tgm
Die Schule der Technik

DIPLOMARBEIT

# Batch_it

Ausgeführt im Zuge der Reife und Diplomprüfung
Ausbildungszweig Systemtechnik

unter der Leitung von
Dipl.-Ing. (FH) Mag. Dr.techn. Gottfried Koppensteiner
Abteilung für Informationstechnologie

eingereicht am Technologischen Gewerbemuseum Wien
Höhere Technische Lehr- und Versuchsanstalt
Wexstrasse 19-23, A-1200 Wien

von
Paul Adeyemi, 5AHITT
Jakob Saxinger, 5AHITT
Nikolaus Schrack, 5AHITT
Philipp Schwarzkopf, 5AHITT

Wien, im April 2015

Abteilungsvorstand: Dipl.-Ing. (FH) Mag. Dr.techn. Gottfried Koppensteiner

Tag der Reifeprüfung: xx. xx xxxx

Prüfungsvorsitzender: Univ.–Prof. Dipl.-Ing. Dr.techn. xxx

Erster Gutachter: Dipl.-Ing.(FH) Mag. Dr.techn. Gotti Koppi

Zweiter Gutachter: Prof. Dr.techn. Wenn Vorhanden

# Vorwort

Diese Arbeit wurde im Jahr 2014 im Zuge unserer Ausbildung in der Abteilung für Informationstechnologie am Technologisches Gewerbemuseum (TGM), HTBLVA Wien 20, durchgeführt.

Dankesworte

Wien, im April 2015 Adeyemi, Saxinger, Schrack, Schwarzkopf

# Abstract

This is the english abstract.

# Kurzfassung

Deutsche Kurzfassung kommt hierher

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Listings

# KAPITEL 1

# Einleitung

Produktionssysteme im 21. Jahrhundert müssen wachsenden Anforderungen gerecht werden. Besonders die auftragsbezogene Einzelproduktion bringt große Herausforderung mit sich. Dabei wird eine kurze Durchlaufzeit bei hoher Qualität ebenso erwartet, sowie keine oder nur geringe Mehrkosten gegenüber der herkömmlichen Serien- und Massenproduktion. Eine Produktionsanlage muss daher unterschiedliche, hoch konfigurierbare, Produkte auf einmal schaffen können. Zusätzlich haben sich Lebenszyklen von Waren verringert, welches ein schnelles umstellen der Produktionssysteme fordert. [1]
Diese hohe Flexibilität hat allerdings ihren Preis. Vielfach ist es so, dass bei Änderungen wie dem Hinzufügen oder Entfernen von Teilen großer Aufwand nötig ist, um das System in neuer Zusammensetzung in Betrieb zu setzen. Besonders für die Steuereinheiten von Produktionsanlagen, die sog. speicherprogrammierbare Steuerung (SPS), muss ein Großteil des Programm-Codes neu implementiert werden. [2] Das Diplomprojekt hat sich zur Aufgabe gemacht, diesen Prozess näher zu untersuchen und zu klären in welchem Ausmaß und in welcher Art und Weise sich dieser Schritt anhand einer Demoimplementierung auf einer Laboranlage automatisieren lässt.

Diese Problematik der Automatisierung wird in den nächsten Abschnitten weiter beleuchtet. Dabei wird auf Motivation und Hintergrund der Arbeit eingegangen und der Stand der bereits realisierten Projekte erläutert. Ein Überblick über die Aufgabenstellung sowie ein Leitfaden durch die Arbeit schließen dieses Kapitel ab.

## 1.1 Hintergrund und Ausgangspunkt

Der Verein Practical Robotics Institute Austria, kurz PRIA, dient der Förderung des wissenschaftlich-technischen Nachwuchses durch Robotik und nimmt sich der Forschungsaufgaben in verwandten Fachgebieten der Robotik und Automation an. Im Rahmen diese Instituts wird in dem Projekt "Batch Process Automation with an Ontology-driven Multi-Agent System (BatMAS)", das von der österreichischen Forschungsföderungsgesellschaft (FFG) gefördert wird, an einem ontologiebasierten Informationsmodell, um die Konzepte von Produktionssystemen zu beschreiben, geforscht. Anhand dieses sollen intelligente Softwarekomponenten (Agenten) eingesetzt werden, die Aufträge dynamische Zuteilung um die Produktionsdauer zu verringern und damit den Durchsatz des Systems zu erhöhen.

Das Projekt Batch_it greift diese Ansätze auf und hat sich zum Ziel gesetzt eine modellbasierte Entwicklung der Steuerung einer Produktionsanlage umzusetzen. Hierfür wird in einem Modell eine Produktionsanlage abgebildet sein, um im weitern Schritt eine automatische Codegenerierung anhand dieses zu ermöglichen. Wenn es zu Änderungen der Produktionsanlage kommt, können diese im Modell angepasst werden und durch einen Codegenerierung der nötige Anpassungsschritt der Implementierung der SPS automatisiert werden.

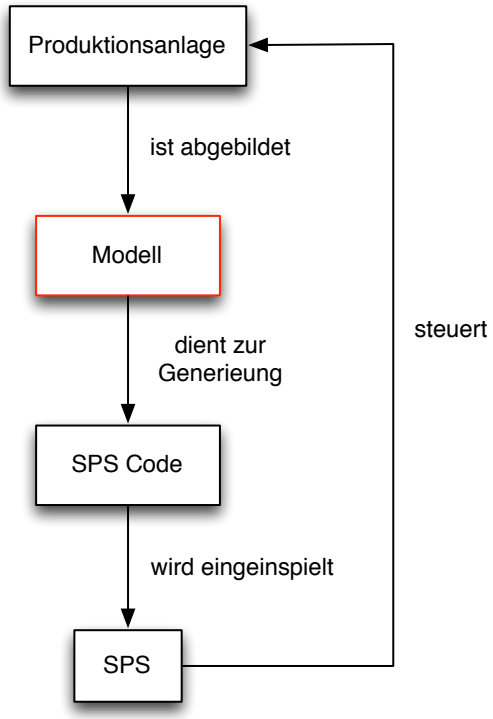


Abbildung 1.1: Konzept für die modellbasierte Entwicklung der Steuerung

In der Abbildung 1.1 ist zu sehen, dass die Produktionsanlage in dem Modell abgebildet ist, aus dem der SPS Code generiert wird. Daraufhin kann dieser in die SPS eingespielt werden und im weitern die Anlage gesteuert werden.

## 1.2 Aufgabenstellung

Im Rahmen des Projekts Batch_it soll eine Visualisierung und Steuerung sowie eine modellbasierte Entwicklung der Steuerung für eine Laboranlage für Chargenprozesse, die sich durch einen chargenorientierte Fahrweise (Batch Control) kennzeichnet, implementiert werden. Dies umfasst das Erstellen von Phases nach dem IEC 61512 Standard. Hierbei handelt es sich um Funktionen wie zB. das Pumpen von Flüssigkeiten von Tank zu Reaktor oder das Heizen bzw. das Vermengen einer Flüssigkeit. Darüber hinaus werden daraus Rezepte nach dem IEC 61512 Standard entwickelt. Zusätzlich werden Fehlerszenarien einer Produktionsanlage aufgestellt, die zur spätere Entwicklung einer Fehleranalysen, dienen. Im weitern Schritt wird eine Ontologie, welche das Modell darstellt, für die Abbildung der Anlage erstellt. Anhand dieser wird eine automatische Codegenerierung für die Steuerung der Anlage implementiert.

## 1.3 Leitfaden durch die Arbeit

TODO

KAPITEL 2

# State of the Art

## 2.1 Hardwareaufbau einer SPS

Um die im späteren Verlauf darauf aufbauende Hardware genauer beschreiben zu können, muss zu Beginn ein wenig thematisch ausgeholt werden. So hat der Begriff der Steuerung einen undenkbar hohen Stellenwert, welcher ebenso geklärt gehört.

Unter einer Steuerung eines Prozesses versteht man nach der standardisierten Norm DIN 19226 einen Vorgang, bei dem durch Rückführen gemessener Prozesszustände, verglichen mit von der speicherprogrammierbaren Steuerung formulierten Parametern, Sollwerte zur Beeinflussung des gesamten Prozesses erzeugt werden. Im Gegensatz zu einer Regelung, muss der von einem Sensor ausgelesene Ist-Wert jedoch nicht zwanghaft rückgeführt werden.

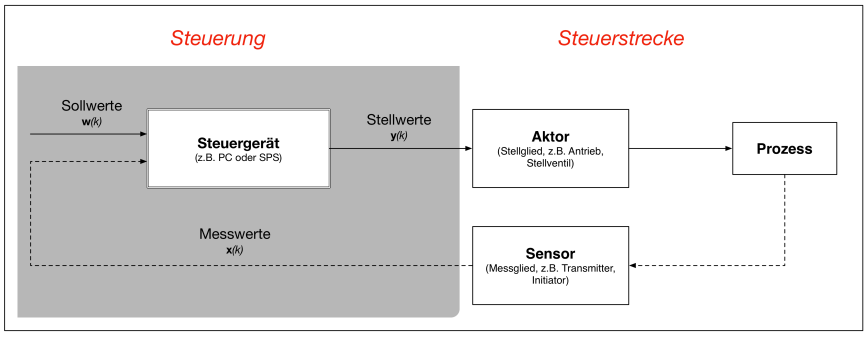


Abbildung 2.1: Allgemeiner Aufbau eines Steuerkreises

Wie bereits zuvor teilweise angeführt, besteht der Steuerkreis aus Sensoren, einem Steuerungsrechner, welcher meistens mittels einer SPS realisiert wird, sowie Aktoren. Die Steuerungseinheit selbst ist ein Rechner, mit dem Hauptaufgabenbereich, die programmierten, im Speicher abgelegten Anweisungen zyklisch auszuführen und über einfach anzusprechende Schnittstellen Daten einzulesen oder auszugeben. [8]

Der Begriff SPS wird in der Norm DIN EN 61131-1 (IEC 61131-1) nochmals wie folgt definiert:

„*Ein digital arbeitendes elektronisches System für den Einsatz in industriellen Umgebungen mit einem programmierbaren Speicher zur internen Speicherung der anwenderorientierten Steuerungsanweisungen zur Implementierung spezifischer Funktionen wie z.B. Verknüpfungssteuerung, Ablaufsteuerung, Zeit-, Zähl- und arithmetische Funktionen, um durch digitale oder analoge Eingangs- und Ausgangssignale verschiedene Arten von Maschinen und Prozesse zu steuern. Die Speicherprogrammierbare Steuerung und die zugehörige Peripheriegerät (das SPS-System) sind so konzipiert, dass sie sich leicht in ein industrielles Steuerungssystem integrieren und in allen ihren beabsichtigten Funktionen einsetzen lassen.*“ [6]

Zum genaueren Aufbau einer SPS gehört eine Stromversorgung, eine Verarbeitungseinheit (CPU), digitale sowie analoge I/O Anschlüsse, eine Feldbusschnittstelle und das Programmiergerät, wobei es sich dabei heutzutage eigentlich ausschließlich um einen externen Computer handelt. Die Stromversorgung PS (**P**ower **S**upply) wandelt gleichzeitig die Netzspannung in eine 24-V-Gleichspannung um, mit der die Elektronik der SPS versorgt wird.
Man unterscheidet mittlerweile drei verschiedene Aufbauarten bei SPSen

1. Hardware - SPS
2. Slot - SPS
3. Soft - SPS

**zu 1.)** Der im vorigen Abschnitt beschriebene Aufbau einer SPS bezieht sich auf die klassische Aufbauform einer *Hardware - SPS*. Ihre Komponenten sind als gewöhnliche Einsteckkarten in einem Gehäuse oder Schaltschrank angeordnet und sind über einen Rückwandbus miteinander verbunden. Die Hardware - SPS benötigt einen externen PC als Programmiergerät.

**zu 2.)** Eine Slot - SPS ist eine Einsteckkarte für den PC, welche alle Module einer SPS enthält. Anstatt einer CPU befindet sich ein Co-Prozessor in ihr, auf dem ein eigenes multitaskingfähiges Betriebssystem läuft. Zusätzlich verfügt sie

über einen sogenannten multi-ported RAM (ein geteilter Speicher, der sowohl für SPS, als auch für PC zugreifbar ist).

**zu 3.)** Zu guter Letzt gibt es die Soft - SPS, welche im Gegensatz zu den anderen Steuerungseinheiten eine reine Softwarelösung ist, die komplett auf der CPU des Host-PCs läuft und auch deren Hardware nutzt. Zur Ankopplung der Sensoren und Aktoren wäre eine Einsteckkarte zur Feldbuskopplung notwendig, die mit einem Prozessor zur Buskommunikation und einem dual-ported Ram ausgestattet ist.

Die Vorteile der SPS im PC ergeben sich hauptsächlich dadurch, dass die rasante Entwicklung der PC-Leistungen für SPSen genau dafür genutz werden kann.

Die Informationsverarbeitung in einer SPS verläuft zyklisch. Die Verarbeitungsschritte lassen sich vereinfacht mit dem EVA - Prinzip beschreibe.

- **E**inlesen der Sensordaten
- **V**erarbeiten der Informationen im SPS-Programm
- **A**usgeben der Soll-Werte an die Aktoren.

Anfangs fragt die CPU nacheinander alle Eingangskanäle ab und legt anschließend die Daten in den Arbeitsspeicher - es entsteht das sogenannte „Eingangsabbild“. Hierbei handelt es sich jedoch nicht um die aktuellen, sondern um die zum Abtastzeitpunkt ausgelesenen Werte. Die erstellten Programme werden dann von der CPU jeweils Schritt für Schritt abgearbeitet. Erst nach Abarbeitung aller Programme werden die im Ausgangsabbild abgelegten Sollwerte nacheinander an die Ausgangskanäle übertragen. [8]

Kleinere SPS - Hersteller versuchen ihre eigene Hardware möglichst offen für fremde Software zu produzieren, um mit ihrem Produkt eine möglichst große Zielgruppe anzusprechen. Große Hersteller hingegen möchten ihr System eher geschlossen für den Zugriff mittels fremder Software halten, um die Kundenbindung auf den verschiedensten Ebenen der Automatisierungspyramide zu erhöhen.

## 2.2 Programmierung einer SPS

In der Automatisierungs- und Regelungstechnik gibt es zum Erfüllen der gegebenen Anforderung mehrere Wege um diese zu lösen. Genauer gesagt handelt es sich hierbei um ganze sechs verschiedene Programmiersprachen, um auf jede möglich auftretende, sowie spezifische Anforderung individuell eingehen zu können. Diese unterteilen sich wiederum in zwei differenzierbare Untergruppen. So gibt es zum einen die textbasierten Programmiersprachen

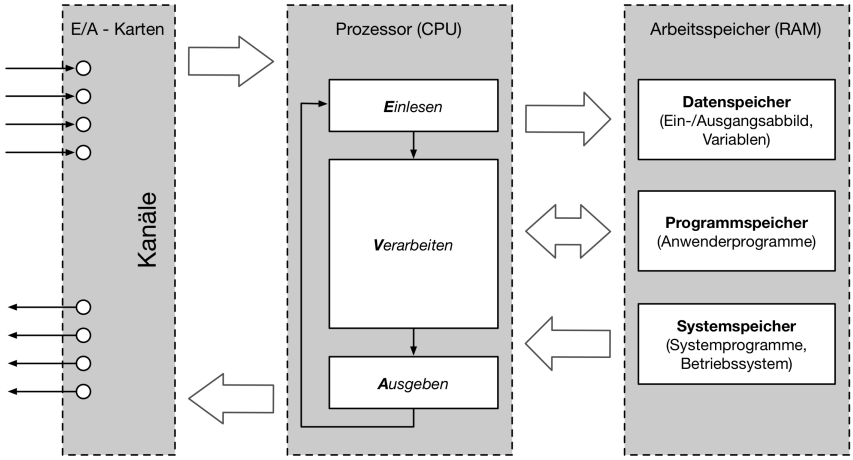


Abbildung 2.2: Signalverarbeitung und Arbeitsweise einer SPS

a) Anweisungsliste (engl. *Instruction List*)

b) Strukturierter Text (engl. *Structured Text*)

c) Ablaufsprache (engl. *Sequential Function Chart*)

Im Gegensatz dazu gibt es noch die grafikbasierten Programmiersprachen

d) Kontaktplan (engl. *Ladder Diagram*)

e) Ablaufsprache (engl. *Sequential Function Chart*)

f) Funktionsbausteinsprache (engl. *Function Block Diagram*)

Ziel dieser Vielfalt ist es, eine Vereinheitlichung der Programmierung von SPSen zu erreichen. Der Standard 61131 ist seit 1993 eingeführt und industriell etabliert. Mittlerweile schon in der dritten Edition verfügbar, und somit auch Objektorientierung unterstützend, sind die Programmiersprachen für zentrale & eng gekoppelte Systeme ausgelegt.

**zu a) - Anweisungsliste**

Diese textbasierte Programmiersprache nach der Norm IEC DIN EN 61131-3 ist sehr maschinennahe. Vergleicht man AWL mit höheren Programmiersprachen der Informatik, so ist es eine Art Assemblersprache, die normalerweise 1:1 den jeweiligen Maschinencode übersetzt. Es werden die einzelnen Anweisungen in der Reihenfolge geschrieben, wie sie die Maschine (CPU) abarbeiten soll (auch „Stackorientierte Abarbeitung“ genannt). Ein großer Vorteil gegenüber allen graphischen Programmiersprachen ist die Tatsache, das AWL funktionell über diese

hinausgeht, weil beispielsweise ein komplexer Zählvorgang mittels eines Kontaktplans nicht realisierbar sein könnte.[3,4]

**Beispiel einer Anweisungsliste**

```
VAR
s1: BOOL; //[input] Sensor Deklaration
s2: BOOL; //[input] Sensor Deklaration
s3: BOOL; //[input] Sensor Deklaration
M: BOOL := 0; //[output] Motor Deklaration (muss initialisiert werden)
END_VAR

LD s1; //[load] Lade den boolean-Wert ins Register
OR s2; //Oder-Verknüpfung mit s1
ANDN s3; //Und-Verknüpfung mit invertiertem Eingang
ST //[store] Ergebnis wird in die Variable Motor geschrieben
```

**zu b) - Strukturierter Text**

Diese Programmiersprache der Automatisierungstechnik orientiert sich an der Sprache Pascal, enthält aber neben dieser Sprache zugehörigen spezifischen Elementen aber auch noch SPS-typische Elemente. Geeignet ist ST am vorteilhaftesten für Aufgaben mit mathematischem Hintergrund, sowie zum Beschreiben komplexer Algorithmen. Auch für Rezept- und Datenverwaltung hebt sich diese Art der Programmierung durch enorme Vereinfachung vor. Typische Anweisungen für ST sind solche, die in höheren Sprachen durch Bedingungen oder Schleifen ausgeführt werden können.[5]

**Beispiel eines strukturierten Textes**

```
VAR
s1: BOOL; //[input] Sensor Deklaration
s2: BOOL; //[input] Sensor Deklaration
s3: BOOL; //[input] Sensor Deklaration
M: BOOL := 0; //[output] Motor Deklaration ( muss initialisiert werden)
END_VAR

M := (s1 OR s2) AND ( NOT (s3)); //ODER-Verknüpfung von s1 und s2, anschließendes UND-Verknüpfung von Ergebnis mit negiertem s3.
```

**zu c, e) - Ablaufsprache**

Eine Sonderstellung unter den Sprachen zur Programmierung einer SPS nimmt die Ablaufsprache ein.

**zu d) - Kontaktplan**

Die Darstellungsart des Kontaktplans ermöglicht SPS-Programmierern ein Programm auf graphischer Ebene zu erstellen und darzustellen. Ein KOP ist einem Stromlaufplan sehr ähnlich, um Programmieranfängern, die beispielsweise nur in der Elektronik tätig sind/waren und noch nie zuvor analytisch hinterfragten Code entwickelt haben, dein Einstieg zu erleichtern. Es werden Elemente wie Spulen, Öffner/Schließer, Eingänge/Ausgänge, usw... verwendet, die zu logischen Blöcken zusammengefasst werden können und so einen Teil des gesamten Programms ergeben. Ein Nachteil dieser standardisierten Programmiersprache ist jedoch, dass es nicht für alle möglichen Operationen auch ein einheitliches Symbol in einem Stromlaufplan gibt. Das wiederum bedeutet, dass bei komplexen Steuerungen oft eine Mischung aus KOP und der Funktionsbausteinsprache (FBS) verwendet wird.

**Beispiel eines Kontaktplans**

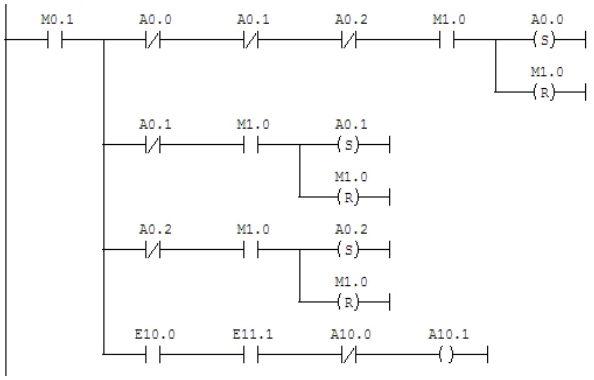


Abbildung 2.3: Kontaktplan

**zu f) - Funktionsbausteinsprache**

Diese graphische Programmiersprache verwendet für ihre Anweisungen logische Symbole der Boolschen Algebra. Insbesondere ist diese Sprache für Verknüpfungssteuerungen geeignet und vorallem bei Anfängern oder Nicht-Programmierern beliebt. Durch den einfachen graphischen Aufbau ist die Programmlogik relativ schnell zu erkennen und nachzuvollziehen.

**Beispiel eines Funktionsbausteinsplans**

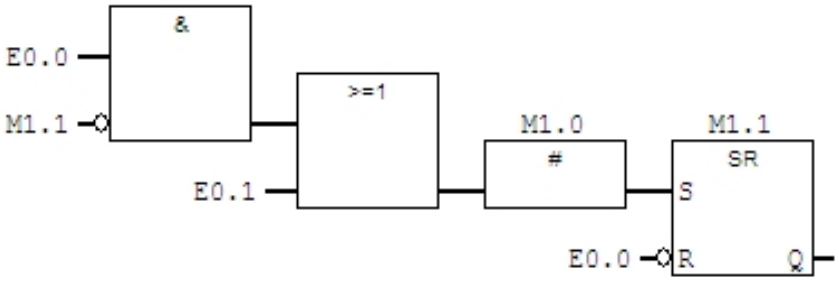


Abbildung 2.4: Funktionsbausteinplan

## 2.3 Phasen und Rezepte

TODO

## 2.4 Visualisierung von Produktionsanlagen

### 2.4.1 Anlagen Visualisierung

### 2.4.2 SCADA

(Supervisory Control And Data Acquisition)
Ein SCADA System ist im Allgemeinen eine Sammelstelle für alle aus der Anlagen generierten Werte. Diese Daten müssen dann weiterverarbeitet werden um aus ihnen Analysen zu erstellen.

Zu den Elemente eines SCADA Systems gehören:

**SCADA Master Station Computer Systems**
Sammelstelle für Echtzeitdaten die von RTUs generiert werden. Meist herkömmliche Computer Hardware.

**Human-Machine Interface**
Aus den gesammelten Daten Analysen (Prognosen, Diagnosen) so aufbereiten das sie für den Menschen leicht verständlich dargestellt werden.

**Remote Terminal Units (RTUs)**
Sensoren die Physikalische Änderungen in der Anlage mit einem Signalumformer

in Elektrische Werte umwandelt. Je nachdem was gemessen werden soll, entstehen Analoge (Füllstand, Helligkeit, Druck, ...) oder Digitale (z.B. Status eines Geräts) Werte.

## 2.5 Konzeptioneller Entwurf, Modellbasierte Entwicklung

Das Gebiet der Modellbasierten Entwicklung bzw. der automatisierten Codegenerierung aus so einem Modell als ganzes kann man nicht wirklich als State of the Art erklären.

Wenn man sich jedoch die Gebiete als einzelnes anschaut, erkennt man das diese als einzelnes schon heute im Betrieb eingesetzt werden.

### 2.5.1 Modell

Ein Modell der gesamten Anlage wird schon im frühen Stadium der Entwicklung in Form eines RI-Fließschemas hergestellt um klar zu definieren welche Bauteile benötigt werden und welches mit welchem Bauteil verbunden ist. Dies ist nur das erste Modell das in der Industrie eingesetzt wird.

Oft werden aber mehr Informationen benötigt als im RI-Fließschema abbildbar sind, weswegen man auf eine andere Art umsteigen muss ein Modell abzubilden. Auf die gängigsten Methoden (XML,Ontologie,UML) wird in den nächsten Punkten genauer eingegangen.

#### 2.5.1.1 XML

XML ist eine Standardisierte Sprache in der die meisten Strukturen in irgend einer Form abgebildet werden können. Dieser Aspekt macht es zu einem Universalen Standard der in verschieden Sparten eingesetzt wird.
Ein Nachteil von XML ist es, dass es sich hierbei um eine Textuelle Form eines Modells handelt was für Menschen schwerer zu verstehen ist als eine Visuelle Abbildung.
Dafür gibt es die nächsten 2 Werkzeuge die beide auf XML basieren.

#### 2.5.1.2 Ontologie

> ... we have at least two parts to the overall philosophical project of ontology: first, say what there is, what exists, what the stuff is reality

> is made out off, secondly, say what the most general features and relations of these things are.

Eine Ontologie ist ein auf XML basierendes Konstrukt das einem das Abbilden von Dingen und deren Relationen ermöglicht. Um solch Komplexe Zustände darzustellen stehen einem neben einfachen Entitäten(Dinge) auch Eigenschaften zur verfügen.

**Allgemeiner Aufbau einer Ontologie**
Das Modell an sich ohne zusätzliche Attribute oder Eigenschaften ist vom Aufbau her so zu erklären, dass es ein Thing gibt von dem alles aus geht (Sozusagen der Ursprung von allem). Von diesem Punkt aus können weitere Entitäten abgeleitet werden (Eine Vererbung, vergleichbar mit einem Stammbaum). Die neu erstellte Entität ist immer noch ein Thing, jedoch genauer beschrieben. Diese Vererbung wird dann soweit wie benötigt weitergeführt.

**Object Properties**
Object Properties sind Eigenschaften die die Relationen zwischen den Einzelnen Objekten/Entitäten genauer beschreiben.

**Data Properties**
Data Properties sind Eigenschaften die die Entitäten genauer beschreiben bzw. erklären.

Dies ermöglicht ein Modell aufzubauen das nicht nur Triviale Daten wie: Welche Bauteile gibt es..., abbilden kann, sondern komplexe Strukturen zu konstruieren. Ein Anwendungsfall bei einem Modell für eine Produktionsanlage ist es etwaigen Sensoren direkt im Modell einen Wertebereich festzulegen in dem dieser Sensor arbeitet bzw. Werte liefert, Beispiel hierfür: Ein Füllstandsensor mit einem Wertebereich von 4-20V, wobei 4V=Leer und 20V=Voll bedeutet, kann in einer Ontologie mit wenigen Data Properties abgebildet werden.

#### 2.5.1.3 UML

UML ist eine Methode die in vielen verschiedenen Gebieten Einsatz findet da sie sehr Unabhängig aufgebaut ist. Man kann sowohl in der Softwareentwicklung ganze Programme abbilden als auch eine ganze Produktionsanlage. Anders als bei Ontologien gibt es bei UML jedoch andere Modelle als nur eine Abbilden der Anlage, beispielsweise kann in einem Aktivitätsdiagramm eine Phase oder ein ganzen Rezept abgebildet werden.

### 2.5.2 Automatische Codegenerierung

TODO

## 2.6 Fazit

KAPITEL 3

# Konzept/Architektur

## 3.1 Einleitung

## 3.2 Modell der Anlage

## 3.3 Diagramm der Phasen

## 3.4 Codegenerierung

TODO

KAPITEL 4

# Implementierung

## 4.1 Einleitung

## 4.2 Auslesen des Modells

## 4.3 Codegenerierung mittels Wizzard

TODO

KAPITEL 5

# Evaluation

## 5.1 Erfolgreiche Codegenerierung

TODO

KAPITEL 6

# Zusammenfassung und Ausblick

TODO

# ANHANG A

## Appendix

TODO

# Literaturverzeichnis

[1] W. Lepuschitz, *Self-reconfigurable Manufacturing Control based on Ontology-Driven Automation Agents*, 2014.

[2] W. Lepuschitz, B. Groessing, E. Axinia, and M. Merdan, "Phase agents and dynamic routing for batch process automation."

[3] SPS Lehrgang
06.02.2015 - 12:29
http://www.sps-lehrgang.de/programmstruktur/

[4] SPS - Speicherprogrammierbare Steuerungen als Bausteine verteilter Automatisierung
Autor: Eberhard E. Grötsch
Seite 69 - „Die Listensprache Anweisungsliste (AWL) vor dem Standard IEC 61131-3"

[5] Grundlagen der Automatisierungstechnik
06.02.2015 - 13:10
http://www.ipsta.de/download/automationstechnik/Kap13_ST20.pdf

[6] SPS Programmierung
02.03.2015 - 15:36
http://www.sps-lehrgang.de/sps-programmierung/

[7] Vorlesung: Industrielle Systemtechnik, Kapitel: Verknüpfungssteuerung , Seite 9 - SPS-Automatisierungsgerät
11.03.2015 - 14:24
IndustrielleSteuerungstechnik-SS06.pdf

[8] Speicherprogrammierbare Steuerung
System- und Programmentwurf für die Fabrik- und Prozessautomatisierung, vertikale Integration
Autor: Matthias Seitz
Verlag: Hanser
*2. vollständig überarbeitete und erweiterte Auflage*

[9] Siemens
03.02.2015 - 13:32
http://w3.siemens.com/mcms/automation/de/ automatisierungssysteme/ automatisierungssoftware/Seiten/Default.aspx?stc=wwcg100253

[10] Rockwell
03.02.2015 - 14:11
http://ab.rockwellautomation.com/de/Programmable-Controllers

# Erklärung

Hiermit erklären wir, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.
Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, im April 2015

Name1

Name2

Name3

Name4