Fortgeschrittene Softwaretechnik
Wintersemester 2024/25
Name: Simon Szulik
Matr.Nr: 1474315

Datum: 20. November 2024
Reading Group Sessions
Inhaltsskizze zu Block: 03

Titel des Papiers

Harmful Effect of God Class Refactoring on Power Consumption

Kurzzusammenfassung

- **Hintergrund und Motivation**
  - Das Paper untersucht die Auswirkungen von Refactoring-Techniken, insbesondere die Transformation von God Classes, auf den Stromverbrauch von Software. God Classes, also Klassen mit übermäßig vielen Funktionen, werden oft refactored, um die Wartbarkeit zu verbessern, da solche Klassen komplex und schwer zu testen oder zu warten sind. Dies kann dazu führen, dass der Nachrichtenaustausch zwischen Objekten und damit der Stromverbrauch deutlich steigt. Dies wirft grundlegende Fragen darüber auf, wie Nachhaltigkeit in die Softwareentwicklung integriert werden kann.
- **Forschungsfragen**
  - Erhöht das Anwenden von traditionallen Refactoringtechniken (God Classes auslagern), welche die Architektur verbessern, den Stromverbrauch von Software?
- **Vorgehen**
  - Zwei Open-Source-Systeme, Informa und NekoHTML, wurden analysiert. Die Refactoring-Transformationen wurden mithilfe des Tools JDeodorant (Eclipse Plugin) durchgeführt, das God Classes identifiziert und Vorschläge für ihre Refaktorisierung liefert. Nach der Anwendung der Refactoring-Techniken wurden die resultierenden Architekturen unter identischen Bedingungen getestet, um die Auswirkungen auf die Anzahl der Nachrichten und den Stromverbrauch zu messen.
  - Der Stromverbrauch wurde mit einem Volcraft Energy Logger gemessen, der den Gesamtverbrauch des Computers während der Ausführung protokolliert. Gleichzeitig wurden auch die Prozessorauslastung und die Anzahl der Nachrichten protokolliert, um die Ergebnisse besser zu verstehen.
- **Ergebnisse**
  - Der Nachrichtenaustausch stieg in den refactorten Systemen erheblich an: Bei Informa war die Anzahl der Nachrichten 14-mal höher, während sie bei NekoHTML 4-mal höher war. Diese Steigerung resultierte aus den zusätzlichen Objekten, die durch die Verfeinerung der ursprünglichen God Classes entstanden sind. Obwohl diese Änderungen die Wartbarkeit der Systeme verbesserten, führte dies zu einem Anstieg des Stromverbrauchs von 7,56% (Informa) und 20,1% (NekoHTML)(Auch aufgrund des längeren Programmlaufs)
  - Die Wartbarkeit verbesserte sich messbar durch reduzierte Komplexität und erhöhte Kohäsion der Klassen. Allerdings verschlechterte sich die Kopplung der Klassen, was zeigt, dass Verbesserungen in der Wartbarkeit oft mit Kosten in der Nachhaltigkeit verbunden sind. Dies deutet darauf hin, dass alternative Ansätze für nachhaltiges Refactoring notwendig sind.

Eigene (offene) Diskussionspunkte

- **Trade-offs zwischen Wartbarkeit und Nachhaltigkeit**
  - Wie kann man eine Balance zwischen besserer Wartbarkeit und geringerem Stromverbrauch finden? Sollten alternative Refactoring-Strategien entwickelt werden, die sowohl die Kohäsion verbessern als auch den Stromverbrauch reduzieren?
- **Limitierungen des Papers**
  - Die Studie wurde nur mit zwei Systemen durchgeführt. Die Generalisierbarkeit der Ergebnisse ist dadurch eingeschränkt. Es wäre interessant, ähnliche Experimente mit größeren und vielfältigeren Systemen durchzuführen, um repräsentativere Ergebnisse zu erzielen.
  - Die Messung des Stromverbrauchs umfasste den gesamten Computer, einschließlich Betriebssystem und Hintergrundanwendungen, was die Genauigkeit der Ergebnisse beeinflussen könnte. Eine präzisere Methode zur Messung des spezifischen Stromverbrauchs der getesteten Software wäre wünschenswert.