Arbeitspaket 5: Skills

Einführungstext

**Kompetenzen von Skills**

Für die Definition eines Skills wird nicht die Funktion des Gesamtsystem so weit wie möglich in Teilfunktionalitäten heruntergebrochen, sondern die Funktionalitäten der Komponenten im System. Dabei betrachtet man jeden Komponenten-Typ, so weit wie möglich, einzeln. Der Vorteil dieser Definition ist, dass für neue oder andere Systeme dieselben Skills verwendet werden können. Ein Roboter hat in jedem System die gleichen Grundfunktionalitäten und somit Skills. Der Skill ist Anwendungsunabhängig. Die aus den Skills erstellten Sequenzen bilden die Funktion der Anwendung ab. Damit unterscheidet sich die Definierung der Skills von vergleichbaren Projekten wie z.B. der bereits durchgeführten Master-Thesis auf Basis von ROS (Ref). Dort haben Skills Funktionalitäten von mehreren Komponenten ineinander kombiniert, wodurch der Skill stärker an das spezifische System gebunden war.

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

Automatisch generierte Beschreibung**

Die Kompetenzen eines Skills lassen sich in vier Bereiche aufteilen:

|  |  |
| --- | --- |
| Zuweisung: | Der Skills ist für die Zuweisung einer Komponente zuständig. Es muss definiert werden können, welche Komponente den Skill ausführt. |

|  |  |
| --- | --- |
| Umsetzung: | Die definierte Grundfunktion muss innerhalb des Skills umgesetzt werden. Der Skill muss mittels Parameter-Inputs flexibel eingesetzt werden können. |

|  |  |
| --- | --- |
| Verarbeitung: | Die Informationen der Grundfunktion wird so ausgegeben, dass das Anlagenobjekt damit arbeiten und die reale Komponente ansteuern kann. |

|  |  |
| --- | --- |
| Auswertung: | Die momentane Situation der Komponente wird überwacht und ausgewertet. Der Skill kann auf bestimmte Situationen reagieren. |

**Definition von Anwendungs-Skills**

Um die benötigten Skills für die Anwendung zu definieren, werden im ersten Schritt die allgemeinen Arbeitsschritte basierend auf dem mechanischen Aufbau (siehe Verweis) festgelegt. Dabei wird von der Ausgangssituation ausgegangen, dass alle Teile in ihren Lagerpositionen abgelegt sind, der Roboter sich in der Home-Position befindet und alle Komponenten eingeschaltet sowie betriebsbereit sind.

|  |  |  |
| --- | --- | --- |
| Schritt | Aktivität | Komponente |
| 1 | Position von Platte 1 in Lagerung erkennen | Kamerasystem |
| 2 | Platte 1 an Montageposition bringen und mittels L-Stück ausrichten | Roboter, Greifer, Kraftsensor |
| 3 | Position von Platte 2 in Lagerung erkennen | Kamerasystem |
| 4 | Platte 2 an Montageposition bringen und mit Platte 1 zusammenführen | Roboter, Greifer, Kraftsensor |
| 5 | Position der Befestigungslöcher in den Platten ermitteln | Kamerasystem |
| 6 | Position von Befestigungsblech in Lagerung erkennen | Kamerasystem |
| 7 | Befestigungsblech an Montageposition bringen | Roboter, Greifer, Kraftsensor |
| 8 | Position von Stift 1 in Lagerung erkennen | Kamerasystem |
| 9 | Stift 1 an korrekte Position bringen | Roboter, Greifer |
| 10 | Befestigungsblech mit Platte 1 verbinden, durch Eindrücken von Stift 1 | Roboter, Kraftsensor |
| 11 | Wiederholen von Schritt 8 – 10 für Stift 2 bis 3 |  |

Eine detaillierte Auflistung der Arbeitsschritte wird im Anhang beigefügt. Aus diesem lässt sich erkennen, dass sich diverse Schritte mit kleinen Anpassungen wiederholen. Diese sich wiederholenden Arbeitsschritte definierten die Skills. Ein entscheidender Aspekt dabei ist, dass der Roboter und der Kraftsensor als separate Komponenten betrachtet werden. Der Kraftsensor erweitert die Fähigkeiten des Roboters zwar und damit dessen Skills, jedoch kann der Roboter auch ohne Kraftsensor betrieben werden. Der Kraftsensor wird als eigene Objektklasse abgebildet, jedoch besitzt dieser keinen eigenen Skill. Folgende Skills wurden definiert, welche den Prozess abdecken.

|  |  |  |
| --- | --- | --- |
| Komponente | Skill | Bemerkung |
| Kamerasystem: | - Bild aufnehmen |  |
| *(Kamera + Vision)* | - Objekt erkennen |  |
|  | - Greifposition ermitteln |  |
| Roboter: | - Position anfahren | Die Zielposition wird angegeben |
|  | - Kontrolliert bewegen | Die Bewegung wird in Echtzeit vorgegeben und mit Sensor überwacht |
| Greifer: | - Backenposition anfahren | Der Skill kann eine bestimmte Position anfahren. Der Greifer kann dadurch geöffnet oder geschlossen werden. |
| *(mit Sensoren)* |

Die definierten Skills werden innerhalb der Umsetzung (Kapitelverweis) detaillierter beschrieben.

**Definierung der Skill-Struktur**

Alle Skills sollen mit der gleichen Struktur aufgebaut und jeweils nur mit den prozessspezifischen Funktionen ergänzt werden. Ein wichtiger Aspekt dieser Grundstruktur sind die In- und Outputs, welcher ein Skill minimal benötigt und welche Zustände dieser einnehmen kann.

In einem ersten Schritt werden die relevanten Schnittstellenvariablen des Skills definiert. Dabei wird sich an den Standard von PLCopen angelehnt (Verweis). Diese definierten Variablen sind für alle Skills dieselben.

**Input-Variablen:**

|  |  |  |
| --- | --- | --- |
| Variable | Typ | Beschreibung |
| bExecute | BOOL | Trigger für Ausführung von Skill |
| iSysCommand | eSystemCommand | Steuerungsvariabel von System |
| iObjState | eObjectState | Informationen über Zustand von Objekt (Anlagenmodell) |
| iSysState | eSystemState | Informationen über System (Systemparameter) |

**Output-Variablen:**

|  |  |  |
| --- | --- | --- |
| Variable | Typ | Beschreibung |
| bDone | BOOL | Information ob Skill erfolgreich ausgeführt wurde |
| bBusy | BOOL | Information ob Skill im Moment ausgeführt wird |
| bLimit | BOOL | Information ob Skill an einem Limit angekommen ist |
| bError | BOOL | Information ob sich Skill im Moment im Fehlerzustand befindet |
| iErrorID | INT | Information um welchen Fehler es sich handelt |
| iSkillCommand | eSkillCommand | Kontrollvariable für Objekte (Anlagenmodell). |

**Eigenschaften**

|  |  |  |
| --- | --- | --- |
| Variable | Typ | Beschreibung |
| P\_State (GET) | eSkillState | Information über Zustand von Skill |

Die Output-Variable «bLimit» gibt an, ob der Skill ein definiertes Limit erreicht hat. Dies kann z.B. eine Kraft- oder Zeitvorgabe sein. Der Skill gibt dabei aber keinen Fehler an. Die Idee ist, dass der Ablauf auf diese Information reagieren kann, um eine Korrektur vornehmen zu können.

Die zwei Variablen «iObjState» und «iSysState» beschreiben den aktuellen Status des jeweiligen Objektes und des Systems und bilden somit die Schnittstellen zu diesen Elementen. Diese haben einen Einfluss auf den Zustand des Skills, welcher über die Eigenschaft «P\_State» definiert wird. Es ist wichtig die Aufgaben dieser Schnittstellen klar zu definieren. Die Interaktion zwischen Systemparameter, Skills und Objekten muss abgegrenzt sein.

**Objektschnittstelle:**

Die Objektschnittstelle regelt die Interaktion zwischen den Systemparametern und den Objekten des Anlagenmodells. Die Systemparameter steuern dabei die grundlegenden Funktionen der Objekte, wie Ein- und Ausschalten, Zurücksetzen oder Stoppen. Da diese Basisfunktionen nicht durch die Skills aktiviert werden, bleibt deren Aufgabe auf die Verwaltung des Prozesses beschränkt. Dies ist besonders sinnvoll, da ein Objekt mehrere Skills besitzen kann, und so Fragen zur Berechtigung der Skills vermieden werden. Im Gegenzug stellen die Objekte den Systemparametern Informationen über ihren Zustand und Fehler zur Verfügung.

**Modellschnittstelle:**

Die Modellschnittstelle ist für die Interaktion zwischen Prozessmodell und Anlagenmodell zuständig, genauer gesagt zwischen Skills und Objekten. Die Skills schicken Prozessbefehle («iObjControl») und Prozessparameter an das Objekt, auf welche das Objekt reagiert. Das Objekt übergibt den aktuellen Zustand («iObjState»). Zusätzlich werden auch Prozessmesswerte vom Objekt an den Skill übergeben. Die Prozessparameter und Prozessmesswerte sind nicht Teil der Grundstruktur des Skills.

**Koordinationsschnittstelle:**

Die Koordinationsschnittstelle ist für die allgemeine Prozesskoordinations verantwortlich. Es werden Information über den aktuellen Zustand und Fehler des Skills («P\_State») an die Systemparameter übergeben. Der Skill erhält den aktuellen Zustand des Systems («iSysState»). Der Skill kann somit auf systemübergreifende Situationen reagieren und das System kann auf Skill-Zustände reagieren.

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

Automatisch generierte Beschreibung

Die definierten In- und Outputs, sowie die Schnittstellen mit deren Abgrenzungen dienen als Grundlage für die Bestimmung der Zustände. Dabei werden die Zustände für das System, die Skills und die Objekte bestimmt. Die System- und Objektzustände sind entscheidend für die grundlegende Struktur der Skills, da diese auf die jeweiligen Zustände reagieren müssen. Folglich stellen die definierten System- und Objektzustände lediglich die Mindestanforderungen dar, die notwendig sind, um eine Interaktion mit den Skills zu ermöglichen.

**System:**

Das System besitzt mindestens folgende 5 Zustände:

|  |  |  |
| --- | --- | --- |
| Zustand: | | Beschreibung: |
| 0 | AUS | Das System ist ausgeschalten (Startzustand) |
| 1 | BEREIT | Das System ist eingeschalten und bereit einen Prozess durchzuführen |
| 2 | LAUFEND | Ein Prozess wird ausgeführt |
| 3 | GESTOPPT | Ein Prozess wurde gestoppt |
| 4 | FEHLER | Es gibt einen Fehler im System |

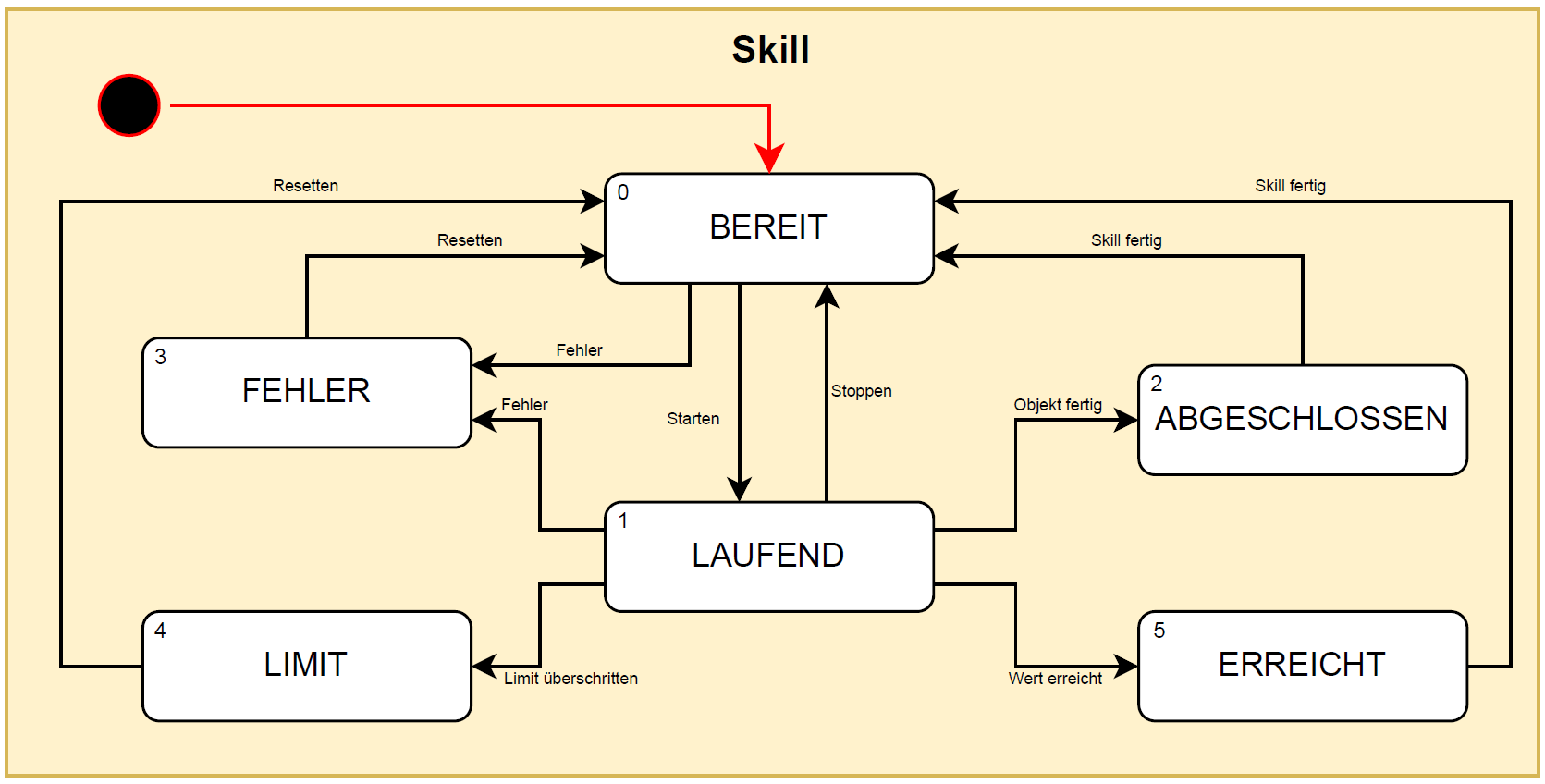
Ein Bild, das Diagramm, Text, Screenshot, Plan enthält.

Automatisch generierte Beschreibung

**Skill:**

Ein Skill besitzt 6 Zustände:

|  |  |  |
| --- | --- | --- |
| Zustand: | | Beschreibung: |
| 0 | BEREIT | Der Skill ist bereit einen Prozess auszuführen (Startzustand) |
| 1 | LAUFEND | Der Skill führt einen Prozess aus |
| 2 | ABGESCHLOSSEN | Der Prozess wurde abgeschlossen (Durch Objekt abgeschlossen) |
| 3 | ERREICHT | Prozessziel wurde erreicht und Prozess wurde durch Skill beendet |
| 4 | LIMIT | Grenzwert wurde überschritten und Prozess wurde abgebrochen |
| 5 | FEHLER | Es gibt einen Fehler bezüglich des Prozesses |

****

**Objekt:**

Ein Objekt benötigt mindestens folgende 6 Zustände:

|  |  |  |
| --- | --- | --- |
| Zustand: | | Beschreibung: |
| 0 | AUS | Das Objekt ist ausgeschalten (Startzustand) |
| 1 | BEREIT | Das Objekt ist eingeschalten und bereit |
| 2 | LAUFEND | Das Objekt ist aktiv |
| 3 | ABGESCHLOSSEN\_INTERN | Das Objekt hat den Prozess durchgeführt und hat selbständig gestoppt |
| 4 | ABGESCHLOSSEN\_EXTERN | Das Objekt wurde durch den Skill gestoppt (ERREICHT / LIMIT) |
| 5 | GESTOPPT | Das Objekt wurde durch das System gestoppt |
| 6 | FEHLER | Es gibt einen Fehler bezüglich des Objektes |

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

Automatisch generierte Beschreibung

**Konfiguration eines Skills**

Nicht jeder Skill benötigt alle Zustände. Damit der Skill so übersichtlich wie möglich bleit, sollen auch nur die Zustände verwendet werden, welche benötigt werden. Ein Skill kann drei Konfigurationen einnehmen.

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

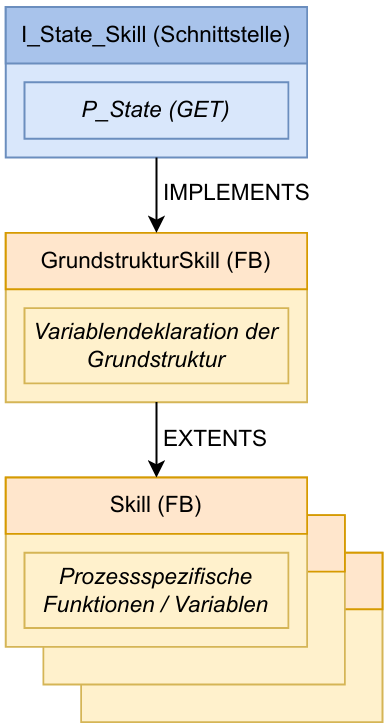
Automatisch generierte Beschreibung**

|  |  |
| --- | --- |
| Konfiguration 1: | Stellt die Grundkonfiguration dar. Jeder Skill muss diese Zustände besitzen. Hierbei handelt es sich um Skills, welche nur durch das Objekt abgeschlossen werden, z.B. eine einfache Punkt-Zu-Punkt-Bewegung des Roboters. |

|  |  |
| --- | --- |
| Konfiguration 2: | Bei dieser Konfiguration kommt der LIMIT-Zustand dazu. Dieser wird benötigt, wenn es eine Limit-Bedingung gibt, z.B. einen Grenzwert für die Kraft oder eine maximale Zeitdauer. |

|  |  |
| --- | --- |
| Konfiguration 3: | Bei der letzten Konfiguration wird der ERREICHT-Zustand ergänzt. Dieser gibt an, ob ein definiertes Ziel erreicht wurde, dies kann z.B. eine Kraft sein. |

**Umsetzung von Skills**

****Die Skill-Struktur gibt vor, dass alle Skills einen identischen Grundaufbau besitzen. Um dies zu realisieren, wird mit «Schnittstellen» und «Vererbungen» gearbeitet. Mit Schnittstellen können Methoden und Eigenschaften vorgeschrieben werden, welche zwingend umgesetzt werden müssen. Jeder Skill besitzt eine Eigenschaft zur Ausgabe des aktuellen Standes (P\_State). Diese Eigenschaft wird im Funktionsbaustein «GrundstrukturSkill» implementiert. In diesem Baustein werden alle Variablen definiert, welche alle Skills besitzen müssen. Dies umfasst die bereits definierten Ein- und Ausgangsvariablen, wie auch die internen Variablen. Die unterschiedlichen Skills können über eine Vererbung «EXTENDS» auf diese Variablen zugreifen. Die Vererbung stellt die Variablen separat für jeden Skill zur Verfügung. Ein Skill kann nicht die Variablen eines anderen Skills verändern. Durch diese Struktur wird der Aufbau eines Skill deutlich vereinfacht. Dieser wird nun noch mit Prozessspezifischen Variablen, Methoden und Eigenschaften versehen.

Neben den definierten Ein- und Ausgangsvariablen besitzen alle Skills die folgenden internen Variablen:

|  |  |  |
| --- | --- | --- |
| Variable | Typ | Beschreibung |
| Managementvariablen | | |
| iState | eSkillState | Informationen über Zustand von Skill |
| bTargetReached | BOOL | Information ob Prozessziel erreicht wurde |
| bLimitReached | BOOL | Information ob definiertes Limit erreicht wurde |
| bGestartet | BOOL |  |
| bGestoppt | BOOL |  |
| Startvariablen | | |
| fTrigger | R\_TRIG | Erkennung einer steigenden Flanke für den Start des Skills |
| fSwitch | SR | Setzten und zurücksetzen des Signals |
| Zustandsvariablen | | |
| bStarten | BOOL | Transition-Variable von BEREIT zu LAUFEN |
| bStoppen | BOOL | Transition-Variable von LAUFEND zu BEREIT |
| bObjektFertig | BOOL | Transition-Variable von LAUFEND zu ABGESCHLOSSEN |
| bWertErreicht | BOOL | Transition-Variable von LAUFEND zu ERREICHT |
| bLimitErreicht | BOOL | Transition-Variable von LAUFEND zu LIMIT |
| bSkillFertig | BOOL | Transition-Variable von ABGESCHLOSSEN / ERREICHT zu BEREIT |
| bFehler | BOOL | Transition-Variable von BEREIT / LAUFEND zu FEHLER |
| bResetten | BOOL | Transition-Variable von FEHLER / LIMIT zu BEREIT |