Arbeitspaket 7: Prozessmodell

Das Prozessmodell ist für den korrekten Ablauf des Prozesses zuständig, Arbeitsplan, Sequenzen und Skills sind Teil des Prozessmodells. Das Prozessmodell ist so definiert, dass dieses anlagenunabhängig aufgebaut werden kann. Das Modell muss nicht wissen, welche Komponenten im System vorhanden sind, sondern nur die Fähigkeiten dieser. Dadurch können Prozess- und Anlagenmodell parallel entwickelt werden, was die Kosten und Entwicklungszeit verringern kann. Diese Trennung ermöglicht eine parallele Entwicklung von Prozess- und Anlagenmodellen, wodurch sowohl Kosten als auch Entwicklungszeiten reduziert werden können. Zudem erlaubt die klare Trennung eine flexible Anpassung oder den Austausch von Komponenten im Anlagenmodell, ohne das Prozessmodell zu beeinflussen. Umgekehrt können neue Prozesse schnell und unkompliziert implementiert werden, ohne dass detaillierte Kenntnisse der Anlagenkomponenten erforderlich sind.

Diese Flexibilität führt insgesamt zu einer effizienteren und schnelleren Entwicklung sowie Inbetriebnahme der Software.

**Struktur von Prozessmodell**

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte BeschreibungDie grundlegende Struktur des Prozessmodells lässt sich klar und einfach definieren: Der gesamte Prozess wird durch einen Arbeitsplan abgebildet. Dieser besteht aus verschiedenen Sequenzen und Skills, wobei Sequenzen aus mehreren Skills zusammengesetzt sind.

Skills repräsentieren die grundlegenden Funktionalitäten einzelner Komponenten, während Sequenzen die Funktionen des Gesamtsystems oder von Teilsystemen darstellen.

Das Prozessmodell verfügt über zwei zentrale Schnittstellen: zum Anlagenmodell und zum HMI. Die Interaktion mit dem Anlagenmodell erfolgt dabei ausschliesslich über die Skills. Über das HMI wird der Arbeitsplan gesteuert, wodurch eine intuitive Bedienung und Anpassung des Prozesses ermöglicht werden soll.

Die Struktur des Prozessmodells beeinflusst massgeblich die Flexibilität bei der Erstellung und Nutzung eines Arbeitsplans. Es stellt sich die grundlegende Frage, ob das Ziel darin besteht, über ein HMI flexibel Abläufe zusammenstellen zu können, oder ob lediglich ein vordefinierter Ablauf gestartet werden soll. Beide Ansätze erfordern unterschiedliche Programmstrukturen:

1. **Flexible Erstellung über das HMI:** Diese Variante ermöglicht eine dynamische Konfiguration von Abläufen direkt über die Benutzeroberfläche. Hierfür ist eine modulare und erweiterbare Programmstruktur notwendig, die es erlaubt, Sequenzen und Skills zur Laufzeit zu definieren und zu kombinieren.
2. **Vordefinierte Abläufe:** Hier werden feste Abläufe vorab definiert und können mit minimalem Aufwand gestartet werden. Dies erfordert eine weniger dynamische, dafür jedoch robuste Struktur, die auf vorgefertigten Arbeitsplänen basiert.

Im Folgenden werden mögliche Strukturen betrachtet, die in TwinCAT implementiert werden können, um beide Anforderungen optimal umzusetzen.

**Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung**

**Ansatz 1:** Der klassische Aufbau für Rezeptursteuerungen

Ansatz 1 folgt einer bewährten Methode, wie sie bereits in TwinCAT für Rezeptursteuerungen eingesetzt wird. Hier werden die Prozessabläufe durch vordefinierte Schrittabfolgen realisiert, die je nach Bedarf ausgelöst werden. Über das HMI erfolgt lediglich der Start des Prozesses sowie die Eingabe der erforderlichen Parameter.

Vorteile: Klare und übersichtliche Struktur.

Einfache Implementierung dank zahlreicher Referenzen und standardisierter Konzepte.

Schnelle Umsetzbarkeit.

Nachteile: Eingeschränkte Flexibilität: Abläufe können nicht dynamisch angepasst oder neu erstellt werden.

Begrenzte Übertragbarkeit auf vollständig flexible Prozesse.

**Ansatz 2:** Dynamische Prozessgestaltung durch Skills und Sequenzen

Ansatz 2 bietet eine wesentlich flexiblere Gestaltung der Prozessabläufe. Über das HMI können Abläufe frei durch die Kombination von Skills und Sequenzen definiert werden. Jeder Skill wird durch eine eindeutige Nummer repräsentiert, während Sequenzen als Arrays von Nummern aufgebaut sind, die den Prozessablauf abbilden. Ein Programm im Prozessmodell analysiert dieses Array und führt die Skills in der definierten Reihenfolge aus.

Vorteile: Hohe Flexibilität: Prozesse können individuell und dynamisch über die HMI erstellt werden.

Anpassbar an variable Anforderungen oder neue Arbeitsabläufe.

Nachteile: Höhere Komplexität bei der Implementierung.

Mögliche Herausforderungen bei parallelen Prozessen, insbesondere in Bezug auf Synchronisation und Timing.

Zunächst wird die Struktur gemäss Ansatz 1 umgesetzt, da sie bewährt und schnell realisierbar ist. Je nach dem welche Erfahrungen mit dieser Struktur gemacht werden, kann in einer zweiten Iteration auf Ansatz 2 umgestellt werden, um mehr Flexibilität zu ermöglichen. Die Skills bleiben in beiden Ansätzen unverändert und können somit nahtlos übernommen werden.

**Struktur von Arbeitsplänen nach Ansatz 1:**

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Die Struktur eines Arbeitsplans ist klar und übersichtlich. Ein Arbeitsplan wird als eigenständiges Programm (PRG) definiert, das den Prozess in Form eines Schrittablaufs abbildet. Jeder Schritt führt entweder Skills oder Sequenzen aus. Innerhalb des Programms werden verschiedene Variablen und Funktionsbausteine instanziiert, um den Ablauf zu steuern.

Die Bedienvariablen dienen zum Starten, Stoppen und Zurücksetzen des Schrittablaufs. Prozessvariablen enthalten alle prozessrelevanten Informationen, die für die Ausführung des Arbeitsplans erforderlich sind. Zusätzlich müssen im Arbeitsplan alle benötigten Skills und Sequenzen instanziiert werden. Dabei werden ausschliesslich die Skills instanziiert, die im Schrittablauf des Arbeitsplans tatsächlich verwendet werden. Skills, die innerhalb von Sequenzen genutzt werden, werden direkt in den Sequenzen selbst instanziiert.

**Struktur von Sequenzen**

Ein Bild, das Text, Screenshot, parallel, Diagramm enthält.

Automatisch generierte BeschreibungSequenzen werden analog zu Skills als Funktionsbausteine definiert, wobei auch hier das Ziel darin besteht, eine einheitliche Struktur bereitzustellen. Ein zentraler Bestandteil dieser Struktur ist die Definition der Schnittstellen einer Sequenz, die in zwei Kategorien unterteilt wurden. Die Steuerungselemente sind für die Bedienung der Sequenz vorgesehen. Die Sequenz wird gestartet, gestoppt oder resettet. Zusätzlich werden auch Informationen über den Zustand der Sequenz angegeben. Der Arbeitsplan startet hierbei die Sequenz oder resettet diese. Über den Zustand weiss der Arbeitsplan, wann eine Sequenz abgeschlossen wurde.

Die Betriebsvariablen sind für den Betrieb der Sequenz angedacht. Im Gegensatz zum Skill wurden die Prozessparameter hier direkt in die Betriebsvariablen integriert.

Die Struktur der Sequenz hat während der Entwicklung mehrere Iterationen durchlaufen. In der Dokumentation wird jedoch ausschliesslich der aktuelle Stand beschrieben, da nicht alle Entwicklungsschritte detailliert erläutert werden. Während der Entwicklungs- und Testphase wurden zahlreiche Erkenntnisse und Erfahrungen gesammelt, die die Struktur massgeblich beeinflusst haben. Viele Anpassungen waren notwendig, um TwinCAT-spezifische Probleme zu bewältigen. Dabei konnte wertvolles Know-how über TwinCAT und dessen Arbeitsweise gewonnen werden.

Die Steuerungselemente umfassen die gleichen Methoden und Eigenschaften wie die Skills, die Umsetzung dieser unterscheidet sich jedoch.

|  |  |  |  |
| --- | --- | --- | --- |
| Art: | Bezeichnung: | Typ: | Beschreibung: |
| Methode | M\_Start | BOOL | Methode zum Starten der Sequenz und des Schrittablaufs |
| Methode | M\_Stop | BOOL | Methode zum Beenden des Schrittablaufs (Intern) |
| Methode | M\_Reset | BOOL | Methode zum Resetten der Sequenz und des Schrittablaufs |
| Eigenschaft | P\_State | INT | Eigenschaft des aktuellen Zustandes |

**Umsetzung in TwinCAT**

**Allgemeine Struktur**

Innerhalb der Umsetzung wurden zwei verschiedenen Strukturen getestet.

**Struktur 1:** Skills instanziiert in separatem Programm

In der ersten Struktur wurden die Skills in einem Programm namens «SkillSet» instanziiert, vergleichbar mit den Objekten innerhalb der Anlage. Diese instanziierten Skills können anschliessend vom Arbeitsplan oder über Sequenzen verwendet werden. Der Arbeitsplan greift dabei direkt auf die im SkillSet-Programm instanziierten Skills zu. Sequenzen hingegen erhalten die Skills über Eingangsvariablen, die mit dem entsprechenden Interface definiert wurden.

**Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung**

Der Vorteil dieser Struktur liegt darin, dass die Skills kontinuierlich ausgeführt werden, wodurch jederzeit der aktuelle Zustand eines Skills festgestellt werden kann. Die Interaktion mit den Skills wird durch diese Struktur einfacher und transparenter. (Probleme bei der Interaktion werden im Zusammenhang mit der zweiten Struktur genauer beschrieben.)

Ein Nachteil dieser Struktur besteht jedoch darin, dass alle benötigten Skills vorab im SkillSet-Programm instanziiert werden müssen. Beispielsweise: Wenn ein Roboter 10 Skills besitzt und das System um einen zweiten Roboter erweitert wird, müssen auch für diesen zweiten Roboter die 10 entsprechenden Skills im SkillSet-Programm instanziiert werden. Dadurch entsteht eine gewisse Abhängigkeit des Prozessmodells vom Anlagenmodell.

Ein weiteres Risiko ist die mögliche Verwechslung von Skills, insbesondere bei zwei gleichartigen Skills für unterschiedliche Komponenten. Wenn beispielsweise Roboter A angesprochen werden soll, könnte versehentlich der Skill von Roboter B ausgewählt werden.

**Struktur 2:** Instanziierung von Skills innerhalb von Arbeitsplänen und Sequenzen

Bei dieser Struktur werden ausschliesslich die Skills innerhalb des Arbeitsplans oder der Sequenz instanziiert, die tatsächlich benötigt werden. Auch die Zuweisung der Objekte erfolgt direkt innerhalb des Arbeitsplans oder der Sequenz. Eine Interface-Implementierung der Skills ist hierbei nicht erforderlich. Der entsprechende Skill wird innerhalb eines Schritts aufgerufen, alle notwendigen Parameter (z. B. das Objekt) werden zugewiesen, und der Skill wird ausgeführt.

**Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung**

Ein wesentlicher Vorteil dieser Struktur besteht darin, dass die Skills lokal in dem jeweiligen Arbeitsplan oder der Sequenz instanziiert werden. Ebenso erfolgt die Zuweisung der Objekte lokal, wodurch das Prozessmodell keine Informationen über das Anlagenmodell benötigt, wie etwa die Anzahl der im System vorhandenen Roboter. Stattdessen wird der benötigte Roboter einfach dem entsprechenden Skill innerhalb des Arbeitsplans oder der Sequenz zugewiesen. Diese Herangehensweise verbessert zudem die Übersichtlichkeit und Verständlichkeit der Software: Durch die instanziierten Skills lässt sich schnell erkennen, welche Aufgaben eine bestimmte Sequenz oder ein Arbeitsplan ausführt.

Ein Nachteil dieser Struktur ist jedoch, dass beim Aufruf, der Zuweisung und der Ausführung der Skills potenziell mehr Probleme auftreten können. Es ist essenziell, ein Verständnis darüber zu haben, wie lange Informationen für die Verarbeitung benötigen und wie auf Methoden zugegriffen werden kann. Diese Struktur erfordert daher ein fundiertes Wissen über TwinCAT und dessen Funktionsweise.

**Aufbau einer Sequenz:**

Der Schwerpunkt liegt dabei, den allgemeinen Aufbau einer Sequenz zu erklären. Es werden nicht alle definierten Sequenzen aufgezeigt und beschrieben. Die Sequenzen unterscheiden sich grundsätzlich nur in den definierten Ablaufschritten und den verwendeten Skills. Zusätzlich konnten zum Zeitpunkt dieser Dokumentation konnten noch nicht alle Sequenzen komplett abgeschlossen werden.

Eine Sequenz basiert auf einem Funktionsbaustein, der in strukturiertem Text erstellt ist. Der Funktionsbaustein umfasst verschiedene Aktionen. Eine Aktion ist für den Ablauf verantwortlich, der als SFC umgesetzt wird. Die übrigen Aktionen bestehen aus Zuweisungen, die ebenfalls in strukturiertem Text geschrieben sind. Diese dienen innerhalb des Ablaufs dazu, Skills aufzurufen und Parameter an diese zu übergeben. Über den Funktionsbausteine «FB\_Basis» werden der Sequenz die Steuerungselemente vererbt.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Als Beispiel wird die Sequenz «Seq\_Teile\_Hohlen» betrachtet. Die Aufgabe diese Frequenz ist es, mit dem Roboter an eine definierte Position zu fahren und ein Teil mit dem Greifer zu packen. Der Funktionsbaustein wurde dabei wie folgt definiert:

|  |  |
| --- | --- |
|  |  |

Für die Funktionalität des Bausteins müssen zwei Prozessparameter sowie Objekte als Eingabevariablen definiert werden. Die Prozessparameter legen die Startposition und die Greifer-Backenbreite im geöffneten Zustand fest. Die erforderlichen Objekte umfassen den Roboter und den Greifer, die jeweils über das entsprechende Interface instanziiert werden.

Innerhalb der internen Variablen werden sowohl die benötigten Skills als auch Variablen zur Zuweisung der Prozessparameter initialisiert.

Im Baustein selbst wird zunächst der Ablauf „ACT\_Ablauf“ ausgeführt. Diese Aktion läuft kontinuierlich. Das Zustandshandling erfolgt durch eine CASE-Struktur, die den aktuellen Zustand verarbeitet und die entsprechenden Aktionen steuert.

Die Aktion «ACT\_Ablauf» ist wie folgt definiert:

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Zu Beginn wird der Roboter und Greifer in eine definierte Startposition gebracht. Von dieser Position fährt der Roboter nach unten und greift nach dem Teil. Als letzter Schritt bewegt sich der Roboter mit dem Teil nach oben. Ein grundsätzlich sehr einfacher und überschaubarer Ablauf. Mit der Variablen Startposition für Roboter und Greifer, kann diese Sequenz für alle Elemente der Anwendung so übernommen werden.

Jeder Schritt ruft zwei Aktionen auf. Innerhalb der ersten Aktion wird der Skill aufgerufen und die Parameter werden zugewiesen. Diese Aktion wird kontinuierlich durchgeführt, solange der Schritt aktiv ist.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Die zweite Aktion führt die die Methoden zum Starten des Skills aus. Im Moment werden die Start-Methoden mit dem SFC-Qualifizierer «P» umgesetzt. Das «P» steht dabei für «Pulse». In TwinCAT bedeutet dies, dass die Methode zwei Mal durchgeführt wird: Einmal, wenn der Schritt aktiv wird und ein zweites Mal im darauffolgenden Zyklus. Dieses Verhalten hat zu Problemen geführt. Zusätzlich kann mit dem Qualifizierer «P» auch nicht innerhalb eines SFC-Ablaufs zwei Mal, in separaten Schritten, auf eine Methode zugegriffen werden. Aus im Moment unbekannten Gründen, ist die Methode nach dem ersten Schritt gesperrt. Es stehen zwei mögliche Herangehensweisen zur Verfügung:

1. **Tiefergehende Analyse des TwinCAT-Verhaltens**: Ziel wäre es, eine Lösung für das Problem zu finden, sodass weiterhin mit dem Qualifizierer **„P“** gearbeitet werden kann.
2. **Umstellung auf den Qualifizierer „N“**: Mit **„N“** wird die Methode so lange ausgeführt, wie der Schritt aktiv ist. Dabei wird die Methode in jedem Zyklus erneut aufgerufen. Um Fehler zu vermeiden müsste die Methode so umgeschrieben werden, dass diese trotzdem nur einmal ausgelöst wird.

Diese Problematik wurde zum Zeitpunkt dieser Dokumentation noch nicht gelöst.

Die Transition zum nächsten Schritt wird basierend auf dem Zustand des Skills ausgelöst. Der Skill muss sich dafür wieder im Zustand „BEREIT“ befinden. Zusätzlich ist sichergestellt, dass der aktuelle Schritt mindestens 0,2 Sekunden lang ausgeführt wird.

Diese zeitliche Verzögerung gibt dem Skill ausreichend Zeit, um ordnungsgemäss zu starten. Ohne diese Bedingung würde die Transition sofort TRUE werden, sobald der Schritt aktiv wird, da sich der Skill bereits vor dem Start im Zustand „BEREIT“ befindet.

**Aufbau eines Arbeitsplanes**

Der Arbeitsplan wird als Programm umgesetzt, das in Form eines Schrittablaufs (SFC) strukturiert ist. Der grundlegende Aufbau entspricht dem des SFC einer Sequenz. Jeder Schritt enthält zwei Aktionen: eine Zuweisung, die als CFC realisiert ist, sowie eine Aktion zum Starten der Sequenz oder des Skills.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Bei der Variablendeklaration des Arbeitsplanes werden verschiedene Elemente instanziiert:

* Bedienparameter
* Prozessparameter
* Skill-Instanzen
* Sequenz-Instanzen

Die Bedienparameter dienen zum Start des Ablaufs. Unter Prozessparameter werden alle Parameter zusammengefasst, die für den Betrieb des Arbeitsplans erforderlich sind. Alle Informationen, die als Eingabevariablen für Skills und Sequenzen verwendet werden, müssen in diesem Bereich definiert sein. Die Instanzen von Skills und Sequenzen repräsentieren die im Arbeitsplan verwendeten Skills und Sequenzen.

Die Prozessparameter könnten über eine «Rezeptverwaltung» verwaltet werden. Damit bietet TwinCAT die Möglichkeit benutzerdefinierte Variablenlisten zu erstellen und zu verwalten. Für einen Arbeitsplan könnte dadurch einfach ein bestimmtes Rezept geladen werden, welches sämtliche Prozessvariablen enthält. Dieses Feature wurde innerhalb dieser Arbeit noch nicht implementiert.