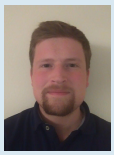
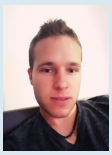
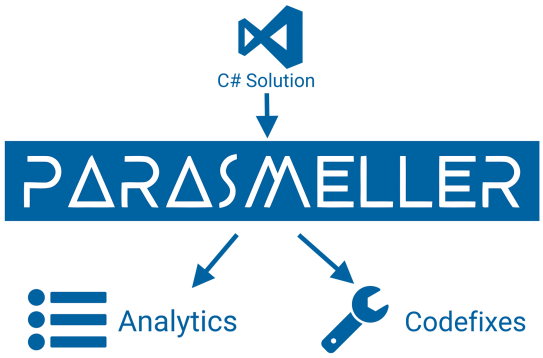

Rafael Krucker


Markus Schaden

Studenten/-innen	Rafael Krucker, Markus Schaden
Dozenten/-innen	Prof. Dr. Luc Bläser
Co-Betreuer/-innen	
Themengebiet	Software

Study on ConcurrencyChecking for .NET



Der ParaSmeller untersucht C# Solutions, zeigt Warnungen an und bietet teilweise die Möglichkeit, diese zu korrigieren

```
class BankAccount
{
    private int balance;
    1 reference
    public void Deposit(int amount)
    {
        lock (this) { balance += amount; }
    }
    0 references
    public void Transfer(BankAccount target, int amount)
    {
        lock (this)
        {
            balance -= amount;
            target.Deposit(amount);
        }
    }
}
```

void BankAccount.Deposit(int amount)

Possible deadlock with double locking

Beispielsweise werden mögliche Deadlocks entdeckt, falls Double Locking verwendet wird.

```
public T Get()
{
    lock (this)
    {
        if (queue.Count == 0)
        {
            Monitor.Wait(this);
        }
        if should be replaced with while
        Show potential fixes (Ctrl+.)
    }
}
```

Andererseits wird gewarnt, falls der Monitor Lock falsch verwendet wird.

Ausgangslage: Diese Arbeit soll eruieren, inwiefern das Erkennen von Code Smells im Bereich Parallele Programmierung das Schreiben von Software weniger fehleranfällig machen kann. Als Grundlage wurde dafür ein Paper verwendet, dass 10 solche Code Smells identifiziert und beschreibt. Das Ziel war das Schreiben einer Library, welche für die statische Code Analyse in C# zuständig ist. Als Grundlage dafür wird der Roslyn Compiler von Microsoft verwendet, welcher eine mächtige API für C# Code-Analysen bereitstellt. Dabei sollte die Library möglichst frei anwendbar sein. So wurden Integrationen für Microsofts Visual Studio sowie SonarQube erstellt, so dass die Analyse von der Kommandozeile bis hin zum Build Sever einsetzbar ist.

Vorgehen/Technologien: Hauptaugenmerk wurde dabei auf Behandlung in die Breite gelegt. In einer Analyse von bekannten Open Source Projekten sowie einigen unbekannten, eher unvollständigen Programmen wurde dann untersucht, ob diese Code Smells in der Praxis vorkommen und auch zuverlässig erkannt werden können. Dabei zeigen sich grosse Unterschiede bei den Häufigkeiten des Auftretens der einzelnen Smells. Es treten bei den verschiedenen Projekten oft die gleichen Muster im Code auf, so dass durch entsprechende Warnungen Programmierer früh auf solche Probleme sensibilisiert werden können. Vergleicht man die Häufigkeit mit den Auswirkungen der Smells auf die Qualität, offenbart sich, welche Smells besonders verfolgt werden sollen.

Ergebnis: Die Untersuchung hat gezeigt, dass Smells im Bereich der parallelen Programmierung erkannt werden können und in der Realität auch auftreten. Dadurch kann gesagt werden, dass der Einsatz von Tools, die Smells erkennen, das Programmieren nebenläufiger Anwendungen sicherer und angenehmer macht. Die Untersuchung von realen Projekten hat weiterhin gezeigt, dass nicht alle Smells gleich häufig auftreten. Daher wurde eine Empfehlung gemacht, auf welche Smells sich die Tools konzentrieren sollten.