

Florian Deißeböck

Erweiterbare Architekturen

23. September 2005

Architektenworkshop



Gliederung

Hintergrund

Persönliches, Software-Qualität, Wartbarkeit

ConQATs Architektur

Entwurf, Prototypen, Konfiguration

Architektur-Evolution

Chancen, Risiken, Anomalien, Konsolidierung

Hintergrund

Persönliches

- | | |
|-----------|--|
| 1998–2003 | Studium der Informatik an der TU München |
| 2001 | Studium am AIT, Bangkok, Thailand |
| 2003 | Diplom (Diplomarbeit Prof. Broy, G. Beneken) |
| seit 2004 | wiss. Mitarbeiter am Lehrstuhl Prof. Broy im
Kompetenzzentrum Software-Wartung (CCSM) |



Erweiterbare Architekturen

- DA: Architektur für erweiterbare Quelltext-Generatoren
- CCSM-Tool-Suite inkl. *CloneDetective*
- Eclipse
- ConQAT

Q-Modell »Wartbarkeit«

- Beschreibung der Einflussfaktoren
- Erklärung der Zusammenhänge
- Anleitung zur Bewertung
- ▶ Einflussfaktoren sind zahlreich und vielfältig

Q-Prozess

- »Erhaltung ist einfacher als Reparatur«
- zeitnahes/kontinuierliches Qualitäts-Controlling
- Erhöhung der Disziplin
- ▶ höhere Produktqualität und geringere Wartungsaufwände

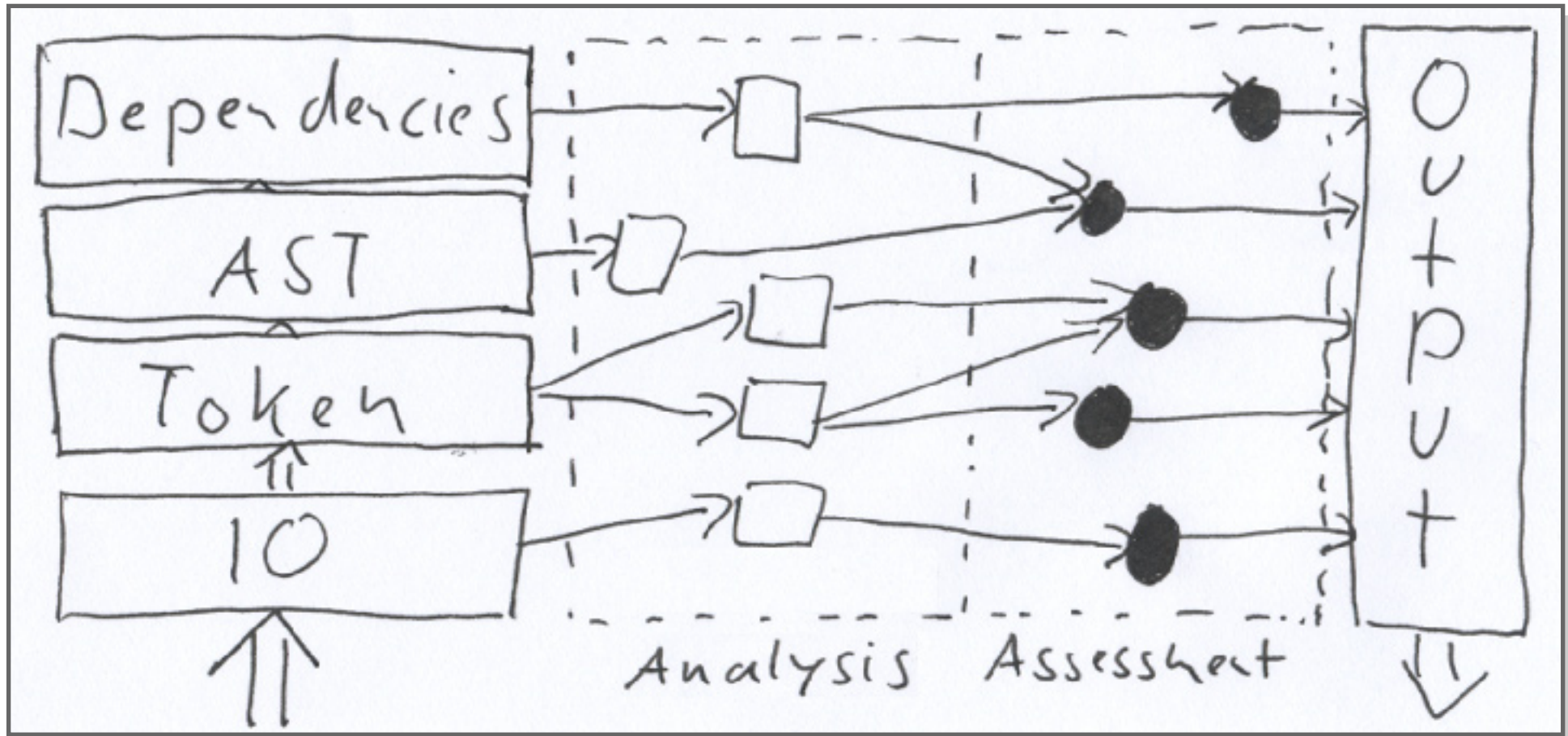
Tool-Support

- Qualitäts-Controlling ist teuer
- Automatisierung notwendig
- ▶ Entwicklung von ConQAT

Anforderungen

- Integrative Darstellung
- Batch-Betrieb
- Informations-Verdichtung
- Vielfältigkeit
- Erweiterbarkeit
- Flexibilität
- Skalierbarkeit

Architektur-Entwurf

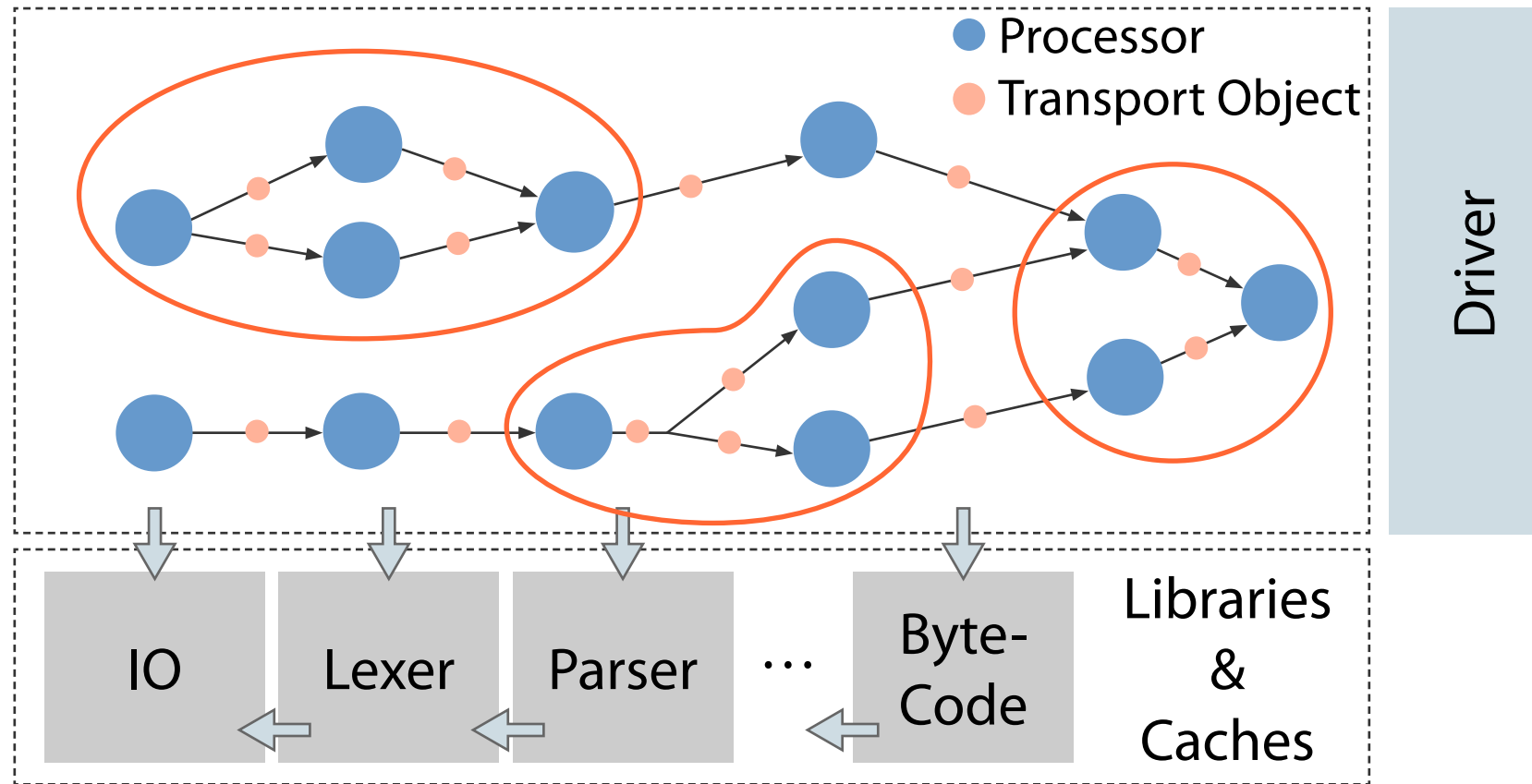


Prototypen

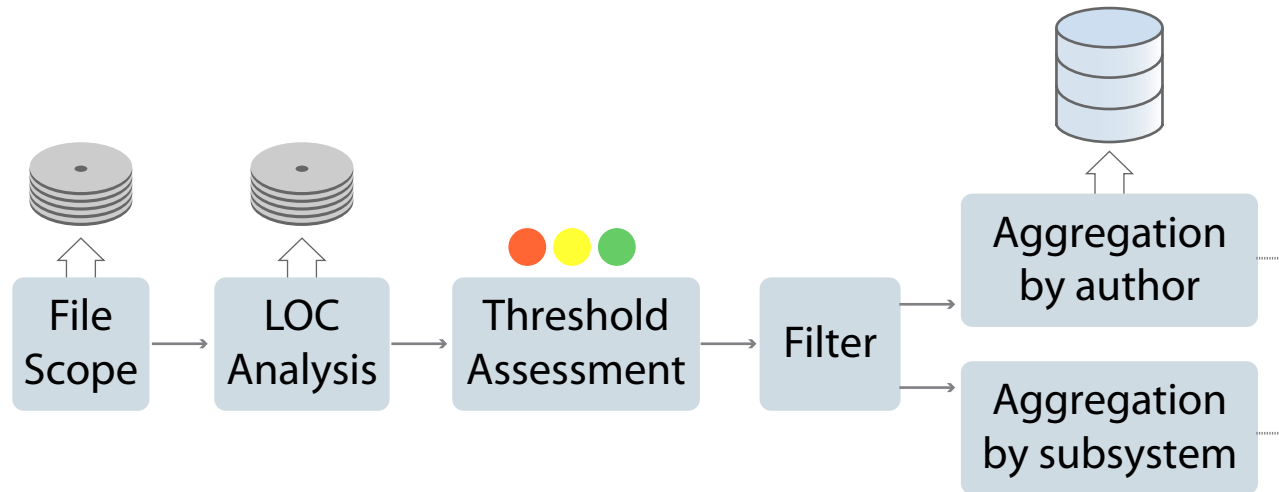
- Entwicklung verschiedener Prototypen
- Spektrum
 - Flexibilität vs Mächtigkeit
 - Komposition
 - Infrastruktur
- keine »ideale« Stelle auf dem Spektrum
- ▶ Ausbruch aus dem Spektrum



Architektur



Beispiele



Scoping

- Dateisystem
- SCM
- Unit-Tests
- UML-Modelle

Analyse

- Metriken
- Toter Code
- PMD-Adapter
- Dokumentation
- Review-Kommentare

Ausgabe

- HTML
- Text

Konfiguration

```
<processor id="source" class="FileSystemScope">  
  <root dir="src" />  
</processor>
```

```
@AConfigProcessor(description = "File systems scope")  
public class FileSystemScope implements IProcessor {  
  
  @AConfigElement(name = "root")  
  public void setRootDirectory(  
    @AConfigAttribute(name = "dir") String rootDir) {  
    ...  
  }  
  
  public IFileSystemElement process(  
    ...  
  }  
}
```

Load-time
Type Checking

```
<processor id="loc-analysis" class="LOCAalyzer">  
  <input ref="@source" />  
</processor>
```

```
@AConfigProcessor(description = „Count LOC“)  
public class LOCAalyzer implements IProcessor {  
  
  @AConfigElement(name = „input“)  
  public void setRoot(  
    @AConfigAttribute(name = „ref“) IFileSystemElement root) {  
    ...  
  }  
  ...  
}
```

ConQAT in Action

TUM ConQAT
CCSM (created at Fri Sep 16 10:26:02 CEST 2005)

Overview

- Repository overview
- Repository overview (full)
- CloneDetective - Unused Code
- CloneDetective - Design
- CloneDetective - Rating
- CloneDetective - Doc
- CloneDetective - Unit Tests
- CloneDetective - Metrics
- ConQAT - Unused Code
- ConQAT - Design
- ConQAT - Rating
- ConQAT - Doc
- ConQAT - Unit Tests
- ConQAT - Metrics
- Config Graph
- Log Messages

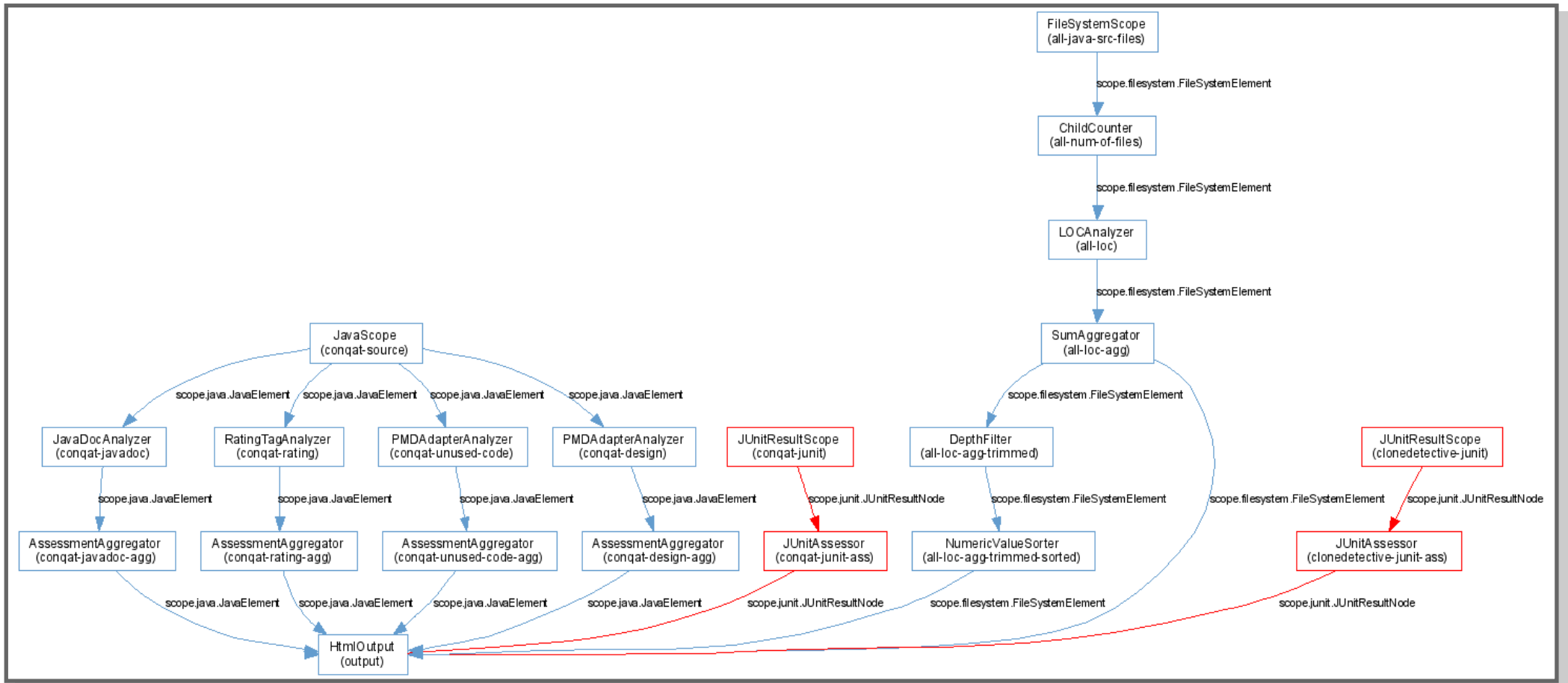
Overview

Assessment	Description	Details
	Repository overview	
	Repository overview (full)	
	CloneDetective - Unused Code	[RED: 4, GREEN: 69]
	CloneDetective - Design	[RED: 28, GREEN: 45]
	CloneDetective - Rating	[RED: 69, GREEN: 4]
	CloneDetective - Doc	[RED: 10, GREEN: 63]
	CloneDetective - Unit Tests	[RED: 2, GREEN: 44]
	CloneDetective - Metrics	
	ConQAT - Unused Code	[RED: 7, GREEN: 131]
	ConQAT - Design	[RED: 32, GREEN: 106]
	ConQAT - Rating	[RED: 3, YELLOW: 5, GREEN: 130]
	ConQAT - Doc	[RED: 10, GREEN: 128]
	ConQAT - Unit Tests	[RED: 3, GREEN: 40]
	ConQAT - Metrics	

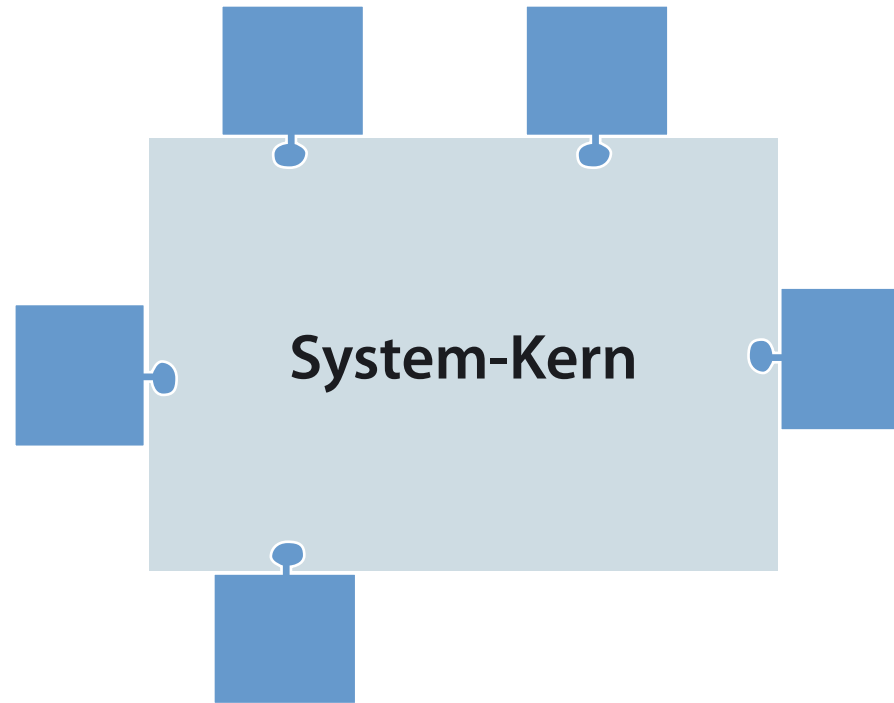
© 2005 Technische Universität München

normalization		
NormalizationFactory		
word		
WordStructure		Method getNextWord() has no JavaDoc comment.
WordNormalization		Method getNextUnit() has no JavaDoc comment.
IStructure		

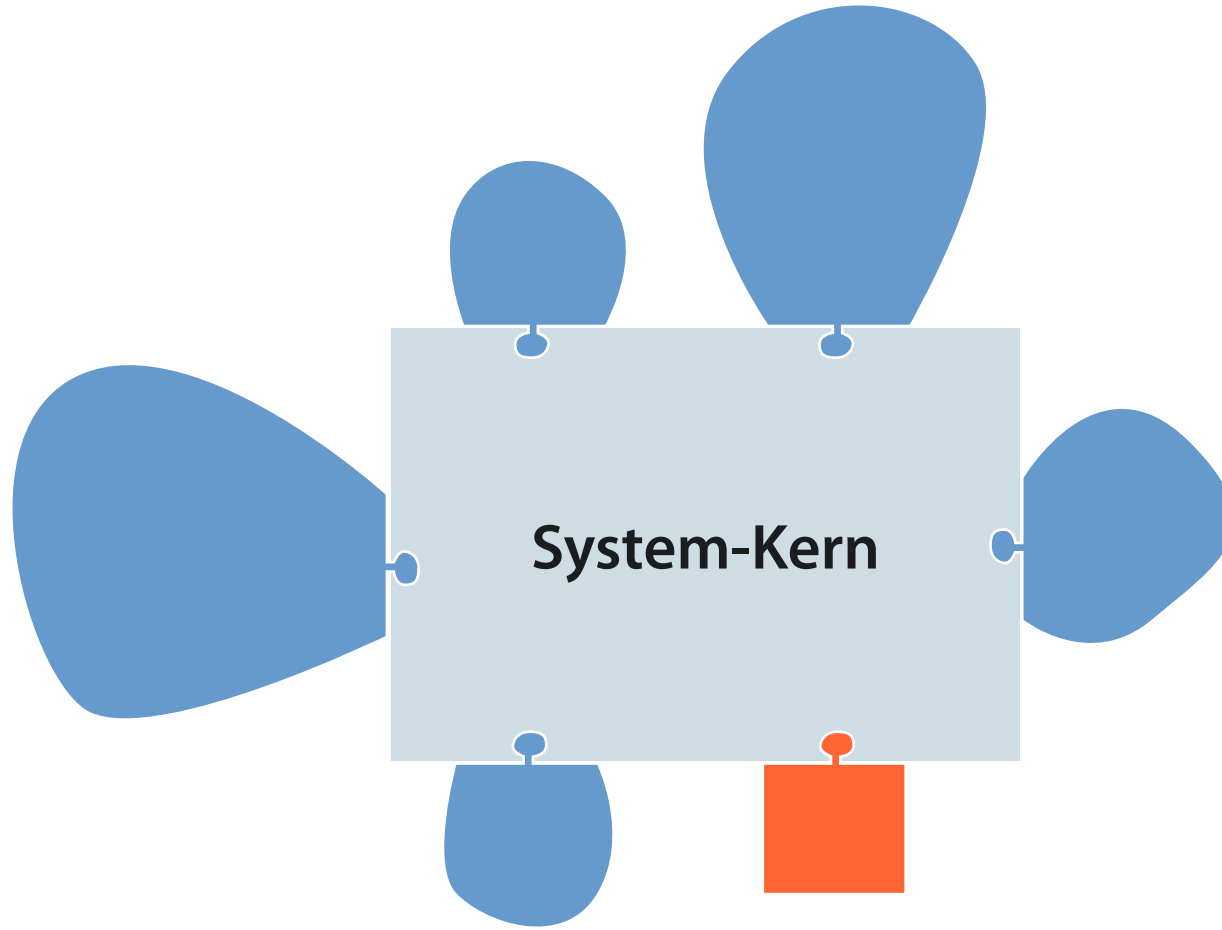
ConQAT in Action



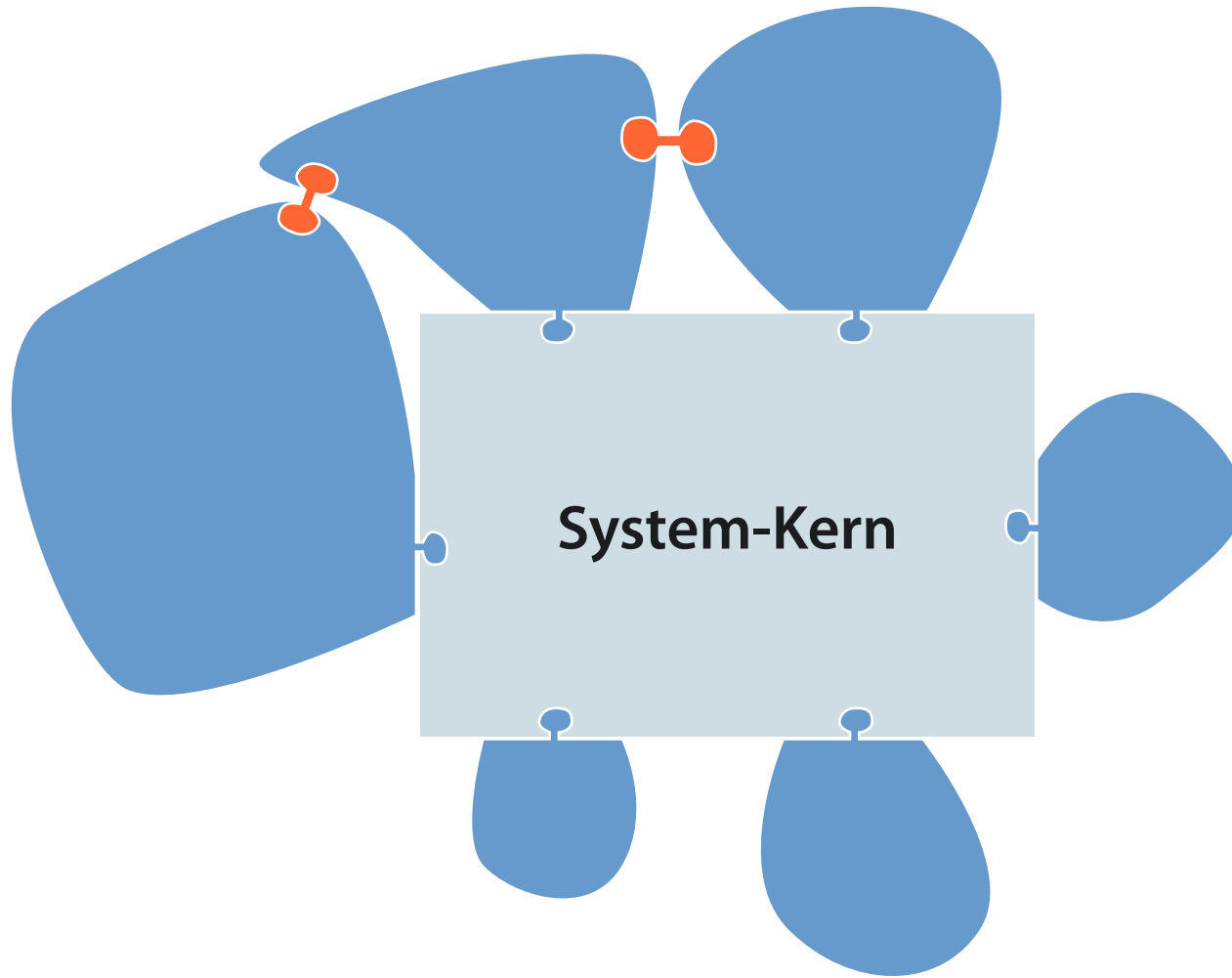
Evolution



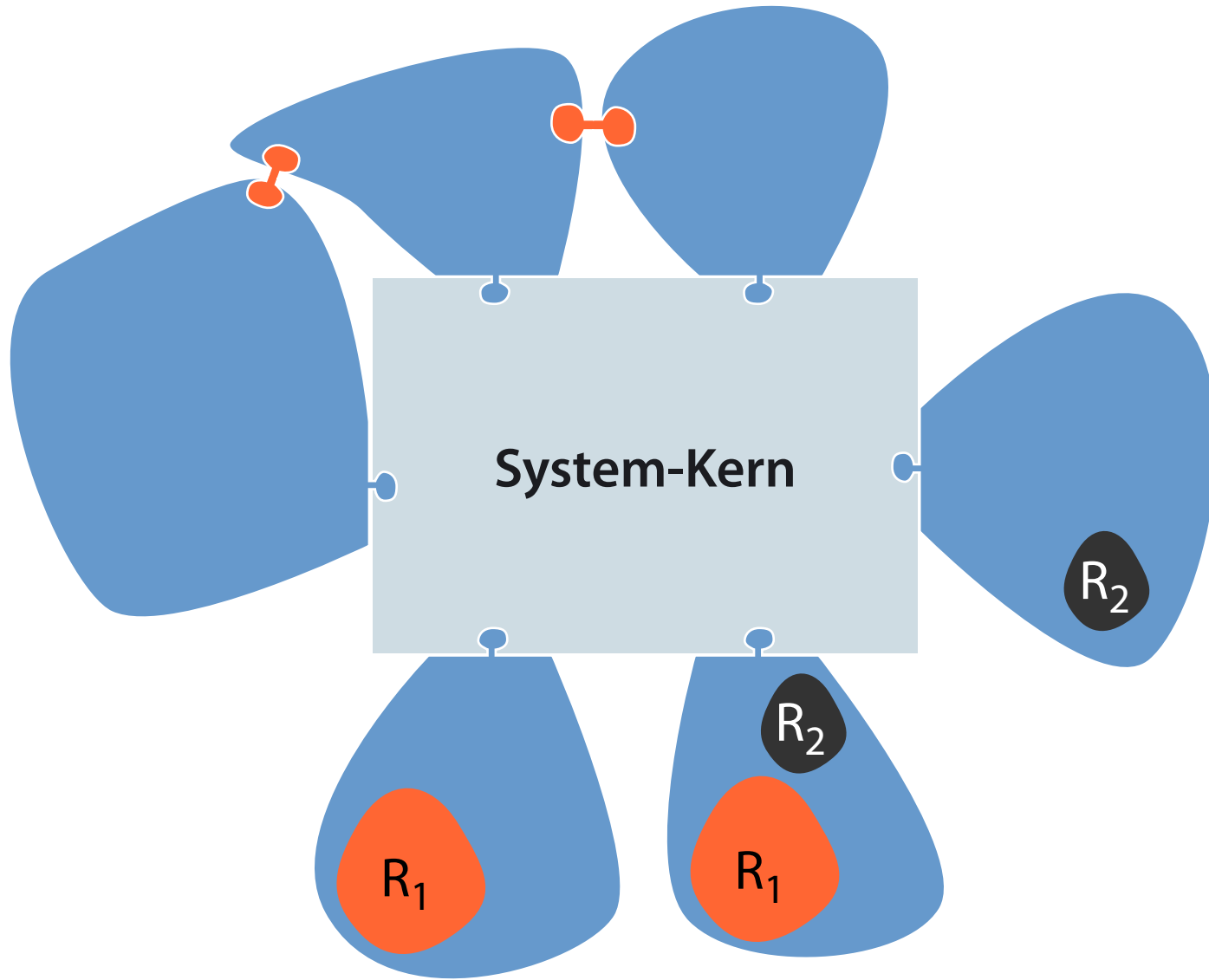
»Bloating«



Abhängigkeiten



Redundanzen



Redundanzen

Compare View

```
file2.setLastModified(time2);
file3.setLastModified(time3);
}

/*
 * Test
 */

public void testModTime() {

    // create scope
    FileSystemScope scope = new FileSystemScope();
    scope.setRootDirectory(rootDir);
    scope.addIncludePattern("*.txt");

    FileSystemElement root = null;
    try {
        root = scope.process();
    } catch (ConQATException ce) {
        logger.debug("Could not process");
    }

    ModificationTimeAnalyzer modTimeAnalyzer = new ModificationTimeAnalyzer();
    modTimeAnalyzer.setRoot(root);

    IFileSystemElement rootWithResults = modTimeAnalyzer.process();

    IFileSystemElement child1WithResults = rootWithResults.getChild(1);
    IFileSystemElement child2WithResults = rootWithResults.getChild(2);
    IFileSystemElement child3WithResults = rootWithResults.getChild(3);

    Object child1Result = child1WithResults.getResult();
    Object child2Result = child2WithResults.getResult();
    Object child3Result = child3WithResults.getResult();

    file1.setLastModified(time1);
    file2.setLastModified(time2);
    file3.setLastModified(time3);
}

public void testModTime() {

    // create scope
    FileSystemScope scope = new FileSystemScope();
    scope.setRootDirectory(rootDir);
    scope.addIncludePattern("*.txt");

    FileSystemElement root = null;
    try {
        root = scope.process();
    } catch (ConQATException ce) {
        logger.debug("Could not process");
    }

    ModificationTimeAnalyzer modTimeAnalyzer = new ModificationTimeAnalyzer();
    modTimeAnalyzer.setRoot(root);

    IFileSystemElement rootWithResults = modTimeAnalyzer.process();

    IFileSystemElement child1WithResults = rootWithResults.getChild(1);
    IFileSystemElement child2WithResults = rootWithResults.getChild(2);
    IFileSystemElement child3WithResults = rootWithResults.getChild(3);

    Object child1Result = child1WithResults.getResult();
    Object child2Result = child2WithResults.getResult();
    Object child3Result = child3WithResults.getResult();
}
```

```
public void putProperty(String name, Object property) {
    if (properties == null) {
        properties = new HashMap();
    }
    properties.put(name, property);
}

public Object getProperty(String name) {
    if (properties == null) {
        return null;
    }
    return properties.get(name);
}

public String[] getPropertyNames() {
    if (properties == null) {
        return null;
    }
    Set propertyNameSet = properties.keySet();
    String[] propertyNames = new String[propertyNameSet.size()];
    propertyNames = propertyNameSet.toArray(propertyNames);
    return propertyNames;
}

String() {
    lngUtils.LINE_FEED;
    lngUtils.INDENT;
}

"ChainedClone [" + #;
"transPos=" + getTransPos() + #;
"refTransPos=" + getRefTransPos() + #;

/* Set position of reference done.
 */
public void setRefPos(int refPos) {
    this.refPos = refPos;
}

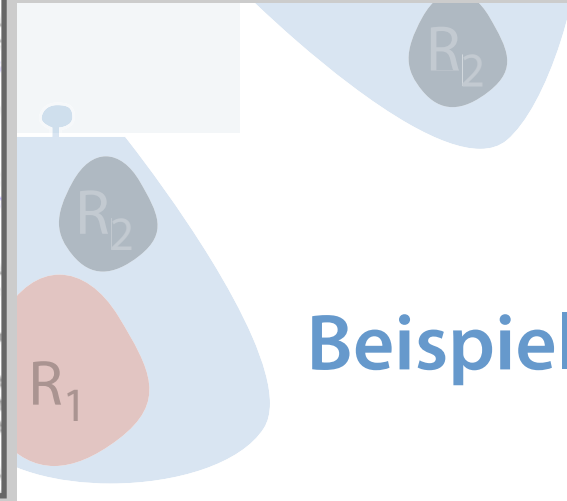
/* Set transformed position of reference done.
 */
public void setRefTransPos(int refTransPos) {
    this.refTransPos = refTransPos;
}

public void putProperty(String name, Object property) {
    if (properties == null) {
        properties = new HashMap();
    }
    properties.put(name, property);
}

public Object getProperty(String name) {
    if (properties == null) {
        return null;
    }
    return properties.get(name);
}

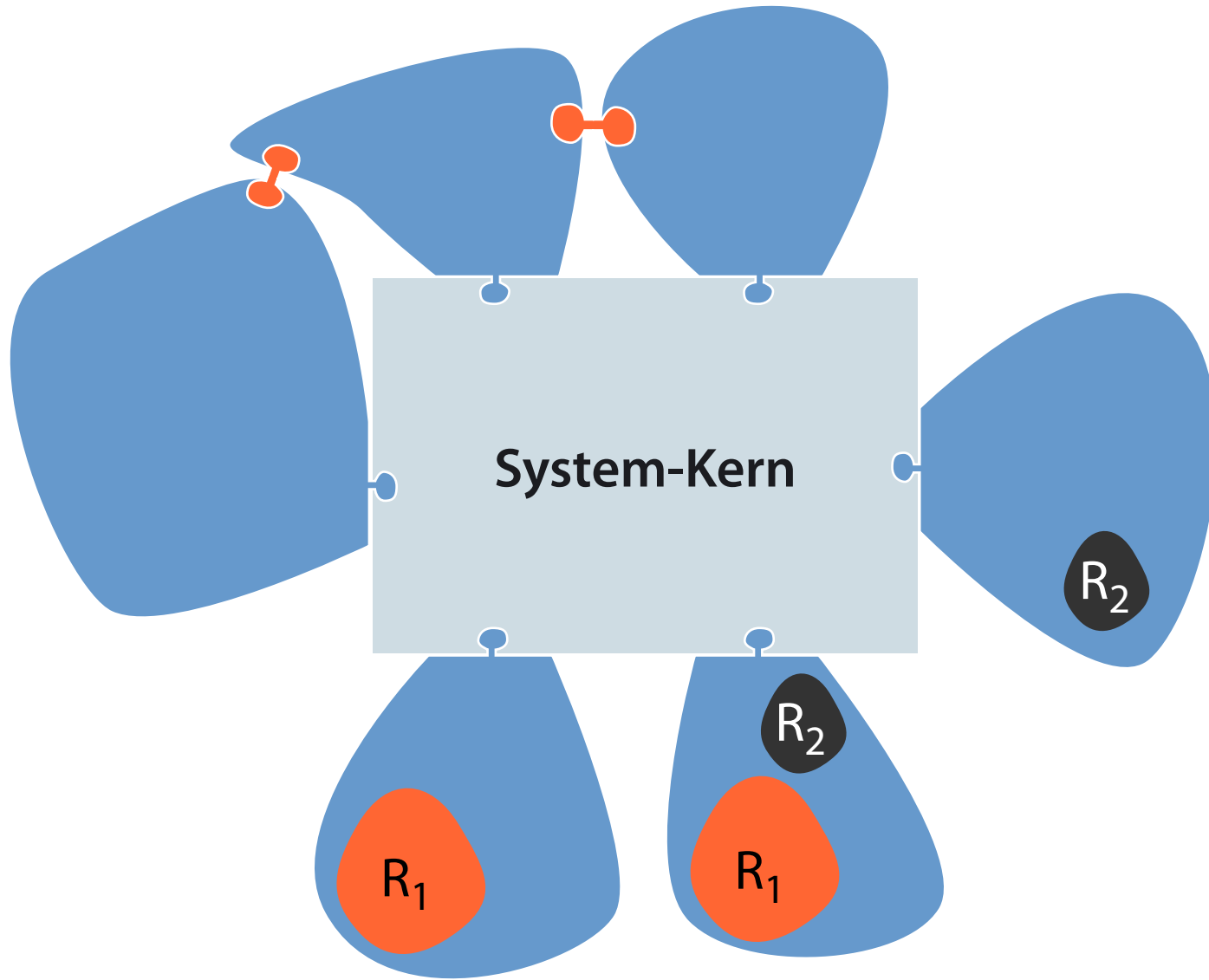
public String[] getPropertyNames() {
    if (properties == null) {
        return null;
    }
    Set propertyNameSet = properties.keySet();
    String[] propertyNames = new String[propertyNameSet.size()];
    propertyNames = propertyNameSet.toArray(propertyNames);
    return propertyNames;
}
```

Clone View

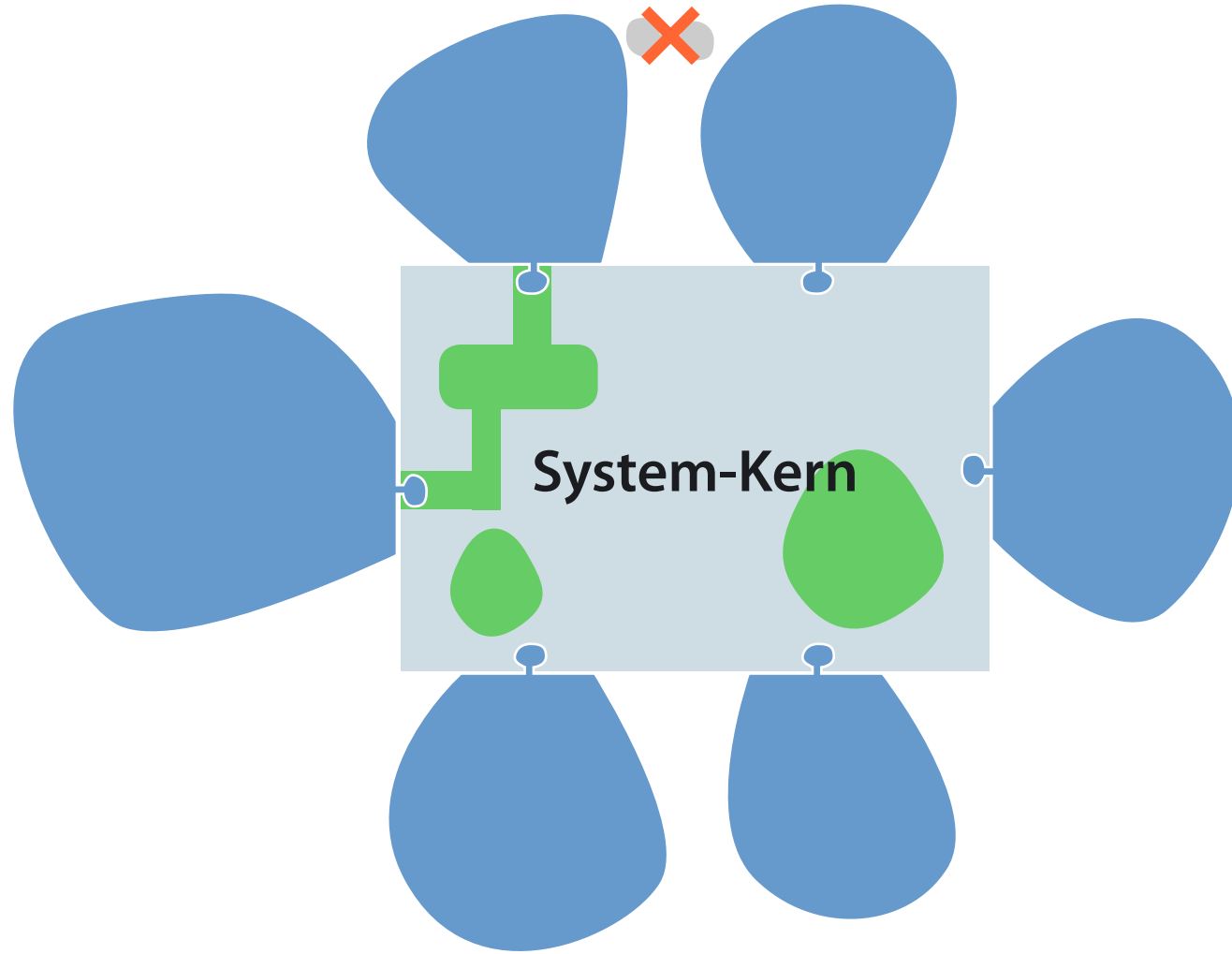


Beispiel: Eclipse 3.1

Redundanzen



Konsolidierung



Agile Architektur-Evolution

- in unserem Projekt sehr positive Erfahrungen
 - gradueller Reifungsprozess (*Sandbox*)
 - kurze Entwicklungszeiten
 - ideal für verteilte Entwicklung
 - erleichtert durch Typ-Sicherheit
- kontinuierliches Controlling
 - Change-Management
 - manuelle Reviews
 - Clone-Detection
 - semi-automatische Architektur-Überprüfung
- kontinuierliche Konsolidierung (*Refactoring*)