State diagrams (Zustandsautomaten)

Allgemeines

Zustandsautomaten geben Antworten auf die Frage „Wie verhält sich das System in einem bestimmten Zustand bei gewissen Ereignissen?“. Sie spezifizieren somit ein Verhalten mittels Zuständen und Übergängen zwischen den Zuständen, die durch interne oder externe Ereignisse initiiert werden können.

BABOK:

9.29.1 Purpose

A state diagram shows how the behavior of a concept, entity or object changes in response to events.

9.29.2 Description

A state diagram specifies a sequence of states that an object goes through during its lifetime, and defines which events cause a transition between those states. The allowable behavior of the object is dependent on its current state. There are many titles for the state diagram including State Machine Diagram, State Transition Diagram, and Entity Life Cycle Diagram.

Die formale Darstellung mit Zustandsautomaten erlaubt eine gut lesbare, nachvollziehbare und leicht wartbare Darstellung eines Verhaltens. Zustandsautomaten sind eindeutiger und weniger frei interpretierbar als eine Beschreibung in sprachlicher Form, erfordern jedoch entsprechende Kenntnis der Notation.

Aus Zustandsautomaten können vergleichsweise einfach Testfälle hergeleitet werden. Ebenso eignen sich diese Diagramme zur Generierung von Code.

Besonders geeignet sind Zustandsautomaten auch zur Beschreibung bzw. Definition von Protokollen wie beispielsweise TCP (Transmission Control Protocol).

Bei der Beschreibung eines Verhaltens mit Zustandsautomaten sind folgende vereinfachende Annahmen zu treffen:

- Das System befindet sich zu einem bestimmten Zeitpunkt in genau einem Zustand
- Der Übergang von einem Zustand in den nächsten erfolgt ohne zeitliche Verzögerung

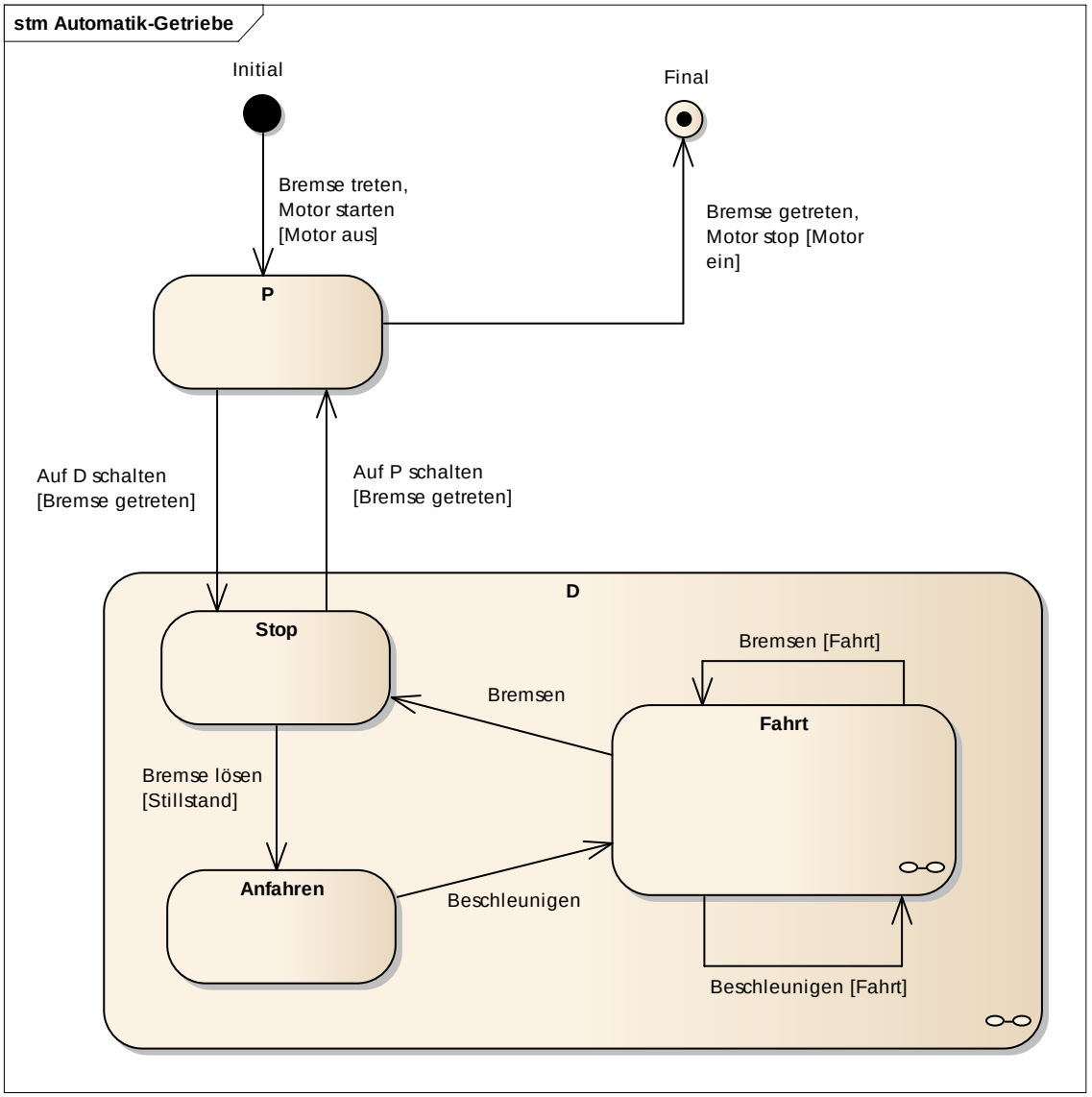
Am Beispiel eines (vereinfacht dargestellten) Automatikgetriebes eines Personenkraftwagens sehen wir die Notationselemente

- Zustände
- Übergänge (Transitions)
- Den Zustandsautomaten an sich
- Start- und Endzustände

Zusätzlich sind noch

- Regionen und
- Pseudozustände

als Notationselemente möglich.



Der Zustandsautomat beginnt mit dem Start des Fahrzeugs. Es ist dabei notwendig, die Automatik auf Stellung „P“ zubringen, die Bremse zu treten und durch Drücken eines Tasters den Motor zu starten. Der Automat ist damit im Zustand „P“ (Parken).

Zum losfahren wird die Automatik in die Stellung „D“ gebracht. Dies ist ein externes Ereignis (der Fahrer bringt den Hebel in die Stellung „D“) und dies verursacht den Übergang in den Modus „Stop“.

Wird nun die Bremse gelöst (Ereignis) beginnt das Fahrzeug zu rollen und kommt damit in den Zustand „Anfahren“. Üblicherweise wird in Folge das Gaspedal betätigt (Ereignis) und damit das Fahrzeug beschleunigt. Der Automat ist im Zustand „Fahrt“.

Im Zustand „Fahrt“ gibt es drei Möglichkeiten:

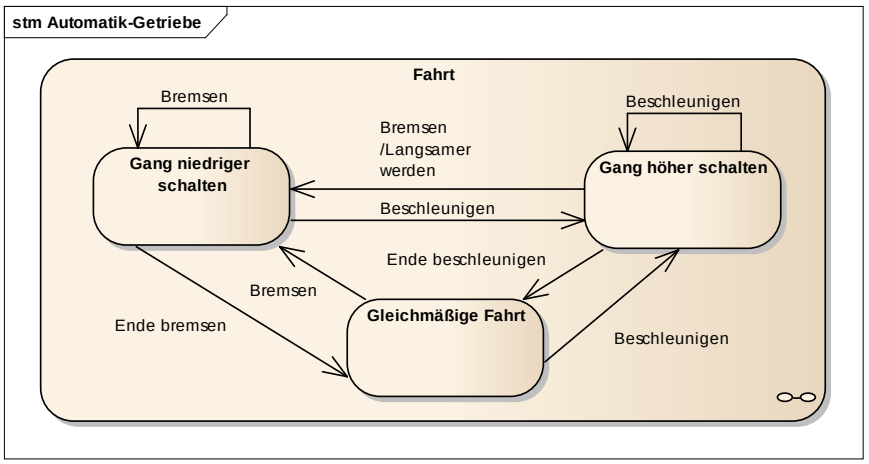
- Weiter beschleunigen (beziehungsweise Geschwindigkeit halten)
- Bremsen (Geschwindigkeit reduzieren)
- Zum Stillstand kommen (auf Geschwindigkeit 0 bremsen)

Solange das Fahrzeug nicht durch Bremsen zum Stillstand kommt, sondern nur langsamer wird, bleibt der Automat im Zustand „Fahrt“. Dies wird eine Selbsttransition genannt.

Wird solange gebremst, dass das Fahrzeug zum Stillstand kommt, geht der Automat wieder in den Zustand „Stop“ über. Zu guter Letzt kann im Zustand „Stop“ die Automatik auch wieder in die Stellung „P“ gebracht werden, womit das Fahrzeug im Zustand „P“ ist. Nur in dieser Stellung kann der Motor des Fahrzeugs durch Drücken der Start-/Stop-Taste abgeschalten werden.

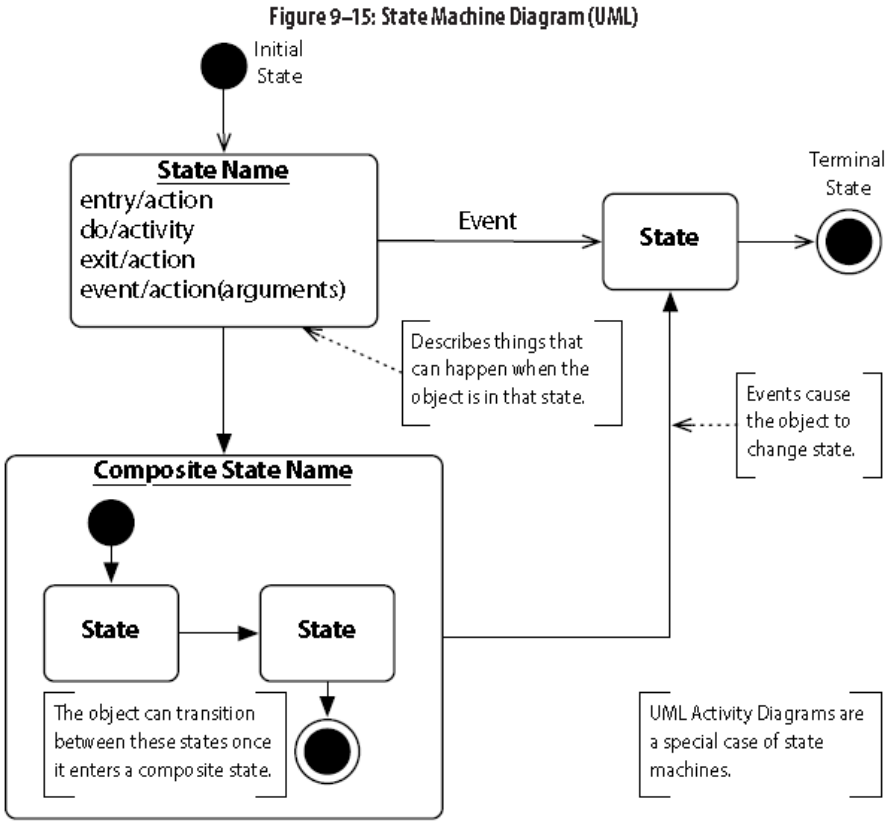
Wie aus diesem Beispiel ersichtlich ist, kann eine vollständige Darstellung eines Zustandsautomaten doch recht komplex und umfangreich werden. Vor allem wären zur Detaillierung der Funktion des Automatikgetriebes noch wesentlich mehr Angaben erforderlich wie beispielsweise

- Wann wird beim Beschleunigen der nächste Gang geschalten?
- Was genau bedeutet „Anfahren“ – mit welcher Geschwindigkeit etc.?
- Wie wirkt der Tempomat auf die Funktionen des Automatikgetriebes?



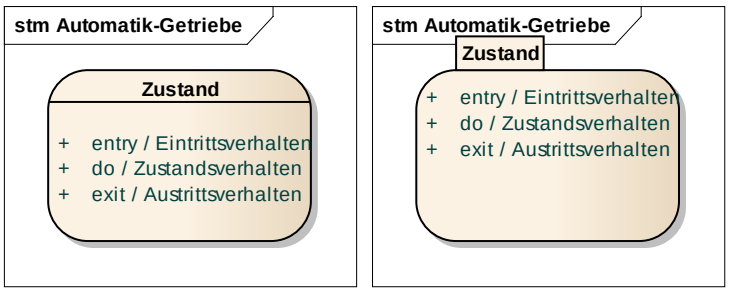
Elemente

BABOK:



Zustand

Ein Zustand wird mit folgender Notation dargestellt:



Die Bezeichnung des Zustands ist optional, wird jedoch wärmstens empfohlen. Ferner kann angegeben werden, welches interne Verhalten und welche internen Transitionen in diesem Zustand ausgeführt werden können.

Die Auslöser sind Schlüsselwörter und dürfen demnach nicht in einem anderen Kontext verwendet werden:

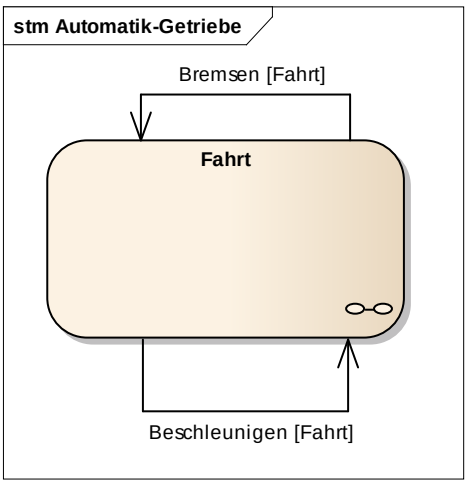
entry definiert das Eintrittsverhalten in den Zustand. Sofort nach Betreten des Zustands wird dessen Eintrittsverhalten ausgeführt.

do definiert das Verhalten innerhalb des Zustands, beschreibt also detaillierter, was innerhalb des Zustands geschieht und definiert das Verhalten, das nach Beendigung des Eintrittsverhaltens aufgerufen wird.

exit definiert das Austrittsverhalten aus dem Zustand. Beim Verlassen des Zustands wird als abschließende Aktion das Austrittsverhalten ausgeführt.

Ein Zustand kann ferner eine Menge von Triggern benennen, die innerhalb des Zustands ausgelöst werden. Mit dem speziellen Kennzeichner „defer“ können interne Transitionen – z.B. bei länger dauernden Operationen – verzögert werden.

Zustände können auch hierarchisch dargestellt werden, um mittels mehrerer Diagramme zwecks Übersichtlichkeit stufenweise in die Tiefe zu gehen:



Wie in obigem Beispiel enthalten ist der Zustand „Fahrt“ ein zusammengesetzter Zustand (gekennzeichnet durch das o-o-Symbol) und wird mit einem weiteren Zustandsdiagramm detailliert modelliert.

BABOK:

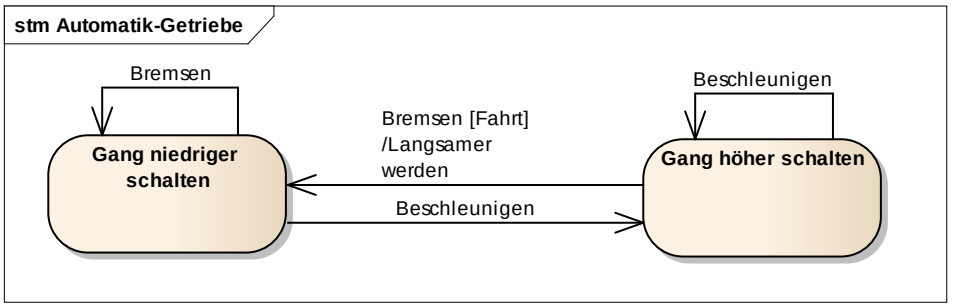
.1 States

A state represents a unique condition that an object can be in or status that it may have. All states for an object are mutually exclusive—an object can be in only one state at a time. The meaning of state is definable within the context of the business area being analyzed. Additional details of the state such as mandatory characteristics and relationships further describe the state. For example, a Canceled Project must have a cancellation date.

All state machines must have an initial state (representing the state of the object at creation) and may have any number of intermediate and end states.

Transition

Eine Transition wird durch eine durchgezogene, gerichtete und sinnvollerweise beschriftete Kante abgebildet:



Transitions können wie folgt beschrieben werden (siehe obiges Bild):

Bremsen ist der Trigger, d.h. der Auslöser für die Transition. Trigger können mit gleichem Namen mehrfach vorhanden sein, wenn sich die Guards unterscheiden.

[Fahrt] ist der Guard, d.h. eine Bedingung, die wahr sein muß, damit die Transition bei Erhalt des Triggers durchlaufen wird.

/Langsamer werden beschreibt das Verhalten, das beim Durchlaufen der Transition durchgeführt wird.

Transitions können auch als Selftransitions verwendet werden, d.h. bei Ausgangs- und Zielzustand handelt es sich um denselben Zustand (siehe obiges Bild). Das Durchlaufen einer Transition nimmt vereinbarungsgemäß selbst keine Zeit in Anspruch.

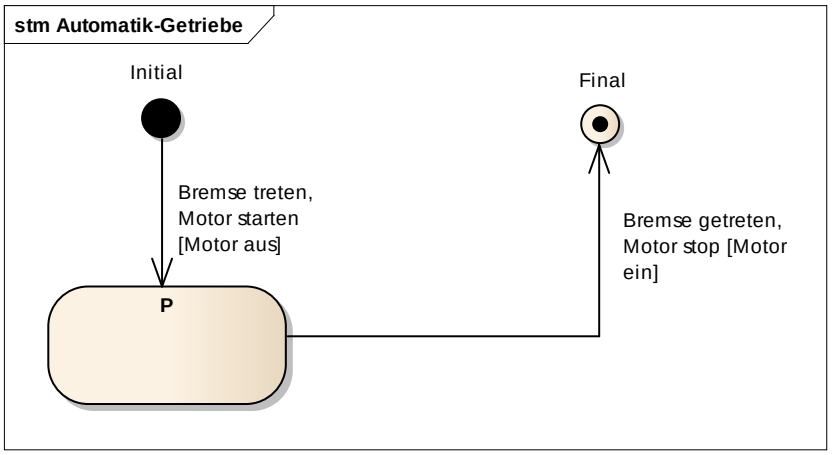
BABOK:

.2 Transitions

A transition represents dynamic behavior that moves an item from one state to another. Transitions are triggered by activities completed, events, or other stimuli. An event may only cause a transition if the object is affected by the event in its current state. In addition, business rules may determine if an object responds to a particular event.

Start- und Endzustand

Ein Startzustand wird als ausgefüllter Kreis, ein Endzustand wird als kleiner, ausgefüllter Kreis, umgeben vom einem unausgefüllten Kreis, dargestellt:



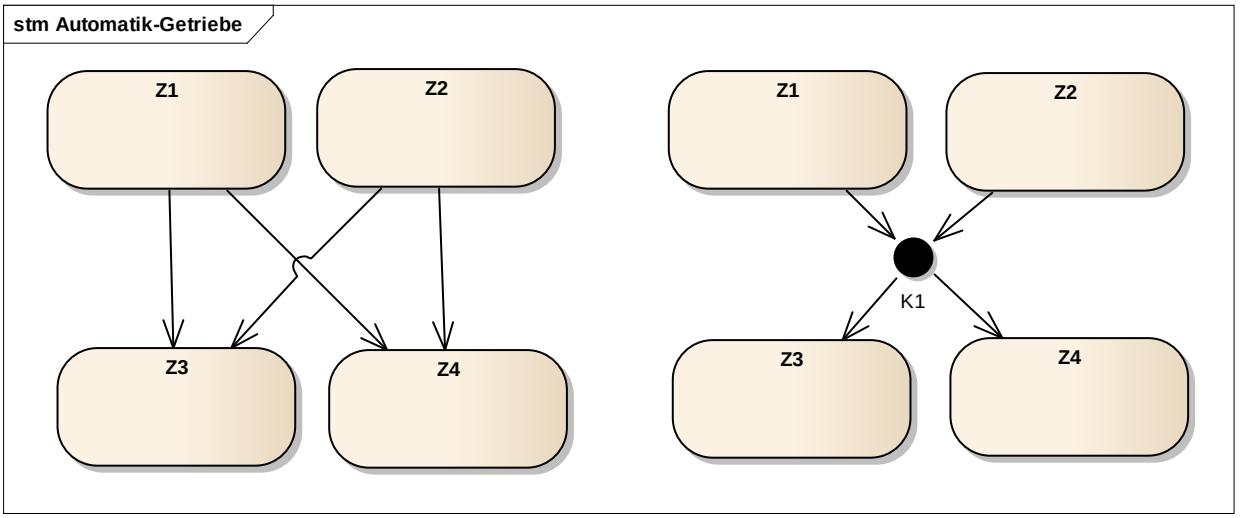
Ein Startzustand bildet den Startpunkt für das Betreten eines Zustandsautomaten. Von diesem ausgehend wird der erste Zustand des Automaten erreicht. Ein Startzustand darf keine eingehenden Transitionen besitzen. Ebenso darf es nur einen Startzustand geben.

Wenn ein Endzustand erreicht wird, ist die Abarbeitung des Zustandsautomaten beendet und es wird kein weiteres Verhalten ausgeführt. Entsprechend darf es bei Endzuständen keine ausgehenden Transitionen geben. Das Beenden des Zustandsautomaten entspricht dem Beenden der Lebensdauer eines Objekts, wenn eine Klasse mit einem Zustandsautomaten beschrieben wird.

Pseudozustände

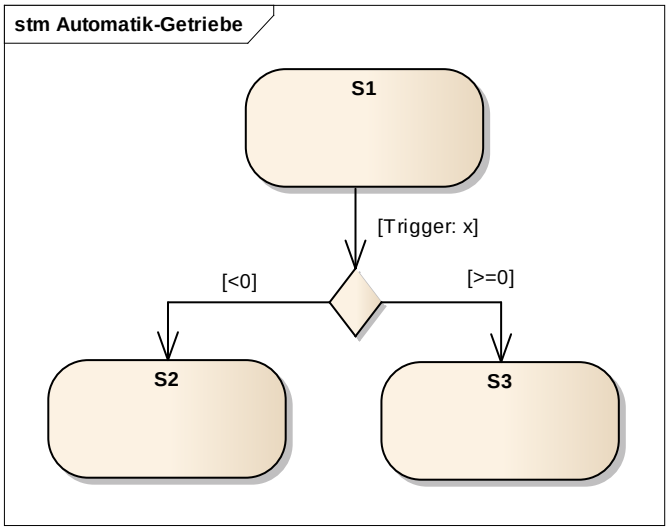
Mit Pseudozuständen können komplexe Beziehungen zwischen Zuständen einfacher dargestellt werden. Dies wird dadurch realisiert, dass Transitionen nicht nur zwischen Zuständen, sondern auch über einen oder mehrere Pseudozustände gehen dürfen.

Kreuzungen:



Beide Darstellungen sind gleichwertig. Auf den Transitions können „wie üblich“ auch wieder Guards angewendet werden.

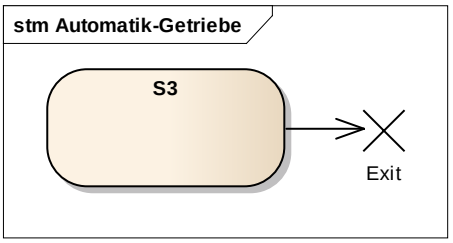
Entscheidungen:



Entscheidungen werden verwendet, wenn Abzweigstellen benötigt werden, bei denen die Auswahl der auszuführenden Transition vom Ergebnis der auf dem Weg zur Entscheidung getätigten Aktionen abhängt.

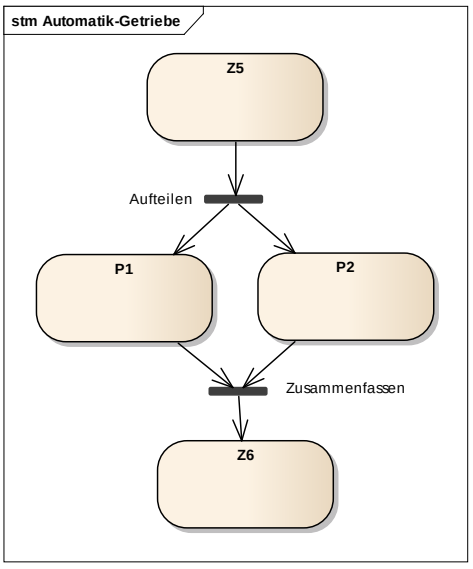
Der wichtige Unterschied zu Kreuzungen liegt darin, dass bei der Kreuzung der Weg schon vor der Ausführung der Transition festgelegt ist, während bei der Entscheidung der aktuelle Wert einer Variablen den Weg weist.

Terminator:



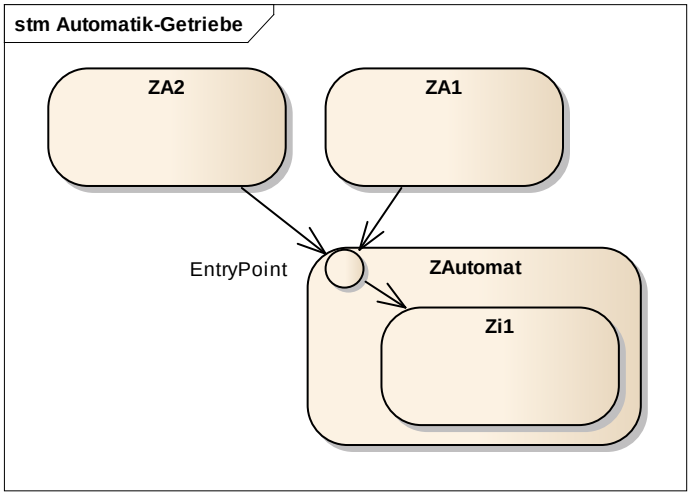
Bei Erreichen eines Terminators bricht die Ausführung des Zustandsautomaten, beispielsweise als Modellierung eines schweren Fehlers, ab. Dieses Element bietet sich an, wenn Objekte in Klassen dynamisch erzeugt und wieder gelöscht werden müssen.

Gabelung und Vereinigung:



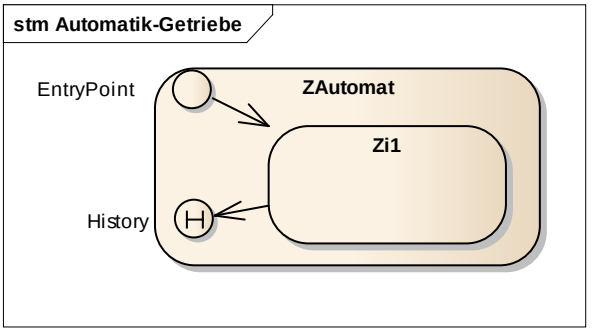
Gabelungen dienen dazu, eine eingehende auf mehrere parallele Ziele aufzuteilen. Die parallelen Bereiche werden zusammen gestartet und wieder gemeinsam beendet.

Ein- und Austrittspunkte



Eintrittspunkte dienen der Übersichtlichkeit, d.h. sie ersparen das Antragen mehrerer Transitionen von außen an einen Zustand innerhalb eines zusammengesetzten Zustands. Gleiches gilt sinngemäß für Austrittspunkte.

Historie



Mit Historienzuständen lässt sich der Zustand, der bei Betreten als Erster aktiv werden woll, dynamisch bestimmen. Dies bedeutet, dass der erste aktive Zustand davon abhängig ist, aus welchem Zustand der (zusammengesetzte) Zustand zuvor verlassen wurde.

BABOK

9.29.4 Usage Considerations

.1 Advantages

Domain SMEs should be intimately aware of life cycle states for their key concerns. Helping them list and describe the states and then draw the allowable transitions between states often uncovers missing data, control and behavioral requirements and may be helpful to clarify confusing or even conflicting requirements.

.2 Disadvantages

Since domain SMEs can understand and develop state diagrams very quickly, it is important not to unintentionally expand the scope. Each state (and associated transitions) should be validated to determine if it is relevant to the solution scope. There may be actual states an object goes through as part of its life cycle that do not have relevance to the domain. These states should not be modeled.

BABOK empfiehlt die Verwendung von State diagrams in den folgenden Tasks:

6.2 Organize Requirements

Events. A request to a business system or organization to do something, such as a customer placing an order, or a manager requesting a report, can be described as an event. The organization must respond to an event, and in most cases an event will trigger or affect a business process. Events can come from outside the business area, from within it, or occur at scheduled times. Events can serve as the basis for a *scope model (9.27)* and may be described in other models, including *process models (9.21)*, *state diagrams (9.29)*, and *use cases (9.26)*.

Processes. Processes are a sequence of repeatable activities executed within an organization. Processes can be simple (involving one person and a system) or complex (involving many people, departments, organizations and systems). Processes describe who and what has to be involved in fully responding to an event, or how people in the enterprise collaborate to achieve a goal. Processes are normally described in *process models (9.21)*, although useful information may also be captured in *organization models (9.19)*, *state diagrams (9.29)*, or *use cases (9.26)*.

Rules. Rules are used by the enterprise to enforce goals and guide decision-making. They determine when information associated with an entity may change, what values of information are valid, how decisions are made in a process, and what the organization's priorities are. Business rules are normally described as such, although they may also be embedded in *process models (9.21)*, *state diagrams (9.29)*, and *use cases (9.26)*.

6.3 Specify and Model Requirements