Interner Bericht

Qualitätsorientierte, prozeß-sensitive Softwareentwicklungsumgebungen im MVP-Projekt

H. Dieter Rombach, Andreas Birk, Alfred Bröckers,
Christopher M. Lott und Martin Verlage

256/94

Fachbereich Informatik

Universität Kaiserslautern • Postfach 3049 • D-67653 Kaiserslautern

Qualitätsorientierte, prozeß-sensitive Softwareentwicklungsumgebungen im MVP-Projekt

H. Dieter Rombach, Andreas Birk, Alfred Bröckers,
Christopher M. Lott und Martin Verlage

256/94

Herausgeber: AG Software Engineering
Leiter: Prof. Dr. H. Dieter Rombach

Kaiserslautern, Oktober 1994

Kurzfassung

Software-Projekte bestehen aus einer Vielzahl von Teilaufgaben, die durch komplexe Wechselbeziehungen miteinander verknüpft sind. Systematische Unterstützung bei der Durchführung von Software-Projekten erfordert deshalb nicht nur die isolierte Unterstützung einzelner Teilaufgaben, sondern insbesondere der Wechselbeziehungen. Außerdem müssen Aktivitäten des Messens und Bewertens durchgeführt werden, um quantitative Aussagen über Produkte und Prozesse ableiten zu können. Ziel des MVP-Projekts (Multi-View Process modeling) ist es, derartige integrierte Unterstützung auf der Basis meßbarer Projektpläne zur Verfügung zu stellen. Projektpläne setzen sich dabei unter anderem aus Prozeß-, Produkt-, Ressourcen- und Qualitätsmodellen zusammen. Meßansätze werden nicht nur zur systematischen Unterstützung von Projekten, sondern auch zur Verbesserung existierender Prozeß-, Produkt-, Ressource- und Qualitätsmodelle aufgrund 'gemessener' Erfahrungswerte verwendet. Die Benutzer des MVP-Entwicklungssystems (MVP-S) werden durch ihre Rollen im Rahmen eines Projekts charakterisiert werden können. Es wird beschrieben, wie Rollen das MVP-System nutzen können. Dies geschieht entweder durch direkte Repräsentation ihrer Aufgaben als Prozesse oder indem die im Projektplan repräsentierte Information ausgewertet und präsentiert wird; entsprechend bezeichnen wir eine Rolle als "zustandsverändernd" oder als "zustandserfragend". Um diese Rollen zu unterstützen, existieren unterschiedliche Möglichkeiten abhängig vom Grad der Automatisierung. Es werden beispielhaft drei Stufen aufgezeigt. Anschließend wird die Realisierung einer prototypischen, qualitätsorientierten, prozeßsensitiven Software-Entwicklungsumgebung diskutiert. Zum Abschluß wird auf gegenwärtige und zukünftige Forschungsfragen im Rahmen des MVP-Projekts eingegangen.

Schlüsselwörter: Qualitätssicherung, Software-Prozeß-Modellierung, Rolle, Software-Entwicklungsumgebung, MVP-Projekt, Quality Improvement Paradigm, Goal/Question/Metric-Paradigm

Dieser Bericht ist eine Überarbeitung und Ergänzung des Papiers "Entwicklungsumgebungen zur Unterstützung qualitätsorientierter Projektpläne", Tagungsband der GI-Fachtagung Softwaretechnik'93, erschienen als: Softwaretechnik-Trends, Band 13, Heft 3, August 1993, ISSN 0720-8928.

Insbesondere gehen wir in diesem Bericht ausführlich auf diejenigen Fragestellungen ein, die Anlaß zur Diskussion nach der Präsentation des Papiers gaben und berücksichtigen neue Ergebnisse aktueller Forschungsarbeiten (im wesentlichen Kapitel 3 und 5).

Inhalt

Verzeichnis der Abbildungen

1 Einleitung

Man versteht ein Software-Projekt als die Summe aller Tätigkeiten mit definierten Qualitätszielen zur Entwicklung oder Betreuung von Software-Produkten. Heutige Projekte sind, zum Leidwesen von Entwicklern und Anwendern, oft durch ungenügende Qualität der Tätigkeiten (z.B. Überziehen des Budgets oder des Zeitplans) und der Produkte (z.B. Anzahl von Fehlverhalten) charakterisiert. Das heißt, diese Qualitätsziele werden nicht erreicht. EIn Grund hierfür ist die zum Teil unpräzise Definition von Qualitätszielen.

Qualitätsmodelle zur quantitativen Charakterisierung sowohl der Tätigkeiten als auch der Produkte werden eingesetzt, um die Erreichung dieser Ziele objektiv nachprüfen zu können. Dabei genügt es nicht nur zu definieren, an welchen Qualitäten man interessiert ist, sondern es müssen auch diejenigen Elemente eines Projekts modelliert werden, deren Qualitäten man definiert, nämlich die Tätigkeiten und die Produkte. Daraus ergeben sich drei wesentliche Typen von Modellen: Qualitäts-, Prozeß- und Produktmodelle. Mit Hilfe dieser Modelle werden Projektpläne beschrieben.

Diese Modellierung der Qualitätsaspekte eines Projekts kann jedoch nicht nur dazu benutzt werden, die Erreichung der Ziele zu überprüfen, vielmehr können die im Projekt eingesetzten Techniken und Methoden bewertet werden; dadurch wird Projekterfahrung objektiv und für Dritte nachvollziehbar dokumentiert. Darüberhinaus werden Ansatzpunkte für Verbesserungen deutlich gemacht. Diese Verbesserungen können auch über mehrere Projekte hinweg beobachtet werden.

Die Bedeutung von 'Messen und Bewerten' für die Praxis im Rahmen von Maßnahmen der Qualitätssicherung [23] und die Validierung von Technologie im Bereich des Software-Engineering [24] wurde in den letzten Jahren erkannt. Die immer stärkere Konzentration auf Qualitätsaspekte und die Verwendung von expliziter Qualitätsmodelle wird durch eine Vielzahl von Meß-Ansätzen und veröffentlichten Erfahrungen dokumentiert (siehe dazu auch [11]). 'Messen und Bewerten' der Produkte und Tätigkeiten eines Software-Projekts wird mit den verschiedensten Motivationen durchgeführt. Wünschenswert wäre eine Erweiterung von Software-Entwicklungsumgebungen um den Funktionsumfang der für Messen und Bewerten notwendig ist.

Aber die einzelnen Tätigkeiten eines Software-Projekts besitzen oft eine hohe Komplexität und stehen darüber hinaus noch durch eine Vielzahl von Wechselbeziehungen miteinander in Verbindung. Traditionelle Software-Entwicklungsumgebungen, die lediglich Werkzeuge integrieren, werden diesen Anforderungen nicht gerecht. Software-Entwicklungsumgebungen müssen darüberhinaus noch mehr Aspekte als nur die der Produkte eines Projekts unterstützen.

Ein neuerer Ansatz ist die explizite *Modellierung von Software-Prozessen* (engl. software process modeling). Die Relevanz einer expliziten Repräsentation in Form von Modellen und Standardisierung der in einer Umgebung existierenden Tätigkeiten wird besonders durch die vermehrte Bewertung von Unternehmen mittels sog. *Maturity*-Modelle (z.B. SEI Capability Maturity Model [14] oder BOOTSTRAP [5]) deutlich. Diese Maturity-Modelle bewerten unter anderem den Stand der Prozeßmodellierung und Qualitätssicherung in einer Software-erstellenden Organisation.

Bei der Prozeßmodellierung wird das Ergebnis einer Modellierung, der Projektplan, als Eingabe für eine prozeß-sensitive Software-Entwicklungsumgebung benutzt [12]. Dieser wird dann schritthaltend mit dem Fortschritt des Projekts in der realen Welt interpretiert. Entwicklungsumgebungen lassen sich also auf einzelne Projekte zuschneiden. Existierende Systeme sind zum Beispiel Marvel, HFSP (Hierarchical and Functional Software Processes), Arcadia oder Process Weaver [20, 21, 8]. Sie konzentrieren sich jeweils auf unterschiedliche Aspekte in einem Software-Projekt, wie etwa Beziehungen zwischen Prozessen, Funktionalität der Prozesse, Werkzeugintegration oder Koordination von Mitarbeitern [15]. Bedingt durch die Heterogenität der Software-Prozesse existieren deshalb die verschiedensten Untersuchungsschwerpunkte auf diesem Gebiet.

Das MVP-Projekt zielt auf die Vereinigung von Ansätzen zum Messen und Bewerten sowie expliziter Modellierung von Software-Prozessen. Dadurch unterscheidet es sich von anderen prozeß-sensitiven Entwicklungsumgebungen. Die explizite Repräsentation von Qualitäten, Prozessen und Produkten in Form von Modellen eröffnet viele Möglichkeiten zur Planung, Durchführung, Verbesserung und Wiederverwendung von Projekten bzw. Projekterfahrungen. Durch den MVP-Ansatz wird es also möglich, Entwicklungsumgebungen zur Unterstützung qualitätsorientierte Projektpläne bereitzustellen.

Wir wollen in dieser Arbeit besonders berücksichtigen, wie einzelne Personen ihre Sicht auf ein Projekt nutzen können. Deshalb werden wir nach einer Vorstellung des MVP-Ansatzes in Abschnitt 2 vertieft auf die Spezifikation von Interessenschwerpunkten einzelner Rollen innerhalb von Abschnitt 3 eingehen. Der vierte Teil dieser Arbeit beschäftigt sich dann mit der Frage, wie die Modelle zu unterschiedlichen Zwecken der Projektsteuerung oder Prozeßverbesserung beitragen können. Nachdem die Nutzung dieser Art von Entwicklungsumgebungen erläutert wurde, werden wir in Abschnitt 5 einen ersten Prototypen einer qualitätsorientierten, prozeß-sensitiven Software-Entwicklungsumgebung vorstellen. Abschließend werden wir in Abschnitt 6 erörtern, welche Fragen noch zu klären sind, um den in diesem Bericht vorgestellten Ansatz vollständig zu realisieren.

2 Grundlagen des MVP-Ansatzes

Das Projekt *multi-view process modeling* (MVP) wurde Ende der 80er Jahre an der University of Maryland at College Park (USA) gestartet und wird seit Anfang 1992 an der Universität Kaiserslautern fortgesetzt [4].

Das MVP-Projekt ist eine Weiterführung der Ideen, die im Rahmen des TAME-Projekts [6] (Tailoring A Measurement Environment) an der University of Maryland entstanden sind. Das wesentliche Prinzip, für welches die Techniken des MVP-Projekts entwickelt werden, ist die schrittweise Verbesserungen der in einer Umgebung eingesetzten Techniken und durchgeführten Prozesse. Diese iterative Qualitätsverbesserung erfolgt nach dem *Quality Improvement Paradigm* [3].

2.1 Qualitätsverbesserung

Die Notwendigkeit der Verbesserung eingesetzter Technologie, d.h. Methoden, Techniken und Werkzeuge, ist heutzutage unbestritten. Eine sprunghafte Verbesserung kann dabei aber nur unter sehr eingeschränkten Bedingungen und günstigen Voraussetzungen erfolgen. Systematische Verbesserung der Qualitäten von Prozessen und Produkten bedarf einer schrittweisen Steigerung der technologischen Leistungsfähigkeit. Stellvertretend für eine Reihe von verschiedenen Ansätzen zur Qualitätsverbesserung ist Demings Plan-Do-Check-Act-Methode [10]. Sie wurde zur allgemeinen Unterstützung der Verbesserung von Entwicklungsprojekten unspezifischer Art entwickelt. Das Hauptaugenmerk liegt jedoch auf der Verbesserung eines einzelnen, festen Prozeßmodells aus der industriellen Fertigung.

Ein Ansatz, der speziell auf die Bedürfnisse von Software-Entwicklungsprojekten abzielt, ist das *Quality Improvement Paradigm (QIP)*. Das QIP besteht aus den folgenden sechs Schritten, welche die Planung, Durchführung, Auswertung und das Ziehen von Konsequenzen beschreiben.

1. Charakterisiere das anstehende Projekt/die Umgebung
2. Formuliere Ziele in quantitativer Form für erfolgreiche Projektdurchführung und Know-how-Verbesserung
3. Wähle ein geeignetes Vorgehensmodell sowie unterstützende Methoden und Werkzeuge
4. Führe das Projekt nach Plan aus, erfasse und analysiere Daten zur Projektkontrolle
5. Analysiere die Daten, bewerte die Effektivität des Plans, schlage mögliche Verbesserungen für zukünftige Projekte vor
6. Verbessere existierendes Know-how für zukünftige Projekte

Der erste Schritt spiegelt die Tatsache wider, daß die Umgebung und die Charakteristika eines Projekts einen wesentlichen Einfluß auf die Prozesse haben. Hier wird z.B. festgehalten, wieviel Erfahrung die Projektmitglieder mit einzelnen Techniken haben, welches der Typ der zu entwickelnden Software ist oder welche Priorität das Projekt innerhalb der Gesamtheit aller momentanen Entwicklungen hat.

Der zweite Schritt verlangt eine Auseinandersetzung mit den Zielen des Projekts. Entsprechend der *experimentellen* Natur heutiger Software-Entwicklungen und -Betreuungen unterscheiden wir zwischen zwei Kategorien von Zielen:

1. projektspezifische Ziele: Die Ziele beschreiben den zu erreichenden Produkt- und Projektzustand. Hierbei wird angegeben, welche Qualitäten die zu entwickelnde oder zu modifizierende Software haben muß (z.B. unter dem Aspekt der Zuverlässigkeit) und in welchen Grenzen sich das Projekt selbst abspielt (z.B. Terminplaneinhaltung). Diese Ziele haben nur Gültigkeit im Projekt selbst und dienen der operationalen Steuerung eines Projekts.

2. organisatorische Ziele: Die Ziele beschreiben Untersuchungen im Einsatz mit der im Projekt verwendeten Technologie. Bei der Definition der Ziele spielen strategische Überlegungen der Organisation die entscheidende Rolle. Das Projekt selbst wird dazu genutzt, um den Wissenstand der Organisation, ein projektübergreifender Standpunkt, zu erweitern oder verbessern. Die Ausrichtung der Ziele erfolgt nach dem Zweck der Untersuchung im Projekt (Verstehen, Entwickeln, Erlernen oder Verbessern), d.h. die Ziele unterscheiden sich je nachdem, ob man eine Technologie z.B. nur Verstehen möchte oder sie verbessern will. Danach richtet sich auch der Aufbau des Experiments [7].

Der dritte Schritt des QIP verlangt eine Auswahl eines geeigneten Prozeßmodells, das die einzelnen Schritte des Projekts beschreibt. Der Detaillierungsgrad dieses Modells, aus dem der Projektplan entsteht, hängt vom Wissen der Organisation und der Stabilität der in ihr ablaufenden Prozesse ab. Je stabiler und besser verstanden die Prozesse sind, um so detaillierter kann die Planung der einzelnen Projektschritte erfolgen. Weiterhin sind in diesem Schritt des QIP die während des Projekts einzusetzenden Methoden, Techniken und Werkzeuge auszuwählen und aufeinander abzustimmen.

Im nächsten Schritt des QIP wird das Projekt entsprechend dem Plan durchgeführt. Dabei werden zu definierten Zeitpunkten oder Ereignissen Meßdaten erfaßt, die mittels eines Soll/Ist-Vergleiches erkennen lassen, inwieweit die projektspezifischen Ziele erreicht wurden. Zeitpunkte, Ereignisse und Soll-Werte sind im Projektplan vorgegeben. Sind Abweichungen vom Plan erkennbar, so ist eine Planänderung vorzunehmen; diese kann je nach Art und Größe der Abweichung eine Änderung der Ziele, des Prozeßmodells oder eine vollständige Neuplanung inklusive "konsistentem Aufsetzen" auf dem alten Projektplan beinhalten. Die Meßdaten, die während der Projektabwicklung gesammelt werden, bilden einen Teil des sog. *Projekttrace*. An dem Projekttrace läßt sich das Fortschreiten des Projekts dokumentieren.

Der Projekttrace bildet die wichtigste Grundlage im fünften Schritt des QIP. Die Abwicklung des Projekts wird durch eine post-mortem Analyse bewertet. Dabei wird vor allem darauf geachtet, inwiefern der vorgegebene Projektplan eingehalten wurde und welche Schwierigkeiten aufgetreten sind. Dies geschieht vor allen Dingen durch eine kritische Analyse der Zielerreichung bzgl. der organisatorischen Ziele. Darüberhinaus können Verbesserungen vorgeschlagen werden, die ohne direkte Motivation aus Abweichungen resultieren.

Der sechste Schritt verlangt eine Beschäftigung mit den Verbesserungsvorschlägen aus dem fünften Schritt. Aufgrund der gemachten Erfahrungen innerhalb des Projekts wird versucht das Wissen der Umgebung bzgl. der eingesetzten Technologie und der verwendeten Prozesse zu erweiteren, korrigieren und zu verbessern. Dies kann Spezialisierung/Generalisierung, Verfeinerung/Abstrahierung oder Addition/Löschen von Wissen bedeuten.

Das QIP ist als eine allgemeine Vorgehensweise für die Durchführung von Projekten und schrittweise Verbesserung von Erfahrung zu verstehen. Es ist die Beschreibung einer systematischen Arbeitsweise, die konsequent die Erfahrungen aus vergangenen Projekten nutzt [1]. Herausstechendes Merkmal ist die Unterstützung eines projektübergreifenden Lernprozesses. Dieser Erfahrungstransfer über Projektgrenzen hinweg ermöglicht eine Weitergabe von Wissen

2.2 Quantifizierung

Sowohl Benutzer und Anwender als auch Entwickler sind an bestimmten Qualitäten der Produkte und Prozesse interessiert. Beispiele hierfür bzgl. des Produkts sind Qualitäten wie Zuverlässigkeit, Korrektheit, Robustheit oder Wartbarkeit; für Prozesse könnten dies zum Beispiel Planeinhaltung, Kosten, Aufwand oder Akkuratheit der Durchführung sein. Oftmals werden diese Kriterien in einer vagen und ungenauen Form verwendet. Die Interpretation der Begriffe ist vom jeweiligen Kontext des Betrachters und dessen Vorwissen abhängig. Eine objektive Definition der Qualitäten erfolgt meist nicht. Die Notwendigkeit einer genaueren Definition wird zum Beispiel deutlich im Zusammenhang mit der Beurteilung eines Testverfahrens; beispielhafte Fragen könnten hier sein:

Wie gut (schlecht) ist das eingesetzte Testverfahren?

Wie teuer ist das eingesetzte Testverfahren?

Welchen Einfluß hat die Modulgröße auf die Tests?

Sind E-/A-Module geeignet für das Testverfahren?

Wieviele Fehlverhalten wurden in getesteter Software entdeckt?

Abb. 1: Fragen im Zusammenhang mit der Beurteilung eines Testverfahrens

Die Beantwortung dieser und ähnlicher Fragen muß nachvollziehbar sein. Dies trifft nicht nur für die Definition der Produktziele (hier die Erreichung eines bestimmten Grads der Zuverlässigkeit) zu, sondern für alle Ziele, die im zweiten Schritt des QIP aufgestellt werden [23].

Um die Erreichung von Zielen objektiv bestimmbar zu machen, bzw. um bestimmte Eigenschaften von Prozessen und Produkten zu definieren, benötigt man die Quantifizierung der Produkte und Prozesse. Erst dadurch wird ein konkretes Verständnis für die Elemente ermöglicht, welches die Basis für eine schrittweise Verbesserung der existierenden Technologien erlaubt. Die Tatsache, daß auch der Prozeß des Quantifizierens und Verstehens iterativ ist, wird durch folgende zwei Zitate verdeutlicht:

> Lord Kelvin (1824-1904): "When you can measure what you are speaking about, and express it into numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind: It may be the beginning of knowledge, but you have scarely in your thoughts advanced to the stage of science."
>
> Tom DeMarco: "You cannot control what you cannot measure."

Nur wenn eine Sache verstanden ist, ist sie meßbar; man muß die Bedeutung der Zahlen kennen, die man den Objekten zuordnet und die man miteinander vergleicht. Die Gültigkeit von Operationen und Interpretationen auf den Meßwerten muß gezeigt werden, was eine Kenntnis der Objekte und ihrer Beziehungen voraussetzt. Andersherum entziehen sich nicht meßbare Objekte einer Kontrolle; ein Vergleich von Soll- und Ist-Werten kann nicht vorgenommen werden. Die Überprüfung, ob ein Ziel erreicht ist, kann nicht objektiv erfolgen. Gleichzeitig können aber durch die Kontrolle meßbarer Produkt- und Prozeßaspekte weitere Eigenschaften der Objekte aufgedeckt werden, die das Verständnis für die Sache erhöhen. Das grundlegende Verstehen einer Sache erlaubt ihre Messung, und das Messen fördert das Verständnis.

Muß die Quantifizierung eines Objekts erfolgen, so stellt sich die Frage welche Maße für diese Charakterisierung ausgewählt werden müssen. Die Auswahl der Maße ist dabei kontextabhängig, d.h. es gibt im allgemeinen keinen universellen Satz von Metriken zur Charakterisierung eines Objekts. Für jede spezifische Zielsetzung einer Charakterisierung oder Messung muß ein geeigneter Satz an Metriken ermittelt werden. Im Rahmen des MVP-Projekts wird dafür das G/Q/M-Paradigma verwendet [6]. Der G/Q/M-Ansatz wurde im Rahmen des TAME-Projekts entwickelt. Er dient der Formalisierung von Meßzielen und der Dokumentation von Kontexten einer Auswertung von Meßdaten [2]. Durch einen 'top-down'-Ansatz wird eine nachvollziehbare Begründung dafür gegeben, warum bestimmte Daten gesammelt werden sollen: durch die Spezifikation des Ziels (Goal) einer Untersuchung, dessen Verfeine-

rung in Fragen (Questions) und daraus abgeleitete Maße (Metrics) erhält man eine schrittweise Dokumentation, in der Motivation, Ansatz und Technik einer Messung dargelegt werden [23]. Ein zusammengehöriger Satz von Fragen und Metriken zu einem bestimmten Ziel wird von uns *Meßplan* bezeichnet. Er bietet die Grundlage, um zu bestimmen, wann eine Messung an einem Objekt durchgeführt werden soll und welche Güte die gesammelten Meßdaten besitzen müssen, um sinnvolle Interpretationen (im Kontext eines konkreten Ziels!) zuzulassen.

Der G/Q/M-Ansatz wurde schon in verschiedenen Projekten erfolgreich zur Spezifikation von Maßnahmen der Qualitätssicherung eingesetzt. Verwendet wurde und wird dieser Ansatz in Firmen wie AT&T, Burroughs, IBM und Teilen der NASA [23].

Eine detaillierte Diskussion über die Verbindung zwischen G/Q/M-Ansatz und existierender Technologie im Bereich 'Messen und Bewerten' findet in [18] statt.

Ein Beispiel für einen nach dem G/Q/M-Ansatz erstellten simplifizierten Meßplan zeigt Abbildung 2. Wie im Ziel beschrieben ist er für den Qualitätsmanager eines Projekts gedacht. Man beachte, daß der Qualitätsmanager sich in diesem Fall für die Qualität der Akzeptanztest-Methode interessiert. Angewandt wird aber die Akzeptanztest-Methode in einem konkreten Projekt von einem Qualitätsingenieur, d.h. der Projektmanager nimmt hier nur einen Beobachterstatus ein.

Goal: Analysiere die Akzeptanztest-Methode zum Zwecke der Verstehens der Effektivität aus dem Blickwinkel des Qualitätsmanagers bei Organisation X.

Q1: Wie wird die Akzeptanztest-Methode durchgeführt?

Q2: Welcher Produkttyp wird getestet?

Q3: Wieviele Fehlverhalten werden entdeckt?

Q4: Wie hoch sind die Kosten von Fehlverhalten?

M1: Anzahl Anforderungen

M2: Bedeutung jeder Anforderung

M3: Anzahl Tests pro Anforderung

M4: Konsistenz zwischen M2 und M3

M5: Länge des Quelltextes in Lines-of-Code

M6: Programmiersprache

M7: Anzahl entdeckter Fehler

Abb. 2: Ein Meßplan nach dem G/Q/M-Ansatz

2.3 Das MVP-Modell

Bisher wurde deutlich gemacht, daß eingesetzte Technologie nur schrittweise verbessert werden kann und daß dazu die Quantifizierung sowohl der Projektziele als auch der Verbesserungsziele notwendig ist. Es muß jedoch auch ein Weg gefunden werde, der die einzelnen Anwendungen der Technologie dokumentiert. Dadurch erhält man eine Beschreibung der Prozesse zur Entwicklung oder Betreuung eines Software-Produkts. Diese Tätigkeit wird Software-Prozeßmodellierung genannt. Das in diesem Abschnitt vorgestellte Modell zur Dokumentation von Software-Prozessen wurde im Rahmen des MVP-Projekts entwickelt.

Prozeßmodellierung als Basistechnologie zur Verbesserung von Prozessen ist in den letzten Jahren populär geworden [20, 21, 22]. Es existieren innerhalb dieser Forschungsrichtung eine Anzahl von verschiedenen Ansätzen, die versuchen die Entwicklung und Betreuung von Software zu unterstützen. Man kann jedoch mindestens die folgenden Motivationen für rechnergestütztes Assistieren von Projekten und Organisationen erkennen [9]:

- Menschliches Verstehen und Kommunikation
  Das Prozeßmodell dient als Grundlage für Analysen und Untersuchungen. Darüberhinaus können sich verschiedene Personen über das Prozeßmodell verständigen. Insbesondere kann die Prozeß-Beschreibung der Weitergabe von Prozeß-Wissen dienen, zum Beispiel in Form von Anleitungen oder Lehrmaterial.

- Unterstützung von Projektmanagement
  Prozeßmodelle unterteilen und klassifizieren Prozesse der realen Welt. Dadurch werden die Aktivitäten in einem Projekt handhabbar gemacht für das Projektmanagement. Aussagen lassen sich auf bestimmte Prozeß-Klassen spezialisieren.

- Rechnergestützte Anleitung während der Projektdurchführung
  Unterstützung durch den Rechner wird durch sog. prozeß-sensitive Software-Entwicklungsumgebungen (engl. process-sensitive engineering environments) geleistet [19]. Das Prozeßmodell ist Eingabeparameter für eine Prozeßmaschine, die das Modell instanziiert und dann mit Unterstützung von Menschen interpretiert.[1]

- Automatische Abwicklung von Prozeßmodellen
  Ist ein Prozeß vollständig verstanden, so kann er u.U. einer vollautomatischen Interpretation (execution) zugänglich gemacht werden. Dies ist bisher nur bei Tätigkeiten im Rahmen von Edit-Compile-Link-Zyklen oder bei Produktverwaltungen erfolgt. Es ist jedoch denk-

1. In der Fachwelt der Prozeßmodellierung hat sich hierfür der Begriff 'enactment' durchgesetzt. Dieser soll im Gegensatz zu 'execution' verdeutlichen, daß hier bei der Interpretation der Mensch eine gewisse Bedeutung hat; insbesondere sind die so abgewickelten Prozesse nicht-deterministisch, was eingeschränkte Aussagen über ihr Verhalten zur Laufzeit zuläßt. Dieser Aspekt spielt bei der Planung und Simulation eine große Rolle, ja er ist sogar der Grund für die Formulierung der projektspezifischen und organisatorischen Ziele.

bar, daß ähnlich wie bei der Kompilierung eine Automatisierungen von einzelnen Schritten bei der Entwicklung oder Betreuung von Software automatisiert werden können.

- Unterstützung von Prozeßverbesserung
  Alle bisher genannten Ziele dienen auch dazu, den Prozeß oder die Durchführung der Prozesse zu verbessern. Desweiteren kann durch Simulationen von Prozessen oder Analysen von Prozeßmodellen und Projekttraces das Wissen in Bezug auf die Technologie bei der Entwicklung und Betreuung von Software erweitert werden.

Das MVP-Projekt versucht auf allen fünf Bereichen Projektmitgliedern Unterstützung bereitzustellen. Die Notwendigkeit dieses Projekts wurde bei Maßnahmen zur Qualitätssicherung im Software Engineering Laboratory der NASA (NASA/SEL, Greenbelt, Maryland, USA) erkannt. Es stellte sich heraus, daß Messen und Bewerten von Software nur dann sinnvoll durchführbar ist, wenn man ein Grundverständnis der in der Umgebung ablaufenden Prozesse hat. Jedoch existierten bei den Entwicklern unterschiedliche Vorstellungen darüber wie im NASA/SEL Software entwickelt und betreut wird. Deshalb entschied man sich ein Projekt zu starten, das es ermöglicht, explizite Repräsentationen der ablaufenden Prozesse aus unterschiedlichen Blickwinkeln (engl. *views*) zu gewinnen.

Ziel dieses Projekts ist die Entwicklung eines Systems zur Definition, Planung, Abwicklung und Wiederverwendung von Software-Prozessen aus unterschiedlichen Blickwinkeln. Besonderer Wert wird dabei auf die quantitative Charakterisierung von einzelnen Prozeß-Elementen gelegt. Im MVP-Ansatz werden deshalb folgende Grundarten von Bausteinen unterschieden:

- *Produkt-Modelle*
  beschreiben die Struktur und die Eigenschaften von Komponenten der Software. Darunter verstehen wir nicht nur den Code sondern sämtliche Artefakte, die bei der Entwicklung oder Betreuung anfallen oder benötigt werden. Ein Produkt-Modell beschreibt die sichtbaren Attribute eines Produkts und dessen etwaige Verfeinerung in Teilprodukte der nächstkleineren Abstraktionsstufe.

- *Prozeß-Modelle*
  sind Repräsentationen für Aktivitäten in einem Projekt. MVP verzichtet aus Gründen der Formalisierbarkeit auf eine Beschreibung der einzelnen Bearbeitungsschritte im Sinne von Anweisungen in einem Programm[2]; MVP konzentriert sich eher auf die Beziehungen zwischen einzelnen Prozeß-Elementen. Somit werden nur Prozesse auf einer abstrakten Entwurfsebene, eben der Modellierungsebene betrachtet. MVP erlaubt deshalb *Software Process Modeling in-the-large*.
  In MVP ist ein Prozeß-Modell aus verschiedenen Teilen aufgebaut. Neben den sichtbaren Attributen und der Verfeinerung, die aber komplexer aufgebaut ist als bei Produkt-Model-

2. Diese Sichtweise wird üblicherweise als 'process programming' verstanden.

len, existieren Konstrukte mit denen die Synchronisation von Prozessen spezifiziert werden kann (Exit- und Entry-Criteria); darüberhinaus kann noch der nicht-lokale Kontext eines Prozesses in Form von zusätzlich benötigten Produkten, Prozessen oder Ressourcen spezifiziert werden.

- *Ressourcen-Modelle*
  beschreiben Typen von Elementen, die für die Durchführung von Prozessen benötigt werden. Bei MVP unterscheiden wir zwischen Personen mit einer bestimmten Aufgabe (Rollen) und Werkzeugen die einer Tätigkeit zugeordnet werden. Ähnlich wie bei Produkten werden hier nur die sichtbaren Attribute und etwaige Verfeinerungen einer Ressource beschrieben.

- *Attribut-Modelle für Produkte, Prozesse und Ressourcen*
  ergänzen die Beschreibungen der ersten drei Modelltypen. Dadurch wird eine Trennung vollzogen zwischen den grundlegenden Konzepten (Produkte, Prozesse oder Ressource) einerseits und charakteristischen Eigenschaften andererseits.

- *Globale Attribut-Modelle*
  Zusätzlich existieren globale Attribut-Modelle, die keinem Objekt der drei Modelltypen Produkt, Prozeß oder Ressource zugeordnet werden kann. Sie werden als globale Attribut-Modelle bezeichnet und beinhalten Informationen über "externe" Aspekte wie zum Beispiel *Zeit*.

Diese sieben Grundtypen (Prozesse, Produkte, Ressourcen und vier Attributtypen) können verfeinert und kombiniert werden. Somit entstehen komplexere Gebilde, die typische Projekte bzw. Projektstrukturen charakterisieren. Instanziiert man solch ein zusammengesetztes Prozeßmodell, so erhält man einen Projektplan, der als Grundlage für Analysen oder eine Projektabwicklung dient.

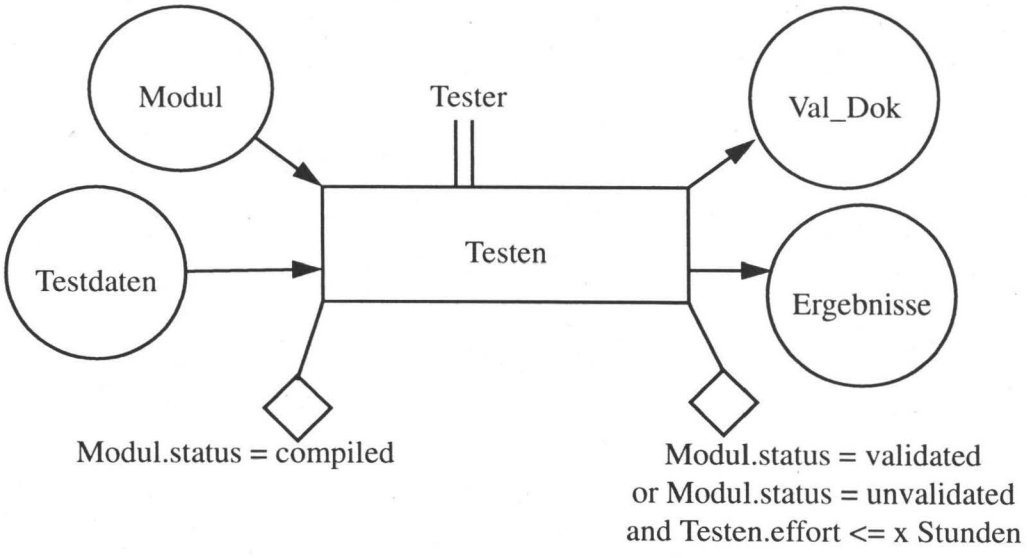

Abb. 3: Grafisch dargestelltes MVP-L-Prozeßmodell eines Testprozesses

Zur Dokumentation der Prozeßmodelle wurde in bisherigen Anwendungen der Formalismus MVP-L (multi-view process language) verwendet. Hierbei werden Modelle textuell beschrieben. Momentan existiert an Werkzeugunterstützung zur Modellierung ein syntaxgerichteter Editor und ein Werkzeug zur Modellierung von Produktflüssen. Eine Schnittstelle für die grafische Repräsentation von Prozeßelementen ist in Vorbereitung.

Ist nun mit diesen Hilfsmitteln die Abbildung auf Elemente der Modelltypen abgeschlossen, so werden Teilmengen der entwickelten Beschreibungen bestimmten Aufgaben zugeordnet. Wie diese Zuordnung nun im einzelnen geschieht, wird im nächsten Abschnitt behandelt.

3 Definition von Rollen

Prozeßmodelle und Projektpläne repräsentieren das Wissen einer Organisation bezüglich der Art und Weise wie Software-Produkte entwickelt und betreut werden. Der Mensch ist integraler Bestandteil der Prozesse. Es stellt sich nun die Frage, wie die Zuordnung der Prozesse zu den 'ausführenden' Menschen geschieht.

3.1 Der Rollenbegriff

In einem Software-Projekt existieren die unterschiedlichsten Arten von Tätigkeiten. Man faßt kleine Mengen dieser Tätigkeiten zusammen zu einer Aufgabe. Die zusammengehörigen Aufgaben einer oder mehrerer Personen in einem Software-Projekt werden als *Rolle* bezeichnet. Eine Rolle besitzt die funktionale Verantwortlichkeit zur Erreichung eines gegebenen Ziels.

In den vielen Systemen zur Modellierung von Software-Prozessen ist nun die Sichtweise realisiert, daß die Unterstützung einer Rolle eine Modellierung (Implementierung) aller ihrer Tätigkeiten im System verlangt. Humphrey und Feiler geben folgende Definition [13]: "[A role is] *the responsibility for enacting a process or subprocess definition.*"

Hier wird deutlich, daß die Zuordnung Mensch-Prozeß durch die beiden Zuordnungen Mensch-Rolle und Rolle-Prozeßbeschreibung vorgenommen wird. Eine Rolle wird dabei als eine Ansammlung von Aktivitäten verstanden. Eine Berücksichtigung des Ziels einer Rolle findet nur implizit statt, d.h. es wird beschrieben *wie* eine Rolle ihr Ziel erreichen soll, aber nicht *was* dieses Ziel ist. Eine Rolle ist also, entsprechend dieser Sichtweise, eine Abstraktion von Aktivitäten.

Einige der existierenden Systeme zur Unterstützung des Software-Prozesses verwenden exakt diesen, und zwar nur diesen logischen Zwischenschritt, durch Definition von Rollen in Software-Prozeßbeschreibungen. Dies impliziert, daß die Aspekte zur Unterstützung einer Rolle (also eines Projektmitgliedes) immer an Prozesse gekoppelt sind, d.h. die zu einer Rolle zusammengefaßten Tätigkeiten immer eine Repräsentation im System finden müssen. Eine Rolle kann mit diesem Ansatz nur dann sinnvoll unterstützt werden, wenn zumindest ein Teil ihrer Tätigkeiten im System formalisiert ist. Wir bezeichnen solche Rollen als "zustandsverändernd", da sie durch die Abwicklung der Prozesse den Projektzustand verändern.

Unserer Meinung nach muß dieser Rollenbegriff noch erweitert werden, damit durch Prozeßmodelle alle Rollen eines Software-Projekts unterstützt werden können. Denn eine Rolle kann auch dann sinnvoll von einer Prozeßbeschreibung oder einem Projektplan unterstützt werden, wenn ihre Aktivitäten nicht explizit repräsentiert sind.

Zum Beispiel führt die Rolle 'Qualitätsingenieur' die Anwendung einer Akzeptanztest-Methode durch. Wird diese Methode in einem Prozeß-Modell beschrieben, so entspricht die

Zuordnung Rolle und Methode obiger Definition. Darüberhinaus gibt es aber noch die Rolle des Qualitätsmanagers, der die Methode ausgewählt hat, und jetzt natürlich an der Qualität ihrer Durchführung interessiert ist. In den bisherigen Ansätzen wäre eine Anbindung nicht ausdrückbar.

Wir erweitern deshalb das existierende Vorgehen. Dadurch wird es möglich, die Verbindungen zwischen Rollen in der realen Welt und Informationen in der internen Projekt-Repräsentation formal zu beschreiben. Somit kann jedes Projektmitglied bei seiner Arbeit, für rechnergestützte Systeme nachvollziehbar, unterstützt werden. Eine Rolle wird im folgenden nicht nur als eine Menge von modellierten Prozessen verstanden, sondern auch als eine Abstraktion für den Informationsbedarf, dessen Deckung zur Erledigung bestimmter Aktivitäten in einem Software-Projekt notwendig ist [25]. Wir bezeichnen solche Rollen als "zustandserfragend". Dies bedeutet, daß die Aktivitäten einer Rolle im Modell selbst nicht beschrieben sein müssen; sie kann trotzdem davon profitieren, indem formal angegeben wird, welche Daten sie benötigt.

3.2 Die Verbindung zwischen Personen und Prozessen in MVP

Wie der vorherige Abschnitt deutlich machte, müssen unserer Meinung nach sowohl zustandsverändernde Rollen als auch zustandserfragende Rollen durch eine Software-Entwicklungsumgebung unterstützt werden.

Zustandsverändernde Rollen werden in MVP wie in anderen prozeß-sensitiven Software-Entwicklungsumgebungen unterstützt. Die Aktivitäten werden als Prozeßmodelle repräsentiert, instanziiert und vom Agenten gestartet oder beendet (mittels 'start' und 'complete' Ereignissen, die vom Agenten an das jeweilige Prozeßmodell verschickt werden). Dazu nimmt der Agent eine Rolle an, für die dann eine Liste von Prozessen existiert, die diese Rolle potentiell abwickeln kann. Diese Liste wird als Agenda bezeichnet. Der Agent kann einen Prozeß der den Zustand 'enabled' besitzt starten, indem er das Ereignis 'start' an den Prozeß schickt. Dieser wechselt dann in den Zustand 'active' und der Agent als Inhaber der Rolle besitzt die Verantwortung für die Abwicklung des Prozesses. Der Prozeß beschreibt den Arbeitsbereich der Rolle (engl. *work context*), indem lokale Daten und Werkzeuge bereitgestellt werden und eine Prozeßbeschreibung dem Agenten mitteilt, wie er den Prozeß abwikkeln soll. Der Agent wird dadurch bei seiner Arbeit angeleitet. Dieser Prozeß unterstützt weiterhin den Agenten, indem z.B. Werkzeuge automatisch aktiviert werden oder der Zugriff auf bestimmte Dateien gewährleistet wird. Außerdem kann die Software-Entwicklungsumge-

bung andere Rollen von Zustandsveränderungen (Produkte und Prozesse) benachrichtigen. Die Umgebung koordiniert verschiedene Rollen.

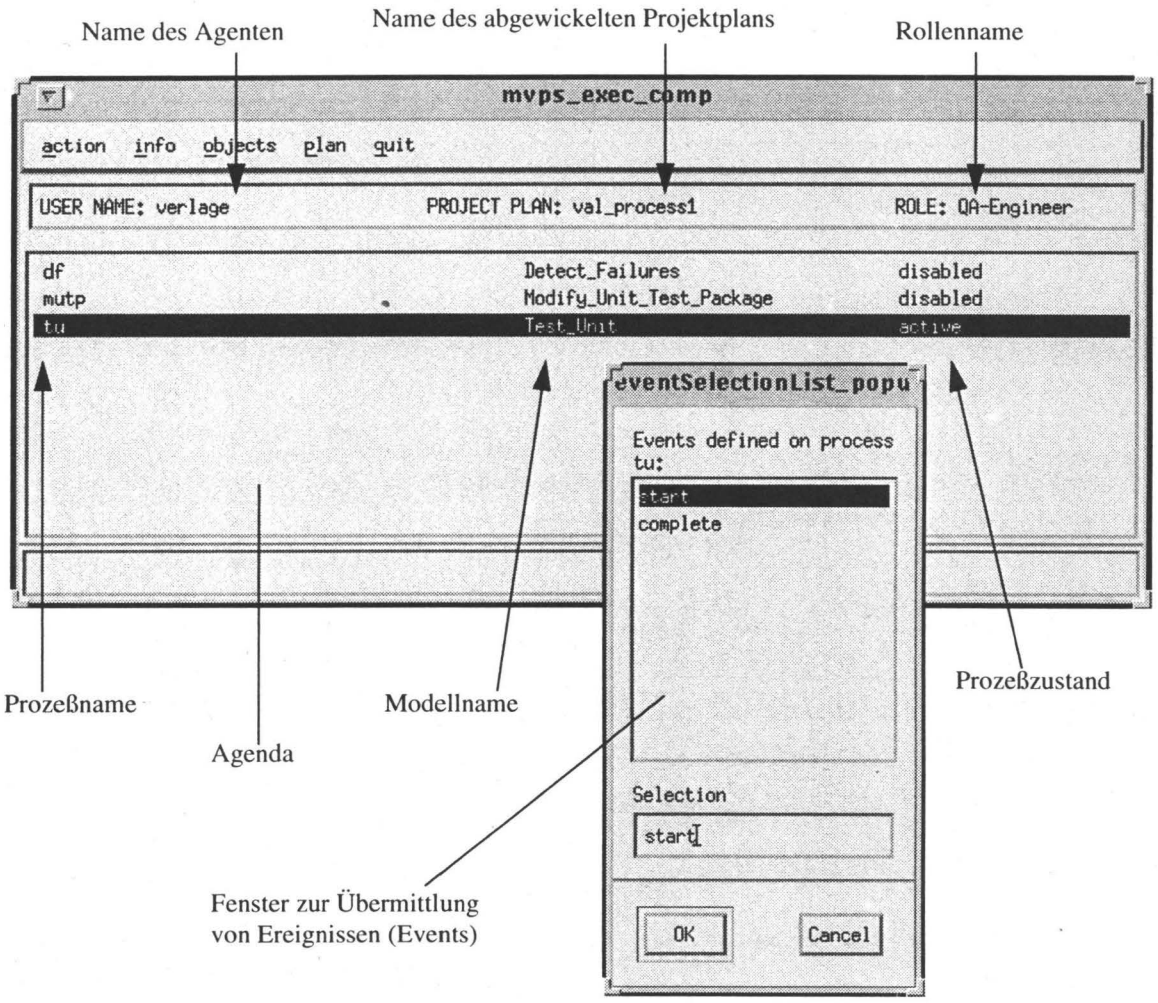

Abb. 4: Beispiel einer Benutzungsschnittstelle für eine zustandsverändernde Rolle

Für die Unterstützung zustandserfragender Rollen benutzen wir existierende Technologie, die schon in realen Projekten eingesetzt wurde. Zum einen ist dies der Goal/Question/Metric-Ansatz (G/Q/M) für die Entwicklung von Meßplänen, der bereits in Abschnitt 2.2 vorgestellt wurde. Zum anderen instrumentieren wir die Prozeßmodelle mit Werkzeugen der Qualitätssicherung. Dadurch erhalten wir eine formale Verbindung zwischen den Anforderungen einzelner Rollen und den Prozeßbeschreibungen. Bisher existieren ein Editor für die Erstellung der Meßpläne und ein Werkzeug für die Auswertung nach erfolgter Datensammlung.

Die Nutzung einer integrierten Projekt-Dokumentation in Form von G/Q/M-Meßplänen und MVP-L-Beschreibungen kann in fünf Schritten beschrieben werden. Dabei gehen wir davon aus, daß bereits die Modelle (Produkte, Prozesse, Ressourcen und entsprechende Attribut-Definitionen) vorliegen. Das heißt, wir betrachten nun, wie diese MVP-L-Beschreibungen als Grundlage für qualitätsorientierte Maßnahmen dienen können.

Zuerst muß eine kontextabhängige Zieldefinition der Messung vorgenommen werden (❶ in Abb. 5). Dieser Meßplan kann dann in einem 'bottom-up'-Ansatz dazu genutzt werden, die Meßdaten im Kontext eines übergeordneten Ziels zu interpretieren (❺ in Abb. 5). Der Meßplan und die gesammelten Werte sind somit auch für andere Personen interpretierbar.

Mit einem nach dem G/Q/M-Ansatz entwickelten Meßplan erhält man die zu betrachtenden Maße und Meßzeitpunkte. Dazu drücken die Projektmitglieder aus, an welchen Informationen sie speziell interessiert sind. In einem zweiten Schritt muß sichergestellt werden, daß die Daten auch gesammelt werden können und wie diese Sammlung vorgenommen werden muß (❷ in Abb. 5), d.h. wie die Meßdaten in den Projektplan integriert werden. In MVP können wir diese Maße als Attribute von Produkten und Prozessen ausdrücken. Diese Attribute werden dann mit Werkzeugen zur automatischen oder manuellen Erfassung von Produkt- und Prozeßcharakteristika verbunden (❸ in Abb. 5). So können oft die Eigenschaften von Produkten (z.B. Komplexität des Entwurfs) automatisch erfaßt werden (❹ in Abb. 5). Andere Daten dagegen müssen von Hand gesammelt werden, z.B. Kategorie des Entwurfs nach einem Review. Somit erhalten wir eine Verbindung zwischen der Rolle in einem Projekt und der Information, die diese Rolle zur Erledigung ihrer Aufgabe benötigt.

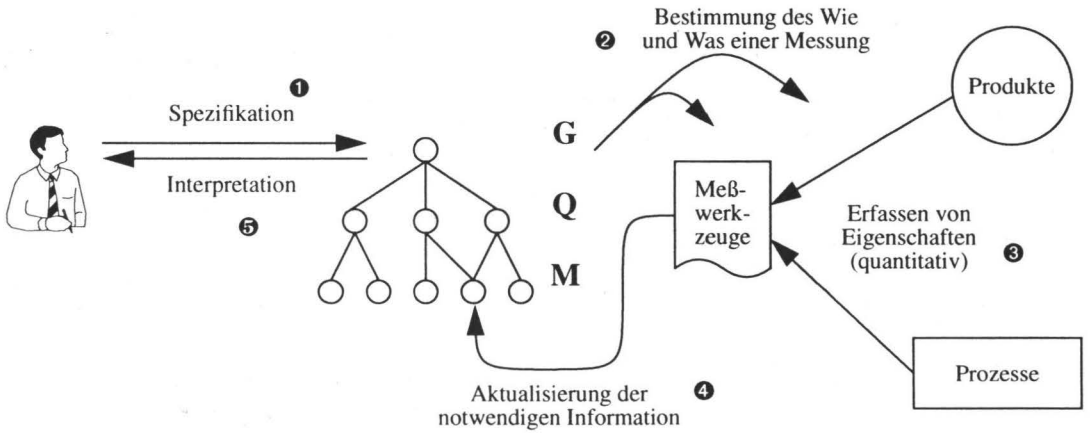

Abb. 5: Produkt- und Prozeßmaße zur Unterstützung einer Rolle

Die Modelle der Produkte, Prozesse und Ressourcen können dazu genutzt werden, die Meßdaten zu interpretieren. Oftmals trifft man in der Praxis auf Meßreihen, bei denen keine Zuordnung zu den Charakteristika der Instanzen vorgenommen werden kann (z.B. eine Menge von Werten die die Komplexität von Modulen charakterisieren, statt getrennte Menge entsprechend den Klassen 'E-/A-Modul', 'Koordinationsmodul' und 'Transformationsmodul'). Hierdurch wird eine Analyse, die auf die Entdeckung bestimmter Beziehungen zwischen Ausprägungen von Prozeß-Elementen abzielt (z.B. "Welchen Einfluß hat die Komplexität auf die Fehlerhäufigkeit?") erschwert. Oftmals sind nämlich die Zusammenhänge zwischen Charakteristika nicht allgemeiner Natur (z.B. "Für alle Module gilt: je höher die Komplexität, um so wahrscheinlicher ist ein Fehler im Modul enthalten!"), sondern Zusammenhänge können

stärker sein, je eingeschränkter die Klasse der betrachteten Population ist (z.B. "Besonders für Koordinationsmodule steigt die Fehlerrate bei steigender Komplexität überproportional an.") [24]. Ein Repository muß also die gesammelten Daten zusammen mit den gemessenen Objekten (Prozesse, Produkte und Ressourcen) verwalten, anstatt direkt zu aggregieren.

In herkömmlichen SEUen werden entweder nur die technischen Rollen oder nur die Management-Rollen unterstützt. Erstere sind hauptsächlich zustandsverändernd, letztere dagegen zustandserfragend im Sinne von MVP. Beide müssen jedoch unterstützt werden. Für jede zustandsverändernde Rolle existieren Prozeßmodelle, die die Aktivitäten repräsentieren. Für jede zustandserfragende Rolle existieren laut dem in Abbildung 5 gezeigten Vorgehen G/Q/M-Meßpläne, die speziell auf die Erledigung ihrer Tätigkeiten zugeschnitten ist; dies entspricht einer Sicht aus einer bestimmten Perspektive. Verschiedene Sichten einer oder mehrerer Rollen werden dann von einem gemeinsamen Projektplan in eine entsprechende Repräsentation projiziert. Alle Rollen, zustandsverändernde und zustandserfragende, beziehen sich also auf die selbe Information, nämlich den Projektplan.

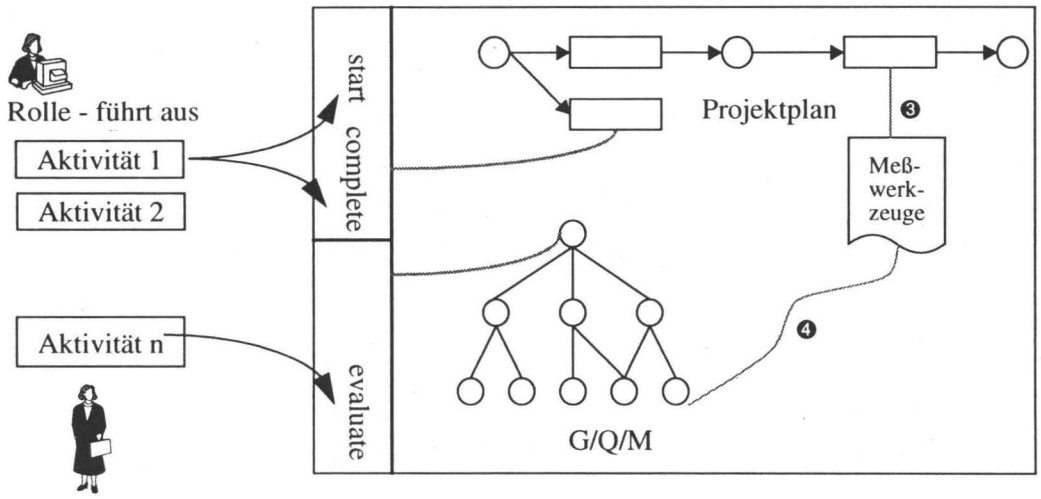

Abb. 6: Unterstützung von zustandsverändernden und zustandserfragenden Rollen

In diesem Abschnitt haben wir gezeigt, wie eine Rolle mit der notwendigen Information versorgt wird, um ihre Aufgabe zu erfüllen. Natürlich werden diese Informationen benutzt, um die Produkte zu verändern und die Prozesse zu steuern. Hierbei gibt es unterschiedliche Wege, inwieweit die gesammelten Daten zu einer maßorientierten Abwicklung genutzt werden können. Darauf gehen wir im nächsten Abschnitt näher ein.

4 Projekt-Abwicklung mittels MVP

Im letzten Abschnitt wurde erläutert, wie der Informationsaustausch zwischen Rollen und Projekt-Repräsentationen aufgebaut werden kann. Es wurde deutlich, daß die einzelnen Modelle einer Projekt-Beschreibung für das Messen und Bewerten von Software-Produkten und -Prozessen genutzt werden können. Jetzt illustrieren wir an diesem Modell die unterschiedlichen Nutzungen von Projektplänen. Hierbei unterscheiden wir drei Arten der Unterstützung durch die Modellierung von Software-Prozessen, die in den nachfolgenden Abschnitten eingehend erläutert werden:

- *Prozeßmodelle für das prinzipielle Verständnis*
  Es existieren die Beschreibungen der Projekte ohne eine direkte Anbindung an konkrete Projekte. Sie werden jedoch für deren Verständnis herangezogen.
- *Prozeßmodelle für Informations-Management*
  Die Modelle beschreiben das Projekt einschließlich der Maße und Meßpunkte (wann und was gemessen wird). Es werden die zustandserfragenden Rollen unterstützt.
- *Prozeßmodelle für Projekt-Steuerung*
  Die Modelle dienen zur Repräsentation und Interpretation von Meßdaten. Weiterhin wird der Projekt-Fortschritt mit ihrer Hilfe protokolliert. Es werden die zustandserfragenden und zustandsverändernden Rollen unterstützt.

Die Alternativen unterscheiden sich sowohl im Aufwand bei der Realisierung als auch bei ihrem Einfluß auf die Projekt-Organisation. Darüberhinaus beschreiben sie drei mögliche Stufen bei einer schrittweisen Einführung der Modellierung von Software-Prozessen in einer spezifischen Umgebung. Jede Stufe ist Ausgangsbasis zur Erreichung der nächsten Stufe. Alle drei Stufen können jedoch dazu beitragen, die Entwicklungs- und Wartungstätigkeiten im Sinne qualitativ höherwertiger Produkteigenschaften zu verbessern.

4.1 Prinzipielles Verständnis

In dieser einfachsten Nutzung wird die Modellierung von Software-Prozessen dazu verwendet, explizite Repräsentationen von Projektwissen zu bekommen. Die erste Stufe des MVP-Systems wird dazu benutzt, Projektpläne aus unterschiedlichen Perspektiven zu erstellen. Die Projektmitglieder beschreiben ihre durchgeführten Tätigkeiten aus ihrer subjektiven Sicht. Diese Ergebnisse der *deskriptiven Modellierung* werden dann miteinander verglichen und ein konsistentes Modell aus verschiedenen Sichten wird abgeleitet. Dadurch können unterschiedliche Auffassungen über ein Projekt ermittelt werden. Das Modell dient der Kommunikation, da ein gemeinsames Verständnis gefördert wird. Das Resultat dieser Modellierung dient der Dokumentation von Projektschritten und der Schaffung eines einheitlichen Verständnisses zwischen Projekt-Mitgliedern über die gesamten Projekttätigkeiten. Dadurch können sich Per-

sonen über bestimmte Aspekte eines Prozesses austauschen. Es werden hier sowohl textuelle als auch grafische Notationen verwendet.

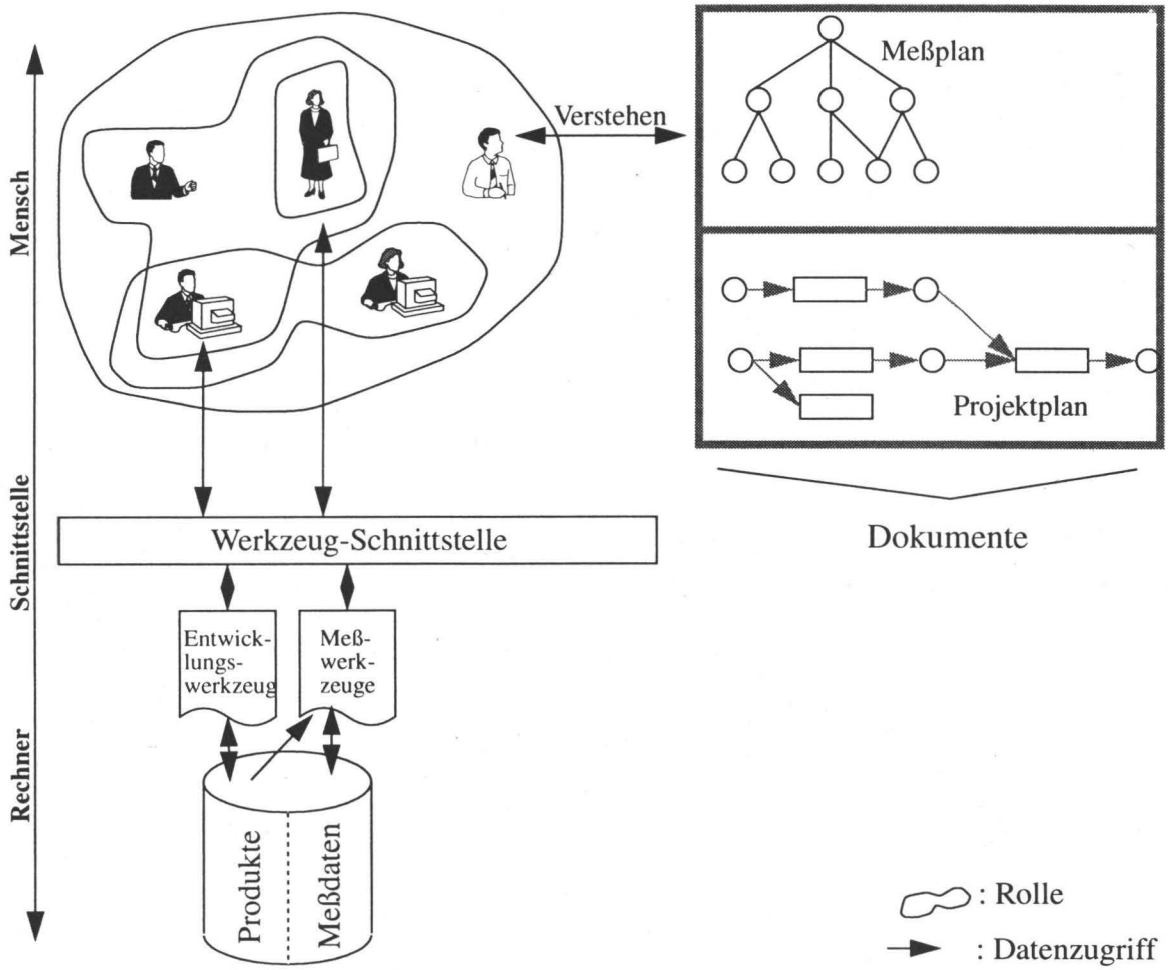

Abb. 7: Erste Stufe: Verstehen von Software-Prozessen

Anwendungen der MVP-Notationen in realen Umgebungen haben gezeigt, daß schon diese Form der Unterstützung von Rollen eines Projekts Vorteile bringt. Durch die Darstellung des Prozesses in einer formalen Notation können Erkenntnisse über diesen abgeleitet werden, die zu dessen Verbesserung beitragen können. So konnten konkret bei einer der oben angesprochenen Fallstudien durch den Einsatz von MVP-L einige Schwachstellen eines in der Praxis genutzten Wiederverwendungsprozesses entdeckt werden [16]:

Während einer Kooperation mit der amerikanischen Firma TRW wurde MVP-L eingesetzt, um eine natürlich-sprachliche Prozeßbeschreibung in einen formalen Projektplan umzusetzen. Dieser wurde dann auf Konsistenz und Vollständigkeit überprüft. Die Ergebnisse wurden dann benutzt, um Probleme zu beseitigen und den Wiederverwendungsprozeß an sich zu verbessern. Letztendlich wurde die Beschreibung in MVP-L wieder in eine natürlichsprachliche Dokumentation umgesetzt, um den Anwendern den Prozeß in einer vertrauten Form anzubieten.

Kennzeichnend für diese Form des Einsatzes von Prozeßmodellen ist jedoch, daß es keine Anbindung von Modellen und Werkzeugen gibt (siehe Abb. 7). Das MVP-System bietet in dieser Stufe nur Hilfe für die Modellierung von Projekt- und Meßplänen. Genutzt werden von den Rollen im konkreten Projekt die Resultate einer Modellierung und nicht das Modellierungswerkzeug selbst. Außerdem wird die Kommunikation zwischen einzelnen Rollen im Projekt nicht unterstützt. Die Schnittstelle zwischen Projekt-Mitgliedern und den Projektdaten ist durch die verwendeten Werkzeuge bestimmt. Erst nach Beendigung des Projekts werden Erfahrungen bzgl. des Modells verarbeitet. Durch die statische Natur ist es ungeeignet, um Projektzustände zu repräsentieren.

In den bisherigen Prozeßmodellierungen mit der Sprache MVP-L hatten wir keine expliziten Anforderungen an die Eigenschaften der modellierten Prozesse. In erster Linie wurden die Modellierungen durchgeführt, um die prinzipielle Eignung des MVP-Modells (Produkte, Prozesse, Ressourcen und deren Attribute inkl. Beziehungen zwischen diesen Elementen) zu zeigen. Spätere Arbeiten auf diesem Gebiet werden sich sicherlich damit beschäftigen müssen, inwieweit bestimmte Eigenschaften der zu modellierenden Prozesse die Aktivität der Beschreibungen dieser Projektschritte beeinflussen. Hiervon erwarten wir auch Ergebnisse, die Auskunft über die generelle Einsetzbarkeit der Prozeßmodellierung auf dem Gebiet der Software-Entwicklung und -Betreuung geben.

4.2 Informations-Management

Die explizite Repräsentation von Tätigkeiten einer Umgebung erlaubt die Definition von Meßpunkten. Aus den Meß- und Projektplänen wird klar ersichtlich, warum, wann und welche Daten gesammelt werden sollen, und wie sie im Kontext ausgewertet werden müssen (siehe auch Abschnitt 3.2). Dadurch werden Daten von Prozessen und Produkten in einem Projekt anhand von Modellen interpretierbar. Außerdem erhält man quantifizierte Aussagen über das aktuelle Projekt, die für jeden nachvollziehbar sind, da sie auf vorher ermittelten, expliziten Modellen beruhen. Werden Modelle instanziiert, so erhält jeder einzelne Prozeß im Projekt seine eigene Repräsentation im unterstützenden System. Nun ist man in der Lage, mit Hilfe der Modelle Projektzustände zu repräsentieren und Ausschnitte den zustandserfragenden Rollen zu präsentieren. Durch die Verbindung von Meßplänen und Projektplänen einerseits und den gesammelten Daten andererseits erhält man quantifizierte Aussagen über das aktuelle Projekt, welche für jeden nachvollziehbar sind, da sie auf vorher ermittelten, expliziten Modellen beruhen. Die automatisierte Unterstützung erfolgt jedoch nur für zustandserfragende Rollen.

Beispiele für die Integration von Meßwerkzeugen in Software-Entwicklungsumgebungen bzw. die Unterstützung von Informations-Management durch Werkzeugverbunde wird in [17] gegeben. Die dort präsentierte Übersicht stellt jedoch klar, daß die Unterstützung heute meist nur in Form von Werkzeugen zur Datensammlung und Datenanalyse angeboten wird. Die

Interpretation von Meßdaten im Kontext eines spezifischen Meßziels ist nicht vorhanden. Dafür müßte eine umfassendere Unterstützung angeboten werden.

Man erkennt in Abbildung 8, daß eine Rolle nicht mehr direkt die Daten eines Meßwerkzeugs interpretiert, sondern dafür den vorher erstellten Meßplan benutzt. Darüberhinaus existiert eine Anbindung der Meßwerkzeuge an den Projekt- und den Meßplan, d.h. es ist dokumentiert, wann welche Daten gesammelt werden sollen. In Erweiterung zur ersten Stufe können die Meßwerkzeuge jedoch darauf zugreifen und damit gesteuert werden.

Diese Form der Nutzung von Prozeß-Modellen für die Unterstützung von Software-Projekten wird seit einiger Zeit beim NASA/SEL angewendet. Projektdaten können durch Einbindung der Meßpläne gesammelt werden. So wird der momentane Zustand eines Projekts auf quantitative Aussagen zur Beantwortung der zusammenhängenden Fragen eines Ziels projiziert. Ein Werkzeug für das Informations-Management ist das Software Management Environment (SME), welches eine Instanz des TAME-Modells [6] ist.

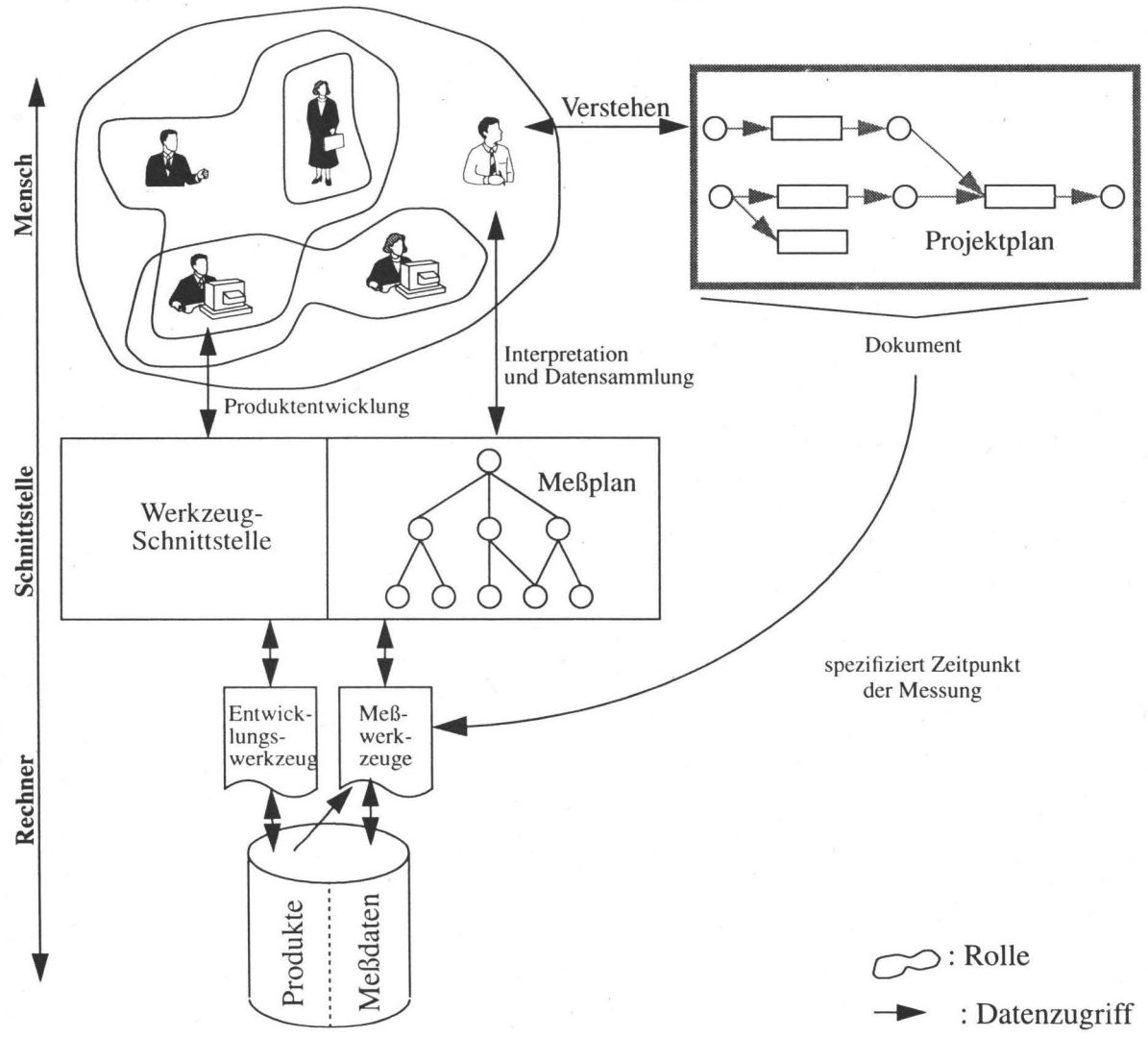

Abb. 8: Zweite Stufe: Informations-Management

In dieser zweiten Stufe ist das MVP-System ein Werkzeug zur Repräsentation eines aktuellen Projekts und dessen Charakterisierung in quantitativen Ausdrücken. Jedoch wird in diesem Modell die Kooperation zwischen Projektmitgliedern nicht durch den Rechner unterstützt. Außerdem ist nicht beschrieben, von wem die Sammlung der Daten veranlaßt wird, bzw. es findet keine automatische, unterstützte Aktivierung der Meßwerkzeuge statt.

Insbesondere wird der Zustand des realen Projekts nicht automatisch erfaßt, sondern jede Änderung von Projektvariablen muß dem MVP-System mitgeteilt werden. Dadurch wird das Risiko erhöht, daß die Daten nicht mehr die Realität widerspiegeln. Die Anbindung wird erst dadurch erreicht, daß man für jedes Projektmitglied eine angepaßte Schnittstelle zum Projektplan anbietet. In diesem sind ja schon die einzelnen Agenten durch Instanzen von Ressourcen-Modellen repräsentiert.

4.3 Projekt-Steuerung

Da schon einzelne Agenten als Ressourcen in den Prozeß-Modellen spezifiziert werden können, benutzt die dritte Stufe der Unterstützung Projektpläne, um ihre Arbeit zu koordinieren. Hier werden jetzt auch zustandsverändernde Rollen unterstützt. Die Arbeitsbereiche eines Prozesses werden mit Eigenschaften, die für das Projektmanagement wichtig sind, attributiert. Dadurch erhält man die Möglichkeit zur rechnergestützten Kooperation der Projekt-Mitglieder und zur automatischen Interpretation des Projekt-Zustandes, indem gleichzeitig zustandsverändernde und zustandserfragende Rollen durch ein und denselben Projektplan unterstützt werden.

Entscheidend ist hier, daß die Projektmitglieder in ihrer Arbeit vollständig vom System unterstützt werden im Sinne einer prozeßorientierten Entwicklungsumgebung (engl. *Process-Centered Engineering Environment - PCEE*). Im Gegensatz zu produkt-orientierten Hilfen, wie etwa Werkzeugen für das Konfigurationsmanagement, wird der Entwickler hier aktiv bei seiner Arbeit unterstützt. Dieser Ansatz zur Entwicklung und Betreuung von Software-Systemen ist schon ausreichend diskutiert worden [21, 22]. Wichtig ist hier, daß wir im MVP-System durch die Integration von Meßplänen in der Lage sind, quantitative Eigenschaften von Produkten und Prozessen zu spezifizieren und zu beobachten. Somit kann ein Agent als Besitzer eines Prozesses von Zeit zu Zeit Informationen über diese bestimmte Tätigkeit angeben, z.B. wie lange es noch bis zur Fertigstellung dauert oder wie seine subjektive Beurteilung der empfangenen Produkte bezüglich ihrer Lesbarkeit ist. Diese Werte können als Parameter in verschiedene Funktionen eingehen, die z.B. den Status des Projekts bewerten oder die Güte anderer Prozesse, die die empfangenen Produkte hergestellt haben, beurteilen. Auf diese Weise kommunizieren zustandsverändernde und zustandserfragende Rollen.

Die Art der Werte, die während der Projektabwicklung gesammelt werden, hängt stark von den Charakteristika des Projekts und der Umgebung ab. Wir stellen deshalb keinen Standardsatz an Attributen bereit, da wie schon erwähnt diese von den Zielen einer Messung und

Bewertung abhängen. Jede Umgebung verwendet verschiedene Maße und Interpretationsmodelle um ihre Projekte zu analysieren, zu kontrollieren und zu prognostizieren. Der MVP-Ansatz bietet jedoch die Möglichkeit einen existierenden Satz an Prozeß-Technologie zu adaptieren und ihn an spezifische Bedürfnisse anzupassen.

Diese Form der Unterstützung von qualitätsorientierten Projektplänen soll im Rahmen des ESPRIT-Projekts #9090 PERFECT[3] entwickelt werden. Die bereits kommerziell erhältliche prozeß-sensitive Software-Entwicklungsumgebung *Process Weaver* [8] soll um ein System für Produktverwaltung (Adele) und Infrastruktur zur Qualitätsverbesserung erweitert werden. Die Bereitstellung der Technologie für den zweiten Ausbaukomplex (siehe Abschnitt 4.2) erfolgt durch die Arbeitsgruppe Software Engineering des Fachbereichs Informatik an der Universität Kaiserslautern. Kernpunkte werden im wesentlichen die Integration von Meßplänen und Meßwerkzeugen sein.

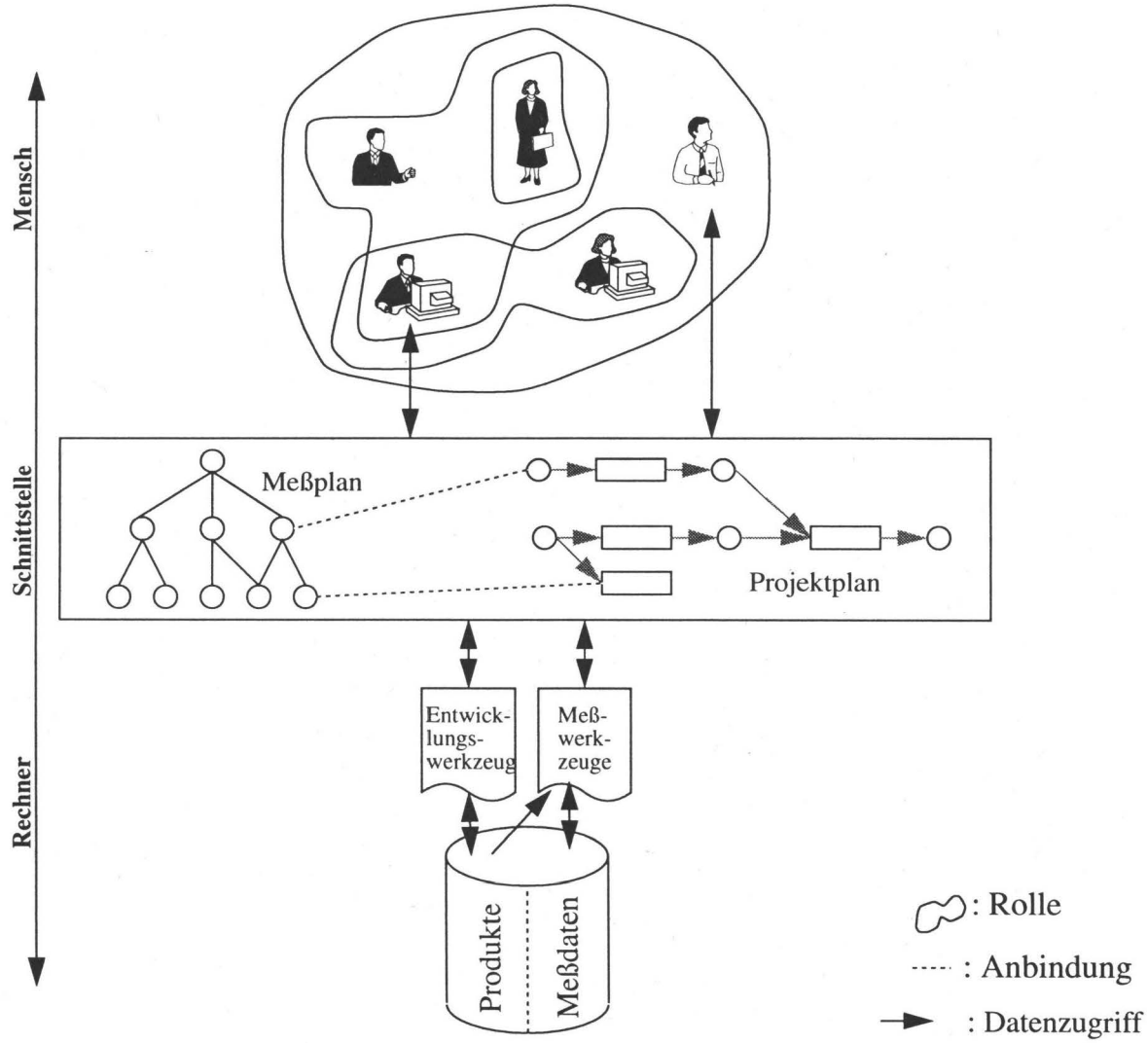

Abb. 9: Dritte Stufe: Maßorientierte Projektabwicklung

3. Das Konsortium besteht aus den Organisationen: Cap Gemini Innovation (Frankreich), Daimler Benz (Deutschland), LGI (Frankreich), Q-Labs (Schweden), Robert BOSCH GmbH (Deutschland), Siemens Norway (Norwegen), Sintef (Norwegen), Universität Kaiserslautern (Deutschland)

Die in diesem Abschnitt präsentierte dritte Stufe der Adaption von qualitätsorientierter Prozeß-Technologie vereint die Vorteile der ersten beiden Stufen und bietet darüberhinaus noch weitere Möglichkeiten der Nutzung einer Lösung, die den Aspekt *Messen und Bewerten* in prozeßorientierte Entwicklungsumgebungen integriert:

- Die Notwendigkeit eines einheitlichen Modells zur Unterstützung aller Rollen macht eine vorherige Abstimmung notwendig. Die unterschiedlichen Sichten einzelner Personen werden zu einer Projektrepräsentation zusammengefaßt. Dadurch werden die Schnittstellen zwischen Personen festgelegt. Das Modell dient als Referenz während der Projektabwicklung.

- Das System unterstützt sowohl die manuelle und automatische Sammlung als auch die Speicherung von Meßdaten. Einige Arten von Messungen bieten sich gerade dazu an, automatisch gesammelt zu werden. Ein einfaches Beispiel sind statische Quelltext-Metriken, die aus einer versionskontrollierten Bibliothek herausgezogen werden können. Andere Daten können nur von Projekt-Mitgliedern bereitgestellt werden, wie etwa die Charakterisierung von entdeckten Fehlern in der Software. Aber auch hier kann automatische Unterstützung in Form von formblattorientierten Werkzeugen angeboten werden. Es kann jedoch nicht genug betont werden, daß diese Maßnahmen nur im Einklang mit den Projektmitgliedern durchgeführt werden können; sie sollten deshalb nicht dazu verwendet werden, das Personal zu überwachen.
  Die Verwaltung der Meßdaten muß in einer kontextabhängigen Form erfolgen. In erster Linie geben die ermittelten Werte nur Auskunft über das konkrete Produkt, den Prozeß oder die Ressourcen. Um spätere Rückschlüsse zu ermöglichen, muß deshalb genau festgehalten werden, unter welchen Umständen und von welchen Instanzen welcher Modelle diese Daten ermittelt wurden. Hierbei dienen Meß- und Projektpläne die geeignete Unterstützung.

- Das MVP-System macht Projektzustände sichtbar. Ein Projektzustand umfaßt die Zustände aller Prozesse und Produkte, und Werte für alle im Projekt definierten Maße. Durch die Sichtbarmachung des Projektzustandes können sowohl Projektverantwortliche als auch Entwickler diejenigen Ausschnitte von Produkten und Prozessen beobachten, die sie als für den Erfolg des Projekts entscheidend definiert haben. Diese Rollen wurden als zustandserfragend charakterisiert.

- Als Projekt-Steuerungssystem kann das MVP-System den Projektzustand dazu benutzen, die Projektmitglieder darüber zu informieren, welche Prozesse sie aktivieren können, welche gerade aktiv sind, und welche Teile des Projektes durch unerwünschte Entwicklungen im Projekt berührt sind. Aktivitäten in diesem Sinne sind nicht kleine Tätigkeiten wie etwa Kompilieren oder Binden, sondern umfassendere Tätigkeiten wie etwa das Entwerfen eines Subsystems oder die Planung eines Treffens der Qualitätssicherungsgruppe. Diese Art der

Steuerung hilft den Projektmitgliedern ihre Aktivitäten miteinander zu koordinieren und es kann entdeckt werden, wann Aktivitäten miteinander in Konflikt kommen.

- Letztendlich ist man in der Lage, Analysen bezüglich eines einzelnen oder einer Menge von Projekten durchzuführen. Die gesammelten Daten werden dazu benutzt, während des laufenden Projekts umfangreiche Informationen über die Erreichung von Zielen, Einhaltung von Zeitplänen oder Qualitätsanforderungen bereitzuhalten. Hier wird noch einmal die besondere Bedeutung der Quantifizierung von projektspezifischen und organisatorischen Zielen deutlich. Dadurch wird ein automatischer Soll-/Ist-Vergleich bezüglich der Erreichung von Zielen ermöglicht. Dieser Vergleich kann periodisch erfolgen, und so ist eine projektbegleitende, ständige Überwachung eines Projekts im Hinblick auf erwartete Ergebnisse möglich. Die Daten werden mit Hilfe der Meßpläne interpretiert, und die Ergebnisse den Projektmitgliedern mitgeteilt. Die Auskünfte dienen dazu, das Projekt zu beurteilen und können als Grundlage für korrigierende Maßnahmen verwendet werden.
  Als Ergänzung dazu existiert die *post-mortem* Analyse von Projekten. Nach der Beendigung eines Projekts wird der während der Abwicklung aufgezeichnete Projekt-Trace dazu benutzt, um umfassende Analysen bezüglich der durchgeführten Prozesse durchzuführen. Aufzeichnungen aus mehreren Projekten können dazu dienen, umgebungsspezifische Charakteristika zu erkennen und zu bewerten. Dadurch könnte man z.B. Zusammenhänge zwischen der gewählten Vorgehensweise (dem Prozeßmodell) und der Qualität der Software erkennen.

5 Ein erster Prototyp des MVP-Systems

Die AG Software Engineering an der Universität Kaiserslautern hat Systeme zur Unterstützung der Prozeßtechnologie entwickelt. Diese Werkzeuge stellen einen initialen Satz an Unterstützungen für die Modellierung und Planung von Projekten und deren automatisierte Abwicklung zur Verfügung.

Zur Modellierung von Prozeßbeschreibungen in MVP-L steht ein Modellierungswerkzeug zur Verfügung (MoST - Modeling Support Tool). Man kann Dateien in sog. *globale Tabellen* laden, die Konsistenz der darin enthaltenen Prozeß-, Produkt, Ressourcen- und Attributmodelle überprüfen und diese Elemente einzeln editieren. Die folgende Abbildung zeigt entsprechende Darstellungen der Benutzungsschnittstelle.

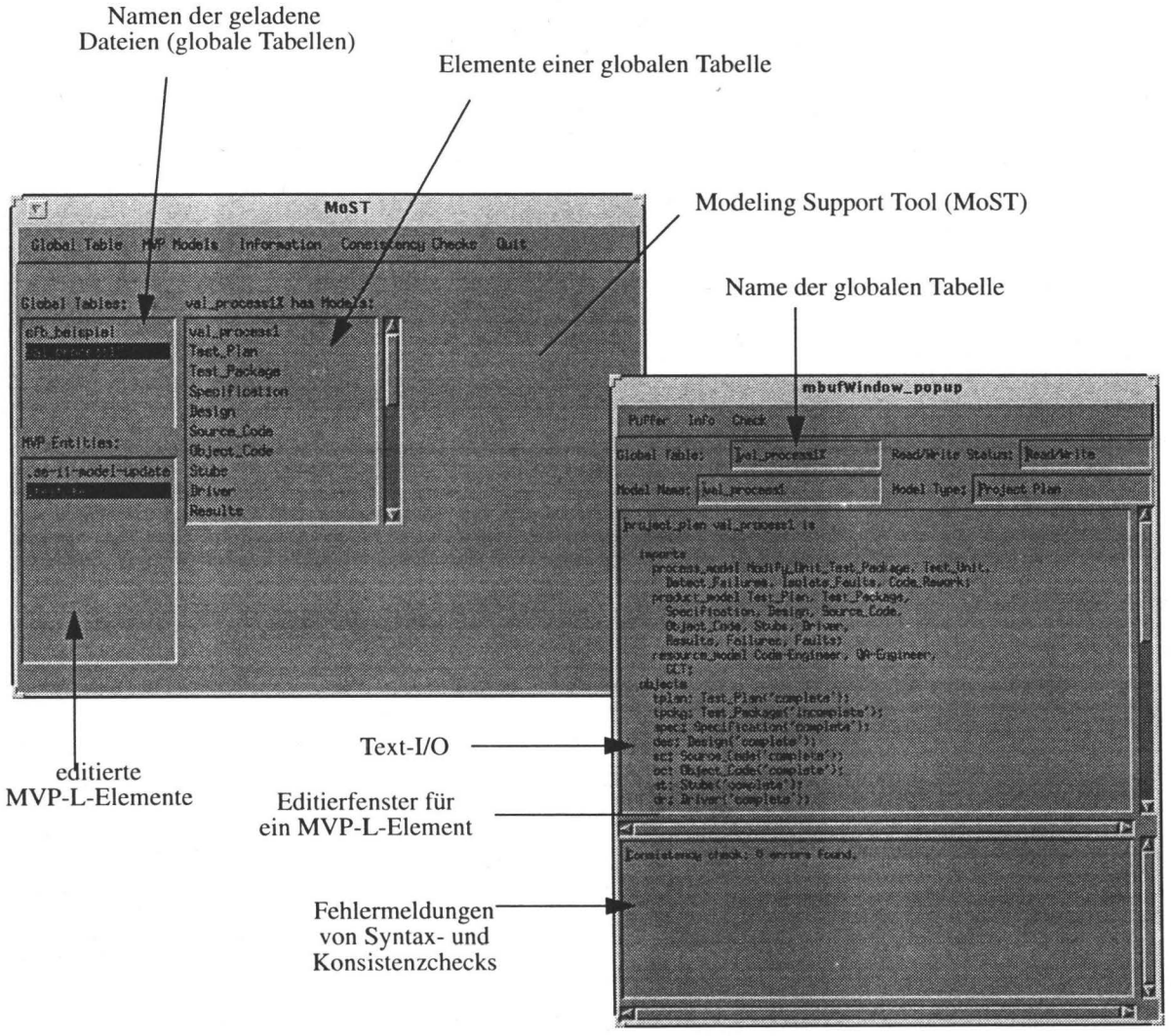

Abb. 10: Benutzungsschnittstelle des Modeling Support Tool (MoST) von MVP-L

Das Werkzeug MoST ermöglicht die Erstellung eines konsistenten Projektplans. Dieser beschreibt die auf ein konkretes Projekt zugeschnittenen Prozeßmodelle und ihre Verbindun-

gen (z.B. über Produktfluß oder Synchronisation durch Vor- und Nachbedingungen). Somit unterstützt MoST die Modellierungs- und Planungsphase eines Projekts.

Der von MoST erstellte Projektplan dient als Eingabe in einen Projektplaninterpreter.Der Projektplaninterpreter zusammen mit Meß- und Entwicklungswerkzeugen bildet eine prozeß-sensitive Software-Entwicklungsumgebung (siehe Abschnitt 4.3) [18]. Der Projektplan wird interpretiert und durch Kommunikation mit dem Benutzer findet eine kontinuierliche Dokumentation und Unterstützung des realen Projekts statt. Ziel des MVP-Projekts ist die Entwicklung einer prozeß-sensitiven Software-Entwicklungsumgebung, die den vollen Sprachumfang von MVP-L operationalisieren kann. Zur Zeit existiert eine erster Prototyp mit einer eingeschränkten Funktionalität. Mehrere Projektpläne können instanziiert und abgewickelt werden. Aber es ist lediglich möglich, sog. "flache" Prozeßmodelle abzuwickeln, d.h. Verfeinerungen von Produkten, Prozessen und Ressourcen sind nicht möglich. Ein Benutzer kann verschiedene Rollen in einem Projekt annehmen und diesen Rollen zugeordnete Prozesse abwickeln (zur Illustration der Benutzungsschnittstelle siehe Abbildung 4). Die Architektur des prototypischen MVP-Systems ist in Abbildung 11 dargestellt. Eine Komponente *Superserver* ist für die Instanziierung der Projektpläne verantwortlich. Jede Instanz eines Projektplans wird von einem eigenen *Projektplanserver* verwaltet. Die einzelnen *MVP-S-Clients* dienen der Kommunikation mit den Benutzern. Hier können sowohl zustandsverändernde als auch zustandserfragende Rollen unterstützt werden. Jede Zustandsabfrage oder Zustandstransformation wird vom Projektplanserver bearbeitet. Dabei werden Anfragen nach Projektplanzuständen, Meßdaten oder Events vom Projektplanserver entgegengenommen, und Informationen dem Benutzer mitgeteilt.

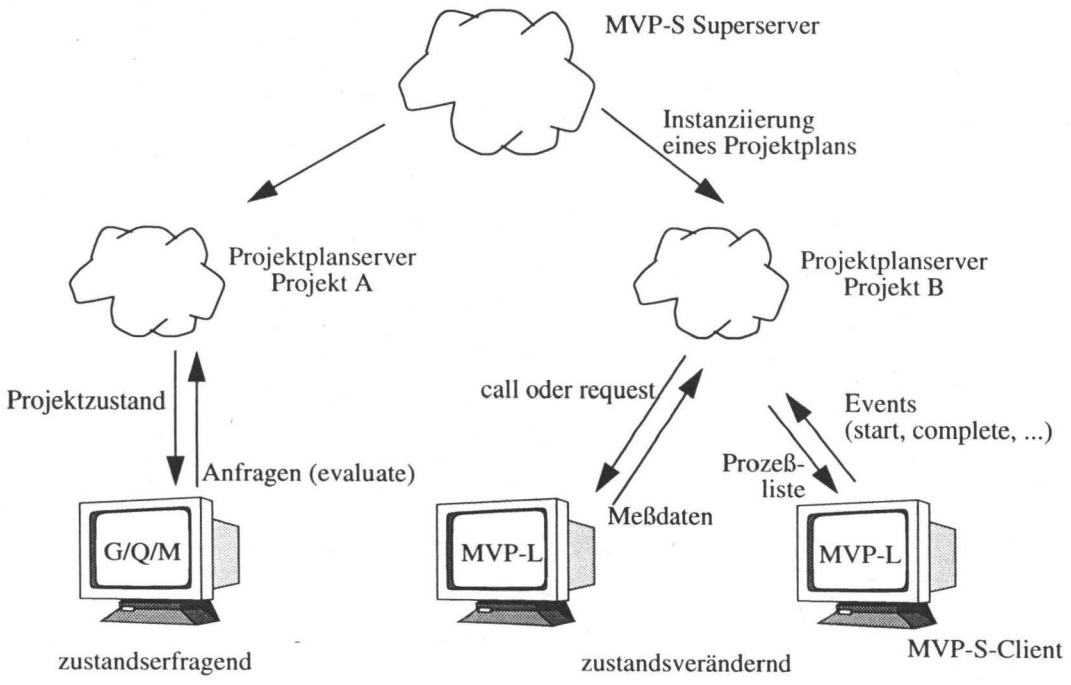

Abb. 11: Architektur des prototypischen MVP-Systems

Die Rollen eines Projekts interagieren auf zwei Arten mit dem MVP-System. Die Nutzung des Projektplans in Form von zustandsverändernden und zustandserfragenden Rollen wurde schon in Kapitel 3.2 behandelt. Dies entspricht im wesentlichen Schritt ❺ in Abbildung 5. Aber die einzelnen Projektmitglieder sind auch für die Angabe von Meßdaten verantwortlich, was dem Schritt ❸ in Abbildung 5 entspricht. Dies wurde im letzten Kapitel nur kurz angesprochen. Hier wollen wir auf den internen Mechanismus des MVP-Systems eingehen, der die Datensammlung ermöglicht.

Die Auswertung von *call-* bzw. *request*-Aufrufen innerhalb von Attributmodellen führt zur Aktivierung eines Meßwerkzeugs (bei call) oder einer Kommunikation zwischen Benutzer und MVP-System mittels electronic mail. Im ersten Fall ist die Interaktion mit dem Benutzer durch das jeweilige Meßwerkzeug festgelegt. Der Aufruf darf nur einen einfachen Wert zurückgeben, der mit dem Typ des Attributs kompatibel ist. Ein Beispiel für eine solche Kommunikation mit dem Benutzer ist in der nachfolgenden Abbildung dargestellt.

```
prf
Press control-g to enter menu. Press control-h for help.
File  Quit

PERSONNEL RESOURCES FORM
    Name: C. M. Lott
    Project: MVP-S                                    Date: 26 Feb 94

Section A: Total Hours spent on Project for the Week:  14

Section B: Hours by Activity
           Predesign
           Create design                  10
           Read/Review Design
           Write Code                     4
           Read/Review Code
           Test Code Units
           Debugging
           Integration Test
           Acceptance Test
           Other                                    Go on to Section C y

Create Design: Development of the system, subsystem, or components design.
    Includes development of PDL, design diagrams, etc.
```

Abb. 12: Formblatt eines durch eine call-Anweisung aktivierten Meßwerkzeugs

Im zweiten Fall, dem request, wird dem Benutzer ein vom MVP-S-Client generierte Mail zugeschickt, die ihn auffordert einen Wert für das entsprechende Attribut einzutragen. Ein Beispiel dafür ist in Abbildung 13 dargestellt. Der Benutzer schickt die beantwortete Mail an den MVP-S-Client zurück, der diese Nachricht dann interpretiert. Nachdem die Mail gescannt wurde, wird der Typ auf Zulässigkeit überprüft und bei erfolgreicher Prüfung als neuer Attri-

butwert eingetragen. Während der asynchronen Kommunikation mit dem Benutzer erhält das Attribut einen undefinierten Wert. Anfragen nach diesem Wert werden verzögert.

```
(Message inbox:36)
Date: Mon, 5 Sep 94 14:18:42 MET DST
From: mvp-s-server@informatik.uni-kl.de
To:   lott@informatik.uni-kl.de
Subject: mvps-request

                        MVP-S-REQUEST
                        -------------

Date    : 19940909
To      : lott
Project : Testplan
Role    : Manager
Authorisation: 2f8a835

Message:
------------------------------------------------------------------------
Please fill in the value for the requested attribute,
restrictung your changes to the field below, and return this
form to the address
mvp-s-server@informatik

Please enter value for attribute 'test_effort'
of type 'int'
that is used in process 'Testplan'

HERE->___________

Thank you!
verlage@kriza[~] ▉
```

Abb. 13: Durch einen request verschickte Mail mit der Bitte um Bereitstellung eines Attributwertes

Die momentane Lösung verlangt für jeden einzelnen Attributwert eine separate Mail. Es muß jedoch noch weiter erforscht werden, ob requests gesammelt werden können, um sie dem Benutzer anschließend als eine Anfrage zu präsentieren. Dies würde die Kommunikation zwischen MVP-S-Clients und Benutzern erheblich vereinfachen.

Das MVP-System bietet bereits heute eine ausreichende Grundlage für die Erprobung maßbasierter Ansätze im Zusammenspiel mit prozeß-sensitiven Software-Entwicklungsumgebungen. Welche methodischen und technischen Erweiterungen an MVP-S vorgenommen werden müssen, wird im nächsten Abschnitt erläutert.

6 Schwerpunkte im MVP-Projekt

Die vorherigen Abschnitte haben einen Eindruck davon vermittelt, wie Entwicklungsumgebungen zur Unterstützung qualitätsorientierter Projektpläne in drei Stufen genutzt werden können. Sie versprechen eine Verbesserung der Qualität von Prozessen und Produkten. Trotz der Vielzahl von Forschungsprojekten in den USA, Europa und Japan existieren noch wesentliche offene Fragen auf diesem Gebiet [9]. Entsprechend unserem Kenntnisstand haben wir uns entschlossen, nachfolgende Schwerpunkte im Rahmen des MVP-Projekts zu untersuchen:

- *Modellierung eines Projekts aus unterschiedlichen Blickwinkeln*
  Verschiedene Projekt-Mitglieder haben unterschiedliche Interessen. Demzufolge betrachten sie nur die für sie relevanten Teile und Aspekte eines Projekts oder dessen Produkte. Diese Sichten müssen zu einem integrierten Modell zusammengefaßt werden, das einer späteren Wiederverwendung zugänglich gemacht werden muß [26]. Das Ziel dieses Forschungsschwerpunktes ist es, die natürliche Modellierung von Prozessen durch wiederverwendbare Komponenten zu ermöglichen. Daraus ergeben sich u.a. folgende Forschungsfragen:
  - Welche unterschiedlichen Notationen sind für die unterschiedlichen Projektmitglieder sinnvoll?
  - Wie können die selben Prozeßelemente in verschiedenen Sichten identifiziert werden?
  - Welches Beschreibungsverfahren ist für eine multi-perspektivische Beschreibung am geeignetsten?
  - Wie muß eine wiederverwendungsorientierte Organisation von Modellen aussehen?

- *Analysemethoden für Projektpläne*
  Ein expliziter Projektplan ist die Grundlage für die Beantwortung von Fragen interessierter Personen. Analysen können zu erwartende Eigenschaften des Projekts erkennen lassen, noch bevor mit der eigentlichen Projektabwicklung begonnen wird. Analysen von Projektplänen sind eine wichtige Planungshilfe. So wird zum Beispiel die Abschätzung des Ressourcenbedarfs in Abhängigkeit von der Projektdauer unterstützt. Risiken, insbesondere bei Änderungen des Projektplans während der Projektabwicklung, lassen sich durch Analysen frühzeitig erkennen. Das Ziel dieses Forschungsschwerpunktes ist u.a. die Beantwortung folgender Forschungsfragen:
  - Was sind charakteristische, insbesondere für Planung und Risikoabschätzung interessante Analysefragen?
  - Was sind geeignete Analysetechniken für die Beantwortung solcher Fragen?
  - Wie ist der Zusammenhang zwischen dem eigentlichen Informationsbedarf und geeigneten Analysetechniken, bzw. wie wird eine passende Analysetechnik ausgewählt und angewendet?

- *Anleitung von Entwicklern auf der Basis von Projektplänen*
  Messen und Bewerten von Prozessen und Produkten ist fundamentaler Bestandteil quali-

tätsorientierter Software-Entwicklung. Eine verläßliche, automatische Sammlung der Daten während der Projektabwicklung ist daher von höchstem Interesse. Das Ziel dieses Forschungsschwerpunktes ist es, das zielorientierte Messen und Bewerten der Prozesse und Produkte eines Projekts zu unterstützen und das Projekt entsprechend der gesammelten Daten zu steuern. Daraus ergeben sich u.a. folgende Forschungsfragen:

- Wie muß eine formale Spezifikation von Bewertungszielen erfolgen?
- Welche Mechanismen werden für die Datenerfassung und -analyse benötigt?
- Wie können verschiedene Qualitätsmodelle integriert und für die Datenerfassung instrumentiert werden?
- Wie kann die Koordination von Entwicklern auf der Basis meßbarer Attribute erfolgen?
- Wie kann die Konsistenz des Modells mit der realen Welt verfolgt und beibehalten werden?

Das MVP-Projekt zeichnet sich gerade durch die Integration dieser einzelnen Forschungsschwerpunkte aus. Dadurch soll ein Instrumentarium geschaffen werden, mit dem die ingenieurmäßige, qualitätsorientierte Entwicklung und Betreuung von Software nachvollziehbar und geplant erfolgen kann.

7 Zusammenfassung

Die heutigen Probleme bei der Entwicklung und Wartung von Software sind zum Teil durch eine ungenügende intellektuelle Kontrolle der verwendeten Prozesse begründet. Oft entziehen sich die einzelnen Tätigkeiten in diesen Bereichen den Maßnahmen einer Qualitätssicherung, da sie nur implizit vorhanden sind. Ebenso muß eine schrittweise Qualitätsverbesserung (z.B. Ansätze nach dem Quality Improvement Paradigm) den momentanen *state-of-the-practice* erfassen, um sinnvolle Verbesserungsvorschläge und -maßnahmen vorzunehmen.

Die Quantifizierung der relevanten Eigenschaften eines Software-Projekts und die Modellierung existierender Software-Prozesse bietet eine Möglichkeit, um Qualitätsmodelle und explizite Repräsentationen für unterschiedliche Zielsetzungen zu gewinnen. Das MVP-Projekt im speziellen zielt dabei auf eine maßorientierte Projektabwicklung. Dazu wird der Goal/Question/Metric-Ansatz benützt, um die kontextabhängigen Metriken zu ermitteln.

In dieser Arbeit wurden Meßpläne und die Grundelemente eines Software-Prozeßplans in MVP vorgestellt. Aus diesen Typen wird der Arbeitsbereich einer Rolle zusammengestellt. Danach wurde erläutert, wie eine Rolle in einem Projekt unterstützt wird. Wir haben drei unterschiedliche Arten der Beziehung zwischen Rollen und Software-Prozeß-Modellen erläuter, nämlich die unterschiedlichen Arten der Nutzung dieser Modelle in Projekten. Diese können schrittweise dazu genutzt werden, um intellektuelle Kontrolle über Software-Projekte zu erlangen und auszubauen.

Zuerst ist eine explizite Repräsentation der Vorgehensweisen in einer Umgebung notwendig. Darauf aufbauend kann eine quantitative Bewertung der Produkt- und Projektzustände durch Messungen erfolgen. Letztendlich ist die automatische Ermittlung dieser Werte sinnvoll, falls diese zu einer maßorientierten Abwicklung von Software-Projekten genutzt werden sollen. Jede dieser drei Stufen kann dazu benutzt werden, um die Rollen in einem Software-Projekt zu unterstützen und Qualitätssicherung zu betreiben. Unterschiedlich ist dabei das Ausmaß der Integration, und somit der Einfluß auf die Software-entwickelnde Organisation. Daraus resultieren natürlich Konsequenzen für die Projekt-Organisation im Umfang des Nutzens im laufenden Projekt. Sie kann mit zunhemnder Integration höheren Nutzen für die Prozeß- und Qualitätskontrolle ziehen.

Bisher wurden von den Mitgliedern der Arbeitsgruppe Software Engineering vor allem Entwicklungen für die erste Stufe einer maßorientierten Projektabwicklung vorgenommen. Exemplarische Untersuchungen in realen Umgebungen haben dabei wertvolle Beiträge zur Verbesserung der eingesetzten Techniken beigetragen und verschafften einen vertieften Einblick in die Natur praktischer Software-Entwicklungen. Diese Erfahrungen werden schon heute genutzt, um die bisher existierenden Werkzeuge für die weiteren Stufen einer maßorientierten Projektabwicklung zu verbessern und sie in einer integrierten Projektumgebung einzubinden.

8 Literatur

[1] Victor R. Basili, "The Experience Factory and its Relationship to Other Improvement Paradigms", Proceedings European Software Engineering Conference, Garmisch Partenkirchen, 1993, pp. 68-83

[2] Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach, "Goal Quastion Metric Paradigm", in: John C. Marciniak (Ed.), Encyclopedia of Software Engineering, John Wiley & Sons, 1994, pp. 528-532

[3] Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach, "Experience Factory", in: John C. Marciniak (Ed.), Encyclopedia of Software Engineering, John Wiley & Sons, 1994, pp. 469-476

[4] Bröckers, Alfred; Lott, Christopher M.; Rombach, H. Dieter; Verlage, Martin, "MVP-L Language Report", Interner Bericht 229/92, Fachbereich Informatik, Universität Kaiserslautern, Dezember 1992

[5] BOOTSTRAP consortium (ESPRIT 5441), "BOOTSTRAP - Measuring Software Management Capabilities. First Findings in Europe", Proc. of the 4th IFAC/IFIP Workshop, Schloss Seggau, Österreich, Mai 1992

[6] Basili, V.R. und Rombach, H.D., "The TAME Project: Towards Improvement-Oriented Software Environments", IEEE Transactions on Software Engineering, vol. 16, no. 6, (Jun. 1988), pp. 758-773

[7] Victor R. Basili, Richard W. Selby, and David H. Hutchens, "Experimentation in Software Engineering", IEEE Transactions on Software Engineering, vol. 12, no. 7, Juli 1986, pp. 733-743

[8] Cap Gemini Innovation, "Process Weaver - User's Manual", Version PW1.2, 1992

[9] Bill Curtis, Marc I. Kellner, Jim Over, "Process Modeling", Communications of the ACM, vol. 35, no. 9, September 1992, pp. 75-90

[10] W. Edwards Deming, "Out of the Crisis", MIT Center for Advanced Engineering Study, MIT Press, Cambridge MA, 1986

[11] Norman E. Fenton, "Software Metrics - A Rigorous Approach", Chapman & Hall, London, Great Britain, 1991

[12] Alfonso Fuggetta and Carlo Ghezzi, "State of the Art and Open Issues in process-centered Software Engineering Environments", Jounal of Systems and Software, vol. 26, no. 1, July 1994, pp. 53-60

[13] Peter H. Feiler, Watts S. Humphrey, "Software Process Development and Enactment: Concepts and Definitions", Technical Report CMU/SEI-92-TR-04, Software Engineering Institute, Carnegie Mellon University, September 1992

[14] Watts S. Humphrey, "Managing the Software Process", Addison-Wesley, 1989

[15] Kellner, Marc I. "Multiple-Paradigm Approaches for Software Process Modeling", Proceedings of the 7th International Software Process Workshop, 1991

[16] Carol D. Klingler, Melissa Neviaser, Ann Marmor-Squires, Christopher M. Lott, H. Dieter Rombach, "A Case Study in Process Representation Using MVP-L", Proceedings COMPASS, June 1992

[17] Christopher M. Lott "Process and measurment support in SEEs", ACM Software Engineering Notes, vol. 18, no. 4, October 1993, pp. 83-93

[18] Christopher M. Lott, H. Dieter Rombach "Measurement-Based Guidance Using Explicit Project Plans", Journal of Information and Software Technology, Butterworth-Heinemann, 1993

[19] Maria H. Penedo and William Riddle, "Process-sensitive SEE Architecture (PSEEA) - Workshop Summary", ACM SIGSOFT Software Engineering Notes, Juli 1993

[20] Proceedings of the 5th International Software Process Workshop, Kennebunkport, USA, October 1989

[21] Proceedings of the 6th International Software Process Workshop, Hakodate, Japan, October 1990

[22] Proceedings 2nd Int. Conferences on the Software Process, Berlin, Germany, Februar 1993, IEEE CS Press

[23] H. Dieter Rombach, Victor R. Basili, "Quantitative Software-Qualitätssicherung", Informatik Spektrum, Nr. 10, 1987, pp. 145-158

[24] Rombach, H.D.; Basili, V.R.; Selby, R., "Experimental Software Engineering Issues", Tagungsband International Workshop, Schloss Dagstuhl, September 1992, Springer LNCS

[25] H. Dieter Rombach, Martin Verlage, "How to Assess a Software Process Modeling Formalism from a Project Member's Point of View", Proceedings Second International Conference on the Software Process, Berlin, Germany, February 1993

[26] Martin Verlage, "Multi-View Modeling of Software Processes", Proceedings 4th European Workshop on Software Process Technology, Grenoble, France, February 1994, Springer LNCS 772, 1994