Künstliche Intelligenz

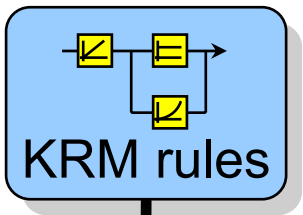
Sebastian Iwanowski
FH Wedel

Kap. 5:
Technische Diagnose

5.3: MDS: SW-Architektur und erweiterte Funktionalitäten

Wertpropagierung und ATMS

Wissensbasis
(Komponentenmodellierung
plus Systemzusammenhang)



KRM:
Knowledge **R**epresentation **M**anager

RP System

RP Inference (Wertpropagierung)
- Regeln nach Wichtigkeit ordnen und ausführen
  (wenn Prämissen im Fokus stehen)
- ATMS-Netzwerk ausbauen

adjust
focus

Candidate
Generator

activate task
(if in focus)

adjust
focus

add nodes and justifications,
report pending tasks

ATMS

report
conflicts

Wertpropagierung und ATMS

Was versteht man unter Propagierung im MDS-Kontext ?

- Propagierung ist die Weiterleitung von Informationen über ein Netzwerk aus Kanten und Knoten.

Trennung von Wertpropagierung und ATMS:

- Das **ATMS** ist verantwortlich für die Propagierung der Environments in einem gegebenen Netzwerk von Wertabhängigkeiten.
- Das Netzwerk von Wertabhängigkeiten wird in einem Rule Propagator (**RP**) hergestellt, der die Justifications aus den Regeln für die Verhaltensmodi der Komponenten zusammensetzt.
- Der RP ist genauso faul wie sein ATMS:
  - Werte werden nur weiterpropagiert, wenn sie von Environments unterstützt werden, die im gegenwärtigen Fokus stehen.
  - Das ATMS teilt dem RP unaufgefordert mit, welche Werte neu von Fokusenvironments unterstützt werden und initiiert die **(Propagation) Tasks**

Wertpropagierung und ATMS

Was bringt die Trennung von Wertpropagierung und ATMS ?

1. Antwort: Bessere Softwarearchitektur durch Modularisierung

- **Werte** entstehen meistens aus Beobachtungen (Messungen) und gezielten Eingaben. Diese sind spärlich, daher gibt es **nicht viele** resultierende Werte.

- **Environments** entstehen aus Annahmen über Verhaltensmodi. Von diesen gibt es **sehr viele** (selbst bei Einfachfehlern mindestens so viele wie Komponenten).

- Daher werden Fokusenvironments häufiger revidiert, als neue Werte berechnet werden. Diese Revision kann dann als ein ATMS-internes Problem behandelt werden.

Anm.: Die Aufteilung in ein RP- und ATMS-Modul fördert enge Modulbindung und lose Modulkopplung

Wertpropagierung und ATMS

Was bringt die Trennung von Wertpropagierung und ATMS ?

2. Antwort: Einsatz des ATMS für erweiterte Aufgaben

- Es können auch andere Annahmen als Verhaltensmodi für Komponenten untersucht werden:

  Beispiele:

  - Annahmen über Werteingaben (control inputs)

    (für die Berechnung sinnvoller Testsituationen)

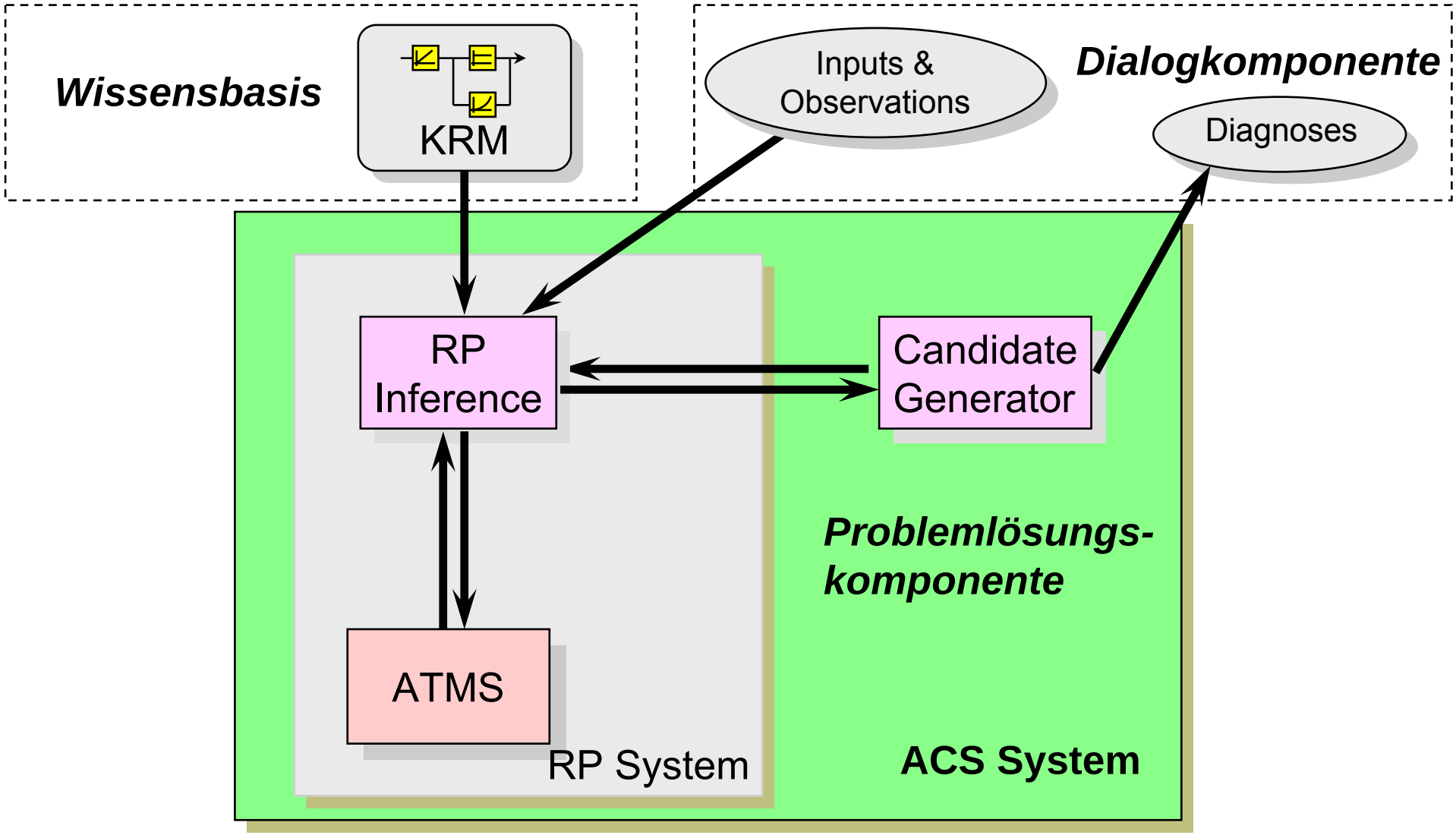
  - Annahmen über Komponentenzustände (bei dynamischen Komponenten)

    (für dynamische Komponenten, deren Zustand unbekannt ist)

  - Annahmen über beliebige andere Werte

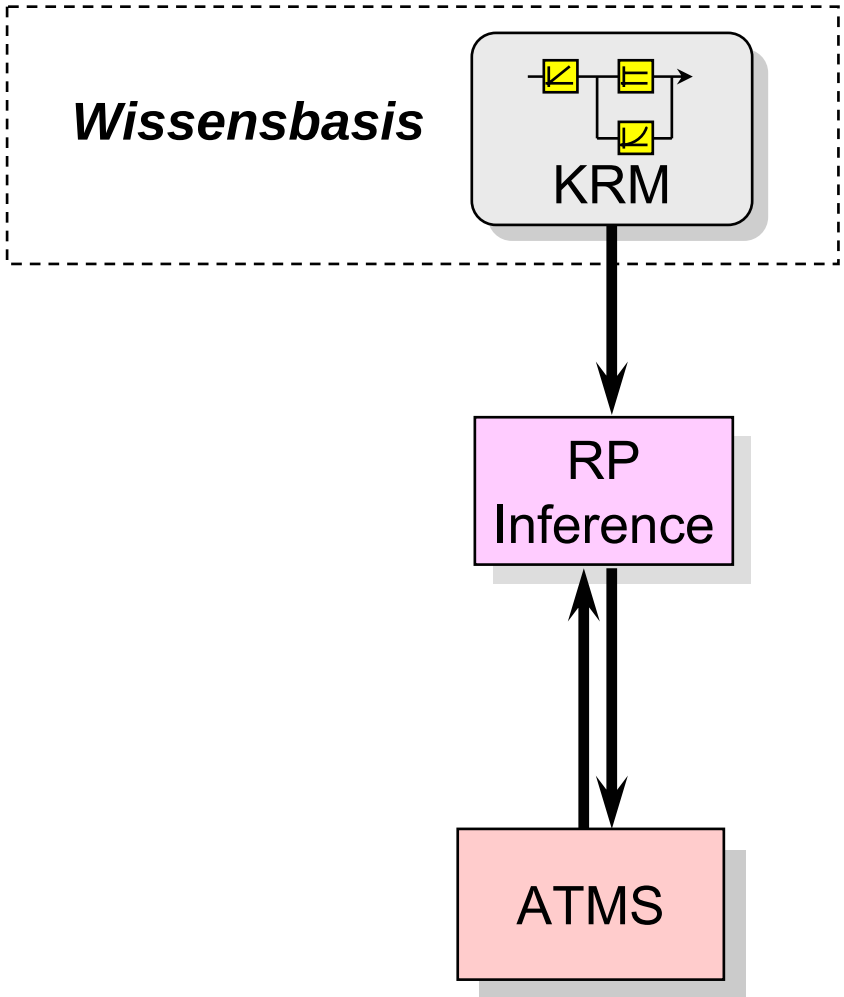
    (könnte für Beobachtungspunkte interessant sein)

Zusammenspiel Kandidatengenerierer, RP und ATMS



ACS: **A**ssumption-based **C**onstraint **S**olver

Anforderung an die Wissensbasis



Was muss die Wissensbasis an die Inferenzkomponente liefern ?

- Regeln für die Wertzusammenhänge in den einzelnen Verhaltensmodi (*Komponentenmodellierung*)
- Kenntnis über die Wertdomänen: Wann gelten zwei Werte als widersprüchlich ?

MDS löst diese Anforderungen durch das Anbieten einer Constraint-Sprache für die Komponentenmodellierung

Vom ACS zur kompletten Diagnosesoftware

Was kann das ACS für den Anwender leisten?

Eingabe:

- Einstellung bestimmter Werte im System
- Beobachtung davon abhängiger Werte im System

Ausgabe:

- Mehrere Diagnosen folgender Art:
  - Jede Diagnose weist jeder Komponente einen Verhaltensmodus zu: entweder ok oder ein definierter Fehlermodus
  - Die Regeln aller zugewiesenen Verhaltensmodi sind konsistent (mit allen eingestellten und beobachteten Werten)

Was braucht der Anwender ?

Eingabe: s.o.

Ausgabe:

- Eine eindeutige Anweisung, welche Komponenten wie repariert werden sollen

Vom ACS zur kompletten Diagnosesoftware

Was fehlt also noch ?

1) Vorschlag von Testeinstellungen (control inputs)

- Einstellung bestimmter Werte an bestimmten Stellen im System

  (derart, dass die zu erwartenden Beobachtungen die bisher gültigen Diagnosen bestmöglich unterscheiden)

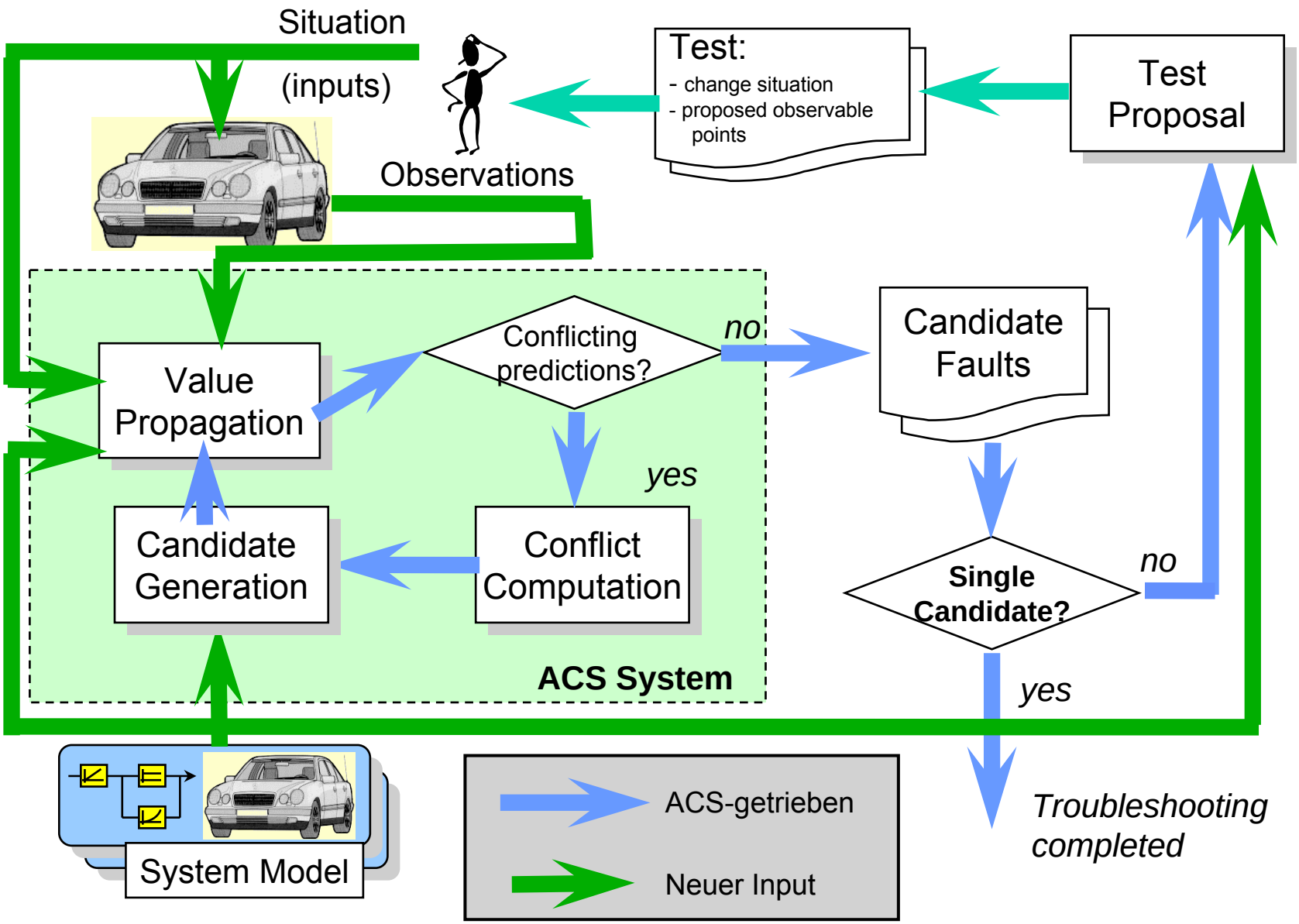
2) Vorschlag von Beobachtungspunkten

- Auswahl von Messstellen im System

  (derart, dass die zu erwartenden Beobachtungen die bisher gültigen Diagnosen bestmöglich unterscheiden)

} **Test**

Vom ACS zur kompletten Diagnosesoftware



Vom ACS zur kompletten Diagnosesoftware

Details:

1) Vorschlag von Testeinstellungen (control inputs)

- Die Control Inputs werden in der Wissensbasis gekennzeichnet und mit einer Bewertung versehen, welche die Schwierigkeit angibt, einen Wert einzugeben (Definition so genannter **Maßnahmen**).

- Die Eingabewerte an diesen Control Inputs werden als Annahmen im **selben** ATMS wie die Verhaltensmodi der Komponenten propagiert.
  Die Menge aller Eingabewerte an Control Inputs heißt **Situation**.

- Das ATMS kann zwischen den Situationsannahmen und Verhaltensannahmen unterscheiden und fokussiert sowohl auf bestimmte Verhaltensannahmen als auch auf bestimmte Situationsannahmen.

➔ SIT-ATMS

- Für jede Situation wird der Informationsgewinn für den bestmöglichen Beobachtungspunkt berechnet.

Vom ACS zur kompletten Diagnosesoftware

Details:

2) Vorschlag von Beobachtungspunkten

- Gegeben sei eine fest vorgegebene Einstellung der Control Inputs (also eine Situation)
- Berechne die Wahrscheinlichkeit, dass an einem bestimmten Beobachtungspunkt in der ausgewählten Situation der Wert x vorliegt (P(x)).
- Berechne den Desinformationsgehalt (**Entropie**) für einen bestimmten Beobachtungspunkt B nach der Formel $E(B) = - \Sigma_x P(x) \cdot \log P(x)$, wobei x alle möglichen Werte für B sind.
- Suche den Beobachtungspunkt B, an dem E(B) maximal ist.

➔ Die Kenntnis des tatsächlichen Werts bringt den größten Informationsgewinn

Vom ACS zur kompletten Diagnosesoftware

Zusammenfassung: Diagnose mit MDS

- MDS schlägt dem Benutzer systematisch Einstellungen im technischen System vor und gibt vor, wo er Werte messen soll.
- Dieser Vorgang wird solange wiederholt, bis eine eindeutige Diagnose mit hinreichend genauer Wahrscheinlichkeit gegeben werden kann.
- Die Diagnose gibt dem Benutzer explizit Hinweise, welche Komponenten defekt sind und wie er den Fehler beheben kann (wegen der Angabe des konkreten Fehlermodus)
- In der Wissensbasis kann für jeden Fehlermodus eine konkrete Abhilfemaßnahme hinterlegt werden.

➔ Die Ausgabe von MDS ist eine konkrete Reparaturanleitung

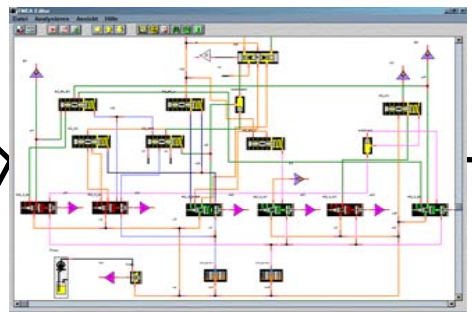
Model based troubleshooting

Weitere Aufgaben für MDS

Situationen (Werte an Control Inputs)

❶ Betriebs-vorgaben

$$\begin{pmatrix} i_1 \\ i_2 \\ \cdot \\ \cdot \\ \cdot \\ i_n \end{pmatrix}$$



Verhaltensmodell
(des Gesamtsystems)

Werte an Messpunkten

❸ Modell-variablen

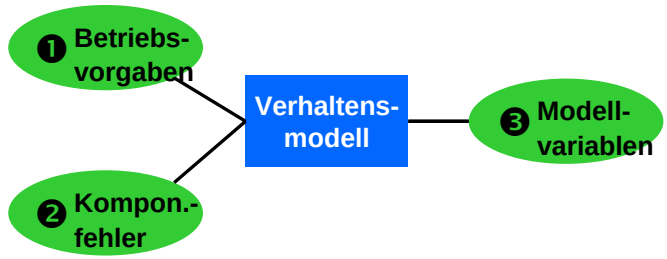
$$\begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

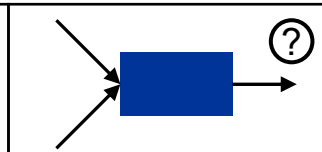
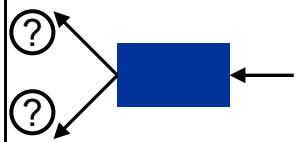
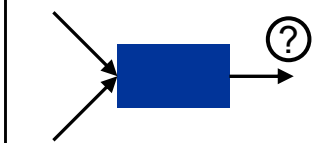
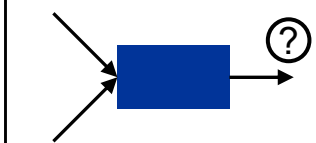
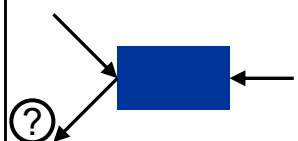
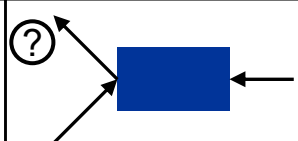
❷ Komponenten-fehler

$$\begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix}$$

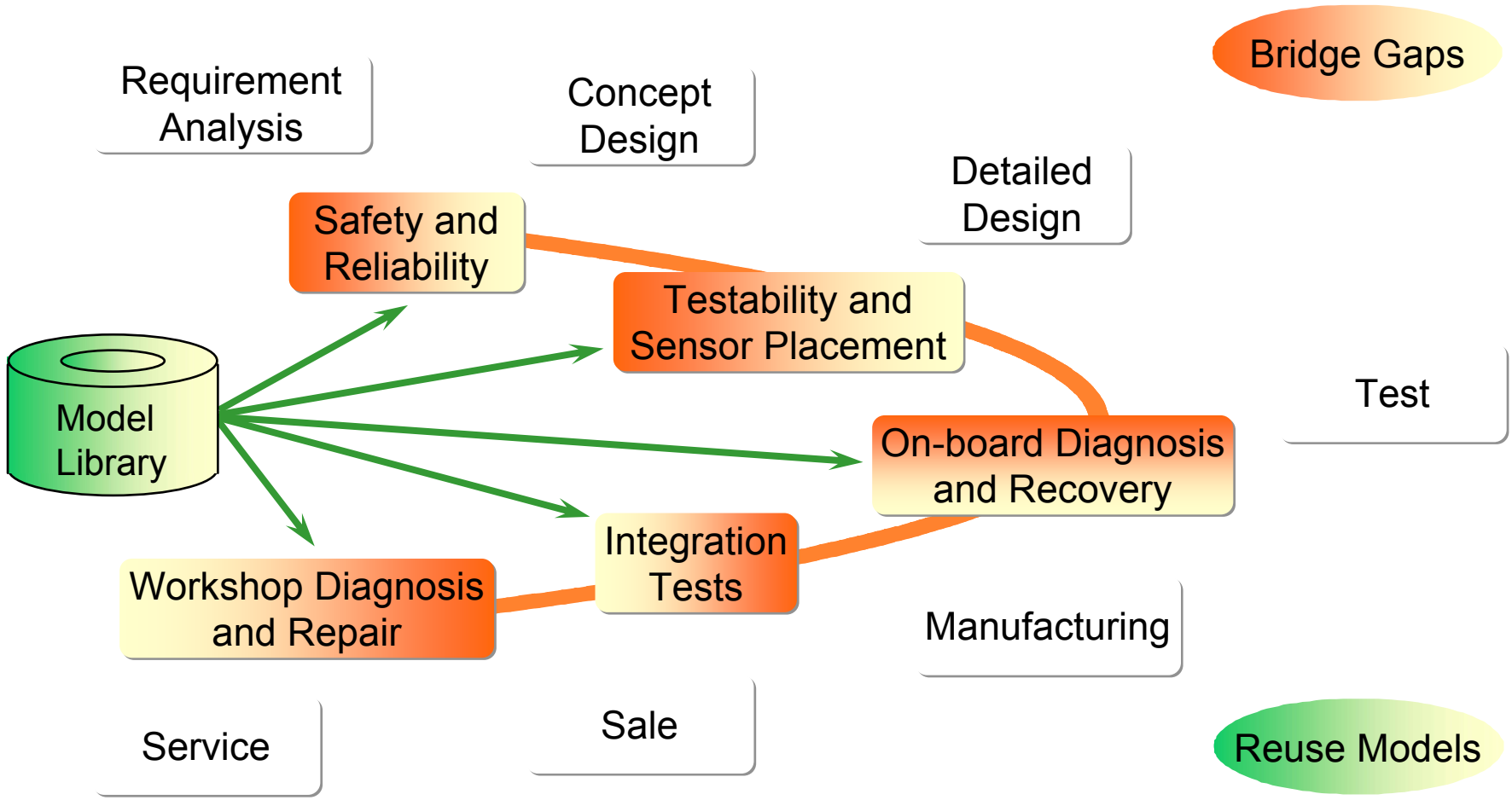
Verhaltensmodelle der einzelnen Komponenten

Weitere Aufgaben für MDS



A		FMEA	❶ vorgeben, ❷ vorgeben, ❸ = sicherheitsrelevante Eigenschaften klassifizieren (FMEA Events)
B		FTA	❸ sicherheitsrelevante Eigenschaften vorgeben, nach ❶ und ❷ suchen
C		Testbarkeits-analyse	❶ vorgeben, ❷ vorgeben, ❸ = vorhandene Sensoren bzw. Meßpunkte (Entscheidungsbaum generieren - eigenst. Fehlerkl.)
D		Sensor-plazierung	❶ vorgeben, ❷ vorgeben, ❸ = alle potenziell mögliche Sensoren / Meßpunkte (Entscheidungsbaum generieren - Sensorplazierung)
E		Diagnose	❶ vorgeben, ❸ messen, nach ❷ suchen = Komponentenfehler
F		Recovery	❷ vorgeben, ❸ Ziele vorgeben, nach ❶ suchen = mögliche Kompensationssteuerung

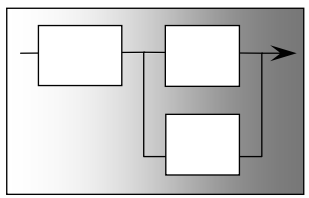
MDS in der Geschäftsprozesskette



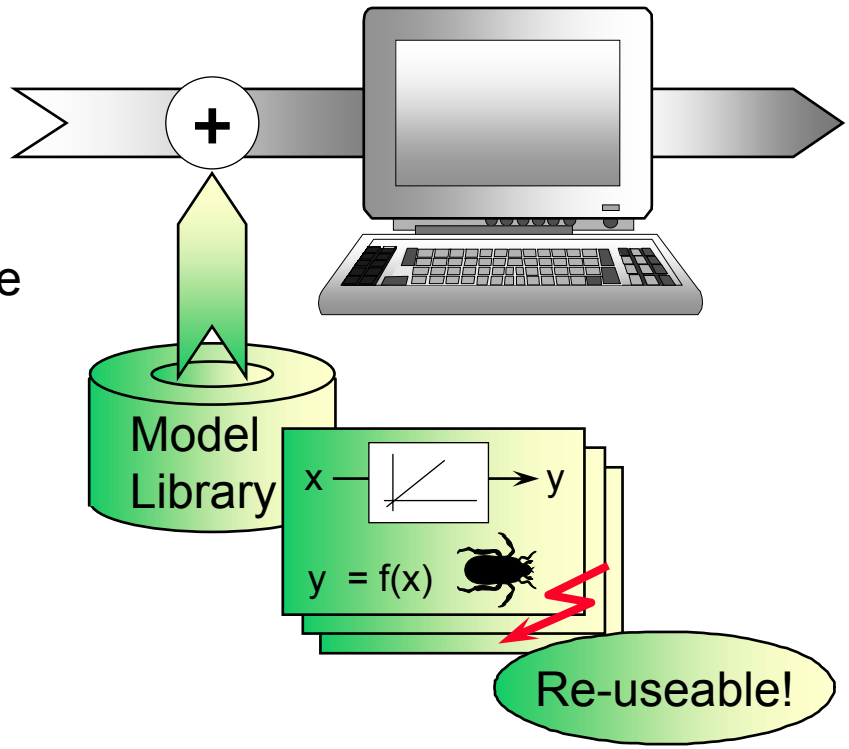
Zusammenfassung: Was kann MDS ?

System Design

Model-based Failure Analysis



System Structure



Analysis Task

- *Result*

Reliability/Safety

- *FMEA/FTA*

Testability

- *Sensor Placement*
- *Fault Classes*

Diagnosis

- *Decision Trees &*
- *Interactive Troubleshooting*