Modellgetriebene Entwicklung mit CIRO, einer UML-Erweiterung mit Fokus auf Embedded Systeme

Johannes Scheier

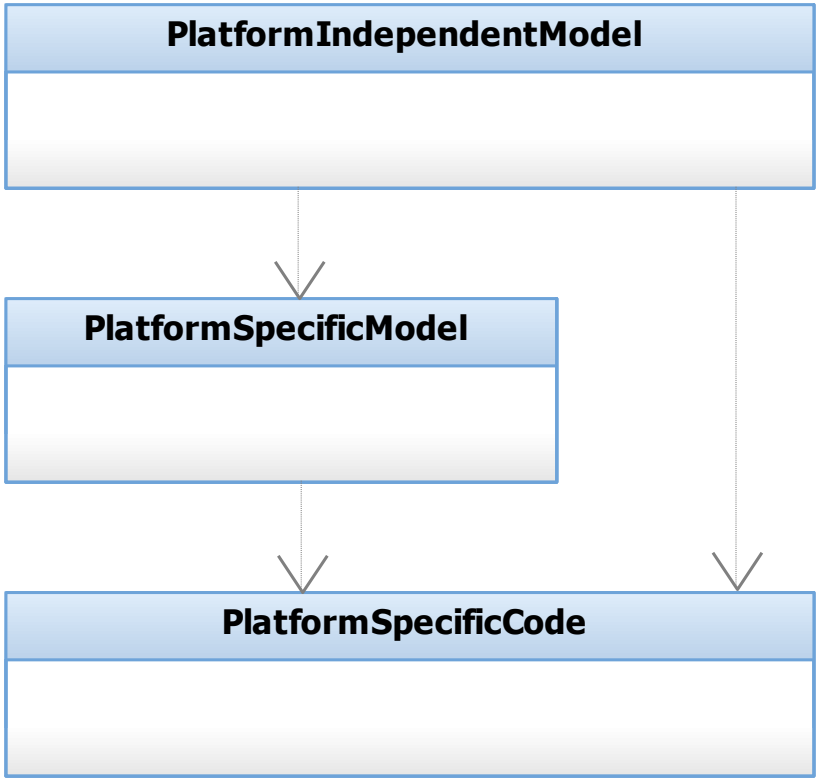
Ein Einblick in die Konzepte

- Model Driven Software Development
- CIRO Erweiterung von UML
- Qualifizierte Konnektoren

Inhalt

- Model Driven Software Development
- Was ist CIRO?
- Konzepte erklärt an einem Beispielmodell
- Simulation und Test
- Code Generierung
- Komposition und qualifizierte Konnektoren
- Ausblick
- Fragen, Diskussion
- Evtl. Demo

Model Driven Development



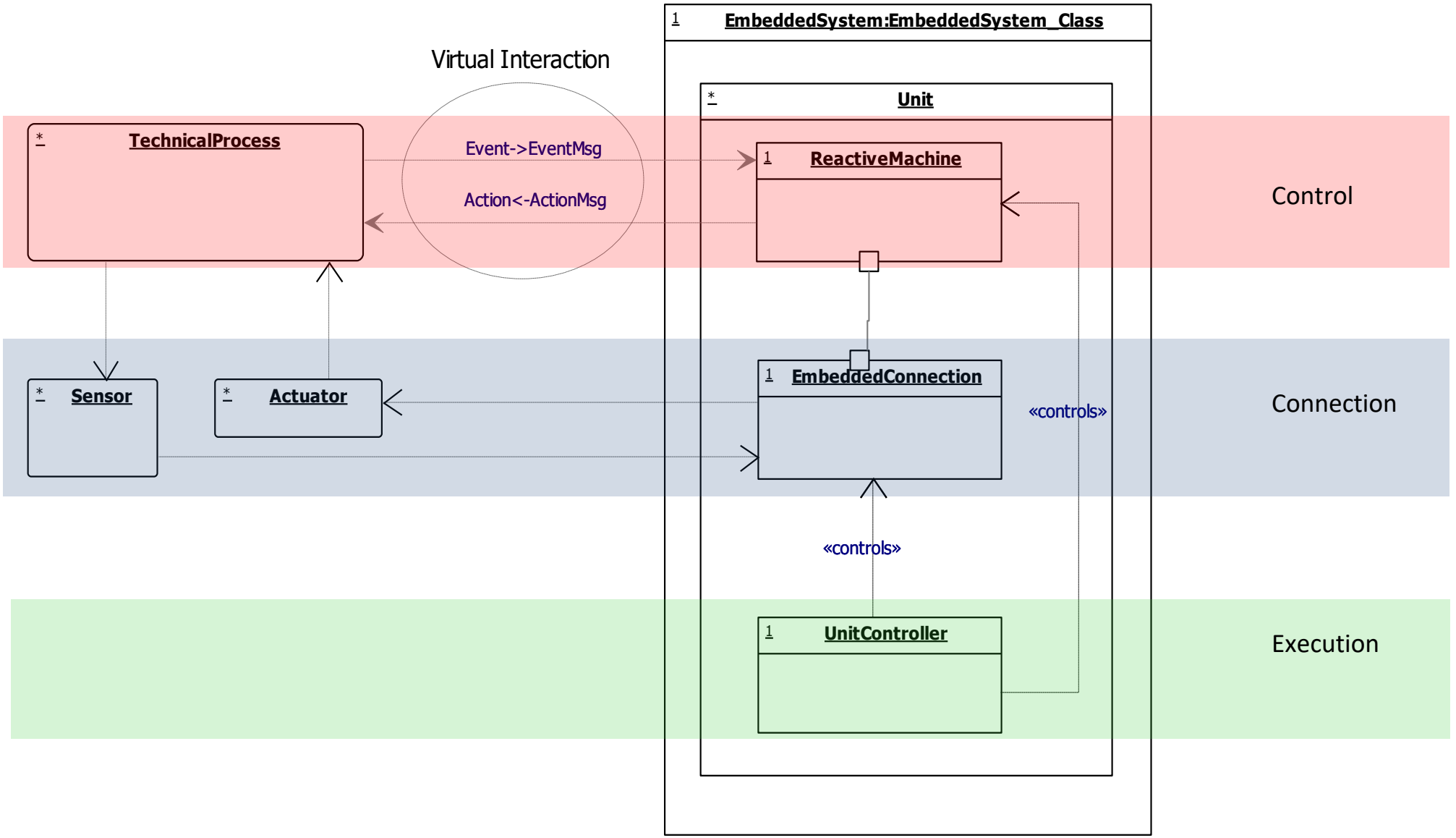
- Modell des Systems
- In einer für die Aufgabe geeigneten Notation
- Modell von Struktur und Verhalten
- Ausführbare und testbare Spezifikation

- Vollständige Code-Generierung aus dem Modell
- Dokumentation und Implementation immer konsistent

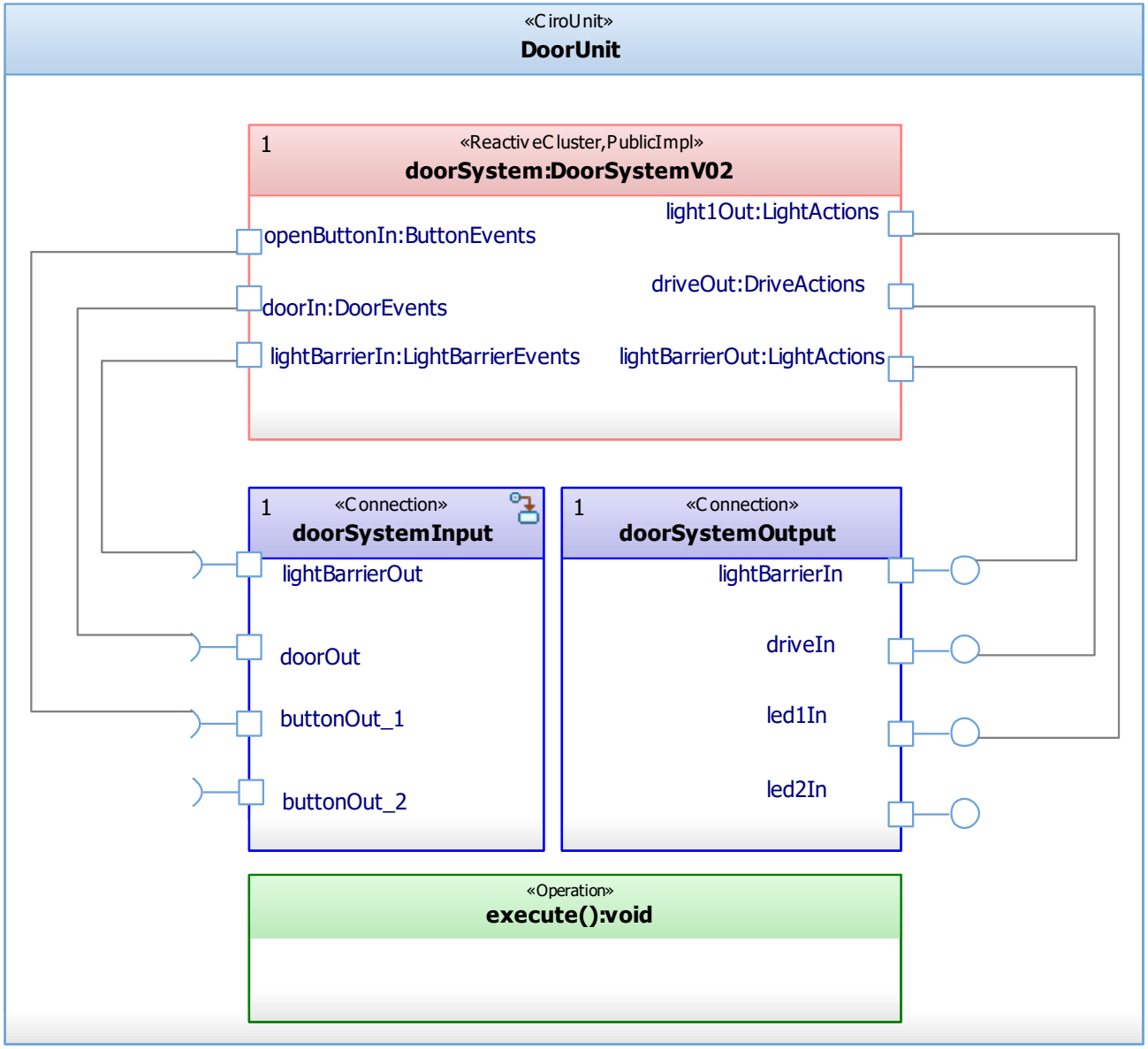
Was ist CIRO?

- Communicating Interacting Reactive Objects
- Eine Erweiterung von UML
- Konzepte geeignet zur Modellierung von Embedded Systemen
- Basierend auf den Ideen von Prof. Dr. Hugo Fierz (CIP, DESC)
- Tool-Unterstützung (Actifsource, Rhapsody)

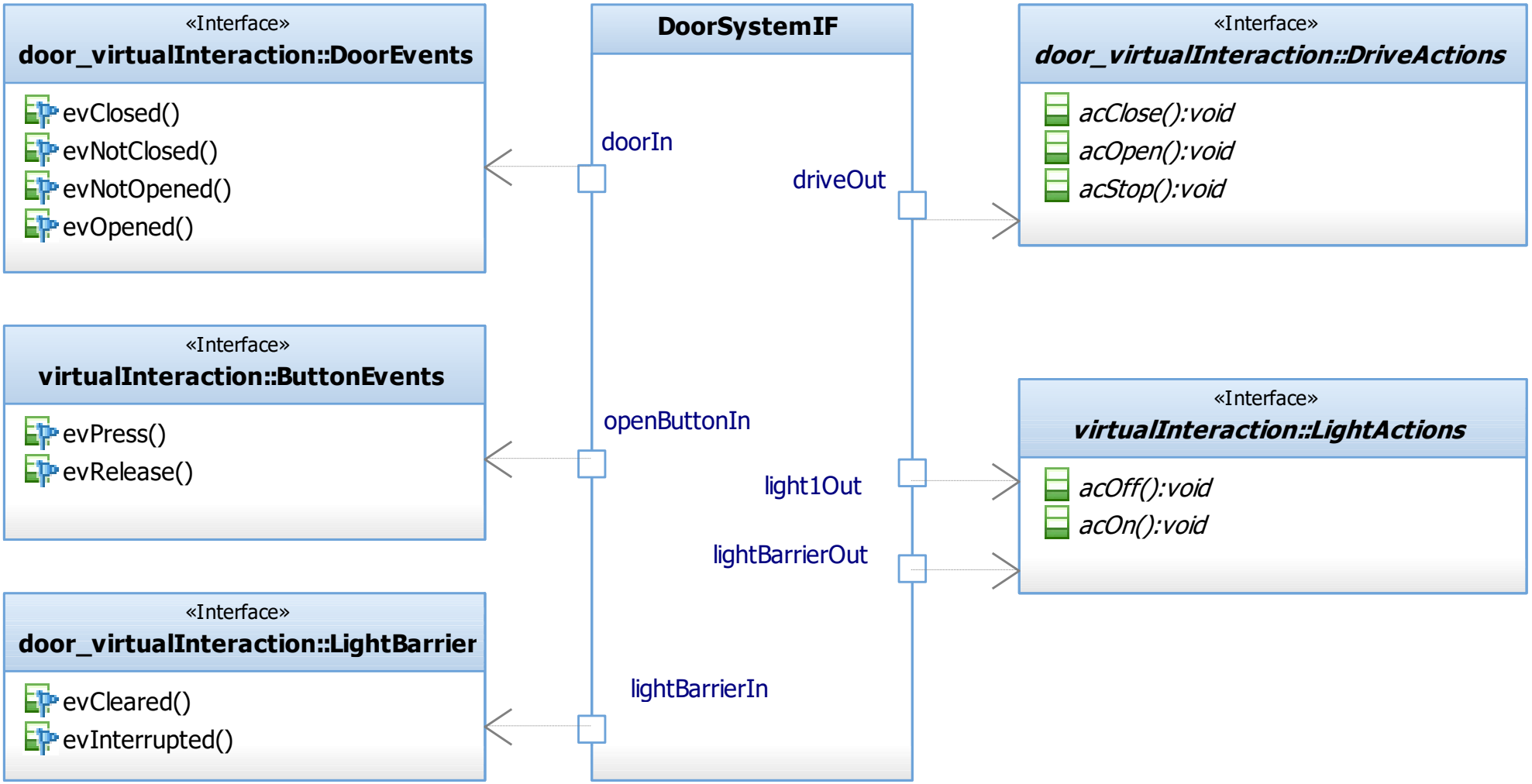
Generische Architektur



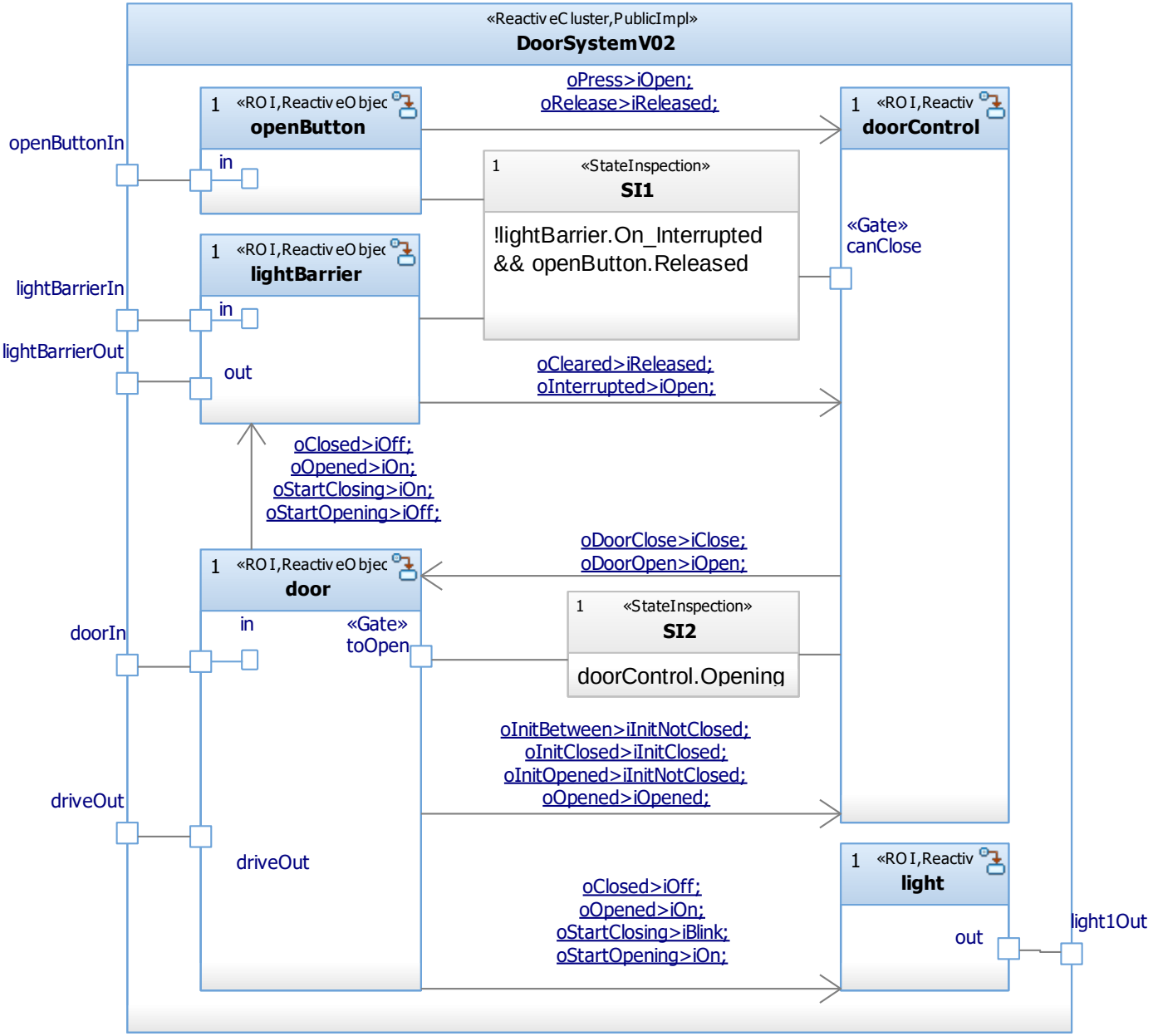
Konkrete Architektur



Reactive Machine Interface (Event- und Action-Messages)



Reactive Machine (Komposition von Reactive Objects)



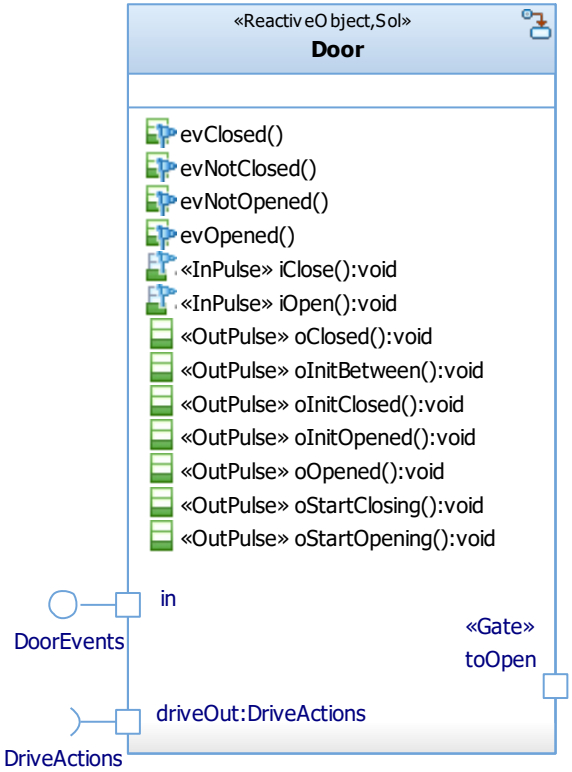
Komponenten

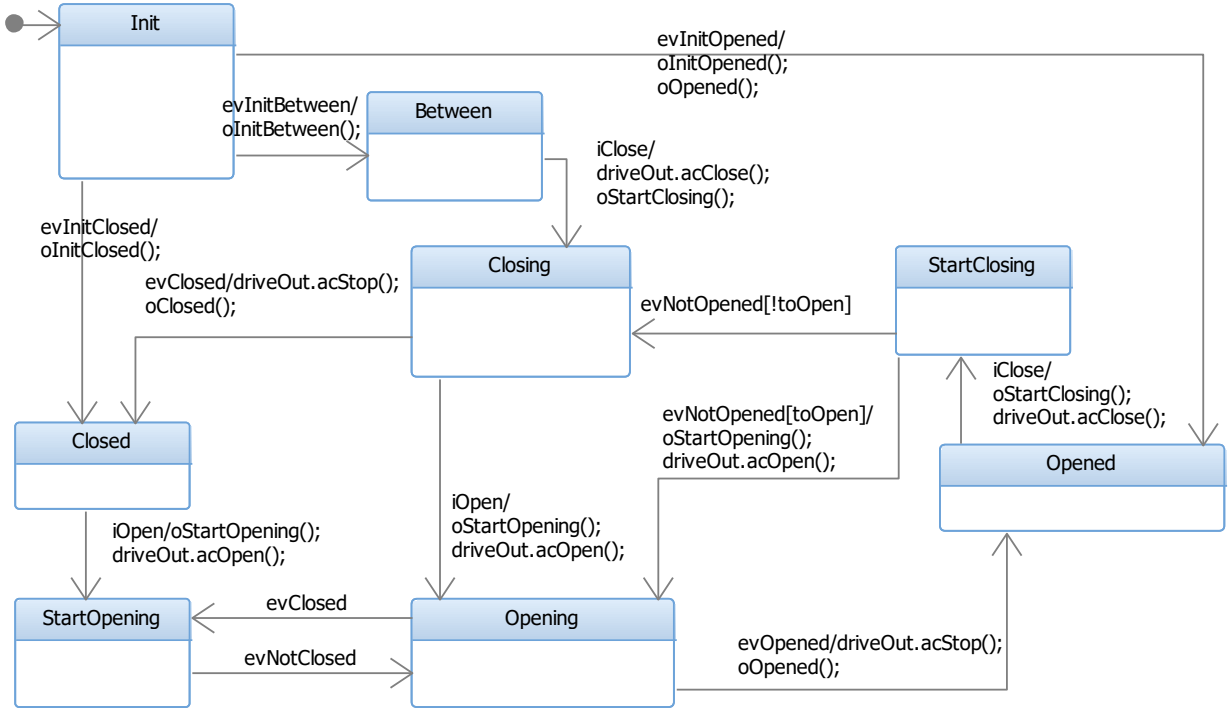
- Struktur und Verhalten durch Klasse definiert
- Public Interfaces
- Private Implementation
- Wissen nichts von den anderen Komponenten

Konnektoren

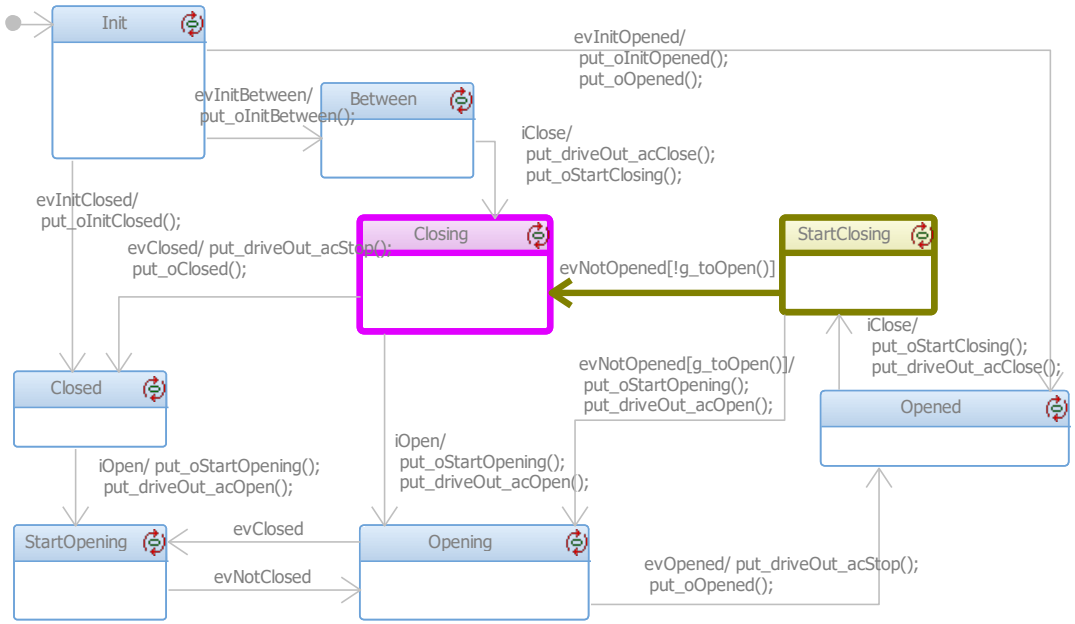
- Gehören zur Komposition
- Definieren die Interaktion zwischen Komponenten
- Interaktion ist synchron

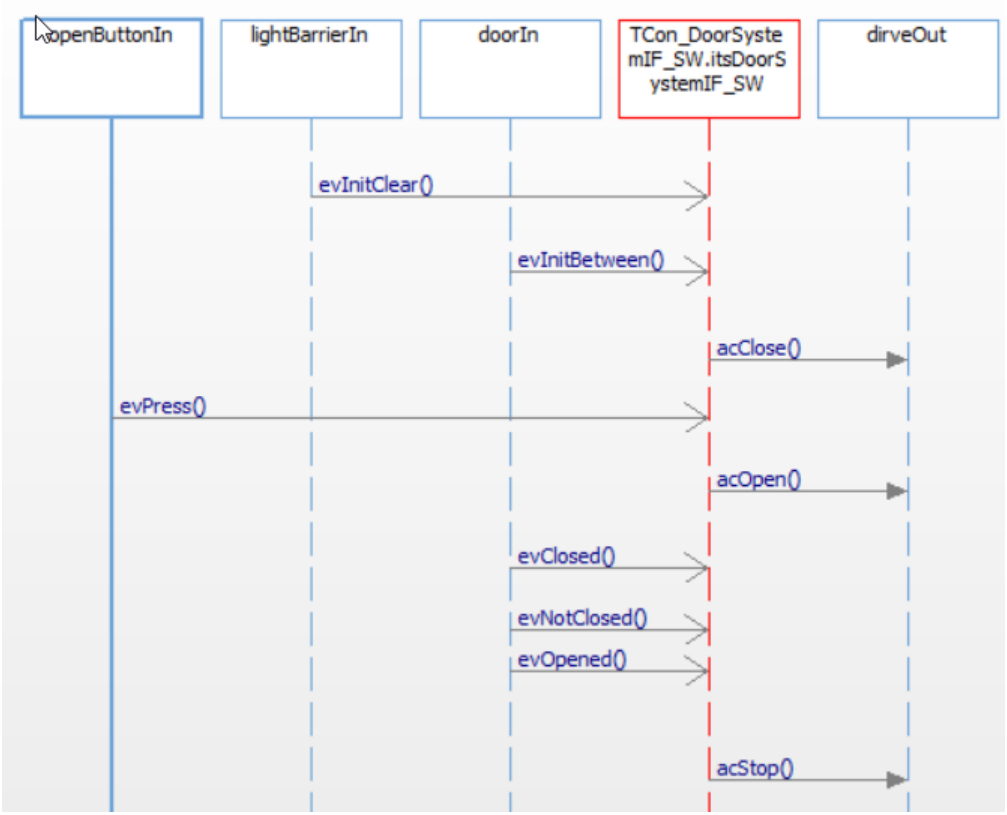
Reactive Object Class



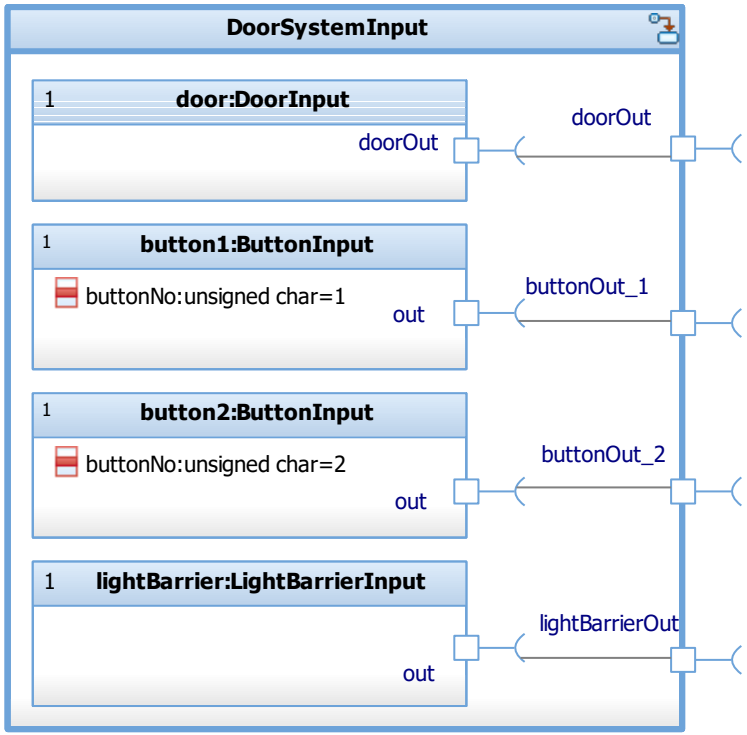


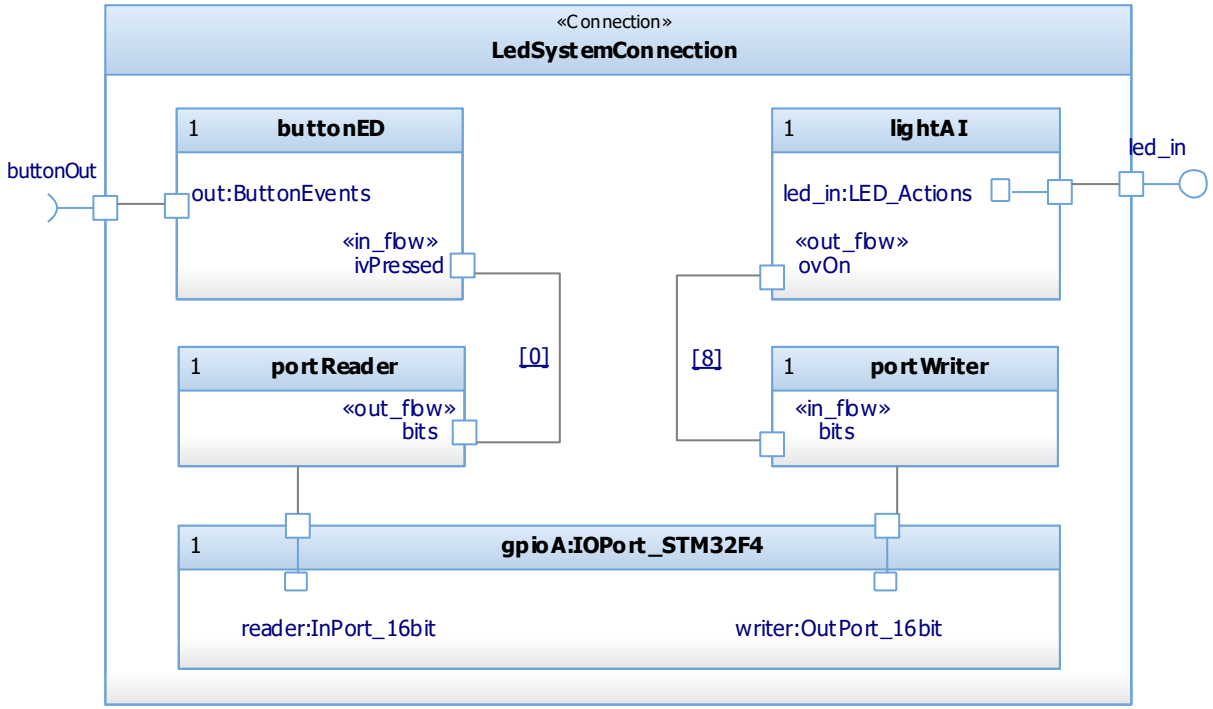
Simulation und Test



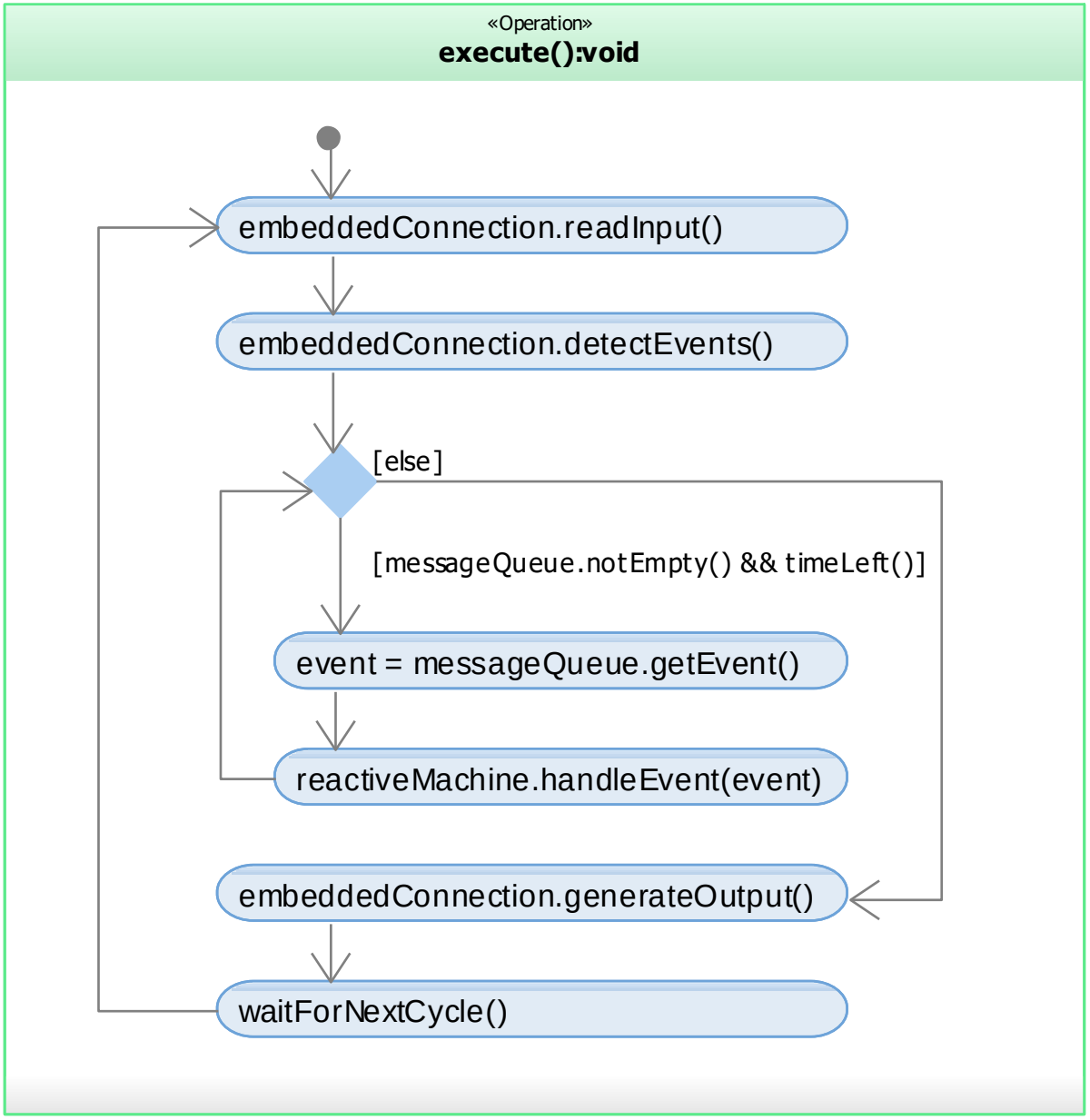


Connection, 2 Beispiele





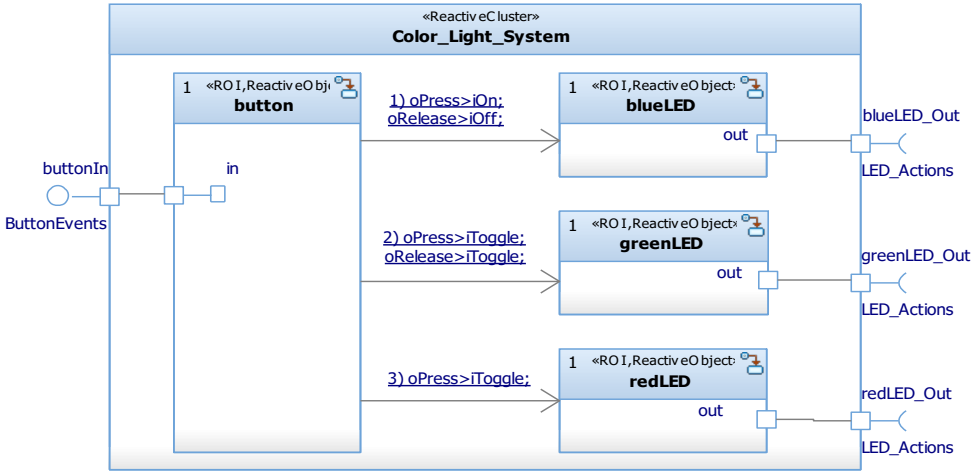
Execution

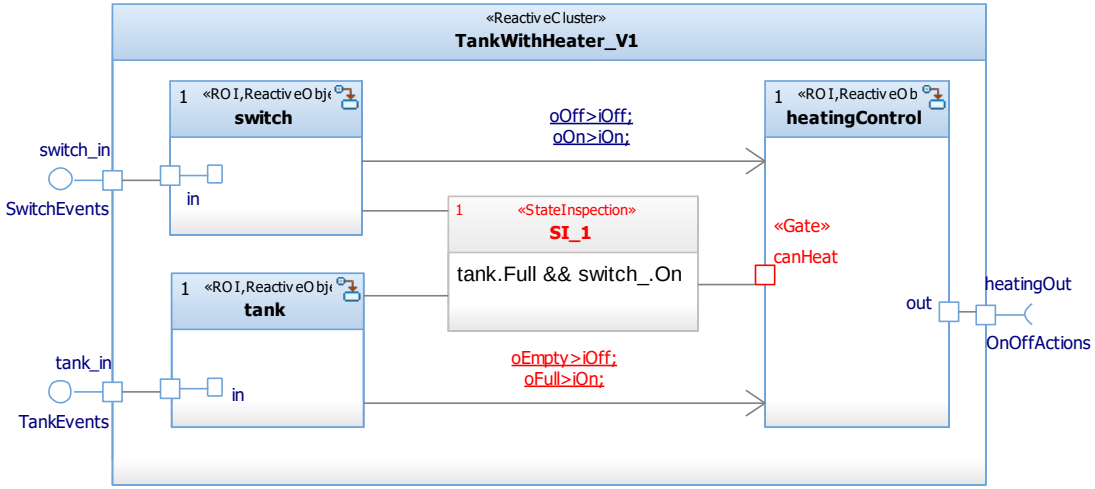

«Operation»
execute():void
embeddedConnection.readInput()
embeddedConnection.detectEvents()
[else]
[messageQueue.notEmpty() && timeLeft()]
event = messageQueue.getEvent()
reactiveMachine.handleEvent(event)
embeddedConnection.generateOutput()
waitForNextCycle()

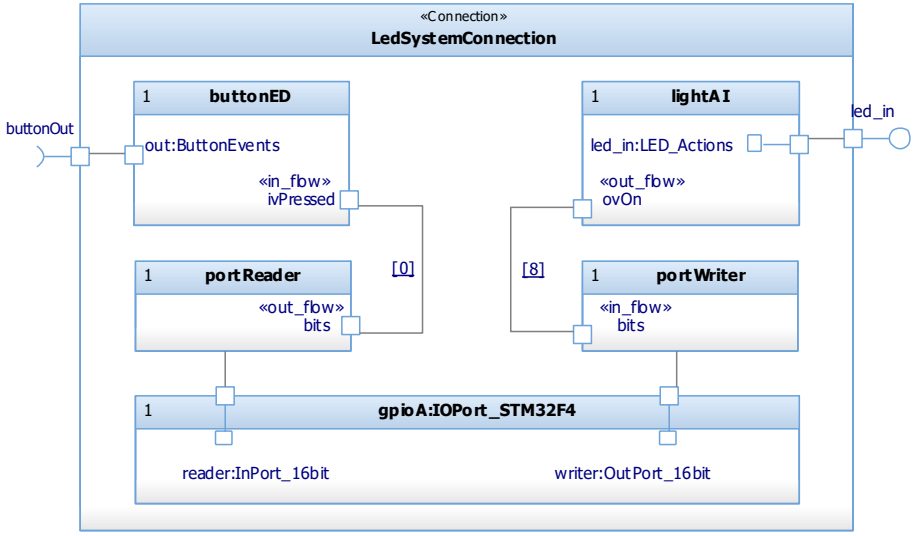
Code Generierung

```
int Door::rootState_dispatchEvent(Rhp_int16_t id) {
    int res = eventNotConsumed;
    switch (rootState_active) {
        // State Closed
        case Closed:
        {
            if(IS_EVENT_TYPE_OF(iOpen_Door_Event_id)==1)
                {
                    //#[ transition 3
                    oStartOpening();
                    driveOut.acOpen();
                    //#]
                    rootState_subState = StartOpening;
                    rootState_active = StartOpening;
                    res = eventConsumed;
                }

        }
        break;
        // State Closing
        case Closing:
        {
            if(IS_EVENT_TYPE_OF(evClosed_door_virtualInteraction_id)==1)
                {
                    //#[ transition 5
                    driveOut.acStop();
                    oClosed();
                    //#]
                    rootState_subState = Closed;
                    rootState_active = Closed;
                    res = eventConsumed;
                }
            else if(IS_EVENT_TYPE_OF(iOpen_Door_Event_id)==1)
                {
                    //#[ transition 9

                    oStartOpening();
                    driveOut.acOpen();
```

UML Erweiterungen, CIRO, Qualifizierte Konnektoren







Qualifizierte Konnektoren

- Definieren die Interaktion
- Erhöht Anwendbarkeit von Komponenten
- Dieses Prinzip lässt sich bei unterschied-lichen Kompositionsarten anwenden
- Vielseitig einsetzbares Konzept
- Erweiterung von UML -> CIRO

Ausblick, Ideen, laufende Entwicklung

Bestreben: Modelle so einfach und übersichtlich wie möglich, so aussagekräftig wie nötig

Modellierung mit CIRO

- Weitere Abstraktionen für GUIs, persistente Daten, Scheduling
- Andere Arten von Interaktion, andere qualifizierte Konnektoren
- Klassenbibliotheken und Patterns für wiederkehrende Problemstellungen
- Weitere Analysen der Modelle (Model-Checking)
- Optimierte Code-Generierung (kleiner, schneller)
- CIRO mit weiteren Tools

Weitere

- Konzepte für weitere Einsatzgebiete verallgemeinern
- CIRO Kurs als Weiterbildung

Zusammenfassend

Modellgetriebene Software-Entwicklung

- Verständlichere, testbare Modelle
- Code konsistent mit Dokumentation
- Hardware unabhängige Modellierung

CIRO

- Generische Software Architektur
- Embedded System als Komposition von wiederverwendbaren Komponenten
- Definition der Interaktion mittels qualifizierter Konnektoren

Fragen?

Bemerkungen?