# Softwareentwicklung wie am Fließband

## Vorbereitung von Model-Driven Engineering im Maschinen- und Anlagenbau

Thorsten Koch[1], Matthias Meyer[1], Masud Fazal-Baqaie[1], Hubert Runschke[2]

**Abstract:** Unternehmen des Maschinen- und Anlagenbaus sind bestrebt, Entwicklungs- und Inbetriebnahmezeiten für Anlagen zu reduzieren und damit den Absatz zu erhöhen. Der Steuerungscode dieser Anlagen wird typischerweise nicht auf Basis anerkannter Softwaretechnik-Methoden entwickelt und nur unsystematisch per Copy & Paste wiederverwendet. Die Anwendung von moderner Softwaretechnik und insbesondere die modellgetriebene Entwicklung bergen daher ein großes Potential für die Steigerung der Qualität und Effizienz. Allerdings meiden Unternehmen die damit verbundene Komplexität und Investitionskosten. Wir präsentieren ein Stufen-Modell für die Verbesserung der Softwareentwicklung, das leichter von Unternehmen adaptierbar ist und berichten von unseren Erfahrungen mit dem Industriepartner Venjakob bei dessen Anwendung.

**Keywords:** Model-Driven Engineering, IEC-61131, Softwarequalität, Maschinen- und Anlagenbau

## 1 Einleitung

Im Maschinen- und Anlagenbau entwickeln und produzieren Unternehmen häufig Anlagen, die zwar kundenindividuell angepasst werden, aber zu großen Teilen aus sehr ähnlichen Teilsystemen bestehen. Die Steuerung solcher Anlagen erfolgt mit Hilfe von speziell darauf ausgelegten Programmiersprachen gemäß des Standards IEC 61131 [In13]. Die Entwickler haben häufig keine Informatikausbildung und nur wenig Softwaretechnik-Knowhow. Oft handelt es sich um Ingenieure aus dem Maschinenbau oder der Elektrotechnik, die das Programmieren nur in den speziellen Sprachen gelernt haben.

Bei der Entwicklung der Steuerungssoftware werden die Gemeinsamkeiten der Anlagen nur sehr unsystematisch ausgenutzt. Typischerweise wird der Softwarestand der ähnlichsten ausgelieferten Anlage per „Copy & Paste“ als Grundlage genutzt und an die Erfordernisse angepasst. Dieses Vorgehen ist sehr aufwendig und fehleranfällig. Gleiche oder ähnliche Funktionalität wird zum Teil mehrfach entwickelt, und je nach Vorliebe des jeweiligen Entwicklers unterschiedlich strukturiert. Entdeckte Fehler werden so oft nur in einer Anlage behoben anstatt in allen, in denen der Code verwendet wird. Gleichzeitig ist der Quellcode oft plattformabhängig, was wiederum ungewünschte Aufwände für die Reimplementierung bei unterschiedlichen Typen von Steuerungen, Sensorik und Aktorik mit sich bringt [Vo14].

[1] Fraunhofer IEM, Paderborn, Germany, vorname.nachname@iem.fraunhofer.de
[2] Venjakob Maschinenbau GmbH & Co. KG, Rheda-Wiedenbrück, Germany, HRunschke@venjakob.de

doi:10.18420/SE2020_58

Abhilfe kann die Nutzung von modellgetriebenen Entwicklungsmethoden (Model-Driven Engineering, MDE) [St12] schaffen. Dabei werden eine Anlage und ihre Konfiguration auf einer höheren Abstraktionsebene als plattformabhängiger Quellcode modelliert und unter Rückgriff auf wiederverwendbare Softwarebausteine durch Codegenerierung weite Teile der Software für eine Anlage automatisiert erzeugt. Manuelle, fehleranfällige Arbeiten werden so reduziert und die Qualität und Effizienz gesteigert. Die abstraktere Modellierung reduziert die Abhängigkeit von der konkreten Steuerungstechnik und maximiert die Wiederverwendung von plattformabhängigen Softwarebausteinen. Die stärkere Standardisierung reduziert schließlich die Abhängigkeit von einzelnen Entwicklern.

Bei all seinen Vorteilen ist MDE bei der Anlagenentwicklung kaum verbreitet [Vy13], denn den Unternehmen fehlt das notwendige Softwaretechnik-Knowhow. Auch scheuen sie die wesentlichen Investitionen in Modellierungssprache, Softwarebausteine und Codegenerierung. In diesem Papier stellen wir einen Ansatz für eine stufenweise Effizienz- und Qualitätssteigerung vor, der den Weg hin zu MDE systematisiert. Dabei profitiert ein Unternehmen mit jeder Stufe direkt von den Softwaretechnik-Verbesserungen, so dass sich die Aufwände direkt auszahlen. Wir berichten von den Erfahrungen mit der Firma Venjakob Maschinenbau GmbH & Co. KG, die wir mit Hilfe des Modells bei der Adaption hin zu MDE begleitet haben.

Unser Papier ist folgendermaßen strukturiert: In Kapitel 2 stellen wir das Modell vor und berichten in Kapitel 3 von dessen Anwendung bei der Firma Venjakob. In Kapitel 4 präsentieren wir unsere Lessons Learned und in Kapitel 5 fassen wir den Beitrag zusammen.

## 2 Stufen zur modelgetriebenen Entwicklung von Steuerungen

Die Tatsache, dass Maschinen- und Anlagen häufig aus einer Vielzahl von gleichen oder sehr ähnlichen Teilsystemen bestehen, die kundenindividuell kombiniert und konfiguriert werden, ermöglicht ein sehr effizientes und systematisches Vorgehen in der Entwicklung der Steuerungssoftware, bis hin zur modellgetriebenen Entwicklung. Solche Ansätze finden jedoch im Maschinen- und Anlagenbau bisher nur selten Anwendung. Auf Basis unserer Erfahrungen aus Projekten mit dem Ziel des Transfers von effizienteren Softwaretechnik-Methoden in Unternehmen des Maschinen- und Anlagenbaus, wie z. B. der Firma Venjakob, schlagen wir das in Abb. 1 gezeigte, dreistufige Vorgehen vor. Mit jeder Stufe wird die Effizienz und Softwarequalität gesteigert und damit die Inbetriebnahme verkürzt:

**Stufe 1 Codeharmonisierung:** Ausgehend von einer, wie eingangs beschriebenen, von Copy & Paste und unsystematischer Wiederverwendung geprägten Entwicklung, sollte in einem ersten Schritt eine Harmonisierung und Standardisierung des erstellten Codes erfolgen, indem Coding Guidelines (einheitliche Sprache, Begrifflichkeiten, Benennung von Variablen und Funktionen, Formatierung) und/oder Templates eingeführt werden. Auf diese Weise werden die Unterschiede zwischen verschiedenen Entwicklern und so die

Abhängigkeit von Einzelpersonen reduziert. Die Einarbeitung in andere Anlagenprojekte, die Wartung und Fehlersuche werden vereinfacht.

**Stufe 2 Softwarebausteine:** Die nächste Stufe treibt die bereits begonnene Standardisierung weiter, indem wiederverwendbare Softwarebausteine identifiziert und für alle Anlagenprojekte bereitgestellt werden. Die Entwicklung kann so effizienter erfolgen, da Funktionalität nicht aufwändig immer wieder neu entwickelt wird. Die Qualität steigt, da qualitätsgesicherte Implementierungen eingesetzt und Fehler vermieden werden. Entwicklungs- und Inbetriebnahmezeiten werden reduziert.

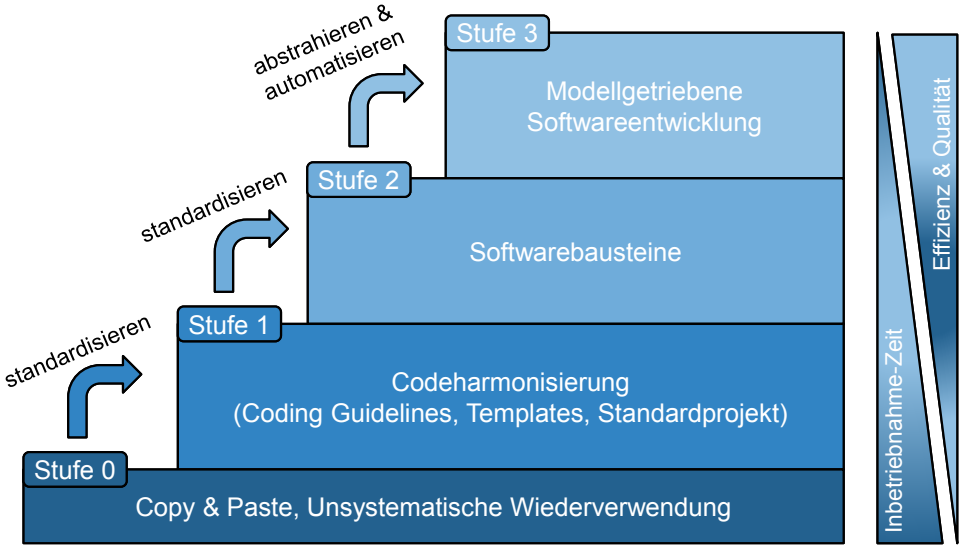


Abb. 1: Stufenmodell der Effizienz- & Qualitätssteigerung

**Stufe 3 Modellgetriebene Softwareentwicklung:** Weitere Effizienzsteigerungen werden durch Abstraktion und (Teil-)Automatisierung der Softwareerstellung möglich. Wissen über den typischen Aufbau von Anlagen wird genutzt, um eine angepasste Modellierungssprache zu entwickeln, die eine Modellierung und Konfiguration einer Anlage auf einer höheren Abstraktionsebene als von der Steuerungstechnik abhängiger Quellcode erlaubt. Aus solchen Modellen können unter Rückgriff auf die Softwarebausteine durch Codegenerierung weite Teile der Software für eine Anlage automatisiert erzeugt werden. Der Rest wird manuell ergänzt. Die abstraktere Modellierung reduziert die Abhängigkeit von der konkreten Steuerungstechnik und verkapselt diese in Codegeneratoren. Durch das stufenweise Vorgehen können Unternehmen mit zunächst geringem Aufwand und geringen fachlichen Anforderungen an ihre Entwickler bereits Verbesserungen in der Softwareentwicklung erzielen. Mit jeder aufbauenden Stufe wachsen Anforderungen, aber auch Mehrwerte. Stufe 2 erfordert z.B. bereits Software-Werkzeuge für die Verwaltung und Verwendung der

Bausteine. Stufe 3 benötigt dedizierte Teams mit weitreichenden Softwaretechnik-Skills, die Modellierungswerkzeug und die Codegeneratoren entwickeln.

## 3 Anwendungsprojekt mit Unternehmen Venjakob

Das Unternehmen Venjakob Maschinenbau GmbH & Co. KG ist ein mittelständischer Hersteller für kundenindividuelle Oberflächenanlagen und fördertechnische Produkte, das langfristig auf modellgetriebene Methoden für die Entwicklung der Steuerungssoftware umstellen möchte. Venjakob entwickelte diese bisher auf Basis eines Referenzprojektes. Für jedes Kundenprojekt wurde dieser Code kopiert und aufwendig manuell angepasst, was zu den in der Einleitung beschriebenen Fehlern und Verzögerungen führte.

Dem gemeinsamen Projekt wurde eine Ist-Analyse vorausgeschaltet. Der Softwareentwicklungsprozess wurde in mehreren Workshops aufgenommen und ein Review der Steuerungssoftware von insgesamt 20 Anlagen durchgeführt. Die resultierenden Handlungsempfehlungen wurden in dem anschließenden Hauptprojekt adressiert. Dazu wurden zwei Entwickler von Venjakob sechs Monate lang von einem wissenschaftlichen Mitarbeiter an zwei Tagen pro Woche begleitet. Die Zwischenergebnisse des Projekts wurden regelmäßig den restlichen Entwicklern vorgestellt. Entsprechend dem Stufenmodell sollte die Codeharmonisierung verbessert werden (Stufe 1). Außerdem sollte eine Referenzarchitektur mit Softwarebausteinen abgeleitet und die Mitarbeitenden zur modernen Softwaretechnik geschult werden (Stufe 2).

Für die Codeharmonisierung wurden gängige Coding Guidelines zur IEC 61131-3 Programmierung (z.B. PLCopen) gesichtet und implizite Coding Guidelines aus dem Standardprojekt extrahiert. Anschließend wurden die Vor- und Nachteile einzelner Regeln mit den zwei Entwicklern diskutiert und integrierte, einheitliche Coding Guidelines für Venjakob entwickelt.

Für die Referenzarchitektur und Softwarebausteine wurde die Einführung einer Bibliothek für Softwarebausteine adressiert, um eine systematische Wiederverwendung zu ermöglichen. Dazu wurden die beiden Entwickler in der Modellierungssprache UML [Ob17] und typischen Architekturmustern geschult. Anschließend wurde eine Referenzarchitektur für die Anlagen von Venjakob mit Hilfe der UML modelliert.

Für die Vermittlung moderner Softwaretechnik wurden die beiden Entwickler parallel zu der Erarbeitung der Referenzarchitektur in dem Versionsmanagementsystem TFS und den objektorientierten Erweiterungen der IEC 61131-3 geschult. Die Schulungsinhalte konnten während der Implementierung einzelner Bausteine der Referenzarchitektur angewendet und vertieft werden. Zudem konnten die definierten Coding Guidelines evaluiert und an einigen Stellen angepasst werden. Um die Qualität der einzelnen Softwarebausteine zu gewährleisten, wurden die beiden Entwickler zudem in Konzepten zum Testen geschult. Der Fokus lag dabei auf Unit-Tests, da ein Integrationstest aus Mangel einer realen Anlage nicht möglich war. Der Support von Unit-Tests innerhalb der IEC 61131-3 Sprachen sowie

der Entwicklungsumgebungen ist relativ gering. Aus diesem Grund wurde auf das freie TcUnit-Framework zurückgegriffen.

Als Ergebnisse lagen zum Projektende Coding Guidelines zur Harmonisierung der Steuerungssoftware inklusive der Dokumentation vor. Zudem lag eine vollständige und dokumentierte Referenzarchitektur der Anlagen von Venjakob vor. Des Weiteren wurden erste Softwarebausteine implementiert und mit Hilft des TcUnit Frameworks getestet. Die Entwickler des Projekts wurden zudem befähigt, die verschiedenen Softwaretechnik-Konzepte anzuwenden und ihre Kollegen und Kolleginnen in der Anwendung zu schulen.

Auf diese Weise konnte Venjakob die Implementierung der Softwarebausteine allein fortsetzen. In regelmäßigen Abständen haben wir uns über den Fortschritt informiert. Insgesamt hat sich gezeigt, dass es während der Implementierung kaum Änderungsbedarf an der Referenzarchitektur und den Templates zur Implementierung der Bausteine gegeben hat. Im letzten Quartal des Jahres 2019 wurde zudem mit dem Testen der Bausteine an einer Anlage begonnen.

## 4 Lessons Learned

Bei der Durchführung des Projekts mit Venjakob haben wir für zukünftige Projekte einige Erfahrungen gesammelt, die wir hier darstellen möchten.

**Stufenmodell strukturiert die Verbesserung:** Wir haben die Erfahrung gemacht, dass eine Orientierung an dem Stufenmodell zum einen die projektinterne Strukturierung von Verbesserungsvorhaben unterstützt, zum anderen aber auch die Kommunikation an Stakeholder vereinfacht, zum Beispiel beim Erwartungsmanagement für Erfolge.

**Verfügbarkeit der Mitarbeitenden sicherstellen:** Wenn Mitarbeitende nicht exklusiv und nicht von Beginn an für das Vorhaben zur Verfügung stehen und parallel Anlagenprojekte betreuen und daher kurzfristig ausfallen, kann das den Projektfortschritt verzögern. Das Wissen dieser Teammitglieder steht ggf. nicht zur Verfügung, wenn es benötigt wird, andererseits müssen Trainingseinheiten für sie wiederholt werden.

**Überlastung der Mitarbeitenden verhindern:** Das Projekt wurde als Chance genutzt, um auf eine neue Version der Entwicklungsumgebung umzustellen. Neben den modernen Softwaretechnik-Methoden mussten die Mitarbeitenden somit auch die neue Umgebung erlernen, was eine sehr steile Lernkurve bedeutete. Für den Projekterfolg ist es daher notwendig, die Belastungsgrenze im Blick zu behalten und Veränderungen einzuschränken.

**Lernphasen mitplanen:** Neben der Verbesserungen der Codebasis hatte das Projekt auch die Vermittlung von moderner Softwaretechnik als Ziel. Hier gilt es genug Zeit für die praktische Vertiefung einzuplanen und diese auch einzuhalten, damit das Ziel der Wissensvermittlung nicht der Arbeit an der Codebasis untergeordnet wird. Dabei können explizit eingeplante Schulungsphasen hilfreich sein.

**Mitarbeitende als Technologie-Botschafter:** Während des Projekts wurden den restlichen Mitarbeitenden der Abteilung wiederholt neue Konzepte vorgestellt und deren Anwendbarkeit in der täglichen Arbeit diskutiert. Wir haben die Erfahrung gemacht, dass diese positiver eingeschätzt wurden, wenn Projektmitarbeitende des Unternehmens (und nicht wissenschaftliche Mitarbeitende) Konzepte und Technologien vorstellten. Diese konnten praktische Vorteile am Beispiel vergangener Projektkontexte darstellen.

## 5 Zusammenfassung

Um die Entwicklung und Inbetriebnahme von Anlagen zu beschleunigen, gilt es die Entwicklung von Steuerungssoftware zu professionalisieren und auf Softwaretechnik-Knowhow zurückzugreifen. Insbesondere modellgetriebene Entwicklung erscheint hier erfolgsversprechend [Li18], den Unternehmen fehlt es aber an Wissen und sie scheuen die hohen Investitionen. Wir präsentierten hierzu ein Modell für die stufenweise Adaption von Verbesserungen, das den Pfad zur modelgetriebenen Entwicklung für Unternehmen strukturiert und erleichtert. Wir berichteten von den gemeinsamen Erfahrungen mit der Firma Venjakob, die langfristig auf modellgetriebene Entwicklung umstellen möchte und mit unserer Hilfe zwei der drei Stufen dazu erklommen hat. In Zukunft planen wir, neben Venjakob auch weitere Unternehmen mit Hilfe des Models hin zu modellgetriebener Entwicklung zu begleiten.

## Literatur

[In13] International Organization for Standardization: IEC 61131-3:2013: Programmable controllers - Part 3: Programming languages, 2013.

[Ob17] Object Management Group: Unified Modeling Language – Version 2.5.1, 2017.

[St12] Stahl, T.; Völter, M.; Efftinge, S.; Haase, A.: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag, s.l., 2012.

[Vo14] Vogel-Heuser, B. et al.: Challenges for Software Engineering in Automation. Journal of Software Engineering and Applications 07/05, S. 440–451, 2014, ISSN: 1945-3116.

[Vy13] Vyatkin, V.: Software Engineering in Industrial Automation: State-of-the-Art Review. IEEE Transactions on Industrial Informatics 9/3, S. 1234–1249, 2013.