HUMBOLDT-UNIVERSITÄT ZU BERLIN

Institut für Informatik
Lehr- und Forschungseinheit
Systemanalyse

Systemmodellierung mit SysML

Studienarbeit von
Peer Hausding

Berlin, 9. März 2010

Ausgeführt unter der Leitung von

Prof. Dr. Joachim Fischer Dipl.-Inf. Andreas Blunk

Inhaltsverzeichnis

1 Einführung

Diese Arbeit soll die Sprachdefinition der *Systems Modeling Language* (SysML) [6] untersuchen und anhand von Beispielsystemen sollen fehlende Konzepte für die Systemmodellierung und semantische Ungenauigkeiten aufgezeigt werden.

1.1 Motivation

Die *Unified Modeling Language* (UML) [7] wurde für die modellgetriebene Softwareentwicklung definiert und ermöglicht es verschiedene Sichten auf ein Softwaresystem zu modellieren. Die Sprache bietet bisher nicht die Möglichkeit Modelle von Systemen, wie sie in der Systementwicklung vorkommen, zu beschreiben. Diese werden derzeit mit verschiedenen Modellierungssprachen spezifiziert, wobei jede darauf spezialisiert ist, das System unter einem konkreten Blickwinkel darzustellen. Bei der Betrachtung von Systemensembles und den Relationen zwischen ihnen werden allerdings Beschreibungen benötigt, die es ermöglichen die Systeme darzustellen und miteinander in Bezug zu setzen. Die Sprache SysML ist eine von der *Object Management Group* (OMG) standardisierte Modellierungssprache für die Spezifikation, die Analyse, das Design, und die Verifikation und Validierung von Systemen und deren Systemelemente wie beispielsweise Software, Hardware, Informationen, Prozesse, Personen und Gegenstände. Dabei dient ein Teil der UML als Sprachkern, welcher um zusätzliche Sprachelemente erweitert wurde.
Eine Systemanalyse ist für einige Systeme bereits auf Grundlage der Modelle durchführbar. Viele Systeme lassen sich allerdings erst durch eine Simulation hinsichtlich ihrer Untersuchungskriterien bewerten. Ausgehend von einem Systemmodell wird mit Hilfe von Konfigurationseinstellungen und der Übergabe von Werten an die äußere Schnittstelle des Systems die Veränderung des Systemzustandes im Verlauf der Zeit beobachtet. Die Modellgrößen in einem System, die sich als Zustandsgrößen identifizieren lassen, können sich dabei grundsätzlich zeitkontinuierlich oder zeitdiskret verändern.
Damit ein Systemmodell nicht nur als Systembeschreibung dient, sondern auch eine Simulation des Systems ermöglicht, muss neben der Systemstruktur auch das Systemverhalten spezifiziert werden. Zustandsänderungen können zu diskreten Zeitpunkten oder kontinuierlich vollzogen werden. Der Aspekt Zeit muss als Kontextinformation bereitstehen und für zeitverbauchendes Verhalten erfassbar sein.
Es wird untersucht inwieweit sich SysML für die Domäne Systemmodellierung von Systemen mit zeitdiskretem und zeitkontinuierlichem Verhalten verwenden lässt.

1.2 Zeitdiskrete und zeitkontinuierliche Systeme

Der Begriff „System“ und Dinge, die damit in direkter Beziehung stehen, werden für viele Objekte in unserer Erfahrungswelt verwendet. In dieser Arbeit gelten die nachfolgenden Definition aus „Objektorientierte Prozeßsimulation in C++“ von Joachim Fischer und Klaus Ahrens [9].
Ein Phänomen wird als System bezeichnet, wenn die folgenden Merkmale erfüllt sind.

1. Das Phänomen erfüllt eine bestimmte Funktion (Systemzweck).
2. Das Phänomen besteht aus einer bestimmten Zusammensetzung von Objekten (Systemelemente), die untereinander in Relation (Wirkungsbeziehung) stehen.
3. Das Phänomen verliert seine Identität (Systemidentität), wenn seine Integrität (Systemintegrität) zerstört wird, d. h. das System ist nicht teilbar.

Ein System lässt sich von seiner Umwelt trennen und die Grenze wird als Systemgrenze bezeichnet. Die Umwelt eines Systems nennt man auch Systemumgebung, wobei Ein- oder Auswirkungen zwischen System und Umwelt bestehen können (siehe Abbildung 1.1). Der Zustand eines Systems setzt sich aus den Zuständen seiner

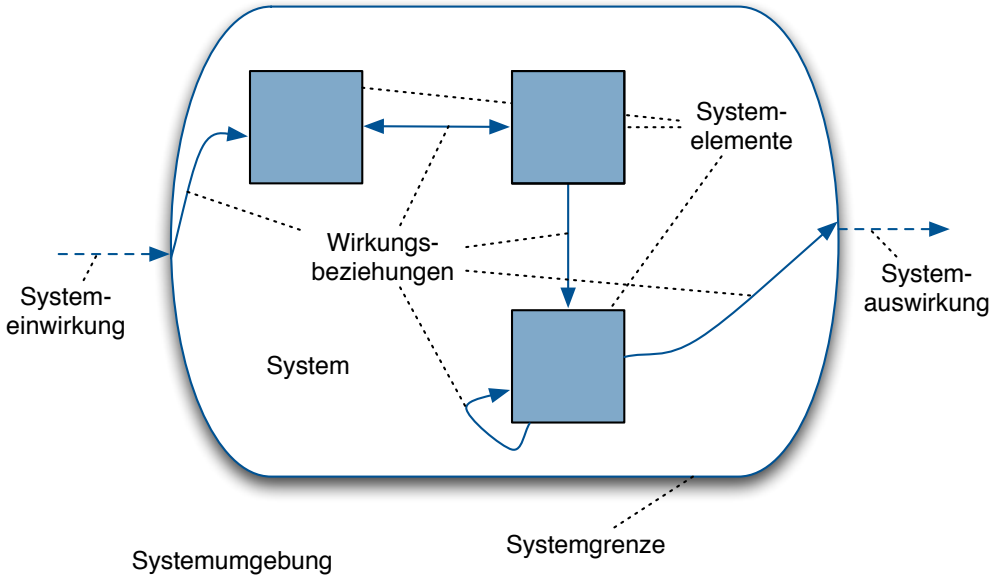


Abbildung 1.1: Bestandteile von Systemen

Systemelemente zusammen. Zustandsgrößen sind voneinander unabhängig und definieren zu jedem Zeitpunkt den Zustand eines System. Das Systemverhalten ergibt sich aus dem Verhalten der Systemelemente und deren Interaktion (untereinander

und mit der Systemumgebung). Die Systemidentität ist durch die Systemstruktur und ihrer Funktionalität bestimmt.
Systeme sind dynamischer Natur, allerdings kann die zeitliche Zustandsveränderung auch durch Alterungserscheinungen oder Ähnlichem erfolgen. Als dynamische Systeme sollen hier aber nur solche bezeichnet werden, die zeitliche Zustandsänderungen in einen für die Systemanalyse interessanten Zeitraum vollziehen. Die Zustandsänderung kann zeitdiskret oder zeitkontinuierlich erfolgen. Prinzipiell kann eine Zustandsänderung durch zwei Ursachen ausgelöst werden. Zum einen können Einwirkungen aus der Systemumwelt und zum anderen können Rückkopplungseffekte der Systemzustände Zustandsänderungen bewirken. Eine zeitkontinuierliche Zustandsänderung einer Zustandsgröße wird mit einer Funktion in Abhängigkeit der Zeit beschrieben, der Zustandsänderungsrate. Aus dem aktuellen Zustand und der Zustandsänderungsrate ergibt sich der Zustand für eine Zustandsgröße zu einem Zeitpunkt.

1.3 Modellbasiertes Systems Engineering

Modellbasiertes Systems Engineering (MBSE) wird als die Modellierung eines Systems im Systementwicklungsprozess verstanden und soll die Analyse, die Spezifikation, das Design und die Verifikation und Validierung anhand eines Systemmodells durchgängig unterstützen. Daneben gibt es das dokumentenbasierte Systems Engineering, welches als Basis für die verschiedenen Prozesse eine Menge von textuellen Spezifikationen besitzt.
„Das Systems Engineering konzentriert sich auf die Definition und Dokumentation der Systemanforderungen in der frühen Entwicklungsphase, die Erarbeitung eines Systemdesigns und die Überprüfung des Systems auf Einhaltung der gestellten Anforderungen unter Berücksichtigung des Gesamtproblems: Betrieb, Zeit, Test, Erstellung, Kosten & Planung, Training & Support und Entsorgung“ (Tim Weilkiens, „Systems Engineering mit SysML/UML“ [11]). Diese Definition ähnelt der der International Council on Systems Engineering (INCOSE) bzw. der Gesellschaft für Systems Engineering (GfSE, German Chapter of INCOSE), einer Organisation, die wissenschaftliche und industrielle Einrichtungen zum Thema „Systems Engineering“ vereint, „Systems Engineering is an interdisciplinary approach and means to enable the realisation of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem“.
Ein Vorteil der modellbasierten Entwicklung ist die automatisierte Weiterverarbeitung der erstellten Modelle. Die Modelle des Systems, welche mit Hilfe eines Model-Repositorys versioniert und verwaltet werden können, können in den Entwicklungsphasen auf verschiedene Aspekte untersucht werden.
Ein Modell ist eine Repräsentation eines realen oder gedachten Systems und beschreibt dieses in Hinblick auf ein konkretes Untersuchungsziel. Es abstrahiert das

reale System und blendet die für das Untersuchungsziel unwesentlichen Elemente, d. h. solche die keine Relevanz auf die Untersuchung haben, aus. Bisher ist es nur möglich von einander unabhängige Teilmodelle des Systems mit verschiedenen Modellierungssprachen zu erstellen. Es fehlt die Modellzusammenführung, damit diese unterschiedlichen Modelle in einem Systemmodell integriert werden können.

1.4 Aufbau der Arbeit

In den folgenden Kapiteln wird die Systems Modeling Language untersucht, wobei die Definition der Sprache und Ihrer Elemente hinsichtlich Ihrer Anwendbarkeit, um zeitdiskrete, zeitkontinuierliche und kombinierte Systeme zu modellieren, im Vordergrund stehen soll.
Das zweite Kapitel beschreibt die Konzepte der Sprache SysML und erklärt als erstes den Profilmechanismus der UML, welcher für die Definition von SysML benutzt wurde. Anschließend wird näher auf die konkreten Sprachkonzepte eingegangen und es werden die verschiedenen Aspekte der Modellierung, die SysML bietet, beschrieben. Das Kapitel bietet jedoch nur einen Ausschnitt der SysML in Hinblick auf die in Kapitel drei modellierten Beispielsysteme. Der letzte Abschnitt behandelt die Möglichkeit der Festlegung zusätzlicher Verbindungen von Modellelementen in den verschiedenen Modellen. Anhand derer können Verbindungen von Modellelementen in den Modellen geschaffen werden, welche die Sprache anhand der vorhandenen Konstrukte nicht bietet.
Im dritten Kapitel wird zunächst die Modellierung von Systemen (allgemein) und die Festlegung von Systemkonfigurationen in SysML beschrieben. Die nachfolgenden Abschnitte zeigen Beispielsysteme, die zum einen ausschließlich zeitdiskretes und zeitkoninuierliches und zum anderen kombiniertes Verhalten besitzen. Es werden die Probleme, die bei einer vollständigen Systembeschreibung mit SysML als UML-Profil auftreten gezeigt und mögliche Lösungen vorgestellt.
Im letzten Kapitel wird zunächst ein Resümee der Ergebnisse der Systemmodellierung aus Kapitel drei gegeben und nachfolgend erfolgt ein Ausblick auf die Definition eines Simulationsprofils, welches ebenfalls als UML-Profil definiert in Verbindung mit SysML für eine Systemsimulation genutzt werden könnte.

1.4.1 Anmerkung

Für Bezeichner von Modellelementen der SysML-Modelle wird im Text der `Schreibmaschinenstil` verwendet. Stereotypen werden ebenfalls in diesem Stil dargestellt und zusätzlich mit den französischen Anführungszeichen (`«...»`; Guillemets) mit Spitzen nach außen versehen.

2 Systems Modeling Language (SysML)

Die OMG hat in Zusammenarbeit mit dem *International Council on Systems Engineering* (INCOSE) im September 2001 Anforderungen, für die Definition einer standardisierten Erweiterung der UML als Modellierungssprache zur Spezifikation, Design und Verifikation von komplexen Systemen, formuliert. Die SysML entstand als ein Profil für die UML aus diesen Anforderungen und erweitert die Konzepte der UML für die Modellierung von Systemen.
Nachfolgend wird zunächst der UML-Profilmechanismus und anschließend die im SysML-Profil definierten Konzepte erklärt.

2.1 Sprachdefinition mit Hilfe eines UML-Profils

Die *Meta Object Facility* (MOF) [4] wurde von der OMG definiert, um Metamodelle und damit unter anderem die Syntax von Sprachen zu beschreiben. Diese wurde benutzt um das Metamodell der UML zu erstellen, welches wiederum die Möglichkeit der Erstellung von Profilen für die UML schafft.
Der Profil-Mechanismus, der in der Spezifikation der UML beschrieben wird, ermöglicht es das UML-Metamodell plattform- oder domänenspezifisch zuzuschneiden ohne neue Metaklassen zu definieren. Daher wird dieser Mechanismus auch leichtgewichteter Erweiterungsmechanismus genannt. Im Gegensatz zu einer Erweiterung des Metamodells durch den Profilmechanismus könnte das Metamodell mit neuen Metaklassen erweitert oder ein neues Metamodell erstellt werden. Mit Hilfe dieses Mechanismus kann die Sprache mit weiteren Vokabular, zusätzlicher oder einschränkender Semantik (für einen semantischen Variationspunkt), Ergänzung von konkreter Syntax und zusätzlichen Constraints an Metaklassen angepasst werden (mit Hilfe der *Object Constraint Language* (OCL) [5]). Eine Erweiterung des Metamodells um neue Metaklassen würde neue Konzepte zu der Sprache hinzufügen und nur dann benötigt werden, wenn die bisherigen Konzepte der Sprache nicht ausreichen würden, um die Sprache für eine konkrete Domäne zu benutzen.

2.1.1 Stereotyp

Die Abbildung 2.1 zeigt einen Ausschnitt des Paketes `Profile` aus dem UML-Metamodell. In den folgenden Abschnitten wird näher auf die Semantik der Metaklassen `Stereotype`, `Profile` und `ProfileApplication` und `Extension` und `ExtensionEnd` eingegangen. Die UML-Metaklasse `Stereotype` ermöglicht es einen

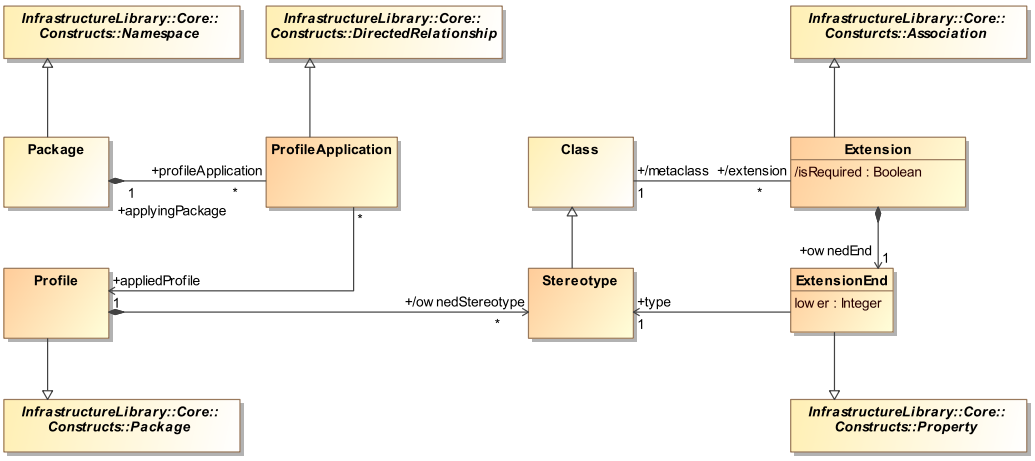


Abbildung 2.1: Ausschnitt UML 2.2 Metamodell Package Profile

neuen Begriff einzuführen, wobei ein Stereotyp eine Erweiterungsbeziehung (Metaklasse `Extension`) zu einer bestehenden Metaklasse (Instanz einer Meta-Metaklasse) besitzt. Es können somit vorhandene Konzepte wie beispielsweise die Beschreibung von diskreten Verhalten mit Aktivitäten für zeitkontinuierliches Verhalten erweitert werden. Eine Aktivität kann beispielsweise mit einer ausgehenden Objektflusskante, die das Stereotyp `«Continuous»` zugeordnet bekommt, einen kontinuierlichen Fluss von Objekten bewirken (Zeit zwischen zwei Objekten ist nahezu null).
Vorhandene Stereotypen können wiederum nicht mit einem Stereotyp erweitert, sondern nur über eine Generalisierung spezialisiert werden. Die Benutzung eines definierten Stereotyps bzw. der neue Begriff kann im Modell mit einem Symbol und dem Namen in Guillemets dargestellt werden, wobei zu beachten ist, das auch die UML-Metaklasse `Stereotype` solche besitzt. `Stereotype` ist kein Stereotyp, trotz der gleichen konkreten Syntax. Da `Stereotype` eine Spezialisierung von `Class` ist, kann dieses auch Propertys besitzen. Die Anwendung eines Stereotypen besitzt in diesem Fall für jedes Property ein sogenanntes Tagged-Value. Des Weiteren können zusätzliche Constraints für Metamodell-Instanzen und damit auch für ein Stereotypen festgelegt werden.

2.1.2 Profil

Ein Profil ist ein spezielles Paket, welches Stereotypen und Modellbibliotheken enthalten kann. Diese sind jedoch erst nach Anwendung des Profils auf ein Paket in einem UML-Modell benutzbar. Für diesen Zweck muss die Zuordnungsbeziehung (`ProfileApplication`) benutzt werden. Diese besondere Beziehung überschreitet das Metalevel von Modell nach Metamodel. Die im Profil definierten Stereotypen können somit über die Anwendung der Erweiterungsbeziehung mit einer Instanze einer Metaklassen verwendet werden. In der Zuordnungsbeziehung wird dabei die Informatione über die Zuordnung des benutzen Profils im Paket festgehalten.

2.1.3 Erweiterung

Die Erweiterung (`Extension`) ist die im Profil definierte Beziehung zwischen Metaklasse und Stereotyp. In `ExtensionEnd` wird diese Erweiterung dann an einen konkreten Stereotypen gebunden. Die Erweiterungsbeziehung besitzt das Property `isRequired`, welches angibt, ob für eine Instanz der Metaklasse der Stereotype oder eine Spezialisierung zwingend verwendet werden muss.

2.1.4 Definition und Anwendung von Stereotypen

UML-Profile werden in einem UML-Modell definiert und im gleichen Metalevel angewendet. Die Definition des SysML-Stereotypen `Block` ist in Abbildung 2.2 ausschnittsweise dargestellt und zeigt gleichzeitig die Anwendung des Stereotypen auf die Klasse System, einer Instanz der Metaklasse `Class` (also einer Klassendefinition). Für das Property `isEncapsulated` besitzt die Instanz des Stereotyps ein Tagged-Value mit dem Wert `false`.

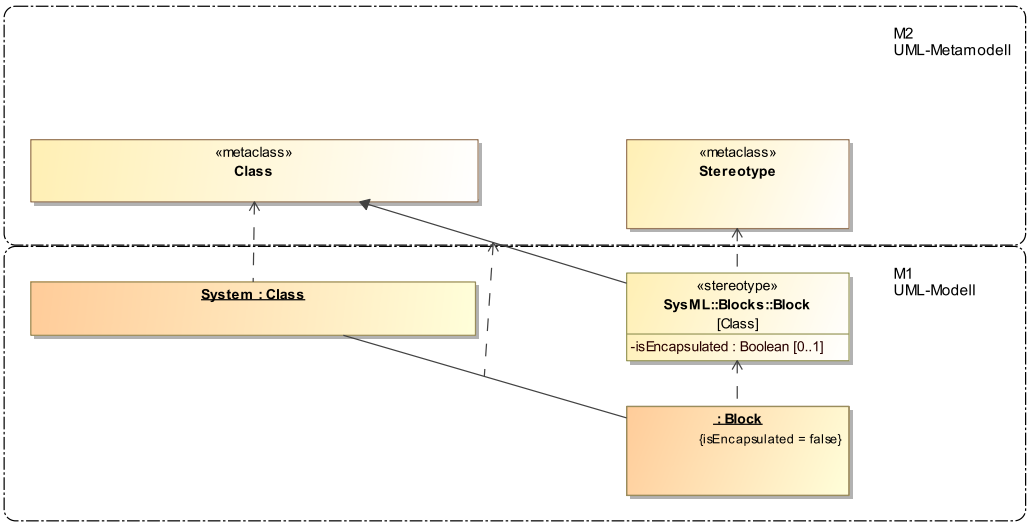


Abbildung 2.2: Definition und Anwendung eines Stereotyps

2.1.5 Vergleich mit Java-Annotationen

Die in Java 5 eingeführten *Java Annotations* ähneln in Ihrer Definition und Anwendung den Stereotypen aus UML. Die Definition des Konzepts Annotation ist in der Java-Sprachdefinition beschrieben. In einem Java-Programm kann man somit beispielsweise für Klassen, Felder und Methoden Annotationen zunächst definieren und verwenden. Sie besitzen genau wie die Propertys der Stereotypen die Möglichkeit Attribute zu definieren. Der Unterschied zwischen Java-Annotationen und UML-Stereotypen besteht in ihrer Definition auf Elemente der Sprache. UML-Stereotypen

lassen sich für beliebige Metaklassen definieren (mit beliebig komplexen Propertys), Java-Anntotationen hingegen nicht.
Im Listing 2.1 wird in dem Paket `SysML.Blocks` die Annotation `Block` mit dem Attribut `isEncapculated` eingeführt (Definition der Annotation) und darunter über der Klassendefinition wird diese Annotation verwendet und ihr Attribut mit dem Wert `false` belegt (Anwendung der Annotation). Dies entspricht der Stereotyp-Definition und Anwendung aus Abbildung 2.2.

```
1 package SysML.Blocks;
2
3 public @interface Block {
4     boolean isEncapsulated();
5 }
6
7 @Block (isEncapsulated = false)
8 class System {
9 }
```

Listing 2.1: Java Annotation: SysML.Blocks.Block

2.2 Konzepte der Sprache SysML

Das SysML-Profil beinhaltet zum einen in Paketen definierte Stereotypen, die Klassen des UML-Metamodels erweitern, Modellbibliotheken (UML-Stereotyp «`modelLibrary`»), die für die Systementwicklung notwendige Modellelemente bereit halten und Diagrammerweiterungen. Das Profil enthält die Pakete `Blocks`, `Activities`, `Requierements`, `ConstraintBlocks`, `Ports&Flows`, `Allocations` und `ModelElements`. Die Spezifikation definiert außerdem die konkrete Notation der eingeführten Stereotypen in den Diagrammarten. Mit Hilfe der Struktur-, Verhaltens-, Anforderungs- und Parametermodellierung können verschiedene Aspekte eines Systems mit SysML modelliert werden.

2.2.1 Strukturmodellierung

Die Strukturmodellierung dient der Modellierung eines Systems hinsichtlich seiner strukturellen Eigenschaften. Die möglichen Systemelemente in einem realen System werden beschrieben und der strukturelle Aufbau eines Elementes und Verbindungen und Vererbungshierarchien zwischen ihnen werden festgelegt. Im Modell werden die Typen der Systemelemente beschrieben und die Modellierung von Instanzen könnte als eine Systemkonfiguration interpretiert werden (siehe Abschnitt 3.2), um somit die initialen Systemelemente festzulegen. Die Instanzen von Systemelementtypen besitzen Identitäten anhand derer sie sich eindeutig von einander unterscheiden.

Wertetyp

Wertetypen («`ValueType`») sind Typen, deren Instanzen keine Identität besitzen. «`ValueType`» ist ein Stereotyp der UML-Metaklasse `DataType`. SysML definiert,

neben den aus UML bereits vorhandenen Typen (Integer, String und Boolean) Real und Complex. Zusätzlich besitzt `ValueType` die Value-Propertys `quantityKind` und `unit`. Mit Hilfe dieser können physikalische Größen und deren Einheiten festgelegt werden. Im Anhang der Spezifikation von SysML befindet sich eine Beispiel-Modellbibliothek für *das Internationale Einheitensystem* (SI).

Block

Modulare Einheiten in der Systembeschreibung können mit einem Block (`Block`) modelliert werden. Ein Block definiert eine Menge von Eigenschaften und repräsentiert eine Systemelementtyp-Definition. Die Eigenschaften können sowohl Struktur (Property) als auch Verhalten (Operation) sein. Ein Block kann auch eine Blackbox definieren, deren Eigenschaften nicht verfügbar sind, sondern nur die Schnittstellen des Blocks als Interaktionspunkte mit dem restlichen System bekannt sind. Hierfür gibt es das Property `isEncapsulated`. Das SysML-Profil definiert `«Block»`, einen Stereotyp für die UML-Metaklasse `Class`.

Property

Ein Property beschreibt eine strukturelle Eigenschaft eines Blocks. Ein Property gehört einer der nachfolgenden Kategorien an:

- Part-Property
  Ein Property bezeichnet man als Part-Property, wenn es eine Kompositionsbeziehung zu einem anderen `Classifier` definiert.

- Reference-Property
  Besteht zwischen dem Property und dem Block jedoch eine Aggregationsbeziehung, spricht man von einem Reference-Property.

- Value-Property
  Wenn es sich bei dem Property um die Verwendung eines Wertetypen handelt, nennt man es Value-Property.

- Constraint-Property
  Ein Property kann auch die Verwendung eines Constraint-Blocks sein und ist dann ein Constraint-Property (siehe Abschnitt 2.2.4).

Distributed-Property

SysML definiert zusätzlich das Stereotype `«DistributedProperty»` für die UML-Metaklasse `Property`. Die Werte eines Distributed-Propertys werden dann anhand der festgelegten Wahrscheinlichkeitsverteilung ermittelt.

Port-Property

Ein Port ist ein Interaktionspunkt für einen Block (UML-Metaklasse `Port`). SysML definiert die Erweiterung `«FlowPort»`. Ein Flow-Port dient dem Austausch von Elementen und kann atomar sein und nur einen Interaktionstyp und -richtung festlegen oder mit einer Flussspezifikation (`«FlowSpezification»`) verschiedene Interaktionstypen und -richtungen definieren. Die Angabe des Typs des Flow-Propertys spezifiziert den Interaktionstyp und die -richtung. Falls der Typ eine Interface-Definition ist, können mit dem Stereotyp (`«FlowSpecification»`) für die UML-Metaklasse `Interface` mehrere Interaktionstypen mit -richtung spezifiziert werden. Eine Flussspezifikation besitzt hierfür besondere Propertys, die Flow-Propertys (`«FlowProper- ty»`). Ein Flow-Property hat das Property `direction`, welches die Richtung (`in`, `out` oder `inout`) festlegt. Da `«FlowProperty»` ein Stereotyp der UML-Metaklasse `Property` ist, ist der Typ des Propertys, der Typ der an diesem Interaktionspunkt ausgetauscht werden kann.

Operation

Ein Block kann auch Operationen besitzen, die durch die Verwendung von Verhaltensdefinitionen Verhalten eines Blockes beschreiben können. Verhaltensdefinitionen können in SysML genau wie in UML verschiedenen modelliert werden (siehe Abschnitt 2.2.2).

Notation

Für die grafische Notation von Blöcken und Wertetypen gibt es zwei auf UML-basierende Diagrammarten. Mit diesen ist es möglich die Beziehungen zwischen den Blöcken und die Struktur eines Blockes oder Wertetyps darzustellen.

- Blockdefinitionsdiagramm (BDD)
  Das Blockdefinitionsdiagramm basiert auf dem UML-Klassendiagramm und ermöglicht es Block- und Wertetyp-Definitionen und die Assoziationen zwischen Blöcken und Propertys darzustellen. Es ist damit eine Modellsicht auf die in einem Paket oder Block vorkommenden Elemente.

- Internes Blockdiagramm (IBD)
  Das Interne Blockdiagramm beruht auf dem UML-Kompositionsstrukturdiagramm und zeigt die Kompositionsstruktur eines Blockes. Es bietet daher die Modellsicht auf die Struktur eines Blockes und zeigt die Verwendung von Typdefinitionen als strukturelle Eigenschaften eines Blocks.

2.2.2 Verhaltensmodellierung

Zeitdiskretes und zeitkontinuierliches Verhalten

Die UML bietet per Definition die Möglichkeit diskretes, ereignisbasiertes Verhalten zu modellieren. In SysML wird die Modellierung von Aktivitäten (`Activities`),

Interaktionen (`Interactions`), Zustandsmaschinen (`StateMachines`) und Anwendungsfällen (`UseCases`) übernommen. Zusätzlich definiert die Sprache Erweiterungen, mit denen zeitkontinuierliche Veränderungen des Systems beschrieben werden können. Die Erweiterungen ermöglichen es einen kontinuierlichen (`«Continuous»`, Spezialisierung von `«Rate»`) oder diskreten (`«Discrete»`, Spezialisierung von `«Rate»`) Fluss von Informationen oder Objekten im System festzulegen, wobei folgende Sprachkonstrukte betroffen sind.

- Flussraten von Entitäten an Kanten von Aktivitäten oder von Input-/Output-Parametern von Verhalten können durch mathematische Modelle beschrieben werden (`«Rate»`).
- Objektknoten (inklusive Pins) können neue Werte kontinuierlich übernehmen (`«Overwrite»`, vorhandene Werte werden ersetzt) oder sie können Werte verwerfen (`«NoBuffer»`), falls diese nicht sofort weiterverarbeitet werden können.

Wahrscheinlichkeit

Für die Verwendung von Wahrscheinlichkeiten (`«Probability»`) in Aktivitäten bietet SysML folgende Erweiterungen.

- Ausgehende Kanten von Entscheidungen oder Objektknoten könne mit einer Wahrscheinlichkeit versehen werden, die angibt, mit welcher Wahrscheinlichkeit ein Token die Kante traversiert.
- Output-Parameter-Sets können mit Wahrscheinlichkeiten versehen werden, um anzugeben, mit welcher Wahrscheinlichkeit Token an einem Output-Parameter-Set bereitstehen.

Kontrolloperator

Aktivitäten, die Werte von Input- und Output-Pins kontinuierlich verarbeiten, können mit einen Kontrolltoken (`«ControlValue»`) gesteuert werden, welches die Werte `enable` und `disable` annehmen kann. Kontrolloperatoren (`«ControlOperator»`) können an einem Output-Pin Kontrolltoken bereitstellen.

2.2.3 Anforderungsmodellierung

In SysML sind Stereotypen definiert, die es ermöglichen textuelle Anforderungen mit Modellelementen zu verbinden. Da die Anforderungsmodellierung eine untergeordnete Rolle bei der Systemanalyse spielt, soll sie hier nicht betrachtet werden.

2.2.4 Parametermodellierung

Die Propertys von Blöcken können in einem Zusammenhang stehen, wobei die Werte der Propertys sich aus der Auswertung formulierter Constraints ergeben können. In einem System kann es sein, dass sich die Werte von Systemelementen gegenseitig beeinflussen, wobei sich ein Wert im Verlauf der Zeit verändern kann.

Constraint-Block

Das Constraint-Property ist die Verwendung einer speziellen Blockdefinition (`«ConstraintBlock»`). Die Propertys eines Constraint-Blocks werden in diesem Zusammenhang als Parameter bezeichnet. In einem Constraint-Block können Constraints über seine Parameter definiert werden. Für die Definition von Constraints ist in SysML keine Sprache vorgegeben und je nach Anwendungsdomäne kann eine passende ausgewählt werden.

Bindungskonnektor

Die Verbindung zwischen den Parametern von verwendeten Constraint-Blöcken und den Propertys von Blöcken kann mit Hilfe des in UML definierten Konnektor geschehen. Ein Konnektor ist dabei eine in einem Classifier definierte Verbindung zwischen zwei Parts des Classifiers. SysML bietet einen Stereotyp für die UML-Metaklasse `Connector`, den Bindungskonnektor (`«BindingConnector»`). Sind die Propertys, die mit einen Bindungskonnektor verbunden sind, Value-Propertys, dann ist sichergestellt, dass die Werte gleich sind. Falls es sich um Instanzen von Blöcken handelt, so referenzieren beide die gleiche Blockinstanz.

Notation

Constraint-Blöcke können in einem Blockdefinitionsdiagramm angezeigt werden. Damit die Constraint-Propertys und ihre Verbindungen dargestellt werden können, gibt es in SysML eine Erweiterung des Internen Blockdiagramm, das Parameterdiagramm. Hier können die Verbindung zwischen den Constraint-Propertys und die Bindungskonnektoren zwischen den Parametern und Propertys angezeigt werden.

2.3 Zuteilungsbeziehung

`«Allocate»` ist eine Erweiterung der UML-Metaklasse `Abstraction`. Dieser Mechanismus erlaubt es Zuteilungsverbindungen zwischen Elementen (UML-Metaklasse `NamedElement`) verschiedenen Typs oder in verschiedenen Abstraktionsleveln zu definieren (`«Allocated»`). Es ist eine gerichtete Beziehung, wobei das eine Ende (`allocatedFrom`) eines mit dem Stereotyp `«Allocated»` verwendeten Elements mit einem anderen Ende (`allocatedTo`) eines anderen Elements zugeteilt wird.

2.3.1 Verhaltenszuteilung

Parts in Blöcken können Aktionen zugeteilt werden. Wird eine Blockdefinition in einer anderen verwendet kann somit Verhalten zugeteilt werden, dass nur im Kontext der Blockdefinition eine Verbindung mit dem Part besitzt. Des Weiteren können Blockdefinitionen mit Aktivitätsdefinitionen verbunden werden.

2.3.2 Flusszuteilung

Ein Fluss von Informationen oder Objekten kann, wie bereits erwähnt, sowohl in der Strukturmodellierung als auch in der Verhaltensmodellierung beschrieben werden. Um einen Zusammenhang zwischen den modellierten Flüssen herzustellen, ist es möglich einen Objektfluss in Aktivitäten zu einen `Connector` in einem Block zu zuteilen (oder anders herum). Es wird festgelegt über welche Verbindung der Fluss in der Aktivität in dem Block stattfinden muss.

3 SysML-Modelle ausgewählter Systeme

In diesem Kapitel sollen drei Systeme (mit zeitdiskreten, zeitkontinuierlichen und kombinierten Verhalten) mit Hilfe der Sprache SysML beschrieben werden.
Der erste Abschnitt beschäftigt sich mit Modellierung von Systemen und der Festlegung einer Systemkonfiguration in SysML. Die drei nachfolgenden Abschnitte beinhalten zunächst jeweils ein Wort- und ein SysML-Modell eines Systems. Die drei Systeme unterscheiden sich in ihren Verhaltensbeschreibungen. Das Autofähre-System (Abschnitt 3.3) besitzt ausschließlich zeitdiskretes Verhalten, das Badewannen-System (Abschnitt 3.4) ausschließlich zeitkontinuierliches Verhalten und das Barrenofen-System (Abschnitt 3.5) besitzt sowohl zeitdiskretes als auch zeitkonitnuierliches Verhalten (kombiniert).
Die Sprache SysML beinhaltet genau wie die Sprache UML semantische Variationspunkt und Interpretationsmöglichkeiten, deren Auftreten in den einzelnen Systemmodellen verdeutlicht und mögliche Lösungen vorgeschlagen werden.

3.1 SysML-Systemmodellierung

Die SysML-Modelle sind objektorientierte Modelle, die eine Instanziierung von Objekten anhand von Typdefinitionen ermöglichen. Typdefinitionen beinhalten sowohl Attribute als auch Verhaltensbeschreibungen (siehe Abschnitt 2.2.1). Es gibt Wertetyp- und Block-Definitionen.
Nur die Typdefinition eines Blocks kann jedoch mit Verhalten assoziiert werden. Da Block ein Stereotyp für die UML-Metaklasse `Class` ist und `Class` von `BehavioredClassifier` erbt, ist es möglich, das Attribut `classifierBehavior` mit einer Verhaltensbeschreibung (UML-Metaklasse `Behavior`) zu belegen, welches von instanziierten Objekten dieses Typs als Objektverhalten ausgeführt werden kann. Auf diese Weise können Werte von Modellbeschreibungsgrößen verändert werden (oder als Resultat der Interaktion mit anderen Objekten).
Die Attribute, aus denen sich zu jeder Zeit der Zustand des Objektes vollständig ergibt, nennt man Zustandsattribute. Der Zustand des Systems ergibt sich somit aus der Gesamtheit aller Objektzustände (siehe Abschnitt 1.2).

3.2 SysML-Systemkonfiguration

Die initiale Systemkonfiguration zeigt den initialen Systemzustand. Die Systemkonfiguration beinhaltet Angaben über die initialen Objekte, deren Attributwerte und Verbindungen zu anderen Objekten. Weder in der Spezifikation der UML noch in der der SysML wird eine Aussage über die Modellierung von Konfigurationen gemacht.
In UML gibt es für die Modellierung von Instanzen die Metaklasse `InstanceSpecification`. Mit dieser könnte die initiale Systemkonfiguration modelliert werden. Eine weitere Möglichkeit die initiale Systemkonfiguration festzulegen, könnte über die Unterstützung durch ein Tool, welches in der Lage ist das Verhalten der initialen Objekte auszuführen, realisiert werden. Hierfür würde es beispielsweise genügen einen Block zu bestimmen, von dem ein Objekt erzeugt werden soll. Anschließend müsste dann wiederum die Erzeugung aller Parts des Blockes veranlasst werden.

3.3 Autofähre-System (zeitdiskretes Verhalten)

Das nachfolgende Beispiel stammt aus „Objektorientierte Prozeßsimulation in C++“ von Joachim Fischer und Klaus Ahrens [9] und wurde leicht abgewandelt, um ein möglichst einfaches Beispiel für ein System mit ausschließlich zeitdiskreten Zustandsänderungen zu modellieren.

3.3.1 Wortmodell

Das Autofähre-System besteht aus einer Fähre und zwei Ufern. An den Ufern erscheint nach einer zufälligen Ankunftszeit fortlaufend ein Auto. Die Autos an einem Ufer erreichen mit Hilfe der Fähre das andere Ufer. Die Fähre bewegt sich zwischen den Ufern und überführt eine begrenzte Anzahl an Autos. Erreicht sie ein Ufer, werden zunächst die aufgeladenen Autos entfernt und dann wartende Autos aufgeladen, wobei das Entfernen und Aufladen eines Autos ein zeitverbrauchender Vorgang ist, genau wie die Überfahrt zum anderen Ufer. Die Fähre verweilt eine Mindestzeit an einem Ufer.
Die Zeiteinheit, die im System verwendet wird, ist Sekunde. Nachfolgend sind die initialen Werte der Variablen bzw. deren Wahrscheinlichkeitsverteilung angegeben.

- Ankunftszeit eines Autos: Exponentialverteilung (Lambda = 0,1 Sekunden^{-1})
- Maximale Anzahl von Autos auf der Fähre: 5
- Auf-/Abladezeit eines Autos: 30 Sekunden
- Überquerungszeit der Fähre: Normalverteilung (Erwartungswert = 300 Sekunden, Varianz = 30 Sekunden)
- Mindestaufenthalt der Fähre: 300 Sekunden

3.3.2 SysML-Modell

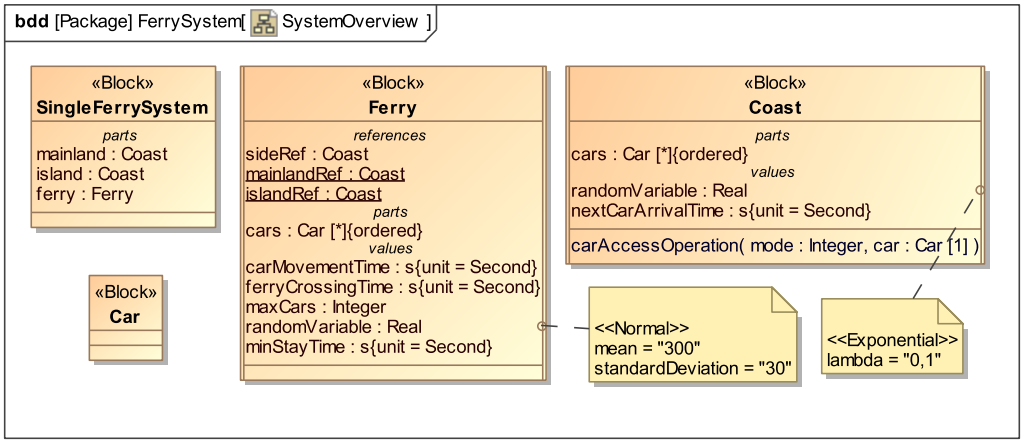
Das SysML-Modell des Systems besteht aus einem Strukturmodell (Blockdefinitions- und internen Blockdiagramm), einer Konfigurationsbeschreibung (Objektdiagramm) und einem Verhaltensmodell (Zustands- und Aktivitätsdiagrammen). Die nicht aufgeführten Diagramme befinden sich im Anhang A.

Strukturmodell

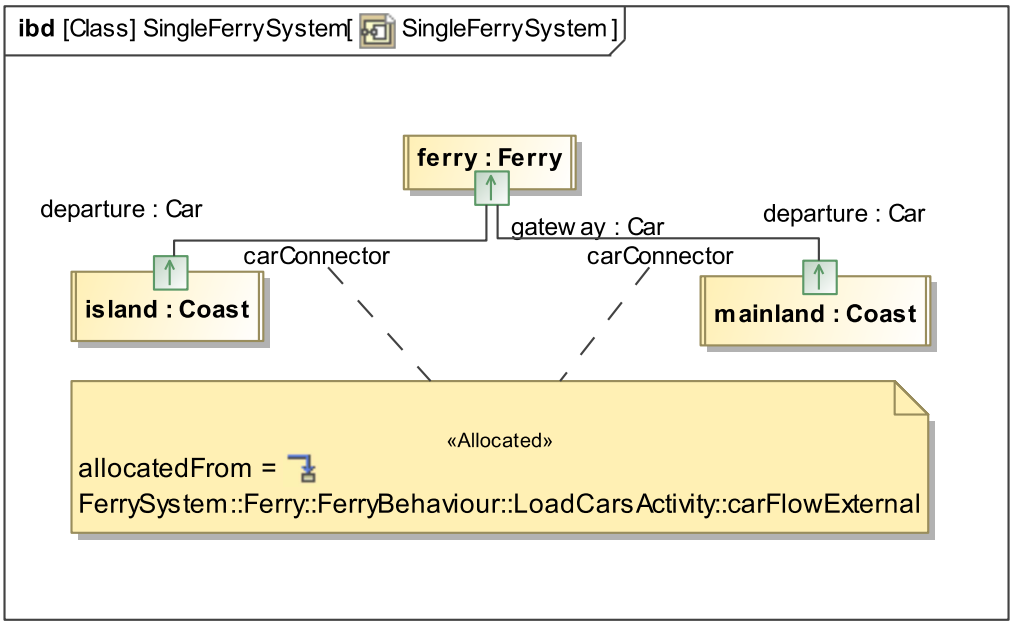
Das Modell enthält die Blöcke `Coast`, `Car` und `SingleFerrySystem` (siehe Abbildung 3.1a). `Ferry` und `Coast` sind zwei aktive Blöcke, d. h. sie besitzen eine Verhaltensbeschreibung, die für das Meta-Attribut `classifierBehavior` angegeben ist. Die Blöcke `Car` und `SingleFerrySystem` beschreiben Datenstrukturen, wobei `SingleFerrySystem` ein Block mit den Part-Propertys Fähre (`ferry`) und zwei Ufern (`mainland` und `island`) ist. Die Parts des Autofähre-Systems besitzen Objektflussports, die untereinander verbunden sind.
In der Abbildung 3.1b sind die drei Parts des Autofähre-Systems dargestellt. Die Blöcke `Ferry` und `Coast` besitzen jeweils einen atomaren Objektflussport. Die FlowPort-Propertys der Parts sind mit Konnektoren verbunden und dienen dem Austausch der Autos (Instanzen des Blocks `Car`). Es besteht ein Zusammenhang zwischen den definierten Konnektoren und den Objektflusskanten in der Verhaltensbeschreibung. Hierfür wurde eine Zuteilungsbeziehung definiert, die als `allocatedFrom` dargestellt wird. Die zugehörige Kante in der Verhaltensbeschreibung erhält ein `allocatedTo`, um ihre Zuordnung zum Konnektor zu bestimmen (siehe Abbildung 3.3c).
Die im Modell durchgängig verwendete Zeiteinheit soll Sekunde in der Dimension Zeit sein. Es wurde ein Wertetyp angelegt, der die Definitionen von Sekunde und Zeit verwendet (siehe Abbildung 3.2). Die Value-Propertys `carMovementTime`, `ferryCrossingTime`, `minStayTime` und `nextCarArrivalTime` sind von diesem Typ und werden in der Verhaltensbeschreibung bei der Definition von Zeitereignissen benutzt.
Attribute, deren Werte einer Verteilungsfunktion entsprechen, sind vom Wertetyp `Real` und benutzen eine Spezialisierung des Stereotyps `DistributedProperty`. Hierfür wurden die Stereotypen `«Normal»` für die Normalverteilung und `«Exponential»` für die Exponentialverteilung definiert.
Die Value-Propertys mit dem Namen `randomVariable` verwenden diese Stereotypdefinitionen.

Systemkonfiguration

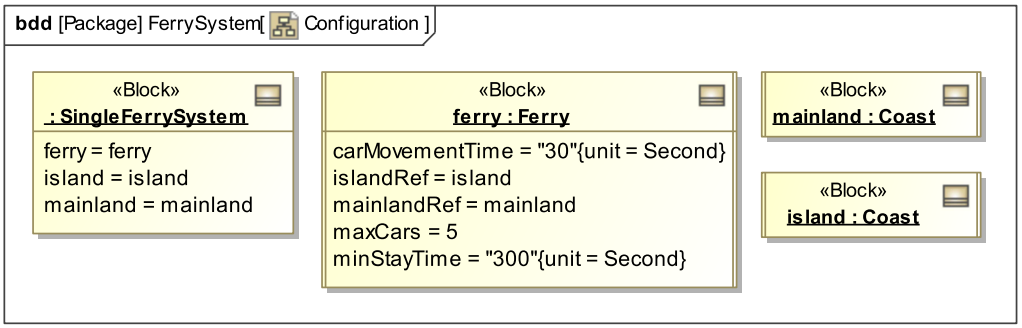
Die Beschreibung der initialen Systemkonfiguration erfolgt über die Angabe der initialen Instanzen und Werte in Form von Instanzen der UML-Metaklasse `InstanceSpecification` wie in Abschnitt 3.2 beschrieben. Das Autofähre-System besitzt initial eine Fähre und zwei Ufer (siehe Abbildung 3.1c). Die Attribute der Fähre haben initial die angegebenen Werte. Die Werte der Tagged-Values müssen bereits bei



(a) Blockdefinitionsdiagramm des Autofähre-Systems

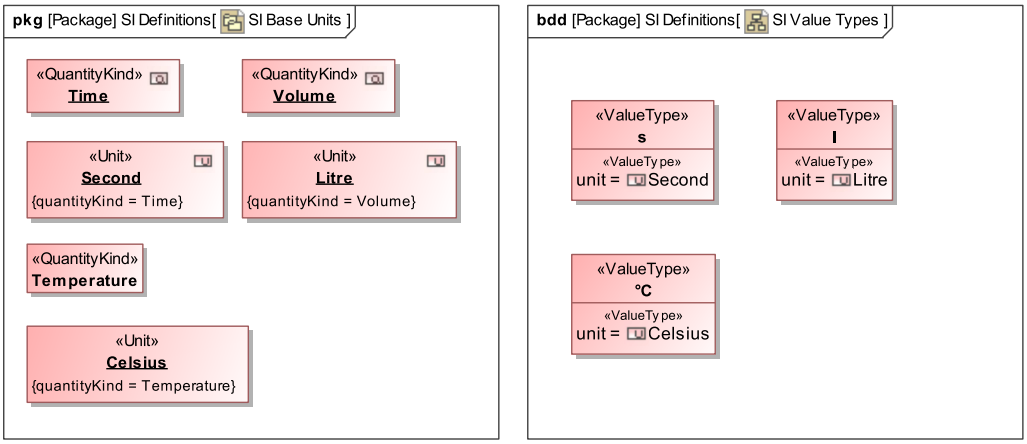


(b) Internes Blockdiagramm des Autofähre-Systems



(c) Konfiguration des Autofähre-Systems

 Abbildung 3.1: Strukturdiagramme des Autofähre-Systems



(a) Definition von Dimensionen und Einheiten (b) Definition von Wertetypen

Abbildung 3.2: Ausschnitt des Internationalen Einheitssystems

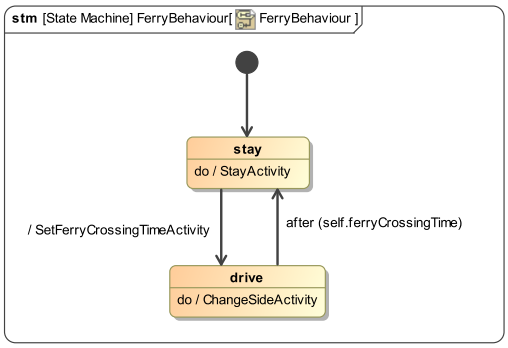
der Typ-Definition (Anwendung der Stereotypen) erfolgen (siehe Abbildung 3.1a).

Verhaltensmodell

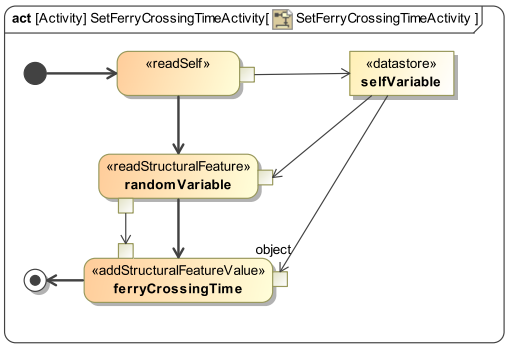
Das Autofähre-System definiert drei aktive Parts, deren Blockdefinition legt eine Verhaltensbeschreibung für `classifierBehavior` fest und das Meta-Attribut `isActive` hat den Wert `true`.

Fähre Die Verhaltensbeschreibung der Fähre ist in diesem Fall eine Zustandsmaschine (siehe Abbildung 3.3a), die die zwei Zustände `stay` und `drive` besitzt. Im Zustand `stay` wird die Fähre zunächst entladen und anschließend mit den an dem angelegten Ufer befindlichen Autos beladen (siehe Abbildung 3.3c).
Wenn diese Aktivitäten abgeschlossen sind und die Mindestaufenthaltsdauer am Ufer überschritten wurde, wird die Aktivität `SetFerryCrossingTimeActivity` ausgeführt. Der Wert, der für die Zeitdauer der Fahrt zum anderen Ufer benötigt wird, hängt von dem Attribut `randomVariable` ab. Der gelesene Wert wird in `ferryCrossingTime` gespeichert (siehe Abbildung 3.3b), da die Semantik für DistributedPropertys hinsichtlich der Bereitstellung von Werten nicht festgelegt ist. Anschließend wird die Referenz auf das aktuelle Ufer neu gesetzt.
Der Übergang vom Zustand `drive` in den Zustand `stay` wird vollzogen, sobald ein relatives Zeitereignis eintritt. Die Dauer, nach der das Zeitereignis stattfindet, wurde auf den Wert des Attributes `ferryCrossingTime` gesetzt, damit dieser nach dem Eintreten des Zeitereignisses mit einem neuen Wert belegt werden kann.

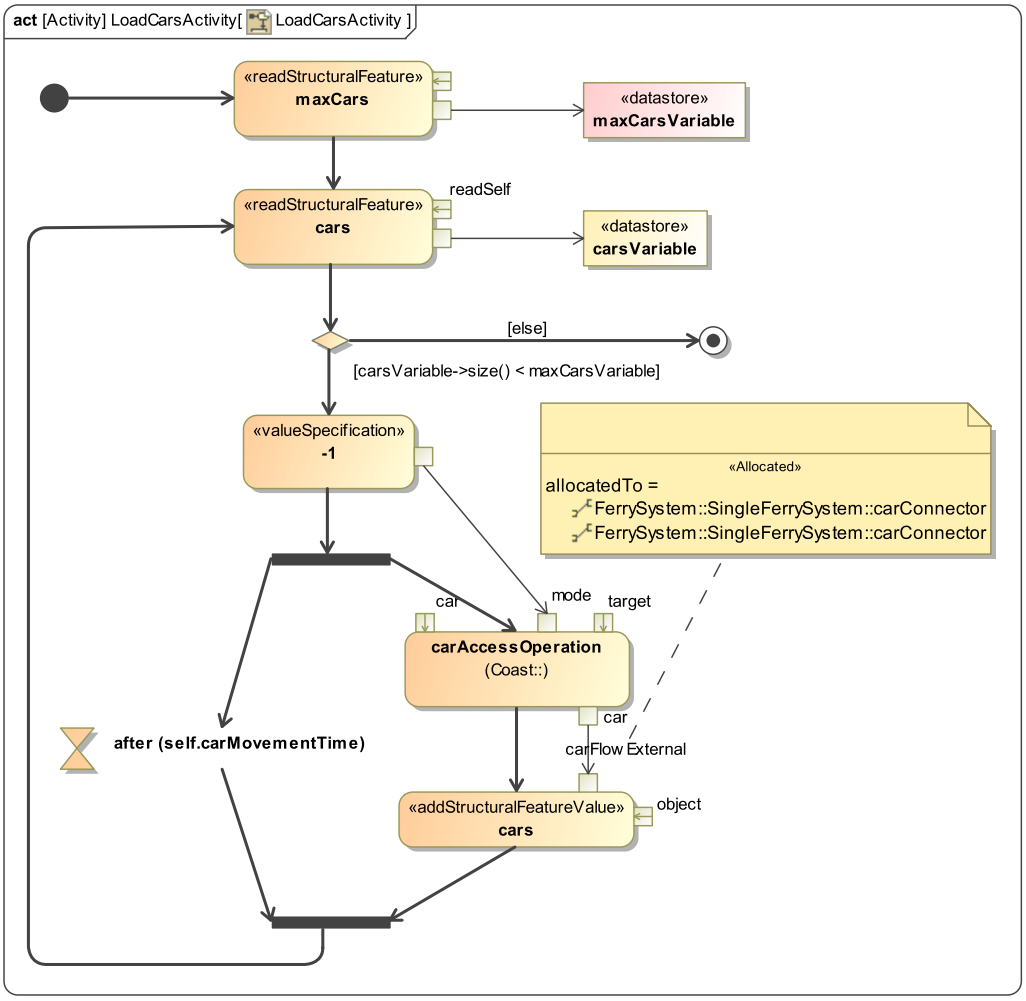
Ufer Die Verhaltensbeschreibung des `classifierBehavior` des Ufers wurde ebenfalls mit Hilfe einer Zustandsmaschine beschrieben. Sobald das Verhalten ausgeführt



(a) Zustandsmaschine der Fähre



(b) Aktivität für das Setzen der Überfahrtszeit



(c) Aktivität für das Beladen von Autos

Abbildung 3.3: Ausschnitt der Verhaltensbeschreibung der Fähre

wird und der Übergang in den ersten Zustand `idle` erfolgt, wird die Aktivität zum Setzen der Zeitdauer für die Ankunft des nächsten Autos ausgeführt. In diesem Zustand wird verblieben bis das relative Zeitereignis eintritt, welches nach der Zeitdauer für die Ankunft des nächsten Autos auftritt.
Im Zustand `carArrival` wird eine Instanz eines Autos erzeugt und in die geordnete Liste der wartenden Autos am Ufer eingereiht (Meta-Attribute `isOrdered` und `isUnique` haben den Wert `true`). Das Attribut für die Menge der Autos hat die Sichtbarkeit `private`, damit eine Manipulation verhindert werden kann. Das Einordnen in die Menge bzw. ein Zugriff zum Einfügen oder Löschen einer Instanz eines Autos erfolgt über die Operation `carAccessOperation`, deren Verhaltensbeschreibung eine Aktivität `carAccessActivity` ist (siehe Abbildung 3.4). Das

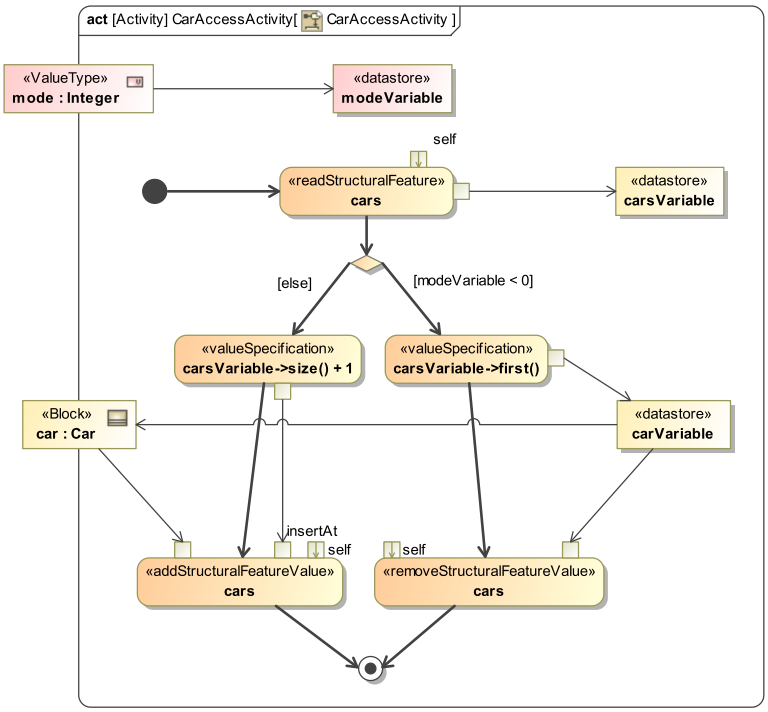


Abbildung 3.4: Aktivität für den Zugriff auf wartende Autos am Ufer

Meta-Attribut `isSingleExecution` der Aktivität hat den Wert `true` (zur Laufzeit gibt es nur eine Instanz der Aktivität). Der Aktivitätsparameter `Car` muss mit einer Instanz der Klasse `car` belegt werden und beinhaltet das zu bearbeitende Auto. Der Aktivitätsparameter `mode` kann mit einem Integer-Wert belegt werden und bestimmt das Ausführen der Aktivität zum Entnehmen (Werte kleiner null) oder Hinzufügen (sonstige Werte) des Autos.

Semantische Erweiterungen

- Werteverteilung:
  Die Verwendung von Spezialisierungen des Stereotyps `DistributedProperty` besagen, dass die Werte, die diese Variable besitzt, einer Verteilung von Werten über einen definierten Wertebereich entsprechen. Es gibt keine Aussage in der Spezifikation über die Realisierung dieser Definition bzw. der Belegung mit Werten. Für die Modelle soll daher gelten, dass jeder lesende Zugriff auf eine solche Variable einen entsprechenden Wert bereitstellt.

- Zeitereignis:
  Zeitereignisse werden über das Meta-Attribut `when` mit einem Zeitausdruck belegt, der relativ zum aktuellen Zeitpunkt sein kann. In den Modellen wurde statt der Angabe eines konstanten Ausdruckes ein Verweis auf einen Wert eines Attributes in Form eines OCL-Ausdruckes verwendet. Zusätzlich bezieht sich der Kontext des Ausdruckes auf den der Verhaltensbeschreibung zugehörigen Kontext (Meta-Attribut `classifier`).

- Parameter:
  Im Metamodell ist kein Zusammenhang zwischen den Parameter einer Aktivität und einer Operation festgelegt. Daher gilt die Erweiterung, dass es sich bei Parametern mit identische Namen und Typen (und Multiplizitäten) um dieselben handelt.

- Systemkonfiguration:
  Die Angabe einer Systemkonfiguration in einem Objektdiagramm ist in SysML bisher nicht standardisiert. Zusätzlich zur Angabe von initialen Objekten und Werten, entspricht die Instanz des Blockes, der alle anderen initialen Objekte enthält, der des Systems und sein Block damit der der Systemtypdefinition.

3.4 Badewanne-System (zeitkontinuierliches Verhalten)

Das folgende Beispiel aus „Systeme, Dynamik, Simulation - Modellbildung, Analyse und Simulation komplexer Systeme“ von Hartmut Bossel [1] beschreibt ein kontinuierliches Ein-Speichersystem mit konstantem Zu- und Abfluss. Der Zustand des Systems ändert sich ausschließlich zeitkontinuierlich.

3.4.1 Wortmodell

Das Badewannen-System besteht aus einer Badewanne mit einem Wasserhahn, durch den neues Wasser in die Wanne gelangt, und einen Abfluss, durch den Wasser aus der Wanne abfließt. Je nach Zu- und Abflussrate verändert sich die Wassermenge in der Badewanne. Nachfolgend sind die initialen Belegungen der Raten und Variablen angegeben.

- Wasserzuflussrate (Wasserhahn): 0,100 Liter/Sekunde

- Wasserabflussrate (Abfluss): 0,001 Liter/Sekunde
- Wassermenge in der Badewanne: 0 Liter

3.4.2 SysML-Modell

Für die Modellierung der Struktur wurde ein Blockdefinitionsdiagramm, für die Konfiguration ein Objektdiagramm und für die zeitkontinuierliche Zustandsänderung ein Parameterdiagramm verwendet.

Strukturmodell

Der Block `Bathtube` besitzt zwei Value-Propertys (`tapWaterFlow` und `drainWaterFlow`) vom Wertetyp `l/s`, die die Zu- und Abflussrate festlegen. Der aktuelle Wasserinhalt der Badewanne wird mit dem Value-Property `waterLevel` vom Wertetyp `l` gespeichert (siehe Abbildung 3.5). Der Zusammenhang zwischen den drei

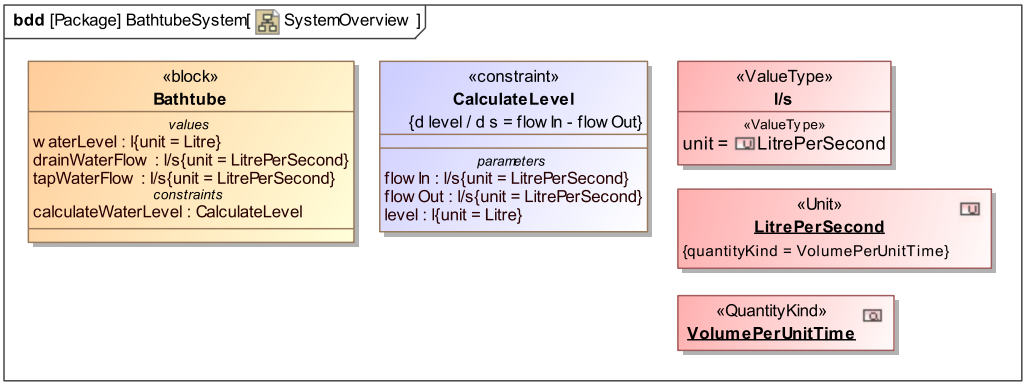
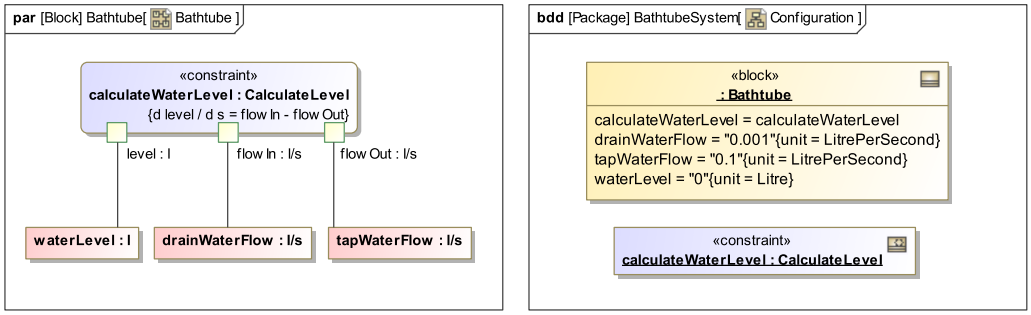


Abbildung 3.5: Blockdefinitionsdiagramm des Badewanne-Systems

Value-Propertys der Badewanne wird über das Constraint-Property `calculateWaterLevel` bestimmt. Es ist vom Constraint-Blocktyp `CalculateLevel`, welches ein Constraint über die drei Parameter `flowIn`, `flowOut` und `level` definiert. Das Constraint ist in Form einer Zustandsgleichung angeben, mit der über den Verlauf der Zeit und den Werten des Zu- und Abflusses der Wert für den Parameter `level` berechnet und im Value-Property `waterlevel` gespeichert werden kann.

Parametermodell

Im Parameterdiagramm wird der Zusammenhang zwischen den Parametern des Constraint-Propertys `calculateWaterLevel` des Constraint-Blockes `CalculateLevel` und den Value-Propertys des Blockes `Bathtube` festgelegt (siehe Abbildung 3.6a). Zwischen den Parametern des Constraints und den Attributen des Blockes wurden Konnektoren definiert, auf denen der Stereotyp `«BindingConnector»` angewandt wurde. Damit ist sichergestellt, dass bei Instanzen des Blockes der Wert



(a) Parameterdiagramm der Badewanne (b) Konfiguration des Badewanne-Systems

Abbildung 3.6: Ausschnitt des Badewanne-Systems

des Attributes und der Wert des Parameters gleich sein müssen (siehe Abschnitt 2.2.4).

Systemkonfiguration

Das System besitzt initial eine Badewanne, die eine Instanz des Constraint-Blocks `CalculateLevel` besitzt. Die initialen Werte der Value-Propertys entsprechen denen aus dem Wortmodell (siehe Abbildung 3.6b).

Semantische Erweiterungen

- Zeit:
  Für den Zugriff auf einen Zeitwert stellt die Sprache kein Konstrukt bereit. Zeit kann somit selbst als ein Property (mit dem Zugriffsrecht `public`) definiert werden, auf welches dann alle Parts Zugriff haben (explizit). Es kann auch implizit das Vorhandensein einer globalen Zeit angenommen werden, welche erst durch ein Laufzeitsystem bereitgestellt wird (implizite Systemzeit). Hierbei ist zu beachten, dass auch der Typ und die Einheit festgelegt werden müssen. Zeit wird im Modell beispielsweise bei der Definition von Zustandsgleichungen verwendet.

- Zeitkontinuierliche Zustandsänderung:
  Die Definition einer zeitkontinuierlichen Zustandsänderung fand im Modell in Form von Constraints statt. Diese beinhalten eine über Veränderungsraten definierte Zustandsgleichung.

3.5 Barrenofen-System (kombiniertes Verhalten)

Das System stammt aus der Vorlesung „Objektorientierte Modellierung, Spezifikation und Implementierung“ von Prof. Dr. Joachim Fischer [2] und beinhaltet Objekte

mit zeitkontinuierlichen und Objekte mit zeitdiskreten Zustandsänderungen.

3.5.1 Wortmodell

Das Barrenofen-System besteht aus einem Ofen, einem Hersteller von Barren und einer Kontrolleinheit.
Die Barren werden in einem festen Intervall erstellt und gelagert. Sie besitzen eine Temperatur, die sich entsprechend der Umgebungstemperatur ändert. Die Kontrolleinheit beschickt den Ofen mit Barren aus dem Lager des Herstellers, sofern die Kapazität des Lagers des Ofens (Ofenkammer) nicht erschöpft ist. Des Weiteren beobachtet die Kontrolleinheit die im Ofen befindlichen Barren und entnimmt einen Barren, falls dieser die Zieltemperatur erreicht hat. Der Ofen heizt kontinuierlich mit der eingestellten Erhitzungstemperatur. Nachfolgend sind die initialen Werte der Variablen angegeben.

- Barrentemperatur: 200 °C
- Barrenerstellungsintervall: 30 Sekunden
- Lagertemperatur: 25 °C
- Barrenzieltemperatur: 380 °C
- Kapazität der Ofenkammer: 10
- Ofenerhitzungstemperatur: 500 °C
- Ofentemperatur: 400 °C

3.5.2 SysML-Modell

Die Struktur des Systems wurde mit einem Blockdefinitions- und einem internen Blockdiagramm modelliert. Zusätzlich wurden die Bedingungen der zeitkontinuierlichen Zustandsänderungen in Parameterdiagrammen beschrieben. Verhaltensbeschreibungen, die ausschließlich zeitdiskrete Änderungen beinhalten, wurden mit Zustandsmaschinen beschrieben und Aktionen sowie Operationen mit Aktivitätsdiagrammen. Die nicht aufgeführten Diagramme befinden sich im Anhang B.

Strukturmodell

Der Block `SingleHeatingManufactureSystem` beschreibt das Barrenofen-System (siehe Abbildung 3.7), welches einen Hersteller von Barren (`factory`), einen Ofen (`furnace`) und eine Kontrolleinheit (`controller`) besitzt. Der Hersteller der Barren hat eine Ofenkammer (`manufacturePlace`)), welche die erstellten Barren lagert. Das Lager verfügt über ein Value-Property `temp`, welches die Umgebungstemperatur für die gelagerten Barren bestimmt. Der Ofen hat ebenfalls ein Lager (`heatingPlace`), eine Erhitzungstemperatur (`heatingTemp`) und eine Temperatur

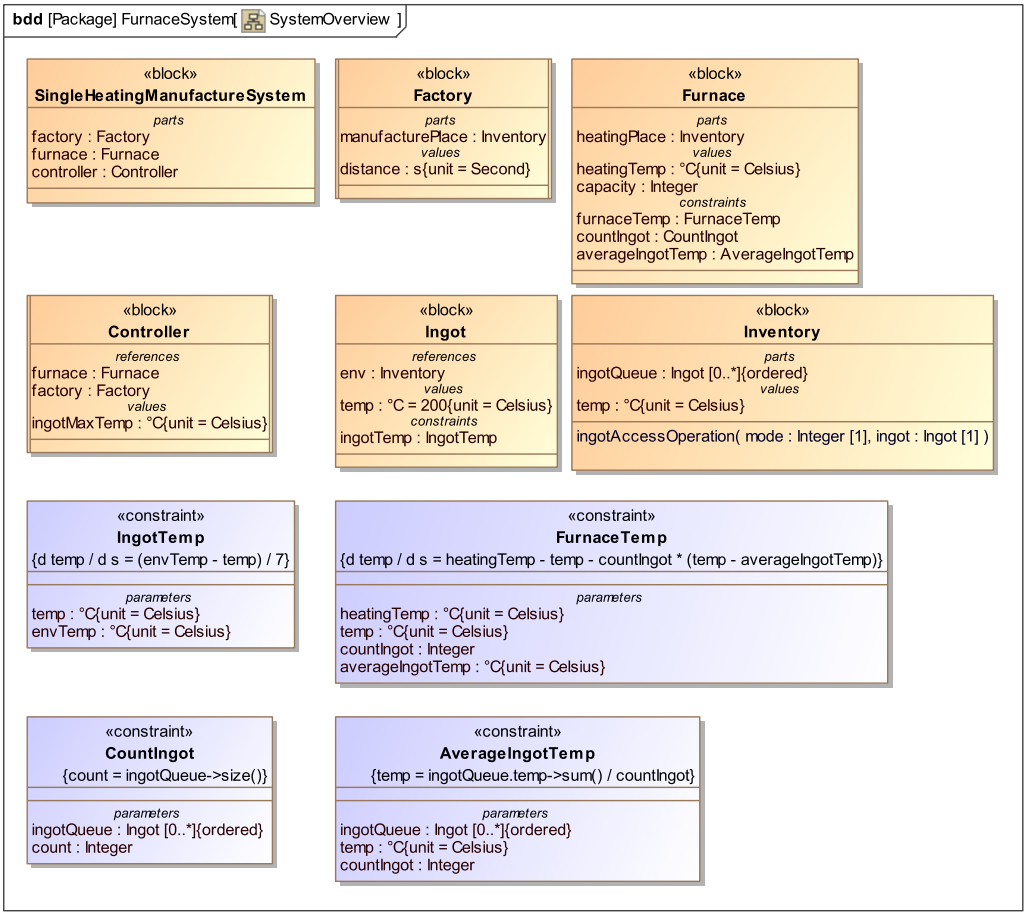


Abbildung 3.7: Blockdefinitionsdiagramm des Barrenofen-Systems

des Lagers im Ofen (Ofenkammer). Barren besitzen selbst ebenfalls eine Temperatur (`temp`) und eine Referenz auf das Lager (`env`), in dem sie sich befinden. Die Kontrolleinheit hat Referenzen auf den Hersteller (`factory`) und den Ofen (`furnace`) und die Zieltemperatur für die Barren (`ingotMaxTemp`).

Parametermodell

Der Ofen besitzt eine Zustandsgleichung (`furnaceTemp`), die die Temperatur des Ofens bestimmt (Temperatur (`temp`) des Lagers (`heatingPlace`)). Die Gleichung der Erhitzungstemperatur (`heatingTemp`) hängt von zwei Parametern (`countIngot` und `averageIngotTemp`) ab, die sich aus im Ofen lagernden Barren (`ingotQueue`) berechnen (siehe Abbildung 3.8).

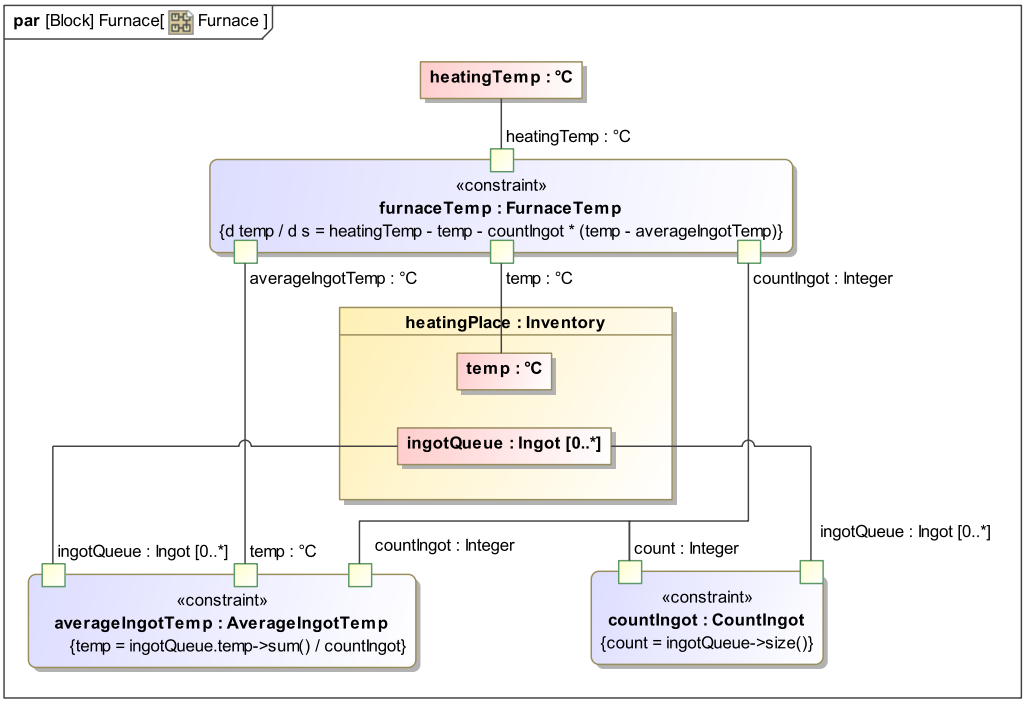


Abbildung 3.8: Parameterdiagramm des Ofens

Systemkonfiguration

Die Abbildung 3.9 zeigt ein Objektdiagramm, das die inititale Systemkonfiguration angibt. Die initiale Temperatur für erstellte Barren ist in der Abbildung 3.7 ersichtlich und musste dort erfolgen, damit zur Laufzeit erstellte Barren einen initialen Wert für die Temperatur besitzen.

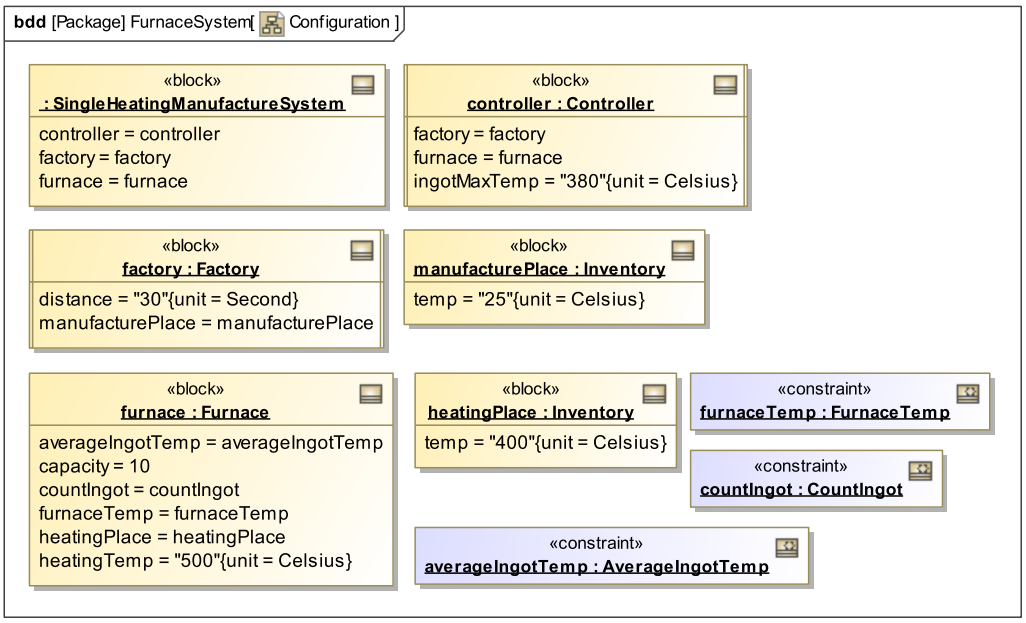


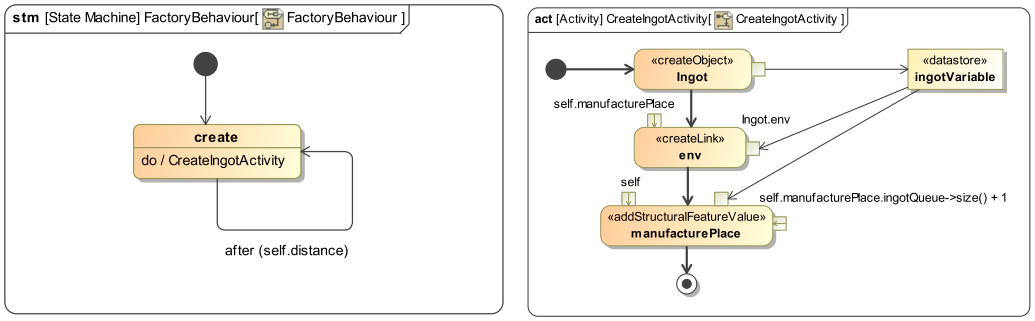
Abbildung 3.9: Konfiguration des Barrenofen-Systems

Verhaltensmodell

Das Modell enthält zwei aktive Blöcke (`Factory` und `Controller`), deren Instanzen das Verhalten des Systems nachbilden.

Hersteller Die Abbildung 3.10a zeigt die Verhaltensbeschreibung für den aktiven Block `Factory`. Der Hersteller erzeugt im Zustand `create` eine neue Instanz von `Ingot` und reiht diese in das Lager (`manufactorePlace`) ein (siehe Abbildung 3.10b). Die Referenz des neu erstellten Barrens zum beinhaltenden Lager muss explizit gesetzt werden, damit das Constraint zur Berechnung der Barrentemperatur ausgewertet werden kann. Anschließend verharrt der Hersteller in diesem Zustand bis die Zeitdauer bis zur nächsten Erzeugung eines Barrens (`distance`) verstrichen ist und erneut der Zustand `create` betreten wird.

Kontrolleinheit Die Verhaltensbeschreibung der Kontrolleinheit besitzt ebenfalls nur einen Zustand (`wait`), der verlassen wird, falls im Lager ein Barren vorhanden ist und auf einen freien Platz in den Ofen transportiert werden kann (siehe Abbildung 3.11). Es wird der Barren transportiert, der am längsten im Lager aufbewahrt wurde. Der Transport des Barrens aus dem Lager in den Ofen erfolgt über den Aufruf der Operation `ingotAccessOperation` am jeweiligen Lagerstandort (Ofenkammer oder Herstellerlager), wobei der Parameter `mode` über Entnahme oder Hinzufügen entscheidet und der zweite Parameter mit dem operierenden Bar-



(a) Zustandsdiagramm des Herstellers

(b) Aktivitätsdiagramm für das Erstellen eines Barrens

Abbildung 3.10: Ausschnitt der Verhaltensbeschreibung des Herstellers

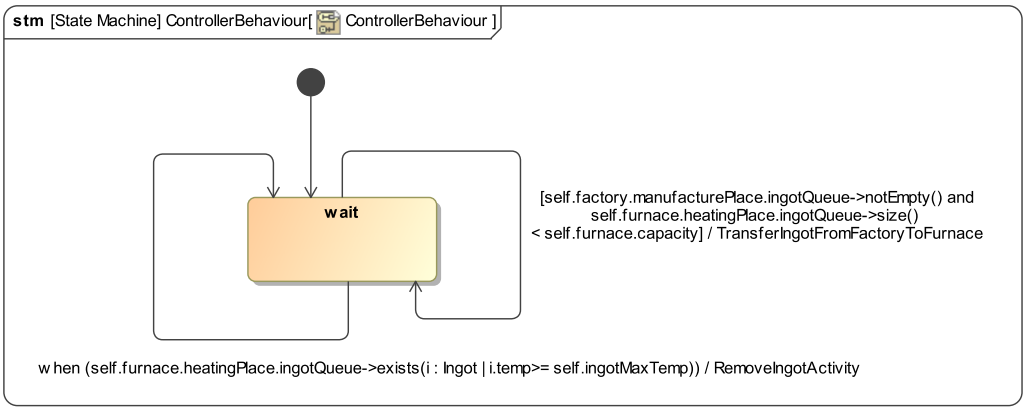


Abbildung 3.11: Zustandsdiagramm der Kontrolleinheit

ren belegt ist (siehe Abbildung 3.12). Den Zustand `wait` kann die Kontrolleinheit ebenfalls verlassen, falls ein Barren im Ofen die Zieltemperatur (`ingotMaxTemp`) erreicht hat. In diesem Fall wird die Aktivität zum Entfernen des Barrens aus dem Ofen (`RemoveIngotActivity`) ausgeführt und ein Barren im Ofen mit der Zieltemperatur oder größer entfernt.

Semantische Erweiterungen

- Objekterzeugung:
  Die Erzeugung von Objekten mit der UML-Aktion `«CreateObject»` macht keine Aussage über die Erzeugung von Parts des Objektes. Daher wurde für das Modell angenommen, dass diese ebenfalls mit der Aktion instanziiert werden.

- Change-Event:
  Change-Events an Transitionen in Zustandsmaschinen beinhalten keine Aussage über die Auswertung zur Laufzeit. Die Realisierung könnte über ein ständiges Abfragen der Bedingung oder durch eine Auswertung der Bedingung bei Änderung an den beteiligten Variablen erfolgen.

- Änderung der Zustandsgleichung:
  Die Zustandsgleichungen, die in den Constraints von Constraint-Blöcken definiert sind, beziehen sich auf Parameter der Constraint-Blöcke, die wiederum mit Value-Propertys von Blöcken verbunden sind (über `«BindingConnector»`. Müssen die Zustandsgleichung verändert werden, kann dies durch Änderung der verknüpften Value-Propertys geschehen (Austausch des verlinkten Value-Propertys oder Wertänderung des Value-Propertys).

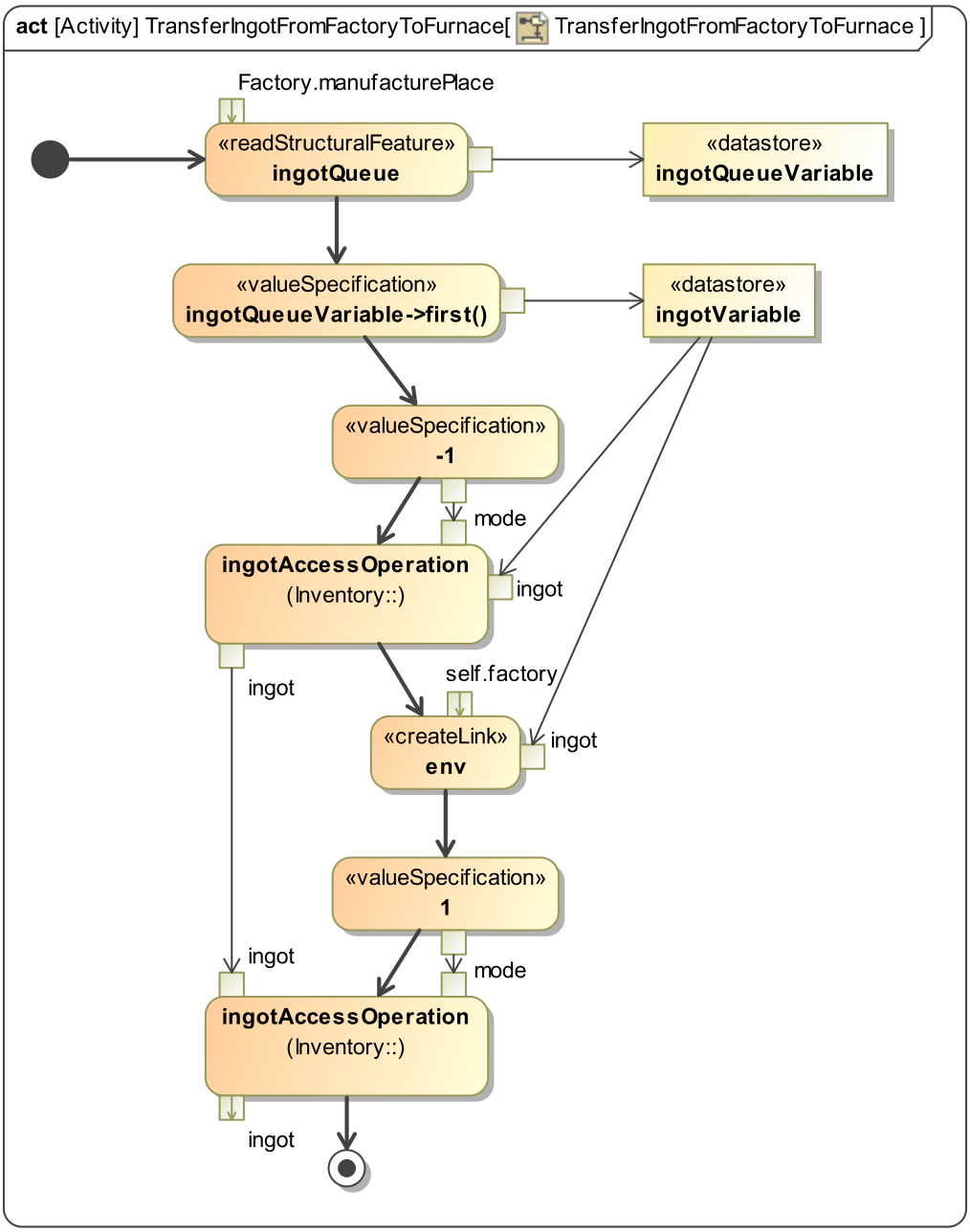


Abbildung 3.12: Aktivitätsdiagramm für die Überführung eines Barrens aus dem Lager in die Ofenkammer

4 Zusammenfassung

In diesem Kapitel werden die Ergebnisse der Untersuchung der Konzepte von SysML und deren Verwendung für die Systemmodellierung zusammengefasst. Die verschiedenen Systemmodelle mit zeitdiskreten, zeitkontinuierlichen und kombinierten Verhaltensänderungen dienten der Überprüfung, inwiefern sich die Sprache zur Modellierung solcher Systeme eignet.

4.1 Ergebnisse

Die Modelle, die mit SysML erstellt werden können, sind objektorientiert und stützen sich auf die Modellierung von Typen (Blöcke oder Wertetypen). Es findet nur die Beschreibung eines Systemtyps statt. Die Modellierung einer Systemkonfiguration (Instanz des Systemtyps bzw. das System) wurde nicht festgelegt und konkrete Instanzen (des Systemtyps und der Typen der Systemelemente) können lediglich mit Hilfe des UML-Konzeptes für Instanzspezifikationen angegeben werden.
Damit ein reales System in Hinblick auf ein konkretes Untersuchungsziel modelliert werden kann, benötigt man mathematische Modelle für die Beschreibung von Werteverteilungen (Zufallszahlen). Die Sprache bietet das Konzept des Distributed-Propertys mit dem ein Property mit einer konkreten Werteverteilung modelliert werden kann, lässt jedoch die Realisierung der Bereitstellung von Werten offen (Zufallszahlengenerator).
Einen weiteren wichtigen Aspekt bei der Modellierung von Systemen spielt die Modellzeit. Diese soll die Zeit des realen Systems abbilden. Die Spezifikation gibt jedoch keine Definition einer globalen Zeit im Modell oder die Definition von Zeit als Property des Systemtyps oder dem Typ eines Systemelements an.
Die Verhaltensbeschreibungen der Typen der Systemelemente können zeitdiskrete, zeitkontinuierliche oder kombinierte Änderungen der Zustände beinhalten. Zeitdiskretes (ereignisbasiertes) Verhalten kann bereits mit UML beschrieben werden und steht auch für die Modellierung von Verhaltensbeschreibungen in SysML zur Verfügung. Für die Modellierung von zeitkonitnuierlichen Zustandsänderungen bietet SysML Elemente, die es ermöglichen die Wertänderung von Propertys eines Typs eines Systemelements in einer beliebigen Sprache anzugeben. Da diese Sprachen nicht festgelegt sind, ist ebenfalls offen gelassen, inwiefern ein Zugriff auf die Modellzeit (falls modelliert) möglich wäre.

4.2 Ausblick

Die Sprache SysML bietet grundlegende Modellierungskonzepte, um Systeme zu beschreiben. Mit Hilfe des UML-Profilmechanismus könnten die Konzepte erweitert oder neue Konzepte und weitere Profile für andere Domänen definiert werden. Für die Domäne der Systemsimulation wäre beispielsweise ein Profil denkbar (z. B. Konzepte für die Überwachung von Werten und Objektzuständen). Erste Ansätze für die Definition eines solchen Profils finden sich in „A Practical Guide to SysML: The Systems Modeling Language“ von Sanford Friedenthal, Alan Moore und Rick Steiner [8].
Eine Promotion zum Thema „Integrating models and simulations of continuous dynamic system behavior into SysML“ von Thomas A. Johnson [3] hat bereits gezeigt, wie Modelle von zeitkontinuierlichen Systemen, die mit SysML modelliert wurden, mit Hilfe einer Transformation in eine Simulationssprache überführt werden können. Simulatoren für zeitdiskrete Systemmodelle in SysML können bereits mit Werkzeugen verschiedener Hersteller erstellt werden. Kombinierte Systeme in SysML (wie in Abschnitt 3.5 beschrieben) können bisher nur partiell für eine Simulation genutzt werden.
Die Modellierung der Verhaltensbeschreibung ist für einfache Aktionen, wie beispielsweise das Lesen eines Wertes eines Propertys, sehr aufwendig und könnte mit Hilfe einer höheren objektorientierten Programmiersprache prägnanter formuliert werden. Ein möglicher Ansatz könnte eine textuelle Sprache der UML-Actions sein, basierend auf der von Stephen J. Mellor und Marc J. Balcer in „Executable UML: A Foundation for Model-Driven Architecture“ [10] für UML 1.4 vorgestellten Sprache.

A Ergänzung zum SysML-Modell des Autofähre-Systems

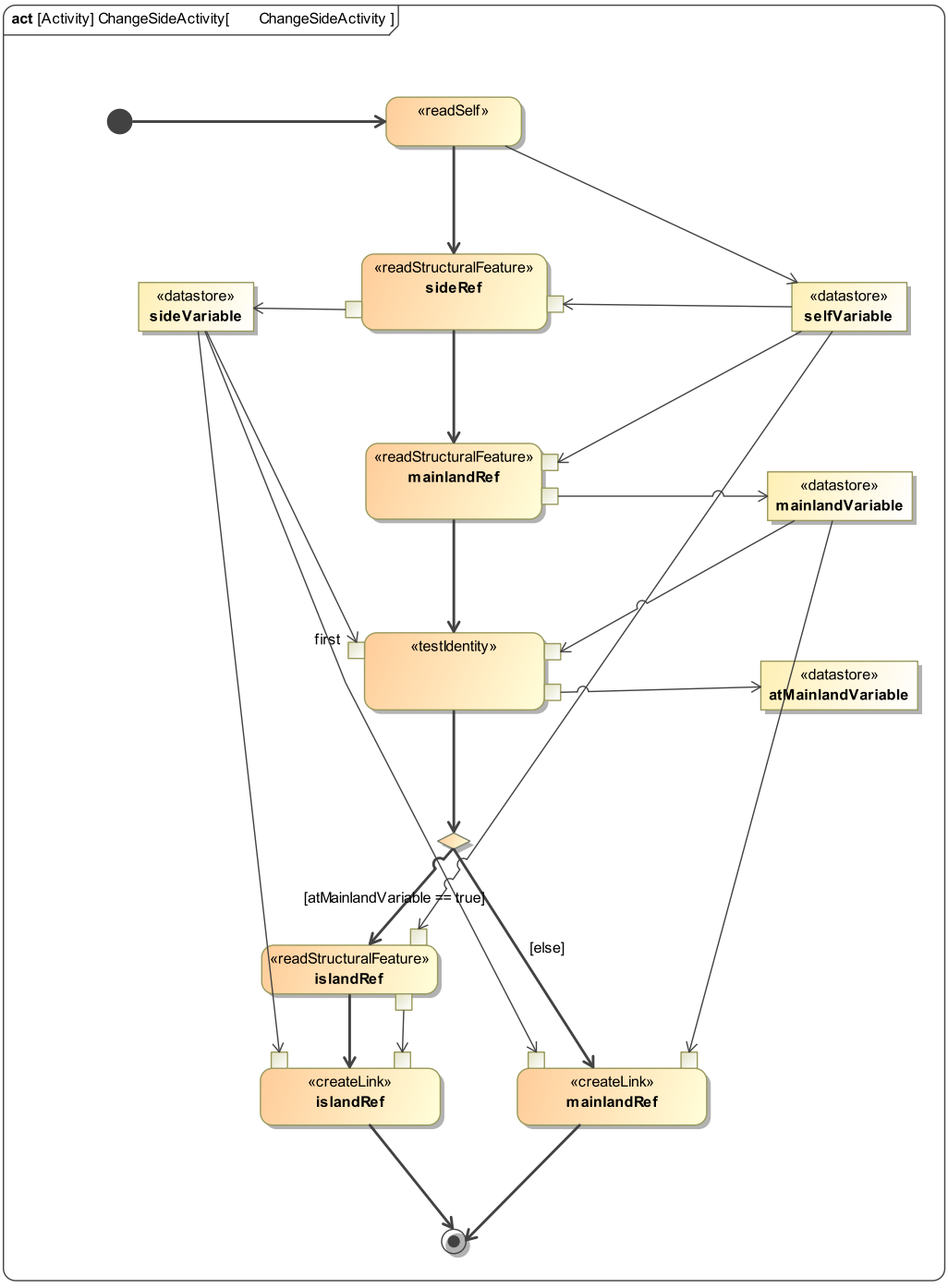


Abbildung A.1: Aktivität für das Wechseln der Ufer

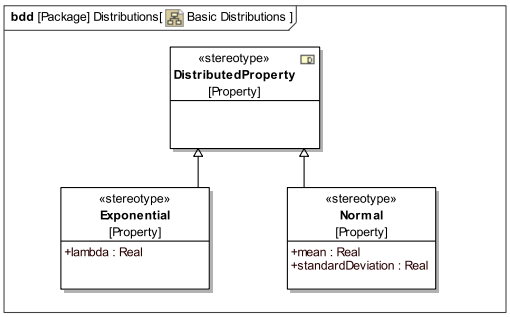


Abbildung A.2: Spezialisierungen des Stereotyps DistributedProperty

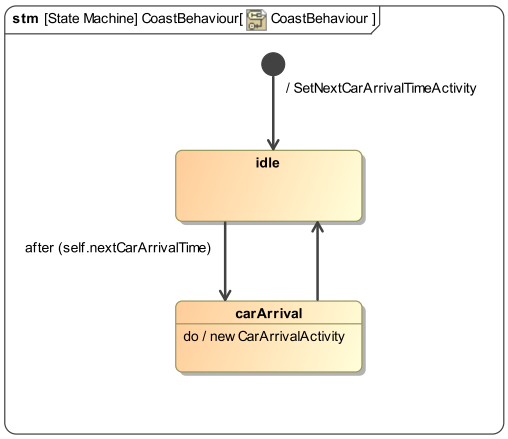


Abbildung A.3: Zustandsmaschine des Ufers

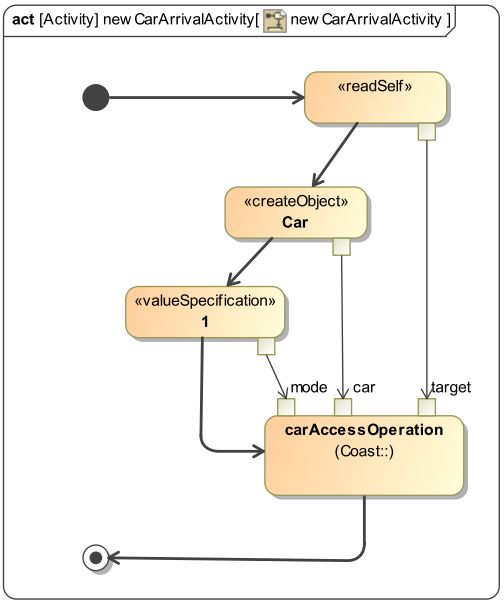


Abbildung A.4: Aktivität für die Ankunft eines Autos

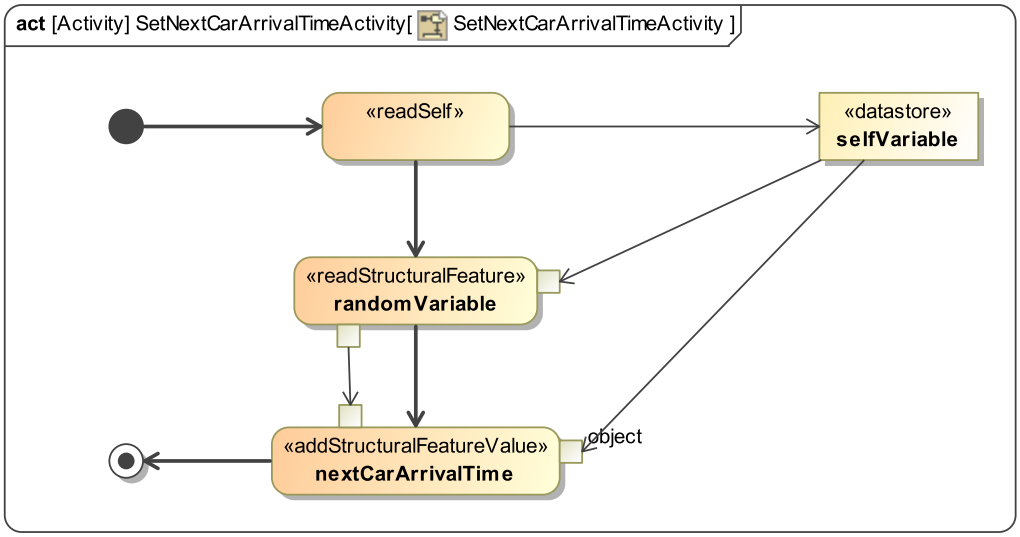


Abbildung A.5: Aktivität für das Setzen Ankunftszeit des nächsten Autos

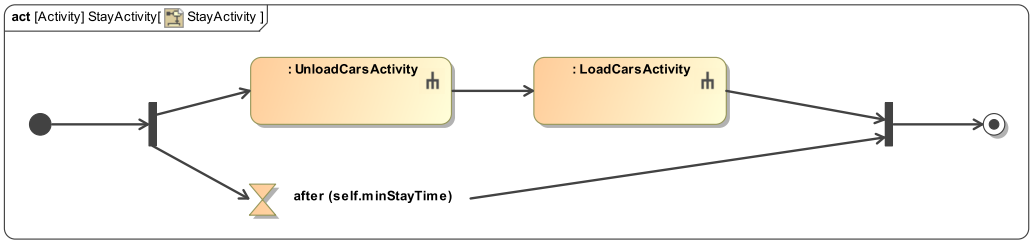


Abbildung A.6: Aktivität für das Ent- und Beladen von Autos

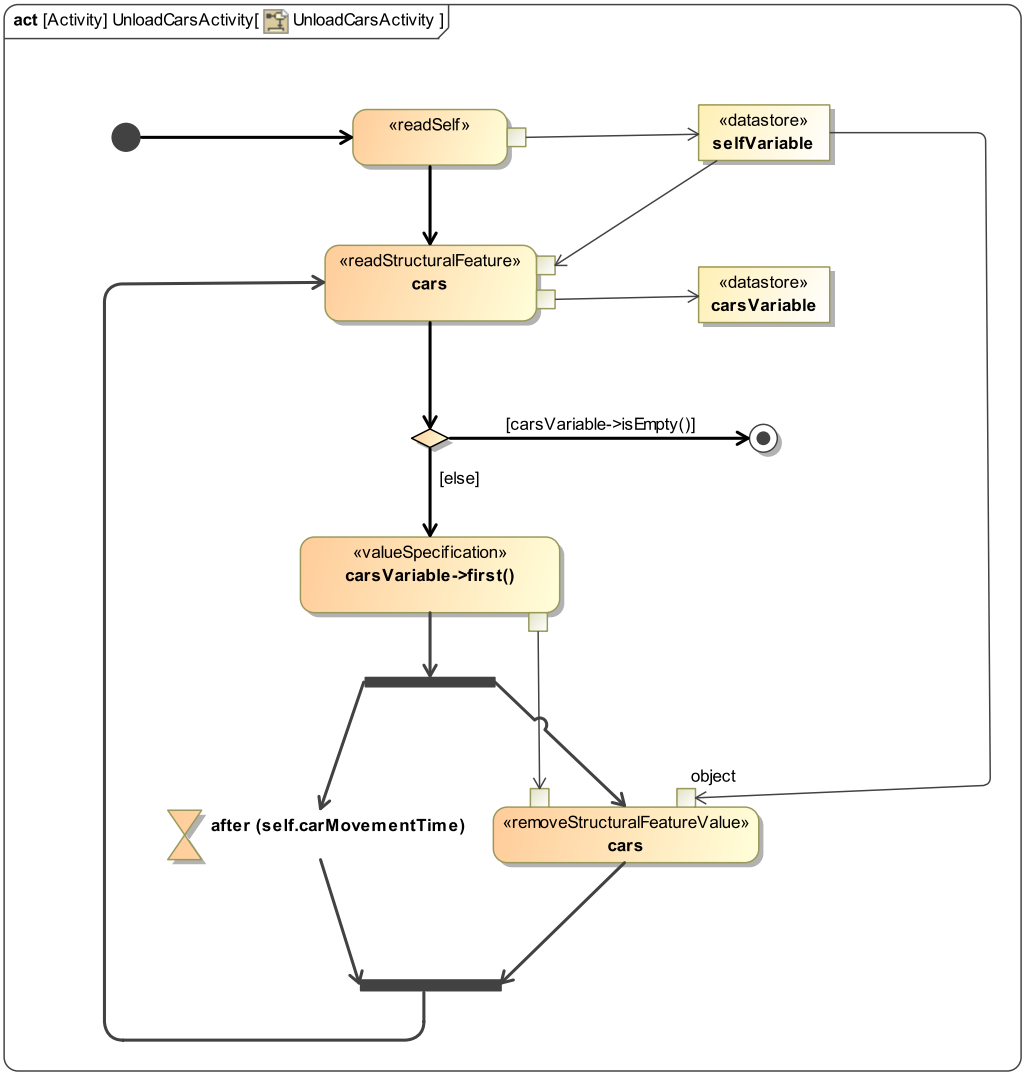


Abbildung A.7: Aktivität für das Entladen der Autos

B Ergänzung zum SysML-Modell des Barrenofen-Systems

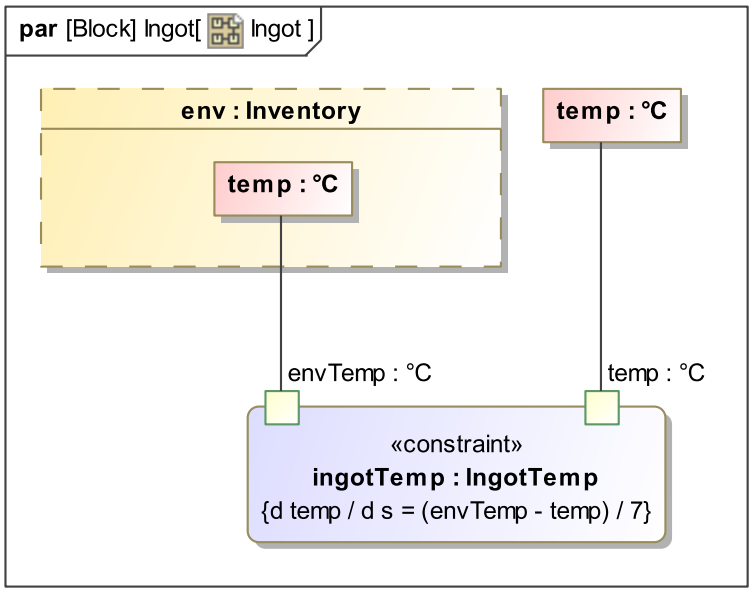


Abbildung B.1: Parameterdiagramm des Barrens

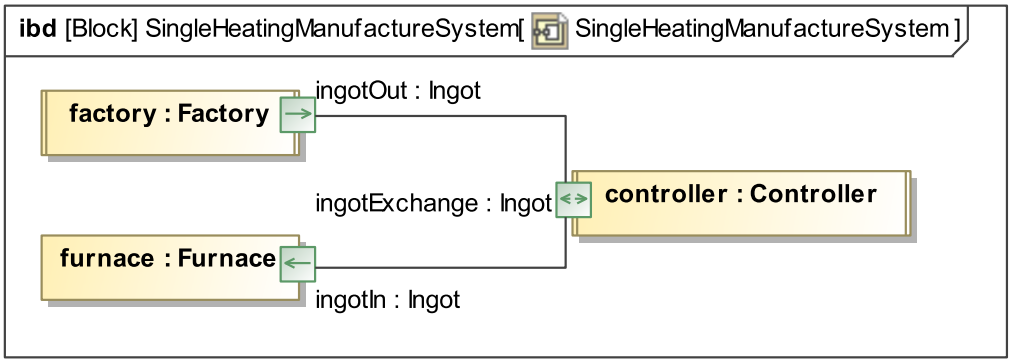


Abbildung B.2: Internes Blockdiagramm des Barrenofens-Systems

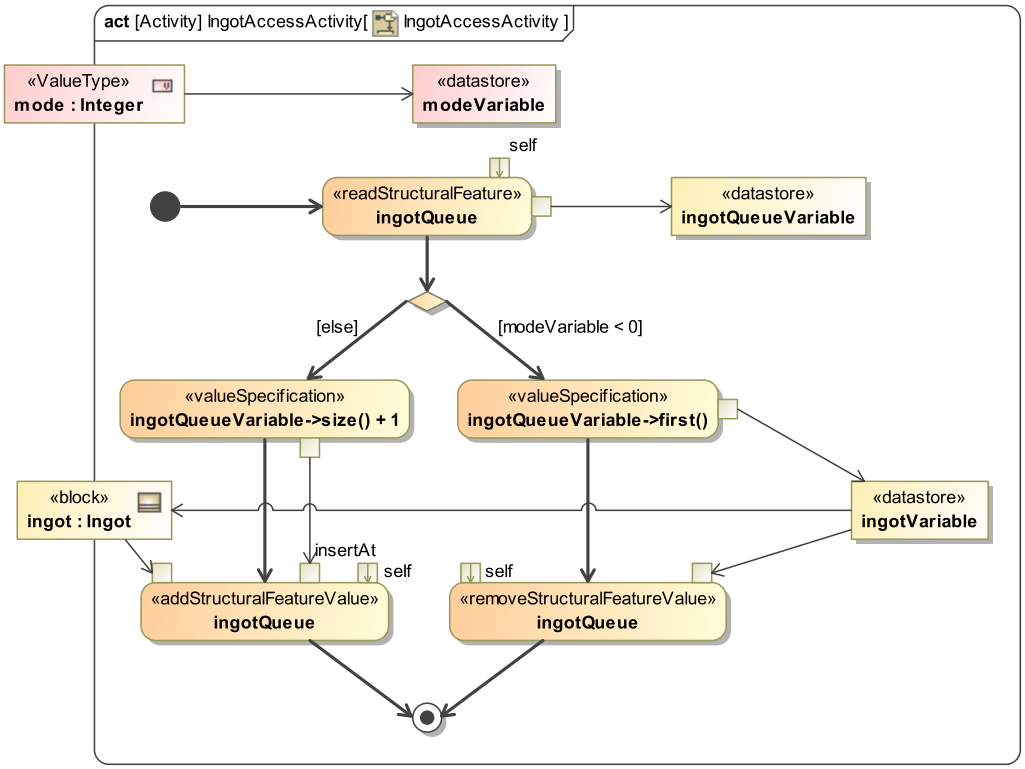


Abbildung B.3: Aktivität für den Zugriff auf die Barren im Lager

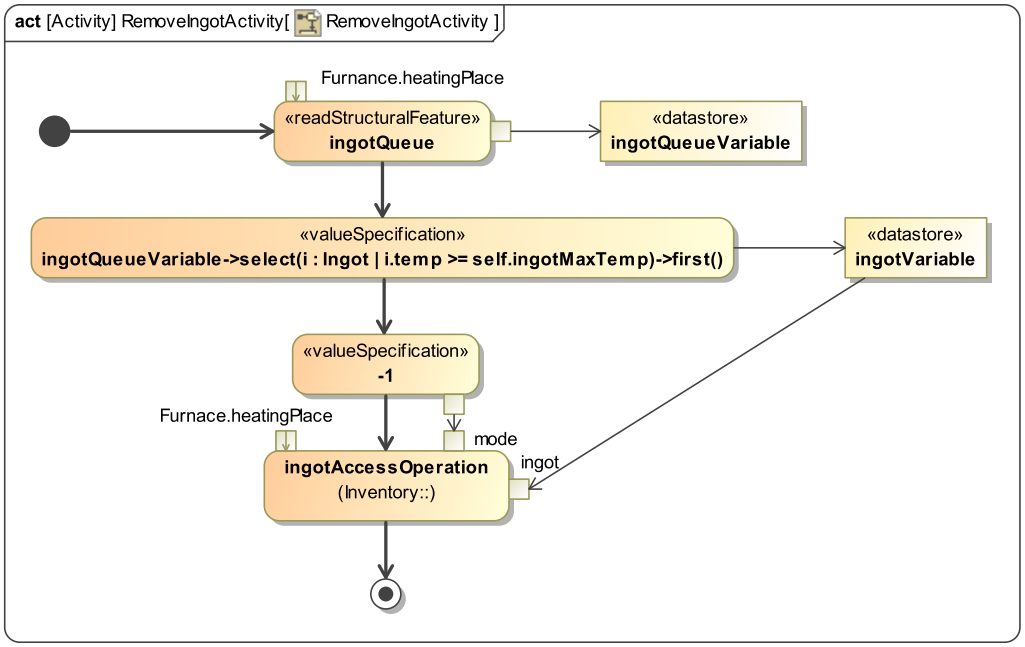


Abbildung B.4: Aktivität für das Entfernen eines Barrens aus der Ofenkammer

Literaturverzeichnis

[1] Hartmut Bossel. *Systeme, Dynamik, Simulation - Modellbildung, Analyse und Simulation komplexer Systeme.* Books on Demand GmbH, Norderstedt, 2004.

[2] Prof. Dr. Joachim Fischer. Objektorientierte modellierung, spezifikation und implementierung. Objektorientierte Simulation mit ODEMx, 2006.

[3] Thomas A. Johnson. *Integrating models and simulations of continuous dynamic system behavior into SysML.* PhD thesis, Georgia Institute of Technology, 2008.

[4] OMG. Meta Object Facility (MOF) Core Specification Version 2.0. Technical Report formal/2006-01-01, OMG, 2006.

[5] OMG. Object Constraint Language (OCL) Version 2.0. Technical Report formal/2006-05-01, OMG, 2006.

[6] OMG. OMG Systems Modeling Language (OMG SysML) Version 1.1. Technical Report formal/2008-11-02, OMG, 2008.

[7] OMG. OMG Unified Modeling Language (OMG UML), Superstructure und Infrastructure. Technical Report formal/2009-02-02, OMG, 2009.

[8] Alan Moore und Rick Steiner Sanford Friedenthal. *A Practical Guide to SysML: The Systems Modeling Language.* Elsvier Inc., 2008.

[9] Joachim Fischer und Klaus Ahrens. *Objektorientierte Prozeßsimulation in C++.* Addison-Wesley (Deutschland) GmbH, 1996.

[10] Stephen J. Mellor und Marc J. Balcer. *Executable UML: A Foundation for Model-Driven Architecture.* Addison-Wesley Professional, 2008.

[11] Tim Weilkiens. *Systems Engineering mit SysML/UML.* dpunkt.verlag GmbH, 2006.

Abbildungsverzeichnis