelt zueinander. Dennoch sind die Bugs in gewissen Codebereichen kaum zurückgegangen. Das lag aber an der hohen Kopplung der technischen Verbesserungen und der Committer. Das ist schon überraschend: Viele Leute arbeiten gleichzeitig an Verbesserungen auf eine leicht chaotische Art und Weise.

Des Weiteren gibt es teilweise wenig bis gar keine Sourcecode-Dokumentation mit für uns abenteuerlichen Ausreden. Eine Analyse dazu mit einer eigenen View ist bei DETANGLE bereits in der Mache.

Und oftmals gibt es alarmierend wenig automatisierte Tests, stattdessen verschwenden die Menschen vor jedem Release sehr viel Zeit mit manuellem Testing.

Lässt sich euer Werkzeug mit anderen Tools integrieren? Ich dachte dabei an Test-, Modellierungs- und Entwicklungswerkzeuge sowie Versionsmanagement und die ganze Palette sonstiger Tools.

Eine Integration mit Code-Repositories und Issue-Trackern ist per Definition gegeben, weil es die wichtigsten Informationsquellen für DETANGLE sind.

Eine Integration mit Testwerkzeugen ist in mehrfacher Hinsicht sinnvoll. Ergebnisse von Testläufen, Coverage- und Mutation-Testing-Tools können leicht von unserer Analyse-Engine erfasst und in unser Datenmodell eingespeist werden. Wir wollen aber nicht zu viel verraten, was wir da eventuell angehen wollen.

Wie erfolgt die Integration in eine DevOps-Umgebung?

DETANGLE muss einmalig konfiguriert werden und lässt sich leicht in den CI/CD-Ablauf integrieren. Mit jedem Nightly Build kann man die DETANGLE-Analyse erneut anstoßen und die Visualisierungen aktualisieren oder Benachrichtigungen generieren lassen.

Unterstützt ihr auch Systemarchitekturen mit mehreren Gewerken, etwa Mechatronik, Elektronik, IT-Infrastrukturen?

Nein. IT-Infrastrukturen würden sicherlich auch sinnvoll sein …

Würde sich ein Unternehmen für den Einsatz von DETANGLE entscheiden, wie genau läuft das Ganze ab, bis es operativ nutzbar ist?

Es sind ca. zwei Wochen Aufwand als Beratungsleistung inkl. Aufsetzen, Durchführung und Vorstellung der Ergebnisse. Der Kunde erhält die Analyseergebnisse, Views und Boards per Dockerimage.

Gibt es eine Testmöglichkeit?

Zurzeit in Form eines von uns begleiteten Proof of Concepts.

Wenn ihr so fünf Jahre in die Zukunft schaut, wo wollt ihr dann stehen?

DETANGLE soll ein „offizielles" Gütesiegel werden. Zum Beispiel bei der Beurteilung der Qualität des Entwicklungsprozesses oder der Erfüllung bestimmter Richtlinien, wie der kürzlich von der BaFin veröffentlichen KAIT-Richtlinien (Kapitalverwaltungsaufsichtliche Anforderungen an die IT).

Es gibt noch mehr Aspekte, die man mithilfe des maschinellen Lernens beurteilen kann, zum Beispiel die Qualität von Commit-Messages und Ticket-Inhalten – beides wichtige Kriterien für die Qualität des Entwicklungsprozesses.

»Es geht uns darum, dass man in einem Meer von Code die Fähigkeit besitzen sollte, gute und schlechte Stücke zu unterscheiden«

Ich bin neugierig. Ihr nennt euer Unternehmen Cape Of Good Code. Wie seid ihr eigentlich auf den Namen gekommen?

Um ehrlich zu sein, war der Name ein Vorschlag von einem Freund von Konstantin (Oleg Pomjanski), der uns unter anderem beim Design des Logos geholfen hat. Wir fanden den Namen auf Anhieb gut und konnten uns damit sofort identifizieren.

Wir haben bisher nur positive Reaktionen bekommen. Selbst die Notarin (bei der GmbH-Gründung) fand den Namen „cool".

Bei der Identifikation mit dem Unternehmensnamen geht es uns darum, dass man in einem *Meer von Code* die Fähigkeit besitzen sollte, gute und schlechte Stücke zu unterscheiden. Wir bieten einen Orientierungspunkt, wie ein Leuchtturm weisen wir unseren Kunden die Richtung.

Am Schluss noch eine völlig andere Frage: Ihr wart zuvor in Unternehmen tätig. Aufgrund der damaligen und eurer jetzigen Erfahrung, was würdet ihr jetzigen beziehungsweise baldigen Berufseinsteigern als Ratschlag mit auf den Weg geben?

Egon: In Deutschland herrscht die Klage, dass wir zu wenig geschäftsorientiert denken. Dennoch möchten wir eine Lanze für die „Technik" brechen: sich tief mit technischen Gesichtspunkten auseinandergesetzt zu haben, gibt einem das Selbstvertrauen, alles, was da kommt, lernen zu können.

Wenn man zwischendurch eine vielversprechende Idee hat, dann sollte der Grundsatz lauten, dass man sich Weggefährten sucht. Und sich selbst nicht so wichtig nimmt, aber die anderen umso mehr. Denn nur mit der Vielfalt der Perspektiven wird daraus eine bessere und tragfähige Lösung.

Und denkt immer daran: Versucht mit eurer Lösung jemanden glücklich zu machen.

Konstantin: Habt keine Angst davor, euren Job zu verlieren. Die meisten von euch werden 100 und älter. Die Auswirkung von so etwas auf euer Leben ist vernachlässigbar klein. Außerdem: Meistens bereut man nur das, was man nicht getan hat, also wenn ihr im Zweifel seid, dann tut es!

Das Interview führte Dr. Michael Stal, E-Mail: michael.stal@gmail.com