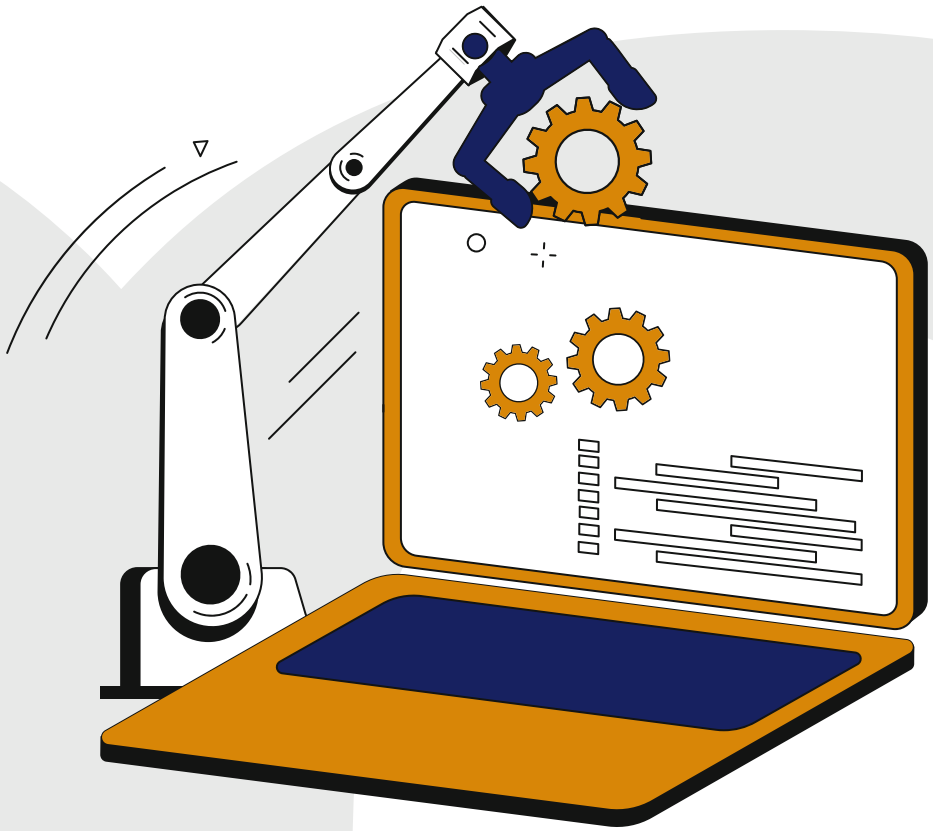
interfaceless design



Automate Your Code

HOOX [SOFTWARE]

HOOX

[SOFTWARE]

Fallbeispiel – Einleitung

Die Firma M&A Dieterle aus Ottenbach entwickelt und fertigt Anlagen zur Herstellung von Carbonfasertapes. Am Anfang fand die gesamte Entwicklung der Funktionen an einer Testanlage statt.

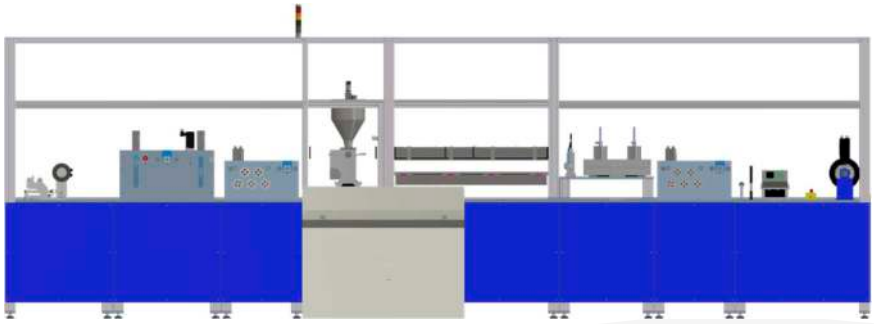
Im Wesentlichen handelt es sich dabei um einen Abwickler, einen Transport und einen Antrieb für den Aufwickler. Dazwischen findet dann ein Prozess statt. Oberste Priorität war zum modularen Hardware-/Maschinenaufbau einen ebenso modularen Softwareaufbau zu erstellen.

Es entstand Version 1.0, klassisch programmiert.

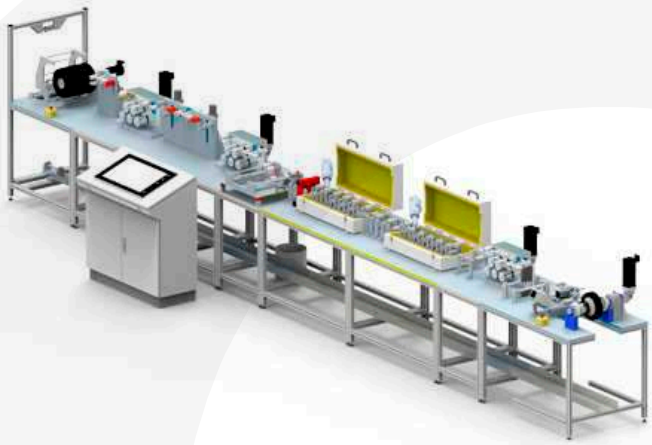
Nachdem die Grundfunktionen fertig waren, wurde mit dem Wissen aus der Testanlage eine Produktionsanlage geplant und gefertigt. Aufgrund der Länge gab es hier schon zwei Transporteinheiten und wir benötigten eine Regelung zwischen den Transporten um die Bandspannung stabil zu halten.

In der Folge kamen die ersten Kundenaufträge die eine Tapeanlage kaufen wollten.
Diese mussten aber, aufgrund der anderen Prozessanforderungen, anders aufgebaut werden als die "Standard"-Produktionsanlage.

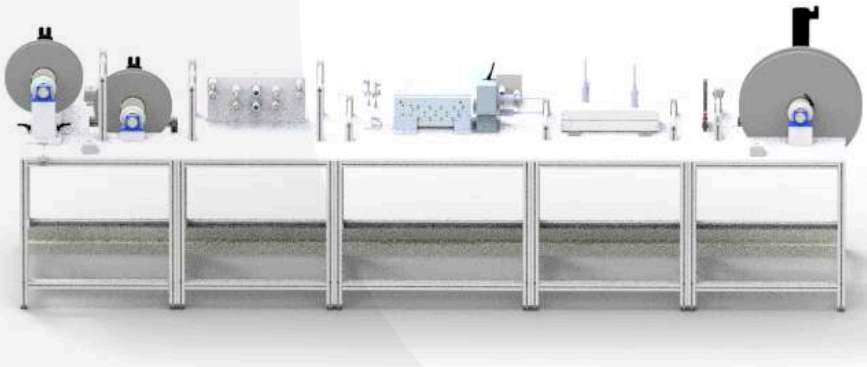
Produktionsanlage: Abwickler, Prozess 1, Transport, Prozess 2, Transport Aufwickler



Beispiel 1: Abwickler, Transport, Prozess 1, Transport, Prozess2, Transport, Aufwickler

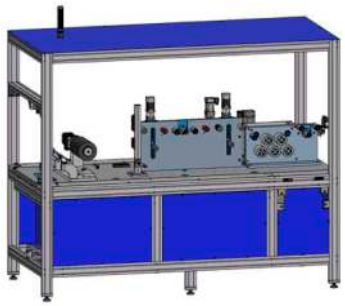


Beispiel 2: 2x Abwickler, Transport, Prozess 1 , Aufwickler



Beispiel 3:

Hier handelt es sich um eine Anlage die in einer Variante "Kurz" und "Lang" betrieben werden kann. Der wesentliche Unterschied zum Standard ist die unterschiedlich Elektrik. Während im Standard ein Schaltschrank alles enthält wurden hier die einzelnen Komponenten auf drei Schränke verteilt.



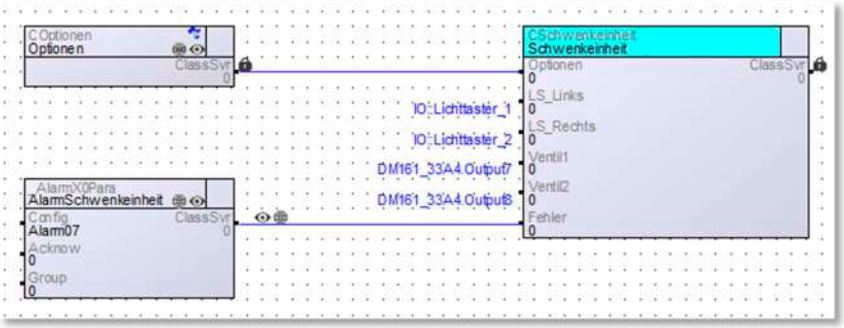
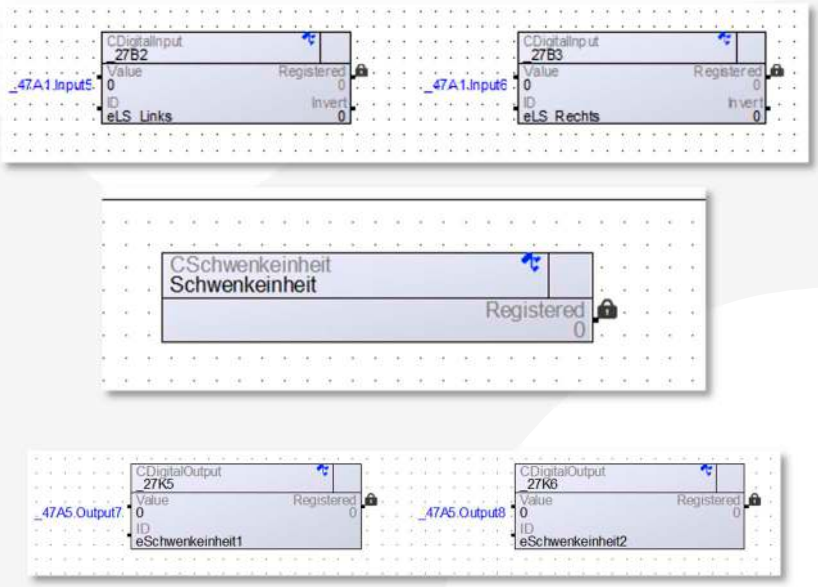
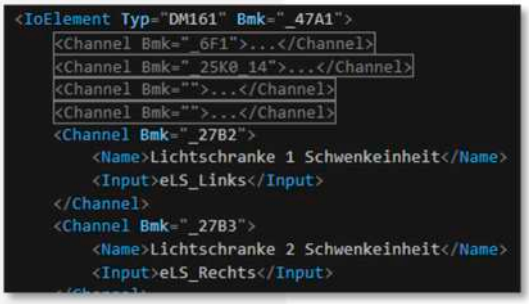
Variante "Kurz":
Abwickler, Prozess, Transport, Aufwickler

Variante "Lang":
Abwickler, Prozess, Transport, Prozess 1, Prozess 2, Transport, Aufwickler

Die ersten beiden Aufträge wurden noch klassisch als angepasstes Programm mit zuschaltbaren Optionen umgesetzt.
In der Phase arbeitete ich parallel an einem Maschinenframework. Ziel war es extrem modular zu sein und per Skript die eigentliche Maschinensoftware zu generieren.
Auf Anfrage beim Kunden ob wir das Ganze als Frameworklösung umstellen wollen wurde dies sofort angegangen.

Es wurden alle Funktionen in einer Bibliothek zusammengefasst und die einzelnen Maschinen im XML definiert und per Python Skript (lauffähig!) generiert.

Beispiel aus dem Programm
(Umsetzung mit Lasal Class 2)

Alt	
	Die Klasse Schwenkeinheit wurde klassisch erstellt, das Objekt instanziert und die Clients verbunden.
Neu	
	Die Klasse Schwenkeinheit wurde mit Hilfe des Interfaceless Designs erstellt. Die Objekte für die Hardware und die Maschinenfunktion wurden mit dem Generator aus einer XML Beschreibung heraus erstellt.
XML	
	Die Datei enthält die BMK aus dem Schaltplan und ist somit eindeutig zuzuordnen.

Variante Alt
wenn keine Schwenkeinheit vorhanden war musste dies im Objekt Optionen konfiguriert werden.
Variante Neu
Wenn keine Schwenkeinheit vorhanden ist, lässt man das Objekt einfach weg.
Ein Fehlen von Instanzen kann auch nicht zu weiteren Fehlern im restlichen Programm führen.
Sollte man vergessen Hardwaresignale zu instanzieren werden Zugriffe darauf abgefangen sodass keine Folgefehler entstehen können.

Zusammenfassung

Die Aufgabe:

Hohe Varianz in Funktion und Hardwareaufbau und trotzdem ein standardisiertes Programm.

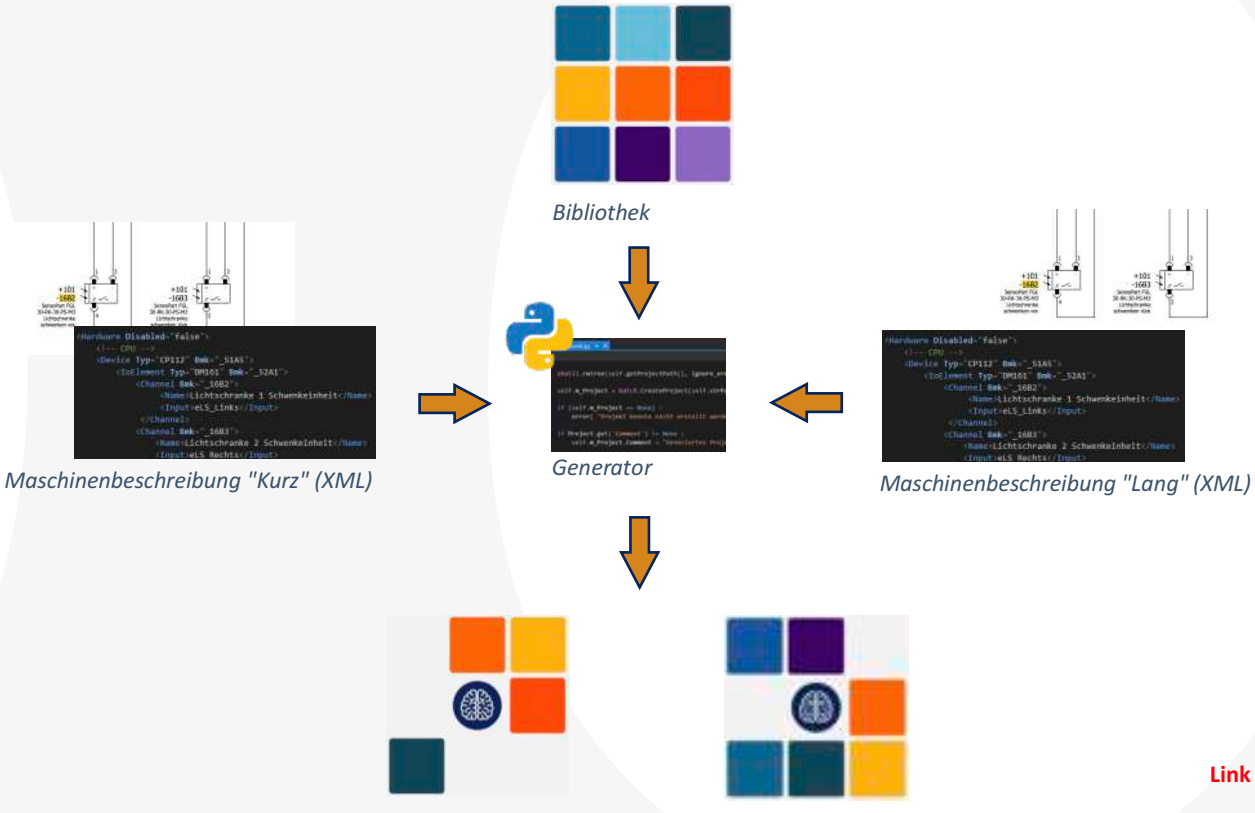
Die Lösung:

Interfaceless Architecture : Möglichst wenig verschiedene Schnittstellen (In dem Projekt genau eine Schnittstelle) zwischen den Funktionen und eine zu den Hardwaresignalen.
Je nach Aufgabenstellung kann dies auch anders aussehen und ist nicht in Stein gemeiselt.
Beschreibung der konkreten Anlage nur noch als leicht lesbare XML Datei.

Das Ergebnis:

- Anlagen/Maschinen können kundenspezifisch vollkommen frei zusammengestellt werden
- Flexible Elektroplanung (jede Anlage komplett frei planbar)
- Übernahme der BMK + Kommentare in das generierte Programm.
- Master/Slave Betrieb zwischen den Antrieben auswählbar
- Bausteine unabhängig voneinander testbar
  mögliche Abhängigkeiten können mit Stubs oder Mocks sehr einfach abgebildet werden.

Der Generator:



Link zum Projekt:

M&A DIETERLE MASCHINEN- & APPARATEBAU GMBH