Softwarequalität mit Teamscale steuern

Die Software-Intelligence-Plattform Teamscale hilft Entwicklungs- und Testteams beim Messen der Code- und Produktqualität.

Von Dr. Carsten Weise und Christoph Singer

-tract

- Teamscale ist eine Software-Intelligence-Plattform, die Software analysiert, überwacht und verbessert.
- Das Tool führt die Ergebnisse statischer und dynamischer Tests mit dem Quelltextrepository zusammen.
- Dashboards geben einen schnellen Überblick über den Stand der Softwarequalität. Treemaps visualisieren unter anderem die Testüberdeckung.
- Teamscale verfolgt Änderungen im Code und ermittelt Testlücken.
- Zahlreiche Programmiersprachen, externe Werkzeuge zur statischen Codeanalyse und Testüberdeckung sowie Versionskontrollsysteme lassen sich einbinden.

Um in einer immer komplexeren und anspruchsvolleren Softwarewelt qualitativ hochwertige Produkte zu liefern, braucht es eine toolunterstützte Qualitätssteuerung. Techniken wie Continuous Integration/Deployment, DevOps, Internet of Things, Internet of Production, Design Thinking und Lean Development sind auf schnellere und kürzere Iterationen sowie kleinteilige Entwicklung angewiesen. Die kleinteiligen Zyklen in der Softwareentwicklung sind Grundlage hoher Softwarequalität: Eine Applikation wird nicht in einem Produktionsschritt zum fertigen, fehlerfreien Produkt. Stattdessen messen Entwicklerinnen und Entwickler deren

Qualität kontinuierlich während der Entwicklung und bessern gefundene Abweichungen vom Soll (Fehler) nach.

Dabei sind sie auf flexible, verlässliche Werkzeuge angewiesen, um schnelles Feedback zur Qualität zu bekommen und dabei nichts an Änderungen und Korrekturen zu übersehen. Solche Werkzeuge sind etwa CodeCity und Seerene sowie Teamscale, das die Firma CQSE (Continuous Quality in Software Engineering) entwickelt hat, ein 2009 gegründetes Spin-off des Lehrstuhls für Software und Systems Engineering der TU München.

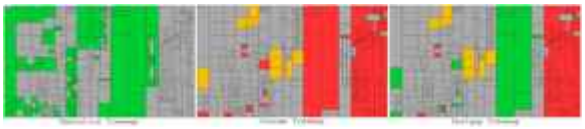
Teamscale sammelt Informationen zur Softwareerstellung und -prüfung und bereitet sie auf. Die Applikation ist ein Server, der über ein Webinterface erreichbar ist. Die Installation des Servers ist nativ, als Docker-Image oder auch als Cloud-Service möglich. Er lässt sich nicht nur über das Webinterface, sondern auch über eine REST-API ansteuern.

Teamscale berücksichtigt zwei wesentliche Prüfaktivitäten während der Softwareerstellung: die statische Analyse und den dynamischen Test (Test des ausführbaren Codes). Das Werkzeug hilft, verschiedene Testarten durchzuführen, die Testdaten zu sammeln und zu überwachen, ob notwendige Änderungen berücksichtigt und anstehende Korrekturen ausgeführt wurden. Der Artikel beschäftigt sich zunächst mit dem dynamischen Test, geht dann auf die statische Analyse ein und betrachtet als weiteres Feature die Architekturanalyse.

Codeänderung + Testüberdeckung = Testlücke

Wie andere Testüberwachungstools kann Teamscale Testüberdeckung messen und darstellen. Es verwendet für die Visualisierung Treemaps (Abbildung 1). Die Execution Treemap zeigt die Testüberdeckung, also die Überdeckung durch die Testausführung (Test Execution). Grüne Flächen sind bereits getestete Anteile, graue Flächen die noch ungetesteten Teile.

Die farbigen Flächen sind hierbei die Prozeduren (Methoden) des Codes. Die Größe der Fläche entspricht der Methodengröße (Komplexität), gemessen als Anzahl Codezeilen (LOC: Lines of Code).



Teamscale zeigt durch Treemaps Testüberdeckung, Codeänderungen und Testlücken an (Abb. 1).
Alle Qualitätsmessungen stellt Teamscale übersichtlich auf einem konfigurierbaren Dashboard dar. Dort findet man alle gängigen Darstellungsformen (beispielsweise Torten- und Balkendiagramme). Die Treemaps stechen dabei optisch hervor und liefern eine schnelle Übersicht für Überdeckungsmaße. Sie sind interaktiv und erlauben, auf die Details zu zoomen: So kann man zum Schluss auch auf detailliertere Information als die Funktionsüberdeckung zugreifen.

Teamscale kennt zusätzlich zur Testüberdeckung auch die Codeänderung (Change Treemap). Kombiniert man diese Informationen, erhält man die Testlücke (Testgap Treemap). Jede Methode ist entweder grau (unverändert – getestet oder ungetestet), rot (neu, aber nicht getestet), gelb (geändert, aber nicht getestet) oder grün (neu oder geändert und getestet).

Diese automatische Lückenanalyse zeigt bei kontinuierlichen Änderungen im System immer den aktuellen Stand und ob alle Änderungen getestet sind. In einem Prozess mit kurzen Iterationen und vielen Änderungen erhält man damit schnelles Feedback: Entscheider sehen, ob das nächste Release reif für den Kunden ist, und Tester, wo noch nachzutesten ist. Die Lückenanalyse sagt allerdings nur aus, ob eine Änderung bereits getestet wurde. Zum Testergebnis, also ob der Test erfolgreich war, gibt sie keine Auskunft. Teamscale kennt aber auch die Testergebnisse, weshalb sich damit auch zum Testerfolg Übersichten als Treemaps erstellen lassen.

Automatisierte Auswirkungsanalyse spart Zeit

Die Lückenanalyse ist eines der wesentlichen Features von Teamscale und auch Alleinstellungsmerkmal unter den Qualitätswerkzeugen. Dahinter verbirgt sich eine inkrementelle Analysemaschine, die kontinuierlich Änderungen im Code und Testvorgänge in ihrer zeitlichen Abfolge registriert und daraus die notwendigen Informationen aufbereitet.

In der Testtheorie spricht man von Auswirkungsanalyse: Bei Änderungen müssen Tester die direkten und indirekten Auswirkungen auf das Gesamtsystem verstehen, um den Umfang der notwendigen Tests zu bestimmen. Teamscale liefert diese Auswirkungsanalyse sofort und automatisch. Das spart Zeit und erhöht die Zuverlässigkeit, was in kontinuierlichen Entwicklungsprozessen förderlich ist.

Allerdings haben solche Methoden Grenzen: Weder eine manuelle noch eine automatisierte Auswirkungsanalyse kann zuverlässig alle Fehlerquellen identifizieren, die durch Änderungen entstehen. In diesem Sinne soll und darf man von der Lückenanalyse keine Vollständigkeit erwarten. Im Normalfall wird sie aber die wahrscheinlichsten Fehlerquellen aufdecken und damit das Risiko unentdeckter Fehler senken.

Bei der Auswirkungsanalyse bietet Teamscale eine Paretooptimierung: Es sind diejenigen Testfälle auswählbar, die bei möglichst kurzer Ausführungszeit die größte Erhöhung der Testüberdeckung erzielen. Testaktivitäten lassen sich daher effizient planen und durchführen. Insbesondere können Entwickler umfangreiche Regressionstests auf ein handhabbares Maß beschränken.

Zugriff auf Coderepositorys mit Adaptern

Für die unterschiedlichen Analysen braucht Teamscale Informationen zu Codeänderungen und zu den von Tests

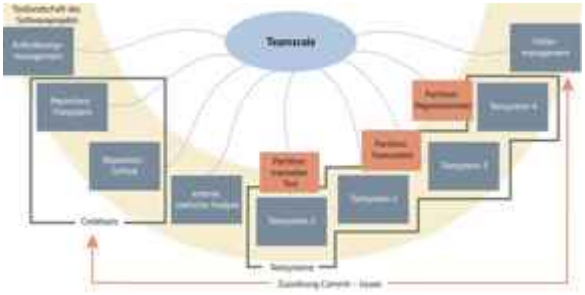
ausgeführten Codeteilen. Damit es Zugriff auf solche Informationen erhält, muss man es mit entsprechenden Quellen in der Entwicklungs- und Testumgebung verbinden. Für die Codeänderung ist ein Zugriff auf die Coderepositorys erforderlich. Teamscale stellt dafür viele Adapter für gängige Versionskontrollsysteme bereit. Nutzer können sich beispielsweise mit GitHub, Azure und SAP-Repository-Servern sowie mit dem eigenen Dateisystem verbinden. Teamscale erlaubt das gleichzeitige Arbeiten mit mehreren verschiedenen Repositorys.

Teamscale benötigt für die Testüberdeckung Überdeckungsinformation aus der Testausführung. Das geschieht entweder durch manuelles oder automatisiertes Hochladen der Information in einem gängigen Dateiformat (etwa JaCoCo, LLVM Coverage oder LCOV) oder automatisch über Coverage-Agenten. Teamscale beherrscht alle geläufigen Programmiersprachen und Dateiformate. Ordnet man die Information verschiedenen Partitionen zu, ist eine kleinteilige Analyse nach eigenen Vorstellungen möglich.

Die Testüberdeckung ist im automatisierten und im manuellen Test messbar. Wichtig ist nur, dass während der jeweiligen Testausführung ein Werkzeug die Testüberdeckung aufzeichnet und die Daten an Teamscale überträgt. Mittels Partitionen lassen sich manuelle und automatisierte Tests unterscheiden – auch weitere Differenzierungen bei den Testaktivitäten sind möglich. Ebenso können Anwender über die Partitionen die Tests auf verschiedenen Testservern zu einer gemeinsamen Messgruppe zusammenfassen. Die Systeme müssen dabei dedizierte Testsysteme sein, da alle Programmausführungen in die Messungen eingehen. Aber selbst hier kann man bei genügendem Verständnis der Teamscale-Schnittstelle individuelle Lösungen aufbauen.

Überwachung mit dem Datenkraken

Teamscale gestaltet sich als Datenkraken im positiven Sinn: Es sammelt die wichtigen Messdaten aus der jeweiligen Komponente der Toollandschaft (Abbildung 2). Verknüpft man die Daten für komplexe Analysen, erhält man aussagekräftige Statistiken.



Teamscale als Datenkraken sammelt Messdaten aus allen Komponenten und überwacht sie (Abb. 2).

Zusätzlich lässt sich Teamscale mit dem Fehlermanagement verbinden. Instrumentiert man das Fehlermanagement, sodass Teamscale den Zusammenhang zwischen Commits im Repository und Issues im Fehlermanagement versteht (zum Beispiel durch Eintrag der Issue ID in der Commit Message), sind Änderungen und Testüberdeckung auch auf Issue-Ebene nachverfolgbar. Anwenderinnen und Anwender erhalten damit eine wertvolle Unterstützung, detailliert den verschiedenen Aktivitäten der Weiterentwicklung und Fehlerkorrektur zu folgen und Lücken aufzuspüren. Zudem verbindet sich Teamscale mit dem Anforderungsmanagement.

Teamscale greift auf Informationen aus Entwicklung und Test zu und kann deshalb viele Messungen, Statistiken und Übersichten erstellen. Die für ein Projekt wichtigsten Messungen und Statistiken stellt es in einem Dashboard dar (Abbildung 3).

Das Dashboard fasst alle Ergebnisse übersichtlich mit Diagrammen und Treemaps zusammen (Abb. 3).

Qualität des Codes durch Analysen steuern

Bei DevOps und Continuous Integration wird nicht nur dynamisch getestet, sondern der Code schon vor dem Build der Software ausführlich statischen Analysen unterzogen. Teamscale beherrscht die statische Analyse, indem es Werkzeuge wie beispielsweise CLang-Tidy (C, C++, Objective-C), ESLint (JavaScript), FindBugs (Java), StyleCop (C#), Pylint (Python) oder SAP Code Inspector (ABAP) einbindet. Auch kennt es eine Reihe eingebauter statischer Analysen wie unbenutzten Code, Verschachtelungstiefe und Clone Detection.

Insbesondere Clone Detection ist eine wertvolle Hilfe, ähnliche Codeteile aufzuspüren, die meist auf Copy-Paste-Verstöße gegen Wiederverwendbarkeit (DRY – Don’t repeat yourself) hinweisen. Ebenso lassen sich eigene Regeln für die statische Analyse hinzufügen. Sowohl beim ersten Hinzufügen eines Repositorys zu den Teamscale-Schnittstellen als auch bei jeder Änderung im Repository durchläuft das Tool automatisch die statischen Analysen.

Teamscale liefert eine umfassende Übersicht über die Befunde der statischen Analyse, die man durch geeignete Diagramme im Dashboard darstellen kann. Entsprechend konfiguriert, sammelt Teamscale weitestgehend automatisch Daten über sämtliche Prüfaktivitäten im Softwarelebenszyklus und zeigt sie übersichtlich auf Dashboards an.

Damit offeriert es umfassendes Fast Feedback bei der iterativen Entwicklung. Die hohe Flexibilität von Teamscale scheint dabei dem Umfang und der Art der erfassten Messungen kaum Grenzen zu setzen. Insofern ist es als Single Point of Truth bei kontinuierlicher Entwicklung einsetzbar und liefert alle für das Bewerten der aktuellen Softwarequalität nötigen Berichte.

Architekturanalyse deckt Abhängigkeiten auf

Zusätzlich zu den Funktionen, die Codequalität und Testdurchführung steuern, bietet Teamscale noch ein weiteres ungewöhnliches Feature: Anwender können die Systemarchitektur spezifizieren, das heißt, das Gesamtsystem in seine Teilsysteme zerlegen und deren Beziehungen zueinander modellieren. Über einen mit dem Coderepository verbundenen UML-Editor sind die Teilsysteme in der Architekturmodellierung direkt aus dem existierenden Code auswählbar.

Softwarearchitekten können beim Architekturentwurf schrittweise vorgehen. Das ist besonders dann vorteilhaft, wenn noch keine vollständige Architektur des Systems besteht, sondern die Architektur aus historischen Gründen erst im Nachhinein durch Reverse Engineering des bestehenden Systems spezifiziert oder im Rahmen einer iterativen Entwicklung angepasst wird.

Die Zerlegung in Teilsysteme erlaubt, die Messungen auf die jeweiligen Teile zu beschränken und dadurch eine feingranulare Qualitätsanalyse zu erhalten, die nur die Teile betrachtet, die von Belang sind. Teamscale stellt durch eine statische Architekturanalyse mit dem Abgleich gegen das Coderepository auch Verstöße gegen die Architektur bei Systemänderungen fest – etwa Abhängigkeiten zwischen Teilsystemen, die im Architekturentwurf nicht vorgesehen sind.

Bewertungsgrundlage: Cloud-Version, Linux und Docker

Für den Artikel kam die Cloud-Testversion von Teamscale zum Einsatz – den Zugriff erhält man durch eine einfache Anfrage über die Webseite. Teamscale ist als native Installation für Windows, Linux, macOS und auch als Docker-Image verfügbar. Da es eine webbasierte Client-Server-Lösung ist, kann man Teamscale ebenfalls als Cloud-Lösung erhalten.

Die Tests sind mit der Cloud-Version durchgeführt. Hierzu wurden zwei Standardbeispiele von CQSE verwendet, aber auch eigene eingebunden. Um die Installation auszuprobieren, haben wir Teamscale nativ unter Linux und mit Docker unter Linux und Windows installiert.

Teamscale bietet viele Features für die Qualitätssteuerung. Um sie zu nutzen, muss man es in viele Bereiche der Softwareentwicklung einbinden. Durch die vielen möglichen Diagnosen, notwendigen Anbindungen an die Entwicklungs- und Testlandschaft und eingebundenen externen Werkzeuge (statische Analyse, Versionskontrolle, Test) ist die Konfiguration von Teamscale eine komplexe Aufgabe, die Anwender nicht unterschätzen sollten.

Auch wenn sich die Testversion in Eigenregie installieren lässt, sollten Anwender auf den angebotenen CQSE-Support zurückgreifen – nur dann können sie das volle Potenzial von Teamscale in einem Pilotversuch nutzen.

Bedienung überwiegend intuitiv

Die User Experience ist gut: Hat man sich erst einmal an die Teamscale-Logik gewöhnt, findet man wertvolle Informationen für Entwicklung und Test – aber es dauert mitunter, die Logik hinter den Funktionen zu erfassen. Im Vergleich mit ähnlich komplexen Werkzeugen ist die Bedienung dennoch einfach und (meistens) intuitiv.

Der Wert von Teamscale für den einzelnen Kunden hängt davon ab, wie gut es in die eigene Entwicklungs- und Testlandschaft eingebunden ist. Je besser die eigenen Prozesse und die eigene Werkzeuglandschaft strukturiert sind, desto einfacher ist die Integration.

Ein wichtiger Punkt ist, ob Teamscale die vom Nutzer verwendeten Techniken berücksichtigt. Es kommt schon jetzt mit einer umfangreichen Menge daher: 29 Programmiersprachen, 18 externe Werkzeuge für die statische Analyse, alle gängigen Versionskontrollsysteme; zudem noch die Integration in IDEs und die Implementierung von Fehlermanagementsystemen (Abbildung 4).

Teamscale beherrscht viele Programmiersprachen und bindet achtzehn Werkzeuge für die statische Analyse ein (Abb. 4). Auch auf den Webseiten und im Gespräch mit dem Support weist CQSE darauf hin, dass es neue Sprachen und Werkzeuge gerne integriert. Nicht die Integration zusätzlicher Technologie ist das Problem, sondern dass Anwender den Überblick darüber bekommen.

Welche Voraussetzungen erforderlich sind

Um Teamscale einzusetzen, braucht es ein Softwareprojekt mit Zugriff auf den Quellcode. Zudem muss Teamscale die verwendete Programmiersprache und die eingesetzten Tools berücksichtigen.

Auf Kundenwunsch passt CQSE es an weitere Techniken an. Auch eine IDE-Integration zum Anzeigen der erhobenen Informationen ist möglich. Arbeiten im Projekt mehrere Teammitglieder zusammen, ist es wichtig, ein gemeinsames Verständnis von Qualität zu schaffen. Dazu zählt es, Coding Guidelines für die Codestruktur (Codezeilen pro Datei, Methodenlänge) festzulegen, Redundanzen zu vermeiden, Code zu formatieren, zu dokumentieren und Tests zu schreiben. Außerdem betrifft es den Prozess der Softwareentwicklung: ein Versionsverwaltungssystem mit einem Branching-Konzept verwenden oder eine CI-Pipeline einsetzen, um Schritte im Entwicklungsprozess und die Ausführung von Tests zu automatisieren.

Teamscale ist auch ohne ein Versionsverwaltungssystem verwendbar. Dazu laden Anwender in bestimmten Abständen den aktuellen Stand des Codes hoch, Teamscale bildet dann das Delta zur vorherigen Version.

Die Installation von Teamscale allein verbessert allerdings noch nicht die Qualität des Quellcodes oder verringert die Anzahl der Fehler. Vielmehr gilt es, die gesammelten Informationen auch zu nutzen. Diese Aufgabe sollten Quality Engineers übernehmen. Sie überwachen, dass die Entwicklungsteams die definierten Guidelines und Prozesse einhalten, kümmern sich um die Konfiguration und Lauffähigkeit von Teamscale und schauen, dass das Architekturdiagramm auf dem neuesten Stand ist. Außerdem liefern sie Informationen zum Qualitätstrend und besonderen Findings und stellen sicher, dass das Team offene Punkte im Backlog anlegt.

Für den Start empfiehlt es sich, kleine und erreichbare Ziele zu definieren. Vermutlich führt Teamscale anfangs viele Findings auf. Hier darf man nicht den Überblick verlieren, sondern muss sich im ersten Schritt auf ein, zwei Ziele konzentrieren. Das kann beispielsweise sein, neu auftretende Probleme im gerade angepassten Code zu beseitigen.

Setzen Anwender ein Versionsverwaltungssystem ein, können Pull

Requests sinnvoll sein. Bevor ein Merge mit dem Hauptzweig stattfinden kann, sind bestimmte Quality Gates zu erreichen: zum einen, dass Teamscale keine neuen Findings listet (der Code muss den geltenden Qualitätsstandards entsprechen), und zum anderen, dass mindestens ein anderer Entwickler den Review durchführt.

Wann ist es sinnvoll, Teamscale einzusetzen?

Mehrwert kann Teamscale in jedem Softwareprojekt bieten, auch wenn es von nur einem Entwickler betreut wird, da die Analysen eine große Arbeitserleichterung sind und helfen, die Softwarequalität zu steigern. Je größer das Entwicklerteam ist, desto wichtiger ist es sicherzustellen, dass jeder die gleichen Richtlinien einhält und somit eine einheitliche Basis gleicher Qualität geschaffen wird. Teamscale fungiert hier als zentrale Anlaufstelle, um einen schnellen Überblick über die derzeitige Codequalität zu bekommen. Entwickler erhalten dadurch direkt Feedback zu ihrem neu eingecheckten Code und darüber, an welcher Stelle sie gegebenenfalls nachbessern müssen.

Darüber hinaus kann es sinnvoll sein, Teamscale einzusetzen, wenn man Software mit einer großen Codebasis neu entwickeln oder die Struktur von bestehendem Code verbessern will. Vor allem, wenn Dritte den Code geschrieben haben, hilft es schnell dabei, Probleme und Schwachstellen im Code aufzudecken: Duplikate, Verstöße gegen die Coding Conventions, zu lange Methoden und Verletzungen der Softwarearchitektur fallen hierunter. Auch herauszufinden, welche Teile der Code beim Nutzen einer Funktion durchläuft, ist beim Refactoring behilflich und beschleunigt den Prozess.

Nützlich ist dabei die Code-Coverage-Analyse, mit der sich die Codeausführung im produktiven Betrieb auswerten lässt. Entwickler können ermitteln, welche Komponenten die Anwender

des Programms wie oft benutzen, um ihnen eventuell eine höhere Bearbeitungspriorität zuzuweisen. Ein Fehlverhalten einer dieser Komponenten würde schnell mehrere Nutzer der Software betreffen, was ein hohes Risiko ist. Zudem kann eine solche Analyse aufzeigen, welche Teile des Codes die Anwender nicht mehr benötigen. Bei einem Refactoring können Entwickler diese Teile des Codes löschen, was in Folge die Codequalität steigert.

Fazit

Teamscale senkt durch sein Fast Feedback die Entwicklungs- und Testlaufzeiten. Es fördert kleinteiliges Arbeiten, was das Aufspüren und Lokalisieren von Fehlern erleichtert. Außerdem erlaubt es Anwendern schnell zu arbeiten, weil es umfangreiche Prüfungen der erreichten Überdeckung automatisiert liefert. Die Paretooptimierung bringt weitere Informationen, um Testlaufzeiten zu verbessern: Wichtige Testfälle sind damit schnell und effizient auswählbar.

Als komplexes Werkzeug besitzt Teamscale Features, die ähnlichen Werkzeugen fehlen. Das hat seinen Preis: Derzeit kostet es 550 Euro im Monat für bis zu fünf Benutzer. Dafür erhält man auch Support. Wie bei solchen Werkzeugen üblich, kann man es mit einer Testversion ausprobieren, für die es ebenfalls Support gibt. Für Lehreinrichtungen und Open-Source-Projekte ist der Einsatz von Teamscale kostenlos.

Teamscale ist ein Tool, das Testspezialisten kennen sollten. Es empfiehlt sich gerade für größere Teams, die mit kurzen Releasezyklen arbeiten. Prinzipiell sind die erstellten Qualitätsmessungen und Berichte für jede Art von Entwicklungs- und Testteam von großem Nutzen. (nb@ix.de)

Wertung

schnelles Feedback bei iterativer Entwicklung und häufigen Änderungen Dashboard an eigene Bedürfnisse anpassbar, bietet

schnelle Übersicht über den aktuellen Qualitätsstand sehr großer Funktionsumfang, der stetig erweitert wird Unterstützung vieler Programmiersprachen und Technologien sehr hohe Konfigurierbarkeit, lässt sich an die eigene Systemlandschaft anpassen leicht erweiterbar durch eine gut dokumentierte REST-API

- sehr guter Support durch den Anbieter

UI nicht immer selbsterklärend, könnte an manchen Stellen benutzerfreundlicher sein anfangs hohe Einstiegshürde, bis alle Systeme korrekt angebunden sind noch kein Visual-Studio-Code-Plug-in (ist geplant), Funktion ist aber über das CLI nutzbar

- preislich im Mittelfeld, für nicht umsatzstarke Softwareprojekte vermutlich zu teuer

Daten und Preise

Hersteller: CQSE

Produktname: Teamscale

Preise: 550 Euro/Monat (fünf Benutzer); kostenlos für Open-Source-Projekte und Bildungseinrichtungen

Download: cqse.eu/de/teamscale/download/

1. Quellen
2. Dokumentation, Download, Videos: ix.de/zm29

Dr. Carsten Weise

arbeitet als Trainer für ISTQB-Zertifizierungskurse und Testautomatisierung bei der imbus AG.

Christoph Singer

ist Berater mit Schwerpunkt Testautomatisierung bei der imbus AG.