# Abgleich von Teilmodellen in den frühen Entwicklungsphasen

Guy Gorek, Udo Kelter
Fachbereich Elektrotechnik und Informatik
Universität Siegen
{gorek,kelter}@informatik.uni-siegen.de

2010-11-29

**Zusammenfassung**

Teilmodelle sind Daten-, Zustands- oder andere Modelle, die die individuellen, initialen Anforderungen eines einzelnen Stakeholders repräsentieren. Dieses Papier adressiert das Problem, wie autark entstandene Teilmodelle zu einem Gesamtmodell abgeglichen werden können. Zunächst analysieren wir die Eignung bisher vorhandener Mischverfahren für Modelle und zeigen, daß sie wesentliche Defizite aufweisen, i.w. weil sie die typischen Verhältnisse der späten Entwicklungsphasen unterstellen. Zur Behebung dieser Defizite schlagen wir modifizierte Matching-Algorithmen, die die relativ große Unsicherheit der Teilmodelle berücksichtigen, sowie eine Methode zur interaktiven Korrektur von Korrespondenzen vor.

## 1 Einführung

In frühen Phasen einer Anforderungsanalyse müssen in mittleren bis großen Projekten viele Stakeholder befragt werden, die unterschiedliche Wissenstände, subjektive Bedarfe, Grade an Betroffenheit und Interessen haben. Es ist sinnvoll, die Anforderungen individuell pro Stakeholder in Form von Modellen passenden Typs (Daten-, Zustands- o.a. Modelle) zu erfassen. Solche Modelle bezeichnen wir hier als **Teilmodelle**, da sie nur einen Teil der Anforderungen repräsentieren[1]. Teilmodelle müssen oft autark erfaßt werden, z.B. bei global verteilter Softwareentwicklung, aber oft auch wegen des überfüllten Terminkalenders wichtiger Knowhow-Träger. Teilmodelle ermöglichen es, Wahrnehmungsdifferenzen der Stakeholder explizit und zum Gegenstand von Diskussionen zu machen. Als Beispiel zeigt Bild 1 zwei Teilmodelle, die bei der initialen Modellierung eines Aktionshauses vorhanden könnten, nachdem ein Anbieter (Teilmodell 1) und eine Auktionator (Teilmodell 2) befragt wurden.

[1] Andere Bezeichnungen hierfür sind **Viewpoints** [3] oder Modelle in **Sichten** [6].

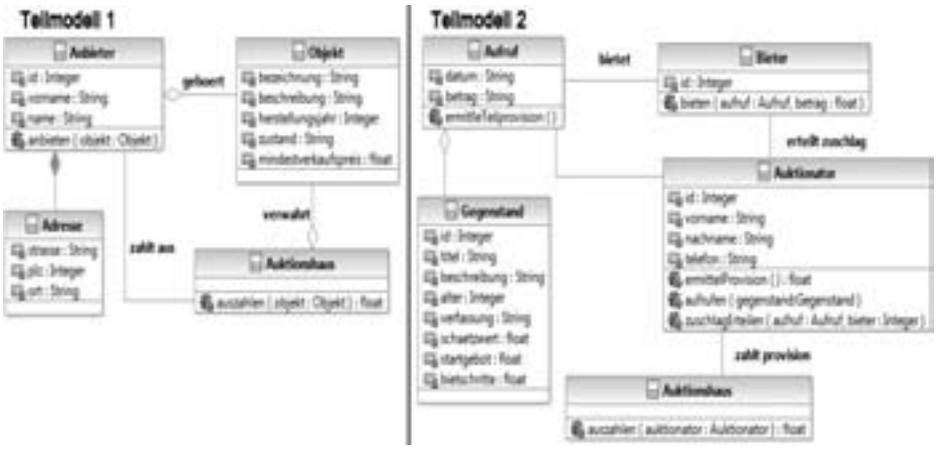


Abbildung 1: Beispiel für zwei Teilmodelle

Diskrepanzen zwischen den Anforderungen verschiedener Stakeholder müssen bereinigt werden, bevor die entsprechende Funktionalität konkret realisiert wird. Hierzu müssen die Diskrepanzen identifiziert und den Stakeholdern geeignet präsentiert werden. Technisch gesehen läuft dies auf den Vergleich und das Mischen der Teilmodelle hinaus.

Das Vergleichen und Mischen von textuellen Entwicklungsdokumenten, die überwiegend in den späten Entwicklungsphasen auftreten, ist tägliche Praxis. Für Modelle werden seit einigen Jahren analoge Algorithmen und Werkzeuge entwickelt. Der Gedanke liegt nahe, die mittlerweile verfügbaren Verfahren für Modelle auch in den sehr frühen Entwicklungsphasen auf Teilmodelle anzuwenden. Technisch ist das durchaus möglich, wie z.B. in [3] gezeigt wird. Es stellt sich allerdings heraus, daß bisherige Verfahren nicht wirklich den Anforderungen genügen, die an Mischwerkzeuge für die sehr frühen Phasen gestellt werden. Bisherige Verfahren unterstellen nämlich stillschweigend analoge Arbeitsprozesse und Randbedingungen, die beim Mischen von (graphischen oder textuellen) Dokumenten in den späten Entwicklungsphasen vorliegen; dies trifft auch auf bisher publizierte Listen von Anforderungen an Mischwerkzeuge für Modelle (z.B. [2]) zu.

Die Anforderungen, die Mischwerkzeuge für die sehr frühen Phasen erfüllen müssen, sind bisher nur oberflächlich analysiert worden [1, 11][2], was nicht zuletzt damit erklärbar ist, daß Mischwerkzeuge für Modelle erst seit kurzem realisierbar sind. Abschnitt 2 analysiert daher zunächst, inwiefern sich die Arbeitsprozesse, in denen Modelle verglichen und gemischt werden, in den späten bzw. sehr frühen Phasen unterscheiden und inwiefern wesentlich andere Randbedingungen vorliegen.

[2]Wenn in den o.g. Quellen vom Vergleich oder Mischung von Modellen die Rede ist, ist meist die Integration von Modellen unterschiedlichen Typs gemeint, die kein Gegenstand dieses Papiers ist.

Wir kommen zugleich zur Erkenntnis, daß bisherige Algorithmen und Mischwerkzeuge diese Arbeitsprozesse nur sehr unzureichend unterstützen.

Die folgenden Abschnitte beschreiben neue oder geänderte Funktionen, die Mischwerkzeuge anbieten sollten, um diese Defizite zu beheben. Eine neue Funktion ist die interaktive Korrektur von Korrespondenzen (Abschnitt 4). Abschnitt 5 beschreibt Lösungen für das Problem, wie die Matching-Algorithmen der Mischwerkzeuge die Unvollständigkeit bzw. Ungenauigkeit der Teilmodelle berücksichtigen können.

# 2 Vergleichs- und Mischprozesse in den frühen Entwicklungsphasen

## 2.1 Hintergrund: Hauptfunktionen in Mischwerkzeugen

Um die Besonderheiten der Vergleichs- und Mischprozesse in den frühen Entwicklungsphasen diskutieren zu können, führt dieser Abschnitt zunächst die Hauptfunktionen ein, die Mischwerkzeuge beinhalten.

Eine Mischung von zwei Dokumenten basiert stets auf einem Vergleich, in dem die korrespondierenden Dokumentelemente ermittelt werden. Ein Paar korrespondierender Dokumentelemente wird als **Korrespondenz** bezeichnet, eine Menge von Korrespondenzen als **Matching**. Mischwerkzeuge haben daher zwei Hauptphasen, die in weitere Unterphasen gegliedert sind:

1. Vergleich
   1.1 Berechnung eines Matchings
   1.2 Bestimmung der Differenz, Vorbereitung der Differenzanzeige
2. Mischung
   2.1 Berechnung von Konflikten
   2.2 Anzeige der Differenz und der Konflikte
   2.3 Erfassung von Mischentscheidungen
   2.4 Erzeugung einer neuen Dokumentversion

Paare korrespondierender Dokumentelemente werden nur *einmal* in das Mischergebnis übernommen. Die Wahl der Korrespondenzen hat daher einen ganz *erheblichen Einfluß auf das Mischergebnis*.

Dokumentelemente, die kein korrespondierendes Element im anderen Modell haben, nennen wir **spezielle** Elemente. Zwei spezielle Elemente, die aus verschiedenen Modellen stammen, können unverträglich sein, weil sie nicht zugleich in das Mischergebnis übernommen werden können. Potentiell sind alle Paare spezieller Elemente unverträglich; die Konflikterkennung versucht, möglichst viele dieser Paare als unkritisch zu erkennen. Die restlichen Paare werden als **Konflikte** bezeichnet, hier ist jeweils eine manuelle **Mischentscheidung** durch den Entwickler notwendig.

Beim 3-Wege-Mischen ist zusätzlich eine Basisversion vorhanden; statt spezieller Elemente werden hier Änderungen gegenüber der Basisversion betrachtet, wobei es sich auch um Löschungen handeln kann.

## 2.2 Unsicherheit der Teilmodelle

Eine erste wesentliche Besonderheit von Teilmodellen in den frühen Phasen ist deren Unsicherheit: Am Anfang eines Analyseprozesses ist das Problemverständnis vieler Stakeholder noch unvollkommen, und die Anforderungen sind teilweise unsicher oder Verhandlungssache. Hierdurch wird vor allem die Korrektheit der berechneten Matchings beeinträchtigt. Wir verstehen hier den Begriff unsicher in einem sehr weiten Sinn: Teilmodelle sind oft *unvollständig* und enthalten ggf. nur 30 - 50% der späteren vollständigen Spezifikationen. Teilmodelle sind ferner oft *unpräzise bzw. unzuverlässig*: die Wahrscheinlichkeit ist relativ hoch, daß vorhandene Angaben noch geändert werden müssen, weil z.B. anfänglich zufällig andere Begriffe verwendet wurden - Bild 1 enthält viele Beispiele hierzu.

Bisherige Matching-Algorithmen berechnen in derartigen Situationen Matchings mit vielen Fehlern, also fehlenden bzw. unzutreffenden Korrespondenzen, die das Mischergebnis völlig wertlos machen können.

## 2.3 Explizite Ungenauigkeitsangaben

Potentiell sind alle Modellelemente von Teilmodellen unsicher im Sinne des vorigen Abschnitts. Daneben ist oft für *einzelne Modellelemente* bekannt, daß diese unfertig sind und noch überarbeitet werden müssen. Es ist sinnvoll, Informationen hierüber explizit zu erfassen. Beispiele für entsprechende Sprachkonstrukte finden sich in der Modellierungssprache SeeMe [3, 5]. SeeMe enthält gegenüber der UML vereinfachte Daten-, Funktions- und weitere Modelle. SeeMe ist für die frühen Phasen optimiert und definiert mehrere Sprachelemente, mit denen explizit angegeben werden kann, daß ein Modellelement unvollständig oder ungenau ist. Diese Sprachelemente können auf die üblichen (UML-) Modelltypen übertragen werden.

Bisherige Matching-Algorithmen können derartige explizite Ungenauigkeitsangaben in Modellen überhaupt nicht verarbeiten.

## 2.4 2- vs. 3-Wege-Mischen

Die Vergleichs- und Mischverfahren, die man von Systemen wie SVN oder CVS kennt und die man in den späten Entwicklungsphasen benutzt, unterstützen das parallele Editieren von Dokumenten durch ein nichtinteraktives 3-Wege-Mischen. Teilmodelle entstehen hingegen anfänglich z.B. durch Befragung einzelner Stakeholder weitgehend unabhängig voneinander. Das 3-Wege-Mischen ist daher praktisch nicht anwendbar, weil entweder gar keine Basisversion existiert oder diese noch sehr unvollständig ist und überwiegend neue Teile der Modelle unabhängig voneinander entstehen.

Es liegt i.w. die Situation des 2-Wege-Mischens vor, für die bisherige Matching-Algorithmen kaum Unterstützung bieten.

## 2.5 Unterstützung von Mischentscheidungen

Bei Mischungen spätphasiger Dokumente spielt die Reduktion der manuell zu behandelnden Konflikte und Automatisierung möglichst aller Mischentscheidungen eine wichtige Rolle, zumal die Dokumente groß und die Änderungen zahlreich sein können. Möglich ist diese Automatisierung aber nur mit Hilfe einer Basisversion, also beim 3-Wege-Mischen. Beim 2-Wege-Mischen können prinzipiell keine Mischentscheidungen automatisiert werden. Man kann allenfalls passend zum jeweiligen Modelltyp die grundlegende syntaktische Verträglichkeit von speziellen Modellelementen prüfen. Im Endeffekt muß der Entwickler alle Entscheidungen selber treffen.

Die Automatisierung von Mischentscheidungen ist daher für Teilmodelle praktisch irrelevant. Wichtiger ist es, die manuellen Mischentscheidungen zu erleichtern, indem den Entwicklern unterstützende Information geliefert wird. Bisherige Matching-Algorithmen liefern keine derartigen Informationen.

# 3 Hintergrund: Verfahren zur Berechnung von Matchings

In den folgenden Abschnitten analysieren wir im Detail, warum bisherige Matching-Algorithmen und Mischwerkzeuge bei Teilmodellen unzureichend arbeiten.

Als Vorbereitung skizziert dieser Abschnitt die bekannten wesentlichen Verfahren zur Berechnung von Matchings. Diese können in mehrere Gruppen eingeteilt werden [2, 9], die aber jeweils nur unter bestimmten Voraussetzungen, die hier meist nicht erfüllt sind, nutzbar sind:

- Verfahren auf Basis von Editorprotokollen und Verfahren auf Basis von persistenten Identifizierern [10]: diese eignen sich prinzipiell nicht für *neu erzeugte* Modelle oder Modellfragmente.
- Verfahren auf Basis von Signaturen und semantik-basierte Verfahren: diese sind nicht geeignet für *unvollständige* Modelle.

Im Endeffekt eignen sich nur ähnlichkeitsbasierte Verfahren in der hier vorliegenden Situation. Ähnlichkeitsbasierte Verfahren [8, 12] zielen darauf, möglichst ähnliche Elemente der beiden Modelle als korrespondierend zu bestimmen.

Hierzu werden für jeden einzelnen Modellelementtyp ähnlichkeitsrelevante Merkmale und deren Gewichtung definiert. Bei Modellelementtypen, die einen Namen haben, ist z.B. der Name ein ähnlichkeitsrelevantes Merkmal mit einer sehr hohen Gewichtung (25 - 50 %), da Elemente mit gleichem Namen sehr wahrscheinlich korrespondieren. Die Merkmale und deren Gewichtung müssen manuell definiert werden, wobei die typischen Strukturen und Editiervorgänge eines Modelltyps zu berücksichtigen sind.

Beim Vergleich zweier Modelle wird zunächst für jedes Element $e1$ des ersten Modells die Ähnlichkeit zu jedem Element $e2$ des zweiten Modells mit gleichem Typ berechnet. Sofern die Ähnlichkeit von $e1$ und $e2$ einen Mindestwert erreicht, wird $e2$ in die **Präferenzliste** von $e1$ aufgenommen und umgekehrt $e1$

in die Präferenzliste von $e2$ (Ähnlichkeiten sind symmetrisch). Ein Matching-Algorithmus berechnet auf Basis der Präferenzlisten unter Berücksichtigung der Dokumentstruktur ein Matching, wobei versucht wird, anhand von Heuristiken die Ähnlichkeiten korrespondierender Modellelemente zu optimieren.

# 4 Interaktive Korrektur von Korrespondenzen

## 4.1 Unsicherheit von Korrespondenzen

Je nach der Struktur der Ähnlichkeiten können die Präferenzlisten suspekt und die daraus abgeleiteten Korrespondenzen unsicher sein. Als Beispiel betrachten wir zwei Elemente $e$ und $f$ aus dem ersten Modell und die jeweils ähnlichsten Elemente, also "Kandidaten" für eine Korrespondenzbildung, aus dem zweiten Modell:

| Ähnlichkeiten zu ... | $e$ | $f$ |
|---|---|---|
| ähnlichster Kandidat | 0.98 | 0.65 |
| 2.-ähnlichster Kandidat | 0.68 | 0.63 |
| 3.-ähnlichster Kandidat | 0.61 | 0.63 |

Bei $e$ ist der ähnlichste Kandidat eine sehr sichere Wahl, weil der zweitähnlichste erheblich unähnlicher ist und, wenn wir die Mindestähnlichkeit mit 0.6 annehmen, nur knapp über dieser Schwelle liegt. Bei $f$ ist der ähnlichste Kandidat eine unsichere Wahl, mit einer signifikanten Wahrscheinlichkeit ist die mit ihm gebildete Korrespondenz falsch: alle Kandidaten liegen nur knapp über der Mindestähnlichkeit, und die geringen Ähnlichkeitsunterschiede können durch Zufälligkeiten verursacht sein (z.B. zufällig ähnliche Bezeichnungen). Im Extremfall weisen zwei Kandidaten die gleiche Ähnlichkeit auf und man muß den Zufall entscheiden lassen.

Suspekte Präferenzlisten können auch beim Vergleich von Modellen aus den späten Phasen auftreten, sind dort aber selten, weil die Modelle detailreich und weitgehend fehlerfrei sind, und machen keine besonderen Maßnahmen nötig. Bei Modellen aus den sehr frühen Phasen sind suspekte Präferenzlisten hingegen häufig: durch die Unvollständigkeit der Modelle ist die Datenbasis, auf der Modelle verglichen werden, wesentlich verkleinert.

Das Mischwerkzeug sollte auf unsichere Korrespondenzen durch entsprechende Warnungen hinweisen, z.B. wie in Bild 2 vorgeschlagen. Unsicher sind Korrespondenzen, wenn eines oder mehrere der folgenden Indizien vorliegen (weitere Indizien werden später vorgestellt):

- die Ähnlichkeit der beiden Modellelemente liegt nur knapp über der Mindestschwelle
- eines der beiden Modellelemente hat nur eine geringfügig bessere Ähnlichkeit als der nächstplazierte Kandidat
- es gibt deutlich ähnlichere Kandidaten, die der Matching-Algorithmus schon vorher für die Bildung von anderen Korrespondenzen verbraucht hat

## 4.2 Vorgaben für Matchings

Schon aus dem obigen simplen Beispiel wird klar, daß es zu falsch berechneten Korrespondenzen kommen kann und Entwickler imstande sein sollten, Korrespondenzen zu korrigieren.

Konzeptuell sind solche Korrekturen Vorgaben für die Korrespondenzbildung. Eine **positive** Vorgabe legt genau eine Korrespondenz fest, eine **negative** Vorgabe verbietet eine Korrespondenz.

Das Mischwerkzeug muß geeignete Bedienschnittstellen anbieten, um diese Vorgaben bequem erfassen zu können. Nach der Erfassung neuer Vorgaben muß das Matching neu berechnet werden, da i.a. die Darstellung der Differenz komplett neu aufgebaut werden muß.

Die Erfassung von Vorgaben ist nur sinnvoll für Modellelemente, die in der Sichtwelt von Entwicklern einzeln identifizierbar sind und i.d.R. autark angelegt und gelöscht werden können. Nicht sinnvoll ist es, Korrespondenzen zwischen sehr kleinen Modellelementen, z.B. Typen von Parametern von Operationen, vorzugeben. Die Modellelementtypen, für die Vorgaben erfaßt werden sollen, müssen daher für jeden Modelltyp individuell bestimmt werden.

Konzeptuell sind die positiven bzw. negativen Vorgaben einfache Listen von Korrespondenzen, die neben den Modellen weitere Eingabeparameter für den Matching-Algorithmus darstellen. Bisherige Matching-Algorithmen unterstützen positive bzw. negative Vorgaben überhaupt nicht, sind aber prinzipiell dahingehend erweiterbar; Details dieser Erweitungen hängen stark von den jeweiligen Implementierungsstrukturen ab.

## 4.3 Ein GUI zur Korrektur von Korrespondenzen

Abb. 2 zeigt GUI-Elemente für die manuelle Korrektur von Korrespondenzen. Links ist ein Baum dargestellt, in dem alle Elemente der beiden verglichenen Modelle repräsentiert sind. Korrespondierende Elemente werden nur durch einen gemeinsamen Knoten dargestellt[3].

Eine falsche Korrespondenz kann aufgelöst werden, indem der Knoten, der das Paar korrespondierender Elemente repräsentiert, selektiert wird und über das Kontextmenü die Funktion “Korrespondenz auflösen” aufgerufen wird. Danach muß das Matching neu berechnet werden, da die beiden zuvor an der Korrespondenz beteiligten Elemente anschließend jeweils durch einen eigenen Knoten repräsentiert werden.

Für ein Element, *das aktuell keinen Korrespondenzpartner hat*, kann ein Korrespondenzpartner nach Aufruf der Funktion “Korrespondenz festlegen” manuell bestimmt werden. Abb. 2 zeigt rechts ein entsprechendes Fenster. Dieses zeigt die Namen der möglichen Korrespondenzpartner an, von denen einer zu selektieren ist, um die Korrespondenz festzulegen. Elemente, welche bereits an einer Korrespondenz beteiligt sind, werden ausgegraut dargestellt und sind nicht mehr selektierbar.

[3] Diese Darstellungsform wird auch als Vereinigungsdokument bezeichnet, s. [7]. Stattdessen könnten auch andere Darstellungsformen für Differenzen genutzt werden, s. [7].

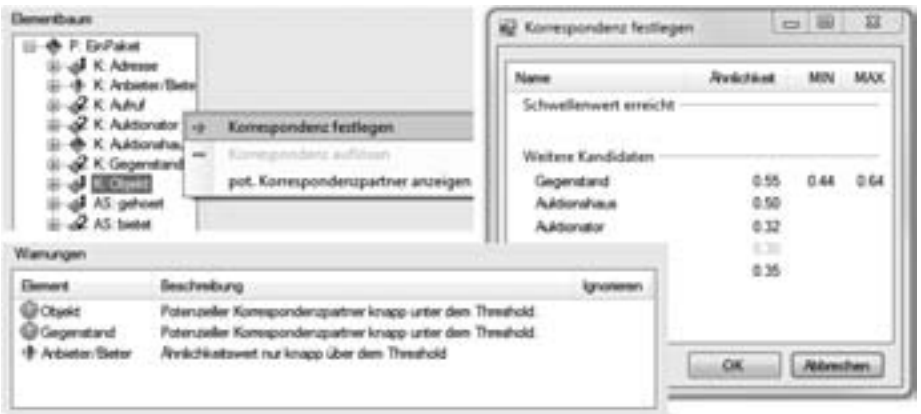


Abbildung 2: GUI-Elemente

Die Liste ist in zwei Bereiche unterteilt. Im oberen Bereich ("Schwellenwert erreicht") wird die Präferenzliste für dieses Element angezeigt, also alle Elemente des anderen Modells, die die Mindestähnlichkeit aufweisen. Im Beispiel in Abb. 2 ist die Präferenzliste leer. Für alle in der Präferenzliste angezeigten Elemente tritt einer der folgenden Fälle zu:

- das Element wurde für eine andere Korrespondenz benutzt (grau dargestellt)
- die zunächst berechnete Korrespondenz mit diesem Element wurde manuell aufgelöst

Der untere Bereich enthält Elemente, deren Ähnlichkeitswert unter der Mindestähnlichkeit liegt. Die Spalten MIN und MAX werden später erklärt.

# 5 Matching-Algorithmen für die frühen Phasen

Im folgenden stellen wir Modifikationen an ähnlichkeitsbasierten Matching-Algorithmen vor, mit denen das Problem der Unvollständigkeit und Unsicherheit der Teilmodelle adressiert werden kann.

## 5.1 Behandlung fehlender Angaben

In den frühen Projektphasen interessieren oft bestimmte Details noch nicht und werden daher noch nicht modelliert, z.B. die Multiplizität einer Rolle eines Beziehungstyps. Herkömmliche Vorgehensweisen führen bei solchen fehlenden Angaben zu einer 100%igen Ähnlich- oder Unähnlichkeit der betroffenen Modellelemente; beides ist nicht angemessen.

Im Prinzip müssen solche nicht erfaßten Merkmale als Nullwerte behandelt werden, d.h. solange nichts explizit definiert wurde, gilt die Eigenschaft als undefiniert. Editoren für Teilmodelle müssen daher so konfigurierbar sein, daß

Nullwerte erfaßt und in den Modellrepräsentationen erkennbar dargestellt werden können.

Zur Behandlung von nicht erfaßten Merkmalen bzw. Nullwerten schlagen wir die folgende Strategie vor, die in einer Variante des SiDiff-Systems [12] implementiert und getestet wurde. Angenommen, zwei Modellelemente werden verglichen und für ein Merkmal $M$ tritt ein Nullwert auf:

- Wenn bei *beiden* Modellelementen ein Nullwert vorliegt, hat $M$ keinen Einfluß auf die Ähnlichkeit dieser beiden Elemente. Die Gewichtung von $M$ wird individuell für dieses Elementpaar auf Null reduziert[4].
- Wenn nur bei *einem* Modellelement ein Nullwert vorliegt, werden die beiden Modellelemente bzgl. $M$ als völlig unähnlich behandelt.

Sei $G$ das Gesamtgewicht aller Merkmale, deren Gewicht bei den gegebenen Modellelementen auf Null reduziert wurde. Sei ferner $S_{min}$ die gewichtete Summe der Ähnlichkeiten der restlichen Merkmale. Dann ist

$$S_{min} \leq 1 - G$$

Sofern alle in $S_{min}$ berücksichtigten Merkmale identisch sind, gilt $S_{min} = 1 - G$. Wenn $S_{min}$ zur Sortierung der Präferenzlisten benutzt wird, führt dies mit hoher Wahrscheinlichkeit zu unpassend sortierten Präferenzlisten. Wenn $G$ groß ist, kann sogar die Mindestähnlichkeit unterschritten werden. Daher muß $S_{min}$ geeignet korrigiert werden. Im Endeffekt sind derartige Korrekturen stets Schätzungen, wie ähnlich die fehlenden Merkmale sind.

- Im günstigsten Fall sind alle undefinierten Merkmale 100% ähnlich, die maximale Ähnlichkeit ist daher $S_{max} = S_{min} + G$. Die korrigierte Ähnlichkeit muß also im Intervall $[S_{min}, S_{max}]$ liegen.
- Sofern man für die undefinierten Merkmale die gleiche Ähnlichkeit wie für die definierten annimmt, ist $S_{korr} = S_{min}/(1 - G)$. Dies ist der gewichtete Durchschnitt der in $S_{min}$ berücksichtigten Ähnlichkeiten.

Wenn z.B. für zwei Elemente $G = 0.2$ und $S_{min} = 0.6$ ist, dann ist $S_{max} = 0.8$ und $S_{korr} = 0.75$.

Als Sortierkriterium für die Präferenzlisten schlagen wir die obige Schätzung $S_{korr}$ vor. Das Intervall $[S_{min}, S_{max}]$ ist der Unsicherheitsbereich für diesen geschätzten Ähnlichkeitswert. Bei der manuellen Korrektur von Korrespondenzen sollte dieser Unsicherheitsbereich bekannt sein. Daher wird in dem in Bild 2 gezeigten GUI der Unsicherheitsbereich, sofern vorhanden, in den Spalten MIN und MAX angezeigt.

Die Unsicherheitsbereiche der Ähnlichkeiten können Korrespondenzen (noch) unsicherer machen, hierauf gehen wir später ein.

[4] Normalerweise ist die Gewichtung der ähnlichkeitsrelevanten Merkmale für alle Instanzen eines Modellelementtyps einheitlich.

## 5.2 Behandlung expliziter Ungenauigkeitsangaben

Die schon in Abschnitt 2.3 erwähnten Sprache SeeMe enthält einen umfangreichen Katalog an Konstrukten, mit denen das Ausmaß der Ungenauigkeit einzelner Modellelemente und weitere, hier nicht relevante Aspekte notiert werden können. Für den Modellvergleich fassen wir die Ungenauigkeit eines Modellelements $e$ als eine Schätzung der Ähnlichkeit zwischen der aktuell vorhandenen Version von $e$ und der zur Zeit noch unbekannten Endversion von $e$ auf und bezeichnen diese Schätzgröße mit $U(e)$.

Offensichtlich sind Ähnlichkeiten, in denen eines der beteiligten Modellelemente ungenau ist, unsicher. Angenommen, zwei Modellelemente $e1$ und $e2$ haben die grundlegende Ähnlichkeit $S(e1, e2)$ und die Ungenauigkeit von $e1$ wird mit $U(e1)$ geschätzt. Wenn man nun die Gültigkeit der Dreiecksungleichung wie in einem metrischen Raum unterstellt[5], dann ist der Unsicherheitsbereich für die Ähnlichkeit

$$[S(e1, e2) - U(e1),\ S(e1, e2) + U(e1)]\ \cap\ [0, 1]$$

Sofern auch für $e2$ eine Ungenauigkeit $U(e2)$ angegeben ist, vergrößert sich der Unsicherheitsbereich an beiden Enden um $U(e2)$ unter Beachtung der Grenzen [0,1].

Offen blieb bisher die Frage, welche konkreten Werte für $U(e)$ sinnvoll sind. In [4] wird anhand eines größeren Experiments gezeigt, daß Werte zwischen 0.05 und 0.15 sinnvoll sind, um intuitive Begriffe wie "etwas ungenau" oder "ziemlich ungenau" im hier vorliegenden Kontext zu quantifizieren. Größere Werte führen zu großen, stark überlappenden Unsicherheitsbereichen und zu einer zu starken Aufblähung der Präferenzlisten.

## 5.3 Einflüsse auf Präferenzlisten

Wir haben oben zwei Arten der Entstehung von Unsicherheitsbereichen von Ähnlichkeiten kennengelernt: implizit durch Nullwerte und explizit durch Ungenauigkeitsangaben. In diesem Abschnitt diskutieren wir die Konsequenzen für Präferenzlisten.

Sofern bei einer Ähnlichkeit beide Ursachen für Unsicherheit auftreten, ist es in diesem Kontext am sinnvollsten, die Unsicherheitsbereiche im Sinne von Intervallen zu vereinigen.

**Sortierung der Präferenzlisten.** Die Unsicherheitsbereiche der Ähnlichkeiten können die Reihenfolge innerhalb einer Präferenzliste nicht beeinflussen, weil keine nutzbaren statistischen Informationen darüber vorliegen, wie sich die möglichen Endversionen der Modellelemente von den aktuell vorhandenen unterscheiden. Allerdings kann die Reihung in Einzelfällen unsicher werden. Betrachten wir als Beispiel eine Korrespondenz zwischen den Elementen $e1$ und $e2$. Angenommen, $e2$ stand in der Präferenzliste von $e1$ auf Platz 3. Ferner habe:

[5] Die Dreiecksungleichung gilt für Ähnlichkeiten nur unter bestimmten Voraussetzungen, ist aber als Schätzung des Unsicherheitsbereichs durchaus brauchbar.

– Platz 1 die Ähnlichkeit 0.71 und Unsicherheitsbereich [0.51, 0.76]
– Platz 2 die Ähnlichkeit 0.58 und Unsicherheitsbereich [0.45, 0.65].

Die Anordnung der Plätze 3 und 4 ist unsicher, weil ihre Unsicherheitsbereiche überlappen.

Eine Korrespondenz, bei der wenigstens ein Element auf einem unsicher sortierten Platz steht, ist unsicher. Diese Form der Unsicherheit kommt zu den anderen in Abschnitt 4.1 aufgelisteten Indizien hinzu. Um eine Informationsüberflutung zu vermeiden, sollte ein binäres Gesamturteil über die Unsicherheit einer Korrespondenz gebildet und durch ein Symbol im Elementbaum angezeigt werden. Zusätzliche Informationen über die Ursachen der Unsicherheit können in Form von Warnungen ausgegeben werden (s. Bild 1).

**Mindestähnlichkeit.** Von Unsicherheitsbereichen betroffen ist ferner die Mindestähnlichkeit: wenn eine Ähnlichkeit $S(e1, e2)$ unter der Mindestähnlichkeit liegt – also normalerweise $e1$ und $e2$ keine Korrespondenz mehr bilden können –, kann trotzdem die *Obergrenze* des Unsicherheitsbereichs deutlich darüber liegen, d.h. unter günstigen Umständen können die beiden Elemente doch noch korrespondieren.

Die offensichtliche Lösung besteht darin, zusätzlich diejenigen Elemente in die Präferenzlisten aufzunehmen, deren Unsicherheitsbereich die Mindestähnlichkeit beinhaltet. In einer umfangreichen Fallstudie [4] wurden die Auswirkungen dieser Lösung anhand mehrerer Teilmodelle (Klassendiagramme) evaluiert; hierbei wurden Standardeinstellungen für den Vergleich von (Entwurfs-) Klassendiagrammen benutzt. Es konnten durch die Erweiterung der Präferenzlisten in der Tat weitere Korrespondenzen gewonnen werden, allerdings nur in wenigen Fällen.

Praktisch der gleiche Nutzeffekt konnte indessen auch erreicht werden, indem die Merkmalsgewichtungen gegenüber den Standardeinstellungen modifiziert wurden. Konkret wurden die hohen Gewichtungen von Namen um ca. ein Viertel reduziert und die Gewichtungen anderer Merkmale, insb. der "Umgebung" der Elemente, entsprechend erhöht. Im Endeffekt war die hierdurch implementierte Heuristik besser an die Editiervorgänge der frühen Phasen adaptiert; nach dieser Optimierung der Merkmalskonfiguration konnten durch die erweiterten Präferenzlisten keine zusätzlichen korrekten Korrespondenzen mehr gewonnen werden.

Letztlich konnten in der Fallstudie [4] keine überzeugenden Argumente gefunden werden, die Unsicherheitsbereiche für die Ausweitung der Präferenzlisten zu nutzen. Als wesentlich wichtiger erwies sich die Anpassung der Ähnlichkeitsdefinition an die Verhältnisse in den frühen Phasen.

# 6 Resümee

Der Abgleich von Teilmodellen in den frühen Entwicklungsphasen unterliegt deutlich anderen Randbedingungen als das Mischen von Modellen in den späte-

ren Phasen. Hieraus ergeben sich neue Anforderungen an Vergleichs- und Mischwerkzeuge. Aus mehreren Gründen ist es wichtig, vom System berechnete ("vorgeschlagene") Matchings korrigieren zu können. Durch erweiterte Matching-Algorithmen können relevante Informationen gewonnen werden, welche Korrespondenzen unsicher sind und welche alternativen Korrespondenzen in die engere Wahl kommen. Die These, durch explizite Angaben zur Ungenauigkeit von Modellelementen ansonsten "übersehene" Korrespondenzen finden zu können, wurde in einer Fallstudie evaluiert. Der erzielte Nutzen war beschränkt, und als sinnvollere Maßnahme erwies sich, die Ähnlichkeitskriterien anzupassen, d.h. wenn der gleiche Modelltyp auch in den späteren Phasen verwendet wird, können die Ähnlichkeitskriterien nicht unverändert übernommen werden.

**Danksagung.** Die Motivation zu diesem Papier entstand im Rahmen einer Kooperation mit dem Lehrstuhl Informations- und Technikmanagement (Thomas Herrmann), U. Bochum, in der das Differenzframework SiDiff [12] an SeeMe [5] angepaßt wurde. Beteiligt an diesem Vorhaben war ferner Michael Goedicke, U. Duisburg-Essen. Allen Beteiligten sei für ihre Unterstützung gedankt.

# Literatur

[1] Cheng, B.; Atlee, J.: Research Directions in Requirements Engineering; p.285-303 in; Proc. Future of Software Engineering; ACM; 2007; ISBN 0-7695-2829-5;

[2] Förtsch, S.; Westfechtel, B.: Differencing and Merging of Software Diagrams - State of the Art and Challenges; p.90-99 in: Proc. 2nd Intl. Conf. Software and Data Technologies (ICSOFT 2007), Barcelona; INSTICC Press; 2007

[3] Goedicke, M.; Herrmann, Th.: A Case for ViewPoints and Documents; p.62-84 in: Proc. 14th Monterey Workshop; LNiCS 5320, Springer; 2008

[4] Gorek, G.: Untersuchungen zum Abgleich vager Modelle in der Systemanalyse; Diplomarbeit, Fachgruppe Praktische Informatik, Universität Siegen; 2010; http://pi.informatik.uni-siegen.de/CVSM/Go2010DA.html

[5] Herrmann, Th.; Loser, K.: Vagueness in models of socio-technical systems; Behaviour and Information Technology 18:5, p.313-323; 1999

[6] IEEE Std. 1471-2000 Recommended Practice for Architectural Description of Software-intensive Systems; IEEE; 2000 (entspricht ISO/IEC 42010:2007)

[7] Kelter, U.; Schmidt, M.; Wenzel, S.: Architekturen von Differenzwerkzeugen für Modelle; p.155-168 in: Proc. Software Engineering 2008; LNI 121, GI e.V.; 2008

[8] Kelter, U.; Wehren, J.; Niere, J.: A Generic Difference Algorithm for UML Models; p.105-116 in: Proc. Software Engineering 2005; LNI 64, GI; 2005

[9] Kolovos, D. S.; Ruscio, D.; et al.: Different Models for Model Matching: An Analysis Of Approaches To Support Model Differencing; p.1-6 in: Proc. 2009 ICSE Workshop on Comparison and Versioning of Software Models; IEEE; 2009

[10] Ohst, D.; Welle, M.; Kelter, U.: Differences between Versions of UML Diagrams; p.227-236 in: Proc. ESEC/FSE 2003, Helsinki; ACM; 2003

[11] Perrouin, G.; Brottier, E.; Baudry, B.; Le, Y.: Traon: Composing Models for Detecting Inconsistencies: A Requirements Engineering Perspective; in: Proc. REFSQ2009; 2009

[12] SiDiff Differenzwerkzeuge; http://www.sidiff.org; 2010