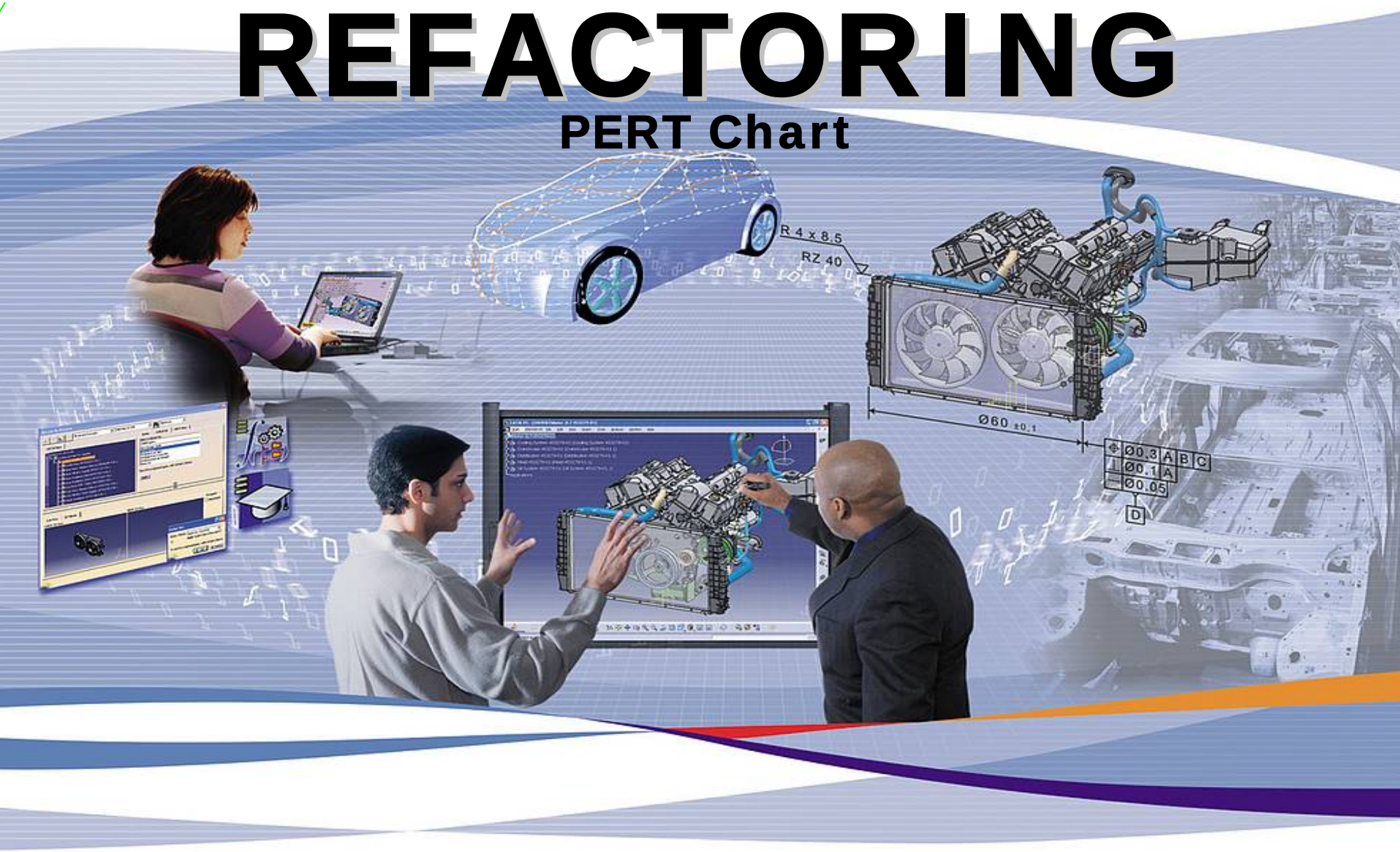
REFACTORING

PERT Chart



Agenda

- Konzeption und Prinzipien des Refactorings
- Prinzip und Aufgaben eines PERT Charts
- Tests aufbauen
- Katalog wichtiger Refactoring-Methoden
- Ausblick

Refactoring-Konzept

- **Refactoring**: eine Änderung an der internen Struktur einer Software, um sie leichter verständlich zu machen und einfacher zu verändern, ohne ihr beobachtetes Verhalten zu ändern
- diszipliniertes Vorgehen, welches die Wahrscheinlichkeit minimiert, Fehler einzuführen.
- methodische Verbesserung der Struktur
- Verbesserung des Designs, **nachdem** Quellcode geschrieben wurde.

- **Verbesserung des Designs, nachdem Quellcode geschrieben wurde**
  - grundsätzlich: zuerst entwerfen, dann programmieren
  - Im Laufe der Zeit wird jedoch der Code verändert und somit schwindet die Integrität des Systems.
    (Das Design - die entwurfsgemäße Struktur zerfällt)
  - die Struktur des Programms erlaubt es nicht mehr, dieses auf einfache Art und Weise zu ändern

→ **Refactoring**

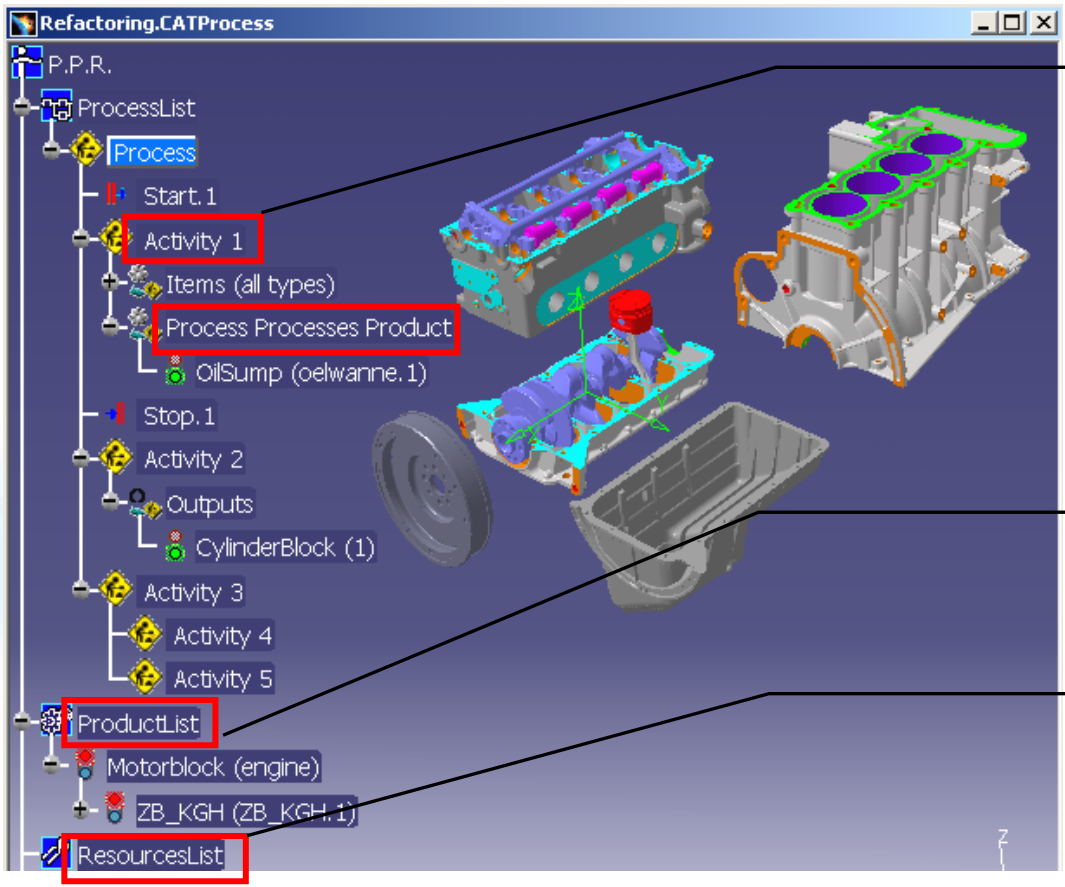
Gründe für Refactoring

- Refactoring macht Software leichter verständlich
- Refactoring hilft, Fehler zu finden und Redundanz aufzulösen
- Refactoring hilft Ihnen schneller zu programmieren und Code schneller zu entwickeln.

Probleme beim Refactoring

- es gibt noch keine hinreichenden Erfahrungen, um zu sehen wo die Grenzen des Refactorings liegen
- Datenbanken
- Veränderung von Schnittstellen
  - z.B. beim Ändern des Namens einer Methode muss Zugriff auf den ganzen Quellcode bestehen

Prozessplanung



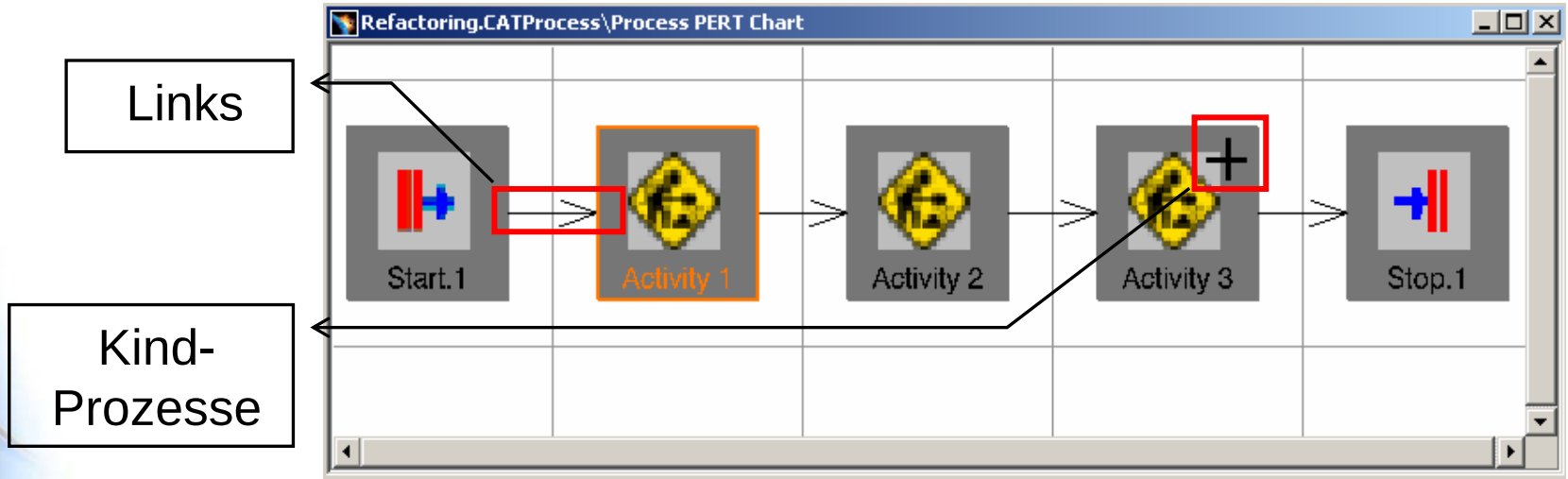
Prozesse

Produkte

Ressourcen

Prinzip und Aufgaben eines PERT Charts

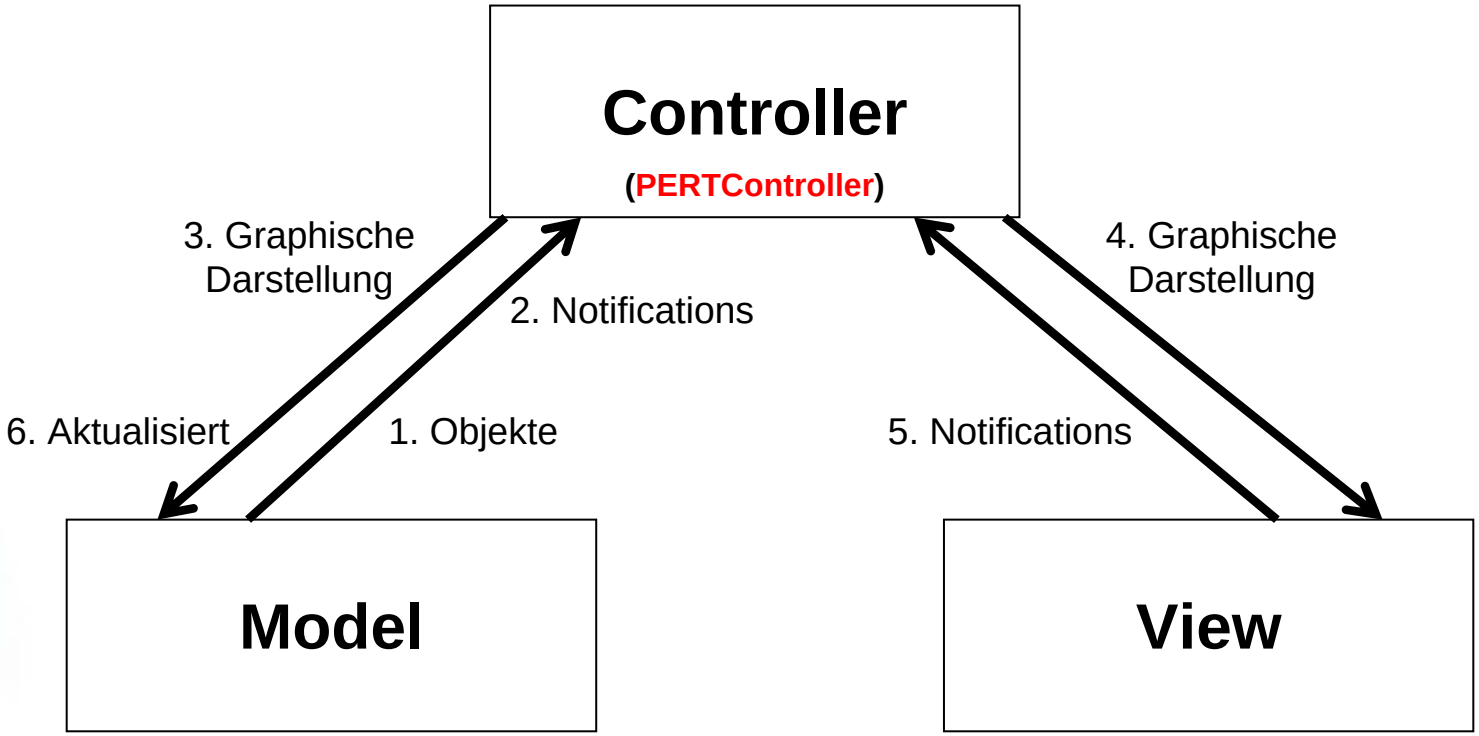
PERT Chart



- PERT Chart erlaubt eine eigene graphische Sicht auf Prozesse, um diese zu planen zu organisieren oder zu koordinieren

Prinzip und Aufgaben eines PERT Charts

Model View Controller in Pert



Schwierigkeiten

- Das PERT Chart
  - Framework, besteht aus einzelnen Modulen.
  - Viele Abhängigkeiten von Codezeilen weshalb es sehr mühsam ist, das Framework zu modifizieren.
  - äußerst unübersichtlich und somit nur sehr schwer zu verstehen.
  - PertController:
    - 4150 Zeilen Code
    - 61 Methoden

Erster Refactoring-Schritt

- eine solide Menge von Testfällen erstellen.
- vollständig automatisiert und selbstüberprüfend
- Macros oder JUnit-Test-Framework
- Funktionale Tests
- Komponententests

→ reduziert die Zeit für die Fehlersuche

Katalog wichtiger Refactoring-Methoden

- **Refactoring-Methoden**
  - Zustand und Verhalten zwischen Klassen verschieben

I. Methoden zusammenstellen

- Struktur verbessern
- Schlüsselrefaktorierung: **Methode extrahieren**
  - aus einem Code-Block eine eigene Methode erzeugen
  - Kommentare loswerden
  - Duplizierten Quellcode beseitigen

Refactoring-Methoden

II. Eigenschaften zwischen Objekten verschieben

- Verantwortlichkeiten verteilen, Verhalten verschieben
- **Methode verschieben**
  - abhängig davon, mit welchem Objekt die Methode am meisten interagiert
- **Klasse extrahieren**
  - Klasse macht die Arbeit von zwei Klassen
  - neue Klasse erstellen und relevante Felder und Methoden verschieben

Refactoring-Methoden

III. Daten organisieren

- Umgang mit Daten vereinfachen
- **Objekte/Felder kapseln**
  - öffentliches Feld wir als ‘private‘ deklariert
  - Zugriff erfolgt über set- und get-Methoden
- **Array durch Objekt ersetzen**
  - Array, in dem verschiedene Elemente unterschiedliche Dinge bedeuten
- **Zahlen durch symbolische Konstanten ersetzen**

Refactoring-Methoden

IV. Ändern der Vererbungshierarchie

- komplexe Klasse in Unterklassen zerlegen
- **Methode verschieben**
  - **nach oben:** Methoden mit identischen Ergebnissen in verschiedenen Unterklassen in Oberklasse verschieben
  - **nach unten:** Verhalten in einer Oberklasse ist nur für eine ihrer Unterklasse relevant
- **Oberklasse extrahieren**
  - zwei Klassen mit ähnlichen Elementen
  - eine Oberklasse sowie eine Unterklasse erstellen

Ausblick

- Integrität des Systems verbessern
  - große Unternehmen, die an unterschiedlichen Orten mit mehreren Entwicklern Software produzieren

    → selbsterklärender Quellcode

- Faktor Zeit
  - Qualität vs. Einhaltung des Zeitplans für den Projekt-Abschluss

Partner

- Dieses Projekt wurde realisiert mit freundlicher Unterstützung von:

Vielen Dank für Ihre Aufmerksamkeit