

Palladio Fluent API Model Generator – Usage Model

Praktikum: Werkzeuge für Agile Modellierung

Eva-Maria Neumann

Betreuer: Yves Richard Kirschner

Inhalt

- Motivation
- Grundlagen
 - Palladio Model
 - Fluent Interfaces
- Palladio Fluent API
 - Vorarbeiten
 - Besonderheiten Usage Model

Motivation

■ Graphische Editoren

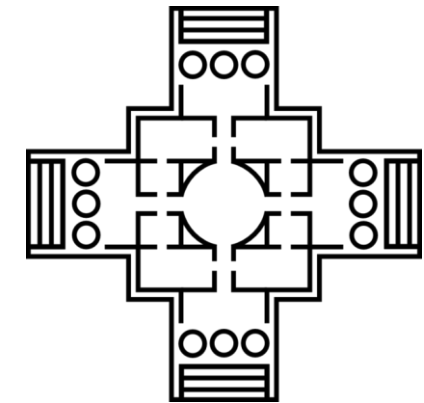
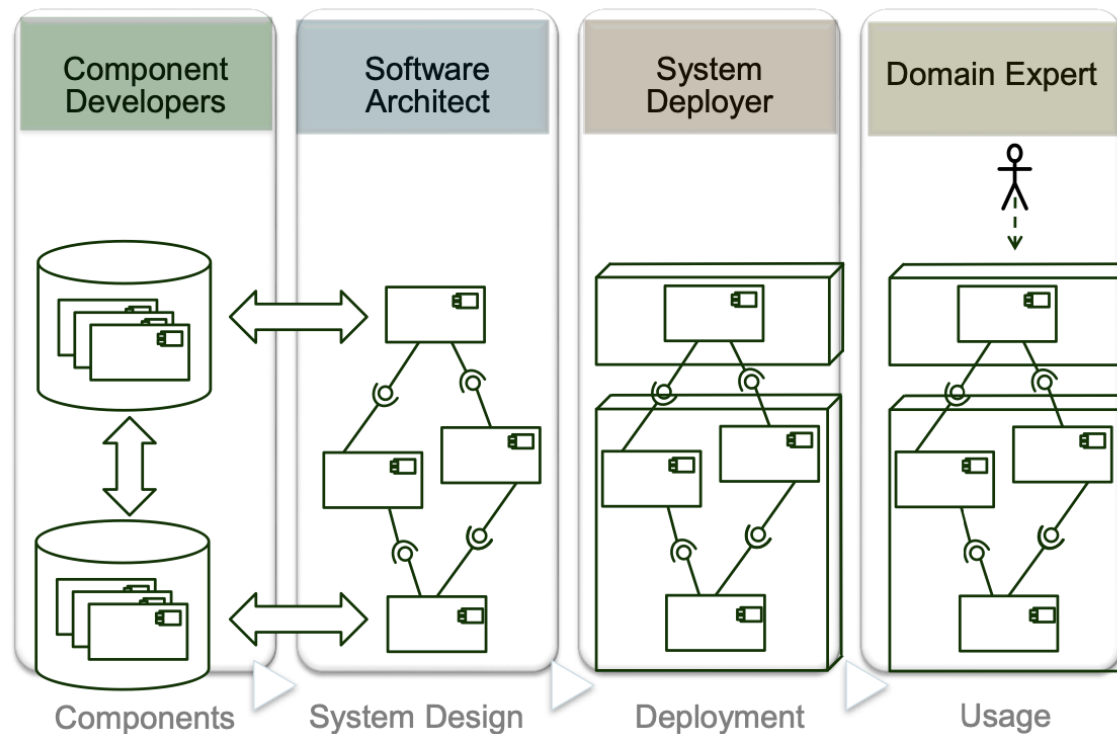
- Unübersichtlich
- Viel Klicken
- Anschaulich

■ Textuell

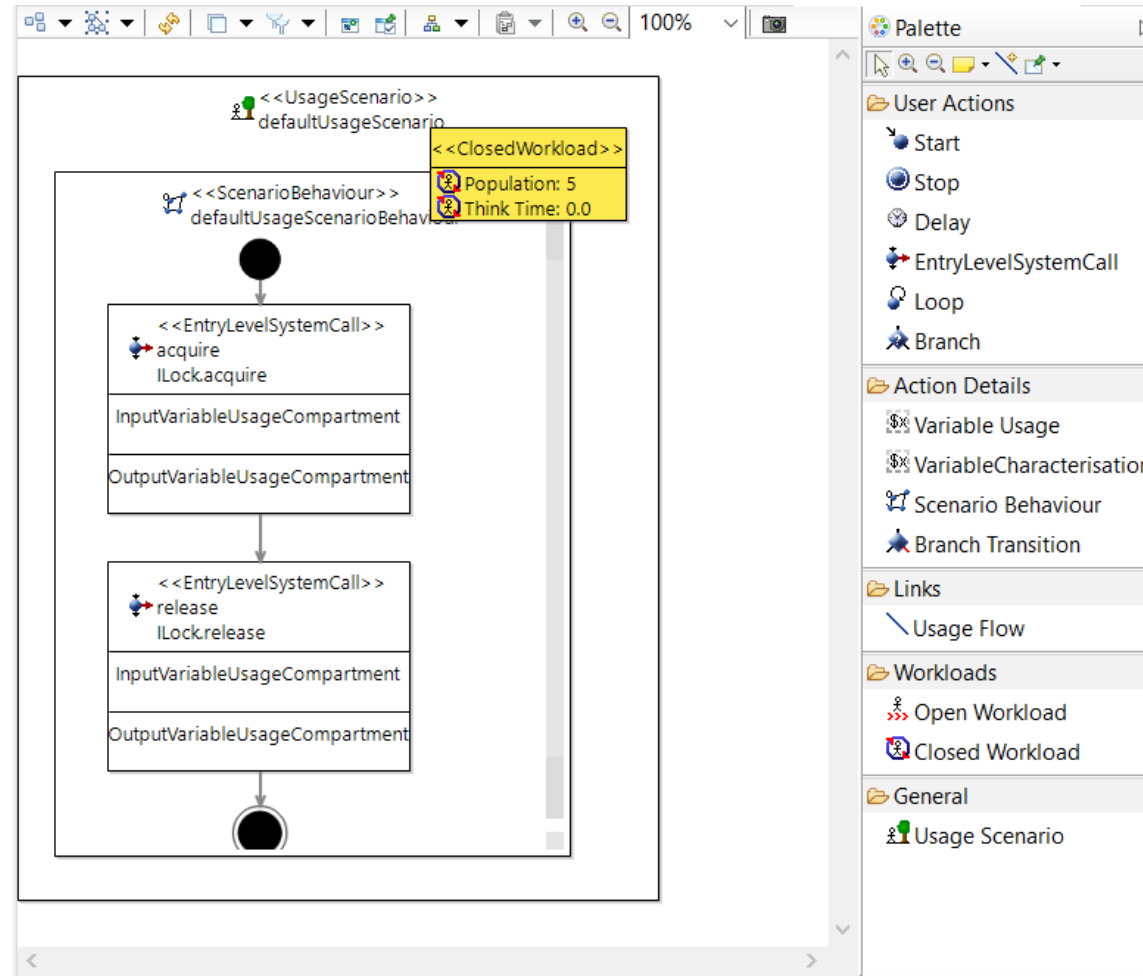
- Schneller als graphisch
- Aufwändig
- Viel Text

Palladio Model

- Ansatz zur Software Architektur Simulation
- Domänspezifische Modellsprache mit Metamodel in EMF/Ecore



Palladio Model



Fluent Interfaces

- Natürliche Sprache
- Verkettung von Methoden
- Ziel: lesbarer, kompakter Code

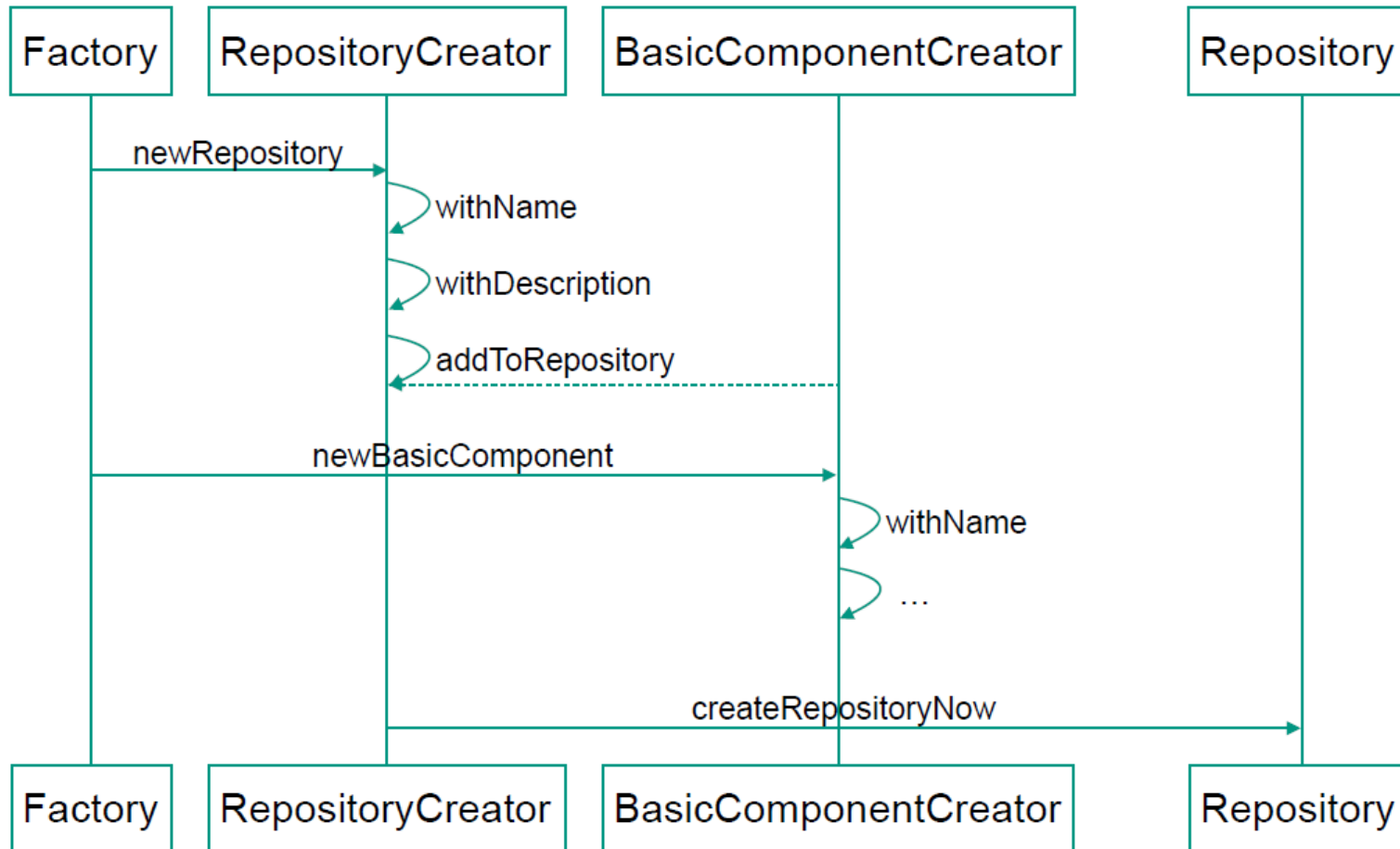
Fluent Interfaces

```
private void makeNormal(Customer customer) {  
    Order ol = new Order();  
    customer.addOrder(ol);  
    OrderLine line1 = new OrderLine(6, Product.find("TAL"));  
    ol.addLine(line1);  
    OrderLine line2 = new OrderLine(5, Product.find("HPK"));  
    ol.addLine(line2);  
    OrderLine line3 = new OrderLine(3, Product.find("LGV"));  
    ol.addLine(line3);  
    line2.setSkippable(true);  
    ol.setRush(true);  
}
```

```
private void makeFluent(Customer customer) {  
    customer.newOrder()  
        .with(6, "TAL")  
        .with(5, "HPK").skippable()  
        .with(3, "LGV")  
        .priorityRush();  
}
```

Quelle: <https://martinfowler.com/bliki/FluentInterface.html> (Martin Fowler)

Palladio Fluent API - Vorarbeiten



Methodenverkettung:
`a().b().c()`



Argument:
`a(x)`



Besonderheiten Usage Model

- Aufbau Fluent API Usage Model in eigener Factory
- Verkettung von Aktionen
 - Start/ Stop automatisiert hinzufügen
 - Reihenfolge wichtig
- Unterstützung der Verifikation durch Parameter
 - Etwas unübersichtlicher
 - Vermeidung von Fehlern
 - Einfach Erzeugung von gültigem Model

10

Ohne Verifikation

- UsageScenario
 - Hat ScenarioBehavior
 - Hat Workload (mit Zeit)

```
UsageModel usgModel = create.newUsageModel().addToUsageModel(  
    create.newUsageScenario()  
        .addToUsageScenario(create.newScenarioBehavior())  
        .addToUsageScenario(create.newOpenWorkload()  
            .withArrivalTime("10"))  
    .withName(name))  
.createUsageModelNow();
```

Mit Verifikation

- UsageScenario
 - Hat ScenarioBehavior
 - Hat Workload (mit Zeit)

```
UsageModel usgModel = create.newUsageModel().addToUsageModel(  
    create.newUsageScenario(  
        create.newScenarioBehavior()  
        , create.newOpenWorkload("10"))  
    .withName(name))  
.createUsageModelNow();
```

```
create.newBranchAction().withName(name).addToBranchAction(  
    create.newBranchTransition(  
        create.newScenarioBehavior().withName(scenName))  
    .withProbability(prop)))
```

■ Problem: Wahrscheinlichkeit nicht überprüfbar

- `org.palladiosimulator.generator.fluent.shared.validate.ModelValidator logResult`
- SCHWERWIEGEND: Validation for model "UsageModel": Diagnostic ERROR
source=org.palladiosimulator.pcm.usagemodel code=0 The 'allBranchProbabilitiesMustSumUpTo1'
constraint is violated on
'org.palladiosimulator.pcm.usagemodel.impl.BranchImpl@4ef18604{#_02Mo5hcBEey7T-wAh8Eu0g}'
data=[Branch[TRANSIENT]]

Verkettung Aktionen

- ScenarioBehaviour hat Aktionen
 - Aktion: Vorgänger, Nachfolger
 - Erste: Immer Start
 - Letzte: Immer Stop

```
UsageModel usgModel = create.addSystem(createSimplifiedMediaStoreSystem())  
    .newUsageModel().addToUsageModel(create.newUsageScenario(  
        create.newScenarioBehavior()  
            .addToScenarioBehaviour(  
                ...  
            ),  
        create.newOpenWorkload("0")))  
    .createUsageModelNow();
```

Verkettung Aktionen

```
.addToScenarioBehaviour(create.newStartAction())  
.addToScenarioBehaviour(create.newDelayAction("10"))  
.addToScenarioBehaviour(create.newBranchAction  
.addToScenarioBehaviour(create.newEntryLevelSystemCall(  
    create.fetchOffOperationRoleAndSignature  
    ("SimplifiedMediaStore System", provRoleName, operSigName))  
.addToScenarioBehaviour(create.newLoopAction("1",  
    create.newScenarioBehavior()))  
.addToScenarioBehaviour( create.newStopAction())
```

- Fehlend: Verkettung
- Problem: Verweis

Verkettung Aktionen

```
create.newStartAction().withSuccessor(  
    create.newDelayAction("10").withSuccessor(  
        create.newBranchAction().withSuccessor(  
            create.newEntryLevelSystemCall(  
                create.fetchOffOperationRoleAndSignature  
                ("System",provRoleName, operSigName))  
            .withSuccessor(  
                create.newLoopAction("1", create.newScenarioBehavior())  
                .withSuccessor(  
                    create.newStopAction()))))  
        ))  
    ))
```

- Prüfung auf Start und Stop
- Vorgänger markieren

Test

■ Vorher

- Model per Fluent API aufbauen
- Model normal aufbauen
- Vergleichen

■ Jetzt


- Model per Fluent API aufbauen
- Unit Test von Parametern im Model
- Großer Test
- Einzelne Bereiche eines Models
















Finished after 9,109 seconds

Runs: 15/15

Errors: 0

Failures: 0

▼  org.palladiosimulator.generator.fluent.usagemodel.factory Failure Trace

-  usageScenarioBehavActionsLoop (6,753 s)
-  usageScenarioWorkloadOpen (0,078 s)
-  usageScenarioWorkloadClosed (0,018 s)
-  basicUserData (0,779 s)
-  usageScenarioBehavActionList (0,066 s)
-  usageScenarioBehavActionsEntryLevelSystemCall (0,351 s)
-  basicUsageScenario (0,009 s)
-  basicUsageModelClasses (0,018 s)
-  usrDataAssemblyContext (0,140 s)
-  mediaStore_Realistic (0,234 s)
-  usageScenarioBehavActionsDelay (0,014 s)
-  usageScenarioBehaviour (0,036 s)
-  usageScenarioBehavActions (0,091 s)
-  usageScenarioBehavActionsBranch (0,013 s)
-  usrDataVariableUsage (0,109 s)

Fragen?