Die unverzichtbare Grundlage für KPIs in der Embedded Softwareentwicklung

Optimierung von Softwareprojekten

Litha Hari, Thomas Hunter, Jason R. Rolles, BlueOptima

Zusammenfassung

Diese Arbeit untersucht Leistungskennzahlen, die für die Optimierung des Preis-Leistungs-Verhältnisses bei der Softwareentwicklung von entscheidender Bedeutung sind, beispielsweise:

- **genaue Zeitschätzung zur Festlegung realistischer Erwartungen und Budgets**
- **Auswahl der geeignetsten Outsourcing-Lieferanten/Personen für jedes Projekt**
- **Zusammenarbeit mit Lieferanten/Managern zur Erzielung optimaler Ergebnisse**
- **die besten Entscheidungen während jedes Auftrags, auch mit der Fähigkeit, die zugewiesenen Softwareentwickler proaktiv zu ändern, um die besten Ergebnisse zu erzielen.**

BlueOptima bietet einen beispiellosen Einblick in diese Aspekte: Die SaaS-Analyse-Plattform bietet ein detailliertes Verständnis des Preis-Leistungs-Verhältnisses, das Softwareentwickler, Teams und Outsourcer auf der Grundlage objektiver, vertretbarer Einheiten intellektueller Anstrengung bieten. Dieser bewährte Ansatz, der auf objektiven und konsistenten Kennzahlen basiert, überwindet die Lücken anderer Ansätze und versetzt Unternehmen in die Lage, die besten Entscheidungen zu treffen sowie erhebliche Kosteneinsparungspotenziale zu nutzen.

Vorwort

„Industrieunternehmen sind im Informations-Business, ob sie wollen oder nicht“

– Jeff Immelt[1]

Herausforderungen entwickeln

Software-Engineering gewinnt im Embedded Kontext zunehmend an Bedeutung, zum Beispiel im Automobilbereich: „Innovation, neue Produkte sowie Forschung und Entwicklung wurden von 21 % der Automobil-CIOs als Top-Priorität genannt, gegenüber nur 13 % aller Unternehmen.“[2] Warum diese Hervorhebung? „Die Automobilindustrie befindet sich mitten in einer Transformation, da das Kernprodukt eines Automobils die geladene Software wird und nicht der Motor, der es antreibt.“[3]

Die Notwendigkeit, Software zu entwickeln, um wettbewerbsfähig zu bleiben, ist für eine effiziente Projektdurchführung von großer Bedeutung. Aber „trotz der besten verfügbaren Beratung werden Informationssysteme nicht mit dem gleichen Grad an Zuverlässigkeit, Integrität und Vorhersehbarkeit gebaut wie andere Ingenieursdisziplinen.“[4] Gutes Projektmanagement – um Zeit- und Budget-Überschreitungen zu vermeiden – erfordert eine genaue Überwachung des Fortschritts. Dies ist eine besondere Herausforderung bei der ausgelagerten Softwareentwicklung.

Gute Arbeit zu leisten, erfordert effektive Leistungskennzahlen (KPIs). In diesem Beitrag wird untersucht, welche KPIs implementiert und wie sie aufgebaut sein sollten, um ein optimales Preis-Leistungs-Verhältnis bei der Entwicklung eingebetteter Software zu erzielen, sei es hausintern oder ausgelagert.

Grundmaße spezifizieren

Die Zuverlässigkeit von KPIs hängt von den zugrunde liegenden Kennzahlen ab, aus denen sie aufgebaut sind. Daher müssen wir zunächst die Stärken und Schwächen der verfügbaren Basiskennzahlen untersuchen.

Ein grundlegendes Ziel des Softwareentwicklungsmanagements ist es, die Produktionskosten für die gelieferte Arbeit im Vergleich zu internen oder lokalen Ressourcen zu minimieren (sofern das Ergebnis eine akzeptable Qualität übersteigt). Kennzahlen, die zur Überwachung des Fortschritts auf dem Weg zu diesem grundlegenden Ziel der Kostenoptimierung je Qualität der Arbeit herangezogen werden, müssen kontinuierlich und klar definiert werden:

- **ein Messwert für den Aufwand, der in die Bereitstellung von Softwareentwicklungsdienstleistungen investiert wird,**
  und
- **entsprechende Maßnahmen der Qualität der Ergebnisse.**

Daher müssen Grundmaße KPIs für diese beiden Faktoren unterstützen.

Grundmaße der Softwareentwicklungsproduktivität

Es gibt verschiedene Möglichkeiten, wie eine Organisation die Leistung von Programmierern messen kann, um Softwareversionen oder Projekte zu liefern. Nachfolgend finden Sie einige der allgemein verwendeten Grundmaße der Softwareentwicklungsproduktivität, die zusammen mit den Stärken und Schwächen der einzelnen Maßnahmen in übergeordnete KPIs eingehen können.

Maßnahme und **Anwendung**	**Vorteile**	**Nachteile**
<u>Versionskosten</u> **Hausintern:** Das pro Version zugewiesene Budget auf einer Plattform. In vielen Fällen handelt es sich hierbei um eine grobe FTE (Full Time Equivalent)-Zuordnung. **Ausgelagert:** Rechnungen, die von Lieferanten eingehen, basierend auf Zeit und Material oder Festpreis.	Genaue Angabe der direkten monetären Kosten, die zur Realisierung der Version anfallen. Leicht vergleichbar mit anderen Plattformen, Methoden und Anbietern.	Gibt keinen Hinweis auf den Arbeitsaufwand, der für die Version erforderlich war. Gibt keinen Einblick in den Gesundheitszustand einer Version im laufenden Betrieb, d. h. Probleme werden erst sichtbar, wenn sie einen erheblichen zusätzlichen Kostenaufwand für Ihr Unternehmen bedeuten.

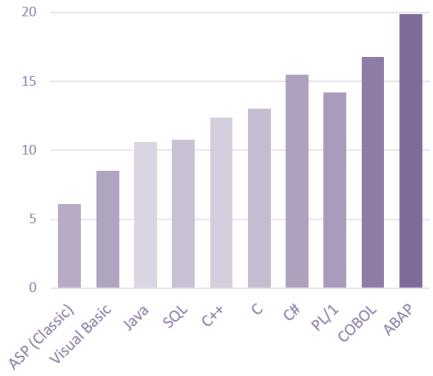
(Fortsetzung nächste Seite)

Maßnahme und **Anwendung**	**Vorteile**	**Nachteile**
<u>Codezeilen</u> **Hausintern:** Die direkte Zählung der von Entwicklern gelieferten Codezeilen, in der Regel basierend auf Zeitpunkt-„Snapshots“. **Ausgelagert:** Wie hausintern.	Leicht zu erfassende Daten. Können unabhängig und objektiv gesammelt werden. Die Daten sind für Entwickler, Manager und andere Stakeholder leicht verständlich.	Nicht direkt auf gelieferte Arbeitsaufgaben zurückführbar. Berücksichtigt nicht alle Änderungen zwischen Snapshots. Berücksichtigt keinen Code, der mit Hilfe von IDEs oder Bibliotheken kopiert und eingefügt/bearbeitet wird. Korreliert nicht verlässlich mit investiertem Aufwand oder Wertschöpfung einer Version. Technologieübergreifend kaum vergleichbar.
<u>Personenstunden (Selbsteinschätzung)</u> **Hausintern:** Im Detail die Zeit, die benötigt wird, um spezifische Aufgaben die an Entwickler vergeben wurden zu erfüllen, anderenfalls die benötigte Zeit für eine Version. **Ausgelagert:** Die Zeit, die ein Lieferant für bestimmte Aufgaben oder Gesamtversionen auf der Basis von Festpreis oder Zeit und Material benötigt.	Leicht verständlich für Softwareentwickler, die aufgefordert werden, die Daten zur Verfügung zu stellen, sowie für diejenigen, die sie verwenden. Wird vermutlich bereits getan, zumindest innerhalb einiger Teams.	Verfolgung von Stunden erfordert Zeit, die nicht direkt zur Lieferung der Software beiträgt; Entwickler vernachlässigen dies, insbesondere wenn dringende Lieferprobleme auftreten. Die meisten Organisationen beobachten deutliche Unterschiede in der Zeiterfassung und Granularität der einzelnen Teams in einer Organisation. Abhängigkeit von vorsätzlicher Manipulation und unbeabsichtigter Ungenauigkeit.

(Fortsetzung nächste Seite)

Maßnahme und **Anwendung**	**Vorteile**	**Nachteile**
<u>Personenstunden (abgeleitet)</u> **Hausintern:** Die Zeit, die in Aufgaben/Versionen investiert wurde, basierend auf einer angenommenen Anzahl von Arbeitsstunden während eines typischen Tages und der Anzahl der Personen, die an einem Projekt arbeiten. **Ausgelagert:** Wie hausintern.	Einfache Berechnung basierend auf High-Level-Planung und Budgetierung. Leicht verständlich und kommunizierbar für Führungskräfte.	Bietet keine Granularität auf Aufgabenebene. Geht davon aus, dass die Mitarbeiter durchgehend an den Projekten arbeiten, an denen sie für eine festgelegte Anzahl von Stunden arbeiten sollen. Auch wenn die Zeit, die tatsächlich an Projekten gearbeitet wurde, den Voraussagen entsprach, zeigt dies nicht, wie hart die Entwickler im Laufe der Zeit arbeiten.
<u>Funktionspunkte</u> **Hausintern:** Die Anzahl der Funktionspunkte, die ein Team bei der Entwicklung einer Anwendung liefert, ermöglicht eine Abschätzung des Funktionsumfangs der Anwendung. Damit kann der Durchsatz gemessen werden, den das Entwicklerteam liefert. **Ausgelagert:** Wie oben. Eine Organisation hat möglicherweise keine ausreichende Fähigkeit, Funktionspunkte zu zählen und muss sich darauf verlassen, dass der Outsourcing-Anbieter dies meldet.	Dieser Ansatz besteht seit mehr als 30 Jahren und verfügt über ein etabliertes Gremium von geschulten Funktionspunktzählexperten. Branchen-Benchmarks stehen zum Vergleich zur Verfügung.	Funktionspunkte variieren in Komplexität und Aufwand, sind also keine konsistente und faire Grundlage für den Teamvergleich. Äquivalente Funktionspunkte erfordern je nach Sprache einen unterschiedlichen Arbeitsaufwand (siehe Grafik). Es werden keine Entwicklungen berücksichtigt, die nicht zur Bereitstellung von Funktionen beitragen. Abhängig von Inter-/Intra-Auswerter-Unterschieden im Zeitablauf, da die Definition des Funktionspunkts subjektiv ist. Die Anzahl der zu liefernden Funktionspunkte wird in

Funktionspunkte werden zur Berechnung der funktionalen Größenmessung (Functional Size Measurement, FSM) von Software verwendet; ein standardisierter Ansatz zur Messung der in einer Anwendung oder Codebasis gehaltenen **Funktionalität** und zur Demonstration des Projektfortschritts. Obwohl es möglich ist, die Änderung der Gesamtfunktionalität einer Version zu messen, korreliert dies nicht mit dem damit verbundenen Aufwand, da jede Funktion eine andere Summe und Komplexität der Arbeit erfordert. Darüber hinaus erfordert die Bereitstellung der gleichen Funktion in verschiedenen Programmiersprachen einen unterschiedlichen Aufwand[1], wie unten dargestellt.



Entwicklungs-stunden pro Funktions-punkt nach den wichtigsten Programmier-sprachen

der Planungsphase vorgegeben und wird in der Regel erhöht, wenn dieser Ansatz in Kraft ist.

Das Erfassen von Daten ist teuer, weil es manuell und iterativ nachgezählt werden muss.

Daten im Sinn von Funktionspunkten sind für Stakeholder ohne Software-entwicklungshintergrund als Maßstab für Preis-Leistungs-Verhältnis oder Produktivität schwer zu verstehen.

Geben nicht an, wie viel Arbeit einzelne Entwickler in einem Team leisten, wenn mehrere Entwickler zu jeder Funktion beitragen. Wenn Entwickler komplette Funktionen besitzen, liefern erfahrenere Entwickler die komplexeren, d. h. schwierigeren Funktionen.

Die oben genannten Aufwandsgrößen können manuell aus verschiedenen Quellen ermittelt und zum Teil mehr oder weniger automatisiert werden. Generell bringt die Automatisierung den Vorteil mit sich, dass Entwickler keine Zeit für das Reporting aufwenden müssen und sie unterstützt die Objektivität. Wir müssen jedoch aufpassen, dass keine der oben genannten Maßnahmen wirklich objektive oder verlässliche Maßstäbe für den Entwicklungsaufwand der Entwickler sind und – schlimmer noch – irreführend sein können und die Arbeitsgewohnheiten der Entwickler von der Softwareproduktion wegführen können.

Eine erweiterte Maßnahme

BlueOptimas Technologie bietet ein durchgängiges Maß für den intellektuellen Aufwand, der in die Entwicklung eines Softwareprojekts oder einer Version investiert wird. Die Erfassung dieser Maßnahme, genannt BlueOptima ACE™ (Actual Coding Effort), ist vollständig automatisiert und kann auch retrospektive Metriken zu Versionen oder Projekten berechnen, die in der Vergangenheit entwickelt wurden.

BlueOptima integriert mit den Quellcode-Repositorien[5] und Task-Tracking-Systemen[6], die von Softwareentwicklungsteams verwendet werden. Der tatsächliche Aufwand für die Codierung wird berechnet, indem 36 verschiedene statische Metriken der Änderungen ausgewertet werden, die alle Entwickler in die Codebasis einbringen. Diese bestimmen drei verschiedene Dimensionen des Fortschritts: ***Volumen*** der Änderung, ***Komplexität*** der Änderung und ***Wechselbeziehung*** der Änderung. Benchmarking-Algorithmen ermitteln dann geeignete Gewichtungen für die 36 statischen Metriken, basierend auf der Analyse einer großen Probenmenge vergangener Commits im Datenspeicher. BlueOptima bietet damit einen beispiellosen Einblick in den Aufwand, der rückwirkend oder gleichzeitig und völlig objektiv in die Entwicklungsarbeit investiert wird.

Grundmaße der Softwareentwicklungsqualität

Die nachstehende Tabelle zeigt Grundmaße zur Messung der Lieferqualität.

Maßnahme und **Anwendung**	**Vorteile**	**Nachteile**
<u>Regelverstöße</u> **Hausintern:** Zählen von Verstößen gegen regelbasierte Standards mit Hilfe von IDE-basierten Tools, anderer Software oder zentralisierten metrischen Berechnungs- und Aggregationsquellen, mit dem Ziel, Technical Debt zu messen. **Ausgelagert:** Wie oben.	Integrierbar in Quellcode/ Task-Repositorien, dadurch zentrale und einheitliche Daten. Mit geeigneten Werkzeugen und trotz der damit verbundenen Konfiguration können Vergleiche zwischen verschiedenen Plattformen erstellt werden.	Nicht sprachunabhängig, bietet daher keinen fairen Vergleich der gelieferten Arbeit. Nicht ein Maß für den Output von Softwareentwicklern, sondern für die erhöhte Wahrscheinlichkeit von Wartbarkeitsproblemen einer Codebasis.
<u>Fehlerzahl</u> **Hausintern:** Anzahl der Fehler, die nach dem Zeitpunkt der Entwicklung in einer traditionellen Projektmethodik erkannt wurden und für die Entwicklung von Agile-Projekten angepasst wurden. Kann in	Ermöglicht eine unkomplizierte Messung der Versionsqualität, die auf allen Ebenen einer Organisation leicht verständlich ist. Nutzt Daten, die bereits im Softwareentwicklungs-prozess erfasst wurden.	Misst nicht den Output von Softwareentwicklungsteams; beschränkt sich auf die Angabe der Wartbarkeit des aktuellen Softwareprodukts und des zusätzlichen Aufwands. Hinsichtlich der Größe oder Bedeutung von Mängeln nicht projektübergreifend oder -

verschiedenen Phasen auftreten: SIT, UAT und Produktion. Fehler können nach Schweregrad gegliedert werden. **Ausgelagert:** Wie oben, wo jedoch auch Tests durch ausgelagerte Entwicklungsdienstleister durchgeführt werden, wird diese Zählung selbst erfasst.		intern vergleichbar; der Aufwand zur Behebung oder das Ausmaß, um den sie die Wartbarkeit und Stabilität mindern. Nicht plattformübergreifend oder zeitlich vergleichbar, da Fehlerklassifikationen und Reportinggranularität unterschiedlich sind. Nicht vergleichbar mit anderen Methoden, z.B. Agile-Team hält die Fehler niedrig.

Verschiedene Produkte[7] bieten regelbasierte Maßnahmen zur Code-Qualität, wie Unit-Testabdeckung, Bereiche mit hoher Komplexität und taktische Compliance. Diese wichtigen Kennzahlen einer Codebasis geben Aufschluss über Anzahl, Schweregrad und Wahrscheinlichkeit potenzieller Probleme der Wartbarkeit, die Softwareentwickler hinzufügen. Dieser Ansatz beziffert weder genau noch verlässlich die Qualität der erstellten Software noch die Fehler, die behoben werden müssen: Er ist kein Maß für die Lieferqualität, das den Umfang der noch anzusprechenden technischen Schuld berücksichtigt. Er bietet auch kein Verständnis für den Beitrag einzelner Entwickler zu qualitativ hochwertigem und minderwertigem Code.

Eine fortschrittlichere Methode zur Beurteilung der Softwarequalität ist BlueOptima ART™ (Analysis of Relative Thresholds). Dieser Ansatz ermittelt Schwellenwerte für Metriken, die für Quellcode-Dateien als Grundlage für die Bestimmung der Wahrscheinlichkeit technischer Schuld innerhalb einer Anwendung dienen. Diese Methodik ermöglicht einen quantitativen Vergleich der Qualität der verschiedenen Softwareentwicklungsprojekte einer Organisation. Anstatt Schwellenwerte auf der Grundlage branchenspezifischen Denkens zu bestimmen, identifiziert dieser Ansatz zunächst Schwellenwerte, die auf eine Organisation – und die von ihr entwickelten Anwendungen – zugeschnitten sind, basierend auf ihren Standards für Anwendungsdateimetriken, die eine normale Verteilung aufweisen. Damit wird anerkannt, dass die Softwareentwicklungsnormen der einzelnen Unternehmen individuell sind, sodass Benchmarks in der Branche keine Rolle spielen. Die vollständige Funktionsweise von BlueOptima ART geht über den Rahmen dieses Dokuments hinaus. Für weitere Informationen besuchen Sie bitte www.blueoptima.com/art oder kontaktieren Sie uns.

BlueOptima ART ist in der Lage, die Menge des potenziell problematischen Codes im Hinblick auf den tatsächlichen Codierungsaufwand zu quantifizieren; die berechnete Produktivitätskennzahl liefert ein einheitliches Maß nicht normgerechter Arbeit, die in projekt- und teamübergreifender Softwareentwicklung investiert wurde.

BlueOptima zeigt, dass der Codieraufwand zur Behebung von Fehlern einen weiten Bereich umfasst.

Verteilung des tatsächlichen Codieraufwands je behobenem Fehler für das 25., 50., 75. und 100. Percentil.

Erstellen von Leistungskennzahlen aus Grundmaßen

Wir haben nun Grundmaße untersucht, die als Grundlage für die Erstellung übergeordneter KPIs dienen könnten, die Managern von Softwareentwicklungsprojekten wertvolle Erkenntnisse liefern. Welche Grundmaße sollten genutzt werden, um KPIs zu liefern, die Führungskräfte benötigen, um die erfolgreichsten Ergebnisse zuverlässig zu lenken?

KPI 1: Lieferrate

Ein starker Hinweis für produktive Softwareentwicklung ist die Geschwindigkeit, mit der Teams in Ihrer Codebasis während der Projekte oder Versionen arbeiten. Von Entwicklern, die einem Agile-Prozess folgen, kann man in den Sprint-Phasen einen relativ hohen Aufwand erwarten, vorausgesetzt, sie stoßen bei der Lieferung der benötigten Funktionalität nicht auf Hindernisse. In der anschließenden Phase des Business Acceptance Testing der Version wäre einer hohe Rate Anlass zur Besorgnis: Dies würde bedeuten, dass ein besonders großer Aufwand, der in die Behebung von Mängeln investiert wird, die während der Sprints nicht korrekt geliefert wurden, oder in die Erfüllung von Neuinterpretationen der Kundenanforderungen, für die in früheren Phasen deutlich mehr Fortschritte hätten erzielt werden müssen, investiert wird. Gleiches gilt für einen hohen Aufwand in der späteren Produktionsphase.

Die Leistungskennzahl Liefermenge kann auf verschiedene Weise erstellt werden, indem eine Kombination bereits untersuchter Grundmaße verwendet wird. Was wäre angemessen effektiv?

- **Versionskosten** bewertet die Gesamtkapitalrendite von Softwareentwicklungsversionen und nicht die Geschwindigkeit während des Fortschritts.

- **Codezeilen** könnten während des gesamten Projekts ein wachsendes Volumen der Codebasis liefern, aber dieses Grundmaß zeigt nicht den Projektfortschritt oder den Entwicklungsaufwand an, wie besprochen.

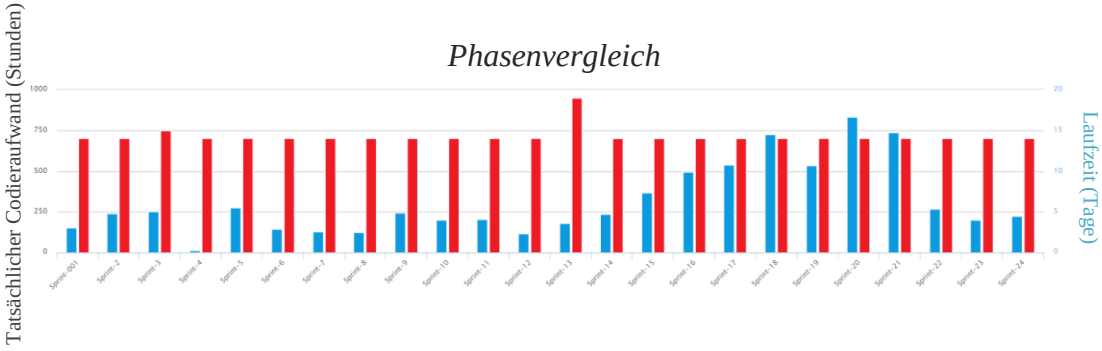
- **Personenstunden** dienen zur genaueren Angabe der Zeitdimension bei der Berechnung der Lieferrate (d. h. Arbeit je Zeiteinheit), was die genaue Zeit sein kann, für die eine Organisation bezahlt (bei der Berechnung der Arbeit je Kosten), aber keinen Hinweis auf den Aufwand gibt, der in irgendeinem Zeitraum oder Projektfortschritt investiert wurde.

- **Funktionspunkt**-Zählung zu verschiedenen Zeitpunkten zeigt den Fortschritt bei der Lieferung von Funktionalitäten an und kann als Rate über die Zeit für ein Projekt ausgedrückt werden (was jedoch keinen fairen Vergleich von Projekten/Lieferanten ermöglicht).

Der Fortschritt bei der Erfüllung von User Stories oder Story Points kann überwacht werden (wie die Agile Burndown Charts zeigen), um eine Lieferrate in der gleichen Art wie das Zählen von Funktionspunkten anzuzeigen. Da sie übergeordnet sind, sind Story Points ein weniger zuverlässiges Maß für die Arbeit, aber es wurde argumentiert, dass dies die beste Methode zur Messung der Produktivität ist[8]. User Stories, Story Points und Funktionspunkte sind noch weniger verlässliche Schätzungen des Aufwands in der Softwareentwicklung, da der Aufwand, den ein bestimmtes Team oder ein Dienstleister für die Erfüllung der jeweiligen Anforderungen aufwenden muss, möglicherweise nicht bekannt ist oder nicht offenbart wird und je nach Anbieter und Team variiert.

BlueOptima liefert ein Maß für den tatsächlichen Codieraufwand auf der Basis von 36 statischen Quellcode-Metriken, die jede Änderung an der Codebasis in den Dimensionen ***Volumen***, ***Komplexität*** und ***Zusammenhang*** von Änderungen bewerten. Die auf dem tatsächlichen Codieraufwand basierende Lieferrate kann verwendet werden, um die Leistung der Lieferanten über eine Vielzahl von Leistungen (beispielsweise Projekte oder Versionen) zu vergleichen.

Der nachstehende Screenshot zeigt den tatsächlichen Codieraufwand (je Stunden) – der linke Balken jedes Paares – investiert in jede der verschiedenen Sprint-Phasen – deren Dauer (je Tag) wird druch den rechten Balken dargestellt.



Der tatsächliche Codieraufwand je Tag gibt die Liefergeschwindigkeit an, die je Projektphase, Lieferung (beispielsweise Version), Lieferant oder Team berechnet werden kann.

KPI 2: Softwarequalität

Eine relativ hohe Lieferrate von Softwareentwicklungsteams ist nur dann von Wert, wenn die Software von akzeptabler Qualität ist. Damit ist die Softwarequalität ein weiterer kritischer KPI. „Qualität“ kann auf verschiedene Weise verstanden werden, sodass wir darauf achten müssen, dass unterschiedliche Vorstellungen zur Softwarequalität zwischen den Parteien eine Arbeitsbeziehung belasten können, noch bevor wir uns der Frage der quantitativen und konsistenten Messung nähern. Um solche Probleme von vornherein zu vermeiden, muss „Codequalität“ klar von „Lieferqualität“ unterschieden werden.

Codequalität basiert auf Metriken wie der Anzahl von Verstößen gegen Syntaxregeln gemäß Frameworks innerhalb verschiedener verfügbarer Werkzeuge[9]. Diese Verstöße zielen darauf ab, signifikante Fehler im Code zu identifizieren, die zu Fehlern und laufenden Wartungsarbeiten führen könnten. Code, der gegen diese Regeln verstößt, führt jedoch nicht immer zu Fehlern in der Software, d. h. die Regeln erzeugen Falschmeldungen und stratifizieren Fehler nicht in Bezug auf die Bedeutung ihrer potenziellen Probleme. Die Verwendung solcher Metriken als Maßstab für die Qualität einer Codebasis kann zu Meinungsverschiedenheiten zwischen Klienten und Anbietern führen, die näher an der Software in der Entwicklung sind und den Fortschritt mit erfüllter und fehlerhafter Funktionalität kennen. Die Beziehung kann durch subjektives Denken um bessere Möglichkeiten zur Bewertung von Softwarequalität im Hinblick auf die Eignung verschiedener Codierstile, Architekturen und Implementierungen noch weiter belastet und komplizierter werden. Die Summierung gültiger potenzieller Probleme stellt die technische Schuld (oder Codeschuld) einer Softwareversion dar, die Arbeit, die getan werden muss, um eine nutzbare Lösung zu erhalten, die alle Kundenanforderungen erfüllt, und die Software zu warten. Technical Debt ist selbst ein nützlicher Indikator für die kontinuierliche Wirkung und Wartbarkeit einer Softwareversion. Am Ende eines Projektzeitplans kommt eine klare Vorstellung von technischer Schuld ans Tageslicht, aber für Manager, die die Effektivität von Softwareentwicklungsprojekten überwachen, müssen sie den Aufbau von technischer Schuld während der Produktion minimieren. BlueOptima ART, zuvor vorgestellt, bietet eine genauere Anzeige der Höhe der technischen Schuld, die ein Projekt mit fortschreitender Entwicklung anhäuft. Im Gegensatz zu anderen Softwarequalitäts-Tools definiert BlueOptima ART geeignete statische metrische Schwellenwerte, die auf der beobachteten Bandbreite von Softwareentwicklungspraktiken innerhalb einer Organisation und nicht auf Branchen- oder De-facto-Standards basieren, und kennzeichnet damit weniger Falschmeldungen. Die Identifizierung der ungewöhnlichsten Dateisammlungen in Ihrer Codebasis hilft Teammanagern dabei, Zeitprioritäten für die wichtigsten Bereiche festzulegen.

Lieferqualität ist insbesondere der anteilige Aufwand zur ***Nachbesserung*** oder ***Mängelbeseitigung*** der Software. Daher muss ermittelt werden, wo der Aufwand in eine Codebasis investiert wird, um Qualitätsprobleme anzugehen, anstatt neue Funktionen zu liefern, und nur diesen Aufwand zu messen. Es gibt zwei Ansätze, diese beiden Arbeitsformen zu unterscheiden:

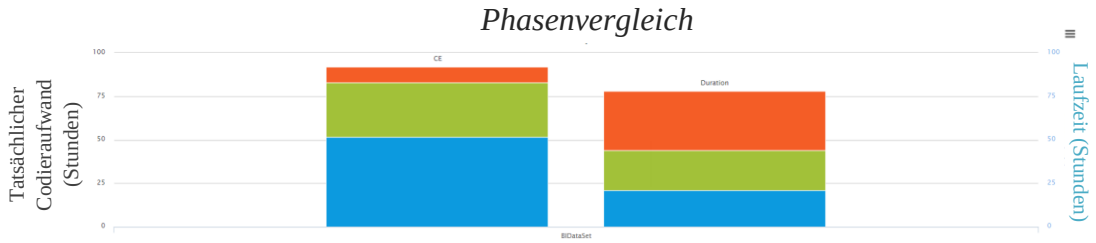
1. Der erste Ansatz erfordert die Ermittlung des Punktes während eines Softwareentwicklungsprojektes, bei dem alle Funktionen der Software und der entwicklergeführte Test (einschließlich der qualitätsorientierten Aktivitäten innerhalb von Agile Sprints) als abgeschlossen betrachtet werden. Darüber hinaus sind alle Arbeiten an der Software entweder auf Mängel oder auf eine Neuinterpretation der Endbenutzeranforderungen zurückzuführen. Alle wichtigen Stakeholder auf Kunden- und Anbieterseite müssen sich auf die Definition dieses Zeitpunkts einigen. Diese Abgrenzung ist erforderlich, um den Aufwand für die Phasen der Softwareerstellung nach anfänglichem Plan und Beginn qualitätsorientierter Phasen zu berechnen. Dieser Ansatz ist an sich problematisch innerhalb des Agile-Prozesses, in dem die Fehlerbehebung in Sprint-Phasen stattfindet und nicht in einer späteren, dedizierten Phase.
2. Der zweite Ansatz erfordert ein Verständnis für den Aufwand, der in jede Aufgabe investiert wird, an der Softwareentwickler arbeiten. Die Aufgaben werden in der Regel mit einer Task-Tracking-Software[10] verwaltet, in der sie typischerweise als geplante Erweiterungen oder Fehlerbehebung gekennzeichnet sind. Der Gesamteffekt für jede dieser beiden Kategorien lässt sich aus dem Aufwand, der in jede Aufgabe oder jeden Typ investiert wurde, zusammenfassen. Für Agile ist dies nach wie vor problematisch, da Defekte innerhalb der Sprint-Phasen behoben werden.

BlueOptima ART definiert Schwellenwerte für Metriken innerhalb verschiedener Eigenschaften für den Softwarebestand Ihrer Organisation als Grundlage für die Bewertung der Wartbarkeit von Code, der in Bezug auf den tatsächlichen Codieraufwand quantifiziert werden kann. Das Hervorheben der fehlerhaftesten Quellcode-Dateien dient als Indikator für technische Schuld und als Ausgangspunkt für die Priorisierung, welcher Code am lohnenswertesten ist, wenn man sich erneut mit Stabilitätsproblemen befasst. Dies liefert eine Momentaufnahme der Codequalität. Sie basiert auf Schwellenwerten, die an Ihre Organisation angepasst sind und nicht auf branchenüblichen Benchmarks oder Regeln, die von Softwarequalitäts-Tools vorgegeben werden. Der Umfang der Softwareentwicklungsarbeit – tatsächlicher Codieraufwand – als Überschreiten von Schwellenwerten identifiziert, und schlimmer noch – den Code weiter von den Normen zu drängen – lässt sich in Form von einzelnen Entwicklern, Teams oder Anbietern zusammenfassen. Dies ermöglicht einen fairen Vergleich der Liefermöglichkeiten für ein Projekt sowie ein Bild von der Qualität laufender Projekte.

BlueOptima ermöglicht die Bewertung der Lieferqualität durch Summierung des tatsächlichen Codieraufwands, der mit Aufgaben verbunden ist, welche die Build-Phase umfassen, im Gegensatz zu den Aufgaben, die für die Fehlerbehebung nach dieser Phase, sowie der Veröffentlichungsphase nach dem Testen erforderlich sind. Dies gibt Projektmanagern und Sponsoren ein Verständnis für die Proportionen des Aufwands, der in die verschiedenen Phasen investiert wurde, und ermöglicht ihnen, in detailliertere Projektebenen einzudringen, um weitere Erkenntnisse zu erhalten. Das Wissen um den Aufwand, der zur Behebung verschiedener Arten von

Problemen erforderlich ist, ist wertvoll, um die Größe der in Zukunft auftretenden Fehler zu verstehen und nachfolgende Korrekturen so zu planen, dass BlueOptima ein Verständnis der Anstrengungen liefert, die alle vergangenen Aufgaben erfordert haben, um eine genaue Einschätzung zukünftiger Projekte zu unterstützen.

Die nachstehende Abbildung zeigt den tatsächlichen Codieraufwand für Build-, Test- und Veröffentlichungsphasen von Projekten (blau, grün und rot). Wie bereits erwähnt, sollte ein möglichst großer Teil des gesamten tatsächlichen Codieraufwands innerhalb der ersten Phase liegen, und so wenig wie möglich in den beiden letztgenannten Phasen.



KPI 3: Lieferkosten

Die Lieferkosten der Software stellen ein wesentliches Maß für jedes Entwicklungsprojekt dar, mit dem Unternehmen typischerweise Kosteneinsparungen realisieren wollen. Die Kosten für eine Version sind einfach zu kalkulieren, aber für Führungskräfte ist ein klares Kosten-Nutzen-Verhältnis bei jedem einzelnen Projekt wichtig, das sie befähigt, das Projekt entlang des optimalen Pfades zu steuern. Eine faire und kontinuierliche Bewertung des Fortschritts erfordert zu jedem Zeitpunkt der Arbeit einheitliche, transparente Kennzahlen. Das Preis-Leistungs-Verhältnis sollte daher nicht auf der Grundlage von Funktionspunkten oder Story Points berechnet werden, die aufgrund von Inkonsistenzen bereits besprochen wurden. Vielmehr ist der Aufwand ein nützliches, vertretbares Maß für die „Größe“ eines Softwareentwicklungsprojekts. Je detaillierter das Aufwandmaß, desto präziser und wertvoller ist das Leistungsmerkmal Lieferkosten.

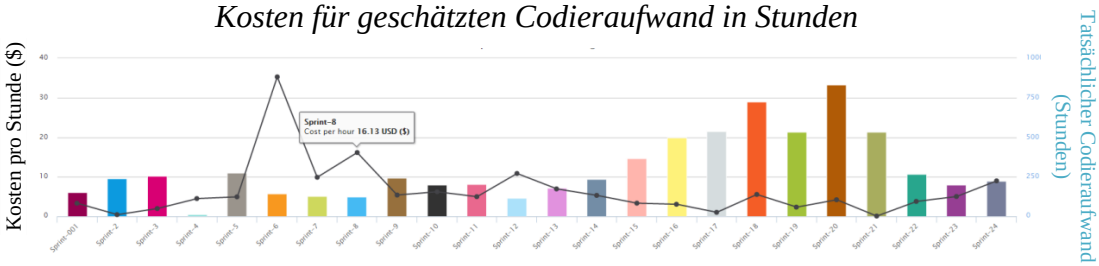
Wie bereits erwähnt, kann BlueOptima den tatsächlichen Codieraufwand in Form von individuellen Sprints oder sogar Aufgaben darstellen, was einen einheitlichen Aufwand über einen beliebigen Zeitraum darstellt. Dies gibt Managern die Möglichkeit, den tatsächlichen Codieraufwand je Aufgabe (im Vergleich zu der zugewiesenen Zeit) oder Zeiteinheit bzw. Kosteneinheit sichtbar zu halten. Umgekehrt ermöglichen die Kosten pro Stunde des tatsächlichen Codieraufwands einen fairen Vergleich der Lieferung zwischen den verschiedenen Anbietern oder Teams im Einsatz.

Der daraus resultierende Einblick in den tatsächlichen Codieraufwand bei der Bereitstellung bestimmter Funktionen oder Anwendungen liefert eine Grundlage für die Abschätzung des tatsächlichen Codieraufwands für zukünftige Projekte. Dieser gesamte tatsächliche Codieraufwand kann in Kosten für jeden verfügbaren Lieferanten umgewandelt werden, für den die Kosten je tatsächlichem

Codieraufwand bekannt sind. Dieser faire Vergleich ermöglicht eine optimale Lieferantenauswahl.

In der folgenden Grafik wurden die Softwareentwicklungsraten auf die Anzahl der Stunden pro Sprint umgerechnet, um die Kosten pro Stunde des tatsächlichen Codieraufwands (schwarze Linie) zu berechnen.

Kosten für geschätzten Codieraufwand in Stunden



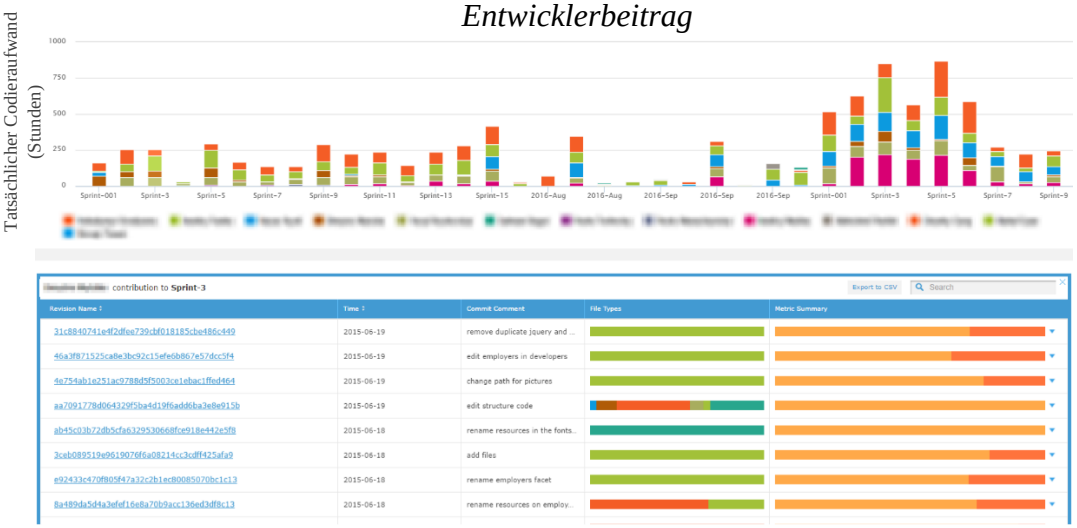
KPI 4: Entwicklertalent

Während die Gebühren (z.B. Kosten pro Stunde, Tag oder Monat) von Softwareentwicklern oder Outsourcing-Anbietern reine Daten sind, bestimmen nicht nur sie das Preis-Leistungs-Verhältnis, das die Entwickler oder Teams liefern können. Manager müssen auch wissen, wie gut Entwickler pro abrechenbarer Stunde liefern können. Die Abwanderung von Entwicklern ist ein Faktor, der mit Talenten gekoppelt sein sollte. Es ist wünschenswerter, dass Outsourcer oder Teams mit geringer Mitarbeiterfluktuation Ihre Softwareprojekte realisieren.

Die Beurteilung, welche Entwickler am effektivsten sind, ist eine naturgemäß subjektive Domäne, aber eine objektive Basis ist genauso kritisch wie für die drei bisher diskutierten KPIs.

Wie bereits erläutert, liefert BlueOptima den tatsächlichen Codieraufwand für jeden Hersteller, jedes Team oder jeden einzelnen Entwickler. Die nachstehende Grafik zeigt den tatsächlichen Programmieraufwand pro Sprint, aufgeschlüsselt nach den Entwicklern, die am Projekt arbeiten. Dies zeigt die einzelnen Beiträge von Entwicklern, die an einem Projekt arbeiten, mit – wenn ausgelagert – detaillierteren Einblicken, als es das Management innerhalb des Herstellers kennt. Nachfolgend sind die einzelnen Aufgaben gelistet, zu denen der Entwickler einen Beitrag leistet.

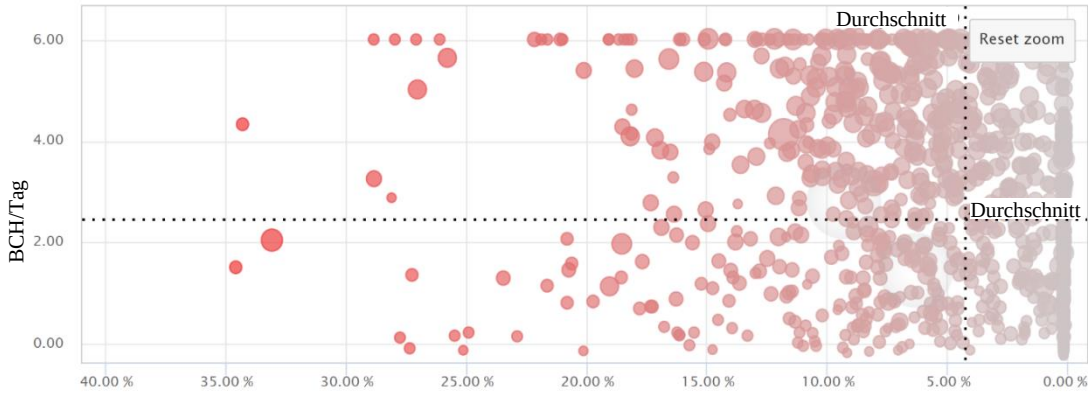
Entwicklerbeitrag



Wie bereits erwähnt, können einzelne Entwickler nicht nur nach ihrer Produktivität beurteilt werden, die Arbeit muss ebenfalls ein zufriedenstellendes Niveau haben. BlueOptima quantifiziert den tatsächlichen Codieraufwand, der mit der Änderung von Code verbunden ist so, dass die Schwellenwerte für verschiedene Quellcode-Datei-Metriken überschritten werden – oder dass der Grad der Quelltext-Dateinormung weiter ansteigt (diese Softwarequalitäts-Messlösung ist BlueOptima ART, wie bereits skizziert).

In der nachstehenden Visualisierung stellt jeder Knoten einen individuellen Entwickler dar, dargestellt durch relative Qualität (y-Achse) und Quantität (x-Achse) des von ihm produzierten Code. Die rechte Seite zeigt hohe Qualität, wobei die rechte obere Ecke Entwickler mit hohen Produktivitäts- und Qualitätsstandards zeigt. Die linke Seite zeigt Entwickler, die abweichenden Code produzieren. Entwickler, die sich in der linken oberen Ecke befinden, tragen aktiv zum Quellcode bei, aber mit so viel anormalem Code, dass die technische Schuld einen kontraproduktiven Beitrag insgesamt darstellt, und das Softwareentwicklungsprojekt ohne sie erfolgreicher wäre.

Es ist möglich, auf einen Knoten in dieser Tabelle innerhalb der BlueOptima SaaS-Benutzeroberfläche zu klicken, um mehr Informationen über den jeweiligen Entwickler zu erhalten (wie im Feld unten dargestellt).



Details of **70015 -Developer** in **1** project on **this year**

Drill Down in Analyze Coding Effort | Export to TSV | Search

Project name	Organization name	BCH/Day	Total BCH	% of Aberrant CE	% of BCH/project
Project-7233	Bank-16	5.00	15.00	0.44 %	100%

Anwendungsbeispiel – Lieferqualität

Die folgende Tabelle zeigt eine Reihe von Versionen, die von einem Outsourcing-Anbieter für ein globales Telekommunikationsunternehmen geliefert wurden. Die Plattform, auf der die Software entwickelt wurde, war bereits als Problem bei der

Verwaltung von Change Delivery bekannt – und auch als unpopulär bei den Stakeholdern. Die Kosten wurden in Relation zur Lieferqualität analysiert, um zu ermitteln, wie die Ausgaben und die Leistung verbessert werden können.

Version	Ist-Kosten	Build:Test %	Keine Verbesserung	Kosteneinsparung
Version C	£ 46,700	5:95	£ 46,700	£ -
Version D	£ 9,700	31:69	£ 12,960	£ 3,960
Version E	£202,364	57:43	£ 501,863	£ 299,499
Gesamt:	**£258,064**		**£ 561,523**	**£ 303,459**

Kosten, wenn keine Verbesserungen vorgenommen worden wären

Die Verwendung von BlueOptima zur Erstellung der vier in diesem Beitrag behandelten KPIs ergab, dass die einige Wochen zuvor an den Kunden gelieferte Version (Version C) in der Tat von sehr schlechter Lieferqualität war: Nur 5 % des gesamten tatsächlichen Codieraufwands wurden während der Build-Phase geliefert und ganze 95 %, nachdem das Produkt in den Händen des Unternehmens für die endgültige Abnahmeprüfung war. Idealerweise würden diese Proportionen mit 95 % der vor UAT geleisteten Arbeit umgekehrt sein. Nach Austausch und Diskussion dieser Daten mit dem Outsourcing-Anbieter wurden die Prozesse verbessert, wie aus den Metriken der beiden folgenden Versionen ersichtlich ist.

Die Kostenanalyse des Klienten stellte fest, dass der tatsächliche Codieraufwand, der in den Testphasen der Endabnahme geliefert wurde, fünfmal so groß war wie die Anzahl der Änderungen, die vor diesem Zeitpunkt vorgenommen wurden, d. h. der größte Teil des Wertes, für den der Klient bezahlte, war Arbeit in der Phase, die einen geringeren Aufwand bei gut geführten Projekten erfordern sollte. BlueOptima hat den Klienten befähigt, Verbesserungen durch die Leistungskennzahl Lieferqualität zu verfolgen, Kosten zu kontrollieren und bedeutende Einsparungen zu erzielen.

Ein weiterer Klient von BlueOptima profitierte von der Identifizierung von Kosteneinsparungen in Höhe von über 60 Millionen US-Dollar durch Analysen basierend auf dem tatsächlichen Codieraufwand.

Fazit

In diesem Beitrag wurden vier KPIs untersucht, die für die Kostenoptimierung bei der Verwaltung effektiver Softwareentwicklungsaufträge und der Verbesserung des Nutzens von einer Version zur nächsten entscheidend sind.

Nach eingehender Prüfung der Stärken und Schwächen der verwendeten Grundmaße und ihres Werts bei der Generierung von KPI-Daten stellen wir fest, dass deren Zuverlässigkeit – insbesondere im Hinblick auf die Schaffung von Erkenntnissen, die Softwareentwicklungsmanager brauchen. Nur BlueOptima liefert genaue,

verlässliche Daten über die Arbeitsleistung, die Softwareentwickler in ein Projekt einbringen, um aussagekräftige, transparente KPIs zu erstellen. BlueOptimas Technologie wurde entwickelt, um eine zuverlässige, objektive, aussagekräftige und transparente Methode zur Messung der Softwareentwicklungsproduktivität zu bieten, die fungible und faire Methoden zum Vergleich der Ergebnisse bietet.

Der Wert, den BlueOptima bei der Steigerung von Produktivität, Qualität und Kosteneffizienz liefert, stellt eine überzeugende Kapitalrendite dar. BlueOptima hat maßgeblich dazu beigetragen, den Erfolg und die Rentabilität von Softwareentwicklungsprojekten zu optimieren, selbst Transparenz in ausgelagerten Projekten zu schaffen und die Kunden-Lieferanten-Beziehungen zu verbessern. BlueOptima-Anwender erzielen in der Regel Produktivitätssteigerungen von bis zu 20 %. Dies geht in der Regel mit einer verbesserten Gesamtzufriedenheit in der Zusammenarbeit zwischen Kunden und Dienstleistern einher.

Diese Technologie ist das Ergebnis von 15 Jahren Forschung und Entwicklung, zunächst an der Cambridge University (Großbritannien), in Zusammenarbeit mit der Judge Business School, The Computer Laboratory und St John's Innovation Centre.

Um herauszufinden, wie BlueOptima Ihre Organisation unterstützen kann, kontaktieren Sie uns bitte telefonisch unter +44/0 207 100 8740 oder per E-Mail an enquiries@blueoptima.com.

For the English version of this paper (The Essential Basis for KPIs in Embedded Software Development) contact info@blueoptima.com or visit BlueOptima’s stand.

Quellenverzeichnis

1. Jeff Immelt, GE CEO bis 1. August 2017: *GE’s Jeff Immelt on digitizing in the industrial space* (McKinsey, 2015).
2. *2017 CIO Agenda: An Automotive Perspective*, Gartner, Februar 2017 (Figure 3: 61 Automotive of 2,581 CIOs in 93 countries)
3. Ibid.
4. Research Director bei Gartner, Darryl Carlton: *IT Projects Need Less Complexity, Not More Governance* (Gartner, 2015)
5. Beispielsweise Subversion, CVS, Clearcase, StartTeam usw.
6. Beispielsweise Intland codeBeamer RM
7. SonarSource ist ein hervorragendes Beispiel für Open Source.
8 *Enhancing the efficiency and effectiveness of application development*, McKinsey 2013.
9. SonarSource bietet hervorragende Dashboards derartiger Metriken.
10. Beispielsweise Intland Softwares codeBeamer Release Manager.