

Vorlesung „Methoden des Software Engineering“

Block A „Requirements Engineering“ **Anforderungsmodellierung: Zielorientierte Methoden**

Martin Wirsing

Einheit A.3, 28.10.2004

Ziele

- Einführung in Zielorientierte Ansätze zu RE geben
- Die KAOS-Methode lernen
 - Unterschiedliche Arten von Zielen
 - Und/Oder-Verfeinerung zur Strukturierung von Zielen
 - Obstruktionsanalyse
 - Konflikterkennung
 - Vorgehensweise zu RE

Zielorientierung ...

- Kommt aus traditionellen Ansätzen der Systementwicklung („Context Analysis“, „Definition Study“, „Participative Analysis“)
- Behandelt in IEEE-Std-830
- Ignoriert durch UML, „wird aber benötigt“ sagen Fowler, Cockburn
- Zunehmend berücksichtigt in RE-Forschung

Zielorientierte Ansätze

- **Ansatz**
 - Zielt auf das „Warum Systeme konstruiert werden“
 - Die Gründe („Warum“) werden ausgedrückt als eine Menge von Zielen der Betroffenen (Stakeholder)
 - Verfeinerung von Zielen führt zu den speziellen Systemanforderungen
 - Zielanalyse
 - Dokumentation, Strukturierung und Klassifikation von Zielen
 - Zielevolution
 - Verfeinerung, Ausarbeitung und Operationalisierung von Zielen
 - Zielhierarchien zeigen Verfeinerungs- und Obstruktionsbeziehungen zwischen Zielen
- **Vorteile**
 - Eher intuitiver Ansatz
 - Explizite Deklaration von Zielen ermöglicht Erkennung und Lösung von Konflikten
- **Nachteile**
 - Änderung von Zielen nur schwer zu beherrschen

Zielorientierter Ansatz: Übersicht

- **Ziel**
 - Eigenschaft, die durch das System erreicht werden soll...
 - Beschreibung einer Absicht
 - „System“ = Software + Umgebung, aktuelles oder geplantes System
- **Anforderung**
 - Spezifiziert, wie ein Ziel durch das neue System erfüllt werden soll
- **Arten von Zielen**
 - Zu erreichende Ziele (achievement goal)
 - Zu bewahrende Eigenschaften (maintenance goal)
 - Weiche Ziele (soft goal)
- **Obstruktionen (Obstacle) und Constraints**
 - Eine Obstruktion ist ein Verhalten, das ein gegebenes Ziel verhindert
 - Ein Constraint ist eine Bedingung für das Erreichen eines Ziels
- **Tips**
 - Unterschiedliche Quellen liefern bessere Ziele
 - Verknüpfe Stakeholder mit jedem Ziel (zeigt Sichten und Konflikte)
 - Use Cases zur Realisierung von Zielen
 - Explizite Berücksichtigung von Obstruktionen hilft zur Erkennung von Ausnahmen

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Arten von Zielen

- **Unterschiedliche Abstraktionsniveaus**
 - Strategische, grobgranulare, organisationsweite Ziele
 - „mehr Fahrgäste bedienen“ (Zugkontrolle)
 - „effektiver Zugriff auf den aktuellen Stand der Wissenschaft“ (Bibliothekssystem)
 - Technische, feingranulare, designspezifische Ziele
 - „Kommando zur Geschwindigkeitsanpassung alle 3 Sek. senden“ (Zugkontrolle)
 - „Mahnung versenden am Ende der Ausleihperiode, falls das Buch nicht zurückgegeben wurde“ (Bibliothekssystem)
- **Funktionale/Nichtfunktionale Ziele**
 - Funktionale Ziele für erwartete Dienste
 - „Zugbeschleunigung berechnen“
 - „Buchbestellung erfüllen“
 - Nichtfunktionale Ziele für Dienstqualität: Sicherheit, Performanz, Kosten, Bedienbarkeit, ...
 - „Minimale Bremsdistanz einhalten“ (Safety)
 - „Zugriff auf Info über Entleiher verhindern“ (Security)
 - Nichtfunktionale Ziele für Entwicklungsqualität: Anpassbarkeit, Interoperailität, Wiederverwendbarkeit, ...

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Ziele und Stakeholder

- **Zielerreichung erfordert die Zusammenarbeit der beteiligten Akteure**
 - „Sicherer Transport“: On-board Zugkontrollsystem, Zugpositionsbestimmungssystem, Stationscomputer, Fahrgäste, Zugführer, ...
 - „Buchexemplar zurückgeben“: Ausleiher, Bibliothekspersonal und -Software
 - Je feingranularer ein Ziel, desto weniger beteiligte Akteure:
 - „Kommando zur Geschwindigkeitsanpassung alle 3 Sek. senden“: Stationscomputer
 - „Mahnung versenden am Ende der Ausleihperiode“: Bibliothekssoftware
- **Ziele von Beteiligten sind mögliche Quelle für Konflikte**
 - „Buch nach 2 Wochen zurückgeben“: Bibliothekspersonal
 - „Buch so lange wie möglich behalten“: Ausleiher

Methoden zur Modellierung von Zielen

- **URN („User Requirement Notation“)**
 - Nachfolger von i* (Yu, Mylopoulos)
 - Standardisierung durch ITU-T erwartet 2004
 - ITU Study Group: D. Amyot, G. Mussbacher, L. Liu, E. Yu, ...
 - Verbindet Zielorientierte Sprache GRL mit Use Case Maps
- **AGORA („Attributed Goal-Oriented Requirements Analysis Method“)**
 - H. Kaia, H. Horai, M. Saeki (Tokyo)
 - Metrik zur Bewertung von Zielen
 - Vergleich der Bewertungen der Stakeholder unterstützt Konfliktentdeckung
- **KAOS**
 - Axel van Lamsweerde, R. Darimont, E. Letier
 - Spezifikation von Zielen:
 - Graphisch und textuell
 - auch formal zur Deduktion von Eigenschaften
 - KAOS/UML: Integration mit UML

Im Folgenden
stellen wir
KAOS vor

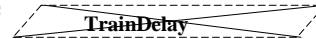
Modellierung von Zielen in KAOS: Typen von Zielen

- **Typ definiert Klasse mit vorgeschriebenem oder bevorzugtem Verhalten**

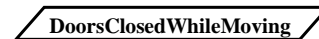
- **Achieve Goal:** Zu erreichendes Ziel
 - CurrentCondition implies **eventually** TargetCondition
Z.B. Achieve[TrainProgress]



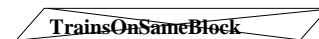
- **Cease Goal:** Unerwünschte Eigenschaft, die abgestellt werden soll



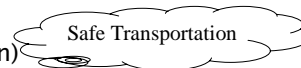
- **Maintain Goal:** Zu bewahrende Eigenschaft
 - CurrentCondition implies **always** TargetCondition **unless** NewSituation
Z.B. Maintain[DoorsClosedWhileMoving]



- **Avoid Goal:** Zu verhindernde Eigenschaft
 - CurrentCondition implies **not** TargetCondition
Z.B. Avoid[TrainsOnSameBlock]



- **Softgoal:** Weiches Ziel (hilft Präferenzen zu setzen, wenn Alternativen ausgewählt werden)
 - Z.B. Safe Transportation

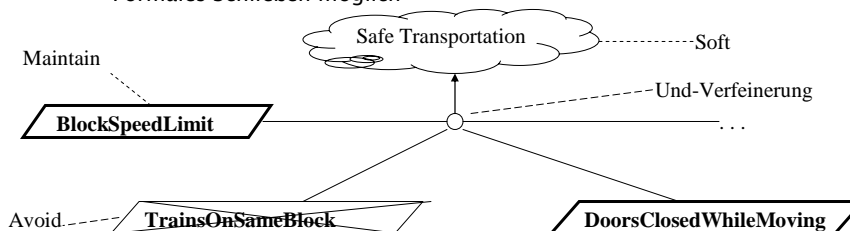


Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Modellierung von Zielen in KAOS: Typen

- **Softgoals versus Achieve/Maintain goals**

- Die Erfüllung eines Softgoals kann nicht eindeutig festgestellt werden
 - Nur qualitatives Schließen möglich
- Die Erfüllung eines Achieve/Maintain Goals kann verifiziert werden
 - Formales Schließen möglich



Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

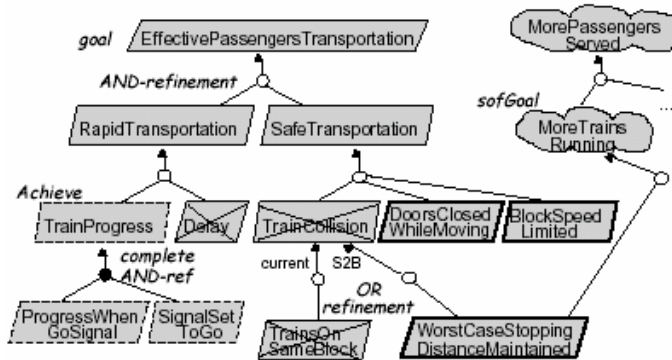
Modellierung von Zielen in KAOS: Definition von Zielen

- **Attribute beschreiben zieleigene Eigenschaften**
 - Name BlockSpeedLimit
 - Definition Ein Zug sollte nicht schneller fahren als die maximal erlaubte Geschwindigkeit des Blocks
 - [Priorität] highest, high, ..., lowest (optional)
 - [Owner] Aktor, der das Ziel verlangt hat
- **Attribute werden zur Spezifikation und zum Schließen verwendet (z.B. zum Konfliktmanagement)**

Modellierung von Zielen in KAOS: Beziehungen zwischen Zielen

- **Beziehungen zu anderen Zielen**
 - And/Or-Verfeinerung
 - Verhinderung (Obstruction)
 - Konflikt
- **Beziehungen zwischen unterschiedlichen Modellen (Verfolgbarkeit)**
 - Referenz auf Objekte
 - Verantwortung von Aktoren
 - Abdeckung von Use Cases
 - Realisierung von Zielen durch Operationen (Operationalisierung)

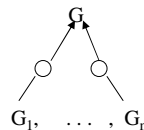
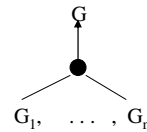
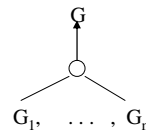
Und/Oder-Verfeinerung: Beispiel



Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

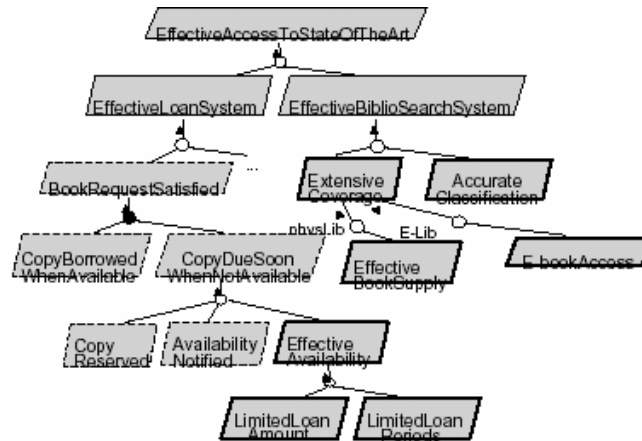
Und/Oder-Verfeinerung

- Ziel G wird **Und-verfeinert** in Unterziele G_1, \dots, G_n , falls die Erfüllung von G_1, \dots, G_n zur Erfüllung von G beiträgt.
- Die Menge $\{G_1, \dots, G_n\}$ ist eine **vollständige Und-Verfeinerung** von G , falls G_1, \dots, G_n zur Erfüllung von G genügen (einschließlich der Anwendungsbereichseigenschaften Dom), d.h. falls $\{G_1, \dots, G_n, \text{Dom}\} \models G$
- Ziel G wird **Oder-verfeinert** in Unterziele G_1, \dots, G_n , falls (für $i=1, \dots, n$) die Erfüllung von G_i zur Erfüllung von G genügt. (G_i heißt auch Alternative für G)



Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

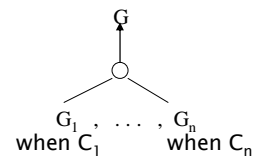
Und/Oder-Verfeinerung: Beispiel



Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Und-Verfeinerung durch Fallunterscheidung

- Ziel G wird **Und-verfeinert** durch **Fallunterscheidung** der Unterziele G_1 when C_1, \dots, G_n when C_n , falls die Erfüllung von C_1 implies G_1, \dots, C_n implies G_n zur Erfüllung von G beiträgt.

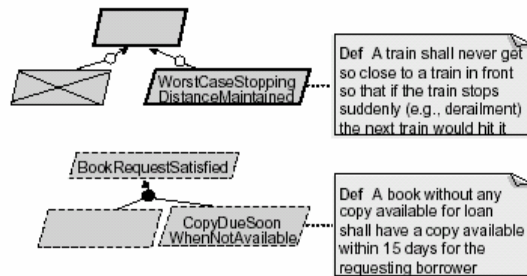


Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Modellierung von Zielen: Tips

- **Zur Vermeidung von Mehrdeutigkeiten bei der Interpretation von Zielen**

- Präzise, konsistente und vollständige Zieldef. wichtig



- Def. muss von allen Beteiligten akzeptiert werden

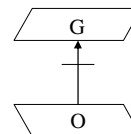
- Z.B. DoorsClosedWhileMoving
... Nur während der Fahrt? ... Zwischen den Stationen?

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Obstruktions-Analyse

- **Problem: Ziele sind oft zu ideal und werden verletzt durch unerwartetes Verhalten von Akteuren**
- **Obstruktion (Obstacle) = Bedingung an das System zur Zielverhinderung („abstrakte Ausnahme“)**

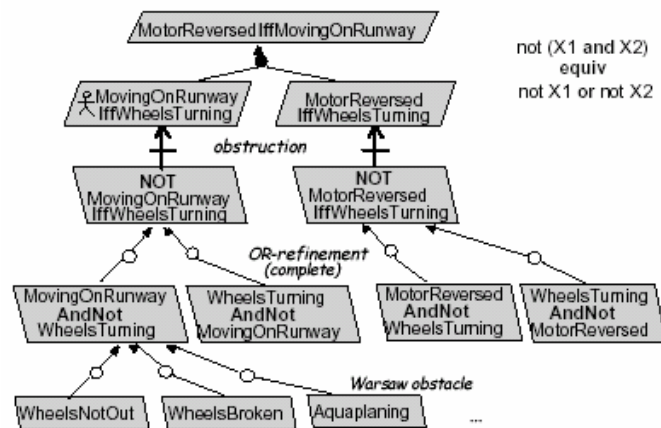
$\{O, \text{Dom}\} \models \text{not } G$ Verhinderung
 $\text{Dom} \models \text{not } O$ Konsistenz



- **Obstruktionen liefern ...**
 neue realistischere Ziele
 vollständigere Anforderungen
 robustere Systeme

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Obstruktions-Analyse: Beispiel



Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

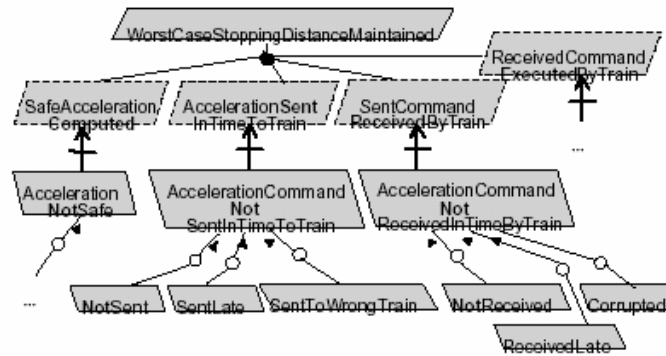
Obstruktions-Analyse: Vorgehensweise

- Für jedes Blatt im Verfeinerungsgraph
 - Identifiziere so viele Obstruktionen wie möglich
 - Behalte die machbaren und plausiblen Obstruktionen
 - Bewerte, wie kritisch sie für die Anwendung sind
- Zur Identifizierung von Obstruktionen für ein Ziel G
 - Negiere G
 - Finde so viele Und/Oder-Verfeinerungen von **not** G wie möglich in Bezug auf die Eigenschaften der Anwendung ...
 - ... bis machbare, plausible und beobachtbare Vorbedingungen für Obstruktionen gefunden sind

= „Zielbasierte Fehlerbaumkonstruktion“!

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Obstruktions-Analyse: Beispiel



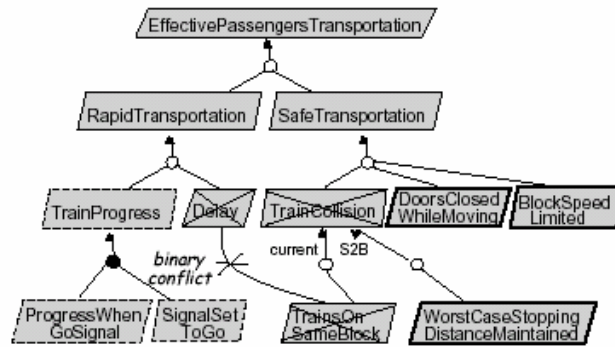
Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Konflikt-Analyse

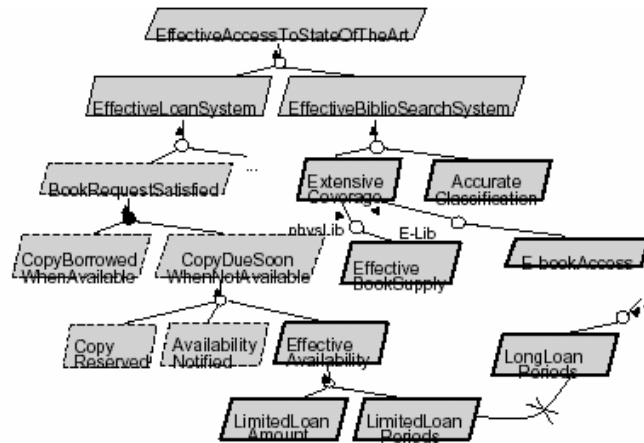
- Ziele G_1, \dots, G_n **divergieren**, falls
es eine (Abgrenz-) Bedingung B gibt mit:
 $\{B, G_1, \dots, G_n, \text{Dom}\} \models \text{false}$ **Inkonsistenz**
für alle $i \in \{1, \dots, n\}$:
 $\{B, G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_n, \text{Dom}\} \not\models \text{false}$ **Minimalität**
- **Beispiel**
 G_1 : DoorsClosedBetweenStations
 G_2 : DoorsOpenWhenAlarm
 B : AlarmRaisedBetweenStations
- **Spezialfall**
Binärer Konflikt mit $B = \text{true}$:
 $\{G_1, \text{Dom}\} \models \text{not } G_2$

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Konflikt-Analyse



Konflikt-Analyse



Übergang zu Use Cases und Szenarios

- Ziel-Operationalisierung:**

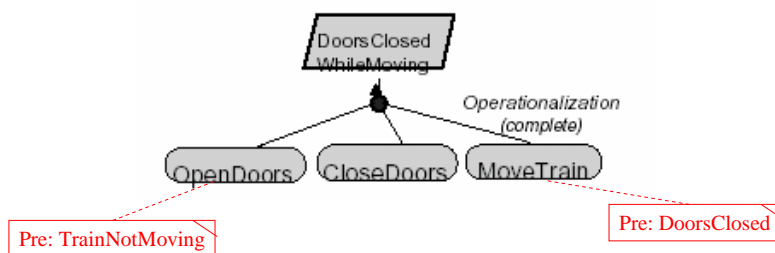
G wird korrekt durch die Operationen/Use Cases Op_1, \dots, Op_n realisiert, falls die Spezifikationen von Op_1, \dots, Op_n notwendig und hinreichend sind um G zu erreichen

$$\{Spec(Op_1), \dots, Spec(Op_n)\} \models G$$

Vollständigkeit

$$G \models Spec(Op_1) \text{ and } \dots \text{ and } Spec(Op_n)$$

Minimalität

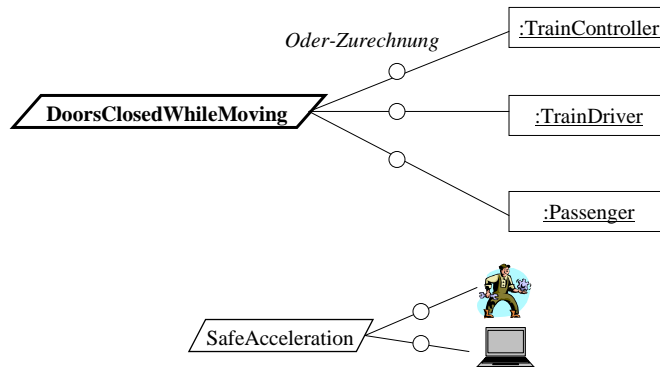


Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Verantwortlichkeiten bestimmen

- Ziel-Verantwortlichkeit:**

G ist zu Ob zurechenbar, falls G von Ob realisiert werden kann.

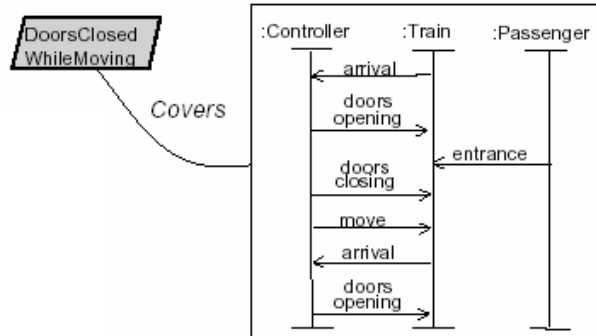


Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Übergang zu Use Cases und Szenarios

- **Überdeckung von Szenarios:**

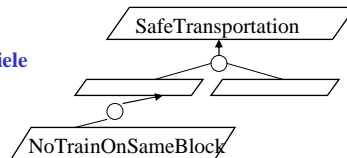
G überdeckt Sc, falls Sc ein Ablauf aus der Menge der von G definierten Verhalten ist.



Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

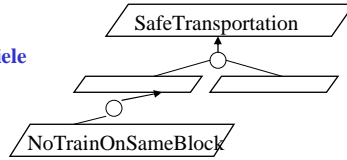
1. Domainanalyse:
Verfeinere/Abstrahiere Ziele



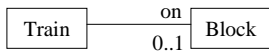
Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

1. Domainanalyse:
Verfeinere/Abstrahiere Ziele

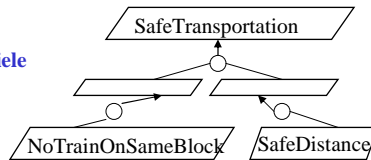


2. Domainanalyse:
Finde/Strukturiere Objekte



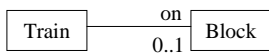
Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

1. Domainanalyse:
Verfeinere/Abstrahiere Ziele



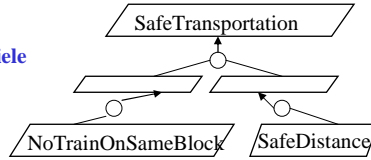
3. S2B-Analyse:
Finde Zielalternativen

2. Domainanalyse:
Finde/Strukturiere Objekte



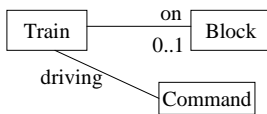
Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

1. **Domainanalyse:**
Verfeinere/Abstrahiere Ziele



3. **S2B-Analyse:**
Finde Zielalternativen

2. **Domainanalyse:**
Finde/Strukturiere Objekte

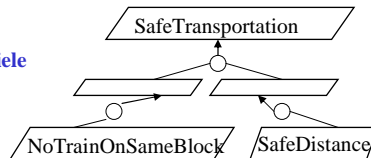


4. **S2B-Analyse:**
Mehr Objekte durch
zusätz. Ziele

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

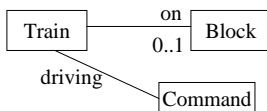
Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

1. **Domainanalyse:**
Verfeinere/Abstrahiere Ziele

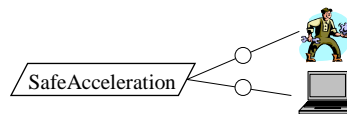


3. **S2B-Analyse:**
Finde Zielalternativen

2. **Domainanalyse:**
Finde/Strukturiere Objekte



4. **S2B-Analyse:**
Mehr Objekte durch
zusätz. Ziele

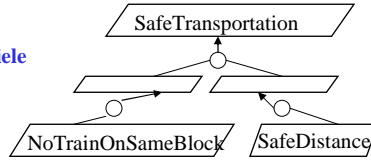


5. **Verantwortlichkeitsanalyse:**
Oder-Zurechnung von Objekten

Methoden des Software Engineering (c) 2004, Koch, Störrle, Wirsing, LMU München

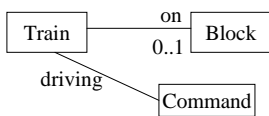
Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

1. Domainanalyse:
Verfeinere/Abstrahiere Ziele

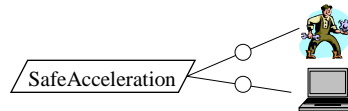


3. S2B-Analyse:
Finde Zielalternativen

2. Domainanalyse:
Finde/Strukturiere Objekte



4. S2B-Analyse:
Mehr Objekte durch
zusätz. Ziele

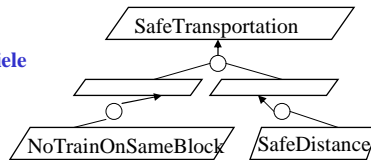


5. Verantwortlichkeitsanalyse:
Oder-Zurechnung von Objekten

1.-5. Obstruktions- &
Konfliktanalyse

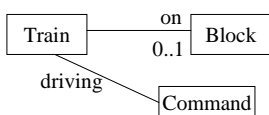
Die KAOS-Methode: Zielorientierte RE-Vorgehensweise

1. Domainanalyse:
Verfeinere/Abstrahiere Ziele

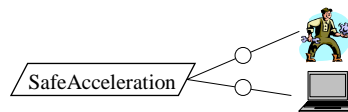


3. S2B-Analyse:
Finde Zielalternativen

2. Domainanalyse:
Finde/Strukturiere Objekte

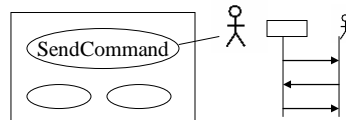


4. S2B-Analyse:
Mehr Objekte durch
zusätz. Ziele



5. Verantwortlichkeitsanalyse:
Oder-Zurechnung von Objekten

1.-5. Obstruktions- &
Konfliktanalyse



6. Übergang zu Use Cases

Zusammenfassung

- Zielorientierte Ansätze zu RE wie KAOS, URN, i*, AGORA ergänzen OOA und Use Case-orientierte Methoden
- Die KAOS-Methode unterstützt
 - Unterschiedliche Arten von Zielen wie Achieve-, Cease-, Maintain-, Avoid- und Softgoals
 - Und/Oder-Verfeinerung zur Strukturierung von Zielen
 - Obstruktionsanalyse
 - Konflikterkennung
 - Mit UML integrierte Vorgehensweise zu RE

Literatur

- A. van Lamsweerde: Goal-oriented Requirements Engineering: A guided tour. In: RE 2001 – Proc. Int. Conference on Requirements Engineering, Toronto, IEEE Computer Science, 2001, 249–263
- URN Web Site. <http://www.usecasemaps.org/urn/>
- D. Amyot, G. Mussbacher: URN: Towards a new standard for the visual description of requirements. In: SAMS'02 – 3rd SDL and MSC Workshop, Aberystwyth, 2002, Springer Lecture Notes in Computer Science 2599, 2002, 21–37.
- H. Kaiya, H. Horai, M. Saeki: AGORA: Attributed Goal-oriented Requirements Analysis Method. In: RE 2002 – Proc. 10th Internat. Conf. On Requirements Engineering, Essen, IEEE Computer Science, 2002, 13–22.