

Palladio Fluent API Model Generator – Usage Model

Eva-Maria Neumann

Praktikum: Werkzeuge für Agile Modellierung

Betreuer: Yves Kirschner

Inhalt

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

Motivation

■ Graphische Editoren

- Unübersichtlich bei großen Modellen
- Viel Klicken
- + Anschaulich
- + Benutzerfreundlich
- + Für Einsteiger geeignet

■ Textuell

- Aufwändig
- Viel Quelltext
- Unverständlich
- Unübersichtlich
- + Hohe Lernkurve
- + Schneller

Fluent Interfaces

- Natürliche Sprache
- Verkettung von Methoden

- Besser Vorstellbar
- Weniger Fehler

Fluent Interfaces

- Natürliche Sprache
- Verkettung von Methoden

```
private void makeNormal(Customer customer) {  
    Order o1 = new Order();  
    customer.addOrder(o1);  
    OrderLine line1 = new OrderLine(6,product("TAL"));  
    o1.addLine(line1);  
    OrderLine line2 = new OrderLine(5,product("HPK"));  
    o1.addLine(line2);  
    OrderLine line3 = new OrderLine(3,product("LGV"));  
    o1.addLine(line3);  
    line2.setSkippable(true);  
    o1.setRush(true);  
}
```

- Besser Vorstellbar
- Weniger Fehler

```
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>

Fluent Interfaces

- Natürliche Sprache
- Verkettung von Methoden

```
private void makeNormal(Customer customer) {  
    Order o1 = new Order();  
    customer.addOrder(o1);  
    OrderLine line1 = new OrderLine(6,product("TAL"));  
    o1.addLine(line1);  
    OrderLine line2 = new OrderLine(5,product("HPK"));  
    o1.addLine(line2);  
    OrderLine line3 = new OrderLine(3,product("LGV"));  
    o1.addLine(line3);  
    line2.setSkippable(true);  
    o1.setRush(true);  
}
```

- Besser Vorstellbar
- Weniger Fehler

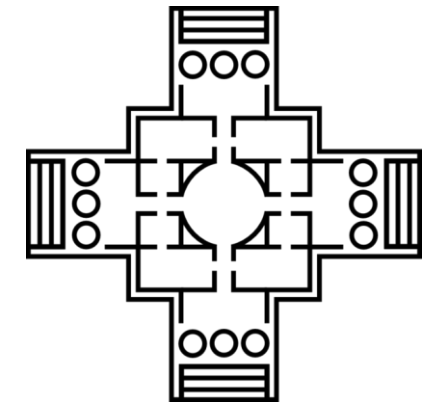
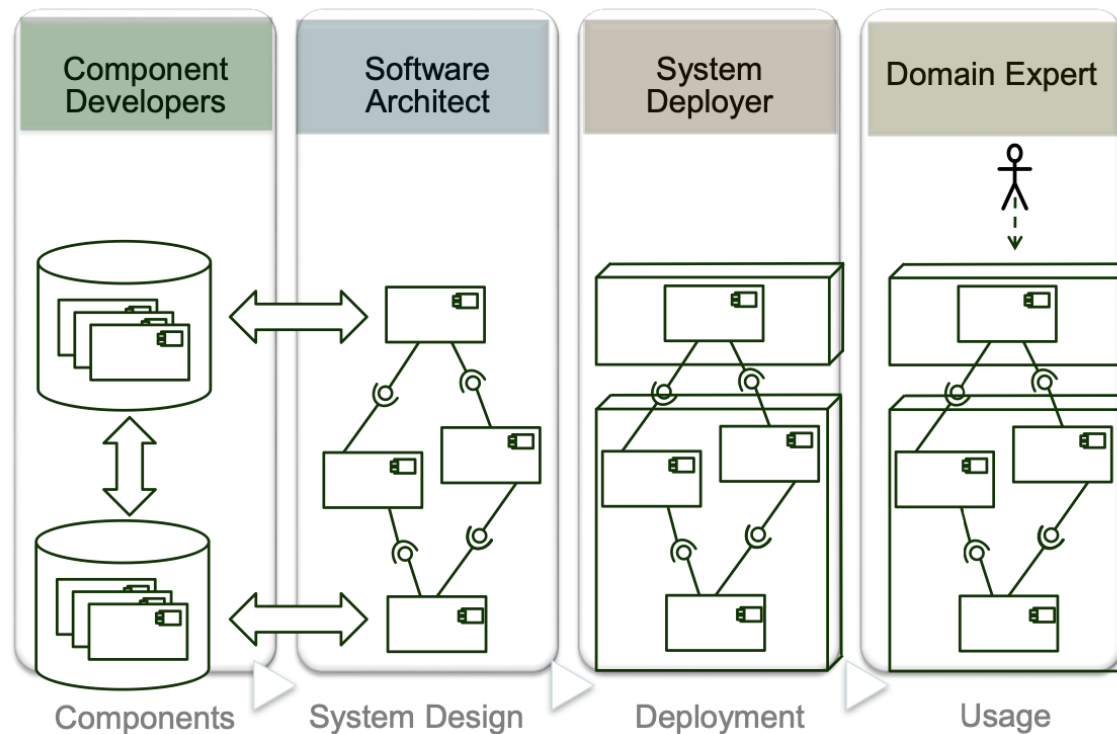
```
private void makeFluent(Customer customer) {  
    customer.newOrder()  
        .with(6, "TAL")  
        .with(5, "HPK").skippable()  
        .with(3, "LGV")  
        .priorityRush();  
}
```

➤ Lesbarer, Kompakter Code

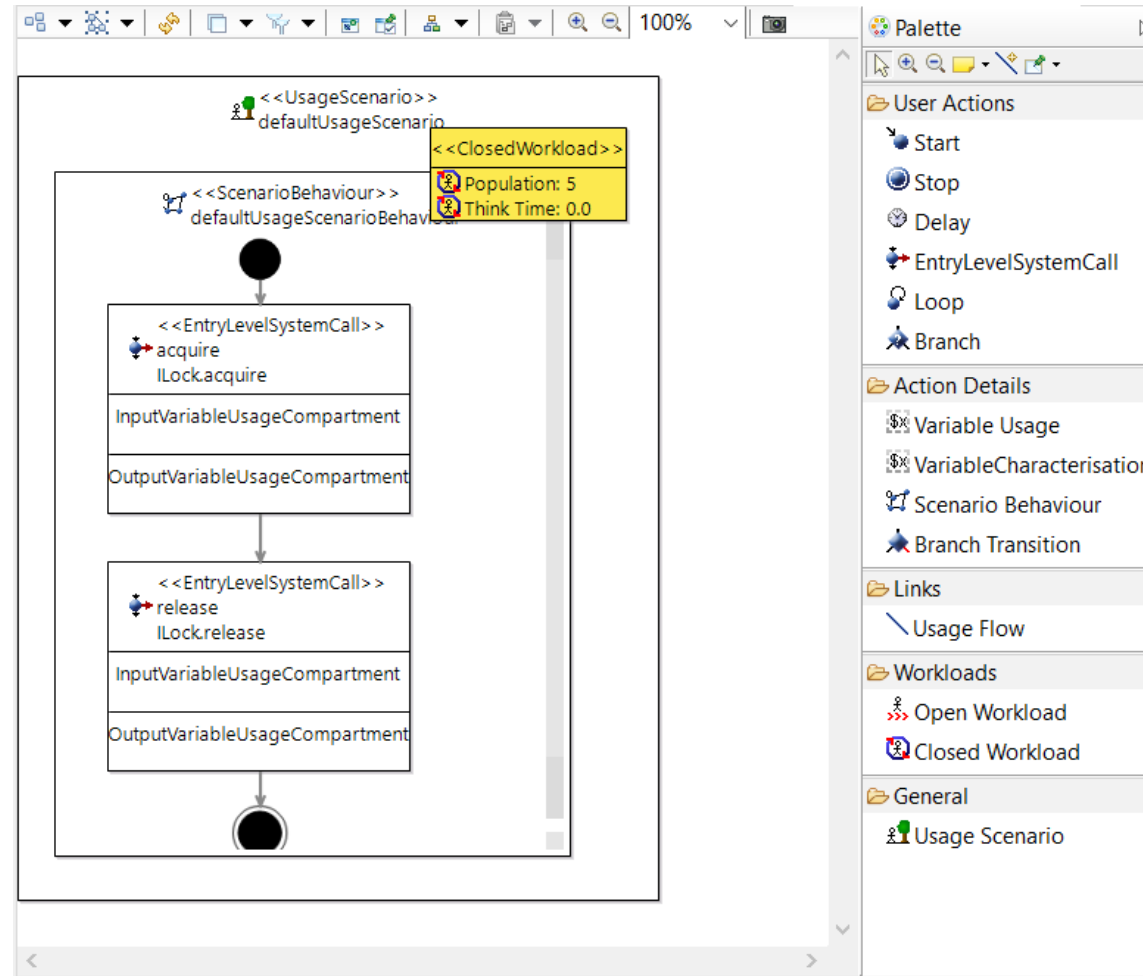
Quelle: <https://martinowler.com/bliki/FluentInterface.html>

Palladio Model

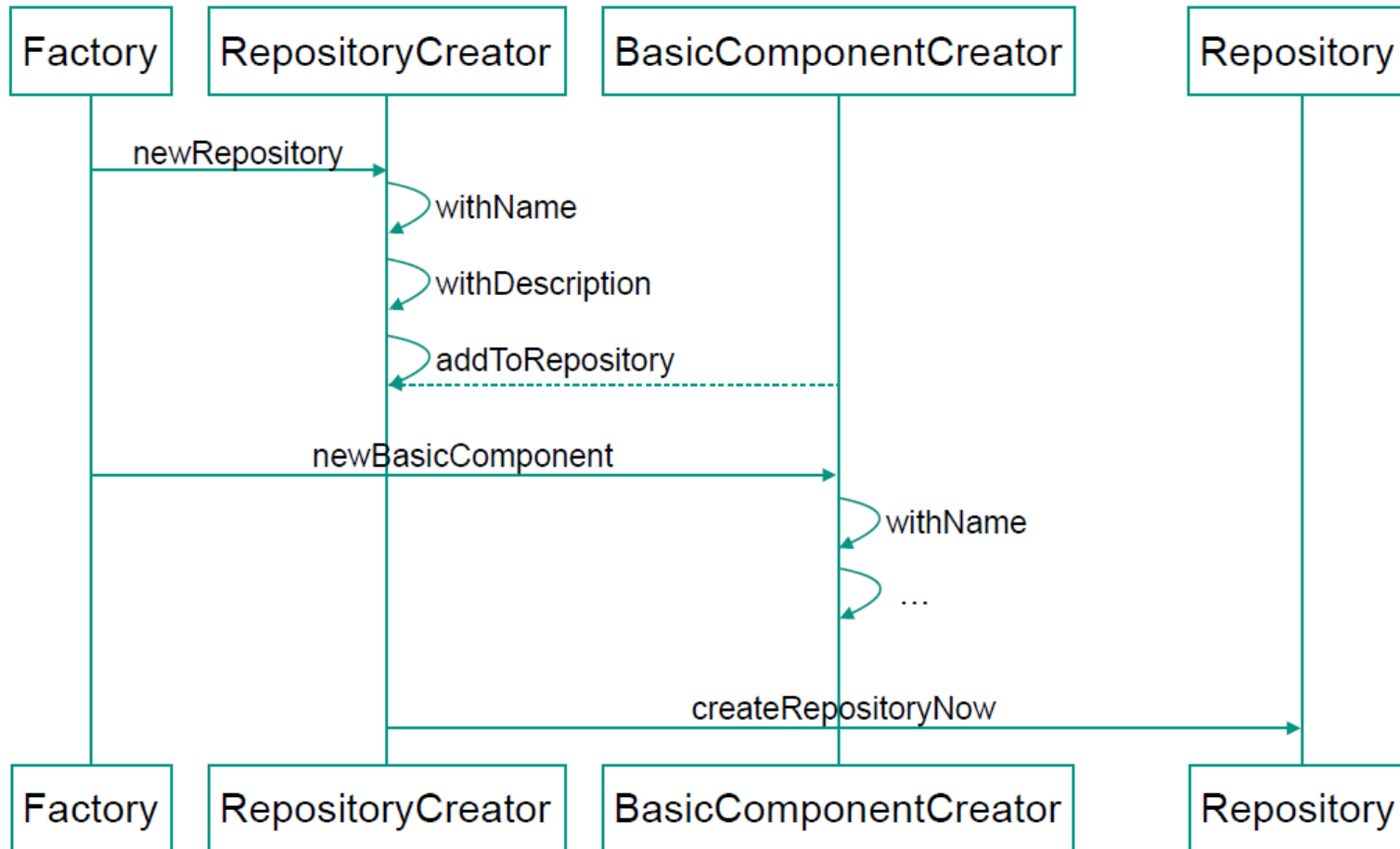
- Ansatz zur Software Architektur Simulation
- Domänspezifische Modellsprache mit Metamodel in EMF/Ecore



Palladio Model



Palladio Fluent API - Vorarbeiten



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



Argument:
a(x)

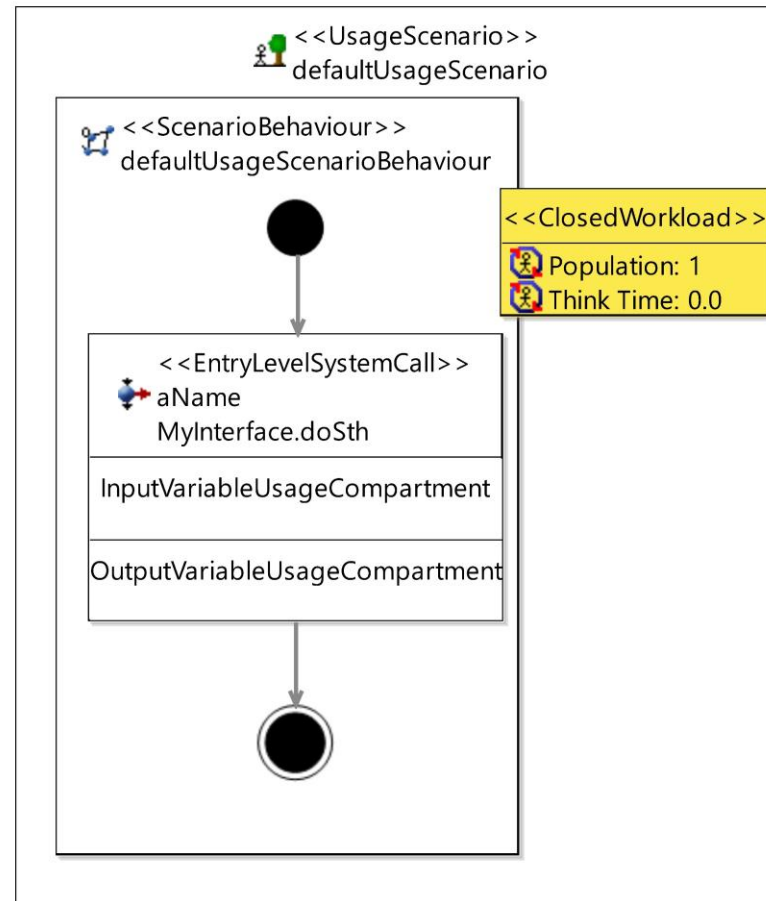


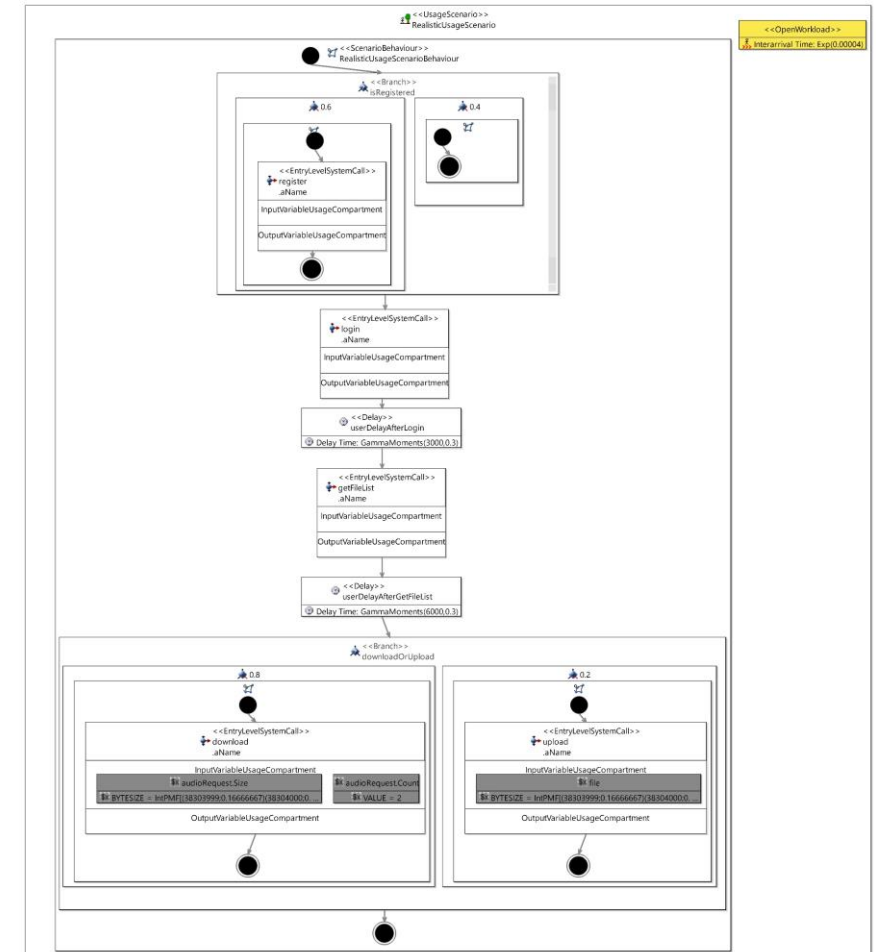
Quelle: Folie von Louisa Lambrecht

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

Usage Model - Beispiel





Ohne Verifikation

■ UsageScenario

- Hat genau ein ScenarioBehavior
- Hat genau einen 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 genau ein ScenarioBehavior
- Hat genau einen Workload (mit Zeit)

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

Verifikation

```
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{#_02Mo5hcBEey7  
T-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

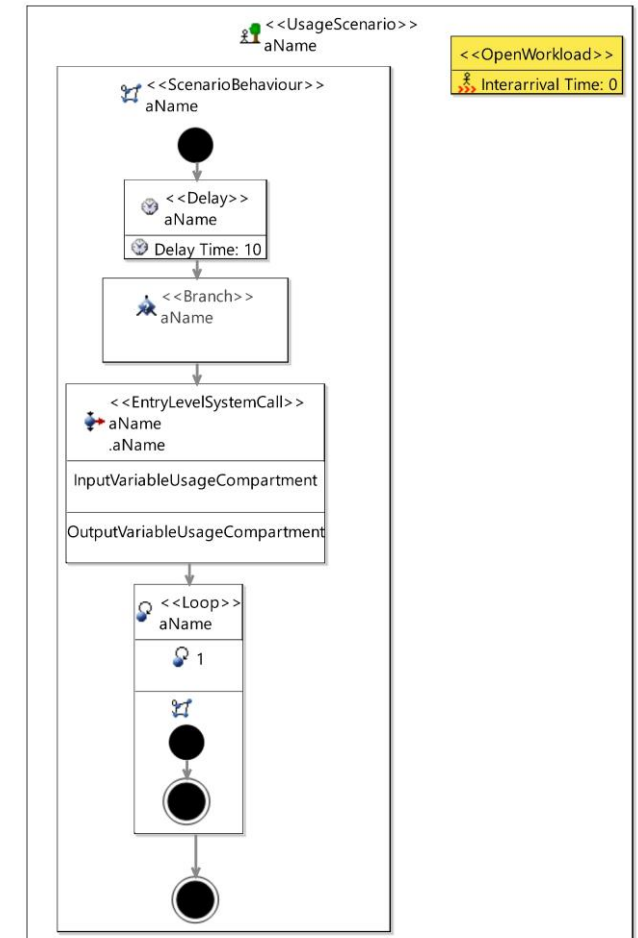
Verkettung Aktionen

```

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



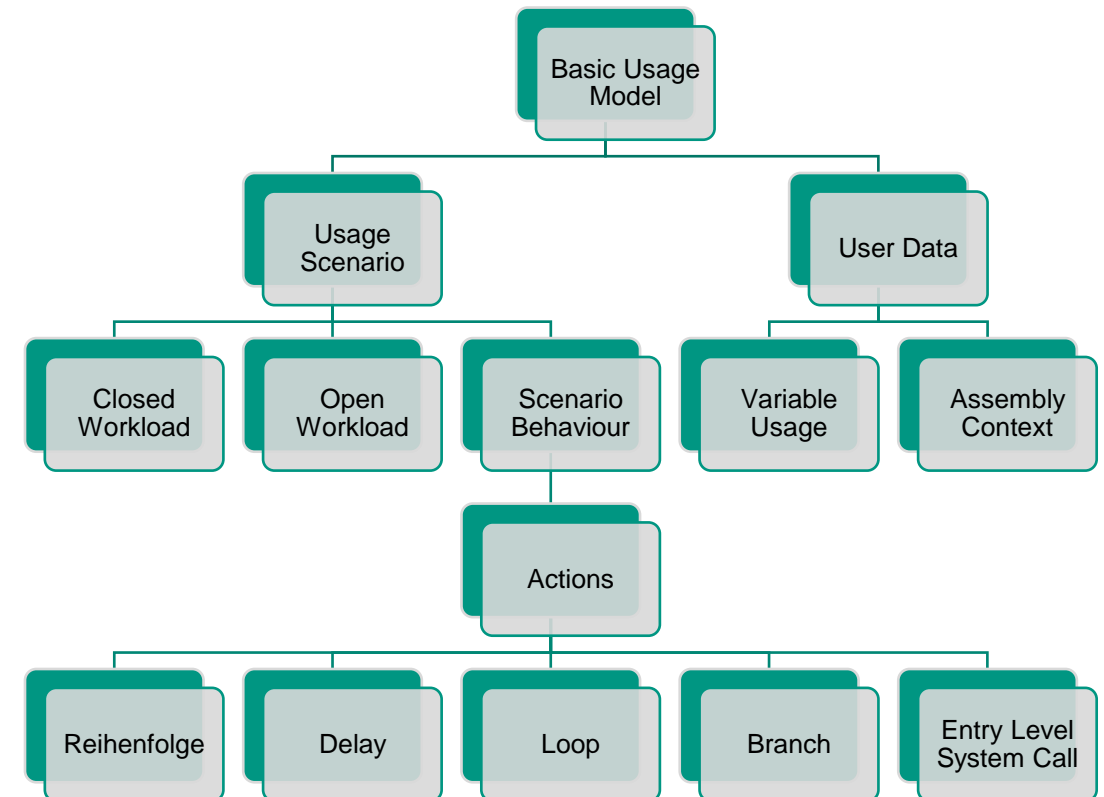
Test

■ Vorher

- Modell per Fluent API aufbauen
- Modell normal aufbauen
- Vergleichen

■ Jetzt

- Modelle per Fluent API aufbauen
- Unit Test von Bereichen und deren Parametern
- Zusätzlich großer Test: „Realistic Media Store“



Fragen?