

Projektgruppe



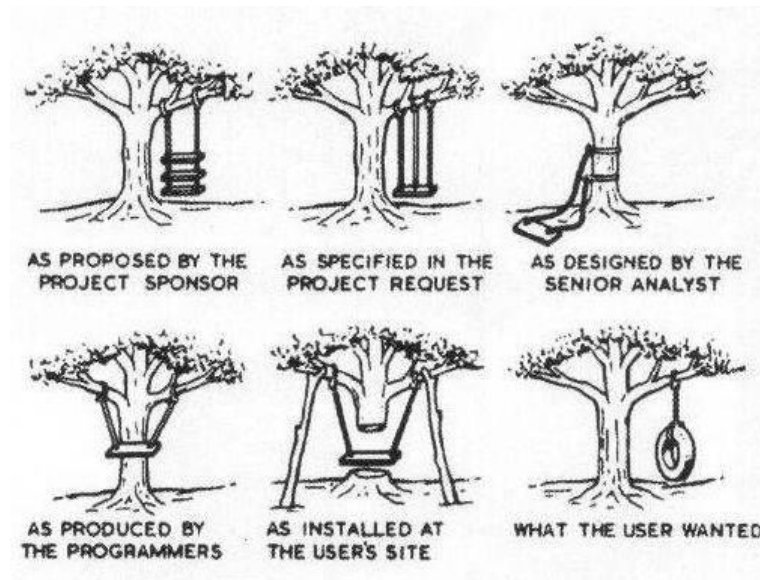
Bearbeiter: **Othmane Rhandor**

Automatische Modellgenerierung aus Systemspezifikationen

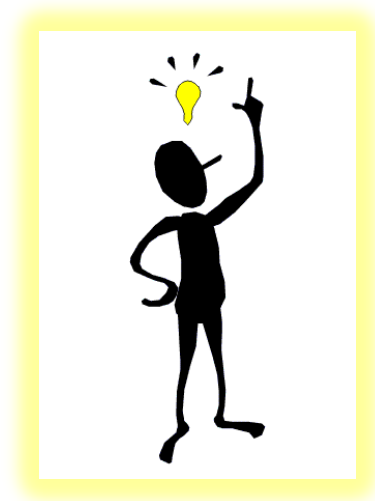
- Motivation
- Ansätze
- MOF und EMF
- Zusammenfassung

Motivation

- Systemspezifikationen (Anforderungen) in natürlicher Sprachen (NL)
 - Teilweise subjektiv → viel Interpretationsraum
 - Erschwert die Kommunikation



→ Verzögern den gesamten Entwicklungsprozess



Motivation

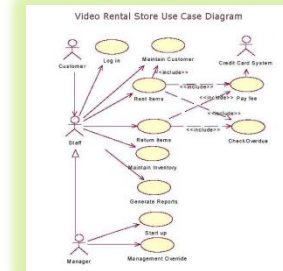
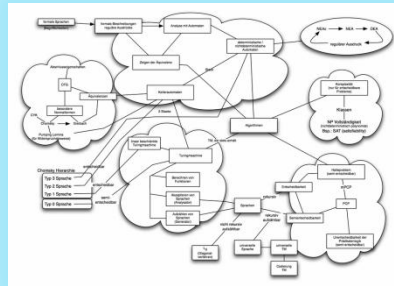
Ziel: Automatische Modellgenerierung aus Anforderungsdokument

Anforderungsdokument
Kategorie: Funktionale Anforderungen (Use Cases)

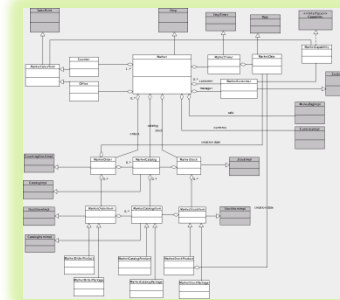
2.3 Use Case: Benutzer registrieren und einsehen
Vorbedingungen
 Website des BIC wurde aufgerufen und die Option „Registrierung“ oder „Login“ für bereits registrierte Benutzer angeklickt.
Nachbedingungen
 Der Benutzer hat ein vollständig ausgefülltes Benutzerdatenprofil erstellt oder die Benutzerdaten eingesehen.
Nichtfunktionale Anforderungen
 Keine
Rollenszenario
 Dieser Use Case beginnt mit dem Erstellen eines Benutzerprofils (S-1) oder – wenn der Benutzer bereits registriert ist – durch Eingabe von Benutzername und Passwort und Klick auf „Login“. Bei einer korrekten Eingabe von Benutzernamen und Passwort werden dem Benutzer das aktuelle Guthaben und die zum Verkauf angebotenen Bücher angezeigt (S-2). Danach kann er mittels des Links „Benutzerdaten einsehen“ sein persönliches Profil überprüfen (S-3).
 In (S-1) überprüft das System ob die Benutzerdaten vollständig sind (E-1) und die beiden Passwortfelder übereinstimmen (E-2). Anschließend wird der Benutzer darüber informiert, dass er in Kürze ein E-Mail mit der Anmeldebestätigung, dem Benutzernamen, dem aktuellen Kontostand und einem elektronischen Einloggschlüssel erhält.
 Bei (S-2) prüft das System ob die eingetragene Benutzererkennung gültig ist (E-3).
Umsatzstrategien
 (S-1) Benutzerdaten registrieren
 Die Registrierung erfolgt mittels Eingabe von Name, Vorname, Adresse, PLZ, Ort, Studiengang, E-Mail, Passwort und Login No.
 (S-2) Bücherdaten einsehen: Die Bücher werden tabellarisch angezeigt. Konkret werden Autor, Titel, Eintrags- und Verkaufsdatum sowie der Verkaufspreis angezeigt. Das Verkaufsdatum wir in den letzten 10 Verkaufstagen mit abgebildet. Zudem sieht der User auch den aktuellen Kontostand.
 (S-3) Benutzerdaten einsehen
 Beim Benutzerprofil werden dieselben Daten angezeigt, welche bereits bei der Registrierung eingetragen wurden. Eine direkte Änderung der Benutzerdaten ist nicht möglich. Will der User seine Daten geändert haben, kann er mittels Klick auf den Button „Mail to Admin“ ein Mail mit den Änderungswünschen verschicken.
Reihenfolgen
 (F-1) Unvollständige Benutzerdaten
 Beim Erstellen eines Benutzerprofils müssen alle Felder ausgefüllt werden. Die Unvollständigkeit kann der Benutzer seine Daten kompletieren oder den Use Case beenden.
 (F-2) Passwort stimmt nicht überein
 Wenn nicht in beiden Passwortfeldern dasselbe Wort eingegeben worden ist, oder eines der beiden Felder leer steht, hat der Benutzer die Möglichkeit, seine Eingabe zu ändern oder den Use Case zu beenden.

© HTW Chur 2008, Invenio 10/25

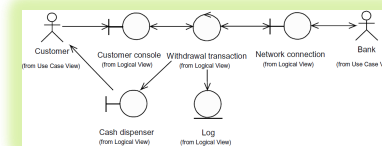
Werkzeug / Methodik



Use Case Diagramm



Klassen-Diagramm



Analyse-Diagramm

Anforderungen in NL



- Motivation
- **Ansätze**
- MOF und EMF
- Zusammenfassung

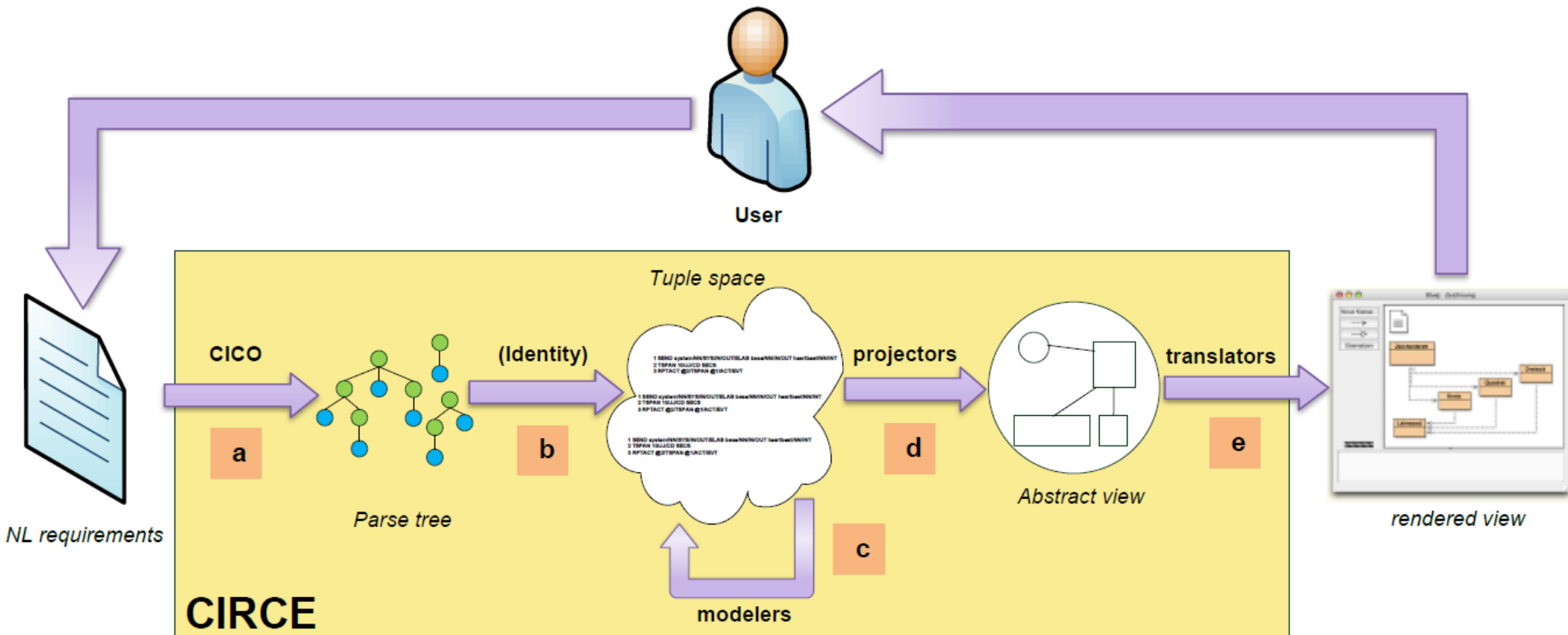
- **Ansatz 1:** Cooperative Interactive Requirements-Centered Environment (CIRCE)
 - Entwickelt von *Vincenzo Gervasi* (Uni Pisa, Italien)
 - Webbasierter Werkzeug
- **Ansatz 2:** Use Case driven Development Assistant (UCDA)
 - Entwickelt von *Dong Liu* (Uni Calgary, Canada)
 - Plug-In für IBM Rational Rose



→ **Methodik!**

Ansatz: CIRCE

- Architektur:



Definitionen

- Anforderungsdokument $D = \langle G, F, R \rangle$
 - **G** = Glossar
 - Domainspezifische Begriffe und Entitäten
 - Synonyme und Metonymien
 - Begriffe markiert (Tags)
 - **F** = Definitionen
 - Kürzel der Anforderungen
 - Bringt Text in kanonischer Form
 - **R** = Anforderungen
 - Erwartetes Systemverhalten
 - Dient als Input für den Parser



NL requirements

Beispiele

- Requirement :: **R**

Wenn das Wasser kocht, dann muss der Wasserkocher innerhalb 5 sec ein Signal zum Abschalten ausgelösen.

- Definitions :: **F**

SCHALTE x AB → SENDE EIN ABCHALTE SIGNAL AN x

- Glossars :: **G**

- Formal: $\{(\tau, \{\tau_i\})\}$

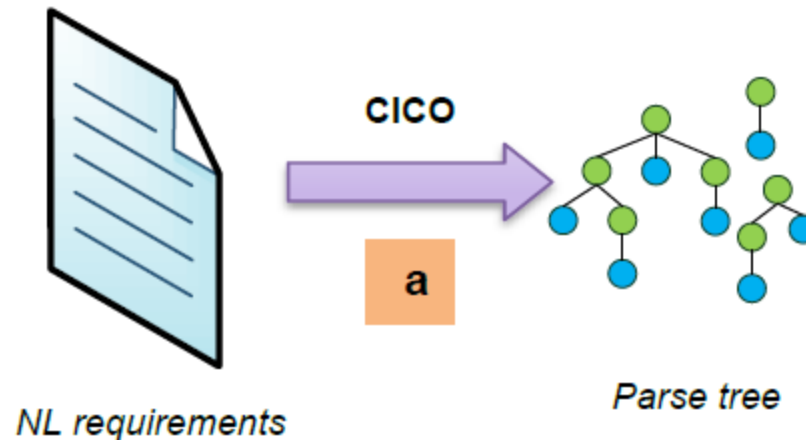
- τ : Menge der Begriff

- τ_i : Synonyme

- Bsp.: $G = \{(\text{System/IN/OUT}, \{\text{Wasserkocher}, \text{User}\})\}$

(a) CICO

- **Input**: Dokument in kanonischer Form



- Eigenschaften:
 - Einfacher Domain-basierter Parser
 - Fuzzy-Matching (templates) anstatt Sprachgrammatik
 - Model-Action-Substitution (MAS)
 - Benutzung Optimierungsheuristiken

(a) CICO – Vorgehensweise

- Morphosyntaktische Analyse – *part-of-speech* (POS)
 - Benutzte Variante: Penn Treebank tagset
 - **Ergebnis:** $s = \langle w_1, \dots, w_n \rangle$
- Model-Action-Substitution Regeln (MAS): $\rho = (\mu, \alpha, \sigma)$
 - Regeln sind frei programmierbar
- Matching-Funktion $match(\rho, s)$

(a) CICO – Beispiel

- $\rho = (\mu, \alpha, \sigma)$

$\mu = \text{snd/OUT SEND data/INF TO rcv/IN}$

$\alpha = \text{SEND \$snd \$rcv \$data}$

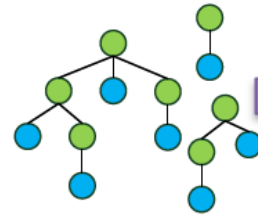
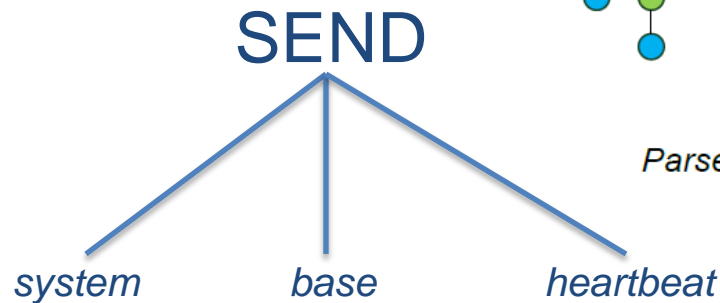
$\sigma = \text{\$ID/ACT/EVT}$

- *Init text = the system shall send a heartbeat to the base*
- $s = \text{the/DT system/NN/SYS/IN/OUT/ELAB shall/MD send/VB}$
 $\text{a/DT heartbeat/NN/INF to/TO the/DT base/NN/IN/OUT}$

$\text{match}(\rho, s) = \langle \underbrace{\{\text{system, base}\}}_{\text{snd}}, \text{SEND}, \underbrace{\{\text{heartbeat}\}}_{\text{data}}, \text{TO}, \underbrace{\{\text{system, base}\}}_{\text{rcv}} \rangle$

(a) Parsen durch CICO

- $\alpha = SEND \$snd \$recv \$data$

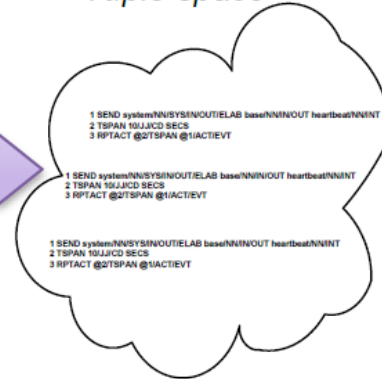


Parse tree

(Identity)

b

Tuple space



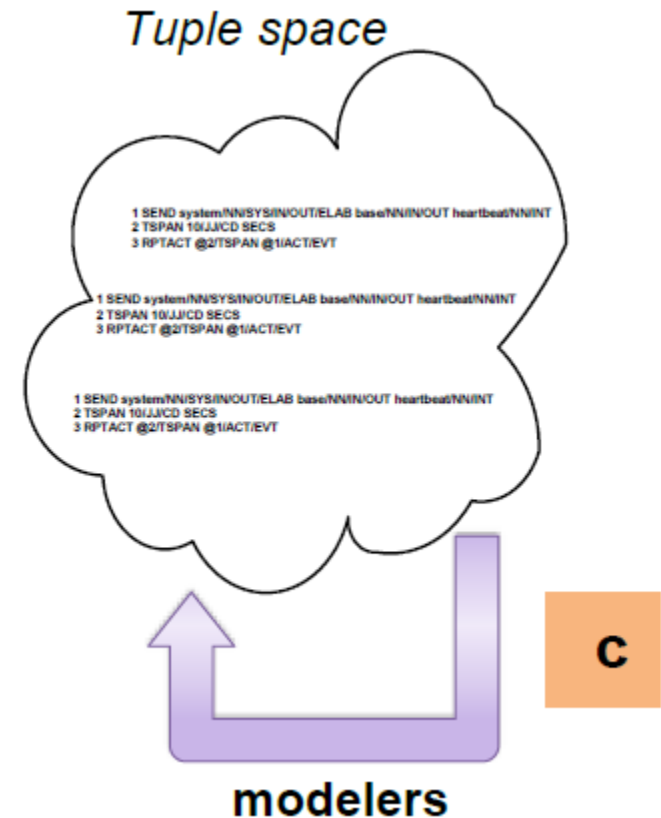
- $\sigma = \$ID/ACT/EVT$

- Codierter Tupel:**

→1 SEND system/NN/SYS/IN/OUT/ELAB base/NN/IN/OUT heartbeat/NN/INT

(c) Modelers

- Erzeugen neue Tupel
- Intensionale Wissen über das System
- Modelers Strunktur: $M = \langle i, o, f \rangle$
 - $i = \text{input tuples}$
 - $o = \text{output tuples}$
 - $f = \text{Logik als Funktion}$
- *Modelers Typen Beispiele:*
 - *Abstraction Modelers*
 - *Intensional Modelers*
 - *Validierungsmodelers*



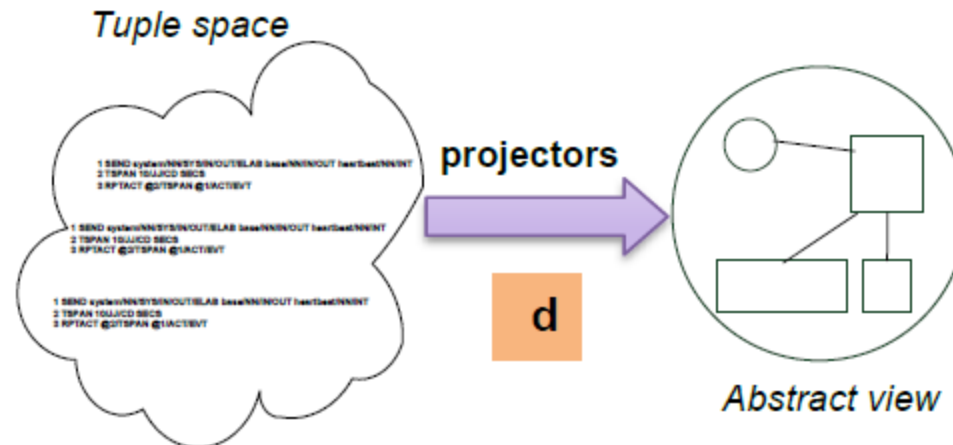
(c) Modelers – Beispiel

Abstraction Modeler:

$$m_{iface} = \langle \{SEND, RECEIVE\}, \{IIFACE, OIFACE\}, f \rangle$$

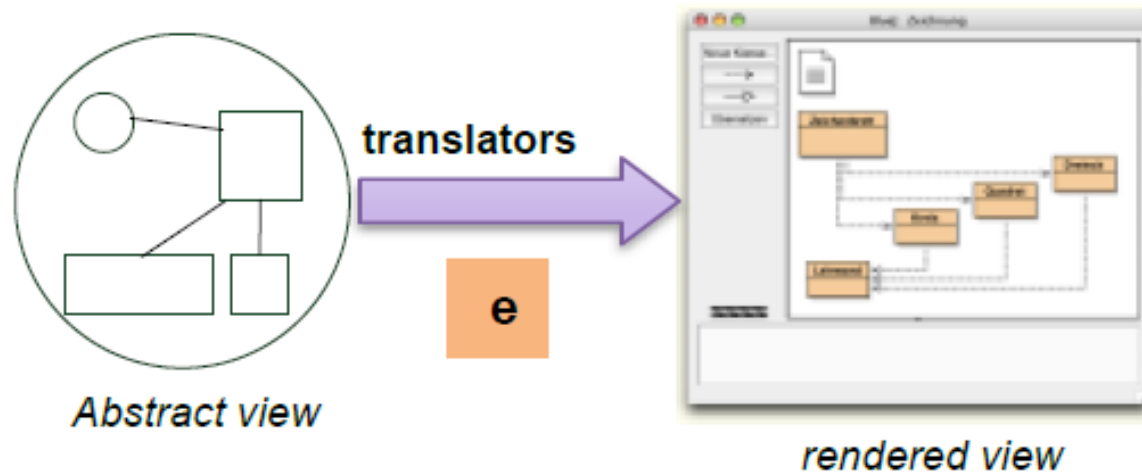
- **SEND, RECEIVE:** Kommunikationsaktionen
- **IIFACE, OIFACE:** System Schnittstelle Tupeltyp
- Funktion **f**: beschreibt die Logik
 - $iiface(src, dst) \leftarrow send(src, dst, data) \wedge \neg system(src) \wedge system(dst)$
 - $iiface(src, dst) \leftarrow receive(dst, src, data) \wedge \neg system(src) \wedge system(dst)$
 - $oiface(src, dst) \leftarrow send(src, dst, data) \wedge system(src) \wedge \neg system(dst)$
 - $oiface(src, dst) \leftarrow receive(dst, src, data) \wedge system(src) \wedge \neg system(dst)$

(d) Projectors



- Native Meta-Model
- Typische Tags:
 - /ENTITY: physikalische und logische Entitäten
 - /ATTR: Attribute der Entitäten
 - Beispiel: Base/ENTITY, System/ENTITY

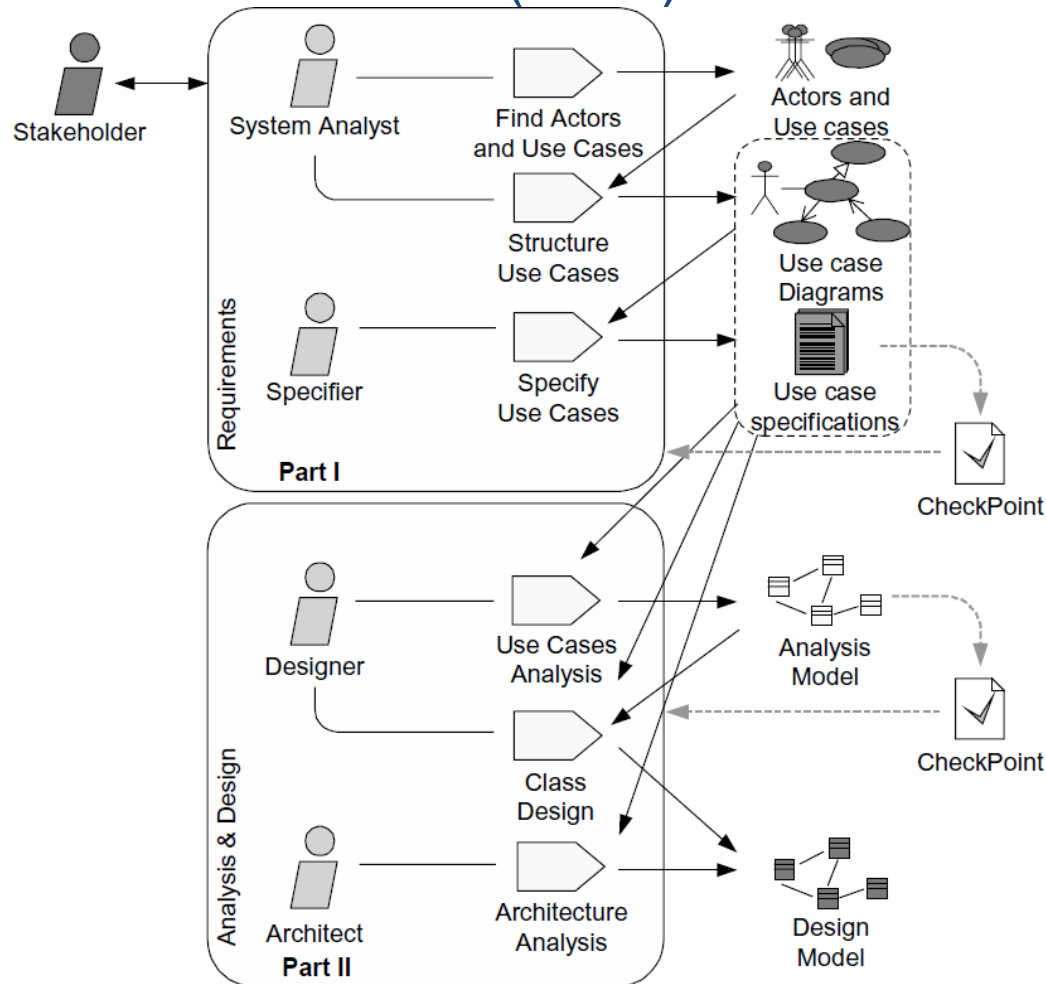
(e) Traslators



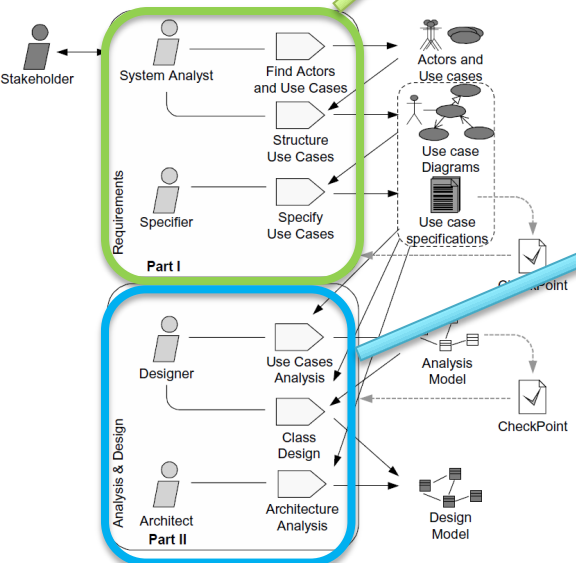
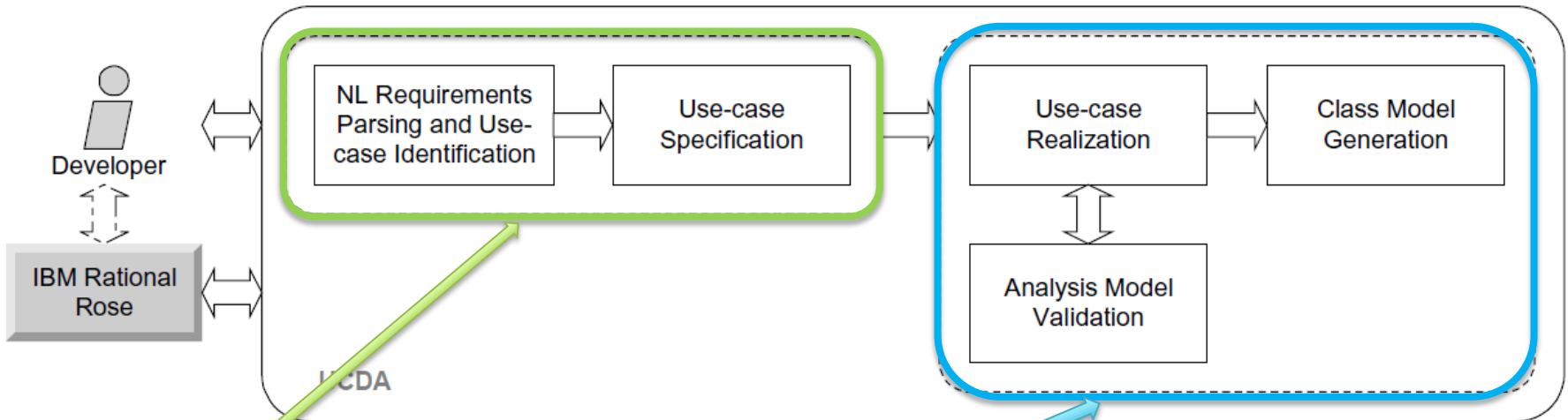
- Abbildung der abstrakten Sicht
- UML Modelle
- Exportieren als XMI

- Motivation
- Ansätze
 - CIRCE
 - **UCDA**
- MOF und EMF
- Zusammenfassung

Rational Unified Process (RUP):



UCDA – Architektur



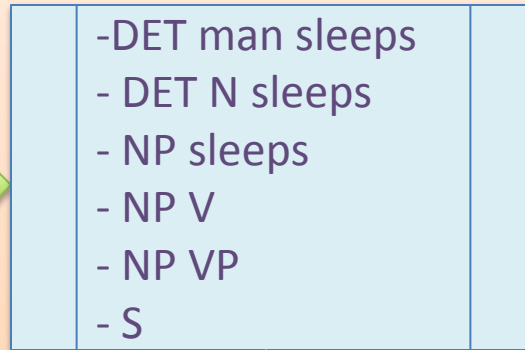
NL parsen

Kunden
Anforderungen
Dokument In NL



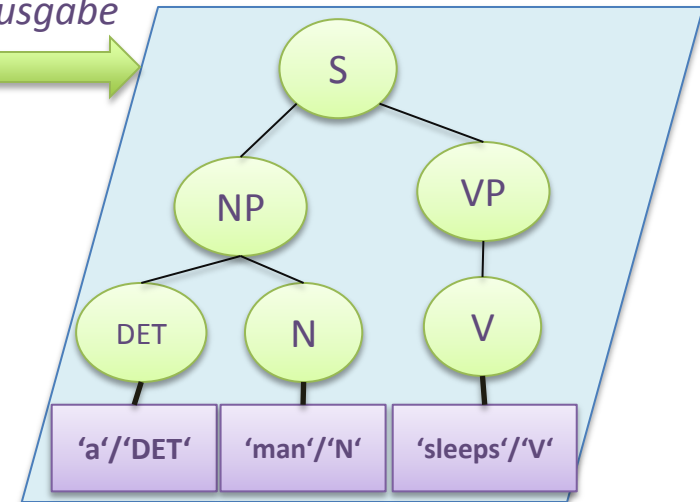
eingabe

shift-reduce-parser



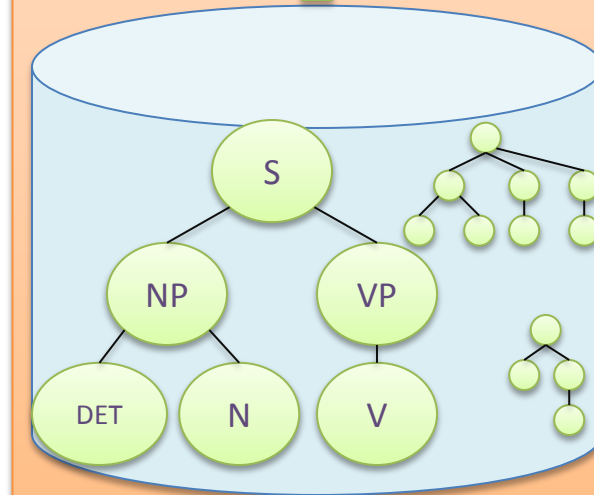
ausgabe

Abstrakte Repräsentation
der syntaktische Struktur
des Text



benutzt

Regel von typischen
Strukturelementen



- **Glossar als Voraussetzung:**
 - Vordefinierte **Aktoren** und **Use Cases**
 - **Wie:** Durch Verwendung von Heuristiken, wie z.B.:
 - Wer stellt zur Verfügung, benutzt oder löscht Informationen?
 - Wer benutzt die Funktionalität?
- **Aktoren** Kandidaten: Nomen (Substantive)
 - Aber: Aktoren die nicht in Glossar werden entfernt
- **Use Cases** Kandidaten: Verb Ausdrücke (Phrasen)

Use Cases identifizieren - Beispiel

Anforderungen einer Bankautomat.

„Der Automat bedient maximal einen Kunden zu jeder Zeitpunkt. Ein Kunde startet eine Sitzung sobald er Bankkarte einsteckt. Der Kunde wird dann in der Lage sein eine Transaktion durchzuführen.“

→ **Aktor** = „Kunde“

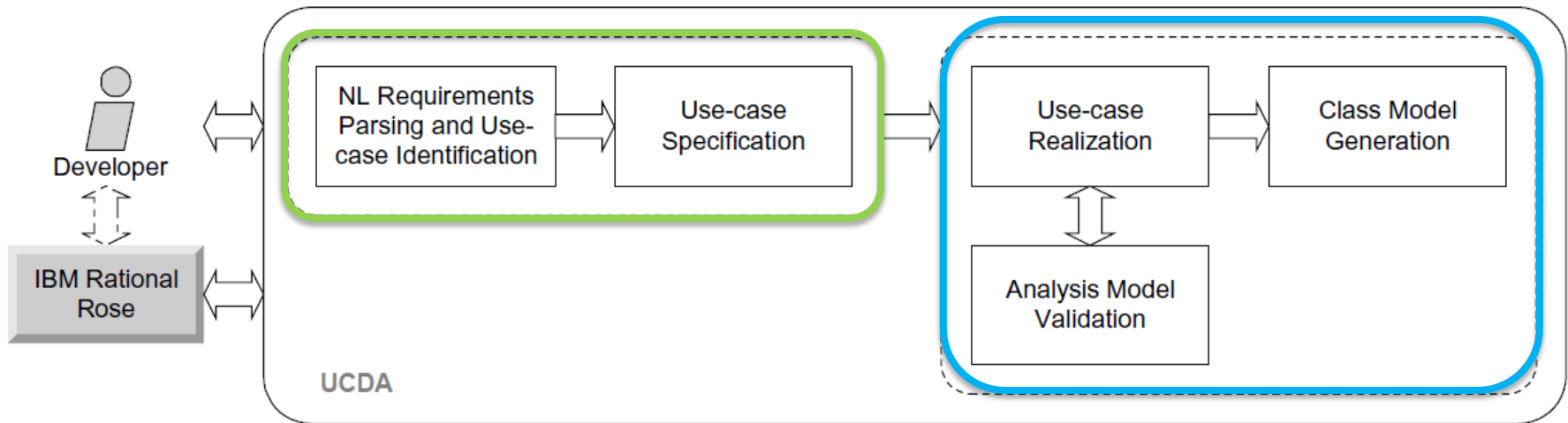
→ **Use Cases:** „Sitzung starten“ und „Transaktion durchführen“

Use Cases spezifizieren

- Basierend auf ein „Template“
 - Use Case name
 - event flow
 - pre-condition
 - Etc.
- Schema zum automatisieren der Use Cases Spezifikation
 - Basic
 - If-then
 - Do-until
 - Con-Noc

Use Cases realisieren, Analyse- und Klassenmodell generieren

Input: Aktoren, Use Cases und Use Case Spezifikation



Output: Aktoren, Use Cases und Use Case Spezifikation

Use Cases Realisierung:

- Struktur der Handlungsaussagen
- Verhaltenstypen des Systems
- Stereotypen der Objekte

: actor, : boundary object, : control object, : entity object

→ Potentielle Instanzen einer Klasse: Entitäten

- Vererbung und Komposition

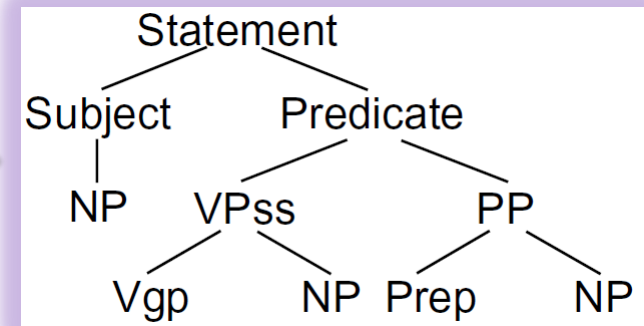
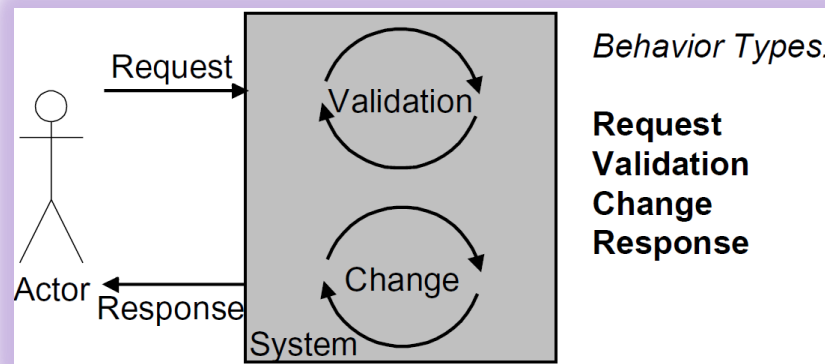
Use Cases realisieren, Analyse- und Klassenmodell generieren (3)



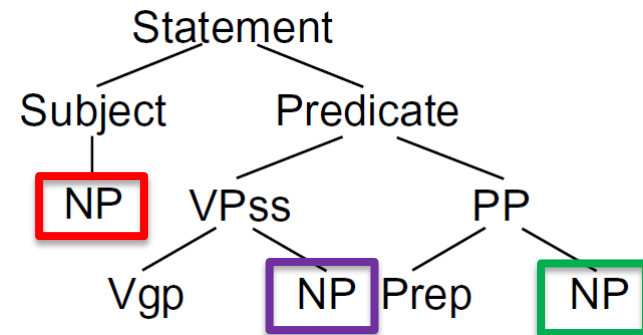
- Für jeden Use Case:
 - Analyseklassen, Stereotype → Robustheitsdiagramm
 - Das Systemverhalten zerlegen → Kollaborationsdiagramm
 - Für die Analyseklassen
 - Verantwortlichkeiten beschreiben
 - Verbindungen beschreiben
 - Die Vererbungsverbindungen erzeugen
- **Ja, aber Wie:** Anwenden von Regel!

Use Cases realisieren, Analyse- und Klassenmodell generieren (4)

- Insgesamt sind es 17 Regeln
 - Identifizierung von Objekte und Nachrichten
- Basierend auf den Satzstruktur und Systemverhalten



Beispiel: Transitiver Satzstruktur

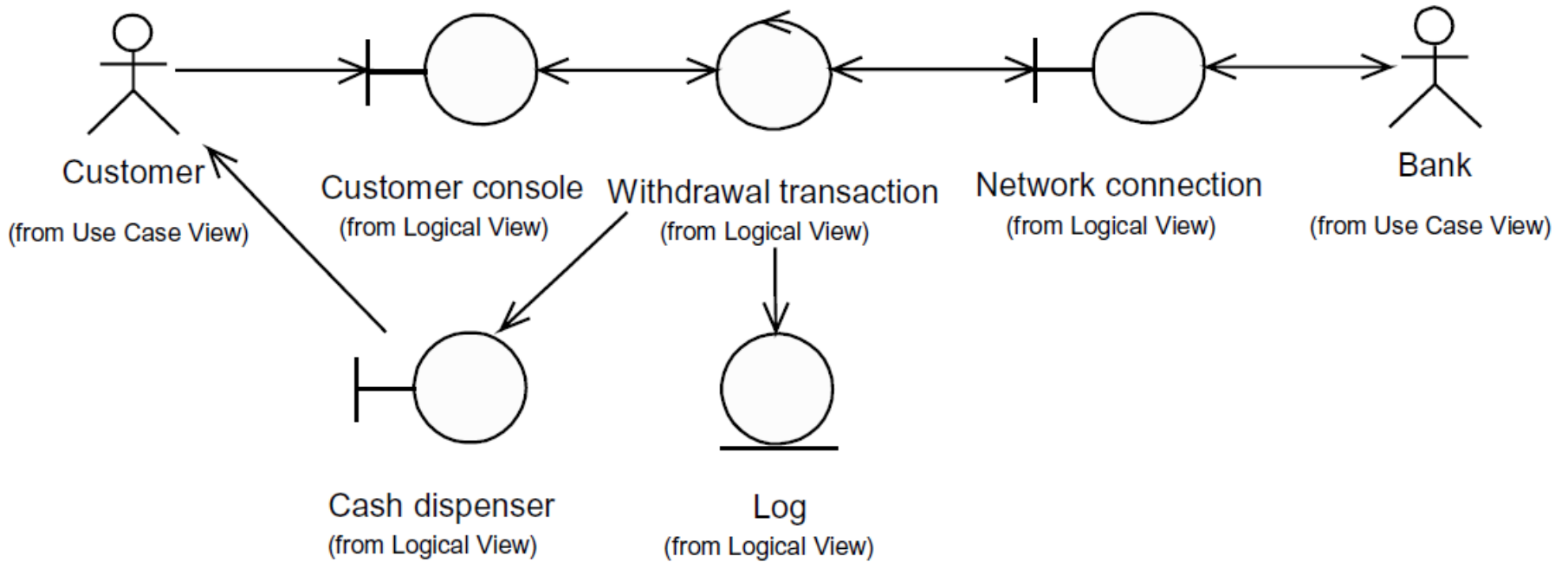


Regel:


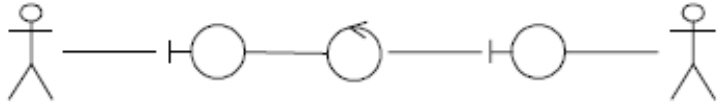
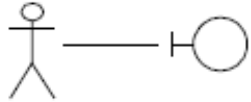
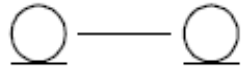

*”If the structure of a statement is transitive, and Subject/NP//Noun(head) is an **actor**, then this statement is corresponding to the **Request** behavior type and Predicate/PP/NP//Noun(head) is a **boundary** object if it exists in the glossary, and Predicate/VPss/NP/Noun(head) is an **entity** object if it exists in the glossary.“*

*Der **Benutzer** bestätigt die **Transaktion** auf der **Konsole**.*

Beispiel: Robustheitsdiagramm



Beispiel für Analyse Regel:

<i>Case</i>	<i>Validation</i>	<i>Suggestion</i>
	<i>Not allowed.</i>	
	<i>Allowed</i>	
	<i>Not allowed.</i>	

- Basierend auf eine systematische Methodik – RUP
- Architektur besteht aus zwei Teile
 - Teil 1: Requirments aus NL parsen, Use Cases identifizieren und spezifizieren
 - Teil 2: Use Cases realisieren, Klassenmodelle generieren und validieren
- Umsetzung als Plug-In für die IBM Rational Rose Entwicklungsumgebung
 - **XMI Export interssant für die PG!**

- Motivation
- Ansätze
- **MOF und EMF**
- Zusammenfassung

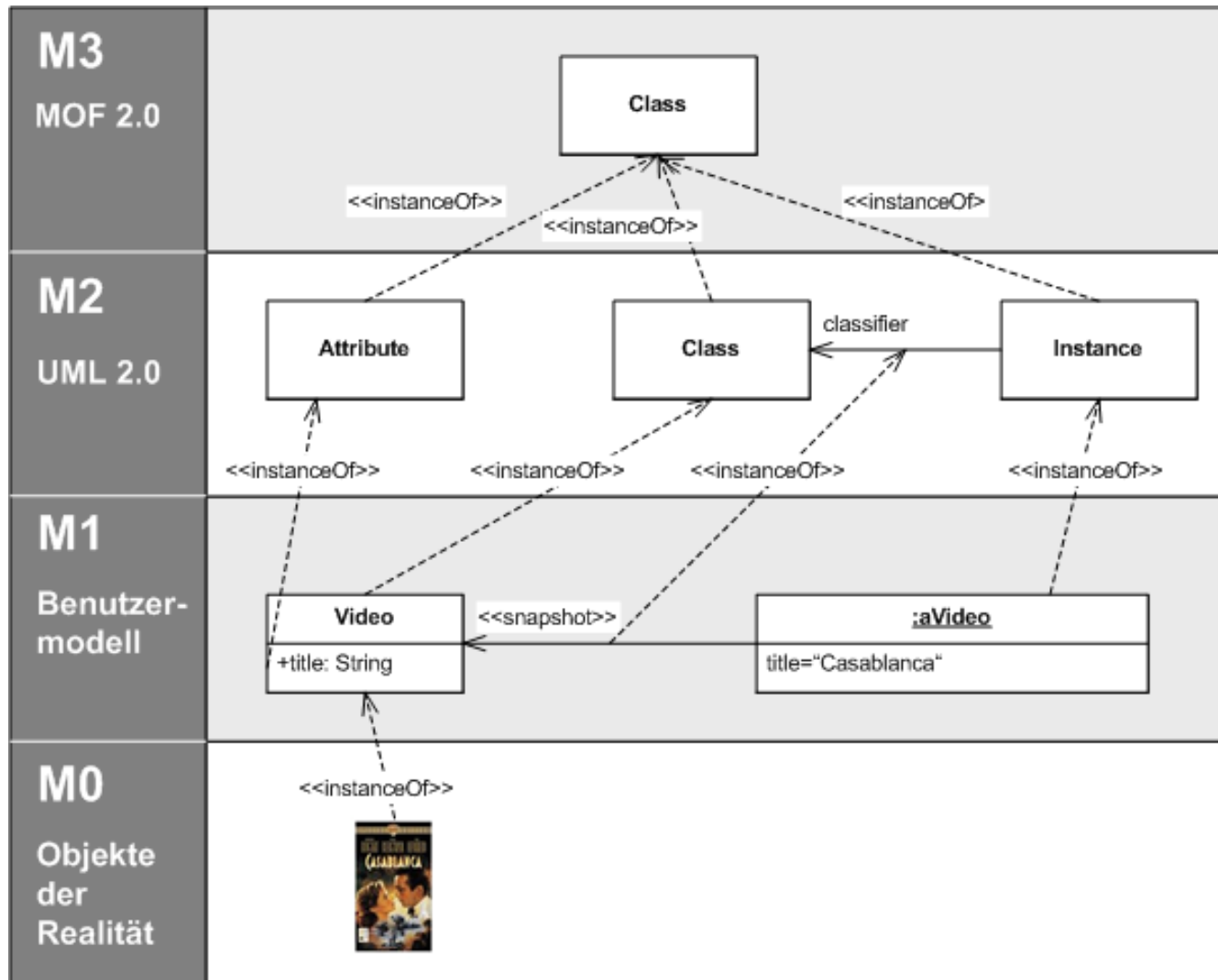
MOF und EMF



Meta-Object Facility (MOF)

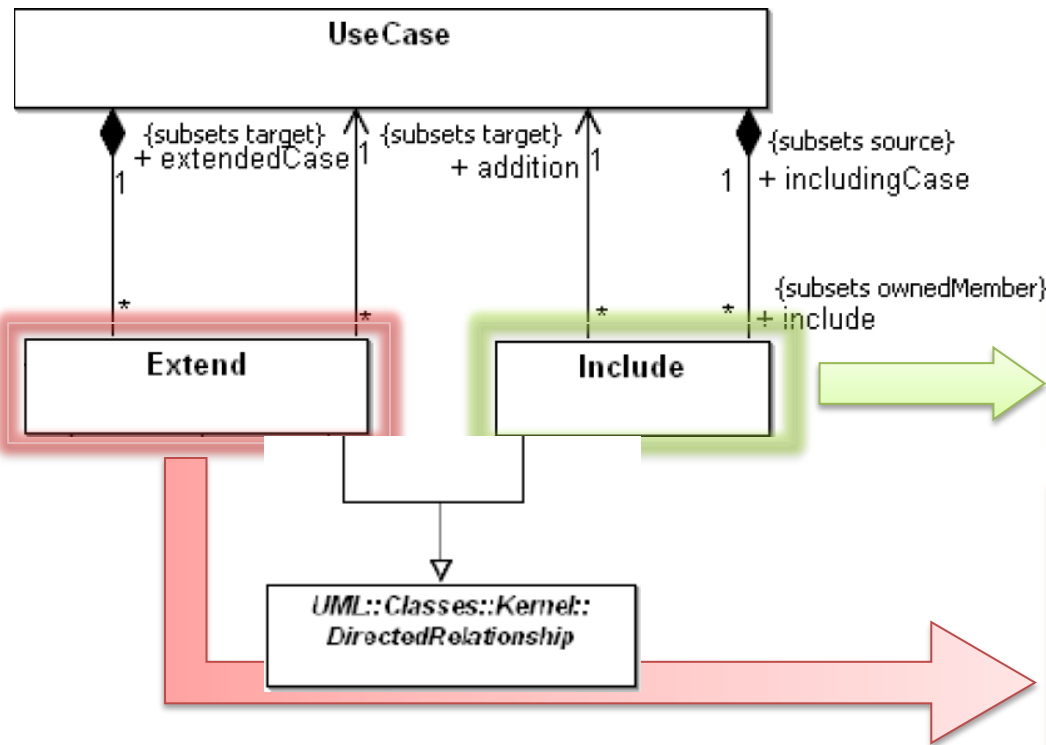
- Object Management Group™ (OMG™)
- Meta-Meta Sprache
 - Meta Modellierung Framework
 - Domain spezifische Sprachen
 - UML Modellierungssprache
- **XMI-Format als Metadaten Austauschformat**
 - Modelle die sich mit MOF ausdrücken lassen (z.B.: UML)



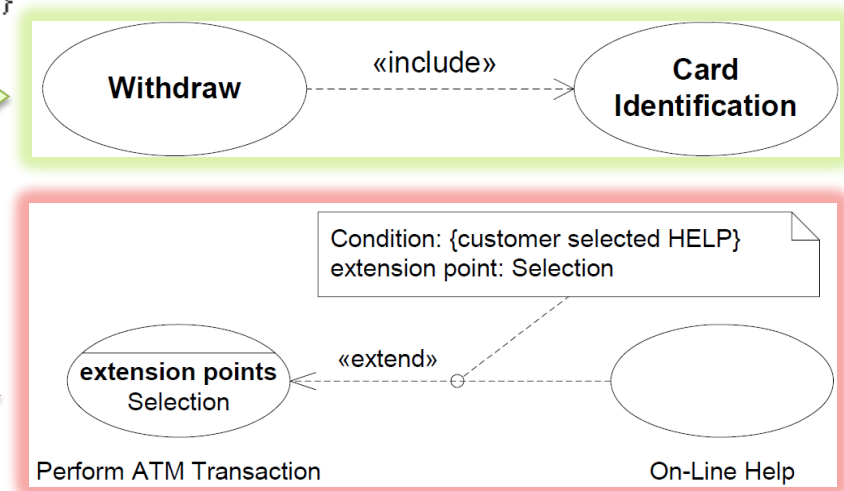


MOF Beispiel: Use Case Diagramm

UML Meta-Modell

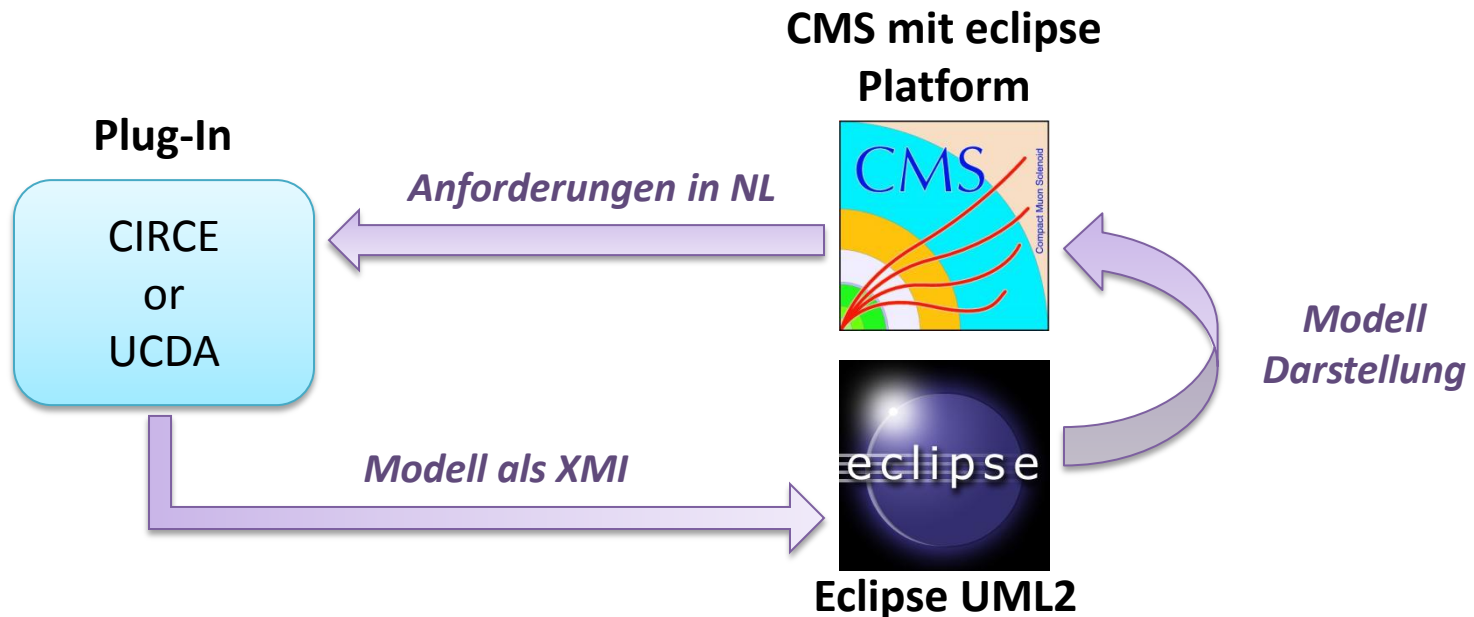


Modell Beispiel



Eclipse Modeling Framework (EMF)

- Meta-Meta Modell für eclipse Platform
- XMI-Format als Metadaten Austauschformat
- **Für die PG:** “*Eclipse UML2*“ als Umsetzung für OMG UML 2.x



- Motivation
- Ansätze
- MOF und EMF
- **Zusammenfassung**

- Problematik:
 - Anforderungen in NL formalisieren (Semi-formale Modelle)
- Vergleich der Lösungsprinzipien

Vergleichspunkt	CIRCE	UCDA
Modelltypen	Statische + Dynamische	Statische
Parsen	Domain-basiert	Sprachgrammatik-basiert
Integration	Nicht Möglich (in Bearbeitung)	Möglich (als Plug-In)
Portierbarkeit der Modell	XMI	XMI

- Modelle in CMS Integrieren
 - XMI als Austauschformat
 - Benutzer Integration für die Modellanalyse
 - EMF als Einsatzmöglichkeit
 - Eclipse UML2 als Umsetzung für UML
- Eclipse als Plattform für CMS

Vielen Dank für eure Aufmerksamkeit!

