TU Clausthal

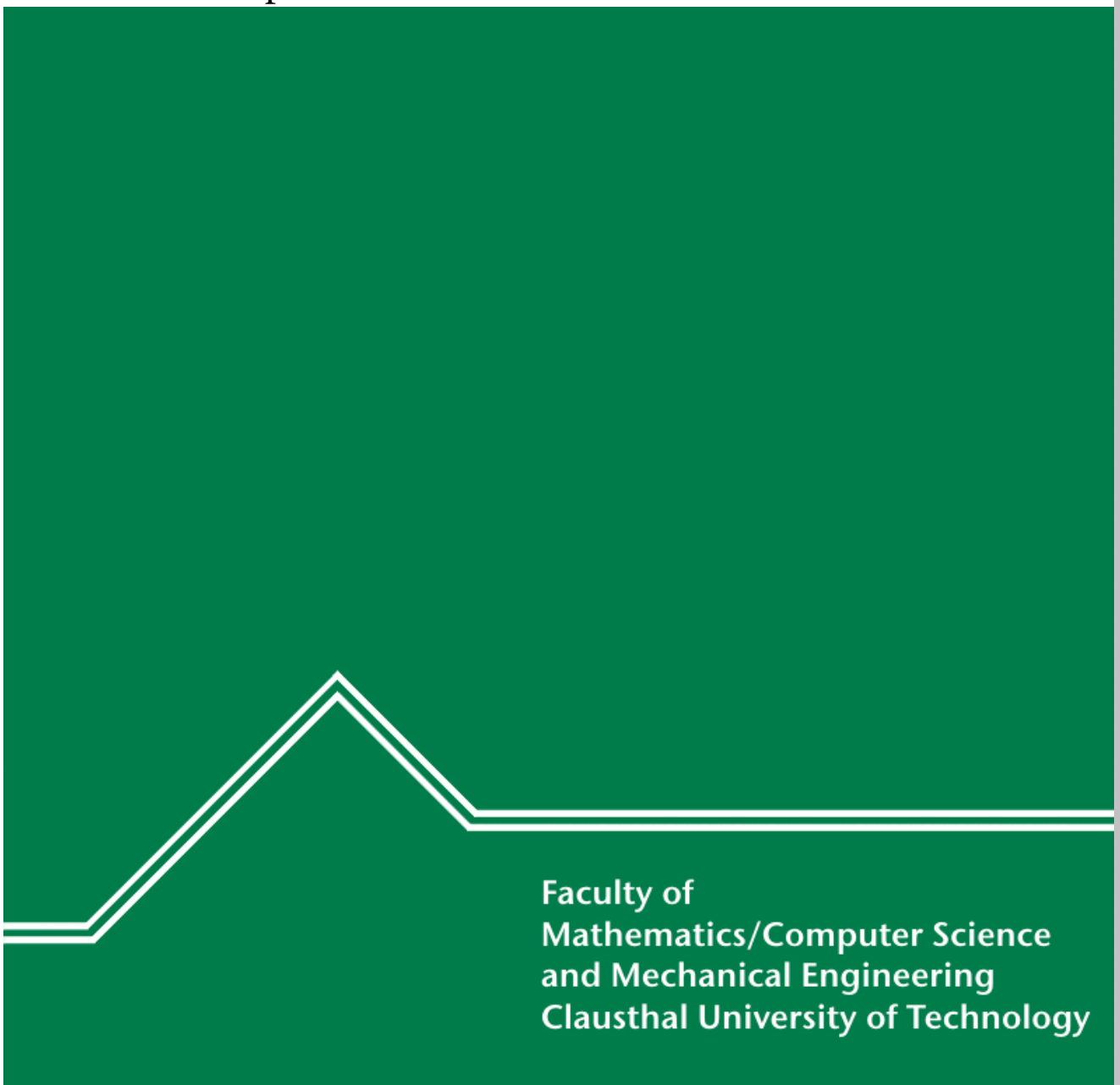
Clausthal University of Technology

Softwareprototyp zur Evaluation eines flexiblen Prozessmanagements in modularen, integrierten Entwicklungsumgebungen

N. Müller, J. Langenbach

Technical Report Series

Fac3-13-01



Impressum

Publisher: Fakultät für Mathematik/Informatik und Maschinenbau,
Technische Universität Clausthal
Leibnizstraße 32, 38678 Clausthal-Zellerfeld, Germany
Editor-in-chief: Alfons Esderts
Technical editor: Martina Wächter
Contact: martina.waechter@tu-clausthal.de
URL: http://www.fakultaet3.tu-clausthal.de/forschung/technical-reports/
ISSN: 1869-8018

The Faculty of Mathematics/Computer Science and Mechanical Engineering Review Board

Softwareprototyp zur Evaluation eines flexiblen Prozessmanagements in modularen, integrierten Entwicklungsumgebungen

Prof. Dr.-Ing. N. Müller
Dipl.-Ing. J. Langenbach

TU Clausthal - Institut für Maschinenwesen
Robert-Koch-Str. 32
D-38678 Clausthal-Zellerfeld
+495323/72-2271

Zusammenfassung

Die Bedeutung eines optimalen Produktentwicklungsprozesses wird zunehmend erkannt. Demgegenüber steht eine lückenhafte Softwareunterstützung in den frühen Phasen des Prozesses. Gerade in der Konzeptphase ist dies ein großes Problem, da hier viele Produkteigenschaften festgelegt werden. Methodisches Vorgehen ist daher genauso wichtig, wie die nachvollziehbare Dokumentation der Ideen und Entscheidungen. Am Lehrstuhl Rechnerintegrierte Produktentwicklung des Institutes für Maschinenwesen (IMW) der TU Clausthal, wird daher ein flexibler Ansatz zur Prozessunterstützung erforscht. Dieser Ansatz basiert auf klassifizierten Werkzeugen und ermöglicht damit eine automatische Prozessnetzgenerierung. Dabei wird die Datenweitergabe optimal unterstützt sowie automatisch eine Dokumentation des Ablaufes in Form eines Entwicklungsprotokolls erstellt.

The commercial relevance of the development process is increasingly recognized. Nevertheless only a few software tools support the engineer during those stages. This is a big problem since most product properties are defined within the concept phase. Therefore a systematic approach is as important as the documentation of the found ideas and made decisions. As a solution the members of the professorship for computer aided product developement at the Institute of Mechanical Engineering developed a new, flexible process management. It is based on classified tools and with it allows automatic generated process maps. The transmission of the produced data between the tools and a documentation of the choosen process is also done automatically.

1 Einleitung

Die Bedeutung der Produktentwicklung rückt immer mehr in den Fokus. Ein Grund liegt unter anderem in der bereits sehr weit fortgeschrittenen Rationalisierung der Produktion. Hinzu kommt das steigende Bewusstsein, dass die Attraktivität der hiesigen Produkte zum großen Teil durch eine gelungene Produktentwicklung garantiert wird [1]. Vor diesem Hintergrund wird deutlich, dass die optimale Beherrschung des Produktentwicklungsprozesses eine wesentliche Unternehmenskompetenz ist. Dies drückt sich auch in der Meinung der Industrie aus, die nach [2] einen zunehmenden Methodeneinsatz in der Produktentwicklung erwartet, um die bestehenden Schwächen in der Methodenkompetenz auszugleichen und die verfügbaren Innovationspotentiale optimal zu nutzen.

Zur Methodenanwendung kann im Bereich des Maschinenbaus aus einem großen Fundus an methodischen Werkzeugen ausgewählt werden. Auch die Gliederung des Prozesses an sich ist innerhalb der VDI Richtlinie 2221 [3] (s. Abbildung 1) dokumentiert und einige Werkzeuge durch [4] in diesen Ablauf eingeordnet. Bei Untersuchungen zu den Auswirkungen des methodischen Entwicklungsprozesses auf dessen Ergebnisse konnte Bender in [5] nachweisen, dass ein methodischer Prozess hilft, besonders schlechte Ergebnisse zu vermeiden. Ein starrer Prozess führt allerdings bei erfahrenen Entwicklern auch dazu, dass besonders gute Lösungen häufig ebenfalls nicht entstehen. Es ist daher nötig, den Prozess so flexibel umzusetzen, dass besonders schlechte Lösungen vermieden, aber besonders gute, gefördert werden. Trotz dieser Erkenntnis, dass ein methodischer Prozess das Entwicklungsrisiko senkt, ist in der Praxis die Anwendung des Prozesses und der Methoden nur in begrenztem Umfang erfolgt, wenn gleich die Erkenntnisse schon seit längerer Zeit zur Verfügung stehen. Ein wichtiger Grund hierfür ist die Auffassung vieler Entwickler, dass der methodische Prozess ein hohes Maß an Mehrarbeit mit sich bringt, für die im Alltag keine Zeit zur Verfügung steht. Damit wird deutlich, dass nicht nur die Anwendung des Prozesses ein wichtiger Aspekt ist, sondern ebenso die Effizienz der spezifischen Umsetzung des Prozesses.

Die Effizienz des methodischen Produktentwicklungsprozesses ist bislang jedoch eher ein wenig beachtetes Thema. Dies zeigt sich auch in der Eigenanalyse aktuell verfügbarer EDV-Systeme (z. B. CAD-, FEM- und PDM-Systeme) im Bereich des Produktentwicklungsprozesses. Eingeordnet in das Schema nach VDI 2221 nimmt die Rechnerunterstützung beginnend mit Phase 3 rasch zu. Dies besonders durch den verbreiteten Einsatz der CAD- und FEM-Systeme. In Phase 1 sind immerhin noch einige spezialisierte Softwarewerkzeuge für die Anforderungsverwaltung vorzufinden. In Phase 2 sind hingegen kaum Anwendungen zu finden. In Anbetracht der Tatsache, dass in dieser Phase jedoch die Mehrzahl der Produkteigenschaften definiert werden und es somit eine sehr bedeutende Phase der Entwicklung ist, überrascht diese Erkenntnis. Auf der anderen Seite ergibt sich hierdurch die Chance, die Effizienz des Prozesses zu steigern und damit eventuell eine höhere Akzeptanz in der Praxis zu erreichen.

2 Lösungsbasis integrierte Entwicklungsumgebungen

Ein Lösungsansatz sind integrierte Entwicklungsumgebungen (IDE), wie sie 1989 bereits von Feldhusen in [6] vorgeschlagen wurde. Jedoch konnte in der ferneren Vergangenheit mit diesem Ansatz, selbst mit Großprojekten wie iViP [7] kein Durchbruch erzielt werden. Dennoch bieten IDEs zwei wichtige Vorteile. Zum einen setzt eine Prozesssteuerung eine Zentrale voraus, an der die Informationen über den aktuellen Stand vorliegen. Zum anderen kann diese Zentrale genutzt werden, um Informationen zwischen den einzelnen Arbeitsschritten weiterzuleiten. Gerade durch die Vernetzung der jeweiligen Werkzeuge und die damit vermiedene doppelte Datenhaltung, können erhebliche Effizienzsteigerungen erzielt werden. Gleichzeitig ist aber auch mit einer Qualitätssteigerung zu rechnen, da die Datenkonsistenz zwischen den Arbeitsschritten gewährleistet werden kann.

Diese Vorteile versuchen sich auch einige aktuelle Projekte zu nutze zu machen. So verfolgen unter anderem Airbus mit Toolkit in Open Source for Critical Applications & Systems Development (TOPCASED) [8], das Deutsche Zentrum für Luft- und Raumfahrt e. V. (DLR) mit RCE Chameleon [9] oder auch das Verbundprojekt Open Model-Driven Whole-Product Development and Simulation Environment (OPENPROD) [10] den IDE Ansatz. Allerdings richten sich diese Systeme auf den Bereich der mechatronischen Systeme bzw. den Flugzeugbau aus. Einen allgemeinen Ansatz zur flexiblen Steuerung des Produktentwicklungsprozesses bieten sie daher nicht.

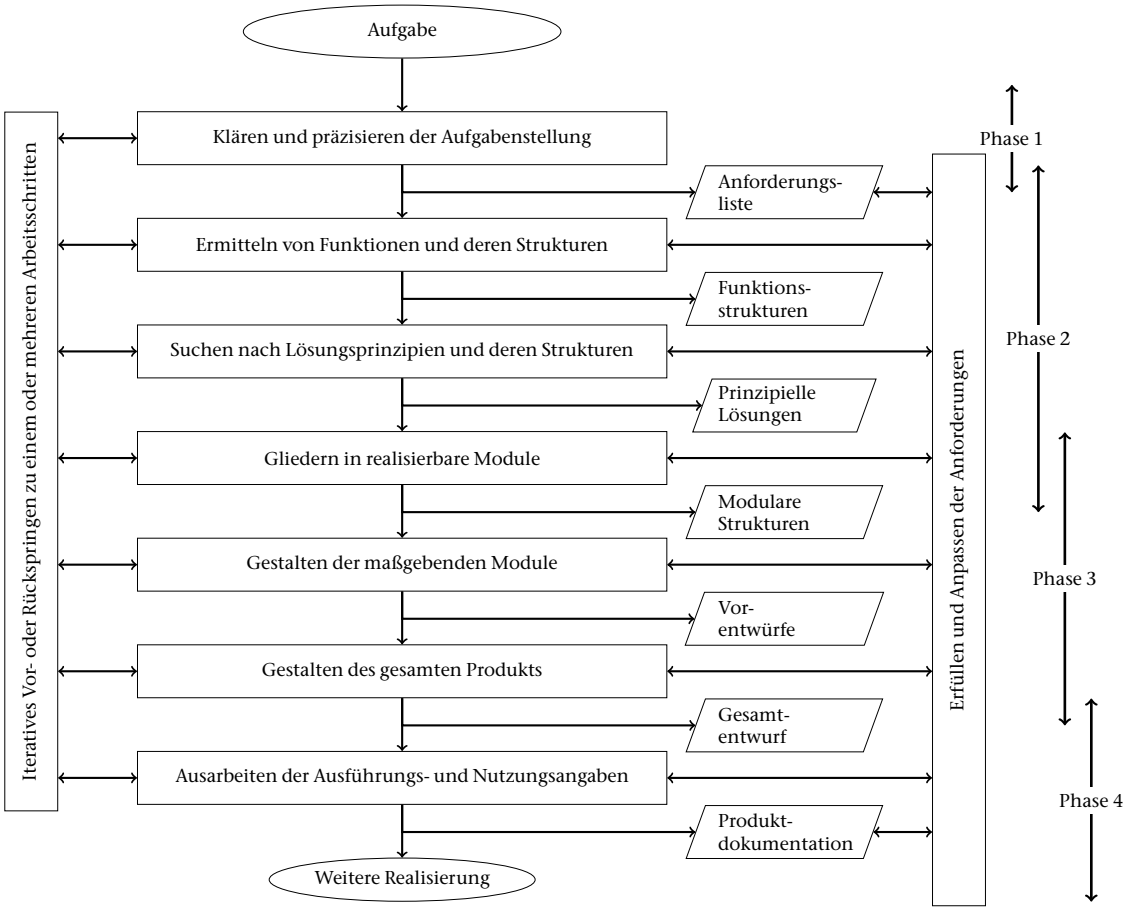


Abbildung 1: Entwicklungsprozess nach VDI 2221

3 Automatische Prozessnetze mit klassifizierten Werkzeugen

Daher wurde am Institut für Maschinenwesen der IDE-Gedanke, im Hinblick auf eine flexible Prozesssteuerung, weiterentwickelt. Die Grundlage bildet dabei die Idee, dass jeder Arbeitsschritt wie eine Black-Box betrachtet werden kann. Dabei werden Daten zur Weiterverarbeitung benötigt und mit Hilfe zusätzlicher Daten in ein Ergebnis transformiert. Ein Beispiel hierfür zeigt Abbildung 2 anhand des Arbeitsschrittes „Kombinieren der Wirkprinzipien zur Wirkstruktur“ aus Phase 2 nach VDI 2221. Weiterführend wurde diese Betrachtungsweise auch auf die Methoden eines jeden Arbeitsschrittes ausgedehnt. Wie bereits in Abbildung 2 dargestellt, trifft die spezifische Zusammenstellung aus Eingangs- und Ausgangsdaten eben nicht nur auf den Arbeitsschritt „Kombinieren der Wirkprinzipien“ zu, sondern auch auf die Methoden, die für diesen Arbeitsschritt herangezogen werden können. In diesem Fall zum Beispiel der morphologische Kasten. Im Umkehrschluss entsteht damit die erste grundlegende These für diesen Lösungsansatz: Ein Arbeitsschritt definiert sich durch eine spezifische Signatur aus Eingangs- und Ausgangsdaten seiner Werkzeuge. Trifft dies zu, können demnach vorhandene Werkzeuge durch die Analyse ihrer Eingangs- und Ausgangsdaten zu Arbeitsschritten zusammengefasst oder klassifiziert werden, ohne das vorher ein solcher Arbeitsschritt, explizit, definiert wäre. Ein Arbeitsschritt entsteht also dynamisch durch die Zusammenfassung von Werkzeugen zu Klassen anhand ihrer Eingangs- und Ausgangsdaten.

Die zweite grundlegende These besteht nun darin, dass die aus der Analyse der Werkzeuge entstandenen Klassen, zu einem Prozess zusammengesetzt werden können, in dem man passende Eingangs- und Ausgangsdaten verbindet. Sind beide Thesen gültig, lässt sich so automatisch ein Prozessnetz generieren, welches nur durch die vorhandenen Werkzeuge definiert ist. Ferner ist bei einer softwaretechnischen Umsetzung dieses Ansatzes immer gegeben, dass die Daten zwischen den jeweils folgenden Arbeitsschritten weitergereicht werden können. In Abbildung 3 ist ein Beispiel für die

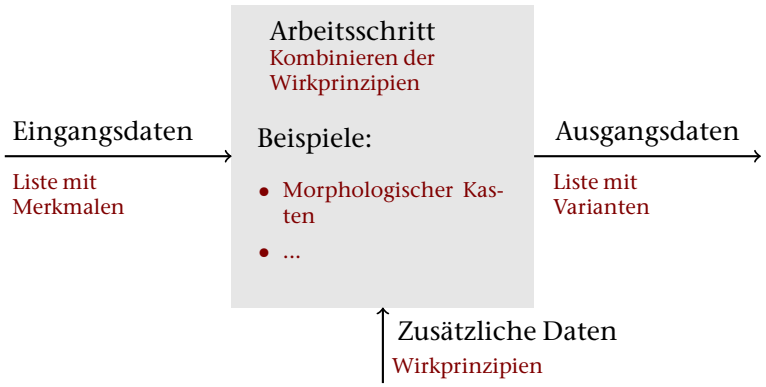


Abbildung 2: Arbeitsschritte und Werkzeuge als Black-Box am Beispiel „Kombinieren der Wirkprinzipien“

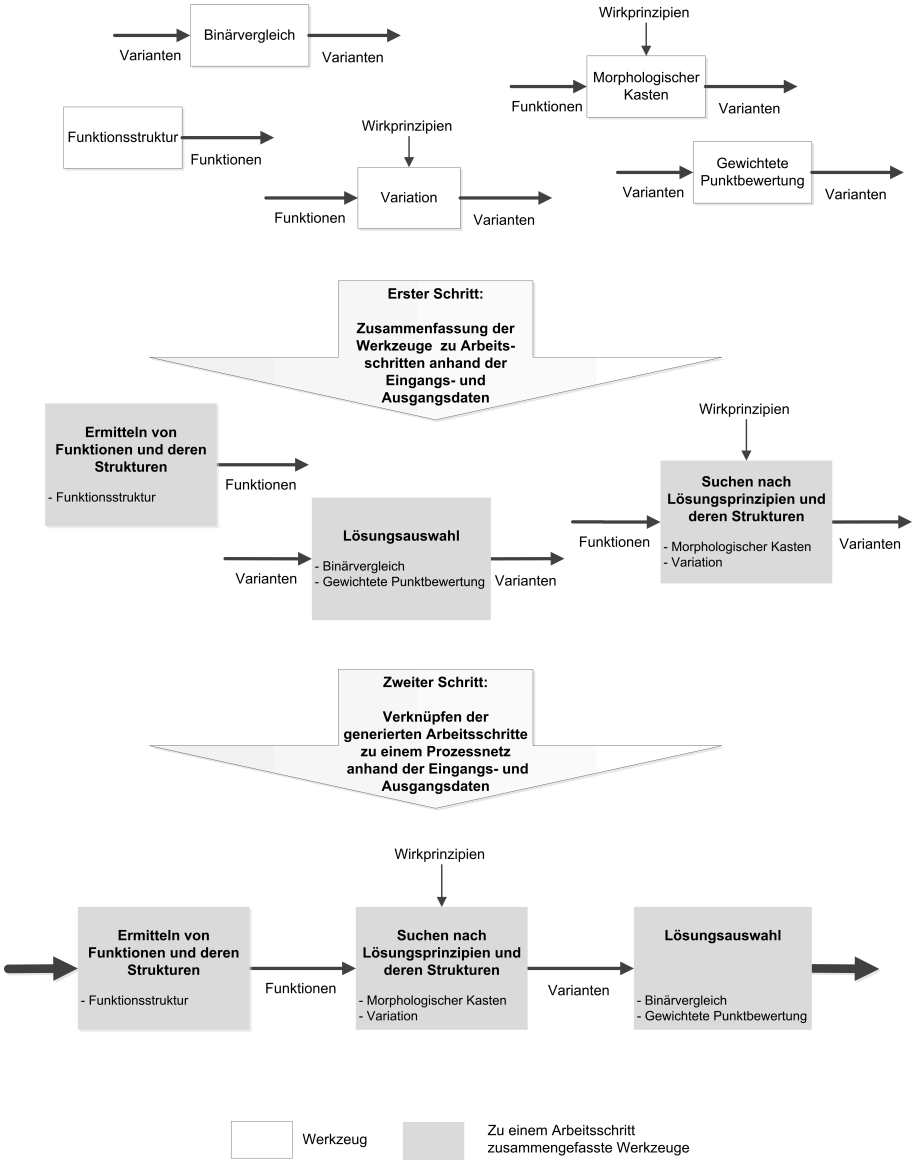


Abbildung 3: Für die automatische Generierung des Prozessnetzes, werden zunächst die Werkzeuge zu Arbeitsschritten zusammengefasst. In einem zweiten Schritt erfolgt die Verbindung der Arbeitsschritte zum Prozess.

Vorgehensweise dargestellt. Ausgehend von den jeweiligen Werkzeugen wurden Klassen gebildet, die einen Arbeitsschritt repräsentieren. Bei diesem Schritt werden Werkzeuge, die eine identische Signatur wie eine bereits vorhandene Klasse aufweisen, dieser zugeordnet. So wird die Variation sowie der morphologische Kasten in einer Klasse zusammengefasst. Analoges gilt für die Bewertungsverfahren. Anschließend werden die Klassen zu einem Gesamtprozess verbunden.

4 Softwareprototyp

Für die erste Evaluation dieses Konzeptes wurde am Institut für Maschinenwesen eine Softwareumgebung als Prototyp implementiert. Dieser besteht, wie in Abbildung 4 dargestellt, aus drei Schichten und nutzt die Qt-Bibliothek für C++. Die unterste Schicht bildet eine eigene Bibliothek mit gemeinsam genutzten Elementen. Darauf aufbauend finden sich in der zweiten Schicht die Plugins, die jeweils ein Werkzeug aus dem Entwicklungsprozess abbilden. In der obersten Ebene findet sich schließlich die Hauptanwendung, welche eine grafische Bedienoberfläche bereitstellt und die Module in diese integriert. Die einzelnen Werkzeuge können ferner auf weiteren Bibliotheken oder Anwendungen basieren, womit auch externe Anwendungen in die Entwicklungsumgebung integriert werden können.

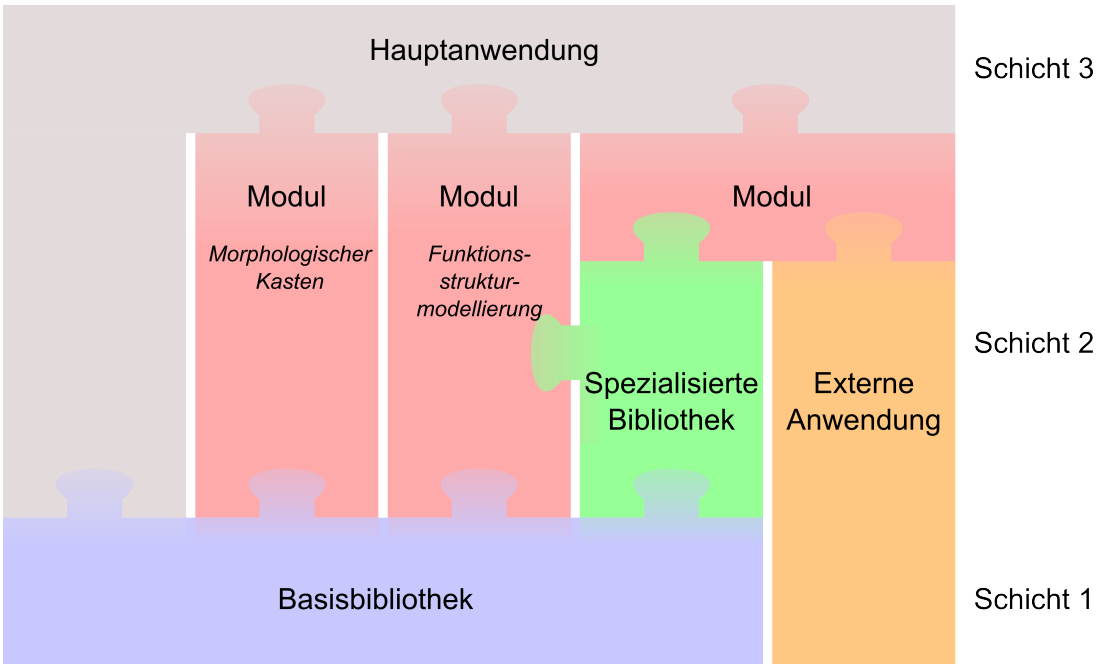


Abbildung 4: Architektur des Softwareprototyps

4.1 Inhalte der Basisbibliothek

Die Basisbibliothek enthält zunächst alle Elemente, die sowohl die Hauptanwendung, als auch einzelne Module nutzen. Dies sind grafische Bedienelemente, die von Modulen sowie der Hauptanwendung genutzt werden, um dem Anwender eine einheitliche grafische Benutzeroberfläche zu bieten. Ebenso sind aber auch Datentypen enthalten, die von vielen Modulen genutzt werden, wie zum Beispiel ein Modell zur Verwaltung von Varianten. Die wichtigste Form enthaltener Datentypen sind aber die Schnittstellen, wie die Definition der abstrakten Modulklasse. Jedes Modul implementiert diese Schnittstelle und wird darüber in die Hauptanwendung integriert. Da die intuitive Bedienbarkeit des Systems einen hohen Stellenwert hat, wird hierbei ein ähnlicher Ansatz verfolgt, wie bei den KParts genannten Komponenten innerhalb der KDE Platform. Diese KParts bilden eigenständige, abgschlossene Module innerhalb der KDE-Desktopumgebung, die jeweils eine bestimmte

Funktionalität und die KPart-Schnittstelle implementieren. Dadurch kann jedes Part als eigenständige Anwendung ausgeführt werden, aber auch von jedem anderen Part oder jeder anderen Anwendung, nahtlos in die eigene Benutzeroberfläche integriert werden. Mit Hilfe der Schnittstelle, kann ein KPart zum Beispiel die Menüstruktur und die Symbolleisten der aufrufenden Anwendung durch eigene Bedienelemente erweitern. Gegenüber einem Common Object Request Broker Architecture (CORBA) basierten Ansatz hat dies nach [11] den Vorteil, dass die Bedienung der Anwendung flüssig und einheitlich bleibt. Ein weiterer Vorteil ist die Einfachheit dieses Konzeptes, wodurch die Einarbeitung leicht möglich ist und schnell eigene Module realisiert werden können. Nachteilig ist allerdings, dass alle Module in der Programmiersprache der Hauptanwendung implementiert sein müssen.

4.2 Die Werkzeugschicht

Jedes Werkzeug des Entwicklungsprozesses wird durch ein Modul bereitgestellt. In Kombination mit der gewählten Schnittstelle für Module, ist die Erweiterung der Anwendung mit wenig Aufwand möglich. Gleichzeitig bildet durch diese Kapselung jedes Werkzeug aus Sicht der Anwendung automatisch eine Black-Box. Dadurch ist die Grundlage für den oben genannten Ansatz des Prozessmanagements bereits gelegt.

Ferner ist in dieser Schicht dafür Sorge zu tragen, dass externe Anwendungen eingebunden werden können, obwohl alle Module in C++ implementiert sein müssen. Hierfür können, wie in Abbildung 4 rechts dargestellt, Proxymodule vorgesehen werden, welche die Kommunikation zur und von der externen Anwendung übernehmen. So könnte ein Modul zur Integration eines CAD-Systems, die erforderlichen CAD-Dateien bereitstellen und in der eigentlichen CAD-Anwendung laden. Nach Beendigung der Arbeiten können die Daten an das nächste Werkzeug übergeben und die CAD-Anwendung geschlossen werden. Diese Vorgehensweise ist in Abbildung 4 ganz rechts dargestellt.

4.3 Aufgaben und Inhalte der Hauptanwendung

Die Hauptanwendung bietet lediglich drei Funktionalitäten. Die erste ist eine minimale grafische Benutzeroberfläche, die aus einem Hauptfenster sowie einem Gerüst für eine Menüstruktur besteht. Die zweite Funktionalität ist das Laden der vorhandenen Module. Während dieses Vorganges werden die Module auf Kompatibilität geprüft sowie die jeweiligen Menüelemente in das Menügerüst integriert. Die Führung des Nutzers durch den Prozess und die vorhandenen Module bildet die dritte Aufgabe der Hauptanwendung.

Das Beispiel aus Abbildung 5 zeigt schließlich Beispielhaft den Arbeitsablauf für einen Benutzer:

1. Wahl des Werkzeuges zur Durchführung des gewünschten Schrittes in der Hauptanwendung

   In diesem Fall wird das Modul „MorphBox“ zur Erstellung von Varianten mit Hilfe eines morphologischen Kastens gewählt.

2. Ausführen der Arbeiten im Modul

   Der Nutzer füllt den morphologischen Kasten mit Merkmalen und Wirkprinzipien. Anschließend generiert das Modul daraus alle möglichen Varianten.

3. Weiterverarbeitung der Ergebnisse

   Als Abschluss der Arbeiten können der morphologische Kasten sowie die Varianten gespeichert werden. Die Varianten können aber auch an alle Module weitergereicht werden, die Varianten als Eingangsgröße akzeptieren. Im Beispiel kann der Nutzer aus den zwei Modulen „CompBinary“ und „CompWePo“ wählen. Ersteres hilft bei der Erstellung von Binärvergleichen, während letzteres gewichtete Punktbewertungen unterstützt.

4.4 Das Prozessmanagementsystem

Elemente des Prozessmanagements sind in der Basisbibliothek sowie in der Hauptanwendung enthalten. In der Basisbibliothek ist die Schnittstelle für Workflowobjekte definiert, die jedes Modul,

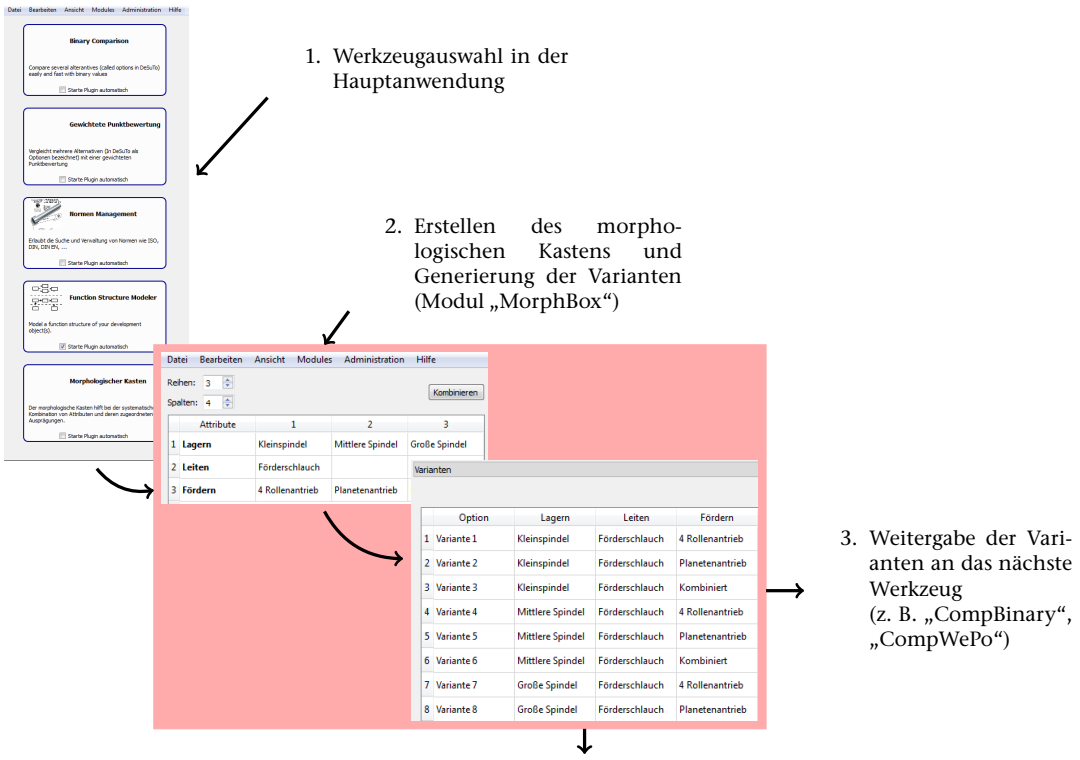


Abbildung 5: Beispielhafter Ablauf in der Anwendung

das in den Prozess integriert werden soll, implementieren muss. Dabei wird definiert, welche Funktionen das Modul für die Dateneingabe und -ausgabe bereithält und welchen Datentyp diese Funktionen für die Ein- bzw. Ausgabedaten nutzen. Zusätzlich ist in der Basisbibliothek auch die Prozessengine enthalten. Dies hat den Vorteil, dass auf die Engine aus der Hauptanwendung zugegriffen werden kann, aber auch die Verlagerung in eine Serverapplikation möglich ist.

Bei jedem Start der Anwendung wird schließlich die Klassifikation der Werkzeugmodule nach dem im Abschnitt 3 beschriebenen Schema durchgeführt. Die dabei generierten Klassen werden mit den dazugehörigen Werkzeugmodulen an die Prozessengine übergeben, die daraus ein Prozessnetz generiert. Das so erstellte Prozessnetz wird in der Hauptanwendung visualisiert und erlaubt dem Benutzer auch die Navigation. Seinen aktuellen Stand im Prozess verdeutlicht ein rotes Kästchen. Wechselt er das Modul, wird automatisch ein Eintrag im Entwicklungsprotokoll vorgenommen, den der Nutzer durch einen Kommentar ergänzen kann. So ist es möglich, im Verlauf der Entwicklung den beschrittenen Weg nachzuvollziehen.

5 Weitergehende Untersuchungen

Neben dieser ersten Evaluation des Konzeptes mit Hilfe des Softwareprototypen sind noch weitere Fragestellungen zu beantworten. So stellt sich die Frage, ob der gesamte Prozess auf diese Weise abgebildet werden kann. Eventuell sind hierfür auch neue Elemente wie Konverter oder Verbindungselemente nötig, die aktuell noch nicht vorhanden sind. Bei der Abbildung des Prozesses müssen ferner insbesondere Querschnittsaufgaben, wie die Verwaltung der Anforderungen oder die Sicherstellung der Rechtskonformität beachtet und integriert werden. Zur weiteren Untersuchung dieser Fragestellungen, ist zunächst eine systematische Erweiterung und anschließende Datenflussanalyse des Entwicklungsprozesses nötig. Darauf aufbauend müssen weitere Module implementiert werden, um die praktische Umsetzbarkeit zu gewährleisten. Zentrale Module bilden hierbei die Anforderungsverwal-

tung sowie Module aus dem Bereich des Normen- und Vorschriftsmanagements. Die Möglichkeit zur Integration von Wissensspeichern ist ebenfalls zu untersuchen. Insgesamt ermöglicht diese Realisierung auch die Evaluation des Ansatzes in Hinblick auf die Bedienbarkeit und den generierten Nutzen. Abschließend ist die Übertragbarkeit auf andere Prozesse zu untersuchen.

6 Zusammenfassung

Zur Lösung der mangelnden Rechnerunterstützung in den frühen Phasen der Produktentwicklung, wird der Ansatz der integrierten Entwicklungsumgebungen aufgegriffen. Für das Prozessmanagement innerhalb dieser IDE wurde ein neuer Ansatz entwickelt, der auf den spezifischen Eingangs- und Ausgangsgrößen der verfügbaren Werkzeuge aufbaut. Anhand dieser Signatur werden die Werkzeuge klassifiziert und anschließend zu einem Prozessnetz verknüpft. Die aktuelle Herausforderung besteht darin, diesen Ansatz auf den gesamten Entwicklungsprozess auszudehnen und insbesondere die Querschnittsaufgaben zu integrieren. Auch die Übertragbarkeit auf andere Prozesse bedarf einer anschließenden Untersuchung.

Literatur

[1] Ulrich Sendler und Volker Wawer. *Von PDM zu PLM: Prozessoptimierung durch Integration*. 3. Aufl. München: Hanser, 2011.

[2] Frank-Lothar Krause, Hans-Joachim Franke und Jürgen Gausemeier. *Innovationspotenziale in der Produktentwicklung*. München: Hanser, 2007.

[3] *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. Richtlinie. Verband Deutscher Ingenieure (VDI).

[4] G. Pahl u. a. *Konstruktionslehre: Methoden und Anwendung*. 7. Aufl. Berlin: Springer, 2007.

[5] Bernd Bender. „Erfolgreiche individuelle Vorgehensstrategien in frühen Phasen der Produktentwicklung“. Dissertation. Technische Universität Berlin, 2004.

[6] Jörg Feldhusen. „Systemkonzept für die durchgängige und flexible Rechnerunterstützung des Konstruktionsprozesses“. Dissertation. Technische Universität Berlin, 1989.

[7] Frank-Lothar Krause, Trac Tang und Ulrich Ahle. *Leitprojekt integrierte Virtuelle Produktentstehung*. Abschlussbericht. Fraunhofer - IPK Berlin, Bereich Virtuelle Produktentwicklung, 2002.

[8] Oliver Diedrich. *Airbus: Systementwicklung mit Open Source*. `http://www.heise.de/open/artikel/Airbus-Systementwicklung-mit-Open-Source-222027.html`. zuletzt abgerufen am 16.12.2012. 2006.

[9] Mathias Huber. *RCE Chameleon: Open-Source-Software entwirft Flugzeuge*. `http://www.linux-magazin.de/NEWS/RCE-Chameleon-Open-Source-Software-entwirft-Flugzeuge`. zuletzt abgerufen am 25.07.2012. 2012.

[10] *Verbundprojekt: OPENPROD - Open Model-Driven Whole Product Development and Simulation Environment*. `http://www.kooperation-international.de/detail/info/verbundprojekt-openprod-open-model-driven-whole-product-development-and-simulation-environment-1.html`. zuletzt abgerufen am 16.12.2012.

[11] David Faure. *Coding with KParts - Understanding the KParts component architecture*. `https://www.ibm.com/developerworks/library/l-kparts/`. zuletzt abgerufen am 16.12.2012. 2002.