Vorwort

Wird die Suche nach der Lösung einer gestellten Aufgabe zu theoretisch oder zu komplex angegangen, entsteht daraus oftmals ein größeres Problem als anfänglich erwartet wurde. Das gilt ganz besonders für spezielle Bereiche der Software-Entwicklung. Hier ist zudem der Nachweis, wie funktioniert meine Software und welche Auswirkung hat diese, vereinfacht gesagt, schwierig zu erbringen und erfordert kosten- und zeitintensive Testsoftware.

Software kostengünstig und erfolgreich zu erstellen und deren Komplexität dabei so auszulegen, dass jedermann diese auch im Nachhinein noch versteht, ist für Techniker und Ingenieure nicht nur aus der Automatisierung eine große Herausforderung, sondern stellt wohl grundsätzlich eine gewisse Hürde –in der technischen Informatik dar!

Werkzeuge aus der UML[1] oder gar Design-Pattern[2] als modellhafte SW-Vorlagen sind hier gegebenenfalls sehr hilfreich, aber in der Welt der Automatisierer noch seltener zu finden. In der Branche SPS/Regelungstechnik sind diese nicht so verbreitet sind. Hier werden eher Simulationen, wie z. B. MATLAB[3], und deren Codegenerierung vorgezogen. Diese sind allerdings nicht Thema dieses Buchs, sondern der Startpunkt einer ersten Feststellung zum Buchthema:

Einfache Lösungen nach Modellen können vielfach effektiver und besser zu durchschauen sein! Sie sind in jedem Fall vergleichbar und bieten damit automatisch eine Kontrolle über dessen Qualität.

In diesem Buch wird ein Lstwechselregler zur Regelung elektrischer Antriebe mit beweglichen Lasten unter einem völlig neuen Aspekt betrachtet. Die Idee entstand sozusagen durch den Nebeneffekt des „Kaputt-Regelns“, welche traditionelle Regler, wie der in den 80er-Jahren entwickelte PID-Regler, bei bestimmten Anwendungen und Einstellungen oft mit sich bringen. Der Einsatz des P-Reglers ist aus meiner Erfahrung speziell bei Antrieben mit nicht kalkulierbarem Lastwechsel[4], wie beispielsweise Krane, oft fatal. Dieser kann auf den Lastwechsel unkontrolliert reagieren und dadurch wird ein Schwingen erzeugt, das zum Pendeln führt und damit den Lastwechsel sogar unterstützt.

Bei den Inbetriebnahmen konnte ich zudem feststellen, dass häufig ein Spezialist für die Einstellung der Frequenzumrichter anwesend sein musste, um besonders spezielle oder oft nicht dokumentierte Zustände anzupassen. Dies ist nicht nur sehr nervend, sondern verur-

[1] Unified Modeling Language ist ein Werkzeug und dient der SW-Modellierung. Sie stellt eine gemeinsame Quasisprache für verschiedene Programmiersprachen dar.
[2] Design Patterns ist der englische Ausdruck für Entwurfsmuster und dient Programmierern sozusagen als Kochbuch für bereits erfolgreiche, bestehende Programmgerüste.
[3] MATLAB ist ein Eigenname für die kommerzielle Software von MathWorks, www.mathworks.de.
[4] Mit Lastwechsel ist nicht die elektrische Last gemeint, sondern die Last als Gewicht (Pendeln).

sacht ein ständig, wiederholt auftretendes Problem, wenn sich die Anforderungen nach der Inbetriebnahme oder die Situation wie Lastenwechsel, Seillänge, wetterbedingte Störeinflüsse etc. ändern.

Daher wurde mein Wunsch, das Steuerungsgeschehen, wie der schon beispielhaft erwähnte Lastenwechsel, mit einem lernfähigen Regler in den Griff zu bekommen, immer größer. Nach einigen erfolglosen Umwegen über Software-Modelle, Hardware-Ausstattungen usw. bin ich schließlich auf die Idee gekommen, alle mir bekannten Technologien, Algorithmen und was es da sonst noch so gibt, einfach fallen zu lassen, um den ganzen Vorgang meiner gewünschten Regelung unvoreingenommen aus einer ganz neuen Perspektive zu betrachten zu können.

Dabei hat mir zuerst die Theorie der C++-Design-Pattern[5] geholfen, welche ich zuvor überhaupt nicht akzeptieren konnte,– oder besser nicht wollte. Wie kann man als begeisterter Programmierer eine technisch orientierte Applikation zuerst mit UML planen, anstatt gleich ins Programm umzusetzen? Ganz allmählich wurde mir allerdings klar, dass eine besondere Planung auch ein gewisses Loslösen von bekannten, immer wieder verwendeten Lösungen verursachen kann. So entstehen neue oder auch noch nicht bewusste Ideen, da die ständigen Überlegungen in der Planungsphase, vorerst das Programmieren verhindern und damit eine gewisse Konzentration auf das eigentliche Problem wächst.

Zusammenfassend aus dieser und anderen Überlegungen ist die Idee entstanden, eine „Testfahrt“ zu unternehmen, um die so entstehenden, realen Istwerte vorerst als Sollwerte zu speichern. Damit können unter verschiedenen Bedingungen verschiedene Modell-Algorithmen das gewünschte Ergebnis liefern, da diese lernen aus den Sollwerten, Stellwerte zu generieren. Und tatsächlich hat diese Idee sehr positive Ergebnisse erbracht. Pendelarmes Fahren, frühe Erkennung von Prozessfehlern, kurze Inbetriebnahmezeiten und viele neue Erkenntnisse des Prozesses sind damit auf fast einfache Weise erreicht worden. Der Erfolg war teilweise so überraschend, dass sogar ausgewachsene Steuerungsspezialisten über dieses Ergebnis völlig erstaunt waren und lange, lange rätselten, warum und wie das so funktionieren kann.

Nun, wie und was und womit man mit Modellkurven regeln und steuern kann, will ich versuchen, in diesem Buch zu zeigen. Ich wünsche jedem Interessierten viel Spaß beim Lesen und Studieren und vor allen Dingen neue, moderne Denkweisen zu entwickeln, damit alte Krusten aufgebrochen werden, um wieder Neues zu entdecken.

Landau/Pfalz, Frühjahr 2014

Johannes Hofer

[5] Herold, H.; Klar, M.; Klar, S.: C++, UML und Design Patterns: Grundlagen und Praxis der Objektorientierung, München: Addison-Wesley, 2005, ISBN 978-3-8273-2267-8