Das Prozeßmodellierungswerkzeug MoST

Entwicklung formaler Prozeßmodelle

Martin Verlage und Ulrike Becker

Fraunhofer-Einrichtung für
Experimentelles Software Engineering IESE

17. März 1997

Gliederung

- Prozesse und Prozeßmodelle
- Formale Prozeßmodellierungssprachen und ihre Benutzung
- Das Werkzeug MoST
- Erfahrungen im Umgang mit MoST
- Zusammenfassung und Ausblick

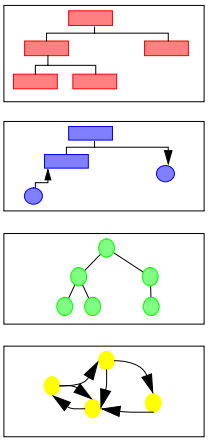
Prozesse und Prozeßmodelle (1 von 3)

Die Durchführung und das Management von Software-Entwicklungsprozessen stellen komplexe Aufgaben dar. Die Prozeßmodellierung unterstützt die Durchführung und das Management der Prozesse. Dies geschieht folgendermaßen:

- Die Prozesse werden sichtbar
- Die Prozesse werden strukturiert
- Schlecht verstandene Prozeßschritte werden identifiziert
- Probleme werden identifiziert
- Wichtiges wird hervorgehoben
- Die Arbeitsweise der Prozesse wird erläutert

Prozesse und Prozeßmodelle (2 von 3)

Prozeß-ingenieur



Was wird getan?

Wer ist beteiligt?

Welche Organisation erledigt es?

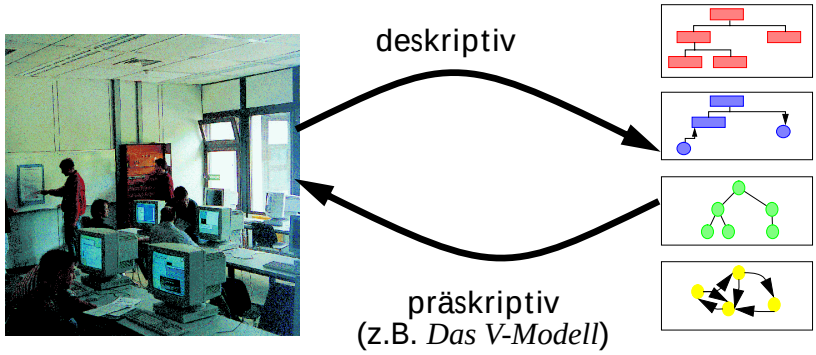
Was ist das Resultat?

Wann wird es erledigt?

... ?

Prozesse und Prozeßmodelle (3 von 3)

Prozesse müssen kontinuierlich verbessert werden. Bei der Prozeßmodellierung als wesentlicher Infrastrukturtechnologie wechseln deshalb Phasen der deskriptiven und präskriptiven Modellierung ab.



Prozeßmodellierungssprachen

Es existieren eine Reihe formaler Prozeßmodellierungssprachen (u.a. Workflow, prozeß-sensitive SEUen). Es gibt jedoch wenige Sprachen, die den Entwurf von Prozeßmodellen unterstützen. Eine dieser Sprachen ist MVP-L (multi-view process modeling language).

MVP-L erlaubt die Beschreibung von Beziehungen zwischen Prozessen, Produkten und Ressourcen. Verfeinerungshierarchien und Produktflüsse sind dabei die wesentlichen Beziehungen. Attribute erlauben die Integration von Meßplänen.

MVP-L ist eine regelbasierte Sprache. Die Bedingungen beziehen sich dabei auf Produktzustände und Prozeßattribute.

Was wird erzeugt?

Was wird geändert?

Wann darf ich starten?

Was ist mein Ziel?

```
process_model E0_Cleanroom() is
  product_flow
    consume
    produce
      T2_SSM: T2_Software_Specification_Files;
      T4_SCF: T4_Software_Certification_Files;
      T7_PRS: T7_Pre_Release_Software;
      T8_FRE: T8_Failure_Rep_ECN;
    consume_produce
      T1_PDF: T1_Project_Document_Files;
      T3_SDF: T3_Software_Development_Files;
      T5_PMF: T5_Project_Management_Files;
      T6_QC: T6_Questions_Container;
  entry_exit_criteria
    local_entry_criteria
      (T1_PDF.status = 'complete' and T2_SSM.status = 'nonexistent'
        and T4_SCF.status = 'nonexistent' and T5_PMF.status = 'complete');
    local_exit_criteria
      (T1_PDF.status = 'complete' and T2_SSM.status = 'complete'
          and T3_SDF.status = 'complete' and T4_SCF.status = 'complete'
          and T5_PMF.status = 'complete' and T7_PRS.status = 'complete'
          and T8_FRE.status = 'complete');
  end process_interface
```


Welche Teilprozesse gibt es?

Welche Aggregtaionsform?

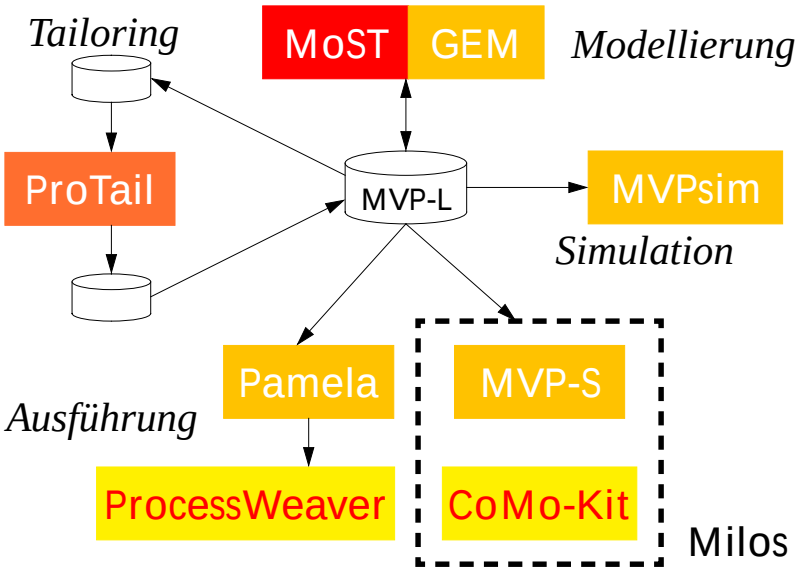
Wie werden die Schnittstellen untereinander verbunden?

```
refinement
  objects
    e4n: E4_Software_Solution_Specification_NULL;
    e4: E4_Software_Solution_Specification;
    e5n: E5_Software_Development_Certification_NULL;
    e5: E5_Software_Development_Certification;
  object_relations
    (e4n &  e4 & e5n & e5);
  interface_refinement
    interface_relations
      e4n(T1_PDF => T1_PDF, T2_SSM => T2_SSM, T3_SDF => T3_SDF,
          T4_SCF => T4_SCF, T5_PMF => T5_PMF);
      e4(T1_PDF => T1_PDF, T2_SSM => T2_SSM, T3_SDF => T3_SDF,
          T4_SCF => T4_SCF, T5_PMF => T5_PMF);
      e5n(T1_PDF => T1_PDF, T2_SSM => T2_SSM, T3_SDF => T3_SDF,
          T4_SCF => T4_SCF, T5_PMF => T5_PMF, T6_QC => T6_QC,
          T7_PRS => T7_PRS, T8_FRE => T8_FRE);
      e5(T1_PDF => T1_PDF, T2_SSM => T2_SSM, T3_SDF => T3_SDF,
          T4_SCF => T4_SCF, T5_PMF => T5_PMF, T6_QC => T6_QC,
          T7_PRS => T7_PRS, T8_FRE => T8_FRE);
end process_body
```

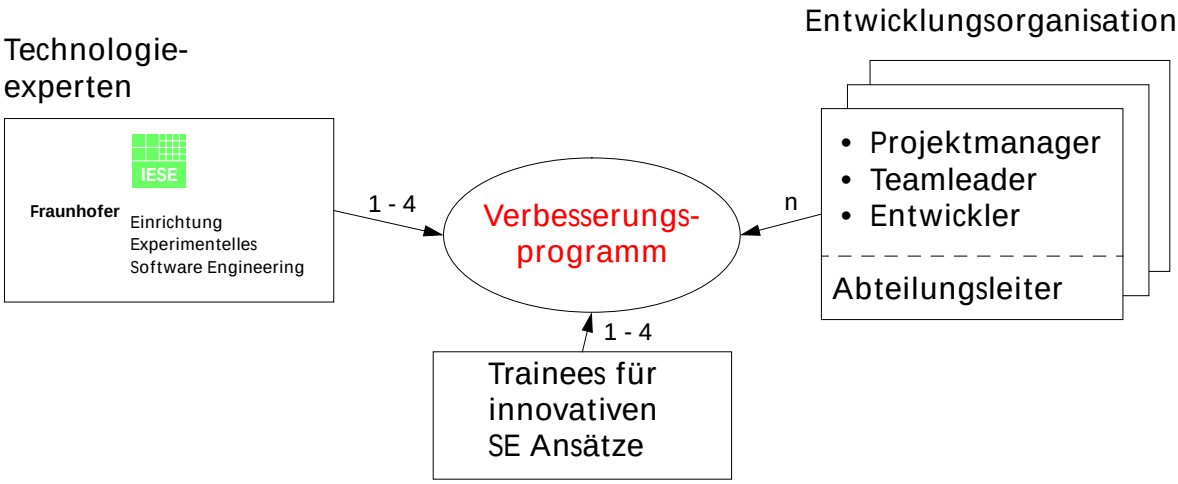
Prozeßmodellierungssprachen

Prozeßmodellierungs-sprachen erlauben die Entwicklung und Integration von Werkzeugfamilien zur Unterstützung des “process engineering”

Die Formalität soll jedoch nicht zentraler Aspekt der Modellierung sein.



Benutzung von Prozeßmodellierungssprachen



Das Werkzeug MoST

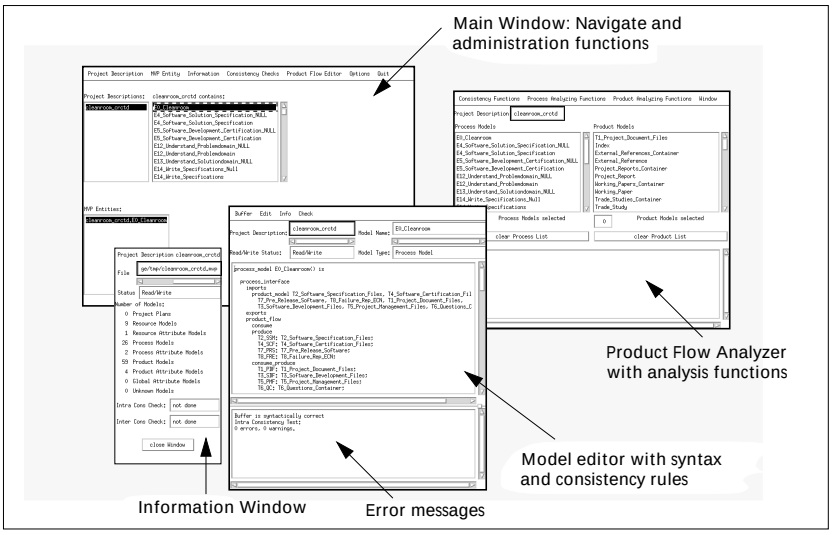
MoST unterstützt den Prozeßingenieur bei der Entwicklung von Prozeßmodellen. Das Werkzeug wurde bereits bei einer Reihe von Industrieprojekten eingesetzt. Die Erfahrungen aus diesen Projekten haben zu entscheidenden Verbesserungen des Werkzeugs geführt.

MoST verwaltet textuelle Darstellungen der Prozeßmodelle

MoST bietet drei Arten von Diensten an:

- Verwaltung von Prozeßmodellen
- Konsistenzprüfung der Prozeßmodelle
- strukturelle Analyse der Prozeßmodelle

Die Benutzerschnittstelle von MoST



MoST: Verwaltung von Prozeßmodellen

Prozeßmodelle sind in der Regel komplex. Die Verwaltung der Prozeßmodelle muß dieser Tatsache gerecht werden.

- Laden/Sichern von Prozeßmodellen
- HTML-Output von Prozeßmodellen
- Parallele Bearbeitung mehrerer Prozeßmodelle
- Spezialwerkzeuge
  (Modelleditor und Produktflußeditor)

Der modulare Aufbau der Prozeßmodelle erlaubt die Anwendung von Funktionen auf Teile eines Vorgehensmodells oder auf seine Gesamtheit.

MoST: Die Analyse von Prozeßmodellen

Im wesentlichen wird der Produktfluß zwischen Prozessen und die Verfeinerung analysiert. Die einzelnen Analysen entsprechen Fragen, die ein erfahrener Prozeßingenieur bei der Modellierung stellen würde. Die Analysen sollen Problembereiche in den Prozeßmodellen identifizieren.

Beispiele für Analysen:

- Greifen atomare Prozesse auf komplexe Produkte zu?
  ⇒ Unterspezifikation der Prozesse
- Werden Produkte nicht benutzt?
- Existiert ein Produktfluß zwischen zwei Prozessen?
- Welche Prozesse erzeugen kein Produkt?
  ⇒ Anzeigen der “End”-Prozesse und evtl. Fehler

MoST: Die Konsistenzprüfung der Prozeßmodelle

Syntaktische Prüfung

Konsistenzprüfung
(15 intra-Modell-Regeln und 17 inter-Modell-Regeln)

Beispiele für Intra-Modell-Regeln:

- Typprüfungen bei Ausdrücken
- Benutzung von Objekten in Ausdrücken

Beispiele für Inter-Modell-Regeln:

- Balanciertheit des Produktflusses
- Konsistenz der Schnittstellen bei Verfeinerungen

Erfahrungen im Umgang mit MoST (1 von 2)

Deskriptive Modelle

- mehrere Industrieprojekte (u.a. Robert Bosch GmbH, Bosch Telekom, Allianz Lebensversicherungs AG)

Präskriptive Modelle

- Reuse Process TRW
- IEEE Standard 1074 - 1991
- IBM STARS Cleanroom Software Development Processes
- Das V-Modell Version 1
- ÄA733 und V-Modell Rohling V8
- Developing Real-Time Systems [Bræk und Haugen]

Erfahrungen im Umgang mit MoST (2 von 2)

MoST ist für den Prozeßingenieur geeignet.

Grafische Darstellungsmöglichkeiten fehlen (⇒ GEM).

Analysefunktionen und Konsistenzregeln helfen, Schwächen in den Prozeßmodellen und in den Prozessen zu entdecken.

Prozeßvarianten und Modifikationen stellen noch immer große Probleme dar.

Generelle Empfehlungen

- Orientiere dich bei der Modellierung an den Konzepten der Prozeßmodellierungssprache
- Gehe “bottom-up” vor

Zusammenfassung

Kontinuierliche Verbesserung von Software-Entwicklungsprozessen benötigt eine deskriptive und präskriptive Verwendung von Prozeßmodellen.

Formale Prozeßmodellierungssprachen erleichtern die Arbeit des Prozeßingenieurs erheblich. Die Formalität soll jedoch nicht überbetont werden.

Das Werkzeug MoST bietet eine auf empirischen Untersuchungen aufbauende Hilfe an.

Die Verwaltung, Analyse und Konsistenzprüfung von MVP-L-Prozeßmodellen wird durch MoST realisiert.

MoST wurde in verschiedenen Kontexten erfolgreich eingesetzt. Es konnten Erfahrungen für die Verbesserung des Werkzeugs gesammelt werden.

Ausblick

Die Verbesserung des Werkzeugs unter Berücksichtigung der bisherigen Erfahrungen wird in den nächsten Monaten vorangetrieben. Insbesondere muß eine stärkere Integration des grafischen Werkzeugs GEM erfolgen.

In Zusammenarbeit mit dem CAESAR (Centre for Advanced Empirical Software Research) der University of New South Wales entsteht das Werkzeug SPEARMINT (Software Process Elicitation, Analysis, Review and Model Integration Tool) zur verbesserten Prozeßmodellierung durch die Realisierung verschiedener Sichten.

In Zusammenarbeit mit dem SEI (Software Engineering Institute) in Pittsburgh entsteht ein Electronic Process Guide auf Web-Basis zur besseren Anleitung von Entwicklern in einem Projekt.

Das IESE berät den SFB 501 (Entwicklung großer Systeme mit generischen Methoden) der DFG im FB Informatik der Universität Kaiserslautern bei der Realisierung einer prozeß-sensitiven SEU (Milos) zur Unterstützung kreativer Prozesse.

Das Prozeßmodellierungswerkzeug MoST

Ausblick

Potentielle Kooperationen mit der Fraunhofer-Einrichtung IESE im Bereich der Prozeßmodellierung können sein:

- Entwicklung kommerzieller Prozeßmodellierungswerkzeuge
- Überprüfung oder Entwicklung hauseigener Standards und Richtlinien
- Definition innovativer Prozesse der Software-Entwicklung
- Forschung im Bereich der Prozeßmodellierung