 DOI 10.1515/auto-2013-1051 — at – Automatisierungstechnik 2014; 62(3): 163–174

Übersicht

Birgit Vogel-Heuser*, Jens Folmer und Christoph Legat

Anforderungen an die Softwareevolution in der Automatisierung des Maschinen- und Anlagenbaus

Requirements on software evolution in machine and plant automation

Zusammenfassung: Die Evolution von Software in der Automatisierungstechnik im Bereich des Maschinen- und Anlagenbaus ist stark mit der Evolution der Mechanik und Elektrotechnik verbunden. Die Herausforderungen an das Software Engineering zur Unterstützung dieser oft 15 bis 20 Jahre dauernden Entwicklung werden erläutert. Zur Detaillierung wurden im Rahmen des DFG SPP 1593 „Design for Future – Managed Software Evolution" Demonstrators Szenarien entwickelt, an denen neun SPP-Projekte ihre Ansätze erproben und evaluieren werden.

Schlüsselwörter: Software Enginering, Softwareevolution, Anforderungen, Automation, Maschinen- und Anlagenbau, Modellierung.

Abstract: Evolution of automation software for machines and plants is tightly correlated with the evolution of its mechanics and electronics. The challenges on software engineering of systems operated and evolved for 15 and 20 years is discussed in this paper. To allow the development and evaluation within the DFG SPP 1593, a demonstration scenario has been developed in order to study such evolutions.

Keywords: Software Engineering, Software Evolution, Requirements, Production Automation, Modeling.

***Korrespondenzautor: Birgit Vogel-Heuser,** Technische Universität München, Lehrstuhl für Automatisierung und Informationssysteme, Boltzmannstr. 15, 85748 Garching bei München, E-Mail: vogel-heuser@ais.mw.tum.de

Jens Folmer, Christoph Legat: Technische Universität München, Lehrstuhl für Automatisierung und Informationssysteme, Boltzmannstr. 15, 85748 Garching bei München

1 Einleitung

Der Anteil der Software im Maschinen und Anlagenbau (M&A) stieg in den letzten Jahren signifikant an: 2008 betrug der Anteil der Software im Lieferumfang einer Maschine 9%; 2012 beträgt der Anteil des Entwicklungspersonals für Software 20% in einzelnen Unternehmen bereits über 50%; für das Jahr 2015 wird ein Anstieg auf 25% mit weiterhin steigender Tendenz erwartet [1]. Folglich steigt der Einfluss von Qualität und Kosten des Software Engineering in Bezug auf eine Maschine bzw. Anlage, weshalb Wiederverwendung von Software stark an Bedeutung gewinnt. Dem Software Engineering steht häufig die Tatsache entgegen, dass Software einfach und jederzeit geändert werden kann, insbesondere da Maschinen und Anlagen oft 15 bis 20 Jahre in Betrieb sind [2]. Dies konterkariert aus pragmatischen Gründen, wie beispielsweise der geforderten schnellen Softwareänderung vor Ort aufgrund des vorhandenen Kundendrucks, häufig das systematische modellbasierte Vorgehen. Der Umgang mit Softwareänderungen während des Betriebs der Maschine oder Anlage ist eine Besonderheit in der Automatisierungstechnik des M&A. Die Mechanik kann häufig für viele Jahre unverändert betrieben werden, wohingegen die automatisierungstechnische Hardware wie Speicherpogrammierbare Steuerungen (SPS) und deren Software deutlich öfter angepasst werden müssen. Gründe hierfür können beispielsweise fehlende Ersatzteile oder Funktionsverbesserungen sein. Neben Anpassungen der Software, beispielsweise zur Optimierung, sind Softwareänderungen oft auch durch Veränderungen in anderen Disziplinen bedingt. Hieraus resultieren unterschiedliche Evolutionsfrequenzen der Disziplinen Mechanik, Elektrotechnik/Elektronik und Software (vgl. Abbildung 1). Die unterschiedliche Lebenszeit einzelner Komponenten (Abbildung 1, unten) beeinflusst diese Evolution ebenfalls. Die Verknüpfung beteiligter Disziplinen auf Modellebene untereinander erschwert die Über-

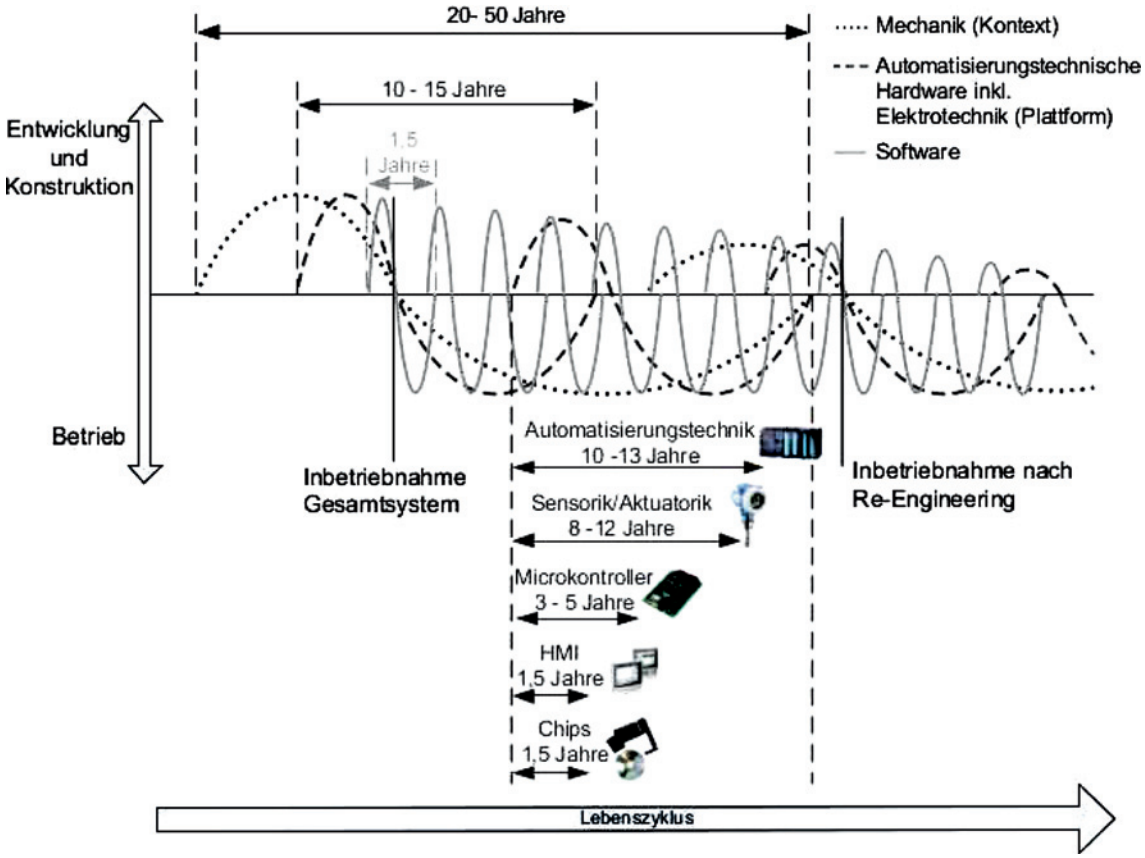

Abbildung 1: Exemplarischer Lebenszyklus im Maschinen- und Anlagenbau in Relation zu den eingesetzten Komponenten (Angaben in Anlehnung an [2]).

nahme erprobter Methoden des Software Engineering, da es sich um eine disziplinenübergreifende Engineeringaufgabe handelt.

Im Rahmen des DFG Schwerpunktprogramms (SPP) 1593 „Design for Future – Managed Software Evolution" wird ein neuer Ansatz für den langfristigen Betrieb von Software unter Einhaltung hoher Qualitäts- und Flexibilitätsanforderungen erforscht. Unter langfristigem Betrieb wird im M&A die Gewährleistung des Betriebs von bis zu 30 Jahren verstanden, bei gleichzeitig immer kürzer werdenden Evolutionszyklen der Industrie-PCs und der Standardbetriebssysteme. Die Produktionsautomatisierung stellt innerhalb des SPP einen der beiden Anwendungsbereiche und zugleich Demonstratoren, an dem neun beteiligte Projekte ihre Ansätze evaluieren.

Innerhalb des SPP wurde bewusst eine gemeinsame Terminologie eingeführt: Unter *Kontext* werden die Randbedingung der Softwaresysteme verstanden; im Falle der Produktionsautomatisierung handelt es sich um die Mechanik. Die *Plattform* beinhaltet die automatisierungstechnische Hardware inklusive der Elektrotechnik/Elektronik sowie das Betriebssystem und ggf. Systemsoftware während unter *Software* die eigentliche Steuerungsapplikationssoftware verstanden wird. Im Rahmen der Demonstratoren für den M&A wird von einer SPS-basierten Plattform (Soft- oder Hardware SPS) und den IEC 61131-3 Sprachen bzw. vorhandenen, integrierten UML-Editoren (Werkzeug CODESYS[1]) ausgegangen.

Eine der Schlüsselherausforderungen der zu erarbeitenden Ansätze des SPP, ist der Umgang mit Modularität, Varianten und Versionen. Dies bestätigt eine Erhebung der Herausforderungen an das Engineering für das Jahr 2020, in der M&A–Unternehmen aus dem Bereich der Fertigungs- und der Verfahrenstechnik Plug&Produce, Modularität, Wiederverwendung und Variantenbildung ebenso wie Versionsmanagement als Schlüsselherausforderungen genannt haben [3].

Der Beitrag ist wie folgt gegliedert: Zunächst erfolgt eine Definition der Begriffe Modul, Variante und Version. Im Anschluss werden in Abschnitt 3 die Anforderungen

1 http://de.codesys.com

an die Softwareevolution im M&A mit dem Fokus auf Varianten und Versionen und deren Bedeutung innerhalb des SPP erläutert. Der Demonstrator Produktionsautomatisierung des SPP wird in Abschnitt 4 beschrieben und die erarbeiteten Evolutionsszenarien, im Sinne der zu erfüllenden Herausforderungen, ebenso wie die verfügbare Dokumentation des Demonstrators erläutert. In Abschnitt 5 wird die Demonstrator-Simulation für die SPP-Projekte beschrieben. Die angestrebten Ergebnisse des SPP werden kurz in Abschnitt 6 erläutert. Abschnitt 7 fasst den Beitrag zusammen und gibt einen Ausblick auf weitere Forschungsfragen.

2 Begriffsklärung – Modularität, Variante und Versionen

Die Begriffe Modularität, Variante und Version haben teils sehr unterschiedliche Bedeutungen, weshalb an dieser Stelle zunächst eine Begriffsklärung erfolgt.

Modularität wird im Software Engineering nach IEEE [4] wie folgt definiert: „[Modularity is] the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components“. Demnach wird Modularität als grundlegendes Konstrukt angesehen, Änderungen mit geringer Auswirkung auszuführen.

„Eine *Variante* bezeichnet aus der Sicht des Maschinenbaus Gegenstände ähnlicher Form und/oder Funktion mit einem in der Regel hohen Anteil identischer Gruppen oder Teile“ [5], die eine „Abart einer Grundausführung“ [6] darstellt. Folglich versteht man unter einer Variante zeitlich parallel existierende, vergleichbare Ausprägungen desselben Erzeugnisses bzw. Ergebnisses. Damit sind Varianten potenziell gegeneinander austauschbar, wobei die Verwendung der Alternativen vom konkreten Anwendungsfall abhängig ist.

Im Gegensatz dazu sind *Versionen* zeitlich nacheinander entstehende, vergleichbare Arbeitsergebnisse bzw. Entwicklungsstufen einer Aufgabe oder eines Erzeugnisses. Eine neue Version ersetzt meist eine ältere Version bzw. geht durch Veränderung oder Weiterentwicklung aus dieser hervor. Version für Software wird durch IEEE in [4] als „an initial release or re-release of a computer software configuration item, associated with complete compilation or recompilation of computer software configuration item“ definiert.

Im nachfolgenden Abschnitt werden nun die Anforderungen and die Softwareevolution im M&A diskutiert.

3 Anforderungen an die Softwareevolution im M&A

Im Rahmen unterschiedlicher Untersuchungen in der Industrie wurden verschiedene Anforderungen des M&A im Bezug auf Softwareevolution identifiziert. Diese werden im Folgenden erläutert.

3.1 Varianten-/Versionsmanagement und Modularität

Eine im Rahmen des DFG Sonderforschungsbereiches SFB 768 (Teilprojekt A6) durchgeführte Umfrage bei zehn M&A-Unternehmen zeigt, dass Schnittstellen zwischen Modulen definiert werden, aber ein disziplinübergreifendes Varianten- und Versionsmanagement (VVM) sowie notwendige Kompatibilitätsprüfungen beim Austausch von Modulen kaum realisiert sind (vgl. Abbildung 2, F8). Diese Anforderungen werden im Folgenden detaillierter betrachtet.

Die Problematik im M&A entsteht durch die Notwendigkeit aus einer Variante A (VA) in der Version 1.1 eine neue Variante B (VB) zu bilden und ggf. Fehler, die in VB erkannt werden in VA in einer späteren Version wieder einzupflegen, weil der Fehler bereits in der Ursprungsvariante unbemerkt eingearbeitet wurde (vgl. Abbildung 3, A1.0), aber erst später erkannt wurde (vgl. Abbildung 3, B2.0). Analog gilt dies auch für in Variante B eingeführte Verbesserungen oder Optimierungen. Erschwerend besteht ein Modul im M&A nicht ausschließlich aus Software, sondern zusätzlich aus Plattform und Kontext, die üblicherweise nicht 1:1:1 organisiert sind [7] und somit nicht in jedem Fall mechatronische Module darstellen. Das VVM muss demzufolge disziplinübergreifende Verknüpfungen abbilden und beherrschen können.

Eine detaillierte Erhebung in einem Unternehmen des M&A mit 24 Softwareentwicklern aus der SPS-Anwendung (IEC 61131-3) zeigte, dass die dort vorhandenen, vorgefertigten Module häufig als nicht einsetzbar angesehen werden. Als Grund werden eine mangelnde Nachvollziehbarkeit sowie ein als zu groß eingeschätzter Mehraufwand durch die notwendige Abdeckung der verschiedenen Varianten genannt; solche Softwaremodule werden als zu unflexibel und zu unterschiedlich angesehen [8].

Zur Verbesserung der Modularisierung, und damit der Steigerung der Wiederverwendbarkeit, sind Analysen typischer Veränderungen in und zwischen Modulen notwendig, wie z.B. in [9] für wiederverwendbare Simulationsmodule beschrieben.

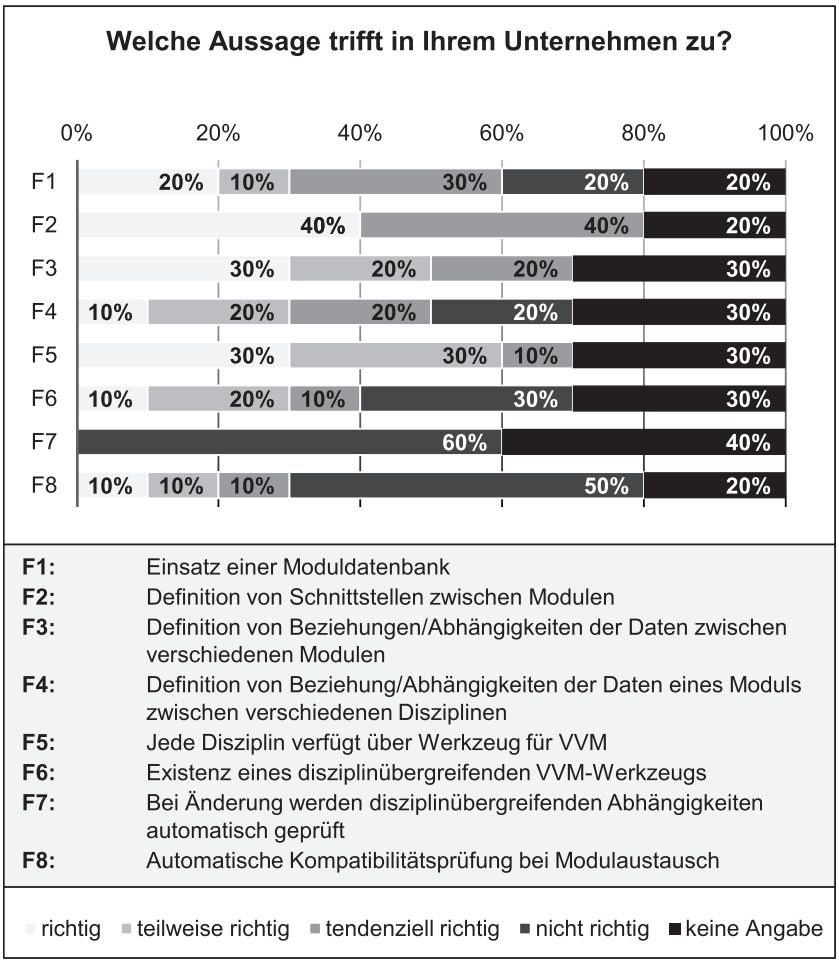

Abbildung 2: Umfrageergebnisse in zehn Unternehmen des M&A.

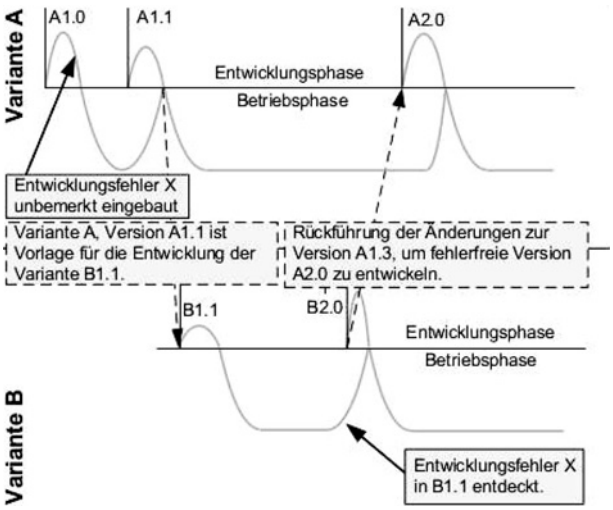

Abbildung 3: Beispielhafte Entwicklung von Variante und Version der Software im M&A.

Auf Basis der bisherigen Anforderungsermittlung ergibt sich eine Unterteilung der M&A-Unternehmen in zwei Gruppen: ein Teil der Unternehmen setzt auf mechatronische Module und strukturiert Software und Plattform strikt nach dem Kontext, bzw. gemäß einer 1:1:1 Zuordnung, während eine zweite Gruppe dies für nicht realisierbar hält. Erstere sind häufig Inhouse-M&A, die für das eigene Unternehmen bzw. den eigenen Konzern Spezialmaschinen inklusive der Automatisierung und Software liefern; letztere sind in der Regel in verschiedenen Anwendungsbereichen projektorientiert tätig. Folglich muss die Anforderung an die Projekte des SPP der allgemeinere Fall des disziplinübergreifenden VVMs ohne mechatronische Modularisierung sein, um allgemeingültige Aussagen für den M&A zu erreichen.

3.2 Änderungsmanagement und Kompatibilitätsprüfung

Wie bereits im vorangegangen Abschnitt erläutert, stellt VVM eine Herausforderung im M&A dar, die bis heute noch nicht adäquat gelöst werden konnte. Aufbauend auf einem VVM ist die Verwaltung von Änderungen des Systems, d.h. an Kontext, Plattform oder Software, und die (automatische) Prüfung deren Auswirkung, z.B. Kompatibilität, auf das Gesamtsystem (vgl. Abbildung 2) gefordert. Weitere Anforderungsanalysen in verschiedenen Branchen ergaben, dass die Berücksichtigung disziplinübergreifender Wirkbeziehungen und Restriktionen von zentraler Bedeutung sind [10].

Im Weiteren werden geplante und ungeplante Softwareänderungen unterschieden. Unter geplanten Softwareänderungen werden Änderungen verstanden, die dem Paradigma des modellbasierten Software Engineering folgen, beinhalten dabei aber auch Änderungen während des Betriebs. Ungeplante Softwareänderungen werden häufig kurzfristig während der Inbetriebnahme und des Betriebs durchgeführt, um beispielsweise Mängel in anderen Disziplinen, wie ein unerwartetes Verhalten des Kontextes, zu beheben. In Folge solch ungeplanter Änderungen wird oft die Dokumentation durchgeführter Änderungen vernachlässigt.

Schrieber *et al.* [11] stellen Kompatibilitätsanforderungen aus Sicht des Life-Cycle-Managements unterschiedlichen Kompatiblitätsgraden, d.h. Graden der Austauschbarkeit, gegenüber. Die Erfüllung dieser Anforderungen hinsichtlich spezifischer, nach Kompatibilitätsprofilen eingeordneter Geräte ermöglicht direkt die Analyse von Änderungsauswirkungen. Die Evolution der Software sowie Wechselwirkungen zwischen Kontext und Plattform stehen dabei nicht im Mittelpunkt; die vorgeschlagenen Kompatibilitätsgrade sind dafür weiterzuentwickeln. Dies bedingt eine detaillierte Analyse von Änderungen der Anforderungen, resultierender Änderungen von Systemteilen, deren Beziehung zur Evolution des Gesamtsystems und sich daraus ergebender Kompatibilitätsprobleme und ist Gegenstand der Forschung des SPP.

Die Evolution der Software wird mittels Änderungen des Kontextes sowie der Plattform im Folgenden erläutert. Typische Änderungen des Kontextes sind das Hinzufügen einer Neukonstruktion oder einer bereits existierenden Konstruktion; im einfachsten Fall eines mechanischen Teils oder auch eines Sensors oder Aktors. Diese Fälle bedingen die Implementierung eines neuen Funktionsbausteins (Abbildung 4 (1)) oder die Anpassung eines existierenden Bausteins hinsichtlich seiner Schnittstellen und Implementierung (Abbildung 4 (2)). Das Entfernen einer bereits existierenden Konstruktion kann im Hinblick auf die Software in einer Verkleinerung der Schnittstelle des Funktionsbausteins und dessen internen Struktur resultieren (Abbildung 4 (5)). Die Änderung des Verhaltens des Kontextes kann in eine Veränderung der Struktur (Abbildung 4 (3)) sowie in der Optimierung des Verhaltens des Funktionsbausteins (Abbildung 4 (4)) resultieren. Typische Änderungen der Plattform stellen die Aktualisierung einer Steuerung auf eine neuere Steuerungsgeneration sowie der Austausch einer Steuerung durch ein anderes Fabrikat aufgrund von Kundenforderungen dar. Dies kann ggf. zu komplexen Softwareanpassungen führen, falls genutzte, plattformspezifische (Software-)Funktionalität durch die Anpassung nicht mehr zur Verfügung steht. Diese komplexen Anpassungen lassen sich jedoch auf eine Kombination der zuvor beschriebenen Grundtypen von Softwareänderungen zurückführen.

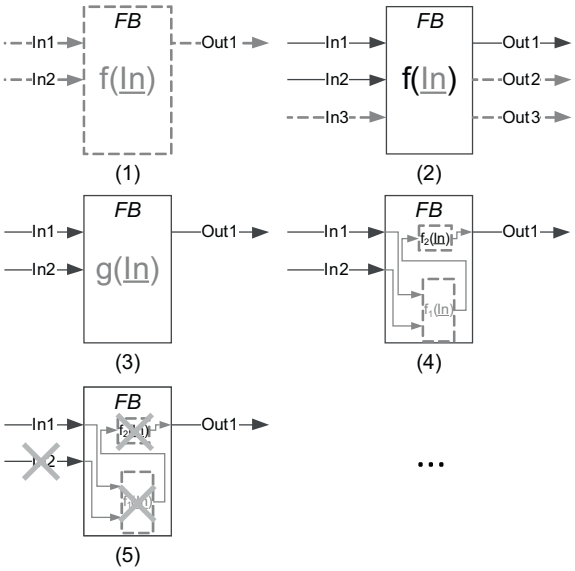

Abbildung 4: Ausschnitt typischer Änderungen von Funktionsblöcken nach IEC 61131-3 (Hinzugefügte Elemente werden gestrichelt/grau, entfernte Elemente durchgestrichen dargestellt).

Zusammenfassend bedeutet dies für das SPP, dass sowohl bei geplanten als auch bei ungeplanten Änderungen (automatische) Kompatibilitätsprüfungen aus Sicht des M&A wünschenswert sind. Hierdurch kann beispielsweise identifiziert werden, ob Änderungen der Software zu vorhandenen Kontext und Plattform (z.B. Geschwindigkeit des Bussystems bzw. Speichergröße) passen. Ferner kann geprüft werden, ob ausgetauschte Softwaremodule oder Programmteile den bis dahin funktionierenden Rest der Software bzw. der Plattform nicht beeinflussen oder stören. Eine deutlich weitergehende Vision aus der Sicht des M&A stellt die automatische Erkennung mechanischer Än-

derungen (geplanter und ungeplanter) dar, die während der Laufzeit durch die Software selbst identifiziert werden und zu einer automatischen Anpassung der Software unter Einhaltung der Funktion der Maschine oder Anlage führen [12].

4 Demonstrator Produktionsautomatisierung

Zunächst werden die Gründe für die Auswahl einer einfachen Stempel- und Sortieranlage (im Folgenden Pick & Place Unit, PPU) als Demonstrator für die Produktionsautomatisierung erläutert und anschließend die erarbeiteten Evolutionsszenarien bzw. deren Sequenzen mit ihren Änderungen von Kontext, Plattform sowie der Software diskutiert. Die von den meisten Projekten ausgewählte Sequenz von Evolutionsszenarien wird detailliert erläutert. Abschließend werden angestrebte Erkenntnisse bei der Nutzung des Demonstrators zusammengefasst.

4.1 Auswahl der Demonstratoranlage

Wesentliche Anforderungen an den Demonstrator seitens des SPP war die Möglichkeit verschiedene Kontext-, Plattform- und Softwareevolutionsaspekte einfach demonstrieren zu können. Hierbei ist eine überschaubare Komplexität notwendig, um unterschiedliche, wissenschaftliche Ansätze der Projekte unter Einbringung realistischen Aufwands darlegen zu können. Aus Sicht einiger Projekte war insbesondere die Verfügbarkeit verschiedener Softwarestände inklusive Dokumentation von Bedeutung, durch die eine zeitliche Evolution über einen längeren Zeitraum untersucht werden kann.

Diese Anforderungen erfüllt die PPU (vgl. Abbildung 5) als einfacher Demonstrator eines diskreten Fertigungssystems, da diese Anlage bereits seit 2001 in Forschung & Lehre eingesetzt wird und Software für verschiedene SPS-Typen und PC-basierte Automatisierungsgeräte erstellt wurde. Aus Sicht der Automatisierungstechnik stellt dieser Demonstrator einen einfachen Fall einer Laboranlage für die steuerungstechnische Ausbildung dar, regelungstechnische Aufgaben werden vernachlässigt.

Um die Komplexität der PPU und damit die Realitätsnähe der Ergebnisse vergleichend mit industrieller Software im M&A (vgl. Tabelle 1) beurteilen zu können, wurde die PPU sowie eine komplexere Laboranlage der Technischen Universität München (TUM), die ähnlich noch an zwei weiteren Standorten in Deutschland im Einsatz ist, mit industrieller Software verglichen. Als Maßzahlen wurden Programmzeilen (LOC), Anzahl der IEC 61131-3 Bausteine (FB[2], FC[3] bzw. OB[4]), Anzahl der Schnittstellen (Funktionsaufrufe, Datenaustausch bzw. Austausch sog. Merker) sowie Speicherverbrauch genutzt (Tabelle 1). Die industrielle Software ist um den Faktor 10 größer als die komplexe Laboranlage (das sog. Hybride Prozessmodell) und circa um den Faktor 100 größer als die PPU. Beide Laboranlagen bieten im Bereich der Logistik nahezu identische Funktionalität. Das komplexere Labormodell weist auch einfache verfahrenstechnische Prozesse auf. Im Vergleich zu den untersuchten Industrieanlagen fehlen anspruchsvolle regelungstechnische Aufgaben insbesondere Antriebsregelung bzw. Antriebssynchronisation. Folglich kann die genutzte PPU lediglich einen Ausschnitt der Auf-

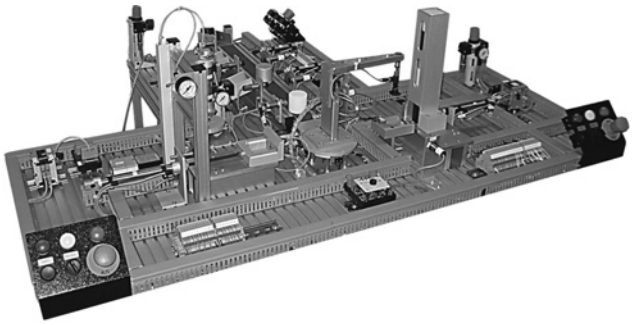

Abbildung 5: Demonstrator Produktionsautomatisierung „PPU".

Tabelle 1: Softwarekomplexität von realen Industrieanlagen in Vergleich mit dem Demonstrator PPU sowie einer komplexeren Laboranlage der TUM.

	Industrieanlagen			Laboranlagen		
Anlage / Beschreibung	Holwerkstoff (Teilanlage)	Vliesproduktion	Getränkeverpackung (Maschine)	Hybrides Prozessmodell, Verfahrenstechnik	Hybrides Prozessmodell, Logistik	Demonstrator SPP (PPU)
Bausteine	581	315	208	294	199	5
Lines of Code	332T	234.5T	170T	41.9T	24.9T	1.3T
Schnittstellen	3.2T	1.2T	?	983	461	4
Speicher [MB]	12.3	9.1	?	1.3	0.7	0.04

Legende: T – Tausend, ? – keine Aussage

2 Funktionsbaustein

3 Funktion

4 Objektbaustein

Tabelle 2: Detailinformationen der Szenarien des Demonstrators Produktionsautomatisierung mit Änderungsarten.

Auslösende Anforderung	Szenario	Relevanz	Stapel			Kran			Stempel			Rutsche			Förderband			Realisierung
			K	P	S	K	P	S	K	P	S	K	P	S	K	P	S	
Steigerung des Werkstück-Durchsatzes	0	3	A	A	A	A	A	A	-	-	-	A	A	A	-	-	-	Neuentwicklung der Maschine
Kapazitätssteigerung der Rutsche	1	3	o	o	o	o	o	o	-	-	-	M	o	o	-	-	-	Y-Form der Rutsche
Zusätzliche Bearbeitung von metallischen Werkstücken	2	5	A	A	A	o	o	o	-	-	-	o	o	o	-	-	-	Induktiver Sensor zur Metall-Detektion
Etikettierung eines Produkts	3	6	o	o	o	M	M	M	A	A	A	o	o	o	-	-	-	Hinzufügen einer Stempeleinheit
Verringerung der Fehlerrate verursacht durch Sensorverschmutzung	4a	6	o	o	o	o	M	o	o	o	o	o	o	o	-	-	-	Ersetzen der Kransensoren
Verfügbarkeitssteigerung der Kranpositionierung	4b	5	o	o	o	o	M	M	o	o	o	o	o	o	-	-	-	Sensorredundanz durch induktive und taktile Schalter
Weitere Steigerung des Werkstück-Durchsatzes	5	6	o	o	o	o	o	M	o	o	o	o	o	o	-	-	-	Optimiertes Kran-Verhalten durch die Nutzung des Stempels als Puffer
Weitere Steigerung des Werkstück-Durchsatzes durch Produktpufferung	6	5	o	o	o	M	M	M	A	o	o	o	o	o	-	-	-	Zusätzlicher mechanischer Puffer; höhere Kran-Optimierung
Erkennung eines weiteren Produkttyps	7	3	A	A	A	o	o	o	o	o	o	o	o	o	-	-	-	Installieren eines Lichtsensors zur Produkterkennung des Materials
Unterschiedliche Prozessführung für verschiedene Materialien	8	4	o	o	o	o	o	o	A	A	M	o	o	o	-	-	-	Unterschiedliche Druck-Profile an der Stempeleinheit
Logistische Optimierung durch Produktförderung	9	4	o	o	o	o	o	o	o	o	o	M	M	M	A	A	A	Ersetzen der Y-Rutsche durch ein Förderband mit Einzelrutsche
Logistische Optimierung durch weiteren Puffer am Transportband	10	6	o	o	o	o	o	o	o	o	o	o	o	o	A	A	A	Zwei zusätzliche Rutschen am Förderband
Änderung der Pufferbefüllungsstrategie	11	5	o	o	o	o	o	o	o	o	o	o	o	o	M	M	M	Sukzessives Befüllen der Rutschen
Sortenreine Sortierung der Produkte in den Puffern	12	5	o	o	o	o	o	o	o	o	o	o	o	o	o	o	M	Festgelegte Sortierung der Werkstücke
Präzisionssteigerung der Kranpositionen	13	5	o	o	o	M	A	M	o	o	o	o	o	o	o	o	o	Ersatz der digitalen Sensoren durch Analogsensor für die Kranposition
Resistenz des Krans gegen ektro-magnetische (EMV) Einflüsse	14	3	o	o	o	M	A	M	o	o	o	o	o	o	o	o	o	Installieren eines Inkrementalgebers für die Kran-Positionierung

Legende: A: hinzugefügt; M: modifiziert; -: nicht vorhanden in diesem Evolutionsschritt; o: keine Änderungen; K: Kontext; P: Plattform; S: Software

gaben der M&A Automatisierung zeigen. Auf Grund seiner geringen Komplexität ermöglicht die PPU die Untersuchungen der grundsätzlichen Anwendbarkeit verschiedener Evolutionsansätze mit akzeptablem Personal- und Zeitaufwand. Anschließend wäre die Anwendbarkeit der Ansätze für kontinuierliche Prozesse und Antriebsregelung zu zeigen, um nur einige zu nennen. Die Analyse der Softwarekomplexität des komplexeren Labormodells zeigt, dass die Art der Programmierung, z.B. SFC[5] und CFC[6], im Vergleich zu Funktionsbausteinsprache und einem modularen Ansatz eine weitere Unschärfe bzgl. der Softwarekomplexität darstellt.

4.2 Beschreibung der Funktionalität des Demonstrators und der Evolutionsschritte

In Abstimmung mit den beteiligten SPP-Projekten wurden sechzehn Szenarien ausgewählt, die unterschiedliche Kombinationen von Kontext-, Plattform- und Softwareänderungen abdecken (vgl. Tabelle 2). Die Evolutionsschritte (Szenarien) entsprechen den zeitlichen Erweiterungen der Laboranlage und wurden um weitere typische Änderungen aus realen Industrieanlagen ergänzt. Diese spiegeln die in Abschnitt 3 beschriebenen Herausforderungen wider: Unterschiedliche Kombinationen von Erweiterungen und Anpassungen von Kontext und Plattform bedingen unterschiedliche Softwareanpassungen. Dazu sind Mechanismen zur Kompatibilitätsprüfung zwischen Modulen einer Disziplin und zwischen Modulen verschiedener Disziplinen (Kontext, Platform und Software) notwendig. Unterschiedliche Ausbaustufen des Demonstrators mit ihren Varianten und Versionen ermöglichen die Untersuchung von VVM für die Automatisierung des M&A.

5 Sequential Function Chart

6 Continuous Function Chart

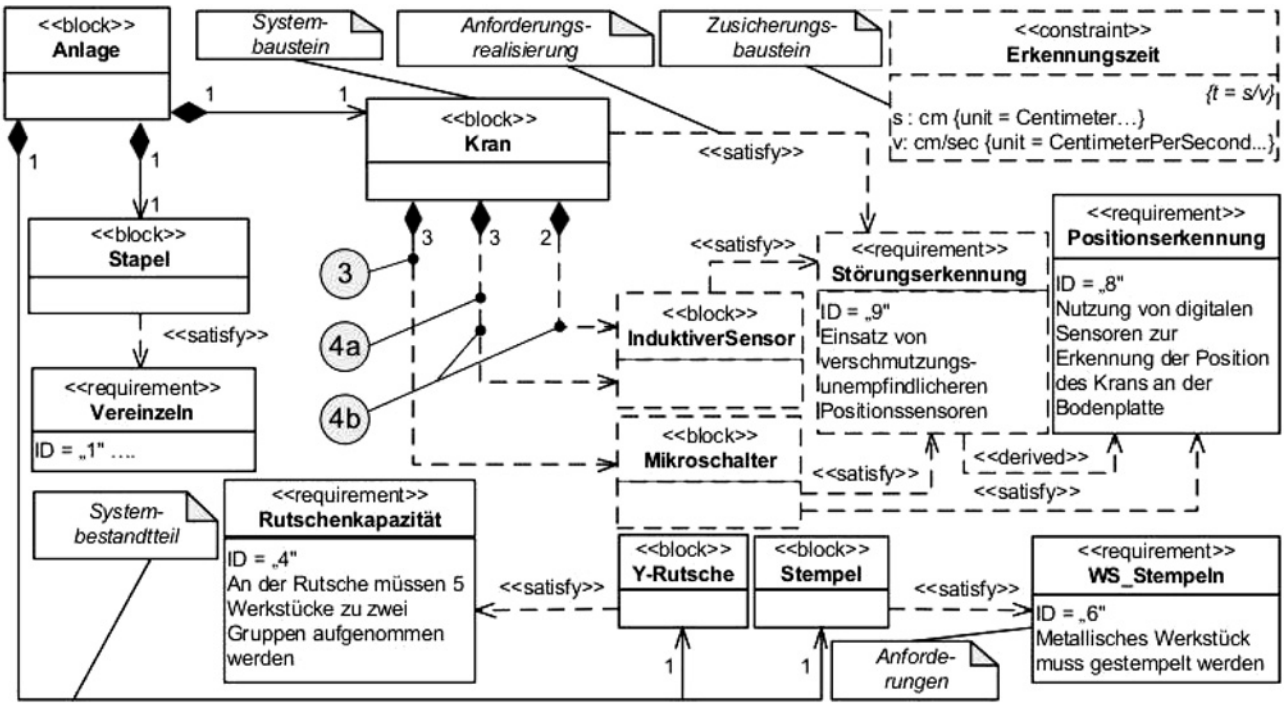

Abbildung 6: Ausschnitt des Strukturmodells des Krans der PPU von Szenario 3, 4a und 4b (Änderungen der Elemente bzgl. Szenario 3 gestrichelt). Graue Kreise mit Nummern der Evolutionsschritte geben an, in welchem Szenario welche Komposition gültig ist.

Die SPP-Projekte haben die für ihre Forschung wesentlichen Szenarien ausgewählt. Die am häufigsten ausgewählte Sequenz von Evolutionsschritten wird im Folgenden detailliert betrachtet (Anzahl der Projekte, die sich an dem Szenario beteiligen, ist in der Spalte *Relevanz* angegeben).

Zu Evolutionsbeginn (Szenario 0) besteht die PPU aus einem Stapel, einem Kran sowie einer einfachen Rutsche. Aus dem Stapelmagazin werden die Werkstücke mittels des Krans in eine Rutsche gebracht. In Szenario 1 wird aus Kapazitätsgründen die Rutsche in ihrer Form (Y-Form) verändert. Diese Änderung erfolgt lediglich im Kontext. In Szenario 2 sollen unterschiedliche Arten von Werkstücken (Plastik und Metall) bearbeitet werden können, weshalb ein induktiver Sensor zur Detektion metallischer Werkstücke hinzugefügt wird. Hierbei ist anzumerken, dass die Einordnung von Sensoren in Kontext bzw. Plattform nicht pauschal beantwortet werden kann. Ist ein Sensor rein mechanisch realisiert, wird er dem Kontext zugeordnet; handelt es sich jedoch beispielsweise um einen Sensor, der bereits eine Werteaufbereitung vornimmt oder mit einer Busschnittstelle ausgestattet ist, wird dieser sowohl dem Kontext als auch der Plattform zugeordnet. In einem nächsten Evolutionsschritt soll eines der Werkstücke zusätzlich gestempelt werden. Deshalb wird in Szenario 3 eine Stempeleinheit hinzugefügt. Während des Betriebs zeigt sich jedoch, dass die am Kran installierten Mikroschalter mit den Anlagenkennzeichen 200S6 bis 200S8 (Komponentenliste in Abbildung 7, oben) störanfällig sind. Aus diesem Grund werden in Szenario 4a diese Mikroschalter durch drei induktiven Sensoren an gleicher Position ersetzt. Die eindeutige Benennung von Komponenten bzw. Bauteilen erfolgt im Folgenden, wie im M&A üblich, anhand der Anlagenkennzeichen (AKZ).

Im Szenario 4b werden die Mikroschalter aus Szenario 3 beibehalten und um zwei induktive Sensoren (AKZ 200S9 und 200S10) ergänzt, die räumlich versetzt angeordnet werden (vgl. Abbildung 7) und damit zur Erkennung von Störungen der Mikroschalter dienen (Anforderung *Störerkennung* der Mikroschalter).

Die Szenarien 5–14 werden aus Platzgründen nicht weiter erläutert, eine Übersicht der Szenarien findet sich in [13]. Im Folgenden werden die erstellten Modelle und die Dokumentationen anhand des Evolutionsschritts des Szenario 3 zu den Szenarien 4a bzw. 4b vorgestellt.

4.3 Dokumentation der Evolutionsszenarien

Im Rahmen der modellbasierten Systementwicklung ist die Systems Modeling Language (SysML) weit verbreitet [14] und wurde gemeinsam von den Projekten zur Do-

Position	AKZ	Klemme	Beschreibung
...	...	...	...
200			Kran
...	...	...	...
200	200S6	x2.6	Mikroschalter Position Stapel
200	200S7	x2.7	Mikroschalter Position Stempel
200	200S8	x2.8	Mikroschalter Position Rutsche
200	*200S9*	*x2.9*	*Induktivsensor Stapel/Rampe*
200	*200S10*	*x2.10*	*Induktivsensor Rampe/Stempel*
...	...	...	...

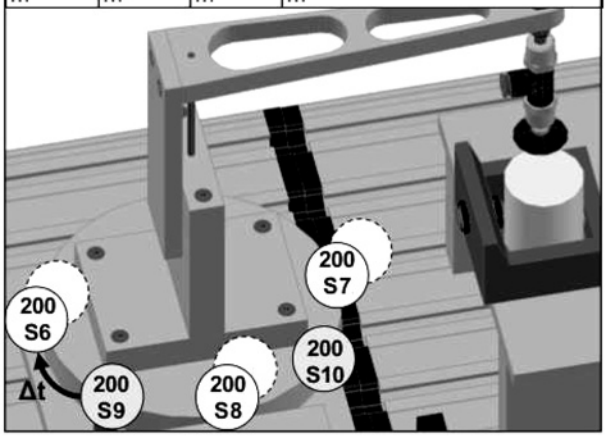

Abbildung 7: Auszug aus der Komponentenliste (oben) sowie Visualisierung der Sensorpositionen (unten) des Krans der PPU für Szenarien 3, 4a und 4b. Als weiß gestrichelte Kreise sind die Mikroschalter in Szenario 3 angedeutet.

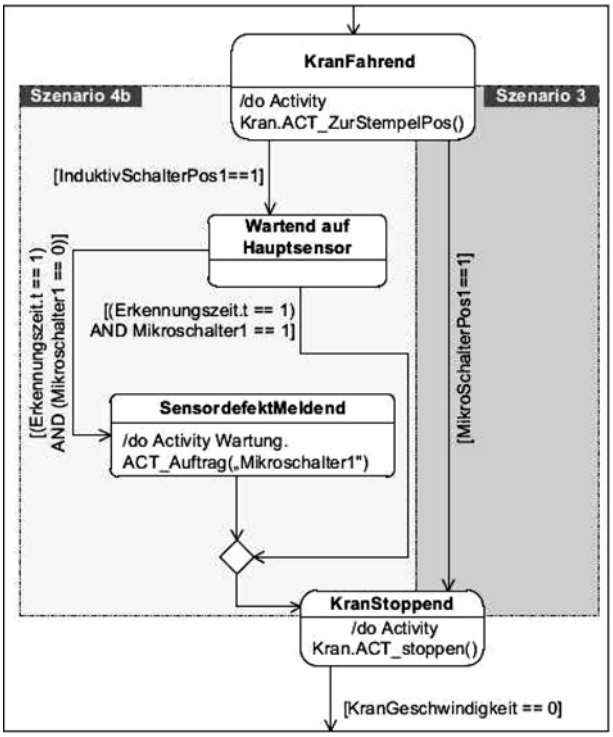

Abbildung 8: Ausschnitt der Verhaltensmodelle der PPU (Szenario 3 und 4b).

kumentation der Evolutionsszenarien der Demonstratoranlage ausgewählt (Werkzeug Papyrus[7]).

Die strukturelle Beschreibung der Anlage erfolgt entsprechend dem SysML-Standard mittels Blockdefinitionsdiagrammen. Da für einen Teil der Projekte der Zusammenhang zwischen Systemkomponenten und Anforderungen von Bedeutung ist, wurden diese den Diagrammen hinzugefügt (vgl. Abbildung 6): Funktionale Anforderungen *(requirement)* werden den erfüllenden Systemelementen mittels Erfüllungsbeziehung *(satisfy)* zugeordnet. Die Systemblöcke sind mit ihren Komponenten sowie mit Randbedingungen *(constraint)* aufgeführt. Anforderung ID 4 repräsentiert beispielsweise die Umsetzung von Szenario 1 (vergrößerte Kapazität der Rutsche realisiert durch eine Y-Rutsche), die Anforderung ID 6 bezieht sich auf Szenario 3. Die Anforderung der Verfügbarkeitserhöhung, die zum Austausch der Mikroschalter durch induktive Sensoren in 4a führt, ist in Abbildung 6 nicht explizit modelliert.

Die Anforderung der Störerkennung (Nutzung Mikroschalter und induktive Sensoren, Szenario 4b), ist in Anforderung ID 9 modelliert. Das Constraint *Erkennungszeit* beschreibt die Zeitdifferenz, die sich aus der räumlich unterschiedlichen Anordnung des induktiven Sensors und der Mikroschalter ergibt. Wenn der induktive Sensor S9 anspricht, muss nach einer definierten Zeit Δt, die sich aus dem Abstand zwischen den Sensoren und der Krangeschwindigkeit in Abhängigkeit der Drehrichtung ergibt, der überwachte Mikroschalter betätigt werden, ansonsten wird der Sensor als gestört erkannt – unter der Annahme, dass der Sensor S9 nicht defekt ist.

Zur Verhaltensbeschreibung können in der SysML unterschiedliche Diagrammtypen genutzt werden, z.B. Aktivitäts- und Zustandsdiagramme. Da die Entwicklung industrieller Steuerungssoftware bei Verwendung (angepasster) Zustandsdiagramme mittels der kommerziellen Entwicklungsumgebung CODESYS erfolgen kann, wird dieser Diagrammtyp für die Dokumentation des Anlagenverhaltens genutzt. In Abbildung 8 ist die Veränderung des Zustandsdiagramms von Szenario 3 nach 4b dargestellt. Ein Zustandsübergang in Szenario 3 von „KranFahrend" zu „KranStoppend" findet auf Basis des entsprechenden Mikroschalters statt. Im Gegensatz dazu werden in Szenario 4b sowohl induktive Sensoren als auch Mikroschalter genutzt: Wird der Mikroschalter nicht innerhalb der erwarteten Zeit aktiviert, wird das Bedienpersonal über

7 www.papyrusuml.com

den defekten Sensor informiert (Zustand „Sensordefekt-Meldend“) und der Kran gestoppt.

Neben der Dokumentation mittels SysML-Modellen stehen den SPP-Projekten für jede Evolutionsstufe die Komponentenliste, elektrische Schaltpläne und Programmcode in IEC 61131-3 sowie die auf Basis von SysML-Zustandsdiagrammen implementierte Software zur Verfügung.

5 Simulation und Schnittstellen zum Demonstrator

Auf Grund der deutschlandweiten Verteilung der Projektpartner innerhalb des SPP ist eine reine Erprobung erarbeiteter Ansätze am Demonstrator in München kaum effizient möglich. Daher ist die Erprobung der durch SPP-Projekte geänderten oder erstellten Steuerungssoftware virtuell notwendig. Für jede Evolutionsstufe wurde dazu ein Simulationsmodell in Matlab mit einer 3D-Visualisierung erstellt, welches das Verhalten der realen Demonstrationsanlage abbildet. Die Matlab-Modelle stehen den SPP-Projekten (Abbildung 9) über eine OPC-Schnittstelle (Object Linking and Embedding for Process Control) mit den bekannten zeitlichen Einschränkungen zur Verfügung. Für die Demonstrationsanlage selbst steht ebenfalls eine OPC-Schnittstelle zur Verfügung, um den Aufwand der Umstellung bei der Evaluation auf der realen Laboranlage gering zu halten.

Alternativ besteht die Möglichkeit (Abbildung 9, rechts oben) direkt in IEC 61131-3 oder dem Zustandsdiagramm (CODESYS) den Steuerungscode zu erstellen und diesen gegen die OPC angekoppelte Simulation oder die reale Anlage zu evaluieren. Auf Grund der Eigenschaften der PPU ist die OPC-Schnittstelle für Steuerungsaufgaben ausreichend leistungsfähig für die meisten Projektansätze.

Zusätzlich steht im Labor der TUM eine SPS-basierte Steuerung zur direkten Anbindung an automatisierungstechnische Hardware zur Verfügung (Abbildung 9, rechts unten).

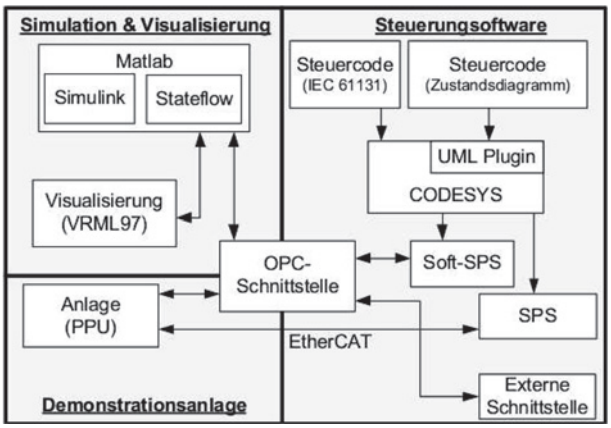

Abbildung 9: Aufbau und Bestandteile der Steuerungs- und Simulationsumgebung der PPU.

6 Erwartete/Angestrebte Ergebnisse durch Nutzung des Demonstrators

Die Bereitstellung des PPU-Demonstrators ermöglicht den Projekten die Erprobung ihrer Ansätze aufbauend auf vorhandenen SysML-Modellen, dem Steuerungscode und den Simulationsmodellen. Durch die Evolutionsszenarien werden Beispiele für die Verknüpfungen bzw. Beziehungen zwischen Änderungen und Teilmodellen der Anforderungen von Kontext, Plattform und Software bereitgestellt.

Exemplarisch wird im Folgenden für einige Projekte die geplante oder bereits realisierte Nutzung des Demonstrators erläutert (Beschreibung der Projekte auf der SPP 1593-Webseite[8]). Zur Unterstützung der Evolution wird im Rahmen des Projektes „*MoDEMAS*“ ein Ansatz zur formalen Verifikation von Systemanforderungen und -eigenschaften auf Basis eines strukturierten, formalen Systemmodells entwickelt [15, 16], um manuelle Prüfungen zu reduzieren und die Wiederverwendbarkeit bereits entwickelter Modelle zu steigern (vgl. Abschnitt 3.2). Das Projekt „*FYPA²C*“ untersucht anhand der PPU den Einfluss der Evolution auf nicht-funktionale Anforderungen, um etwaige Lücken in der Dokumentation zu schließen und den Testaufwand zu reduzieren [17]. Für beide Projekte sind sowohl Dokumentation als auch die reale Anlage relevant.

VVM (vgl. Abschnitt 3.1) wird von mehreren Projekten mit unterschiedlichen Schwerpunkten fokussiert. Die Identifikation typischer, abstrakter Änderungsoperationen wird im Projekt „*MOCA*“ adressiert [18]. VVM von Testartifakten wird im Projekt „*IMoTEP*“ untersucht, um die Wiederverwendbarkeit im Bereich des Testens zu steigern und folglich den Testaufwand zu reduzieren [19].

Im Rahmen des Projekts „*DAPS*“ soll ein Ansatz zur Unterstützung von VVM entwickelt werden, der die Kosten zur Auffindung optimaler Konfigurationen reduziert und die Anpassung des Steuerungsverhaltens bzgl.

8 www.dfg-spp1593.de

nicht-funktionaler Anforderungen zur Laufzeit reduziert [20]. VVM auf Basis von Produktlinienansätzen für die Software im M&A wird durch das Projekt *„Pythia“* auf Basis der bereitgestellten langjährigen Entwicklung der PPU-Software in ST (Strukturierter Text) der IEC 61131-3 analysiert [21].

Während die Evaluationsziele projektindividuell sind, erfolgt die Weiterentwicklung der Evolutionsszenarien in Kooperation mit allen, am Produktionsdemonstrator interessierten Projekten. Inwiefern ungeplante Änderungen automatisch erkannt und kompensiert werden können, ist noch zu beantworten. Die Anforderungen und Änderungen auch während der Laufzeit durchzuführen wird von einigen Projekten bearbeitet und realisiert.

7 Zusammenfassung und Ausblick

Die Zielsetzung des SPP 1593, einen neuen Ansatz für den langfristigen Betrieb von Software unter Einhaltung hoher Qualitätsanforderungen und Flexibilitätsanforderungen in Bezug auf Änderungen von Kontext und Plattform auch für die Produktionsautomatisierung zu erarbeiten, erfordert Ansätze für neue Vorgehensmodelle, Software, die ihre Grenzen und Möglichkeiten kennt (sog. knowledge carrying software), aber auch die Integration von Produktlinien- und Variantenmanagement-Ansätzen in die Engineeringprozesse. Da sich das DFG SPP 1593 im ersten Jahr der Laufzeit befindet, sind erste Lösungskonzepte sichtbar, die mittels des beschriebenen Demonstrators für die Produktionsautomatisierung bereits praktisch erprobt werden. Die Entwicklung der Evolutionsszenarien sowie der Modelle und Dokumentation selbst wird als Anwendungsfall der Softwareevolution von einigen Projekten genutzt. Die für die Produktionsautomatisierung im Maschinen- und Anlagenbau voraussichtlich notwendige Integration verschiedener Lösungsansätze ist zu erarbeiten. Die Detaillierung der Kompatibilitätskriterien (Abschnitt 3.2) ist eine Voraussetzung für die Erarbeitung von Kriterien für die Evaluation der Projektergebnisse, die gemeinsam von den Projekten des Demonstrators vorangetrieben wird.

Ebenso wie die starke Vereinfachung des Demonstrators und die Einschränkung auf diskrete Fertigungsprozesse, mussten bisher wesentliche Aspekte der Software in der Automatisierung für M&A vernachlässigt werden, wie beispielsweise unterschiedliche Betriebsarten, Fehlerbehandlung und komplexe Automatisierungsaufgaben. Dies könnte eine Aufgabe für die nächste Phase des SPP sein.

Danksagung: Die Autoren danken der Deutschen Forschungsgemeinschaft für die Unterstützung im Rahmen des Schwerpunktprogramms SPP 1593: „Design For Future – Managed Software Evolution“ sowie im Rahmen des Sonderforschungsbereich SFB 768 Teilprojekt A6: „Disziplinübergreifendes Modulmanagement von IT-Zyklen in Innovationsprozessen“.

Eingang 1. August 2013; angenommen 1. Oktober 2013.

Literatur

1. G. Reimann: Trendstudie: IT und Automation in den Produkten des Maschinenbau bis 2015. URL: http://www5.vdma.org/wps/portal/Home/de/Branchen/S/SW/Projekte_und_Initiativen/IT%20-%20Umfragen/SW_A120423_Trendstudie_IT_und _Automation_im_Maschinenbau?WCM_GLOBAL_CONTEXT=/wps/wcm/connect/vdma/Home/de/Branchen/S/SW/Projekte_und_Initiativen/IT%20-%20Umfragen/SW_A120423_Trendstudie_IT_und_Automation_im_Maschinenbau [Zugriff am 11.07.2013]
2. R. Birkhofer, G. Feldmeier, J. Kalhoff, C. Kleedörfer, M. Leidner, R. Mildenberger, M. Mühlhause, J. Niemann, R. Schrieber, J. Wickinger, M. Winzenick und M. Wollschläger: Life-Cycle-Management für Produkte und Systeme der Automation – Ein Leitfaden des Arbeitskreises Systemaspekte im ZVEI Fachverband Automation, Zentralverband Elektrotechnik- und Elektronikindustrie e.V. (2010).
3. B. Vogel-Heuser: Visionen für das Engineering der Automatisierungstechnik 2020. – In: atp – Automatisierungstechnische Praxis 51 (2009) Nr. 5, S. 2–9.
4. IEEE Standard Golssary of Software Engineering Terminology. Hrsg. vom IEEE Institute of Electrical and Electronics Engineering. Ausg. 1990.
5. DIN199–1: Technische Produktdokumentation – CAD-Modelle, Zeichnung und Stückliste – Teil 1: Begtriffe. Hrsg. von DIN Deutsches Institut für Normung, Ausg. 2002.
6. VDI: Lexikon der Produktionsplanung und –steuerung Begriffszusammenhänge und Begriffsdefinitionen. VDI-Taschenbuch T77. 3. Aufl. Düsseldorf 1983.
7. S. Feldmann, J. Fuchs und B. Vogel-Heuser: Modularity, variant and version management in plant automation – future challenges and state of the art – In: International Design Conference (2012), S. 1689–1698.
8. S. Dehndorfer: Evaluation objektorientierter SPS-Programmierung auf Basis einer Anforderungsanalyse im in-

dustriellen Umfeld. Bachelorarbeit. Technische Universität München 2012.
9. M. Weyrich und F. Steden: Simulationsmodule methodisch identifizieren: Besseres Aufwand–Nutzen–Verhältnis bei Modellerstellung – In: atp edition – Automatisierungstechnische Praxis (2013) Nr. 7–8.
10. M. Große–Rhode, P. Manhart, R. Mauersberger, S. Schröck, M. Schulze und T. Weyer: Anforderungen von Leitbranchen der deutschen Industrie an Variantenmanagement und Wiederverwendung und daraus resultierende Forschungsfragen – In: Tagungsband des Dritten Workshops zur Zukunft der Entwicklung softwareintensiver eingebetter Systeme (ENVISION2020), Gesellschaft für Informatik, Lecture Notes in Informatics, No. 215 (2013).
11. R. Schrieber, M. Wollschläger, M. Mühlhause und J. Niemann: Kompatibilität:der zentrale Schlüssel für Nachhaltigkeit – Life–Cycle–Excellence durch proaktives Handeln. – In: atp edition – Automatisierungstechnische Praxis (2011) Nr. 11, S. 50–57.
12. B. Vogel-Heuser, C. Diedrich und M. Broy: Anforderungen an CPS aus Sicht der Automatisierungstechnik – In: at – Automatisierungstechnik (2013), DOI: 10.1515/auto.2013.0061, angenommener Beitrag.
13. C. Legat, J. Folmer und B. Vogel-Heuser: Evolution in Industrial Plant Automation: A Case Study – In: 39th Annual Conference of the IEEE Industrial Electronics Society (2013), angenommener Beitrag.
14. M. Hirsch und H. –M.Hanisch: Systemspezifikation mit SysML für eine Fertigungstechnische Laboranlage – In: Fachtagung zum Entwurf komplexer Automatisierungssysteme (2008), S. 23–34.
15. T.Y. Sim, F. Li und B. Vogel-Heuser: Benefits of an Interdisciplinary Modular Concept in Automation of Machine and Plant Manufacturing - In: IFAC Symposium on Information Control Problems in Manufacturing (2009), S. 894–899.
16. M. Broy und K. Stolen: Specification and Development of Interactive Systems – FOCUS on Streams, Interfaces, and Refinement – In: Monographs in Computer Science, Springer (2001).
17. J. Ladiges, C. Haubeck, I. Wior, E. Arroyo, A. Fay und W. Lamersdorf: Evolution of Production Facilities and its Impact on Non-Functional Requirements – In: 11th IEEE International Conference on Industrial Informatics (2013).
18. T. Kehrer, U. Kelter und G. Taentzer: Consistency-Preserving Edit Scripts in Model Versioning – In: 28th IEEE/ACM International Conference on Automated Software Engineering (2013).
19. M. Dukaczewski, I. Schaefer, R. Lachmann und M. Lochau: Requirements-Based Delta-Oriented SPL Testing – In: International Conference on Software Engineering Workshops (2013).
20. M. Manderscheid und C. Prehofer: Network Performance Evaluation for Distributed Embedded Systems Using Feature Models – In: International Conference on Engineering of Complex Computer Systems (2013).
21. L. Passos, K. Czarnecki, S. Apel, A. Wasowski, C. Kästner, J. Guo und C. Hunsen: Feature-oriented software evolution – In: International Workshop on Variability Modelling of Software-intensive Systems (2013), S. 17.

Prof. Dr.-Ing. Birgit Vogel-Heuser
Technische Universität München,
Lehrstuhl für Automatisierung und Informationssysteme, Boltzmannstr. 15,
85748 Garching bei München.
vogel-heuser@ais.mw.tum.de

Prof. Dr.-Ing. Birgit Vogel-Heuser leitet den Lehrstuhl für Automatisierung und Informationssysteme (AIS) der Technischen Universität München (TUM). Ihre Forschungsgebiete adressieren die System- und Softwareentwicklung, insbesondere die Modellierung verteilter, intelligenter, eingebetteter Systeme.

M.Sc. Jens Folmer
Technische Universität München,
Lehrstuhl für Automatisierung und Informationssysteme, Boltzmannstr. 15,
85748 Garching bei München.
folmer@ais.mw.tum.de

M.Sc. Jens Folmer ist wissenschaftlicher Mitarbeiter am Lehrstuhl AIS der TUM. Sein Forschungsinteresse gilt prozessübergreifenden Diagnoseverfahren im Maschinen und Anlagenbau sowie deren Übertragbarkeit auf umgestaltete Anlagenteile und Prozesse.

Dipl.-Inf. Christoph Legat
Technische Universität München,
Lehrstuhl für Automatisierung und Informationssysteme, Boltzmannstr. 15,
85748 Garching bei München.
legat@ais.mw.tum.de

Dipl.-Inf. Christoph Legat ist wissenschaftlicher Mitarbeiter am Lehrstuhl AIS der TUM. Sein Forschungsinteresse gilt der Flexibilisierung von Steuerungssoftware und deren Einfluss auf die Wandlungsfähigkeit mechatronischer Systeme.