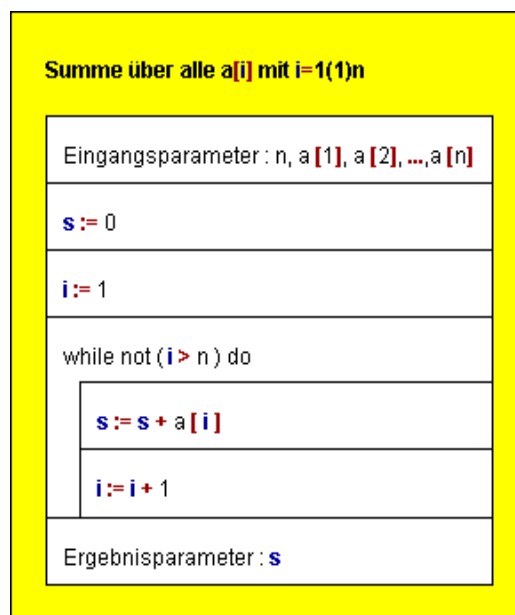


Beispiel $s = \sum_{i=1}^n a_i$ als Struktogramm:

Zu beachten ist, daß der Index **i** in diesem Beispiel mit der Zahl **1** beginnt (wie in der Mathematik). Das beeinflußt wesentlich die Abbruchbedingung der Schleife, die hier als **while not (i > n) do** bzw. äquivalent als **while (i <= n) do** angegeben werden kann.

Die von der Programmiersprache **C** abgeleiteten Sprachen, wie **C++**, **C#** oder **JAVA**, beginnen für Felder/Vektoren jedoch mit dem Index **0**, da der Index hier der Adreßberechnung des Elementes mit Index **i** dient. Der Name eines Vektors ist in **C** immer die Adresse des Elementes mit Index **0**, die Adresse des Elementes mit Index **i** wird in **C** aus der Summe von Vektorname + Wert **i** gebildet, intern wird die Adresse über **Vektorname + i * sizeof(Basistyp des Vektors)** berechnet, wobei **sizeof(Basistyp des Vektors)** die Anzahl der Bytes ist, die ein Vektorelement belegt. Sollte der Index mit 0 beginnen, müßte die Abbruchbedingung der Schleife als **while not (i >= n) do** bzw. äquivalent als **while (i < n) do** angegeben werden.



Wohlstrukturiertes Programmieren:

Insbesondere im Rahmen der strukturierten Programmierung werden nur folgende Elemente für die Algorithmierung verwendet:

- **Aktion** (einfache Anweisung oder Unterprogramm-Aufruf)
- **Sequenz**
- **While-Schleife**
- **Repeat-Schleife**
- **For - Schleife** (Laufanweisung)
- **If - Anweisung** (Alternative)
- **Case - Anweisung** (Verteiler)

Diese eingeschränkte Form des Programmierens wird auch als **goto-freies** Programmieren bezeichnet (im Programm **Structorizer** existiert jedoch eine Sprunganweisung, insbesondere ist der Sprung aus einer Schleife (break) sinnvoll).

2.8 UML Aktivitätsdiagramme

Aktivitätsdiagramme sind Bestandteil der Unified Modeling Language (UML) und das Notationssymbol der Wahl, wenn es darum geht, Abläufe zu modellieren.

Aktivitätsdiagramme sind von Datenflußdiagrammen, Petri-Netzen, Struktogrammen, Programmablaufplänen und der Graphentheorie abgeleitet.

Aktivitätsdiagramme können für die Beschreibung von Algorithmen verwendet werden, insbesondere, wenn es um die Abbildung zeitparalleler Vorgänge (auch nebenläufige bzw. asynchrone Vorgänge genannt) geht, die auch Zeit verbrauchen können. Hauptanwendung ist jedoch die Spezifikation des Verhaltens von Systemen, die algorithmiert und auf Computern programmiert werden sollen.

Aktivitätsdiagramme beschreiben Algorithmen auf einer relativ allgemeinen Ebene und erlangen wegen der zunehmenden Komplexität des auf Computern abzubildenden Verhaltens einen zunehmend größeren Einfluß in der Algorithmierung.

Ein *Aktivitätsdiagramm* stellt *Aktivitäten* mit einem nichttrivialen Charakter dar. Eine *Aktivität* im Sinne der UML spezifiziert dabei die Menge von potenziellen *Abläufen*, die sich in der Realität unter bestimmten Randbedingungen abspielen (siehe folgende Abbildung "Eine Aktivität und ihre Bestandteile").

Aktivitätsdiagramme zeigen den Rahmen und die Regeln von Verhaltensabläufen auf detailiertem Niveau an.

Aktivitätsdiagramme sind gerichtete Graphen mit einer oder mehreren *Aktivitäten*, *Aktionen*, *Objektknoten*, *Kontrollelementen* zur Ablaufsteuerung und verbindende *Kanten*.

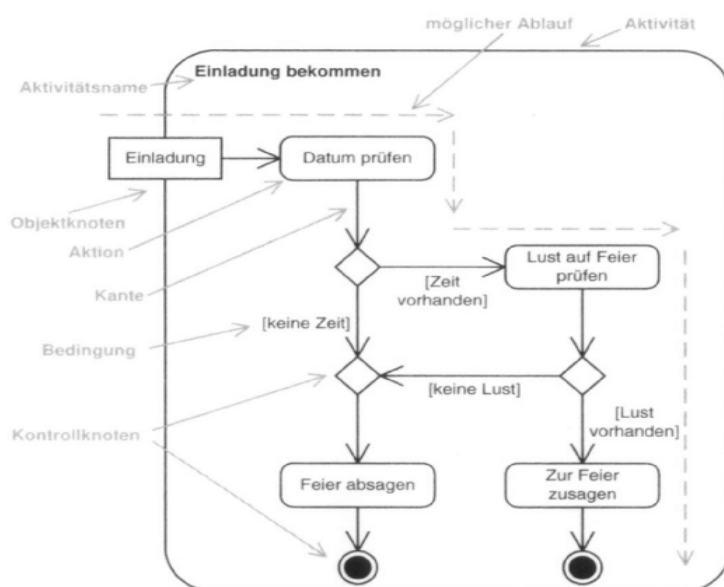


Abbildung Eine Aktivität und ihre Bestandteile

Die Abbildung zeigt ein Aktivitätsdiagramm, bei dem ein möglicher Ablauf gekennzeichnet wurde. Die Gesamtheit aller Abläufe wird – wie bereits erwähnt – als *Aktivität* bezeichnet. Ein *Aktivitätsdiagramm* zeigt einen Ausschnitt davon: genau eine oder auch mehrere *Aktivitäten*.

Eine *Aktion* (z.B. Feier absagen) ist ein Einzelschritt, den ein Ablauf unter Zeitaufwand durchläuft und in dem etwas "getan wird".

Objektknoten (im Beispiel **Einladung**) repräsentieren dabei beteiligte Daten bzw. Gegenstände der realen oder vorstellbaren Welt.

Zwischen den *Aktionen* oder *Objektknoten* befinden sich *Kontrollknoten* zur Flußsteuerung des Ablaufs.

Kontrollknoten geben die Entscheidungsregeln vor, wann und in welcher Reihenfolge die einzelnen *Aktionen* durchgeführt bzw. *Objektknoten* verändert werden.

Durch *Kontrollknoten* lassen sich Abläufe

- parallelisieren und synchronisieren
- verzweigen und zusammenführen
- unter bestimmten Bedingungen lenken
- mehrfach instanziiieren
- asynchron unter- bzw. abbrechen
- parametrisieren und mit Objektknoten verknüpfen

Kanten sind Pfeile, die einzelne Elemente verbinden und damit die zeitlich logische Reihenfolge des Ablaufs herstellen.

Tokenkonzept

Ein *Token* (auch Marke genannt) ist eine *Marke* (auch als komplexes Datenobjekt mit mehreren Einzeldaten vorstellbar), die den Punkt anzeigt, an dem sich ein Ablauf aktuell befindet.

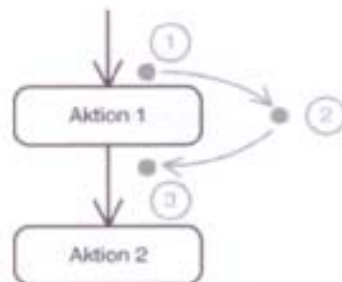
Die Wanderung eines *Tokens* durch eine *Aktivität* repräsentiert die Abarbeitung eines Ablaufs.

In einer *Aktivität* können gleichzeitig beliebig viele *Tokens* unterwegs sein, um dadurch parallele Abläufe oder mehrfach instanziierte Abläufe abzubilden.

Alle wesentlichen Ablaufkonzepte lassen sich mit Token erklären und werden auf den folgenden Seiten vorgestellt.

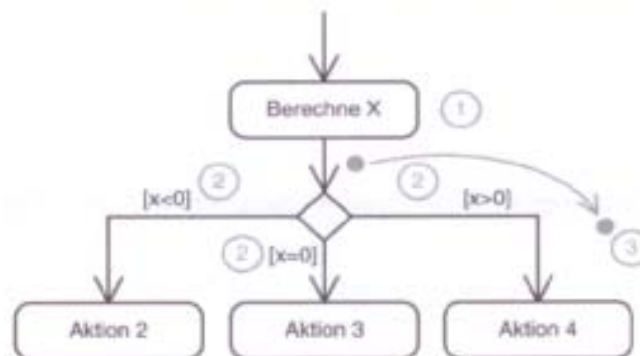
Der Verständlichkeit wegen sind die *Token* durch Punkte symbolisiert. Das ist nicht konform zur UML, da *Token* in Aktivitätsdiagrammen nicht repräsentiert werden, sondern nur der logischen Erklärung dienen.

Abarbeitung von Aktionen

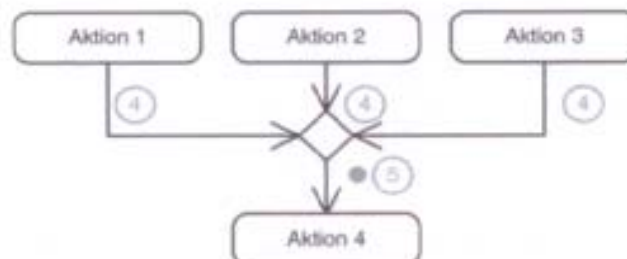


Token lösen eine einzelne Aktion aus. Vereinfacht gesehen startet eine Aktion dann, wenn ein Token auf der eingehenden Kante angeboten wird (1). Dieses Token wird anschließend von der Kante entfernt und für die Dauer des Aktionsablaufes innerhalb der Aktion aufbewahrt (2). Ist der Ablauf abgeschlossen, wird der Token über die wegführende Kante der nachfolgenden Aktion angeboten (3).

Verzweigung und Vereinigung

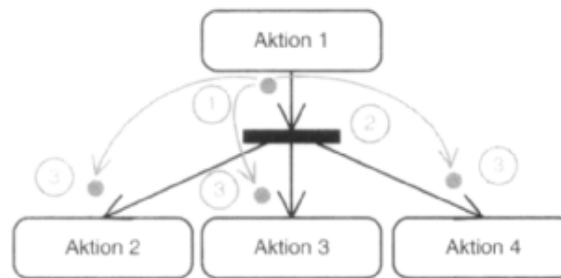


Da die wenigsten Abläufe geradlinig vonstatten gehen, enthalten Aktivitätsdiagramme Notationselemente für Alternativabläufe (Ja/Nein-Entscheidungen, Mehrfachauswahl). Diese Entscheidungspunkte sind als eine Art „Wegkreuzung“ für die Token zu betrachten. Abhängig von dem Ergebnis einer Aktion (1) liegt der Token nur bei *genau einer Folgeaktion* an, für welche die *Bedingung zutrifft* (2). (Im Beispiel: Wenn die Berechnung von x 27 ergeben hätte, würde der Token *nur* bei Aktion 4 (3) anliegen.)



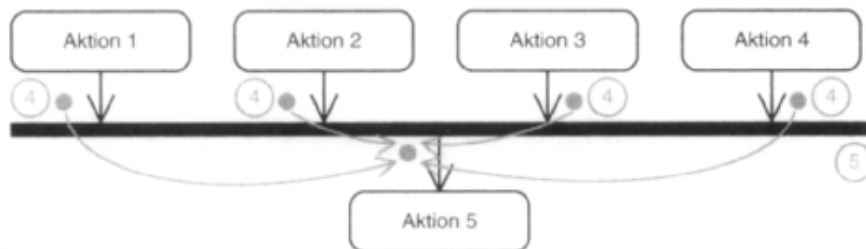
Analog dazu lassen sich verschiedene Alternativzweige (4) zu einem Zweig zusammenführen. Hier liegt der Token – egal von welcher Aktion es ausgeht – immer bei der Folgeaktion (5) an.

Parallelabarbeitung und Synchronisation



Neben den Alternativabläufen ist auch die Aufteilung eines Ablaufs in nebenläufige Zweige möglich. Die Aktionen werden dort parallel abgearbeitet. Das Ausgangstoken (1) wird dabei am „Splittingknoten“ (2) vervielfacht, sodass es an *jeder* Folgeaktion (3) anliegt. In welcher zeitlichen Reihenfolge danach jedes Token abgearbeitet wird, hängt von der jeweiligen Anwendung ab. Nach einer Aufteilung können die einzelnen „Teilabläufe“ also unabhängig voneinander abgearbeitet werden.

Beim Zusammenführen paralleler Abläufe zu einem Ablauf, sprich bei der UND-Synchronisation³, werden die Token so lange an dem Synchronisationsknoten aufgesammelt, bis *alle* Token der jeweiligen UND-Zweige (4) angekommen sind. Danach werden sie wieder zu einem Token „verschmolzen“ (5), das bei der Folgeaktion anliegt.



Objektflüsse

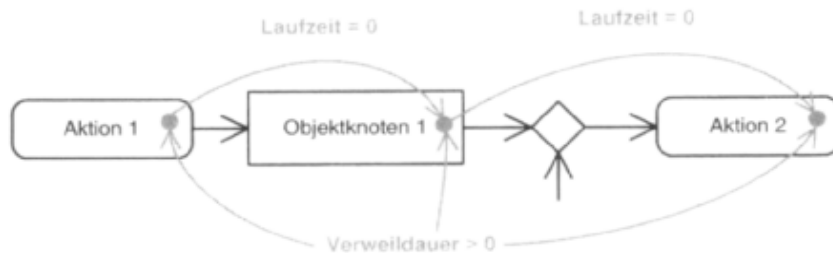


Neben den Kontroll-Token, deren Hauptaufgabe das Anregen von Aktionsabläufen ist, gibt es Daten-Token. Diese dienen als Transportvehikel, auf denen Daten oder Werte durch das Diagramm getragen werden. Solche Token bewegen sich über Kanten, an denen mindestens ein Objektknoten beteiligt ist.

In einen Objektknoten hineingehende Token repräsentieren Daten oder Werte, die in dem Objektknoten gesetzt bzw. gesammelt oder durch den Objektknoten repräsentiert werden. Aus einem Objektknoten herausgehende Token transportieren das Objekt selbst bzw. Daten des Objekts in die Folgeaktion (2). Der Objektknoten ist daher ein *Notationsmittel*, um den Fluss von Objekten und Werten zu modellieren, der Knoten *an sich* stellt *kein* Objekt/Datenwert dar

³ Die UND-Synchronisation stellt „nur“ das Standardverhalten an einem Synchronisationsknoten dar. Sie dürfen zudem weitere beliebige logische Ausdrücke (Oder, Negation, ...) spezifizieren

Verweildauer von Token



Token verweilen nur in Aktionen oder Objektknoten. Übergänge über eine Kante und Abläufe durch alle Kontrollknoten (wie Entscheidungs- oder Parallelisierungsknoten) sind zeitlos. Um diese Bedingung zu gewährleisten, kann eine Aktion nach Beendigung ihres Ablaufs ein Token nicht einfach „vor die Tür setzen“. Kann ein solches Token von der nächsten Aktion oder vom nächsten Objektknoten nicht aufgenommen werden, weil zum Beispiel Bedingungen nicht erfüllt sind, dann würde dieses Token auf der Kante festsitzen. Da dies nicht erlaubt ist, bietet die abgeschlossene Aktion den Folgekanten der Token nur dann an, wenn ein vollständiger Übergang (das heißt auch über Kontrollknoten) möglich ist. Der Token „springt“ dann sofort zum Ziel (zu einer Aktion oder einem Objektknoten). Insofern ist das weiter oben gezeichnete Bild, in dem ein UND-Synchronisationsknoten die Token aufammelt, zu korrigieren.

Bedingte und gewichtete Abarbeitung von Abläufen

Auch mit Hilfe von Bedingungen, die bereits bei den Entscheidungsknoten (siehe oben) verwendet wurden, lassen sich Abläufe steuern. Durch Token lassen sich hier zusätzlich Zähler bzw. Gewichte angeben. Eine Kante kann festlegen, dass eine bestimmte Anzahl von Token vorhanden sein muss, damit ein Übergang möglich ist. Bei diesem Vorgang werden dann gleichzeitig mehrere Token transferiert. Alle bisher beschriebenen Eigenschaften von Token gelten auch für diesen Fall.

Welche Bedingungen es im Einzelnen gibt und wie diese notiert werden, lesen Sie unter den jeweiligen Elementen nach.

Start und Ende von Abläufen

Die Lebensdauer eines Ablaufs entspricht dem eines Tokens. Durch die Erzeugung eines Tokens wird ein Ablauf initiiert, durch die Eigen- oder Fremdzerstörung der Ablauf beendet. Im übertragenen Sinne wird also auch beim Duplizieren eines Tokens ein neuer Ablauf begonnen, beim Verschmelzen zweier Token ein Ablauf zerstört.

Inwiefern dies die Realität widerspiegelt, ist Sache der Implementierung. Token sind kein Konzept, das sich eins zu eins in der Realität in einem Rechner oder in einem Geschäftsprozess wieder findet – sie sind ein theoretisches Anschauungsmodell, um verschiedene Verhaltensmodelle in Abläufen zu erklären.

Anwendungsbeispiel

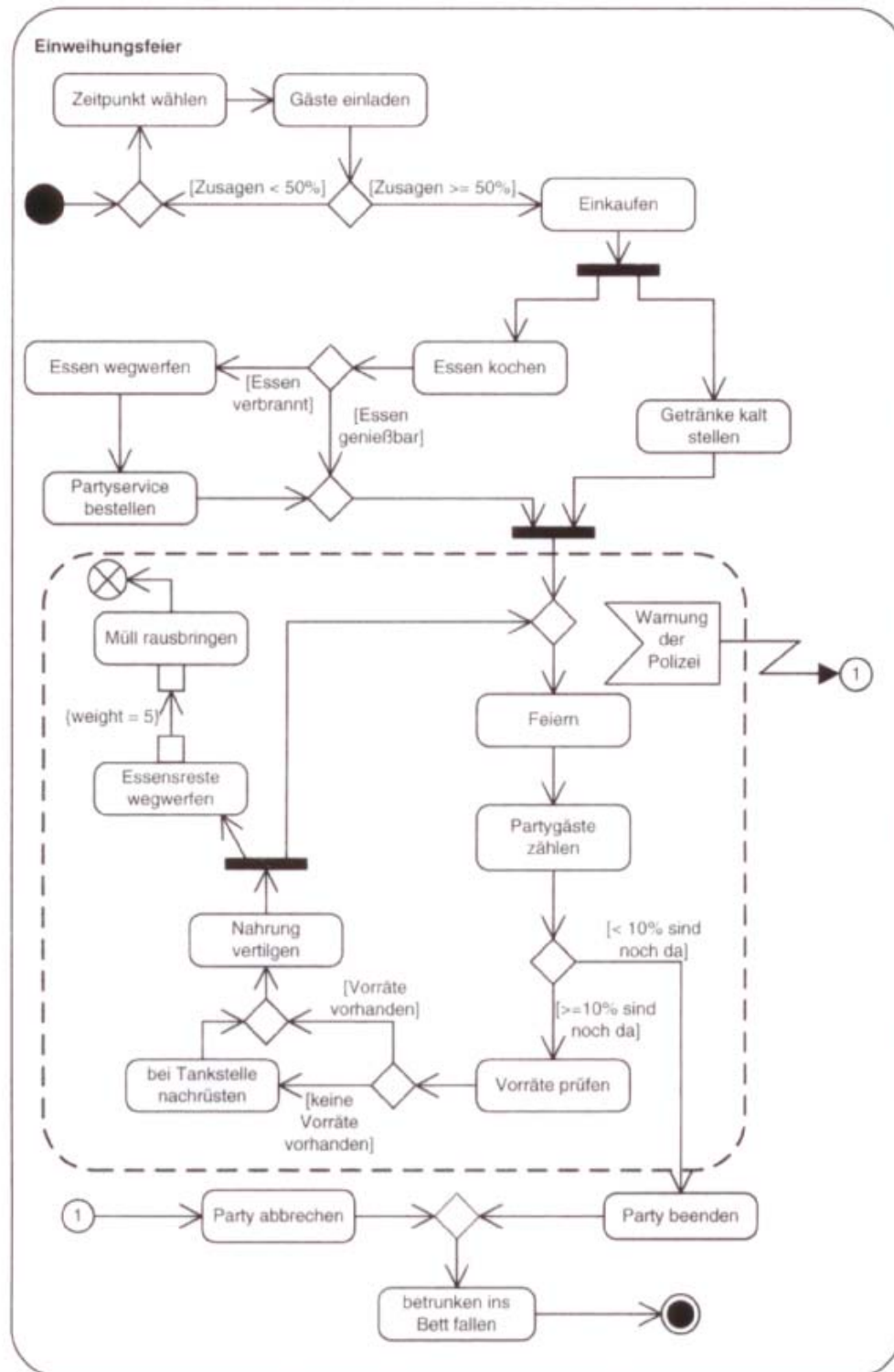


Abbildung Der Verlauf einer typischen Party

Interpretation des Anwendungsbeispiels

Nach dem Start muß der Zeitpunkt der Party gewählt werden. Sagen weniger als 50% der geladenen Gäste zu, muß ein neuer Zeitpunkt gewählt werden. Ansonsten können die Vorräte eingekauft werden.

Parallel zueinander werden die Getränke kalt gestellt und das Essen zubereitet.

Sind die Getränke kalt gestellt und das Essen zubereitet, so kann die Feier beginnen.

Je nach Anzahl der noch anwesenden Gäste wird entschieden, ob die Party weitergeht oder beendet wird.

Wird die Feier fortgesetzt, muß überprüft werden, ob noch genügend Vorräte vorhanden sind. Wenn nicht, muß eine Tankstelle aufgesucht werden. Sind Vorräte vorhanden, kann Nahrung verteilt werden.

Hier setzt wieder ein paralleler Ablauf ein. Es wird weiter gefeiert und parallel werden Essensreste entsorgt. Haben sich genügend Essensreste angesammelt {weight = 5}, wird der Müll raus gebracht und dieser Pfad beendet.

In dem Fall, daß die Feier zu ausgiebig wird und die Polizei einschreiten muß, werden alle Abläufe innerhalb der gestrichelten Linie abgebrochen und der Ablauf setzt sich bei der Aktion Party abbrechen fort.

Am Ende bleibt nur noch der Weg ins Bett. Danach ist die Aktivität beendet.

Anwendung von Aktivitätsdiagrammen

Die wichtigsten Anwendungsgebiete für *Aktivitätsdiagramme* sind:

- ***Geschäftsprozeßmodellierung***
- ***Beschreibung von Use-Cases (Anwendungsfällen)***
- ***Implementierung einer Operation***

Aktivitätsdiagramme lohnen sich umso mehr, je komplexer die darzustellenden Sachverhalte sind.

Geschäftsprozeßmodellierung

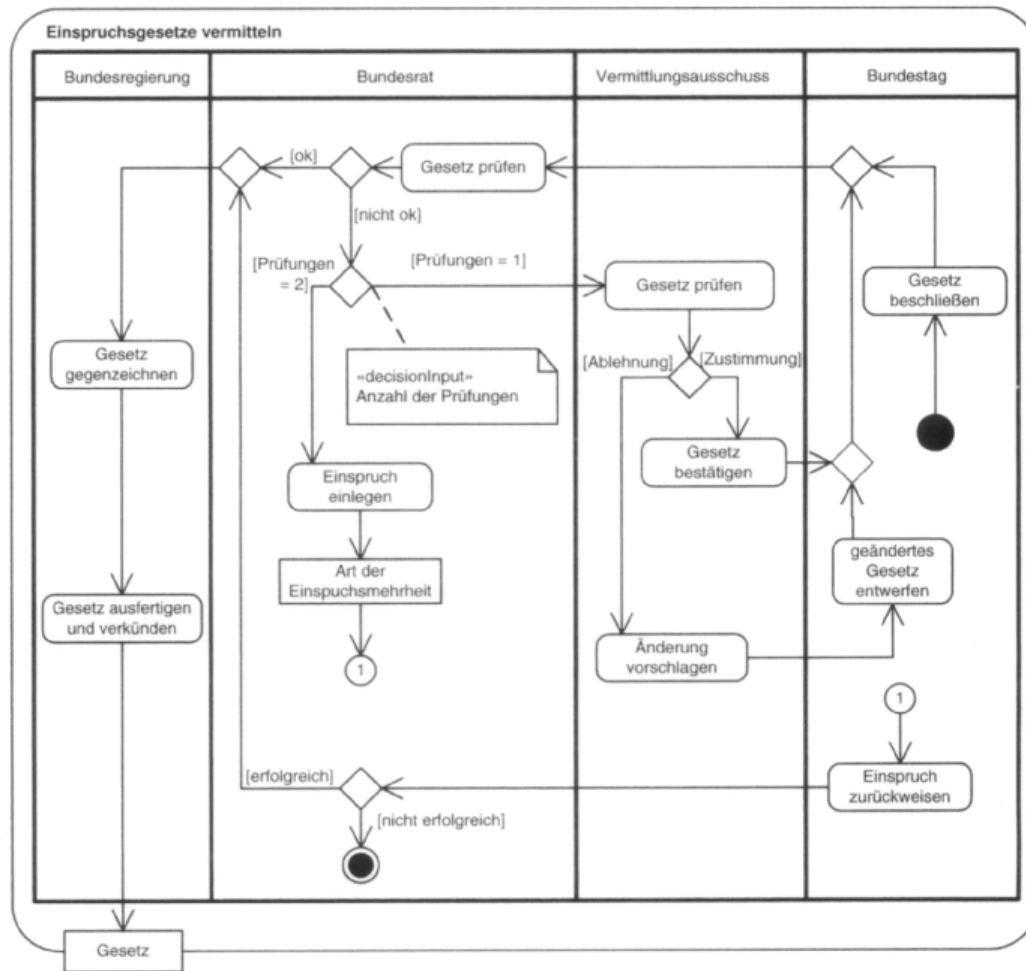


Abbildung Der Ablauf vom Gesetzesvorschlag zum fertigen Gesetz

Das Beispiel zeigt den Weg eines Gesetzesbeschlusses bis hin zum rechtskräftigen Gesetz. Das Diagramm ist in 4 *Aktivitätsbereiche* unterteilt. Damit wird verdeutlicht, welche Instanz für die jeweiligen Aktionen verantwortlich ist.

2.9 UML Use-Case-Diagramme (Anwendungsfall-Diagramme)

Die Frage "Was soll mein geplantes System eigentlich leisten ?" sollte am Beginn jeder Systementwicklung stehen.

Ein *Use-Case-Diagramm* zeigt das **externe Verhalten** eines Systems **aus Sicht der Nutzer**, indem es die *Nutzer* ("Akteure"), die *Use-Cases* und deren *Beziehungen* zueinander darstellt.

Ein *Nutzer* kann eine *Person* oder ein *anderes System* sein.

Use-Cases bilden die Reaktion des Systems auf Ereignisse seiner Umwelt ab und fassen dabei Teile der Systemdienstleistungen zusammen.

In der Praxis werden häufig *Prosabeschreibungen* verwendet. Nebenläufigkeiten und Kontrollanweisungen (Schleifen, Entscheidungen, ...) machen die Erstellung eines Prosatextes mühselig und zeitaufwändig (NR).

Anwender verzweifeln an der Lesbarkeit derartiger Prosatexte. Nachträgliche Änderungen oder Erweiterungen sind kaum möglich (Vergleich mit Gesetzen).

Die formalisierte Darstellung verbessert die Les- und Wartbarkeit. Hinter dem Use-Case steht ein spezielles Ablaufverhalten und kann durch Aktivitäten beschrieben werden.

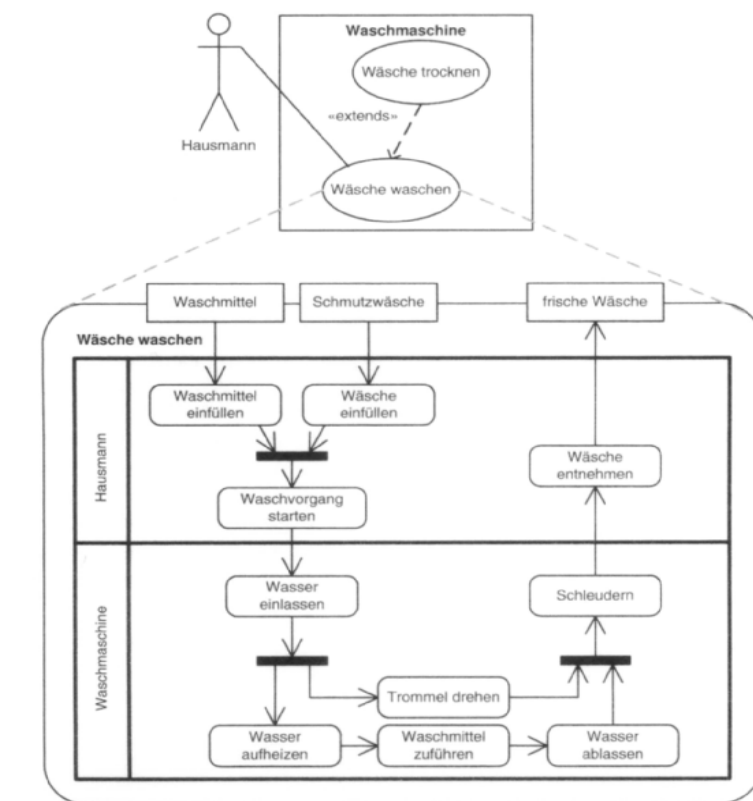


Abbildung Der Use-Case wäsche waschen, dargestellt in einem detaillierten Aktivitätsdiagramm. Einblicke in das Leben eines SOPHISTischen Hausmanns.

Das Beispiel zeigt die Beschreibung des Use-Case "Wäsche waschen" mittels eines Aktivitätsdiagrammes.

Das Aktivitätsdiagramm beschreibt nur diesen Use-Case, nicht jedoch die spätere Modellierung.

Das Diagramm zeigt, welche Aktionen der Akteur (Hausmann) und welche das System (Waschmaschine) ausführt.


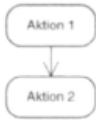
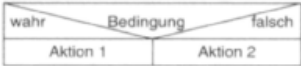
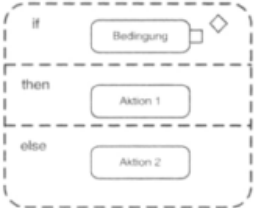
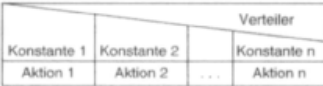
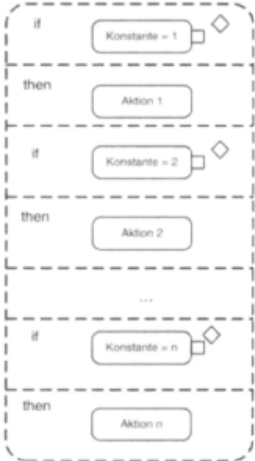
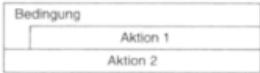


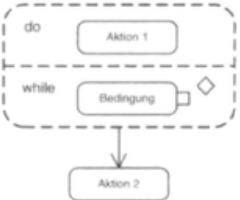
Das für den Akteur sichtbare Ergebnis des Use-Case, nämlich die "frische Wäsche", wird über einen Ausgabeparameter dargestellt.

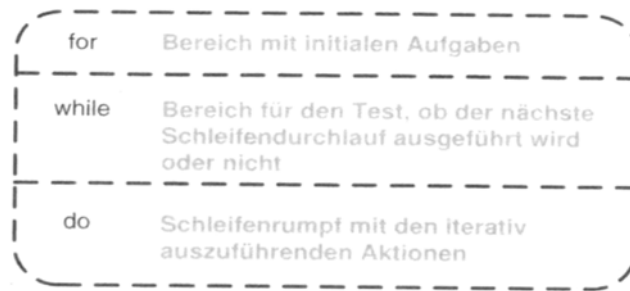
Implementierung einer Operation

Konkrete Realisierungen von Algorithmen können nicht nur in natürlicher Sprache, als Pseudocode oder in einer konkreten Programmiersprache beschrieben werden.

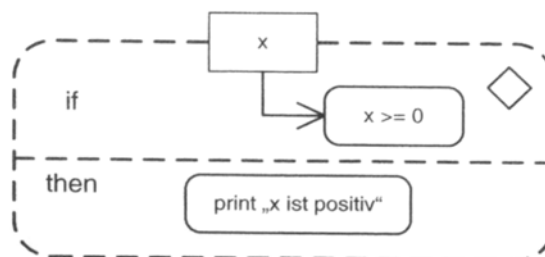
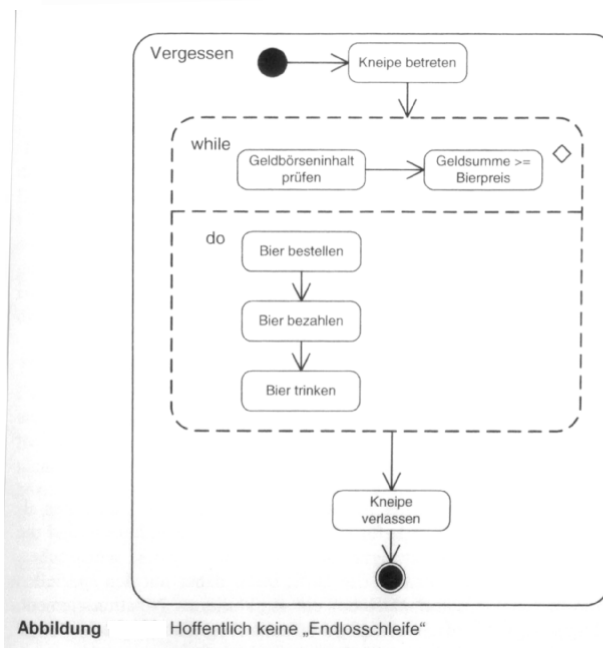
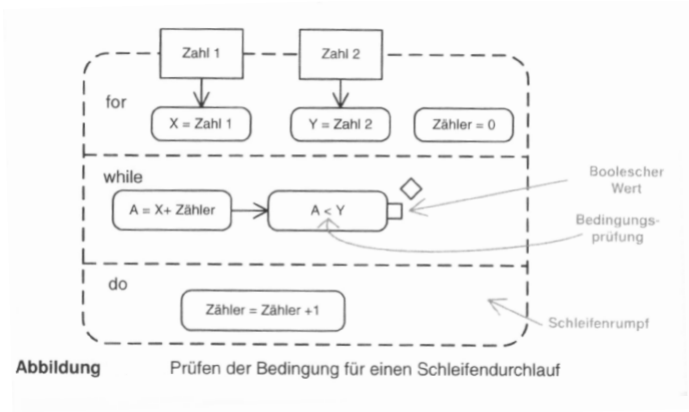
Algorithmen können auch mit Aktivitätsdiagrammen der UML modelliert werden. Das ist insbesondere dann von Bedeutung, wenn ein Quellcode-Generator genutzt werden soll, der UML-Aktivitätsdiagramme zum Ausgangspunkt hat.

Die Strukturelemente der Struktogramme finden sich in leicht abgewandelter Notation in den Aktivitätsdiagrammen wieder, wie die folgende Gegenüberstellung zeigt:

Gruppe	Natürlich-sprachlich	Struktogramm	UML-Aktivitätsdiagramm
Sequenz	<Aktion 1> <Aktion 2>		
Auswahl	WENN <Bedingung> DANN <Aktion 1> SONST <Aktion 2>		
Mehrfach-auswahl	FALLS <Verteiler> <Konstante 1> : <Aktion 1> <Konstante 2> : <Aktion 2> ... <Konstante n> : <Aktion n>		
Wiederholung	SOLANGE <Bedingung> <Aktion 1>		
	WIEDERHOLE <Aktion 1> SOLANGE <Bedingung> <Aktion 2>		



Die drei Bereiche eines Schleifenknotens



Unterbrechungen

Ein *Unterbrechungsbereich* umschließt eine oder mehrere Aktionen. Der Bereich wird durch ein Rechteck mit runden Ecken modelliert.

Ein blitzförmiger Pfeil, der innerhalb des Bereiches beginnt und dessen Ziel außerhalb des Bereiches liegt, beschreibt die Unterbrechungskante.

Im Beispiel unten beginnt die Unterbrechungskante bei "Abbruchknopf gedrückt".

Wird der Bereich über die Unterbrechungskante verlassen, so werden sämtliche in dem Bereich ausgeführten Aktionen abgebrochen, indem alle vorhandenen Tokens verworfen werden. Der Ablauf in der Aktivität setzt sich am Zielknoten der Unterbrechungskante fort.

Tritt die Unterbrechung ein, wenn ein Token über eine Kante den unterbrechungsbereich verläßt, so bleibt dieses Token erhalten.

Ein Unterbrechungsbereich läßt sich sehr gut für Ausnahmen einsetzen, die eine sofortige Beendigung mehrerer Aktionen bzw. Abläufe in einer Aktivität bewirken sollen.

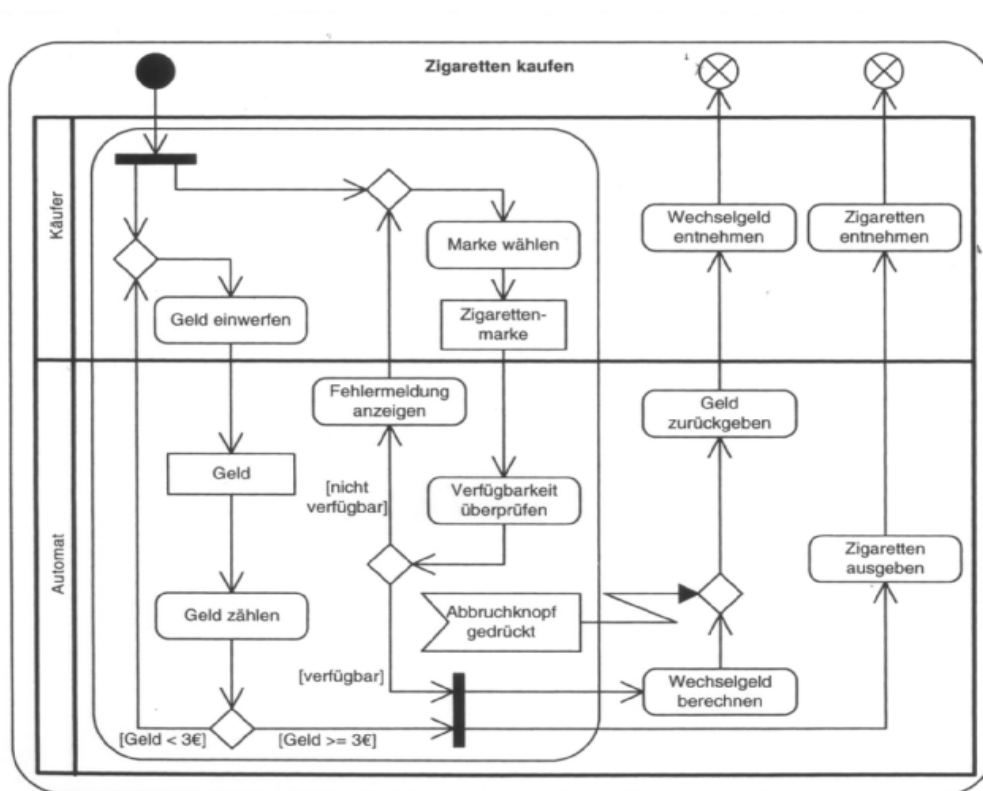


Abbildung Bedienung eines Zigarettenautomaten

Während der Bedienung eines Zigarettenautomaten kann jederzeit die Taste zur Geldrückgabe gedrückt werden. Diesem Umstand wird durch die Verwendung des Unterbrechungsbereiches Rechnung getragen. Nach einem Abbruch wird das zurückzugebende Geld bestimmt ausgeworfen, ansonsten gibt der Automat zusätzlich Zigaretten der gewählten Marke aus.

Verzweigungs- und Verbindungsknoten

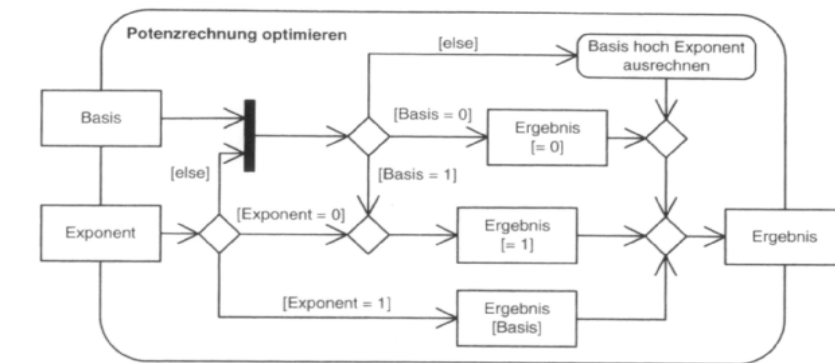


Abbildung Aktivitätsdiagramm mit optimiertem Berechnungsalgorithmus für die Potenzrechnung

Das Beispiel zeigt eine Möglichkeit, die Potenzrechnung bei trivialen Operanden zu vereinfachen. Wenn der Exponent untersucht wird, stoppt der obere Verlauf, der mit der Basis beginnt und wartet auf das von unten kommende Token. Im Falle von Exponent = 0 und Exponent = 1 wird das Ergebnis sofort ausgegeben. Die Berechnung außerhalb der Trivialfälle ist als Aktion dargestellt.

Parallelisierungsknoten

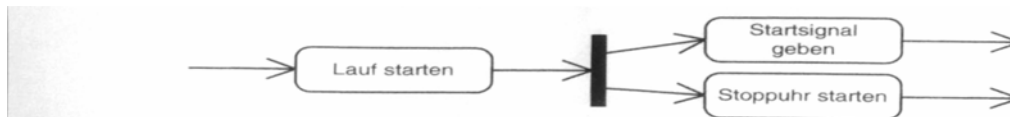


Abbildung Ist die Entscheidung gefallen, das Rennen zu starten, müssen zeitgleich die Stoppuhr gestartet und ein Startsignal gegeben werden

An einem Parallelisierungsknoten (engl. ForkNode) wird der eingehende Ablauf in mehrere parallele Abläufe aufgeteilt. Jedes eingegangene Token wird dabei dupliziert und an jeder ausgehenden Kante angeboten. Sowohl die eingehenden als auch die ausgehenden Kanten können mit Bedingungen versehen sein.

Synchronisationsknoten

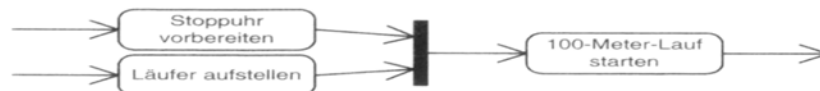


Abbildung Der Lauf kann erst starten, wenn die Stoppuhr bereit ist und die Läufer aufgestellt sind

Die Abläufe werden am Synchronisationsknoten durch ein implizites UND verbunden. Soll eine andere Verhaltensweise v (OR, XOR, ...) modelliert werden, so ist dies durch eine Synchronisations-Spezifikation $\{joinspec = v\}$ anzugeben.

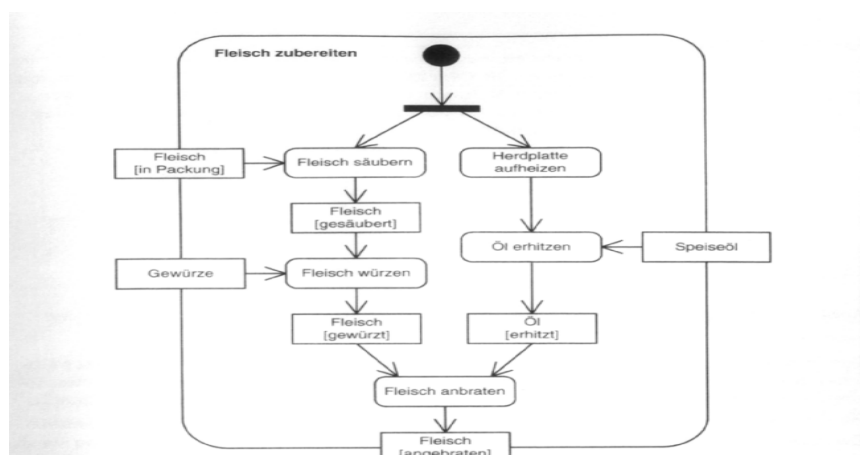


Abbildung Warum viele Menschen mit dem Kochen Probleme haben ...

Bei allen Aktionen, die mehrere eingehende Kanten haben, muss nach den Gesetzen des Token-Konzepts auf jeder Kante je ein Token anliegen, und somit werden die Abläufe wieder synchronisiert, bis es bei der Aktion Fleisch anbraten nur noch eine Aktion gibt, die gleichzeitig ausgeführt wird.