1 Allgemeines

Neben Sequenzdiagramm, Kollaborationsdiagramm, Zustandsdiagramm und Anwendungsfalldiagramm ist das Aktivitätsdiagramm eines von fünf Diagrammen in UML, welches dynamische Zusammenhänge von Systemen modelliert. Aktivitätsdiagramme legen das Hauptaugenmerk auf:

- was ein einzelner Schritt tun soll (also die Aktionen der Aktivität)
- in welcher Reihenfolge die Aktionen durchgeführt werden sollen
- wer für welche Aktionen verantwortlich ist (dies ist optional und wird meist über sogenannte „Swimmlanes“, also abgetrennte Streifen dargestellt). Diese Verantwortungsbereiche werden auch „Partitions“ genannt.

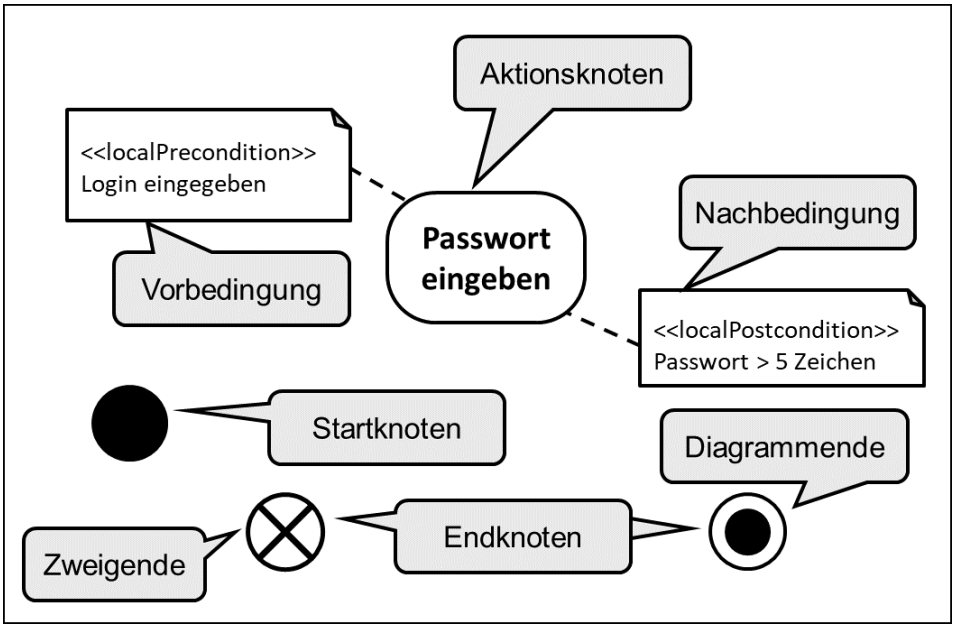
Die Idee hinter dem Aktivitätsdiagramm ist es, möglichst flexibel beliebige Abhängigkeiten darzustellen. Dies erklärt die Vielzahl von verschiedenen Notationselementen. Es können somit recht komplexe aber dennoch übersichtliche Zusammenhänge dokumentiert werden.

2 UML-Aktivitätsdiagramm, Notation

Die wichtigsten Elemente des Aktivitätsdiagramms.

2.1 Aktivitäts-, Start- und Endknoten

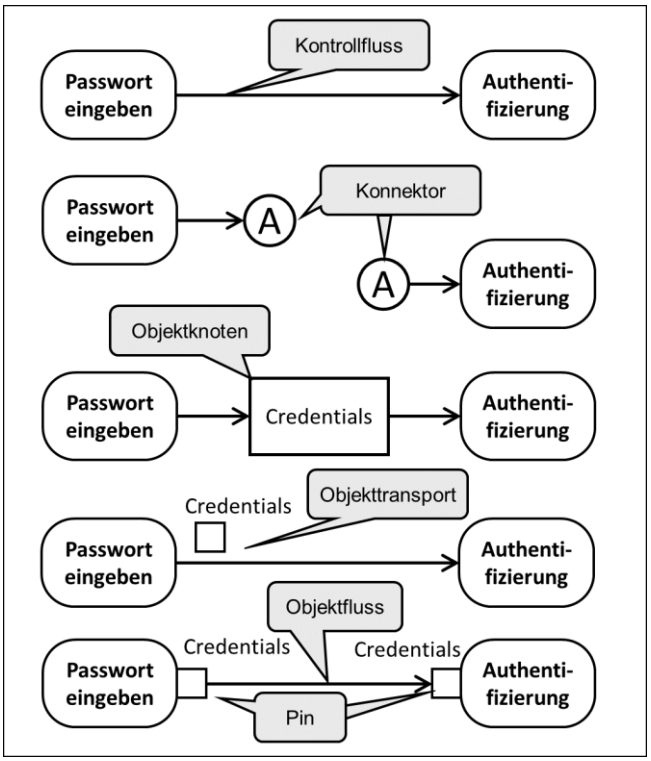
Eine **Aktion** ist „ausführbar“, weshalb sich die Namensgebung auch am „tun“ orientieren sollte. Im **Rahmen des Diagramms** ist die Aktion **atomar**, was aber nicht bedeutet, dass sie in der späteren Umsetzung nicht in kleinere Teilaspekte zerlegt werden kann. Aktionen sind **keine Aktivitäten**, sondern Einzelschritte von Aktivitäten. Aktionen zeichnen sich dadurch aus, dass sie **einen Eingang und einen Ausgang** besitzen (Anm.: dies wird weiter unten bei den Kontrollflüssen klarer – dort wird auch der Sonderfall „Parametersatz“ besprochen). Weiterhin kann bei Aktionen eine **optionale Vor**- und eine **Nachbedingung** ergänzt werden, um den späteren Fluss der Aktionen besser verstehen zu können.



Start- und Endknoten führen **keine Aktion** aus, sie Kontrollieren lediglich den Ein- und Ausstieg aus der Aktivität. Es können pro Aktivitätsdiagramm **mehrere Start- und Endknoten** existieren. Endknoten gibt es in zwei Ausführungen. Der rechts gezeigte Endknoten beendet das gesamte Aktivitätsdiagramm – der **Gesamtprozess** wird somit **abgebrochen**, auch wenn noch parallele Aktivitäten durchgeführt werden. Der links gezeigte **bricht** lediglich den aktuellen **Parallelzweig ab**.

2.2 Kontrollfluss

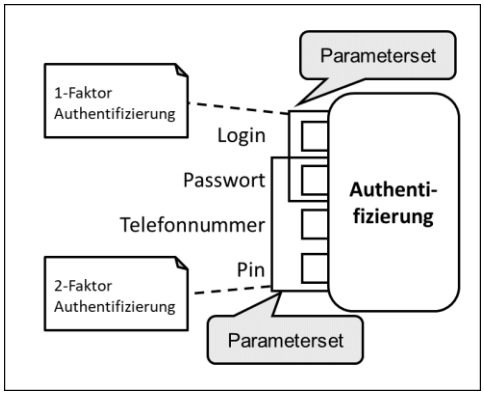
Kontrollflüsse sind **gerichtete Pfeile**, welche die **Reihenfolge** der Aktionen festlegen. Mit diesen Pfeilen werden keinerlei Aktionen verknüpft. Für sehr komplexe Diagramme können die Kontrollflüsse zur **Vermeidung von Kreuzungen** mit Hilfe von **Konnektoren** aufzutrennen. Hierbei müssen die Benennungen der Konnektoren eindeutig sein (also A zu A, B zu B etc.).



Die Kontrollflüsse können noch mit Objektinformationen angereichert werden. Diese Objekte stehen für **Transportbehälter von Informationen**, welche vom Fuß zur Spitze wandern. Hierbei können pro Objekt auch mehrere Informationen verpackt werden. Wir kennen hier drei verschiedene Notationen:

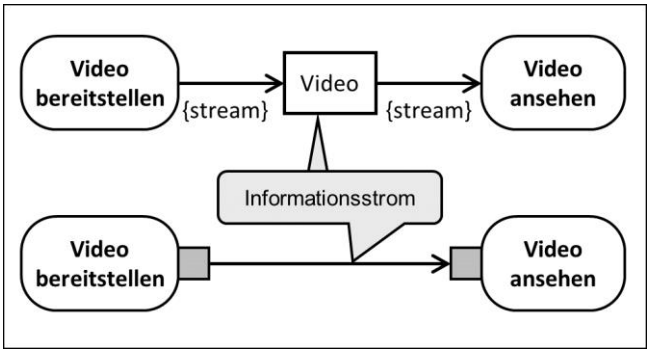
- **Objektknoten**: Werden in den Kontrolluss eingefügt.
- **Objekttransport**: Gleichbedeutend mit dem Objektknoten.
- **Objektfluss**: Hier werden **Ausgangs**- und **Eingangspins** an die Aktionen geheftet. Diese symbolisieren Eingabe- und Ausgabeparameter.

Die Objekte können noch mit einem **Zustand** in eckigen Klammern versehen werden (bspw. *[vollständig]*). Dies kann in allen drei Notationen ergänzt werden.



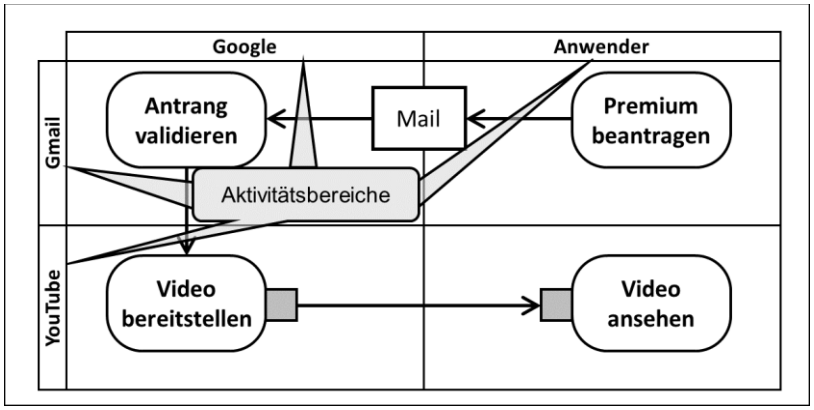
Eingabepins können auch zu einem **Parametersatz** zusammengeführt werden, wodurch eine Aktion mehrere Eingänge hat. Die zusammengehörigen Eingänge werden mit einem Rechteck markiert und mit einem Label benannt. **Mehrere Parametersätze** werden **XOR** verknüpft, es wird also immer nur ein Parametersatz ausgewertet.
Die **Pins innerhalb** eines **Parametersatzes sind UND** verknüpft, das heißt alle Parameter müssen vorliegen, damit die Aktion ausgeführt werden kann.

Eine Sonderform der Objektnoten sind die **Streams**, welche den Sonderfall berücksichtigen, dass beim Informationstransport der Empfang während des Sendens erfolgt (also eine **Zeitüberschneidung** stattfindet). Hierbei werden entweder die Kontrollflüsse vor und nach dem Informationsstrom-Objektes mit *{stream}* markiert, oder die Pins werden grau dargestellt.



2.3 Aktivitätsbereich

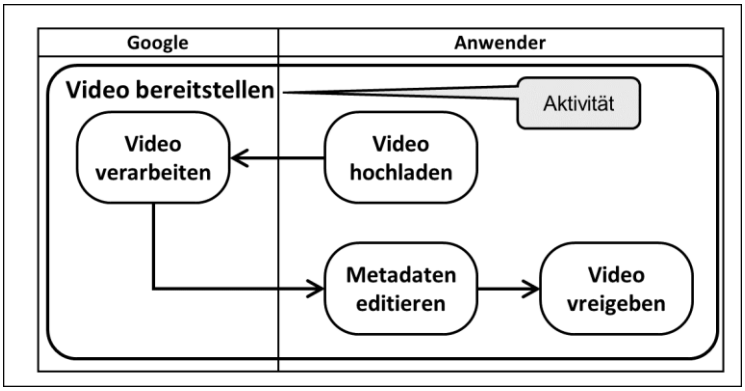
Aktivitätsbereiche werden auch **Verantwortungsbereiche** genannt. Die Aktionen werden in die Spalte / Zeile desjenigen gesetzt, der die Verantwortung dafür hat – sprich, sie ausführt. Diese Bereiche kann man waagerecht und / oder auch senkrecht anordnen, wobei letzteres am häufigsten zu sehen ist. Diese Bereiche werden auch „**Swimmlanes**“ genannt.



Wie im oberen Bild dargestellt, können es auch „Orte“ sein, wo die Aktionen ausgeführt werden. Weiterhin kann man mit dem Stereotpyen <<***external***>> markieren, dass der Aktivitätsbereich **außerhalb** des zu designenden Systems liegt. Dies ist dann wichtig, wenn externe Akteure in den Prozess eingebunden werden müssen.

2.4 Aktivität

Zusammengehörige Aktionen werden zu einer **Aktivität** zusammengefasst. Diese Aktionen können sich über mehrere Verantwortungsbereiche ziehen. Diese werden entweder wie rechts dargestellt außerhalb, oder auch innerhalb der Aktivität definiert (Anm.: ein Beispiel für innerhalb sehen sie im letzten Kapitel).



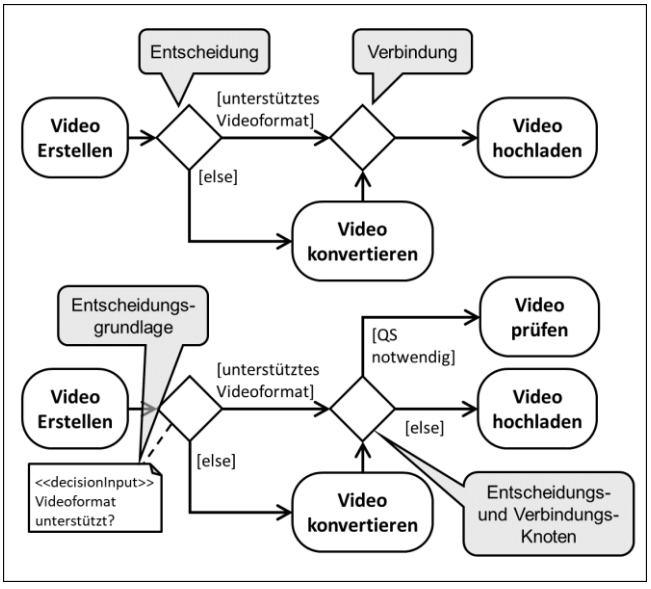
Aktivitäten können entweder als Teilaktivitäten eines größeren Diagramms notiert werden, oder – was im Regelfall vorzufinden ist – das gesamte Diagramm wird als Aktivität formuliert.

2.5 Alternative Pfade

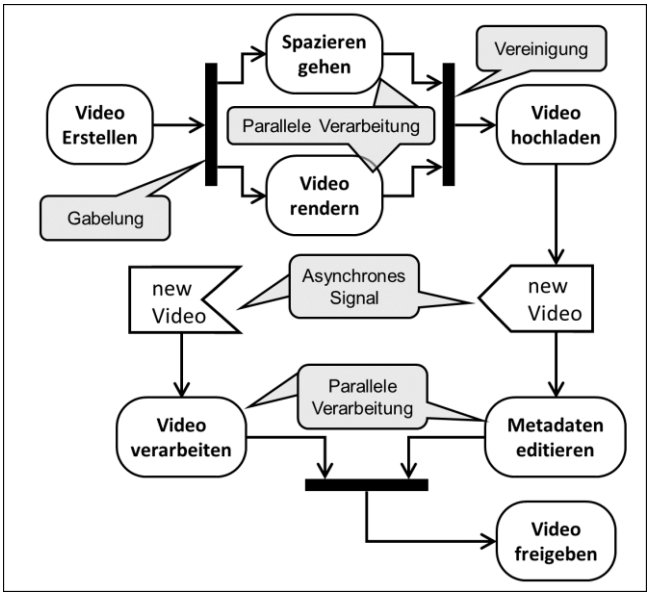
Für **Verzweigungen** in Alternative Pfade wird die Entscheidung genutzt. Sie beinhaltet entweder in den Ausgangspfaden **Guards**, welche disjunkt (also überschneidungsfrei) sein müssen oder eine Entscheidungsgrundlage mit den beiden *[true]* oder *[false]* Pfaden. **Entscheidungen** sind also „**entweder oder**“ Verzweigungen.

Das Pendant zur Entscheidung ist die **Verbindung**, welche **mehrere Eingänge** zu einem Pfad **zusammenfasst**. Dieses Element ist notwendig, da Aktionen nur einen Eingang besitzen (außer bei Parametersätzen).

Es ist auch möglich, Entscheidungen und Verbindungen in einem Symbol zusammenzufassen.



2.6 Parallele Pfade

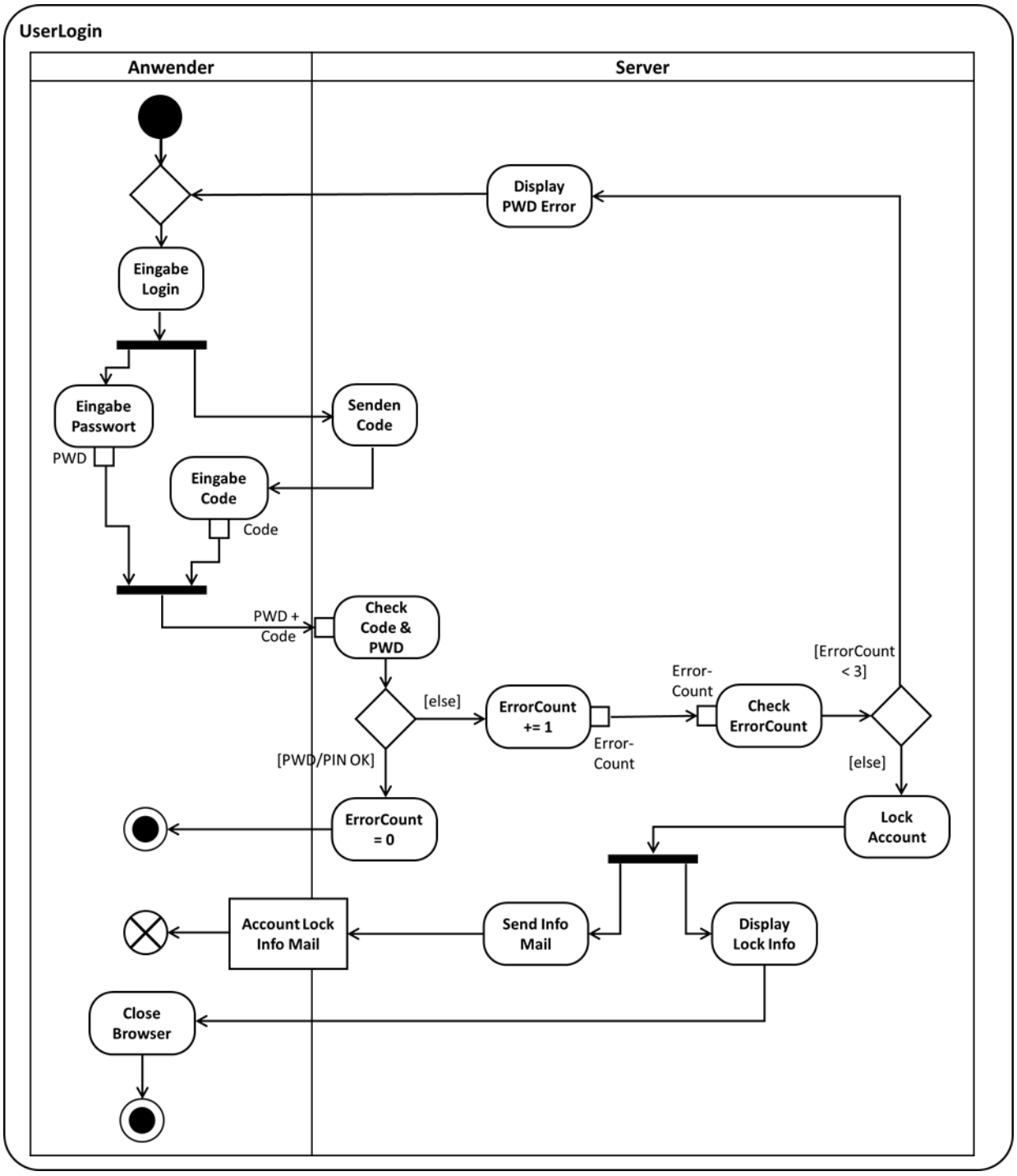


Prinzipiell gibt es zwei Möglichkeiten, eine parallele Verarbeitung zu modellieren. Die gebräuchlichste ist die **Gabelung**. Hier werden bei Erreichen des Symbols alle **ausgehenden Pfade parallel** abgearbeitet. Es können beliebig viele Pfade abgehen.

Die zweite Option ist mittels eins **asynchronen Signals**. Dieses wird von einem Akteur gesendet, wobei er nicht auf eine Rückmeldung des Empfängers wartet, sondern sofort die nächste Aktion ausführt. Der Empfänger arbeitet somit einen parallelen Zweig ab.

Das Gegenstück zur Parallelisierung ist die Vereinigung. Diese wartet, bis **alle eingehenden Pfade** abgearbeitet wurden und triggert anschließend die nächste Aktion. Dies stellt eine UND Verknüpfung dar. Sollten jedoch andere boolschen Verknüpfungen gefordert sein, so können die eingehenden Pfade eindeutig benamt werden und neben der Vereinigung die Verknüpfung näher spezifiziert werden (wenn bspw. die Pfade *A* und *B* heißen, so würde für eine *ODER* Verknüpfung stehen: *{joinSpec = A or B}*).

3 Beispiel


UserLogin
Anwender
Server
Display PWD Error
Eingabe Login
Eingabe Passwort
PWD
Senden Code
Eingabe Code
Code
PWD + Code
Check Code & PWD
[else]
ErrorCount += 1
Error-Count
Check ErrorCount
[ErrorCount < 3]
[else]
[PWD/PIN OK]
ErrorCount = 0
Lock Account
Account Lock Info Mail
Send Info Mail
Display Lock Info
Close Browser

.

4 Aufgabenstellung

Zeichnen Sie ein UML Aktivitätsdiagramm für den Prozess „Hausbau“ (ohne Verantwortungsbereiche). Verwenden Sie dabei folgende Elemente: