

# Hierarchische Designmodelle im Systementwurf mechatronischer Produkte

Dipl.-Ing. Dr. **Peter Hehenberger**, Prof. Dipl.-Ing. Dr. **Alexander Egyed**,  
Prof. Dipl.-Ing. Dr. **Klaus Zeman**  
Johannes Kepler Universität Linz, Österreich

## Kurzfassung

Innerhalb der verschiedenen Disziplinen der Mechatronik werden Informationen und Daten mit hohem Detaillierungsgrad erzeugt, die nur zum Teil in anderen Disziplinen (Domänen) benötigt werden. Dabei handelt es sich z.B. um Daten, die für das Gesamtverhalten bzw. Verständnis des Gesamtsystems niedrige Relevanz haben. Das Ziel ist nun, ein Systemmodell zu schaffen, das jene Informationen aus den einzelnen Domänen abbildet, die für andere Domänen von Bedeutung sind. Die Herausforderung ist dabei, dass das Wissen über das Gesamtsystem nicht gleich der Summe des Wissens aus den entsprechenden Domänen ist. Das Domänenwissen muss daher verallgemeinert (abstrahiert) und integriert werden. Die separate Behandlung von Systemmodellen und Domänenmodellen hat zur Folge, dass die Informationen auf verschiedene Stellen verteilt sind und daher die Rückverfolgbarkeit der Modelldaten (-parameter) gefordert ist, um Designentscheidungen nachvollziehen sowie Änderungen konsistent durchführen zu können. Die Umsetzung der vorgestellten Methode wird anhand der Analyse eines Spannsystems gezeigt.

## 1. Einleitung

Mechatronische Systeme entstehen durch die synergetische Integration von Maschinenbau, Elektrotechnik / Elektronik, Regelungstechnik und Informationstechnik. Die verschiedenen Disziplinen (im Folgenden auch Domänen genannt) verwenden unterschiedliche Entwicklungsprozesse, welche wiederum auf unterschiedlichen Modellbeschreibungen basieren [1 und 2]. Diese unterscheiden sich in der jeweiligen (domänenspezifischen) Sichtweise auf das System sowie im Detaillierungsgrad (z.B. Anzahl der betrachteten Systemparameter). Um den heutigen, vielfältigen Kundenwünschen gerecht zu werden, müssen fortschrittliche Produkte immer mehr Funktionen erfüllen. Durch ständig steigende Kundenwünsche müssen daher immer komplexere Produkte mit immer kürzeren Produkt-Entwicklungszyklen entstehen. Wegen des Drucks zur Verkürzung der Zeit für die Produkteinführung (time to market) und die dadurch erzwungene Reduktion der Entwicklungszeiten erhöht sich aber der Druck auf die einzelnen Domänen, die jetzt

schneller und effizienter kommunizieren und integrieren müssen (Vermeidung von Fehlern und Inkonsistenzen). Auch aus diesem Grund wird es immer wichtiger, die Produkteigenschaften schon in den frühen Phasen des Entwicklungsprozesses vorherzusagen, zu analysieren und zu bewerten. Die Mechatronik bietet durch Integration verschiedener technischer Einzeldisziplinen ausgezeichnete Potenziale für derartige Innovationen. Eine treffsichere, zumindest vergleichende Bewertung vieler verschiedener Lösungskonzepte wäre daher gerade in diesen „frühen Phasen“ der Produktentwicklung besonders hilfreich, stößt aber in der Praxis auf nahezu unüberwindbare Schwierigkeiten, da entweder die gefundenen Lösungskonzepte für eine Bewertung noch zu wenig detailliert sind oder der Aufwand zur Detaillierung aller aussichtsreichen Lösungskonzepte einfach zu hoch ist. Die Definition von hierarchischen Designmodellen im Systementwurf bei gleichzeitig möglichst effizienter Nutzung der immer leistungsfähigeren, rechnergestützten Werkzeuge bildet daher eine große Herausforderung für die Produktentwicklung. Von der modellbasierten Integration and Kommunikation profitieren dann die Entwicklungsingenieure der verschiedenen Domänen, da dies die Zusammenarbeit vereinfacht.

## **2. Hierarchische Designmodelle im Systementwurf**

Die in einem bestimmten Entwicklungsschritt erzeugten Ergebnisse (Informationen, Daten, Parameter) bilden Eingangsgrößen für den nachfolgenden Entwicklungsschritt, woraus zwangsläufig hierarchische Beziehungen der Produktdaten entstehen [3]. Solche Abhängigkeiten ergeben sich innerhalb der Disziplinen, aber auch zwischen den Disziplinen, wenn Daten bzw. Kenngrößen Domänen übergreifende Bedeutung haben. Zur Analyse und Synthese des Gesamtsystems sollen verschiedene Sichtweisen mit den dafür benötigten Systemeigenschaften herangezogen werden.

Ideal wäre es, das gesamte System in Form eines gemeinsamen, Domänen übergreifenden Modells abzubilden. Das Problem ist dabei, dass die verschiedenen Disziplinen unterschiedliche Modellierungskonzepte und Modelle bzw. Modellbeschreibungen verwenden, deren Integration nach wie vor zu wünschen lässt. Außerdem werden innerhalb der Disziplinen Informationen und Daten mit hohem Detaillierungsgrad erzeugt, die nur zum Teil in anderen Disziplinen benötigt werden. Dabei handelt es sich z.B. um Daten, die für das Gesamtverhalten bzw. Verständnis des Gesamtsystems niedrige Relevanz haben.

**Das Ziel ist nun, ein Systemmodell zu schaffen, das jene Information der einzelnen Domänen abbildet, die für andere Domänen von Bedeutung sind.** Die Herausforderung ist dabei, dass das Wissen über das Gesamtsystem nicht gleich der Summe des Wissens aus den entsprechenden Domänen ist, da jede Domäne im Prinzip in sich abschlossen ist

und daher nur beschränkt auf die Bedürfnisse der anderen Domäne eingehen kann. Das Domänenwissen muss daher verallgemeinert (abstrahiert) und integriert werden. Die separate Behandlung von Systemmodellen und Domänenmodellen hat zur Folge, dass die Informationen auf verschiedene Stellen verteilt sind und daher die Rückverfolgbarkeit der Modelldaten (-parameter) gefordert ist, um

- 1) Designentscheidungen nachvollziehen zu können
- 2) Änderungen konsistent durchführen zu können.

So kann z.B. die Änderung des Domänenwissens auch eine Änderung des Systemwissens bedeuten, was wiederum eine Änderung in einer anderen Domäne zur Folge haben kann, welche diese Information weiterverarbeitet. Außerdem ist (die Gültigkeit von) Wissen immer an einen bestimmten Zusammenhang gebunden (Kontextbezug), der bei konsistenter Verwendung des Wissens berücksichtigt und daher erfasst werden muss.

Die Entwicklungsarbeit in den einzelnen mechatronischen Disziplinen kann in verschiedene Hierarchiestufen zerlegt werden, die durch den zunehmenden Detaillierungsgrad bestimmt sind [4]. Inhalte der einzelnen Stufen stellen z.B. die Funktionsmodule und deren Parameter dar. Diese Modelle existieren bereits und werden in der Praxis auch eingesetzt. Wir bezeichnen diese Modelle als „Interne Modelle“ (d.h. jede Domäne hat ihr eigenes, internes Wissen). Es ist nicht das Ziel dieser Arbeit, die internen Modelle zu verändern (obwohl auch diese sicherlich niemals perfekt sind). Es ist vielmehr das Ziel dieser Arbeit, jenes Wissen der einzelnen Domänen zu extrahieren, das für (eine oder mehrere) andere Domänen relevant ist. Dieses Wissen wird in Form eines eigenen Modells abgebildet, das wir als „Externes Modell“ bezeichnen (d.h. die Information, die eine Domäne nach außen kommuniziert).

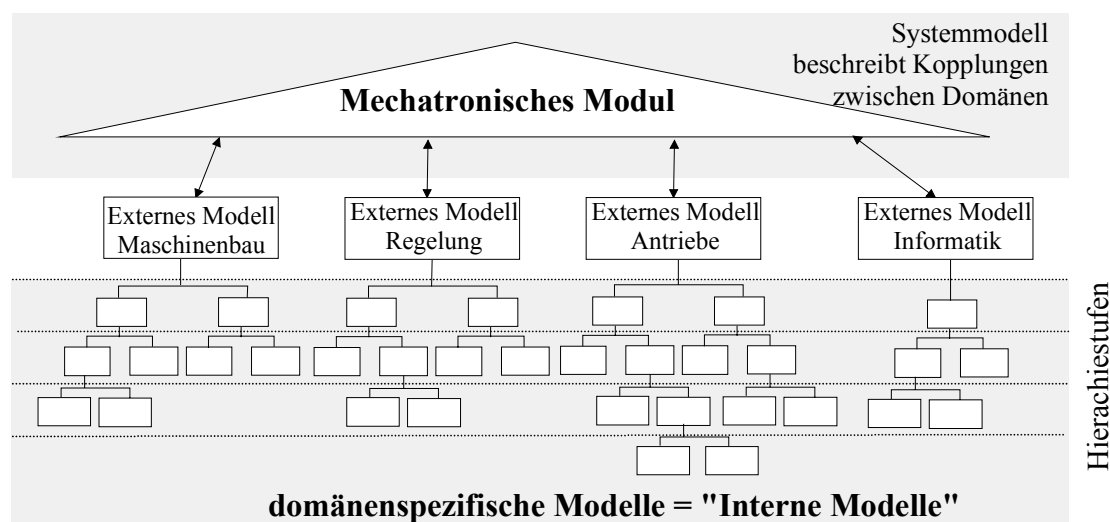


Bild 1: Mechatronische Domänen und Systemmodell

### **3. Anwendungsbeispiel und Implementierung**

#### **3.1 Modellierung eines Spannsystem**

In Rahmen dieses Abschnittes wird nun ein Spannelement näher betrachtet und mit Hilfe des mechatronischen Systemmodells analysiert. Die Aufgabe eines Spanners ist es, Werkstücke in möglichst kurzer Zeit mit einer bestimmten Kraft in einer bestimmten Lage zu fixieren. Nach der Fixierung erfolgt in der Regel die Bearbeitung des Werkstücks. Das heißt, das Spannelement muss eine große Anzahl von Anforderungen erfüllen. Dies sind z.B. die Möglichkeit des schnellen Spannens (wegen kurzer Taktzeiten), große Öffnungswinkel, kompakte Bauweise, geringer Energieverbrauch, verformungsarmes Spannen des Werkstücks sowie gute Zugänglichkeit zum Werkstück für die betreffenden Fertigungsarbeiten. Des Weiteren soll die Möglichkeit bestehen, dass der Spannkraftverlauf gemessen werden kann. Dies kann einerseits durch zusätzliche Elemente am Spanner (z.B. Dehnungsmessstreifen, Drucksensor) oder durch Messung von Systemgrößen (Strom, Druck) direkt am antreibenden Element (Elektromotor, Pneumatikzylinder) erfolgen. Als wichtige Anforderung ergibt sich hierbei, dass eine messtechnische Erfassung der Spannkraft, bzw. des Spannkraftverlaufs während des Schließvorganges möglich sein soll. In diesem Beispiel wird nun ein Spannelement in Basiskomponenten zerlegt. Dies sind:

- **Mechanische Komponenten:** In dieser Domäne werden die geometrische Form, Werkstoffe und Abmessungen der einzelnen Teile des Spanners betrachtet. Dies sind vor allem der bewegliche Teil, der Spannarm, sowie der feststehende Teil der Aufhängung. In der untersten Stufe (mit dem größten Detaillierungsgrad) werden die speziellen Anforderungen der Positionierung auf den Spannrahmen (z.B. schnelle Auswechselbarkeit, Montierbarkeit), sowie Zugänglichkeit z.B. für Schweißzangen usw. abgebildet.
- **Antriebstechnische Komponenten:** Nach Art der Ausführung muss unterschieden werden zwischen elektrischen, pneumatischen und hydraulischen Antriebselementen sowie den dazu benötigten Ansteuerungen und Systemressourcen.
- **Messtechnische Komponenten:** Die Ermittlung der Spannkraft kann z.B. mit Hilfe eines Messtasters oder über Dehnungsmessstreifen erfolgen, die auf dem Spannarm angebracht sind. Hiermit verbunden sind alle Einrichtungen zur Messsignalübertragung bzw. -auswertung, welche in weiterer Folge benötigt werden.

Ein Modell des Spannsystems samt seinen Funktionen und seiner Realisierung ist in Bild 2 schematisch dargestellt.

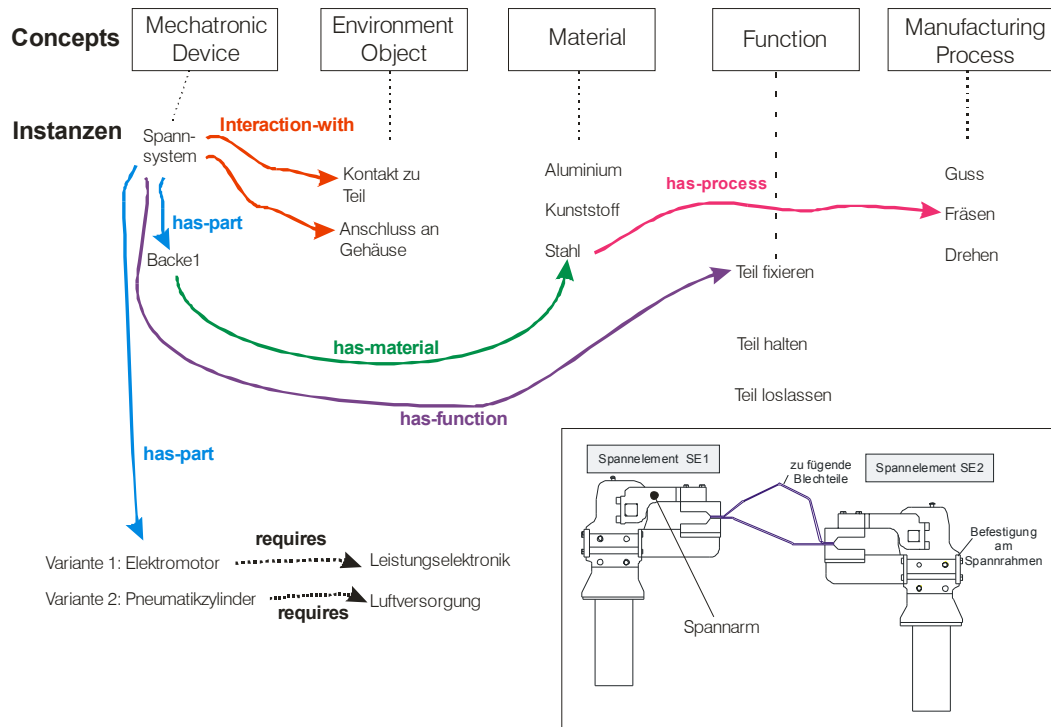


Bild 2: Modell des Spannsystems

Das Modell des Spannsystems beschreibt notwendige und optionale Komponenten des Systems. Manche Optionen gehören zusammen, aber manche Optionen passen nicht zusammen. So braucht z.B. ein Elektromotor eine Leistungselektronik (requires) und kann nicht zusammen mit einem Pneumatikzylinder eingebaut werden (zumindest nicht in diesem Modell). Aber gerade solche Entscheidungen werden oft domänenübergreifend getroffen. Z.B. eine Änderung von Elektromotor zu Pneumatikzylinder bedarf dann auch einer Änderung von Berechnungen über Leistungselektronik und Luftversorgung, welche teilweise oder vollständig in anderen Domänen stattfinden können.

Externe Modelle kommunizieren Wissen, das für andere Domänen wichtig ist. Jede Domäne hat ein externes Modell (in and out), und im Systemmodell werden diese externen Modelle integriert. Dieses Systemmodell ist damit auch die Basis dafür, dass die Domänen konsistent arbeiten und entscheiden können. Damit ein konsistenter Entwicklungsprozess über alle mechatronischen Disziplinen hinweg durchgeführt werden kann, ist es notwendig, die Konsistenz der einzelnen Modelle und Modelldaten mit Hilfe von Regeln zu überwachen.

### 3.2 Implementierung

Eine Konsistenzregel definiert sich üblicherweise durch einen Geltungsbereich (in welchem sie ausgeführt werden soll) und einer Bedingung, die in diesem Geltungsbereich wahr sein

muss. Eine Konsistenzregel liefert üblicherweise einen Bool'schen Wahrheitswert zurück – wahr falls das Modell der Bedingung standhielt, sonst falsch. Letzteres bezeichnet man als Inkonsistenz. Üblicherweise müssen alle Konsistenzregeln wahr sein, damit das Modell als konsistent gilt. Der verfolgte Ansatz ([5]) erlaubt es, Konsistenzregeln auf inkrementelle Art zu entdecken, d.h. die Konsistenz des Modells wird nur aufgrund der Änderungen wiederevaluiert, wobei mit jeder Änderung nur jene Konsistenzregeln überprüft werden, die von der Änderung betroffen waren. Bezug nehmend auf Bild 2 werden die Konsistenzregeln in „Domain-Konsistenzregeln“ (Regel 1 und 2) und „Meta-domain- Konsistenzregeln“ (Regel 3) unterteilt und sind nachfolgend angegeben. In der Softwaretechnik werden zur Beschreibung von Regeln oft formale Sprachen eingesetzt. Wir verwenden nachfolgenden eine vereinfachte Darstellung der Object Constraint Language, welche Teil der Unified Modeling Language ist:

Konsistenzregel 1: applies to <Elektromotor>  
requires(this).exists(part | part is <Leistungselektronik>)

Konsistenzregel 2: applies to <Spannsystem>  
has-part(this).exists(part | part is <Elektromotor>) xor has-part(this).exists(part | part is <Pneumatikzylinder>)

Konsistenzregel 3: applies to instance of <Mechatronic Device>  
if (has-part(this) is empty) then not(has-material(this) is empty)

Regel 1 wird nur auf Elemente angewandt, die vom Domaintype „Elektromotor“ sind. Das Schlüsselwort „this“ bezieht sich also auf einen Elektromotor, requires(this) ermittelt welche anderen Elemente eine „requires“ Beziehung mit diesem Elektromotor hat. Die Regel ist konsistent, wenn unter dieses Elementen eine vom Typ Leistungselektronik gibt. Regel 2 gilt für jedes Spannsystem. Es stellt sicher, dass wenn ein Spannsystem eine has-part Beziehung zu einem Elektromotor hat, dann darf es keine andere has-part Beziehung zu einem Pneumatikzylinder geben (und umgekehrt). Regel 3 definiert, dass jedes Mechatronic Device entweder aus einem Material besteht (has-material), oder aber Teile hat, welche aus Materialien bestehen.

Bei komplexen Modellen mit mehr als tausend Modellelementen ist es schwierig, neue Inkonsistenzen im Zuge von Modelländerungen zu erkennen. Ein automatisierter Ansatz ist daher erforderlich. Der verfolgte Ansatz ist nicht nur inkrementell, sondern auch sehr schnell, wie anhand einer größeren Anzahl von Konsistenzregeln (aus dem Software Bereich) gezeigt wurde [5]. Zur Unterstützung der schnellen, inkrementellen Überprüfung von

Designänderungen, identifiziert das Tool alle Modellelemente, die von bestimmten Konsistenzregeln beeinflusst werden. Eine Konsistenzregel muss nur dann neu bewertet werden, wenn sich eines dieser Modellelemente ändert. Die Ermittlung ist einfach im Prinzip, aber es ist nicht möglich vorherzusagen, welche Modellelemente von bestimmten Konsistenzregeln beeinflusst werden. Dieser Ansatz umgeht dieses Problem, indem das Laufzeit-Verhalten der Konsistenzregeln während ihrer Bewertung berücksichtigt wird. Dazu werden „Profiling-Daten“ verwendet, um eine Korrelation zwischen Modell-Elementen und Konsistenz (und Inkonsistenzen) zu beschreiben. Auf der Grundlage dieser Korrelation, kann entschieden werden, wann diese neu zu bewerten sind. Der Entwicklungsingenieur hat die Möglichkeit, den Fehler sofort zu beheben, kann ihn aber auch ignorieren (d.h., Leben mit Inkonsistenzen [6]).

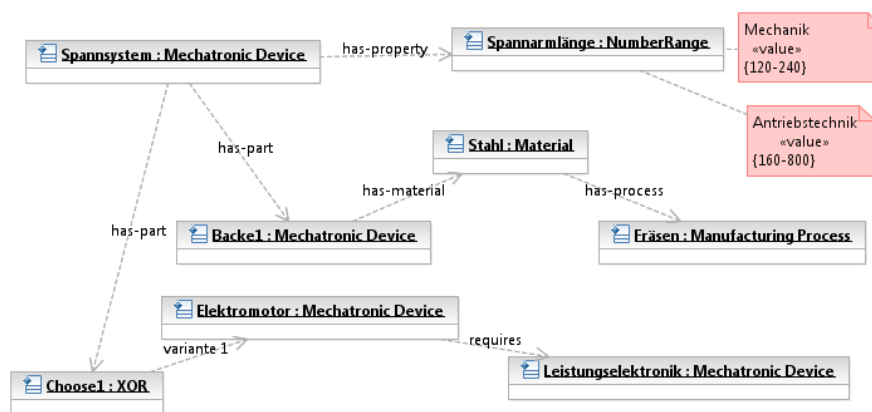


Bild 3: Implementierungsmodell für Konsistenzüberprüfung

Es ist auch interessant zu wissen, dass Systemvariationen nicht nur aufgrund von vordefinierten Alternativen existieren (Elektromotor oder Pneumatikzylinder). Variationen existieren auch auf Grund von Unklarheiten oder sogar Uneinigkeiten unter den Entwicklungsingenieuren. Diese können dann solche Variationen definieren. In Bild 3 sehen wir, dass sich die Spannarmlänge  $L$  auf Grund der Domänen Mechanik und Antriebstechnik unterschiedlich einschränkt. So muss  $L$  aufgrund von gegebenen mechanischen Eigenschaften, wie dem erforderlichen Mindestabstand zwischen Spanner und Einspannstelle in einem Bereich von 120mm bis 240mm liegen. Bei der Erfüllung einer maximalen Spannkraft von  $F_S = 1\text{kN}$  ergibt sich ein von der antriebstechnischen Modellsäule festlegbarer Bereich von 160mm bis 800mm. Daraus ergibt sich nun ein „konsistenter Designbereich“ von 160mm bis 240mm (siehe Bild 3). Die entsprechende Konsistenzregel stellt daher sicher, dass beide Domänen unter konsistenten Annahmen arbeiten. Die Konsistenzregel lautet dazu:

Konsistenzregel 4: applies to <NumberRange>

If (value-constraints(this).size>1 and min(max-values(value-constraints(this)))>max(min-values(value-constraints(this))))

Regel 4 überprüft Elemente vom Type „NumberRange“. Wenn einem NumberRange („this“) mehrere Wertebereiche (Value Constraint) zugewiesen werden (size>1), dann ermittelt die Regel den kleinsten der Maximalwerte und den größten der Minimalwerte und stellt sicher, dass der kleinste Maximalwert größer ist als der größte Minimalwert.

#### **4. Zusammenfassung und Ausblick**

Die Umsetzungs- und Nutzenpotenziale der vorgestellten Konzepte zur hierarchischen Modellbeschreibung von mechatronischen Systemen in den frühen Phasen des Entwicklungsprozesses liegen nicht nur in der Neuentwicklung, sondern auch in der Analyse und Adaption bestehender Lösungen, wenn diese an veränderte Randbedingungen angepasst werden sollen. Es wurde ein Systemmodell beschrieben, das ausgehend aus dem Domänenwissen („Internes Modell“) jenes Wissen verwendet, das für die anderen Domänen relevant ist. Dieses Wissen wird in Form eines eigenen Modells abgebildet, das wir als „Externes Modell“ bezeichnen (d.h. die Information, die eine Domäne nach außen kommuniziert). Die Integration ist nicht einseitig, so zwingt das Systemmodell kein Wissen auf, dem sich Domänen unterordnen müssen, sondern ermöglicht, dass Domänen ihrerseits Wissen in das Systemmodell weiterleiten können. Es wurde skizziert, wie die softwaretechnische Umsetzung von internen und externen Modellen erfolgen kann.

#### **5. Literatur**

- [1] VDI 2206: Entwicklungsmethodik für mechatronische Systeme. Berlin: Beuth 2003
- [2] De Silva C. W.: Mechatronics – an integrated approach. Boca Raton: CRC Press 2005
- [3] Ehrlenspiel K.: Integrierte Produktentwicklung. München: Hanser-Verlag 2007
- [4] Hehenberger P., Zeman K.: The Role of hierarchical Design Models in the Mechatronic Product Development Process. In International Symposium Series on Tools and Methods of Competitive Engineering, Izmir, Türkei, 21.-25.April 2008
- [5] Egyed A.: Instant Consistency Checking for the UML. In International Conference on Software Engineering (ICSE) 2006, pp. 381-390
- [6] Fickas S., Feather M., Kramer J.: Proceedings of ICSE-97 Workshop on Living with Inconsistency. Boston, USA, 1997