# Entwicklungsziele

(Erstellt: 30.03. Letztes update: 11.04.)

## Implementierung:

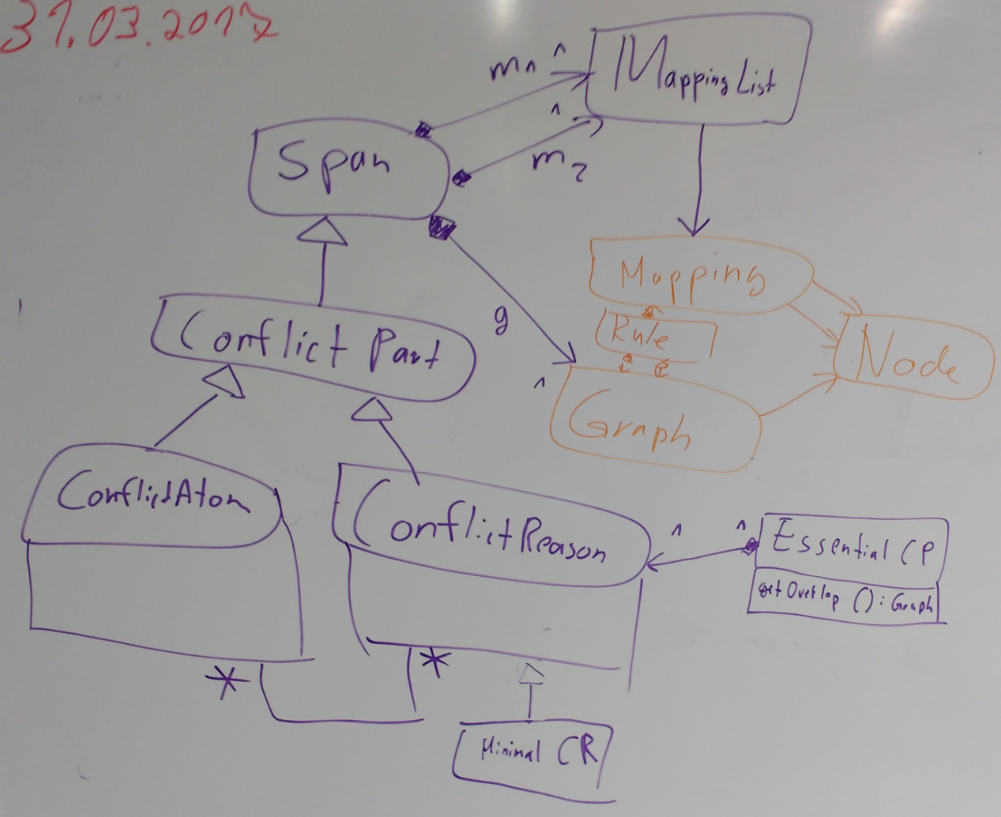
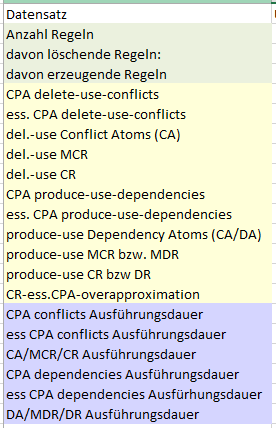
* Wrapper für produce-use-dependency
* Programmatisches erstellen der „use“-Regeln aus den „delete“-Regeln.
  + Erfolgreich umgesetzt!
    - 🡪 Messungen zu „overapproximation“ ausstehend.
      * Detailierte Erfassung zu jedem Regelpaar notwendig.
        + 🡪 vorerst nur esse

## Architektur überarbeiten

- Einsatz von „TreeSet“ anstelle von „HashSet“  
 Ursache für Missverständnis identifiziert. Fehlerhafte Annahme zur Auswirkung von identischen .hashCode() Ergebnissen. .equals() ist die kritische Methode! Keine spezifische Implementierung beider *Collections*-Methoden in HJenshin.Graph.

- Klassendiagramm noch relevant? (In der TelKo mal angesprochen)

An Whiteboard umgesetzt!



## Messung relevanter(interessanter) Daten

Siehe Excel sheet

* Regelanzahl
* Anzahl der
  + CPA Konflikte
  + Ess. CPA Konflikte
  + Confl. Atom
  + MCR
  + CR

## Weitere TODOs:

* Java Methode in Algorithmus Syntax
* Namen im Algorithmus kürzen

## Benutzerstudie:

* Betrachtung der „high-performer“ verstärkt unsere Aussage
  + Betrachtung der „low-performer“ ist in der Darstellung nicht so schön.
    - Ergebnis: Dennoch diese beiden Abbildungen nutzen!
* 🡪 TODO KB: gleiche 50/50 Aufteilung der Teilnehmer bei den Resultaten vornehmen!
* Teilnehmer näher beschreiben. Hochschulabschluss.

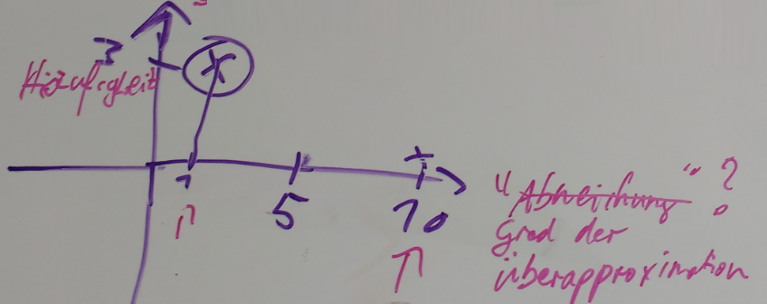
## Initiales Beispiel im Papier

* TODO KB: nachbauen + prüfen
  + Einfluss der Typisierung?

## Research Questions

* RQ1 – RQ5 für Chapter V (Experimental Evaluation) und Chapter VI (User Study)
  + DONE

## Overapproximation

* Geeignete Darstellung wählen!
  + 
* Keine Summen der Abweichung über den gesamten Regelsatz, da wenig Aussagekraft!