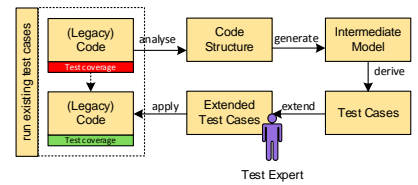


Semi-automatische Generierung von Testfällen für Legacy Code



Ein generischer Ansatz ermöglicht die semi-automatische Generierung von Test Code basierend auf existierendem Legacy Code. Durch die Analyse von Source Code wird ein abstraktes Zwischenmodell abgeleitet, das es erlaubt, mittels Äquivalenzklassen geeignete Testfälle zu erzeugen und für die automatische Testfallausführung bereitzustellen.

grundlegenden Testfälle können automatisch ausgeführt werden. Abb. 1 zeigt diesen grundlegenden Ablauf.

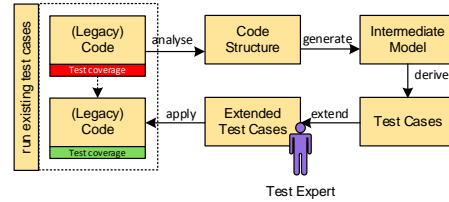


Abb 1. Anwendungsszenario.

Aus dem generischen Ansatz für die semi-automatische Testfall-Generierung ergeben sich folgende konkrete Anwendungsszenarien (S):

- S1: Erstellung von fehlende Tests und damit Erhöhung der Testabdeckung von Legacy-Code.
- S2: Unterstützung bei der Systemmigration durch Testfallwiederverwendung bzw. der abstrakten Baumstruktur des Codes.

Aufgabe

Legacy Systeme aber auch aktuell erstellter Source Code beinhalten oft Produktivcode ohne entsprechende Software-Tests. Diese Tests können oft nur aufwändig ergänzt werden. Durch einen generischen Ansatz für die semi-automatische Testfallgenerierung werden Entwickler dabei unterstützt, fehlende Testfälle effizient und systematisch zu erstellen. Diese Testfälle können auch bei Migrationsprojekten eingesetzt werden, um die Produktqualität des migrierten Produktes effizient überprüfen zu können. Ein sprachunabhängiges Codemodell unterstützt verschiedene ausgewählte Programmiersprachen.

Umsetzung

Der Prototyp für die Testfall-Generierung, der von der Software Quality Lab GmbH und der TU Wien entwickelt wurde, zeigt anhand eines einfachen Beispiels, wie bestehender Software Code analysiert und eine abstrakte Baumstruktur des Software-Codes abgeleitet werden kann. Ein Äquivalenzklasseneditor ermöglicht es, basierend auf der abstrakten Baumstruktur Testfälle mit Äquivalenzklassen zu erstellen. Diese

Anwendungsfall

Der Forschungsprototyp bildet einen Testautomatisierungsprozess ab (siehe Abb. 2). Dieser Prozess besteht aus sechs wesentlichen Schritten.

1. **Parse Code:** bestehender Code, z.B. in C, wird analysiert und als abstrakte Baumstruktur persistiert.
2. **Code Classification:** Die Komplexität des Codes (z.B. einfache Bedingungen, Schleifen, aber auch Abhängigkeiten oder globale Variablen), erfordert unterschiedliches Vorgehen.
3. **Test Approach:** Als Testansatz werden derzeit Äquivalenzklassen mit geeignetem Editor unterstützt.
4. **Test Case Generation:** Die Generierung der Testfälle umfasst (a) die Erstellung von Test-Stubs und (b) die Generierung von Test Code.
5. **Derive Test Code:** Der Test Code wird im aktuellen Anwendungsfall automatisch generiert, kann aber bei Bedarf manuell ergänzt werden.
6. **Run & Analyze Test Code:** Die Ausführung der Testfälle ist im Rahmen der Eclipse Entwicklungsumgebung realisiert.

Technische Daten

- Prozess für die semi-automatische Ableitung von Testfällen basierend auf Codeanalyse.
- Einbettung von Best-Practices aus der Softwareentwicklung.
- Flexibilität und Erweiterbarkeit durch modulare und Komponenten basierte Architektur.
- Integrierte Werkzeugintegration anhand realer Kundenbedarfe.

Wesentliche Merkmale

- Offenheit für unterschiedliche Testentwürfsverfahren.
- Bessere Testfälle durch semi-automatisierte Generierung, ergänzt durch manuelle erstellt Testfälle für das effiziente Finden von Fehlern
- Offenheit gegenüber unterschiedlichen Unit-Test-Frameworks.

Ihr Nutzen

- Unterstützung bei der systematischen Erstellung von (fehlenden) Software-Tests.
- Qualitätssicherung bei Migrationsprojekten durch wiederverwendbare Testfälle.
- Verwendung von Best-Practice Ansätzen.
- Sprachunabhängige Modellierung von Software Code.

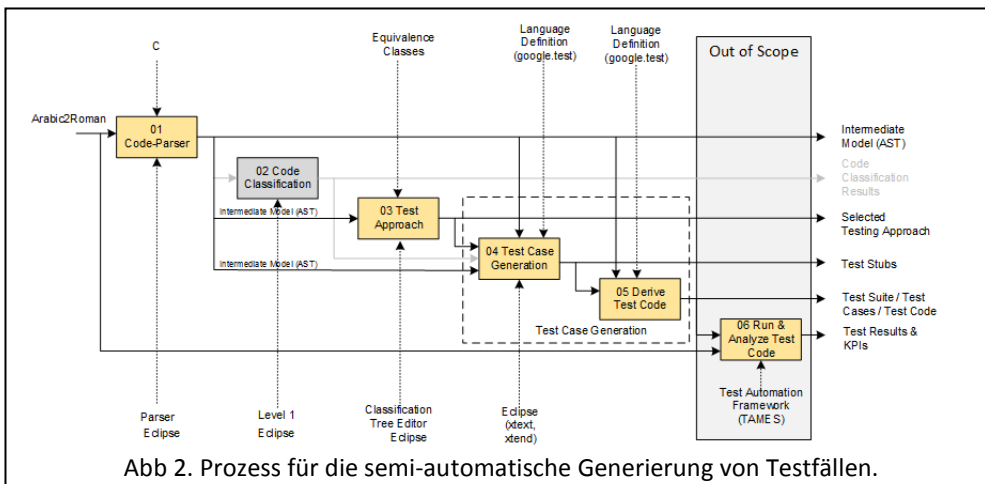


Abb 2. Prozess für die semi-automatische Generierung von Testfällen.



Kontakt:

Prof. Dr. Stefan Biffli
TU Wien
stefan.biffli@tuwien.ac.at
qse.ifs.tuwien.ac.at

Johannes Bergmann
Software Quality Lab GmbH
johannes.bergmann@software-quality-lab.com
www.software-quality-lab.com