Herausforderungen in der Laufzeitdarstellung von Anforderungen für eingebettete Systeme

Fabian Kneer und Erik Kamsties
Fachhochschule Dortmund
{erik.kamsties, fabian.kneer}@fh-dortmund.de

Vorstellung

FH STRUKTUR - Process Improvement for Mechatronic and Embedded Systems - pimes (2013 – 2016)

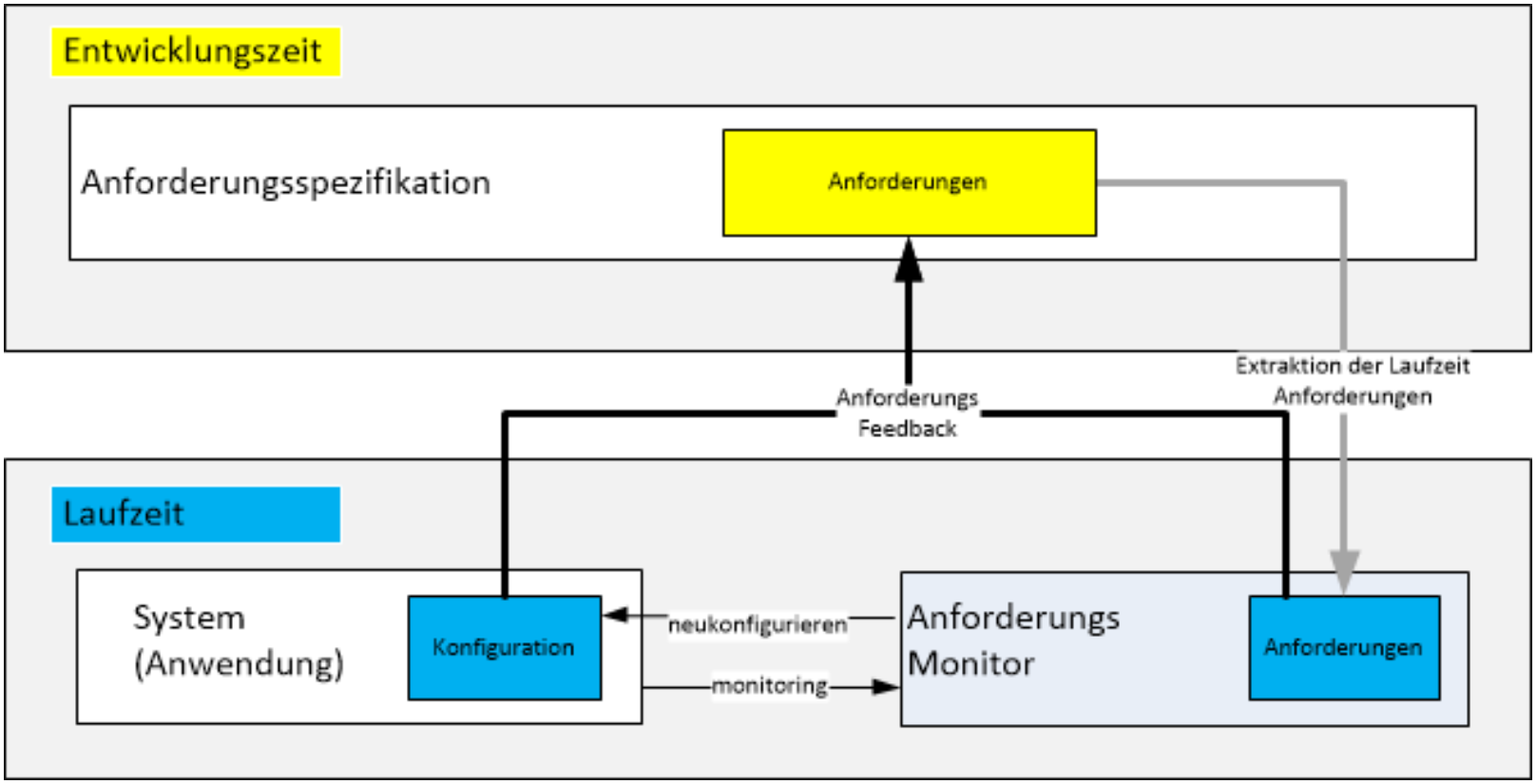
Ziel des Projektes: FH STRUKTUR ist darauf ausgerichtet, der Forschung an Fachhochschulen Impulse zur Identifizierung neuer Forschungsansätze zu geben. In diesen Forschungsschwerpunkten sollen Professorinnen und Professoren disziplinen- und fachbereichsübergreifend besonders zukunftsrelevante Fragestellungen angehen, die einen Beitrag zur Lösung der großen gesellschaftlichen Herausforderungen liefern.

Motivation

- Problem:
  - Dynamisch anpassbare Systeme (DAS)
    - Unvollständiges Wissen über die Umgebung
    - Umgebung unterliegt häufigen Änderungen
    - Nicht durch traditionelle Methoden des Requirements Engineering beherrschbar
- Lösung:
  - Laufzeit-Darstellung der Anforderungen
    - Anpassbare Anforderungen, um sich an die Umgebung zu adaptieren

Ziel

Traceability: Realisierung der Verbindung zwischen der statischen Anforderungsspezifikation und den dynamischen Laufzeitanforderungen.



Agenda

Motivation

Verwandte Arbeiten

- Vergleich
- Abgrenzung

Vorstellung der einzelnen Komponenten des Anforderungsmonitors

- Laufzeit Model
- Monitor
- Auswirkungsanalyse
- Zusammenspiel der Komponenten, anhand eines Beispiels

Ausblick

Vergleich

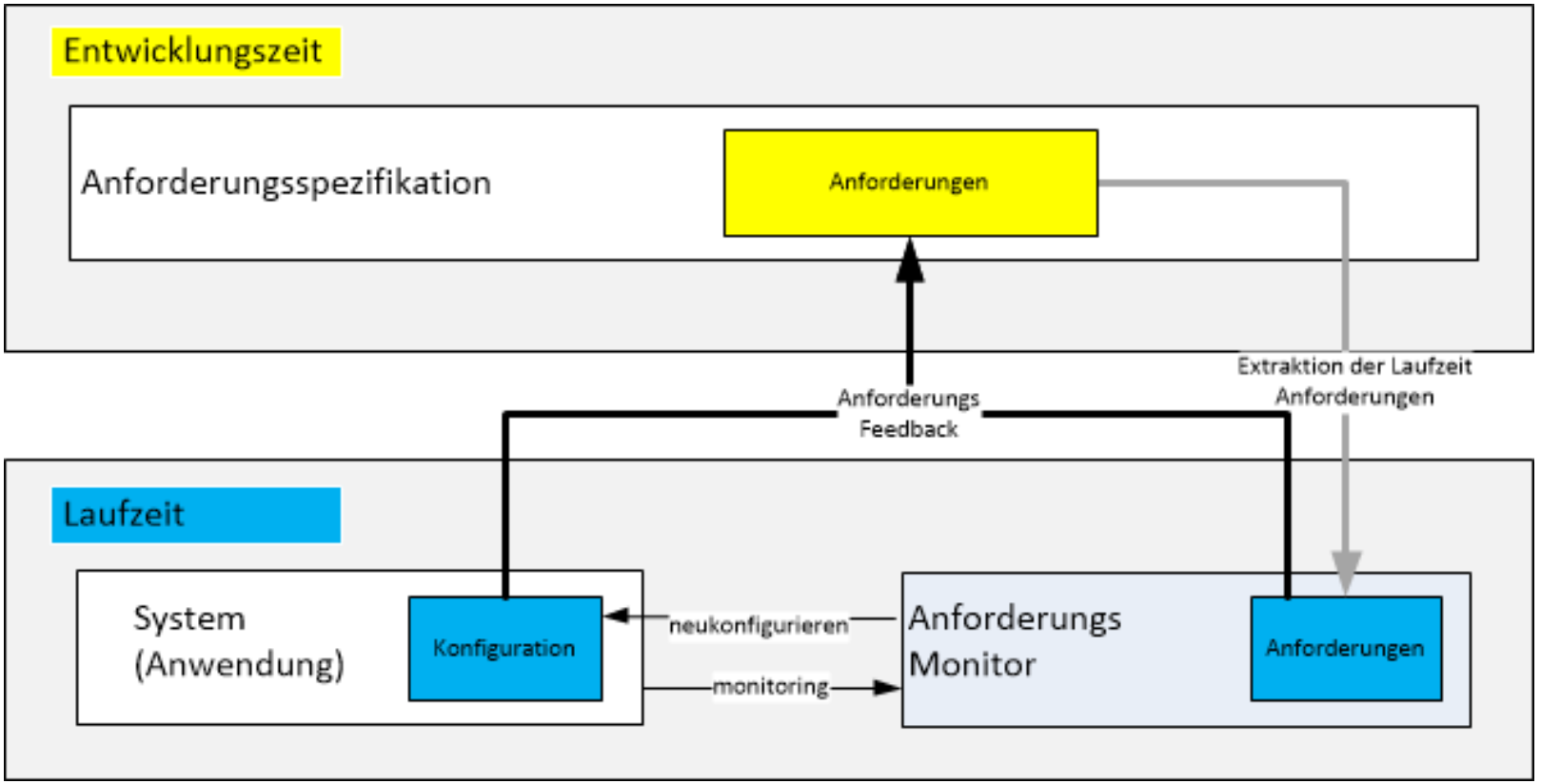
Framework/ techniques	Usage of a Goal Model	Adaption support	Different Monitoring Configurations	Goal reasoning (button-up)	General/ Service Based
Fickas et al. [1,2]	Yes: KAOS	No	No	Partially (Based on activities)	General
Robinson[3]	Yes: KAOS	No	Partially (Implemented by the designer)	-	Service
Wang et al. [4]	Yes	No	Partially (Testing not supported)	-	Service
Baresi et al. [5]	Yes: KAOS	Yes	No	-	Service
Oriol et al. [6]	Yes: ARML	Yes	Yes	Furture work	Service
Bencomo et al.[7]	Partially (A DDNs is generated out of a i*-model)	Yes	-	-	General

Abgrenzung – eingebettete Systeme

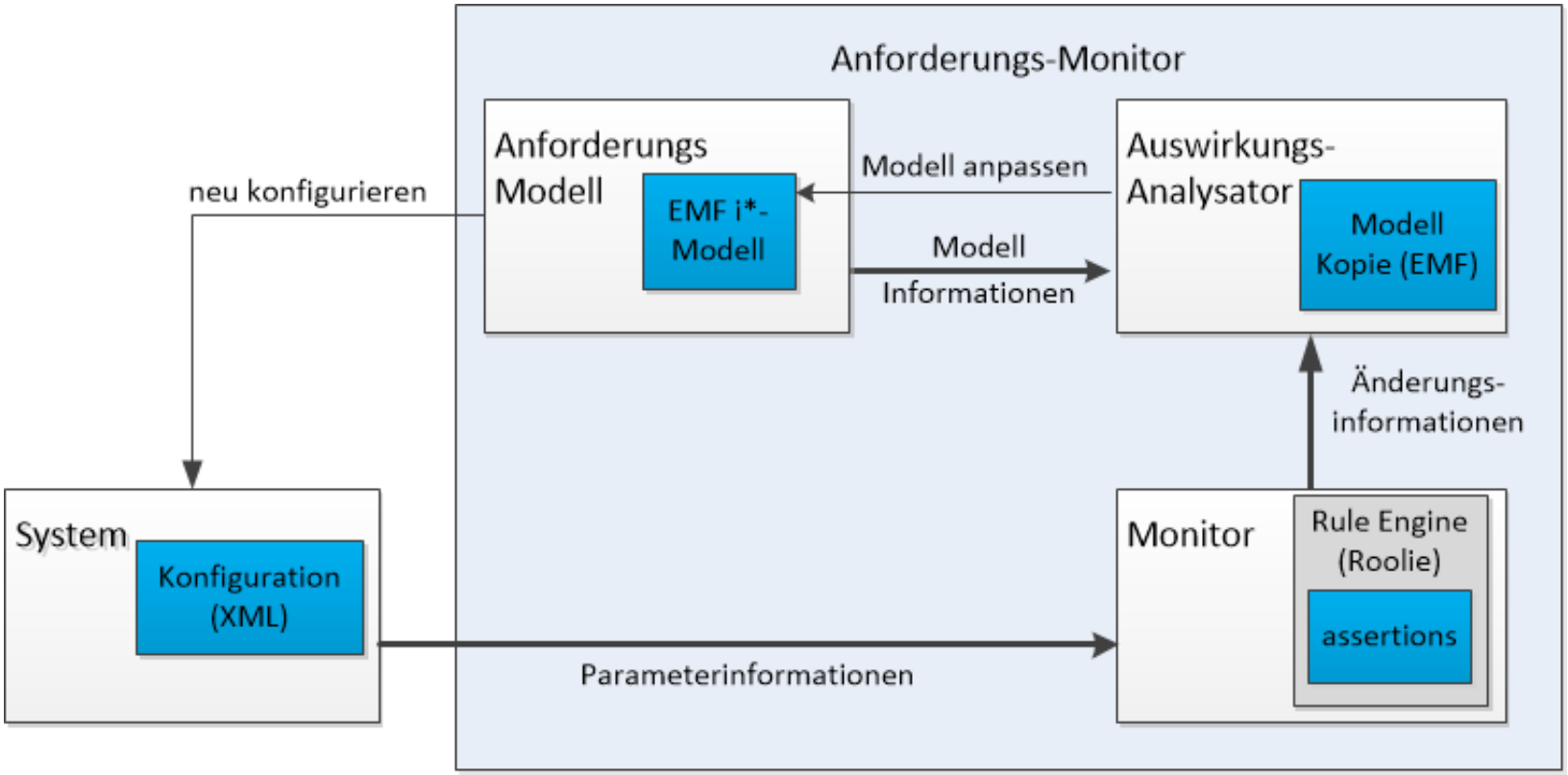
- Massenprodukte
  - Ressourcenarm (Speicher, Rechenleistung)
  - Vorhersagbares zeitliches Verhalten
  - Anzeigemöglichkeiten
  - Konnektivität (Bluetooth, Wifi, etc.)
  - Verteilte Systeme mit hoher Dynamik

- In Bezug auf die Implementierung
  - Leicht gewichtige Implementierung
  - Anpassbare Komponenten (Frameworks, Modelle, etc.)
  - Keine Service orientierte Architektur
  - Automatische Auswertung
  - Automatische Adaptionen
  - Maschinen nahe Programmierung

Anforderungs-Monitor

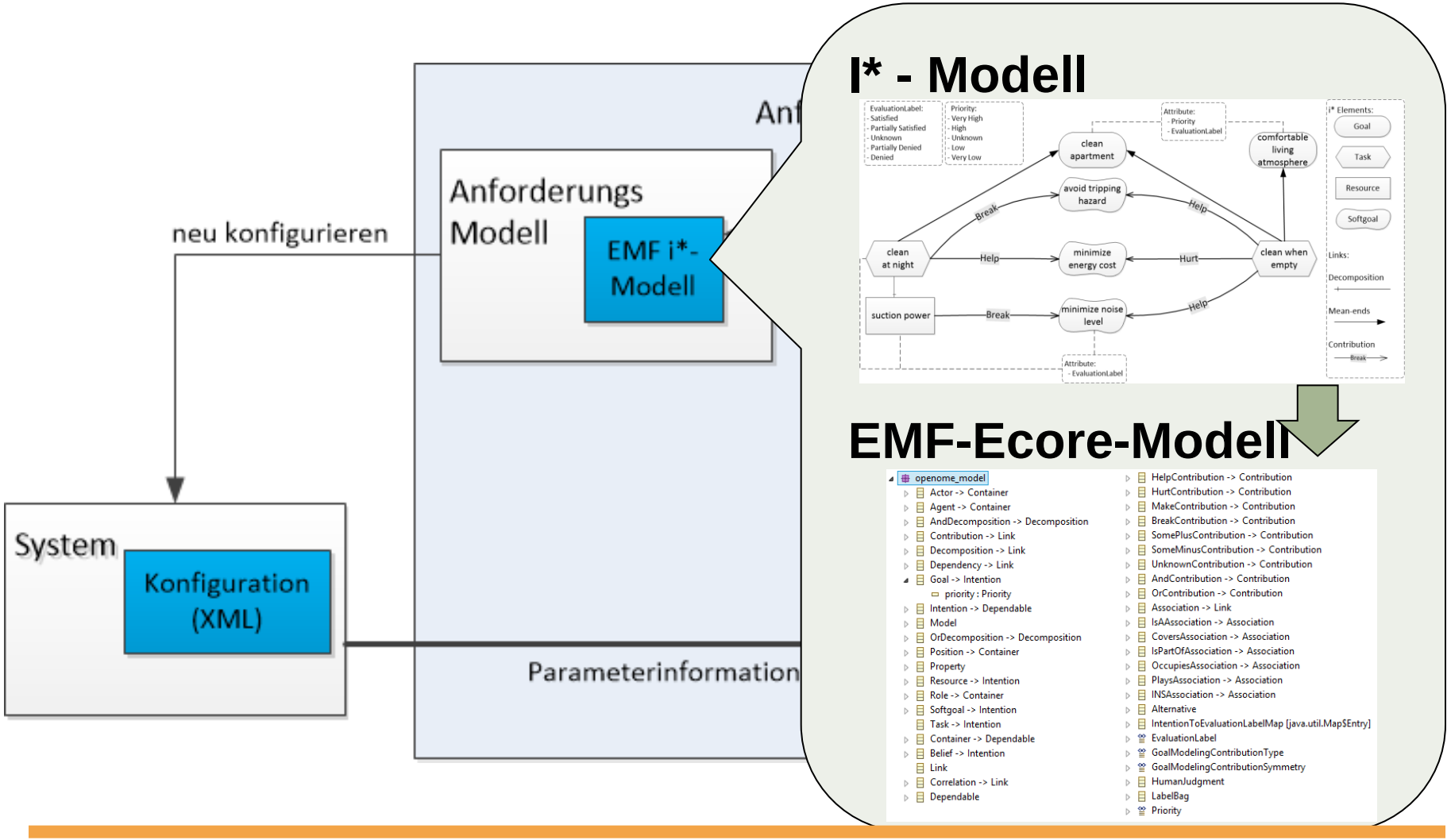


Anforderungs-Monitor

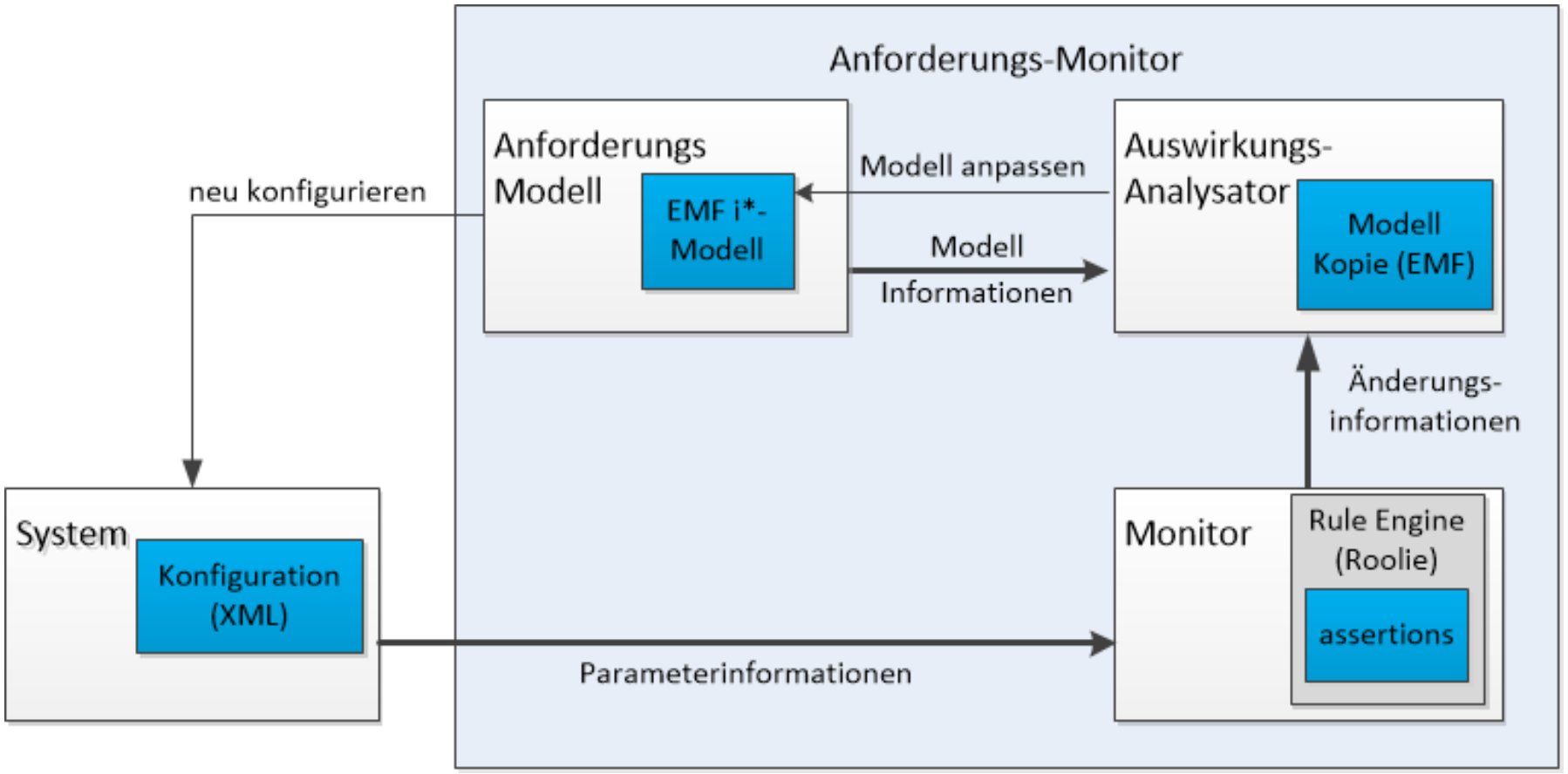


EMF: http://www.eclipse.org/modeling/emf/
Roolie: http://roolie.sourceforge.net/)
iStar-Wiki: http://istar.rwth-aachen.de

Komponentenbeschreibung



Komponentenbeschreibung



Komponentenbeschreibung

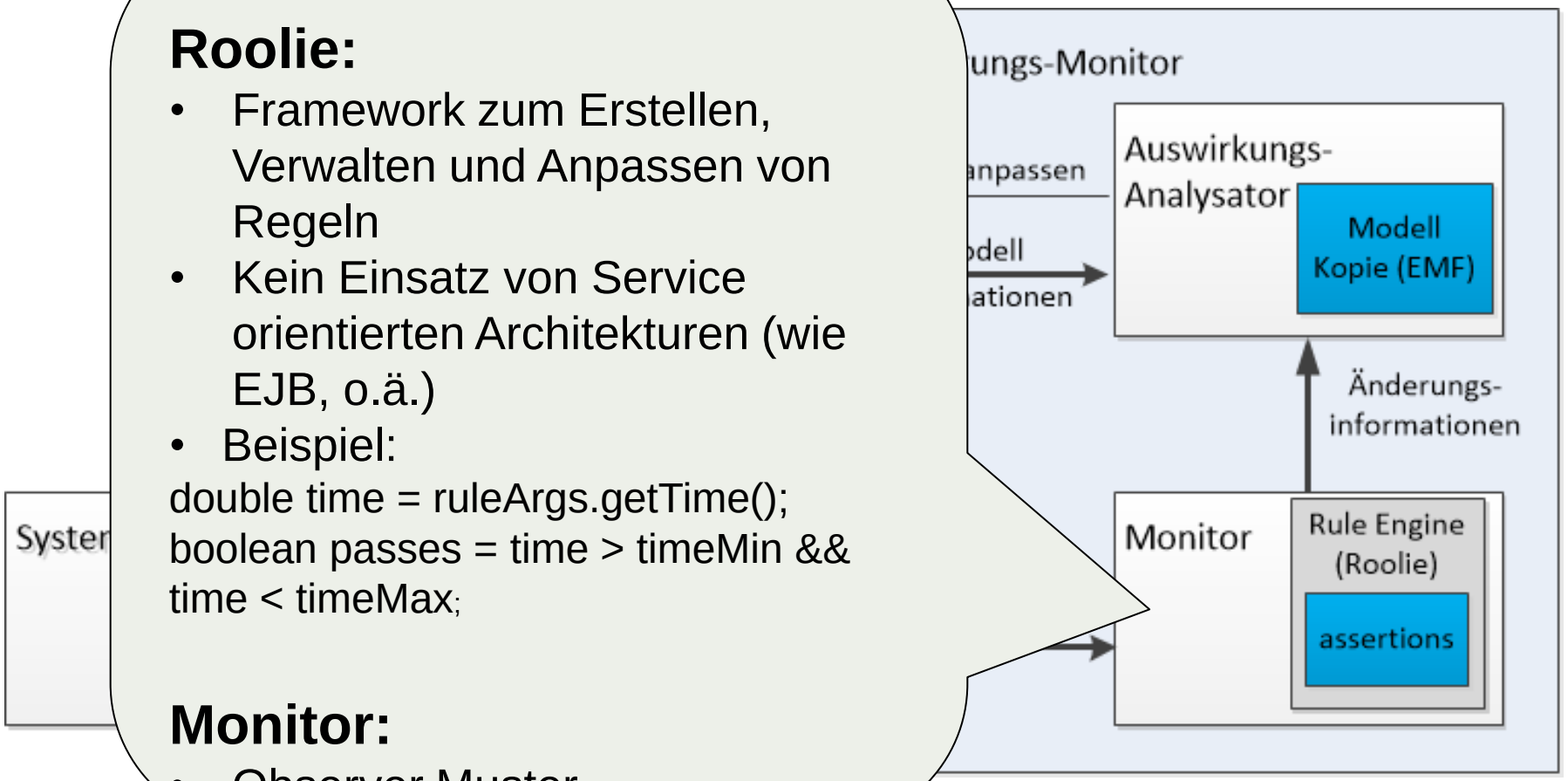
Roolie:

- Framework zum Erstellen, Verwalten und Anpassen von Regeln
- Kein Einsatz von Service orientierten Architekturen (wie EJB, o.ä.)
- Beispiel:

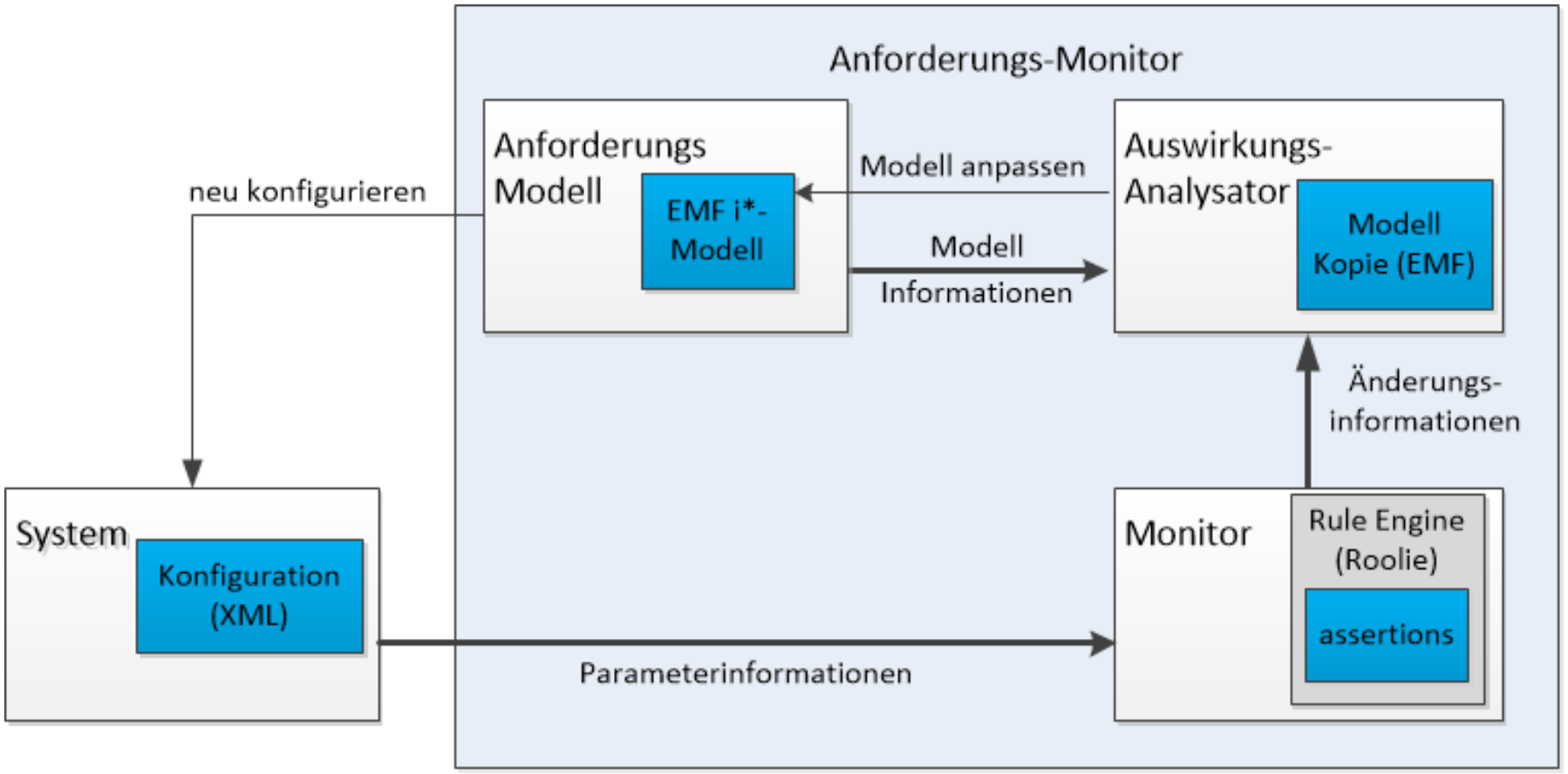
```
double time = ruleArgs.getTime();
boolean passes = time > timeMin &&
time < timeMax;
```

Monitor:

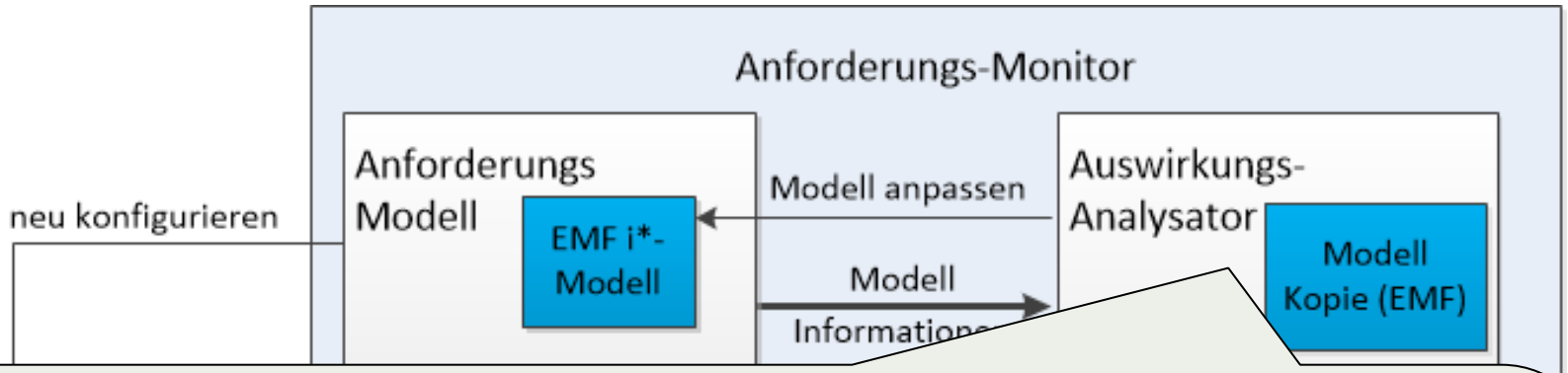
- Observer Muster



Komponentenbeschreibung



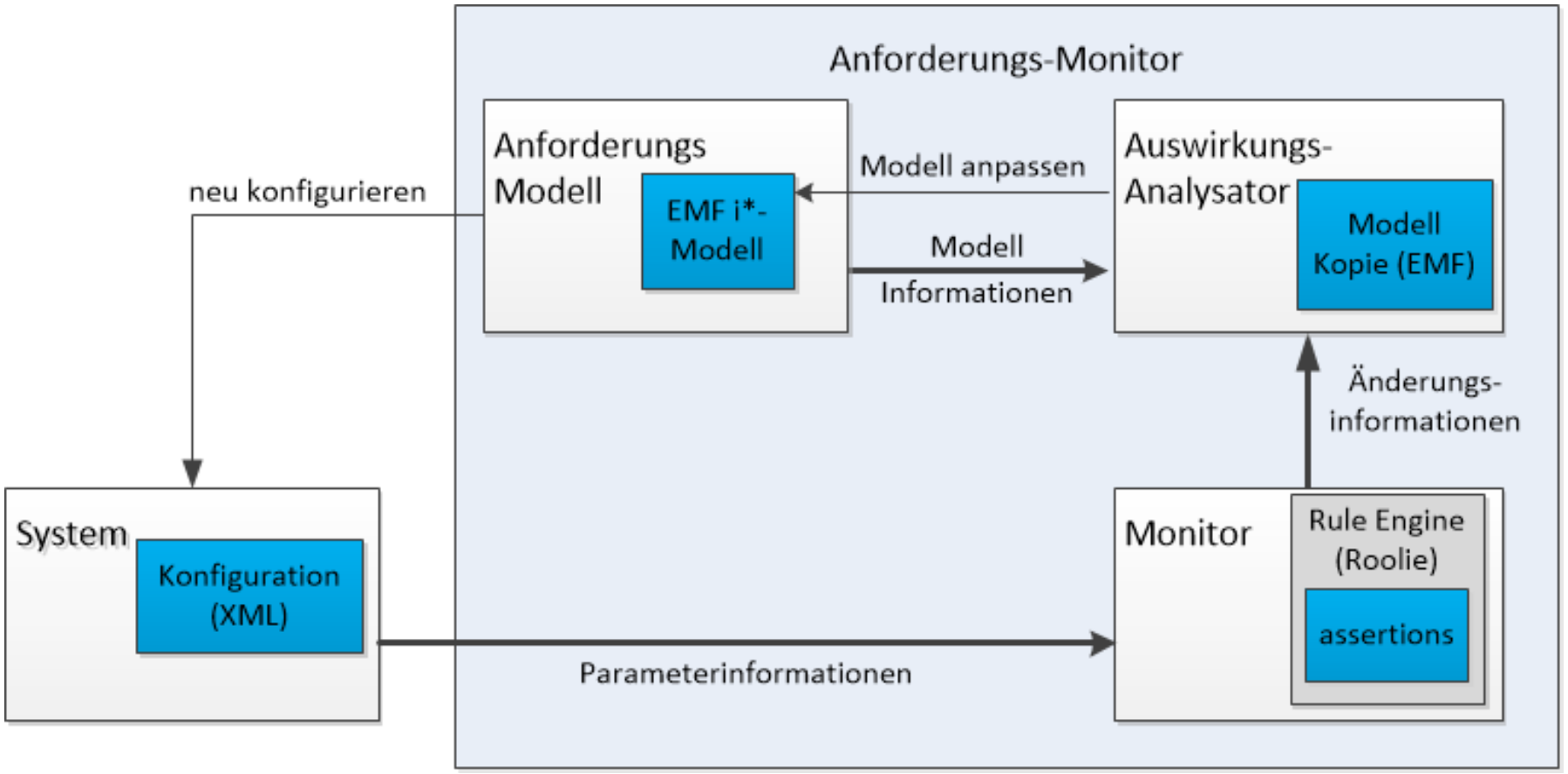
Komponentenbeschreibung



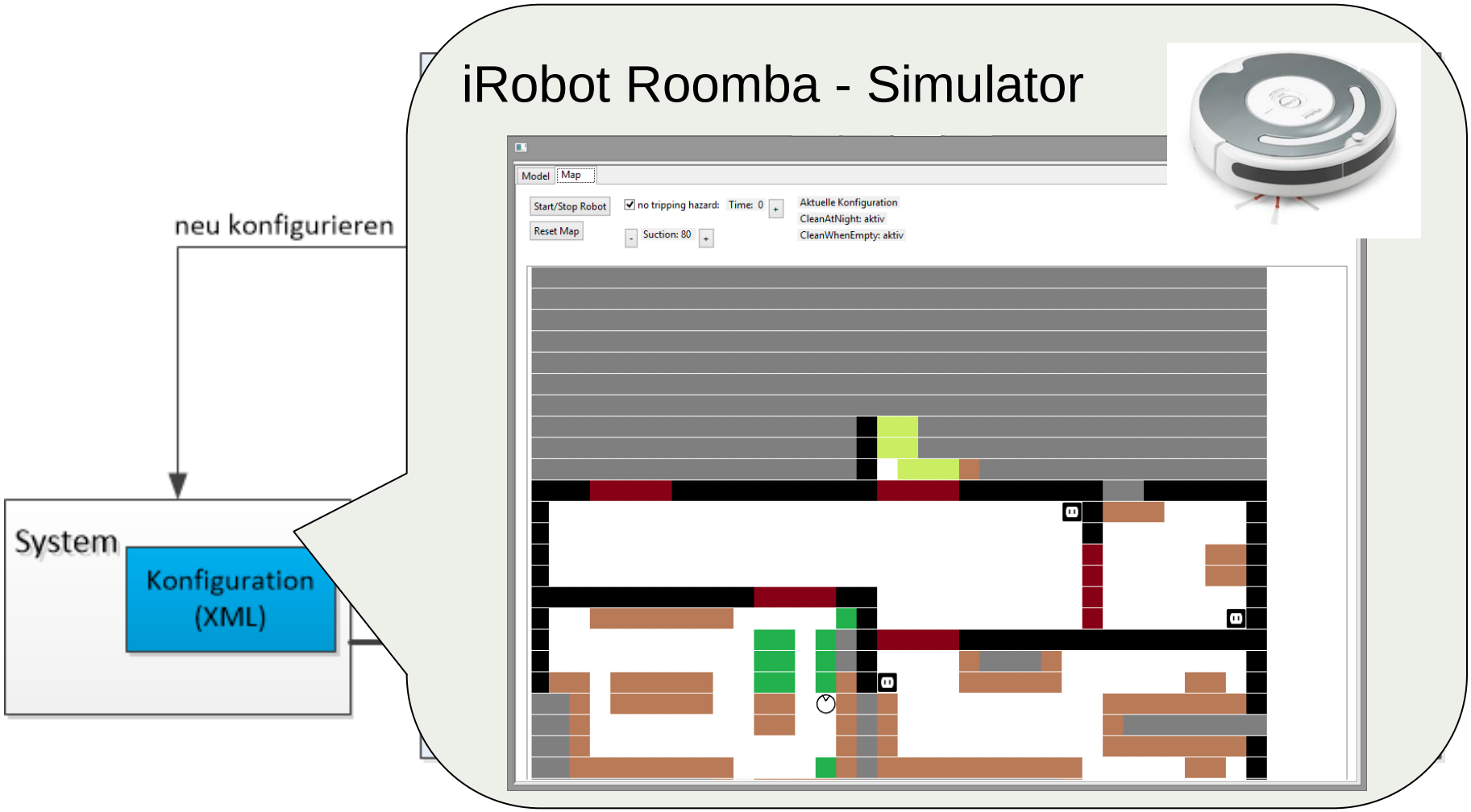
Auswirkungsanalysator:

- **i* Evaluierungsverfahren**:
  - Setzen von Initialwerten (Softgoals)
  - Schrittweises Durchlaufen der Elemente
    - Bestimmung der Zufriedenheit (Guidelines)
- **Bestimmung Notwendigkeit der Anpassung**
  - Vergleich der Anzahl an zufriedenen Zielen vor und nach Anpassung.
- **Anpassung des Modells oder der Assertions**

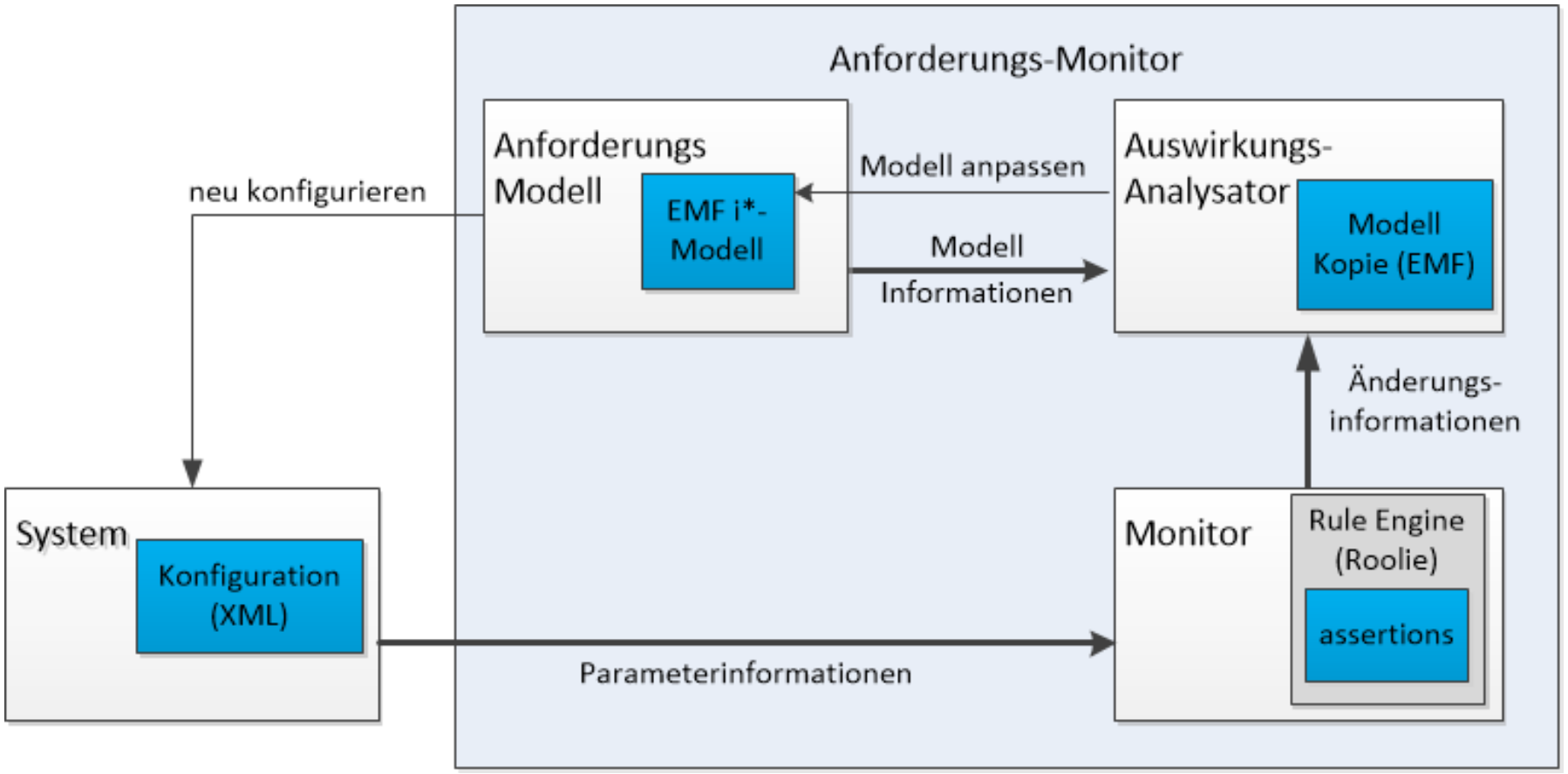
Komponentenbeschreibung



Beispiel


iRobot Roomba - Simulator
neu konfigurieren
System
Konfiguration
(XML)
Model
Map
Start/Stop Robot
Reset Map
no tripping hazard:
Time: 0
Suction: 80
Aktuelle Konfiguration
CleanAtNight: aktiv
CleanWhenEmpty: aktiv

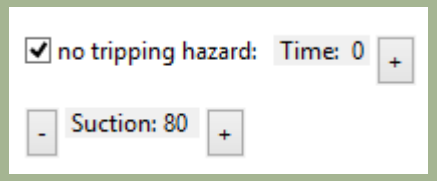
Beispiel


Anforderungs-Monitor
Anforderungs Modell
EMF i*-Modell
Modell anpassen
Modell Informationen
Auswirkungs-Analysator
Modell Kopie (EMF)
neu konfigurieren
Änderungs-informationen
System
Konfiguration (XML)
Parameterinformationen
Monitor
Rule Engine (Roolie)
assertions

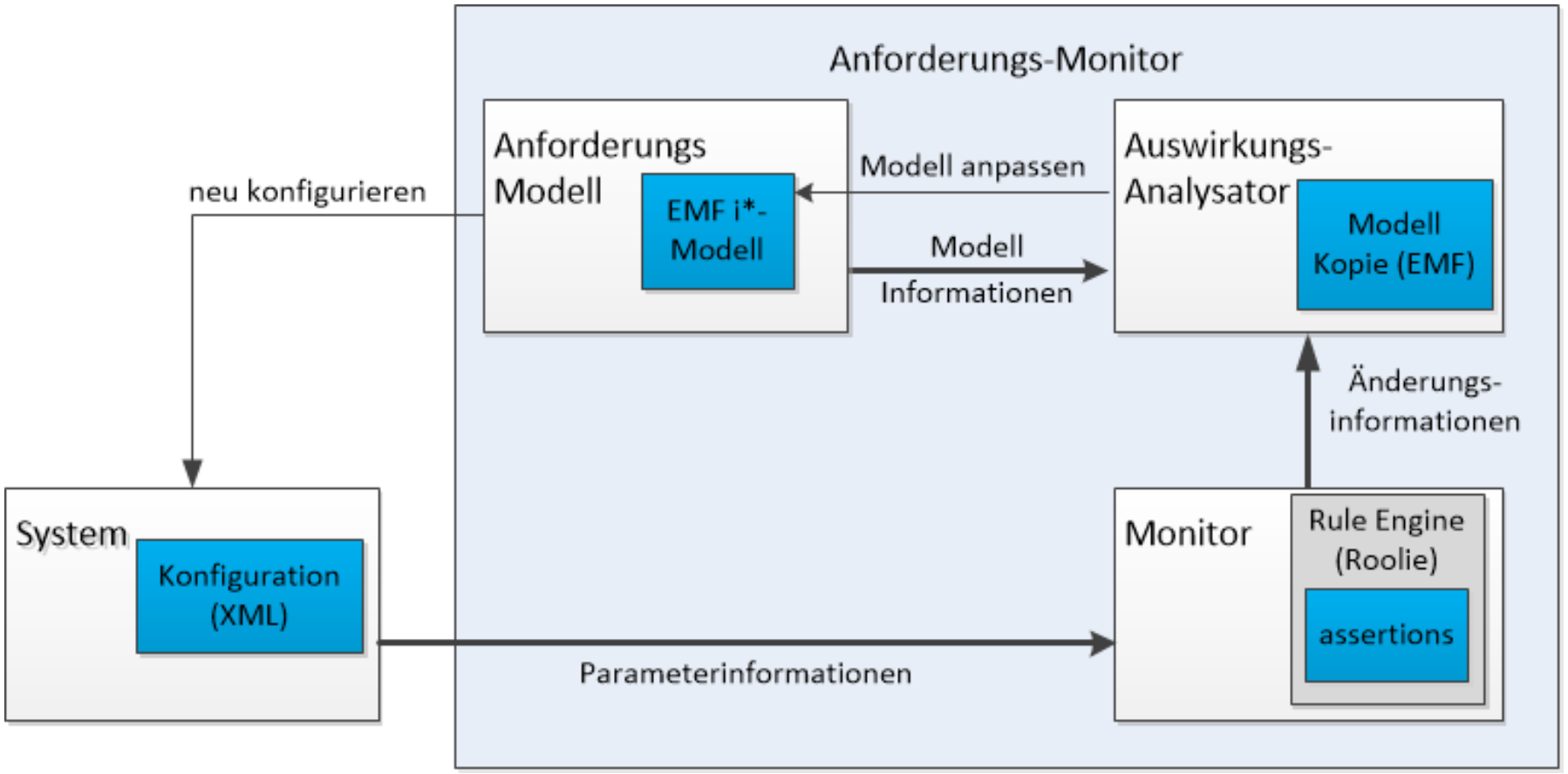
Beispiel

Beobachtung:
- Zeit
- Geräuschpegel (Saugkraft)
- Gefahr durch Stolperfallen

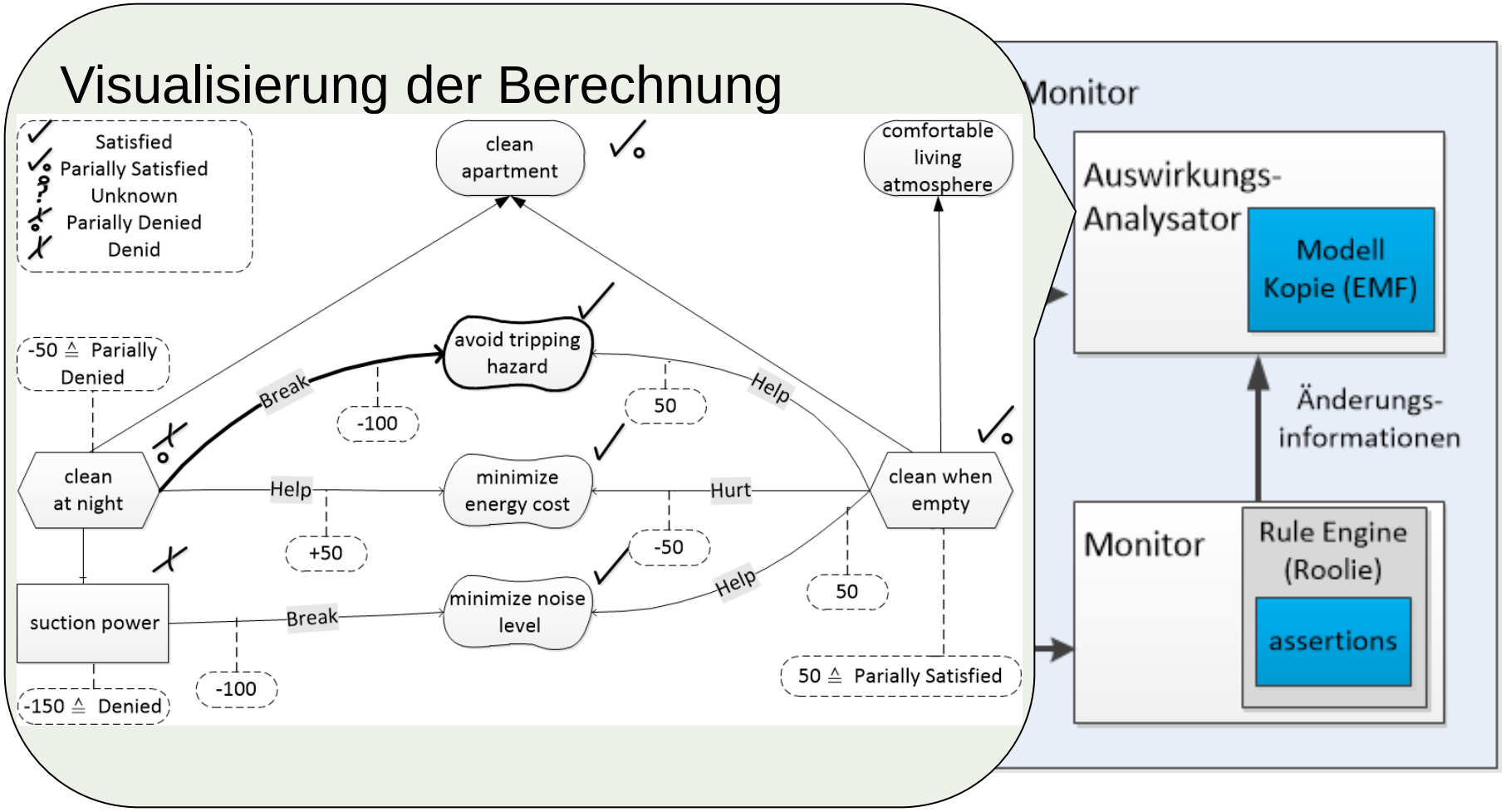
Assertions:
1. Niedrige Stromkosten zwischen 22 und 8 Uhr
2. Geräuschpegel zu Hoch, bei Saugkraft über 50%
3. Keine Stolperfallen



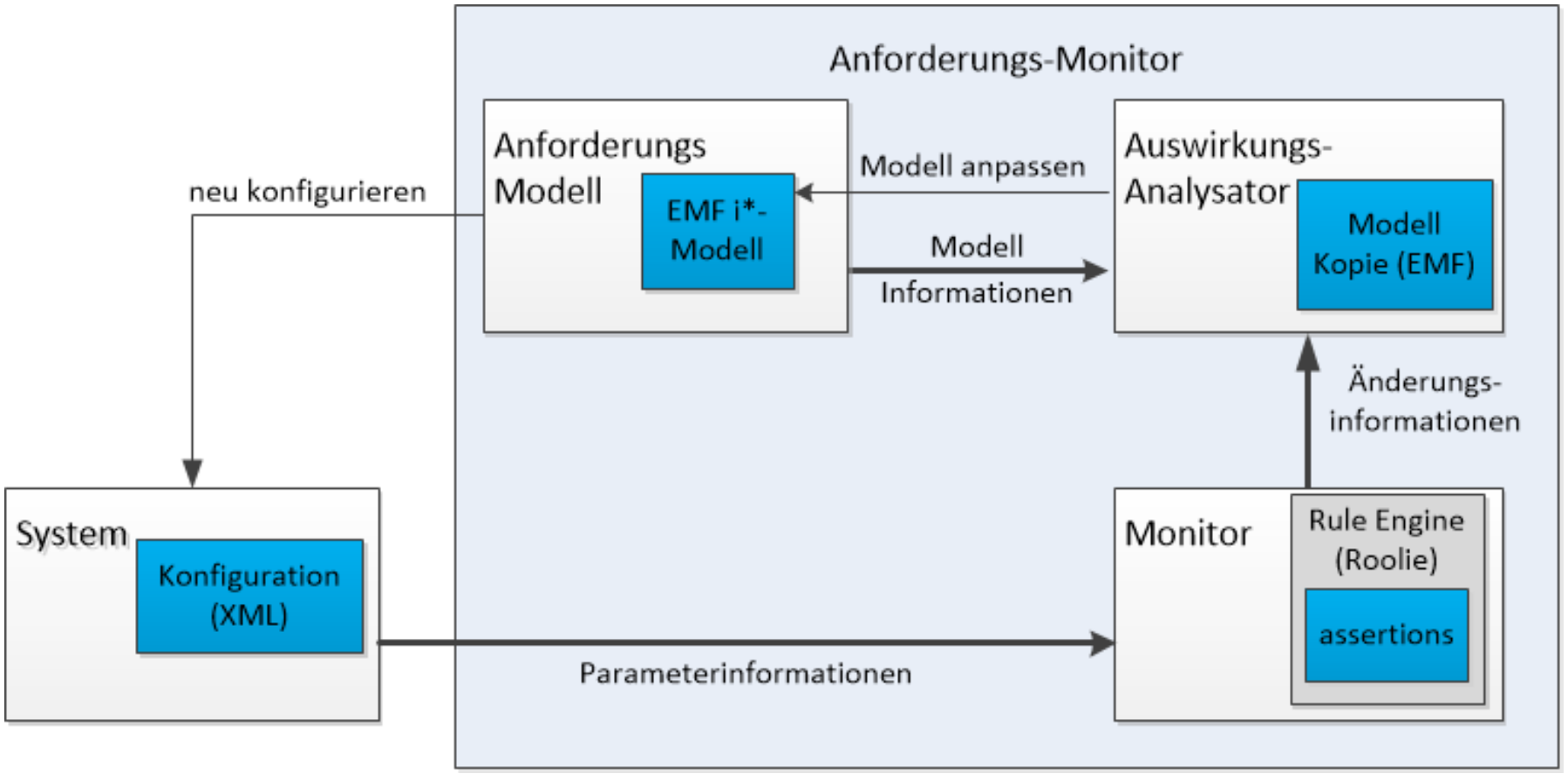
Beispiel


Anforderungs-Monitor
Anforderungs Modell
EMF i*-Modell
Modell anpassen
Modell Informationen
Auswirkungs-Analysator
Modell Kopie (EMF)
neu konfigurieren
Änderungs-informationen
System
Konfiguration (XML)
Parameterinformationen
Monitor
Rule Engine (Roolie)
assertions

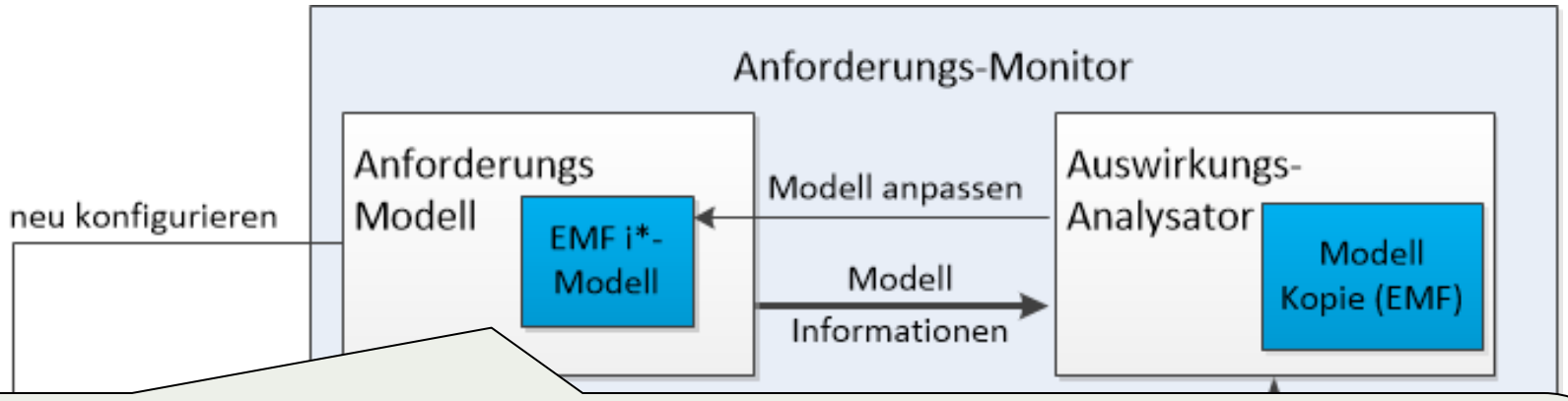
Beispiel


Visualisierung der Berechnung
Satisfied
Parially Satisfied
Unknown
Parially Denied
Denid
clean apartment
comfortable living atmosphere
avoid tripping hazard
-50 ≙ Parially Denied
Break
-100
50
Help
clean at night
Help
minimize energy cost
Hurt
clean when empty
+50
-50
suction power
Break
minimize noise level
Help
50
-100
50 ≙ Parially Satisfied
-150 ≙ Denied
Monitor
Auswirkungs-Analysator
Modell Kopie (EMF)
Änderungs-informationen
Monitor
Rule Engine (Roolie)
assertions

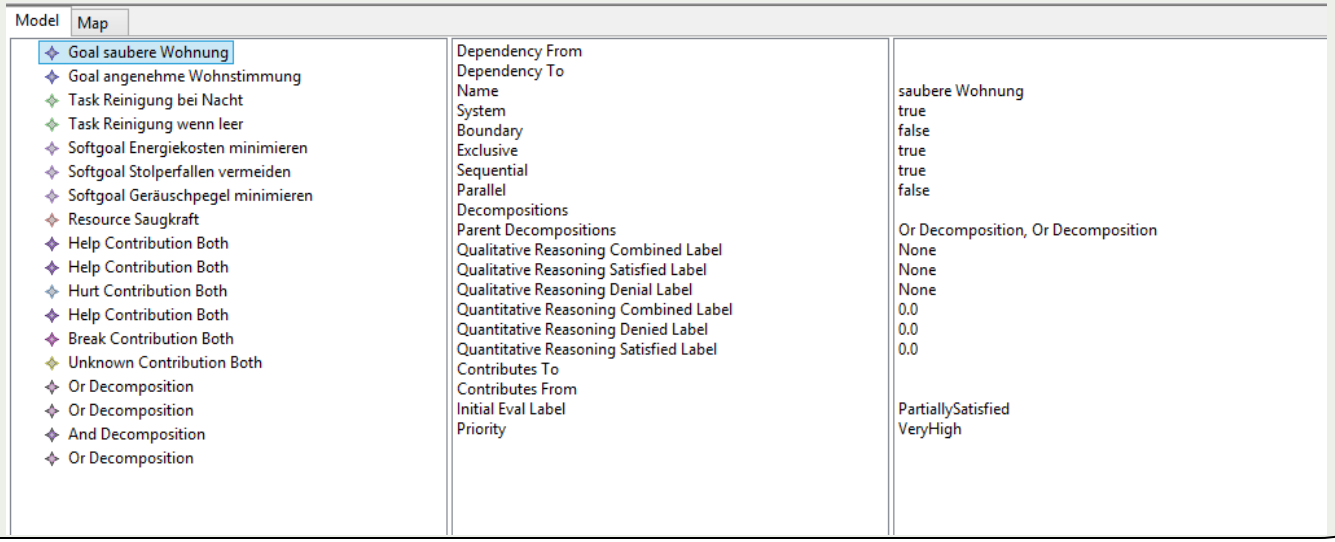
Beispiel


Anforderungs-Monitor
Anforderungs Modell
EMF i*-Modell
Modell anpassen
Modell Informationen
Auswirkungs-Analysator
Modell Kopie (EMF)
neu konfigurieren
Änderungs-informationen
System
Konfiguration (XML)
Parameterinformationen
Monitor
Rule Engine (Roolie)
assertions

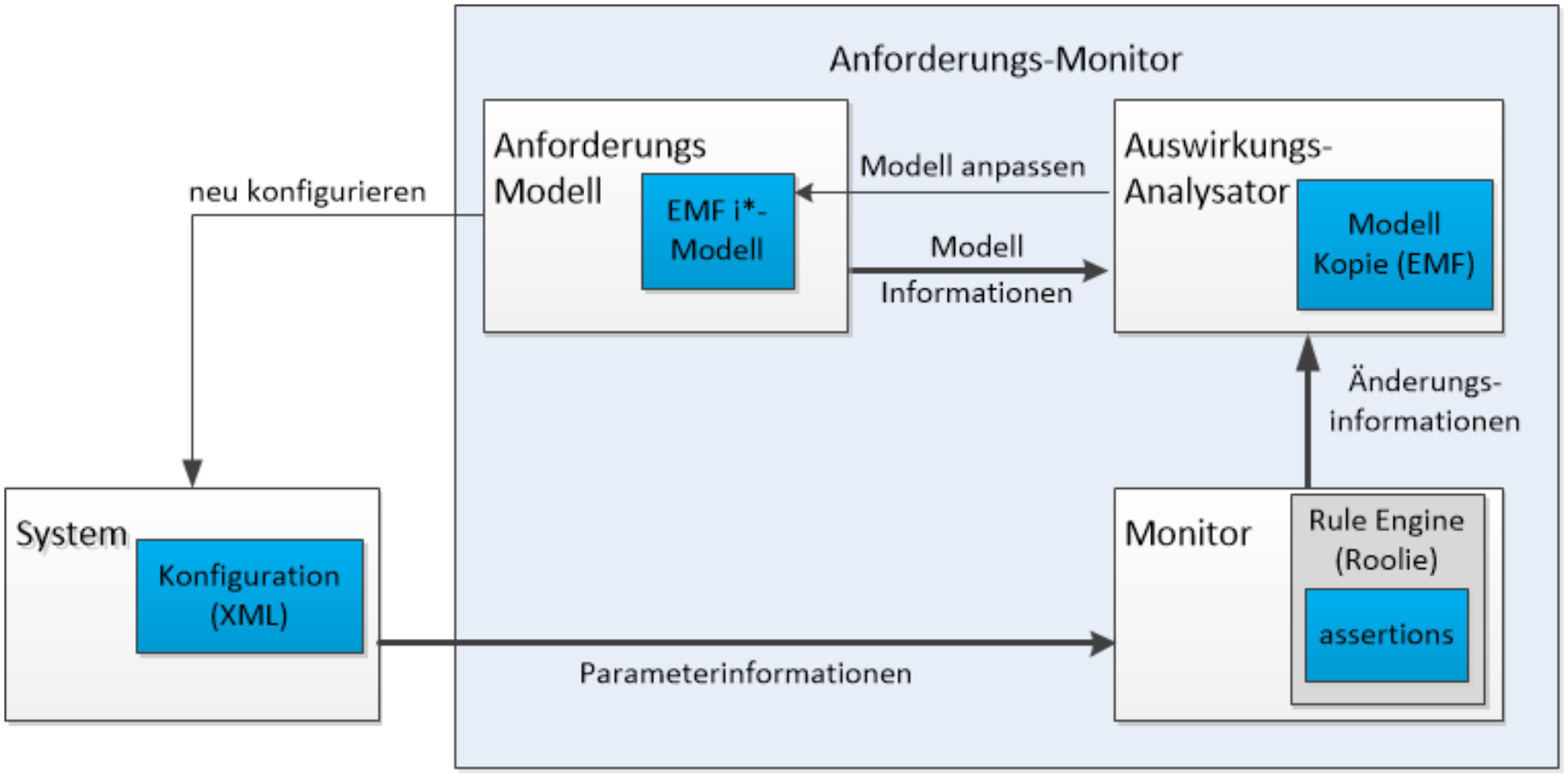
Beispiel



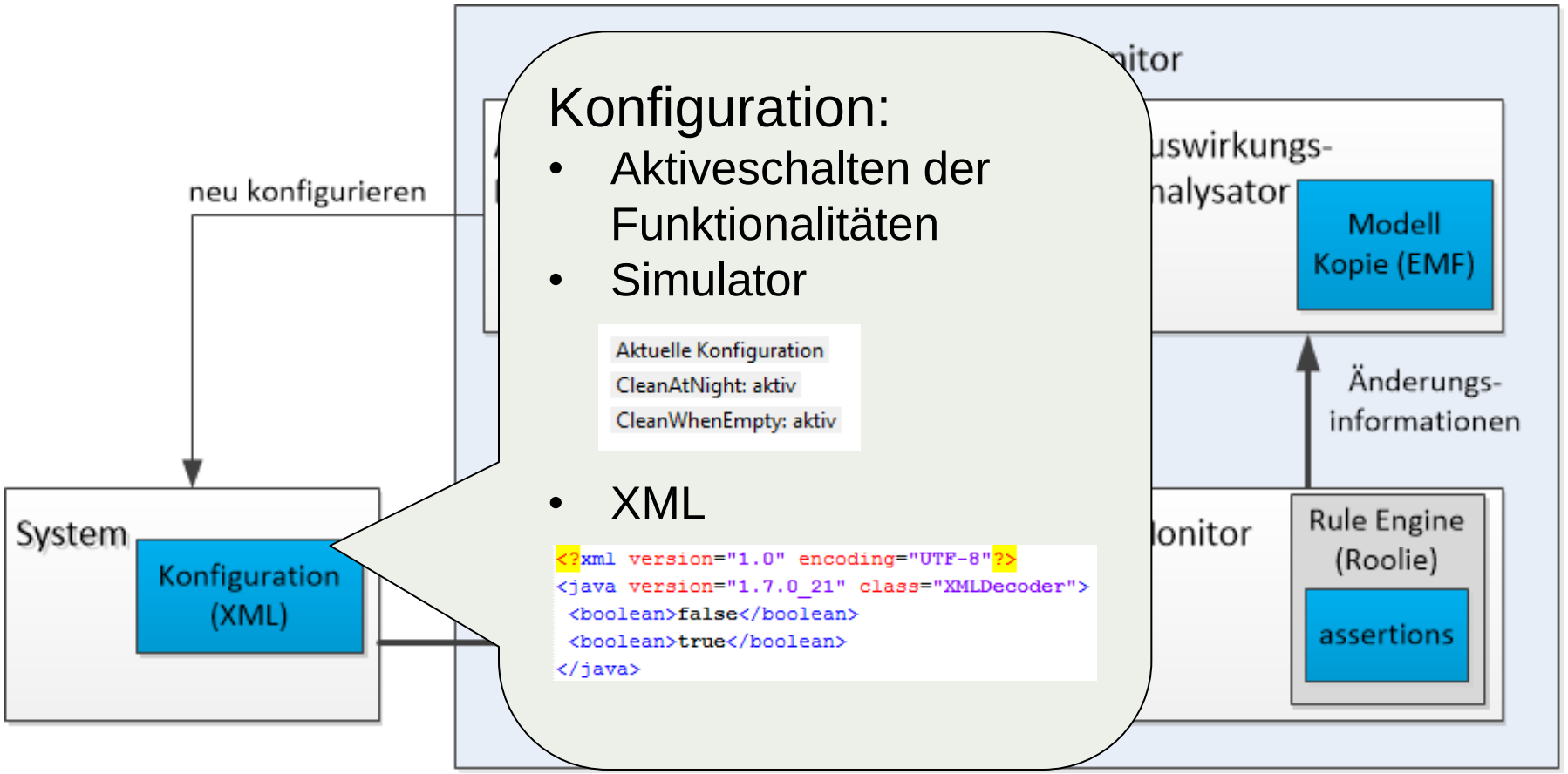
EMF Modell im Simulator



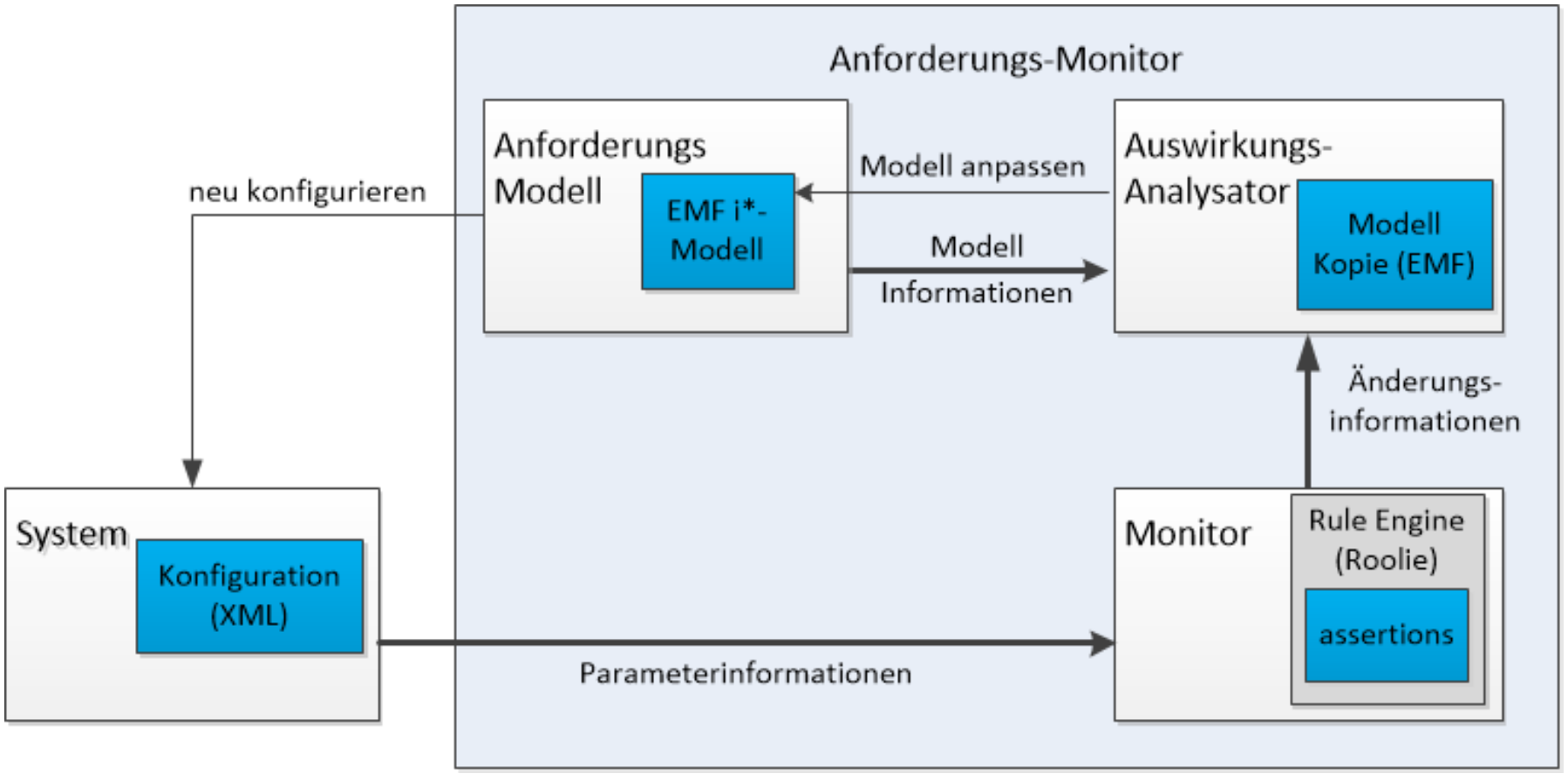
Beispiel


Anforderungs-Monitor
Anforderungs Modell
EMF i*-Modell
Modell anpassen
Modell Informationen
Auswirkungs-Analysator
Modell Kopie (EMF)
neu konfigurieren
Änderungs-informationen
System
Konfiguration (XML)
Parameterinformationen
Monitor
Rule Engine (Roolie)
assertions

Beispiel


neu konfigurieren
System
Konfiguration (XML)
Modell Kopie (EMF)
Änderungs-informationen
Rule Engine (Roolie)
assertions
Konfiguration:
• Aktiveschalten der Funktionalitäten
• Simulator
Aktuelle Konfiguration
CleanAtNight: aktiv
CleanWhenEmpty: aktiv
• XML
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.7.0_21" class="XMLDecoder">
<boolean>false</boolean>
<boolean>true</boolean>
</java>

Beispiel


Anforderungs-Monitor
Anforderungs Modell
EMF i*-Modell
Modell anpassen
Modell Informationen
Auswirkungs-Analysator
Modell Kopie (EMF)
neu konfigurieren
Änderungs-informationen
System
Konfiguration (XML)
Parameterinformationen
Monitor
Rule Engine (Roolie)
assertions

Zusammenfassung

Erster Schritt um die Brücke zwischen Entwicklungszeit- und Laufzeit-Anforderungen zu schlagen.

- Implementierung der Laufzeitumgebung für eingebettete Systeme
  - Laufzeitdarstellung der Anforderungen
  - Vereinfachte und automatische Auswirkungsanalyse
  - Automatische Adaption
- Einschränkungen durch die prototypische Implementierung in Java.
  - Verwendung von einfachen und portierbaren Komponenten, um die Portierung nach C zu ermöglichen.

Ausblick

- Aktuelle Arbeiten:
  - Verbindung von Entwicklungs- und Laufzeit-Anforderungen mit Hilfe von mbeddr
    - Nutzt gezeigten Ansatz für ein in C implementiertes System
    - Hinterlegung des i*-Models und der Anforderungen in mbeddr
- Zukünftige Arbeiten:
  - Implementierung der gezeigten Lösung in C
    - Gängige Programmiersprache im eingebetteten Bereich
  - Testen der Lösung in einer größeren Fallstudie
    - Zur Zeit theoretisches Problem „Staubsauger Roboter“
    - Testen des Zeitverhaltens
  - Erstellung eines Tools/ Framework zur Unterstützung bei der Entwicklung von DAS

Vielen Dank für Ihre Aufmerksamkeit.

Gibt es noch Fragen?

Quellen

1. Fickas, S., Feather, M.S.: Requirements monitoring in dynamic environments. In: RE 1995: Proceedings of the Second IEEE Intl. Symp. on Req. Eng., p. 140. IEEE CS (1995)
2. Feather, M.S., Fickas, S., Lamsweerde, A.V., Ponsard, C.: Reconciling system requirements and runtime behavior. In: IWSSD 1998: Proceedings of the 9th Intl. Workshop on Software Specification and Design, p. 50. IEEE CS (1998)
3. Robinson, W.N.: A requirements monitoring framework for enterprise systems. Requirements Engineering Journal 11(1), 17–41 (2006)
4. Wang, Y., McIlraith, S.A., Yu, Y., Mylopoulos, J.: Monitoring and diagnosing software requirements. Autom. Softw. Eng. 16(1), 3–35 (2009)
5. Baresi, L., Pasquale, L.: Live goals for adaptive service compositions. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2010 (2010)
6. Marc Oriol, Nauman A. Qureshi, Xavier Franch, Anna Perini, and Jordi Marco. Requirements monitoring for adaptive service-based applications. In Bjrn Regnell and Daniela Damian, editors, Requirements Engineering: Foundation for Software Quality, volume 7195 of Lecture Notes in Computer Science, pages 280–287. Springer Berlin Heidelberg, 2012
7. NaumanA. Qureshi, IvanJ. Jureta, and Anna Perini. Towards a requirements modeling language for self-adaptive systems. In Bjrn Regnell and Daniela Damian, editors, Requirements Engineering: Foundation for Software Quality, volume 7195 of Lecture Notes in Computer Science, pages 263–279. Springer Berlin Heidelberg, 2012
8. Nelly Bencomo and Amel Belaggoun. Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks. In Joerg Doerr and AndreasL. Opdahl, editors, Requirements Engineering: Foundation for Software Quality, volume 7830 of Lecture Notes in Computer Science, pages 221–236. Springer Berlin Heidelberg, 2013.