MBEES 2015
MBEES 2015
MBEES 2015
MBEES 2015
MBEES 2015

Tagungsband des Dagstuhl-Workshops

Modellbasierte Entwicklung eingebetteter Systeme XI

Matthias Riebisch
Michaela Huhn
Jan Philipps
Bernhard Schätz


Tagungsband

Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme XI

Model-Based Development of Embedded Systems
22.03.2015 – 25.03.2015

fortiss GmbH
Guerickestr. 25
80805 München

Integration von Virtueller Inbetriebnahme und Variantenmanagement

Johannes H. Möck, Jens Weiland

Reutlinger Research Institute
Hochschule Reutlingen
Alteburgstraße 150
72762 Reutlingen
johannes.moeck@reutlingen-university.de
jens.weiland@ reutlingen-university.de

Abstract: Im Maschinen- und Anlagenbau wird im Kontext der Virtuellen Inbetriebnahme (VIBN) ein reales Produkt anhand virtueller Modelle abgebildet und simuliert. Durch die Simulation dieser Modelle kann vor der tatsächlichen Fertigstellung des realen Produktes die benötigte Steuerungssoftware entwickelt und gegen die virtuellen Modelle getestet werden. Die VIBN resultiert somit neben einer beschleunigten Produkteinführungszeit auch in einer qualitativ ausgereifteren Steuerungssoftware. Bei der Betrachtung von variantenreichen Maschinen oder Anlagen entsteht je nach Simulationsumfang, -fokus und/oder -domäne eine Reihe von Modellen, welche sich in verschiedenen Simulationswerkzeugen wiederfinden. Dabei gilt es die unterschiedlichen Simulationsmodelle, wie auch die dazu passende Steuerungssoftware inhaltlich konsistent auf die gewünschte Variante zu konfigurieren, damit nach der Konfiguration ein reibungsloses Zusammenspiel zwischen Steuerungssoftware und Simulationsmodellen gewährleistet werden kann. Im Rahmen dieses Artikels werden Konzepte aufgezeigt, wie variantenreiche Simulationsmodelle und die dazu gehörige Steuerungssoftware hinsichtlich einer konkreten Variante konsistent konfiguriert werden können. Die hierfür notwendige Varianteninfrastruktur, in welcher die unterschiedlichen Werkzeuge interagieren, wird beschrieben und eine mögliche Umsetzung aufgezeigt.

1 Einführung

Die Wettbewerbsfähigkeit im Maschinen- und Anlagenbau basiert zu entscheidenden Teilen auf der Fähigkeit des Systemherstellers zeitnah eine kundenspezifische, qualitativ hochwertige aber dennoch kostengünstige Lösung anbieten zu können. Diese Prämissen lassen sich im Entwicklungsprozess der Maschine oder Anlage aber nur begrenzt gemeinsam maximieren und stehen sich ab einem bestimmten Zeitpunkt zwangsläufig als konkurrierende Charakteristika gegenüber. So liegt es beispielsweise im Ermessen des Systemherstellers wieviel Zeit und somit Kosten er in die Änderbarkeit und Übertragbarkeit seines Produktes investiert. Werden nur die vereinbarten Funktionalitäten entwickelt, oder wird bereits an dieser Stelle das Produkt im Hinblick auf zukünftige Erweiterungen bzw. die Wiederverwendung einzelner Komponenten

vorbereitet? Die hier notwendigen Entscheidungen müssen zumeist individuell getroffen werden, da diese durch die unterschiedlichen Anforderungen, Voraussetzungen und Gegebenheiten des jeweiligen Auftrages beeinflusst werden.

Angesichts dieser Zusammenhänge stellt die Virtuelle Inbetriebnahme (VIBN) im Maschinen- und Anlagenbau eine immer größere Bedeutung dar. Verspricht sie doch qualitativ bessere Steuerungssoftware, eine kürzere „Time-to-Market“ und ein kostengünstigeres Vorgehen gegenüber dem klassischen Entwicklungsprozess.

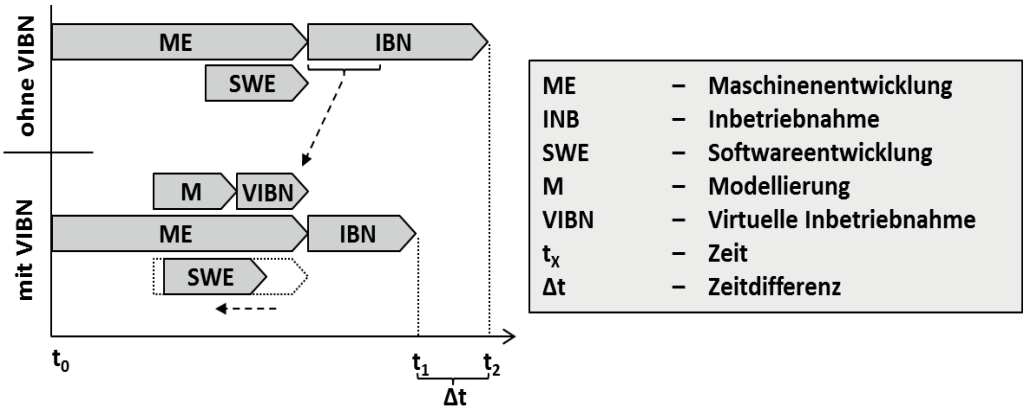

Abbildung 1: Grundidee der VIBN [Wu07]

Das Konzept der VIBN ist in Abbildung 1 dargestellt. Es sieht das Vorwegnehmen von simulierbaren Aufgaben aus der Inbetriebnahme vor. Im Maschinen- und Anlagenbau wird die Inbetriebnahme als die erstmalige bestimmungsgemäße Verwendung der Maschine oder Anlage beschrieben. [EG06] Für die VIBN wird ein virtuelles Modell erstellt, welches die reale Maschine oder Anlage abbildet und im Entwicklungsprozess mit der Simulation schon frühzeitig Verwendung findet. Parallel zur Maschinenentwicklung (Konstruktion und Montage) der realen Maschine oder Anlage können anhand des virtuellen Modells viele Arbeitsschritte und Optimierungen aus der Inbetriebnahme vorweg genommen werden. Die in der Inbetriebnahme umfangreichste Aufgabe ist das Anpassen und Testen der für den Betrieb der Maschine oder Anlage benötigten Steuerungssoftware. Da die Phase der Inbetriebnahme selbst bis zu einem Viertel der gesamten Projektlaufzeit ausmachen kann, bietet sich hier erhebliches Einsparungspotenzial. Die VIBN resultiert also in einer zeitlichen Verkürzung der Inbetriebnahme und somit in einer früheren finalen Abnahme des Produktes. Die Steuerungssoftware lässt sich durch die VIBN früher gegen das virtuelle Modell entwickeln und testen. Dadurch kann der Reifegrad der Steuerungssoftware in erheblichem Umfang gesteigert werden. Auch kann der bei der Simulation des virtuellen Modells gewonnene Wissensgewinn direkt in die Konstruktion der Maschine bzw. Anlage einfließen.

Ein Grund, warum die Simulationsmodelle noch nicht industrieweit Verwendung finden ist der Aufwand zur Anpassung der Modelle an die jeweilige Maschinenvariante. Bei einer steigenden Anzahl von zu entwickelnden Maschinen oder Anlagen entsteht für den

Entwickler, bzw. Produktverantwortlichen die Anforderung, sich sinnvoll mit der schnell steigenden Anzahl an möglichen Varianten seiner Maschinen oder Anlage auseinanderzusetzen. Eine systematische Wiederverwendung ist hierbei die entscheidende Grundlage für eine kundenindividuelle Entwicklung von Produktvarianten. Eine möglichst hohe Wiederverwendung bereits erstellter Software- und Modellelemente senkt die entstehenden Aufwände und Kosten. Zusätzlich wird durch die bereits vorhandenen und damit schon ausgereifteren Elemente die Produktqualität gesteigert und der zeitliche Umfang der Entwicklung reduziert.

Der Fokus dieses Artikels liegt auf der Präsentation einer Infrastruktur für die Handhabung von Variabilität - und der darin vorhandenen Variantenschnittstelle - im Kontext der VIBN im Maschinen- und Anlagenbau. Im Abschnitt 2 werden zunächst grundlegende Informationen über das Abbilden von Variabilität mittels Merkmalsmodellen beschrieben. Anschließend werden die für diese Arbeit notwendigen Konzepte aufgeführt. Schließlich wird im Abschnitt 3 ein Konzept vorgestellt wie eine Varianteninfrastruktur und deren Variantenschnittstelle umgesetzt werden können. Das vorgestellte Konzept wird schließlich in Kapitel 4 anhand eines Beispiels validiert. Abschnitt 5 fasst den Artikel zusammen.

2 Modellierung und Darstellung von Variabilität

Eine Möglichkeit um die Variabilität in einem System abstrakt darzustellen, ist die Modellierung von Merkmalmodellen. Mit Hilfe dieser Merkmalmodelle kann anschließend eine gültige Variante des Systems konfiguriert werden. Für die Entwicklung von Merkmalmodellen werden die variablen Elemente eines Systems in Form von Merkmalen in einer Baumstruktur abgebildet und miteinander in Beziehung gesetzt. Dies kann durch eine Zuordnung innerhalb der Hierarchie im Merkmalbaum geschehen und/oder durch das Festlegen des Variabilitätstyps der einzelnen Merkmale. Hierbei werden vier Variabilitätstypen unterschieden. Bekommt ein Merkmal den „*Pflicht*“- Variabilitätstypen, so bedeutet dies, dass die Merkmalspezifikation nur gültig ist, wenn dieses Merkmal Teil diese Spezifikation ist. Ein „*optionaler*“-Variabilitätstyp dagegen legt die Entscheidung frei, ob das Merkmal Teil der Merkmalspezifikation ist oder nicht. Die Verknüpfung zweier oder mehr Merkmale mit dem „*alternativen*“-Variabilitätstyp bedeutet, dass nur genau eines dieser Merkmale Teil einer Merkmalspezifikation sein darf. Schließlich gibt es noch den „*oder*“-Variabilitätstyp, welcher auf zwei oder mehr Merkmale angewendet werden kann. Er besagt, dass mindestens eins aber auch mehrere Merkmale für eine gültige Merkmalspezifikation ausgewählt werden kann. [ES04]

Um im Maschinen- und Anlagenbau einen möglichst modularen Aufbau aus bereits vorhandenen und getesteten Teilmodulen zu ermöglichen muss zuerst zwischen den gemeinsamen und variantenspezifischen Komponenten der Steuerungssoftware unterschieden werden. Hierfür eignet sich das Konzept des Engineering von Softwareproduktlinien, welches mit der Zielsetzung einer systematischen Wiederverwendung den Fokus auf das Unterscheiden von gemeinsamen und variablen Artefakten legt. [CN01][GS04] Das Engineering von Softwareproduktlinien geht an

dieser Stelle allerdings nicht auf die Vorgehensweise ein, wie eine Systemvariante „gebaut“ werden kann. Dies liefert das Konzept der Generativen Softwareentwicklung, welche eine automatische Generierung von Systemvarianten beschreibt. [CE00] Dabei wird zwischen der Abstraktion der Variabilität mittels Merkmalmodellen auf der einen Seite, der konkreten Implementierung der Variabilität in der Software auf der anderen Seite und dem Konfigurationswissen welches Abhängigkeiten zwischen beiden Seiten beschreibt, unterschieden.

Im Rahmen des Engineering von Softwareproduktlinien wird zwischen den variablen und den gemeinsamen Komponenten einer Software unterschieden. Um die variablen Elemente einer Software abzubilden werden Variationspunkte (VP) in der Software benötigt. Diese VP erlauben es, dass die Software hinsichtlich einer Variante konfiguriert werden kann. Für die Implementierung wird ein eindeutiger Kennzeichner benötigt, welcher den VP für die automatisierte oder manuelle Bearbeitung eindeutig erkennbar macht. Zusätzlich benötigt der VP einen Mechanismus, welcher für die konkrete Umsetzung der Konfiguration verantwortlich ist. Je nach Implementierung werden in dem VP direkt die unterschiedlichen Varianten oder die für ihre Konfiguration notwendigen Informationen gespeichert.

Derzeit bieten überhaupt nur wenige Simulationswerkzeuge wie beispielsweise Matlab/Simulink von MathWorks Methoden an um mit Variabilität im Simulationsmodell umzugehen [CM12][HKM13]. Die meisten Simulationswerkzeuge bieten kein explizites Variantenhandling an. Es muss eine generische Schnittstelle gefunden werden, wie das jeweilige Simulationsmodell in den Werkzeugen angesprochen und die darin befindlichen Variationspunkte konfiguriert werden können. Dazu bedarf es je nach vorhandener Werkzeugschnittstelle unterschiedlicher Lösungen. Aktuell existiert keine generische Variantenschnittstelle, welche ein Konzept bietet, automatisiert auf unterschiedliche Simulationswerkzeuge und die darin sich befindenden Simulationsmodelle zuzugreifen und diese gemeinsam hinsichtlich einer Variante zu konfigurieren.

3 Konzepte für eine Varianteninfrastruktur

Um im Kontext der VIBN eine automatisierte sowie werkzeugübergreifende Konfiguration über mehrere variantenreiche Simulationsmodelle und der dazugehörigen Steuerungssoftware zu ermöglichen, bedarf es einer funktionsfähigen Varianten-infrastruktur. Die hier skizzierte Varianteninfrastruktur wird für das Konfigurieren des Systems vor der darauf folgenden Simulationsphase verwendet. Die Varianten-infrastruktur besteht dabei aus einem oder mehreren Simulationswerkzeugen, welche jeweils ein oder mehrere variantenreiche Simulationsmodelle simulieren können. Die Steuerungssoftware muss hierbei mit ihrer Variabilität und Konfiguration zwingend konsistent zu den Simulationsmodellen sein. Ferner wird eine Konfigurationssoftware benötigt, welche auf der einen Seite dem Anwender der Varianteninfrastruktur die im System vorhandene Variabilität abstrakt darstellt und es ihm auf der anderen Seite erlaubt das variantenreiche System hinsichtlich seiner Vorgaben zu konfigurieren. Um den für die Konfiguration notwendigen Datenaustausch zu gewährleisten und dabei die

zwischen den einzelnen Werkzeugen vorhandenen Schnittstellen zu implementieren, bedarf es einer Variantenschnittstelle. Diese Variantenschnittstelle fungiert als Bindeglied zwischen den einzelnen Werkzeugen der Varianteninfrastruktur, ermöglicht es neue Werkzeuge auf einfache Weise in die Varianteninfrastruktur zu integrieren und sorgt für das konsistente Zusammenspiel sowie die Synchronisierung aller Teilnehmer.

Abbildung2: Varianteninfrastruktur Überblick

In Abbildung 2 wird ein Überblick über das Konzept der Varianteninfrastruktur anhand eines Simulationswerkzeuges und einer Steuerungssoftware skizziert. Auf Basis der Variantenschnittstelle kann die Varianteninfrastruktur selbst um eine beliebige Anzahl weiterer Simulationswerkzeuge und Simulationsmodelle erweitert werden. Ob die nach der Konfiguration auszuführenden Simulationen in Form einer Co-Simulation oder individuell ausgeführt werden, wird durch die Varianteninfrastruktur nicht vorgegeben.

Für eine möglichst einfache Integration der Werkzeuge in die Varianteninfrastruktur ist ein wesentliches Ziel der Variantenschnittstelle den Informationsaustausch zwischen den Werkzeugen über gängige Schnittstellentechnologien durchzuführen. Hierzu gehören z.B. das Functional Mock-up Interface (FMI) oder die Component Object Model (COM)-Schnittstelle. Daneben gibt es eine Reihe weiterer Anforderungen an die Variantenschnittstelle. So muss neben dem reinen Informationstransport innerhalb der Varianteninfrastruktur für die Synchronisation aller Infrastrukturteilnehmer auch sichergestellt werden, welches Werkzeug aktuell in die Infrastruktur eingebunden ist und dass die aktuellen Zustände dieser Modelle auch auslesbar sind. Weitere Werkzeuge und Modelle sollten sich mit nur minimalem Aufwand in die Infrastruktur integrieren lassen, ohne dass diese dafür speziell angepasst werden müssen. Die Funktionalität der Varianteninfrastruktur lässt sich auf verschiedene Aufgabenbereiche unterteilen: Für das An- oder Abmelden von Werkzeugen oder Modellen an der Varianteninfrastruktur sind administrative Funktionen notwendig. Zu den administrativen Funktionen lassen sich auch Funktionen der Synchronisation der Varianteninfrastruktur und ggf. der Netzwerkadministration zählen. Als weiterer wesentlicher Bestandteil der Varianteninfrastruktur sind die Funktionen für das Variantenhandling, welche beispielsweise für die Darstellung der Variabilität, oder das Konfigurieren einer spezifischen Steuerungssoftware-/Modellvariante in den einzelnen Werkzeugen verantwortlich sind. Daneben existieren eine Reihe weiterer Funktionen, welche die Funktionalität der Varianteninfrastruktur auf unterschiedliche Art erweitern. Beispielsweise können hier Analysefunktionen aufgeführt werden, welche sich die Varianteninfrastruktur zu Nutze machen.

Die prinzipielle Arbeitsweise der Varianteninfrastruktur soll anhand des Use Cases zur Konfiguration einer Variante von Steuerungssoftware und Simulationsmodellen über verschiedene Werkzeuge hinweg aufgezeigt werden. Abbildung 3 zeigt den strukturellen Aufbau. Der Ablauf dieses Use Cases beginnt damit, dass der Anwender sich in der Konfigurationssoftware mittels eines Merkmalsmodelles eine gültige Merkmalauswahl zusammenstellt.

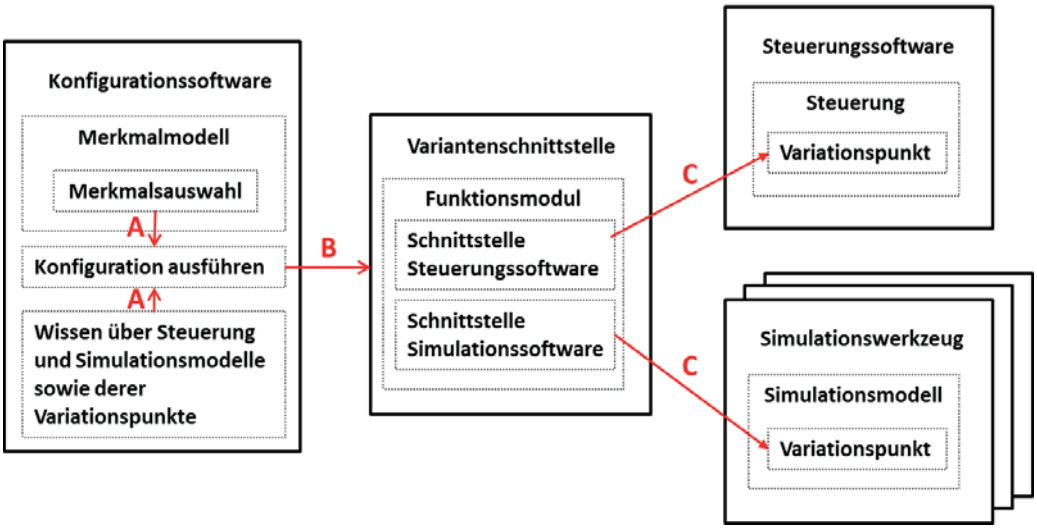

Abbildung 3: Komponenten der Varianteninfrastruktur für die Ausführung einer Konfiguration

In der Konfigurationssoftware wird die die Merkmalauswahl des Anwenders mit dem Wissen über die einzelnen Variationspunkte in den Simulationsmodellen und der Steuerungssoftware abgeglichen. In Abbildung 3 ist dies durch die zwei Pfeile mit der Kennzeichnung *A* dargestellt. Dabei entsteht die konkrete Konfiguration aller für diese Variante involvierten Variationspunkte innerhalb der Varianteninfrastruktur. Diese Informationen werden in Form einer Varianteninfrastruktur-Nachricht (VI-Nachricht) verpackt und an die Variantenschnittstelle übermittelt. Dies findet sich in Abbildung 3 an durch den Pfeil an der Position *B* wieder.

Die wesentlichen Daten, welche an dieser Stelle übermittelt werden müssen, lassen sich in zwei Teilbereiche untergliedern. Der erste Bereich bestimmt alle Metadaten dieser VI-Nachricht. Insbesondere von wem die VI-Nachricht kommt, an welches Ziel (Kombination aus Simulationswerkzeug und Modell, bzw. Steuerungssoftware und Steuerung) sie gerichtet ist und welche Funktion mit ihr ausgeführt werden soll. Zusätzlich gehören zu den Metadaten diejenigen Daten, welche für das Nachverfolgen und Handling der einzelnen VI-Nachrichten notwendig sind. Neben den aufgeführten Metadaten besteht der zweite Teilbereich der VI-Nachricht aus den Nutzdaten, welche im vorliegenden Use Case die Konfigurationen der einzelnen Variationspunkte sind. Falls eine andere Funktion der Varianteninfrastruktur ausgeführt werden sollte, so können diese Nutzdaten entsprechend der gewählten Funktion verwendet werden.

Die Varianteninfrastruktur – bestehend aus einem Werkzeug zur Merkmalmodellierung, einem Werkzeug zur Modellierung der Steuerungssoftware und verschiedenen Werkzeugen zur Entwicklung der Simulationsmodelle – kann zentral auf einem Rechner realisiert sein. In der industriellen Anwendung werden diese Werkzeuge voraussichtlich in einem Rechnernetz verteilt sein. Ein sog. *Variantenmanager*, der die Software der Varaintenschnittstelle implementiert ist verantwortlich für den konsistenten Austausch von VI-Nachrichten im Rechnernetz. Je nach Verteilung der Werkzeuge ist der Variantenmanager zentral als einzelne Softwarekomponente oder in Form von dezentralen Komponenten auf den Netzknoten realisiert.

Beim Versand der VI-Nachrichten untersucht der Variantenmanager zunächst anhand er Metadaten, ob für das angesprochene Werkzeug die notwendigen Funktionsmodule geladen sind. Ein Funktionsmodul ist ein Set von Funktionen für das Variantenhandling, welches für ein konkretes Werkzeug in seiner unterstützten Schnittstellentechnologie (siehe oben) implementiert ist. Im vorgestellten Use Case ist das passende Funktionsmodul im Speicher des Variantenmanagers geladen und kann auf die spezifizierten Werkzeuge angewendet werden. In Abbildung 3 wird dies anhand der zwei roten Pfeile mit der Markierung *C* dargestellt. Wird die Funktion erfolgreich ausgeführt, so generiert der Variantenmanager anschließend eine positive Rückmeldung. Falls der Variantenmanager kein passendes Funktionsmodul findet oder ein Fehler beim Ausführen der Funktion auftritt, so wird dies rückgemeldet.

4 Einsatz der Varianteninfrastruktur

Um das dargelegte Konzept einer Varianteninfrastruktur zu validieren wurde eine prototypische Implementierung aufgesetzt und anhand eines des in Kapitel 3 beschriebenen Use Cases ausgeführt. Dabei wurden zwei unterschiedliche Simulationswerkzeuge exemplarisch verwendet. *SimulationX* von der Firma *ITI GmbH* und *RecurDyn* von der Firma *FunctionBay Inc.* Für die Steuerungssoftware wurde *CODESYS* von *3S-Smart Software Solutions* verwendet. Als Konfigurationssoftware wurde *pure::variants* von der Firma *pure-systems GmbH* in die Infrastruktur integriert. Aktuell stehen weitere Werkzeuge zu Validierungszwecken in Planung. Als abzubildendes Modell wurde die Automatisierungslösung *SpeedPicker* der Firma *Manz AG* verwendet.

Der hier skizzierte Use Case sieht vor, dass sich in den beiden Simulationswerkzeugen je ein Simulationsmodell des variantenreichen *SpeedPickers* befindet. In *RecurDyn* soll ein kinematisches Modell für die Kollisionsberechnung simuliert werden. In *SimulationX* wird dagegen ein Verhaltensmodell des *SpeedPickers* abgebildet. Die gleiche Variabilität wie in den beiden Simulationsmodellen befindet sich auch in der *CODESYS* Steuerung. Abgebildet wird diese Variabilität in *pure::variants* mittels eines Merkmalmodells.

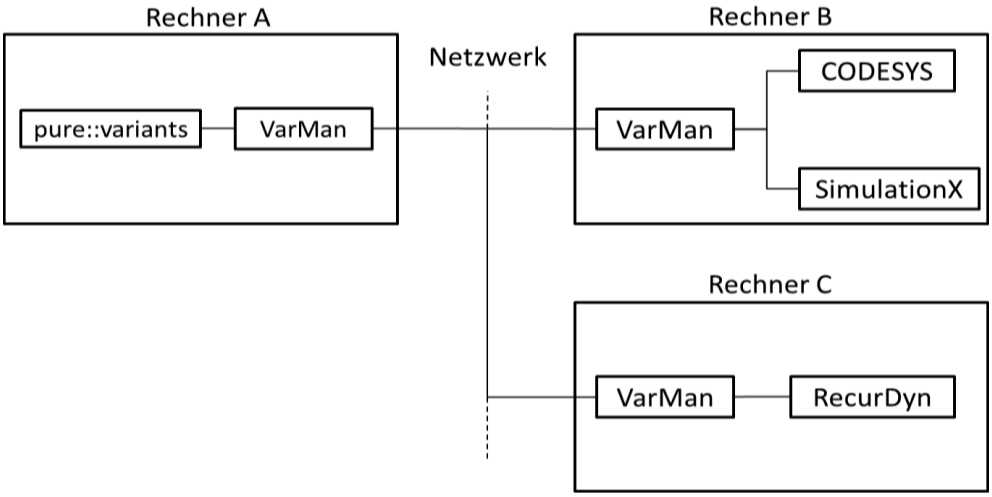

Abbildung 4: Skizzierter Aufbau einer prototypischen Implementierung der Varianteninfrastruktur mit einer dezentralen Variantenschnittstelle

Wie in Abbildung 4 beispielhaft dargestellt befinden sich die einzelnen Werkzeuge innerhalb eines Netzwerkes auf unterschiedlichen Rechnern. Exemplarisch wurden die einzelnen Werkzeuge der Varianteninfrastruktur auf drei Rechnern verteilt. Dabei wird *pure::variants* auf Rechner *A* gesetzt, *SimulationX* und *CODESYS* befinden sich dagegen zusammen auf dem Rechner *B*. Auf dem Rechner *C* befindet sich *RecurDyn*. Eine lokale Instanz des Variantenmanagers wird auf jedem der drei Rechner ausgeführt. Die Variantenschnittstelle wird mit einem dezentralen Aufbau des Variantenmanager implementiert. In *pure::variants* wird eine gültige Konfiguration des *SpeedPickers* bestimmt. Die Varianteninfrastruktur soll diese Variante in den Simulationsmodellen und der Steuerung umsetzen. Als Nachrichtenformat für den Datenaustausch innerhalb der Varianteninfrastruktur wurde eine *XML*-basierte Datenstruktur spezifiziert.

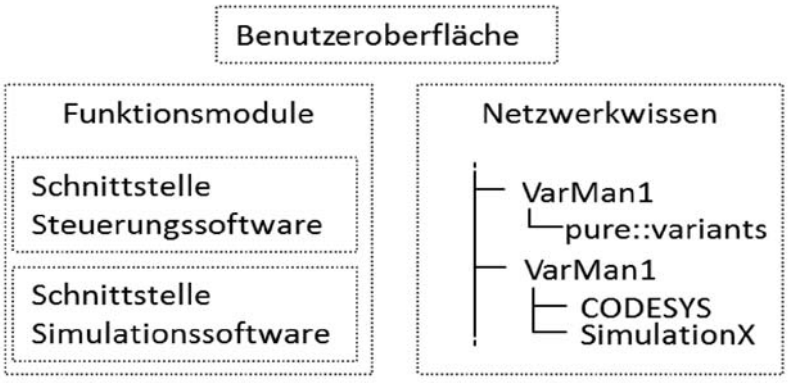

Abbildung 5: Skizzierter Aufbau des „Variantenmanagers“

Sobald alle Infrastrukturteilnehmer gestartet sind, werden sie über die Benutzeroberfläche des Variantenmanagers angemeldet. Dieses sich so aufbauende Netzwerkwissen über die Varianteninfrastruktur wird aus der Kombination aus Sobald alle Infrastrukturteilnehmer gestartet sind, werden sie über die Benutzeroberfläche des Variantenmanagers angemeldet. Dieses sich so aufbauende Netzwerkwissen über die Varianteninfrastruktur wird aus der Kombination aus Variantenmanager, Simulationswerkzeugen (bzw. Steuerungssoftware) und Simulationsmodellen (bzw. Steuerung) in einer internen Datenbank gespeichert. In Abbildung 5 ist dies durch eine Baumstruktur skizziert. Ausgehend von der Konfiguration, welche in *pure::variants* mittels eines Merkmalmodells des *SpeedPickers* erzeugt wurde, wird eine VI-Nachricht generiert und an den lokalen Variantenmanager auf Rechner *A* gesendet. Dieser verarbeitet die Nachricht und sendet sie je nach Zieladresse an den lokalen Variantenmanager auf Rechner *B* und/oder *C* weiter. Hierbei wird von dem Variantenmanager jeweils die passende Schnittstellentechnologie aus seinen internen Funktionsmodulen werkzeugspezifisch gesucht. Der Variantenmanager auf Rechner *B* verwendet für das Ansprechen der Steuerungssoftware *CODESYS* ein Python Skript in Kombination mit der FMI-Schnittstelle, welche die gewünschten Konfigurationen in den Variationspunkten innerhalb der *SpeedPicker*-Steuerung vornehmen. Für das Ansprechen des *SpeedPicker*-Verhaltensmodells in *SimulationX* verwendet der Variantenmanager ein Visual Basic Skript, welches über eine COM-Schnittstelle die gewünschten Konfigurationen ausführen kann. Auf Rechner *C* verwendet der Variantenmanager die Microsofts .Net – Technologie und den Import einer .dll um das *SpeedPicker*-Kollisionsmodell in *RecurDyn* zu konfigurieren. Die Simulationsmodelle in *SimulationX* und *RecurDyn*, sowie die dazu gehörige Steuerung in *CODESYS* sind nun konsistent auf eine Variante konfiguriert und können gemeinsam simuliert werden.

5 Zusammenfassung

Damit im Maschinen- und Anlagenbau die VIBN an variantenreichen Systemen wirtschaftlich durchgeführt werden kann, wird eine systematische Wiederverwendung von Steuerungssoftware und Simulationsmodellen benötigt. Das Engineering von Softwareproduktlinien stellt Konzepte bereit für eine systematische Wiederverwendung von Softwarekomponenten. Der Schwerpunkt liegt hierbei auf der Betrachtung von gemeinsamen und variablen Artefakten eines Systems. Das Konzept der Generativen Softwareentwicklung beschreibt hierzu, wie eine automatische Generierung von Systemvarianten erstellt werden kann. Es gibt im Rahmen der VIBN eine Reihe von Werkzeugen zur Entwicklung von Simulationsmodellen und Steuerungssoftware. Um das Engineering von Softwareproduktlinien auf die VIBN anwenden zu können braucht man zum einen Möglichkeiten zur Realisierung von Variabilität in den Werkzeugen und zum andern ist zwingend eine Infrastruktur erforderlich, in welcher diese Softwareentwicklungswerkzeuge integriert werden können. In diesem Artikel wurde eine Varianteninfrastruktur vorgestellt, die es ermöglicht unterschiedlichste Werkzeuge zur Entwicklung von Simulationsmodellen und Steuerungssoftware zu erstellen. Ein wesentliches Augenmerk liegt hierbei auf der Schnittstelle, mit denen diese Werkzeuge in die Infrastruktur integriert werden können.

Acknowledgement

Die vorgestellten Forschungsarbeiten wurden im Rahmen des Förderprojekts Virtuelle Inbetriebnahme Variantenreicher Systeme (VivaSys) erarbeitet. VivaSys wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) unter dem Förderkennzeichen „03FH085PX2“ gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Literaturverzeichnis

[CE00] Czarnecki, K.; Eisenecker, U.W.: Generative Programming – Methods, Tools, and Applications. Addison-Wesley, 2000.

[CM12] Mengi, C.: Automotive Software – Prozesse, Modelle und Variabilität, Shaker Verlag, Aachen 2012

[Cz05] Czarnecki, K.: Overview of Generative Software Development. In: J.-P. Banâtre et.al. (Edit.): Unconventional Programming Paradigms (UPP) 2004, Mont SaintMichel, France, LNCS 3566, 2005.

[CN01] Clements, P.C.; Northrop, L.: Software Product Lines – Practices and Patterns. SEI Series in Software Engineering, Addison-Wesley, 2001.

[EG06] Richtlinie 2006/42/EG des Europäischen Parlaments und des Rates vom 17. Mai 2006 über Maschinen

[ES04] Eisenecker, U.W.; Schilling, R.: Merkmalmodellierung – Konzepte, Notationen und Einsatzmöglichkeiten. In: Adelsberg, A.A.; Eicker, S.; Krcmar, H.; Palowski, J.M.; Pohl, K.; Rombach, D.; Wulf, V. (Hrsg.): Multikonferenz Wirtschaftsinformatik (MKWI) Band 1: E-Learning: Modelle, Instrumente und Erfahrungen, Software-Produktlinien, Communities in E-Business. Universität Duisburg-Essen, Akademische Verlagsanstalt Aka GmbH, 2004.

[GS04] Greenfield, J.; Short, K.: Software Factories – Assembling Applications with Patterns, Models, Frameworks, and Tools. John Wiley & Sons, 2004.

[HKM13] Arne Haber, Carsten Kolassa, Peter Manhart, Pedram Mir Seyed Nazari, Bernhard Rumpe and Ina Schaefer.: First-Class Variability Modeling in Matlab / Simulink. In Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems, New York, NY, USA, 2013. ACM

[JW08] J. Weiland: Variantenkonfiguration eingebetter Automotive Software mit Simulink, Dissertation, Wirtschaftswissenschaftliche Fakultät, Universität Leipzig, Juli 2008

[Wu07] G. Wünsch: Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme. Herbert Utz Verlag, 2008