Author

Title

Sub/Work Title

University of Applied Sciences: Vorarlberg

Department of Computer Science
Professor / Supervisor

Dornbirn, 2014

Abstract

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

Abstract iii

1 Einführung 1
- 1.1 Was ist Software? . . . 1
- 1.2 Beobachtungen von Modellen . . . 1
- 1.3 Probleme von Modellen . . . 1
- 1.4 Wärs nicht schön wenn ...? . . . 2
- 1.5 Sinnhaftigkeit . . . 2
- 1.6 Modellkontext . . . 2
- 1.7 Model Scope . . . 2
- 1.8 Modelzweck . . . 3
- 1.9 Modelvollständigkeit . . . 3
- 1.10 Modelltypen . . . 3
- 1.11 Metamodell-Architektur . . . 4

2 Einführung in ALF und fUML 5
- 2.1 UML-Model: Ordering . . . 5
- 2.2 State-Diagramm: Ordering . . . 6
- 2.3 Executable UML Foundation (fUML) . . . 6
  - 2.3.1 Key Komponenten . . . 7
- 2.4 UML Action Language (Alf) . . . 7
  - 2.4.1 Key Komponenten . . . 7
  - 2.4.2 Elements of Executable UML . . . 8

3 Zusammenfassung 11

Literaturverzeichnis 13

Listings

Abbildungsverzeichnis

1.1 Folien Bernd Wenzel 3
1.2 Folien Bernd Wenzel 4

2.1 Bestellung Klassen-Diagramm 5
2.2 Bestellung State-Diagramm 6
2.3 Beispiel Ausführbares UML 9

Todo list

1 Einführung

1.1 Was ist Software?

Artifakte im Kontext einer Domäne (eine Beschreibung davon was wir machen) und im Kontext einer Technologie (eine Beschreibung davon wie wir etwas erreichen). Seit 1950 versucht man die Technologie zu abstrahieren um sich auf die jeweilige Domäne konzentrieren zu können (OPCodes, Speichermanagement, ...). Jede zusätzliche Abstraktionsschicht hat allerdings auch neue Probleme mit sich gebracht z.B. schlechtere Performanz, größerer Speicherbedarf, etc.

Mit der Zeit wurden diese Probleme beseitigt und auch die Hardware hat sich dementsprechened weiterentwickelt (Speichergröße, CPU-Leistung, ...).

1.2 Beobachtungen von Modellen

- auch Quellcode ist ein Model
- Graphische Modelle
  - Kontrollflussmodelle
  - Bachmanndiagramme
  - UML
  - ...

Modelle werde verwendet um Anforderungen zu Beschreiben, eine Situation zu erfassen oder inzwischen auch um Quellcode daraus zu genieren.

1.3 Probleme von Modellen

Oft erkennt man nicht den Nutzen eines Modells (oder auch den Grad der Vollständigkeit). Außerdem können Modelle nicht nur richtig oder falsch sein sondern auch irrelevant.

1.4 Wärs nicht schön wenn ...?

... es eine Umgebung gibt in welcher man

- aus einem Modell eine vollständige Software generieren könnte
- ein Model ausführen kann ohne dafür Code generiern zu müssen
- Plattformunabhängig ist und somit wiederverwendbar

1.5 Sinnhaftigkeit

- die Möglichkeit Modelle in Code umwandeln gibt es breits
- Transformation von abstrakten Code zu einem konkreten Model wird breits von Kompilern gemacht
- Parametrisierung von Modelltransformationen wird von STP und den meisten Kompilern unterstützt

1.6 Modellkontext

Klassifizierung von Artifakten anhand von Domänen

- IT Domäne
- Applikationsdomäne (Bank, Regierung, Kommerzielles)

1.7 Model Scope

- Strukturelles Modelle (Klassenmodell, Komponentemodell, ...)
- Verhaltens Modelle (Usecases)
- Installations Modelle

1.8 Modelzweck

- Illustrationsmodell (Verständlichkeit)
- Anforderungmodell (Anforderungen an einen Zustand)
- Analysemodell (Repräsentation eines Zustands)
- Designmodell (Zustand den man haben möchte)

1.9 Modelvollständigkeit

Klassifizierung anhand des Grades an Vollständigkeit

- Unspezifisch
- Funktion-Vollständigkeit (z.B. alle Operationen in einem UML-Modell aber nicht zwangsläufig eine Implementierung)
- Resourcen-Vollständigkeit (Vollständig im Sinne von Ressourcen die funktionale Anforderungen implementiert haben)

1.10 Modelltypen

Model Type	Scope	Context	Purpose	Completen.
Abstract Business Proc. Model	Applic. Domain	Process / Act.	Requirements	functionally complete
Concrete Business Model	Applic. Domain	Structure and/or Behaviour	Analysis or design	resource complete
Ideal PIM	IT Domain	Structure and/or Behaviour	Design	functionally complete
Ideal PSM	IT Domain	Structure and/or Behaviour	Design	resource complete

Abbildung 1.1: Folien Bernd Wenzel

1.11 Metamodell-Architektur

- Model - Formale Definition von Struktur, Verhalten und Einschränkungen für eine Gruppe von Objekten (in einem Moodell) gelten.
- Metamodell - Formale Definition von Struktur, Verhalten und Einschränkungen für eine Gruppe von Modellelementen.
- Metalevel - M0 (Daten), M1 (Modelle - Datenbanken), M2 (Metamodelle - Programmiersprache, UML), M3 (Metametamodelle - Selbstbeschreibend)

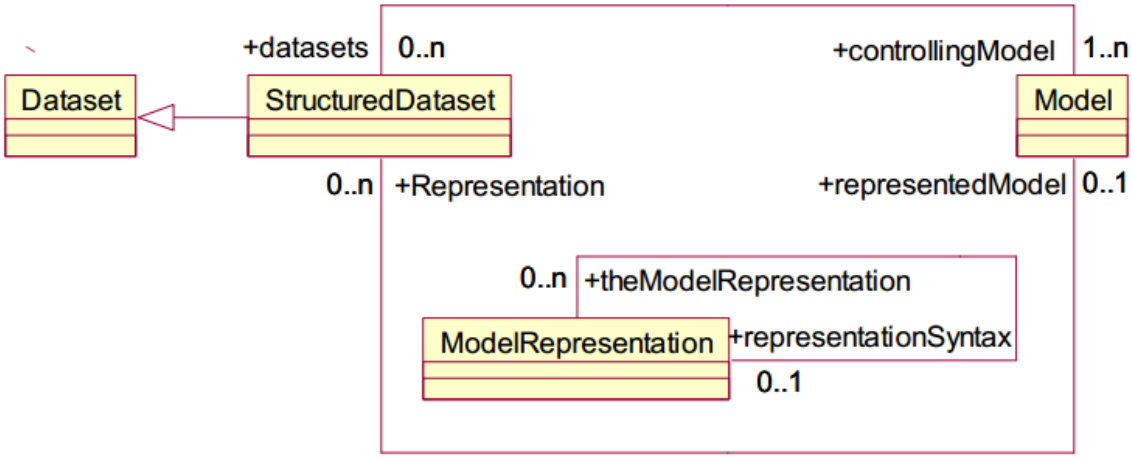

Abbildung 1.2: Folien Bernd Wenzel

2 Einführung in ALF und fUML

Motivation

- E-Commerce Ordering System
- Designed in UML
- Implementiert im Web (Programmiersprachen unabhängig

2.1 UML-Model: Ordering

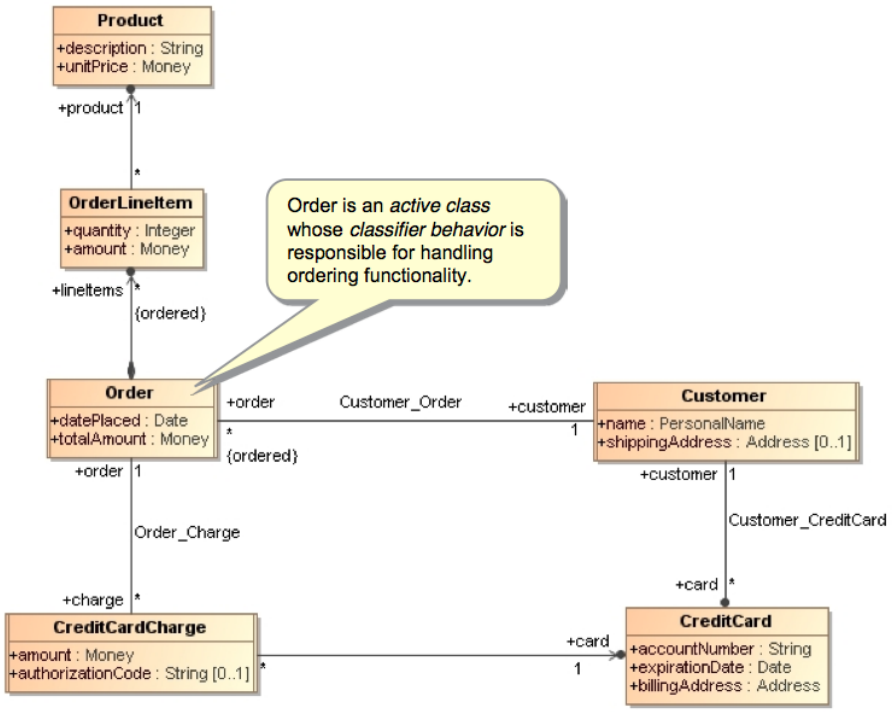

Abbildung 2.1: Bestellung Klassen-Diagramm

In Abbildung 2.1 wird ein UML-Diagramm verwaltung von Bestellungen dargestellt. Die Klasse Order ist Verantwortlich für die Sortierung.

Anmerkung zu UML: Der Punkt beim Pfeil: Das OrderLineItem kann direkt auf das Produkt zugreifen aber umgekehrt nicht. Im OrderLineItem ist ein Property product aber im Product ist keine Liste von OrderLineItem. Nur ein Pfeil ist was ähnliches aber nicht so strikt.

2.2 State-Diagramm: Ordering

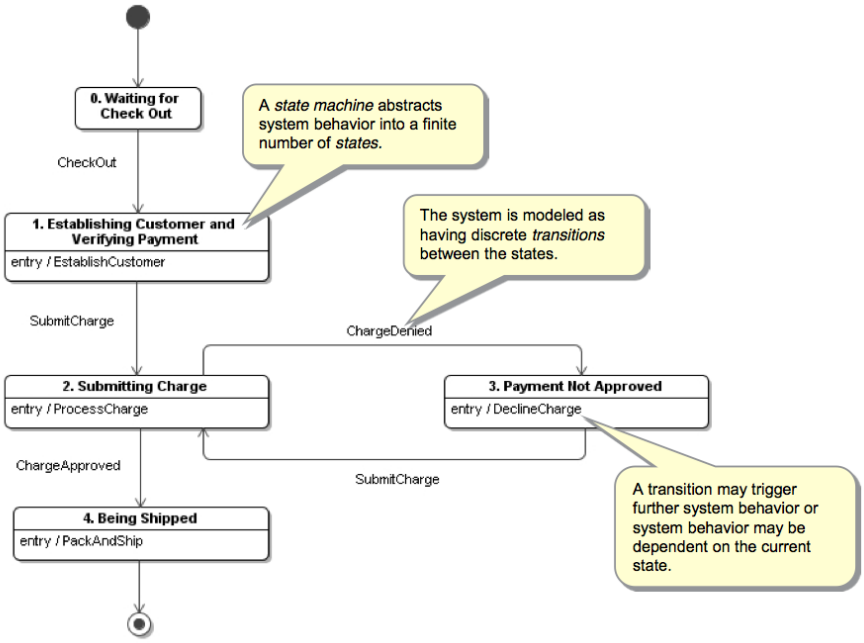

Abbildung 2.2: Bestellung State-Diagramm

In Abbildung 2.2 ist das Zustandsdiagramm einer Bestellung dargestellt. Die Schleife bei 2 und 3 besagt, dass der User aufgefordert wird die Zahlungsmöglichkeit auszuwählen. Fals das ncht klappt wird noch einmal gefragt. Es fehlt hier eine Abbruchbedinung.

2.3 Executable UML Foundation (fUML)

Foundational UML (fUML) is an executable subset of standard UML that can be used to define, in an operational style, the structural and behavioral semantics of systems.

Ziele:

- UML ist nicht genugt spezifiziert um ausführbar zu sein
- fUML präzisiert ein ausführbares subset von UML
- Graphiches Modellierung ist nicht genug programmierung
- ALF spezifiziert eine Action Language für UML

2.3.1 Key Komponenten

Foundational UML Subset (fUML) – A computationally complete subset of the abstract syntax of UML (Version 2.3)

- Kernel–Basicobject-orientedcapabilities
- Common Behavior – General behavior and asynchronous communication
- Activities – Activity modeling, including structured activities (but not including variables, exceptions, swimlanes, streaming or other "higher level" activity modeling)

Execution Model – A model of the execution semantics of user models within the fUML subset

Foundational Model Library

- Primitive Types – Boolean, String, Integer, Unlimited Natural
- Primitive Behaviors – Boolean, String and Arithmetic Functions
- Basic Input/Output – Based on the concept of "Channels"

2.4 UML Action Language (Alf)

The Action Language for Foundational UML (Alf) is a textual surface representation for UML modeling elements with the primary purpose of acting as the surface notation for specifying executable (fUML) behaviors within an overall graphical UML model. (But which also provides an extended notation for structural modeling within the fUML subset.)

2.4.1 Key Komponenten

Concrete Syntax – A BNF specification of the legal textual syntax of the Alf language.

Abstract Syntax – A MOF metamodel of the abstract syntax tree that is synthesized during parsing of an Alf text, with additional derived attributes and constraints that specify the static semantic analysis of that text.

Semantics – The semantics of Alf are defined by mapping the Alf abstract syntax metamodel to the fUML abstract syntax metamodel.

Standard Model Library

- From the fUML Foundational Model Library

Primitive Types (plus Natural and Bit String)

Primitive Behaviors (plus Bit String Functions and Sequence Functions)

Basic Input/Output

- Collection Functions – Similar to OCL collection operations for sequences
- Collection Classes – Set, Ordered Set, Bag (Stack), List, Queue, Deque, Map

2.4.2 Elements of Executable UML

Abbildung 2.3 zeigt ein Beispiel für ein Ausführbares UML.

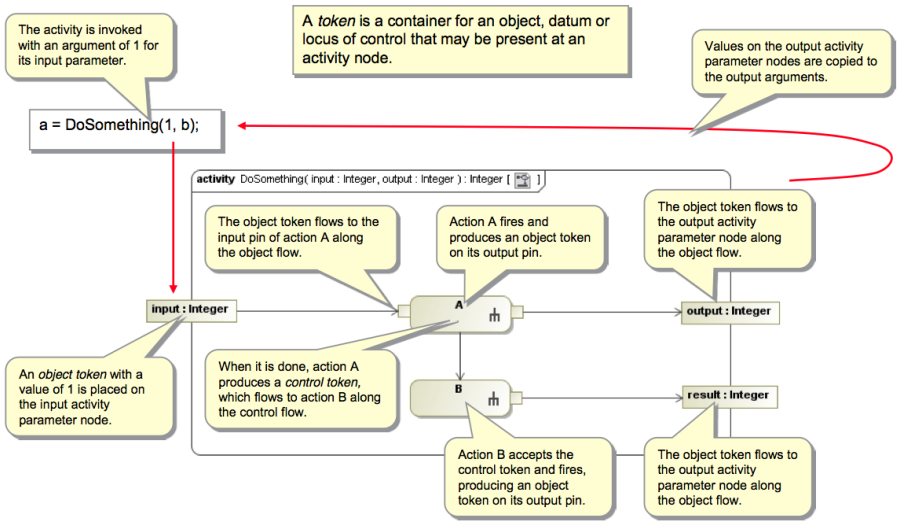

Abbildung 2.3: Beispiel Ausführbares UML

3 Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Literaturverzeichnis