

Vom Suchen und Finden der Komponente* **Organisation und Strukturierung von** **Modellkomponenten**

Diss Seminar DD

* ohne Mimi, ohne Venus

- Titel im Halle-Seminar: Verbesserung modellgetriebener Softwareentwicklung durch die semantische Integration verschiedener Sichten auf ein System
- Kritik an Konzepten der UML (Semantik, Konsistenz, Abstraktion, etc.)
- Vorstellung der Vision (Diagramme als Sichten)
- (sehr naiver) Zeitplan

- Alternative zur UML ist theoretisch interessant aber praktisch nicht durchsetzbar
- Fragestellung: Wie kann die Arbeit mit der UML / Modellierung verbessert werden?
 - Sammeln vorhandener Modelle (Repository)
 - Organisation vorhandener Modelle
 - Retrieval vorhandener Modelle

- Problembeschreibung
- Forschungsziele und zentrale Fragen
- Vorgehensweise
- Strategien zur Evaluierung
- Veröffentlichungsplanung
- Executive Summary

Zitate aus einem bahnbrechenden Paper

Software components (routines), to be widely applicable to different machines and users, should be available in families arranged according to precision, robustness, generality and time-space performance.

[...]

The ultimate consumer of systems based on components ought to see considerably improved reliability and performance, as it would become possible to expend proportionally more effort on critical parts of systems, and also to avoid the now prevalent failings of the more mundane parts of systems, which have been specified by experts, and have then been written by hacks.

[Guess Who?]



- Mass – Produced Software Components, Douglas McIlroy 1968
- Prägte den Begriff Softwarekomponenten

Bild: <http://www.cs.dartmouth.edu/~doug/>

Biographie: <http://www.cs.dartmouth.edu/~doug/biography>

- Kontext: Entwicklung von Software aus Komponenten, Wiederverwendbarkeit von Komponenten
- Wiederverwendung muss geplant werden, verursacht Aufwände
- Komponenten sind en masse vorhanden, wir müssen sie nur nutzen
- Funktioniert auf Quellcodeebene mittelmäßig gut
 - Frameworks, Bibliotheken

- Zentrales Problem: Komponenten finden, die zu meiner Software passen
 - Komponenten müssen strukturiert und dokumentiert sein, um gefunden zu werden
 - Aufwand, um Komponenten zu finden, zu verstehen und zu integrieren muss geringer sein, als sie neu zu entwickeln

Problembeschreibung

- Finden von Komponenten
 - Stichwortsuche
 - Search Driven Development (koders.com)
 - Test Driven Search Driven Development [Lazzarini2007]
 - Formale Komponentenbeschreibungen
- Effizientes Retrieval setzt Organisation der Komponenten voraus

Forschungsziel / abgeleitete Fragen

- Ziel: vorhandene Modelle sollen so organisiert werden, dass sie bei der Entwicklung neuer Software benutzt werden können
- Wie lassen sich Modelle sinnvoll organisieren?
 - Alphabetisch, Chronologisch, Thematisch, ...
 - Modell-Browsing
- Wie wird nach Modellen gesucht?
 - Deskriptiv, Testfall, Stichwörter, ...
 - Modell-Retrieval

- Extraktion sinnvoller Modelleigenschaften
- Analyse existierender Klassifizierungsverfahren
- Anwendung der Verfahren
- Evaluierung der Verfahren
- Integration in den Entwicklungsprozess

- Klassifizierung: Objekte anhand bestimmter Eigenschaften in Gruppen einteilen
- Welche Eigenschaften gibt es bei Modellen?
- Wie lassen die sich automatisiert auslesen?

- Merkmal Text: ähnliche Modelle haben ähnlichen Text
- Vorteil: sehr einfach zu ermitteln
- Nachteil: versagt schon bei unterschiedlichen Sprachen, nicht aussagekräftigen Bezeichner
- Vermutung: Text gruppiert unterschiedliche Modelltypen (UML-Diagrammtypen, EPKs, etc.)
 - Unterschiede in XMI-Repräsentationen der Diagrammtypen
 - Strukturierung unstrukturierter Modellsammlungen

- Gruppierung unterschiedlicher Modelltypen
 - Tags in Ecore-Klassendiagrammen: `uml`, `elementImport`, `importedElement`, `packagedElement`, `ownedAttribute`, `ownedOperation`
 - Tags in Ecore-Aktivitätsdiagrammen: `uml`, `elementImport`, `importedElement`, `packagedElement`, `node`, `edge`

→ Gruppierung erfolgt anhand disjunkter Tags

- Struktur: formale Syntax der Modellierungssprachen als Grundlage
 - Ansatz: Action Patterns aus EPKs [Smirnov2009], aus ähnlicher Struktur folgt ähnliches Verhalten
 - Metamodell zur Ermittlung von Beziehungen
- Verhalten: (semi)formale Semantik als Grundlage
 - Work in Progress

- Klassifizierungsverfahren muss sich auf Modelle anwenden lassen
 - Distanzverfahren, Bayes, Clustering, fuzzy etc.
- Anwendung auf Text sollte noch simpel sein, bei Struktur / Verhalten müssen Verfahren angepasst werden
- Festlegung des Formats der Modelle zur Verarbeitung

- Anwendung auf verschiedene Modellsammlungen
 - LDraw Standard für Lego CAD Programme
 - Google 3D Modelle für Google Earth
 - Open Model Initiative
 - SAP Referenzmodell
- Hoffnung, dass zumindest die verschiedenen Typen korrekt gruppiert werden
- Anwendung in anderen Bereiche als SE möglich?
 - Gruppierung von 3D-Modelltypen

- Suche nach Komponenten als zentrale Aufgabe im SDP
 - Nahtlose Integration notwendig
 - Werkzeugunterstützung
- Framework, welches die Verwaltung der Komponenten übernimmt
 - Unterstützung durch Modell-Repository
 - Suche nach Komponenten, Einfügen von Komponenten
 - Auto-Complete

- Evaluierung mittels Vergleich automatischer mit manueller Gruppierung
 - Einteilung der verschiedenen Notationsarten ist manuell objektiv
 - Einteilung der Modelle nach Verhalten ist idR. subjektiv
 - SAP Referenzmodell besteht aus vielen Prozessen, die manuell eingeteilt wurden
- Berechnung der Erfolgsquote
 - Anteil fehlerhafter Gruppierungen

- Metriken aus dem Information Retrieval
 - Recall / Precision
 - Hoher Recall ist nicht immer sinnvoll, z.B. wenn extrem viele Modelle zurückgegeben werden
 - Metriken, die auf Klassifizierung / Clustering angepasst sind
 - Güte und Stabilität

- 16.04. Workshop: From Code to Model Centric
 - Modelle clustern statt nach Code zu suchen
- 14.05. ASE Doctoral Symposium
 - Research Roadmap
- **21.05. Workshop on Reuse in BPM**
 - Anwendung von Clustering auf BPMs

- Problembeschreibung: viele Modelle existieren bereits, sind aber nicht organisiert und schwer zu finden, Entwicklung aus Komponenten wird unnötig erschwert
- Ziel: Methode, mit der vorhandene Modelle automatisch organisiert werden können, damit sie bei der Entwicklung auffindbar sind
- Grundlage: ähnliche Modelle haben ähnliche Wörter / Struktur / Verhalten
- Ansatz: Klassifizierung von Modellen mit Methoden der automatischen Sprachverarbeitung
- Evaluierung: Vergleich automatisch erzeugter mit manuell gebildeten Clustern

- [Smirnov2009] Smirnov, S.; Weidlich, M.; Mendling, J. & Weske, M. Action Patterns in Business Process Models
- [Lazzarini2007] CodeGenie: a tool for test-driven source code search