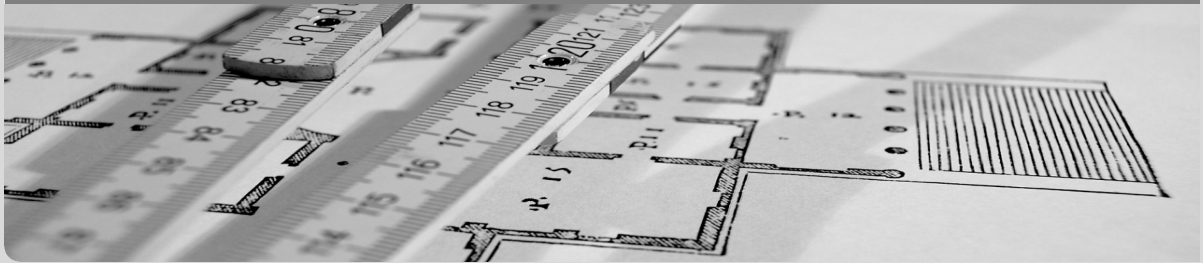
Karlsruher Institut für Technologie

Extrahieren von Code-Änderungen aus einem Commit für kontinuierliche Integration von Leistungsmodellen

Codereview für eine Bachelorarbeit

Ilia Chupakhin, Betreuerin Manar Mazkatli | 24.07.2020

INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION



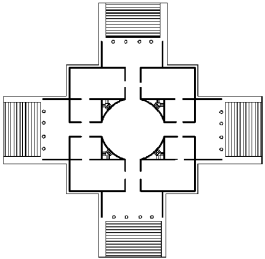
Motivation

Quellcode

```java
public class Human {
    String name;
    int age;
    int money = 0;
    boolean isHungry = false;
    boolean isTired = false;

    public Human(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void eat() {
        money--;
        isHungry = false;
    }

    public void sleep() {
        isTired = false;
    }

    public void work() {
        money++;
        isTired = true;
    }

}
```

Konsistenzhaltung

Performance-Modell



Motivation: Continuous Integration of Performance Models (CIPM)

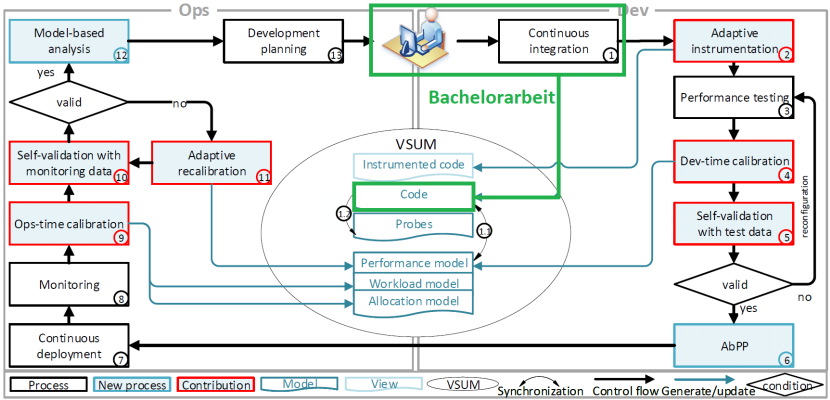
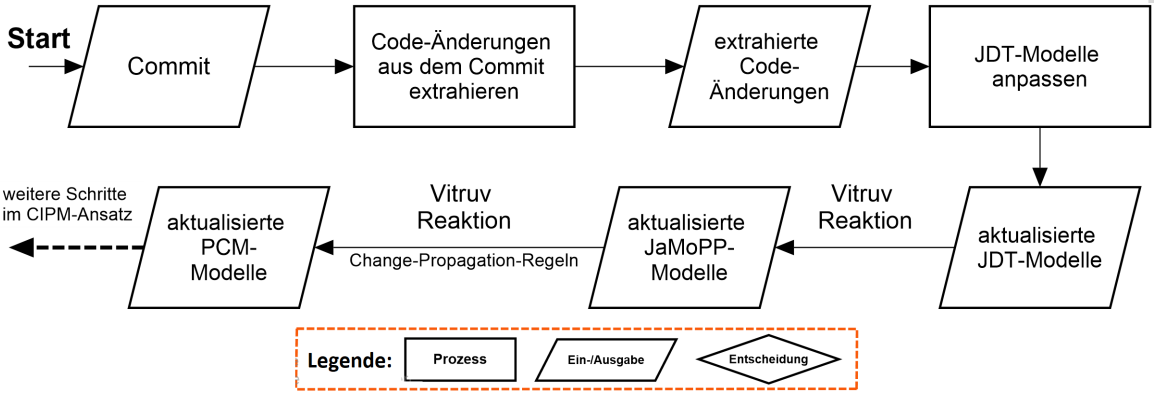
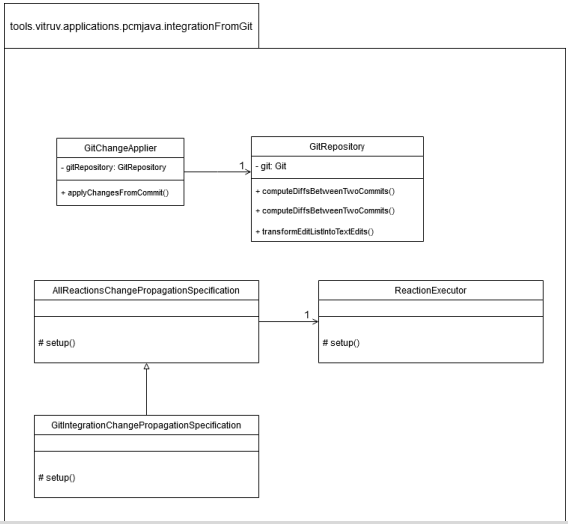


Abbildung: Modellbasierte DevOps-Pipeline mit dem CIPM-Ansatz, Quelle: Mazkatli u. a. [1]

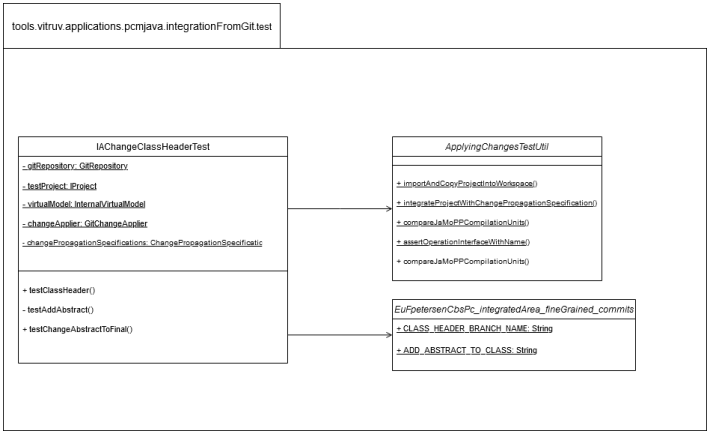
Lösungsansatz



Implementierung: Haupt-Plugin



Implementierung: Test-Plugin



Beispieltest: Hinzufügen einer Klassenvariablen

```java
private void testAddField() throws NoHeadException, GitAPIException, IOException, CoreException, InterruptedException {
    //Apply changes
    changeApplier.applyChangesFromCommit(commits.get(EuFpetersenCbsPc_integratedArea_fineGrained_commits.ADD_IMPORT_FOR_FILED),
            commits.get(EuFpetersenCbsPc_integratedArea_fineGrained_commits.ADD_FIELD), testProject);
    //Checkout the repository on the certain commit
    gitRepository.checkoutFromCommitId(EuFpetersenCbsPc_integratedArea_fineGrained_commits.ADD_FIELD);
    //Create temporary model from project from git repository. It does NOT add the created project to the workspace.
    projectFromGitRepository = ApplyingChangesTestUtil.createIProject(workspace, workspace.getRoot().getLocation().toString()
            + "/clonedGitRepositories/" + testProjectName + ".withGit");
    //Get the changed compilation unit and the compilation unit from git repository to compare
    ICompilationUnit compUnitFromGit = CompilationUnitManipulatorHelper.findICompilationUnitWithClassName("GraphicsCard.java", projectFromGitRepository);
    ICompilationUnit compUnitChanged = CompilationUnitManipulatorHelper.findICompilationUnitWithClassName("GraphicsCard.java", testProject);
    //Compare JaMoPP-Models
    boolean jamoppClassifiersAreEqual = ApplyingChangesTestUtil.compareJaMoPPCompilationUnits(compUnitChanged, compUnitFromGit, virtualModel);
    //Ensure that there is a corresponding PCM model to the field.
    boolean pcmExists = ApplyingChangesTestUtil.assertFieldWithName("field", compUnitChanged, virtualModel);

    assertTrue("In testAddField() the JaMoPP-models are NOT equal, but they should be", jamoppClassifiersAreEqual);
    assertTrue("In testAddField() corresponding PCM model does not exist, but it should exist", pcmExists);
}
```

Codereview


Manar Mazkatli u. a. “Incremental Calibration of Architectural Performance Models with Parametric Dependencies”. In: (2020).