Studienarbeit

Benutzerstudie zur Usability einer Break The Glass Annotationssprache für Business-Processes

Tutorial zum Erlernen der BTG-Annotationssprache

Mit Übungsaufgaben

Herzlich Willkommen

- Danke für Ihr Interesse an der heutigen Nutzerstudie
- Das heutige Experiment dient der Überprüfung der Usability einer Geschäftsprozess-Annotationssprache
- Diese Sprache wurde am IPD entwickelt und soll nun hinsichtlich ihrer Praxistauglichkeit optimiert werden

Zum Ablauf der Studie

- Zuerst: Einführung/Wiederholung von Grundlagen
  Motivation der Thematik
- Dann: Aktives Erlernen der Annotationssprache anhand eines Beispiels
  Kleine Aufgabe am Ende jedes Kapitels
- Danach: Einsatz des Gelernten an drei praktischen Beispielen
- Zuletzt: Feedback mittels eines Fragebogens

Hinweise zu den Aufgaben

- Es gibt kein „Richtig“ oder „Falsch“
  - Wir wollen die Sprache überprüfen
  - Jede Antwort hilft uns weiter!
- Wenn Sie eine Aufgabe gar nicht lösen können ist das ebenfalls ein Ergebnis, das uns hilft!

- Die von uns gezeigten Musterlösungen können von Ihren Lösungen abweichen. Das heißt allerdings nicht, dass Ihre Lösungen falsch sind!

- Fragen dürfen Sie gerne stellen, ich werde diese allerdings nicht an jeder Stelle der Studie beantworten

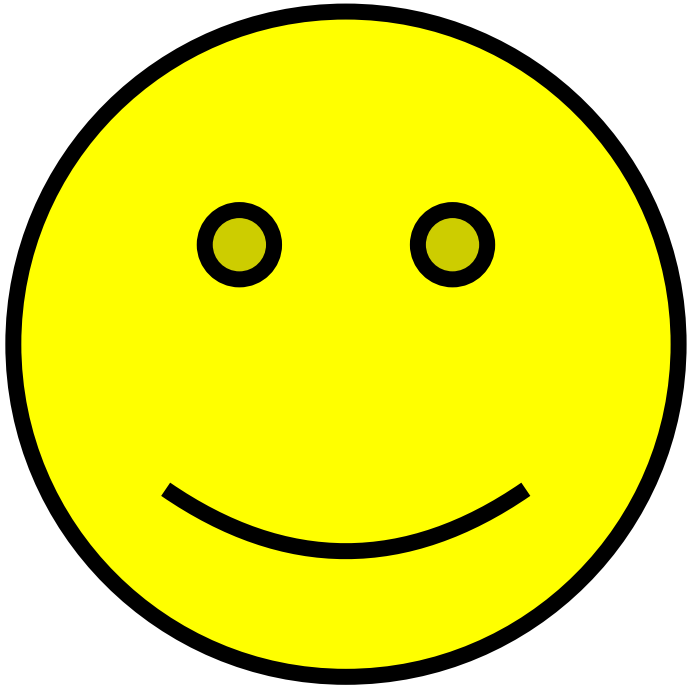
Hinweise zu den Aufgaben (2)

- Bitte konzentrieren Sie sich auf das Wesentliche!
  - Die zusätzlichen Materialien sind so einfach wie möglich gehalten
  - Die gezeigten Prozess-Diagramme sind vereinfachte Ausschnitte
  - Bitte nicht auf dadurch entstandene Formalismus-Fehler versteifen ☺

- Bitte seien Sie ehrlich!
  - Beschönigungen helfen uns nicht und sind nicht nötig, da niemandes Karriere von diesem Test abhängt
  - Wenn Sie krampfhaft Fehler suchen kann es passieren, dass wir an den falschen Stellen optimieren
  - Sagen Sie also einfach, was Sie denken

Hinweise zu den heutigen Tests (3)

- **Und nochmal:**
  Es gibt kein „Richtig“ oder „Falsch“



AGENDA

1. **Einführung und Motivation**
2. BTG-Konzept
3. Rollenverteilung
4. BTG-Annotationen
5. Obligations
6. BPCC
7. Weitere Aufgaben
8. Fragebogen

1. Einführung und Motivation

Grundidee von Workflows

- Kontrollfluss eines Programmes von dessen Ausführung trennen
- Dazu: Abbildung dieses Kontrollflusses in Diagrammen
- Allgemeiner: Abbildung von Prozessen in Diagrammen
- Diagramme werden interpretiert und ausgeführt

Business Process Model and Notation (BPMN)

- Darstellung von Geschäftsprozessen in Diagrammen
- Grundlage für heutige Thematik
- Für die Studie beschränken wir uns hierbei auf das Wesentliche

1. Einführung und Motivation (2)

Personen und Rollen

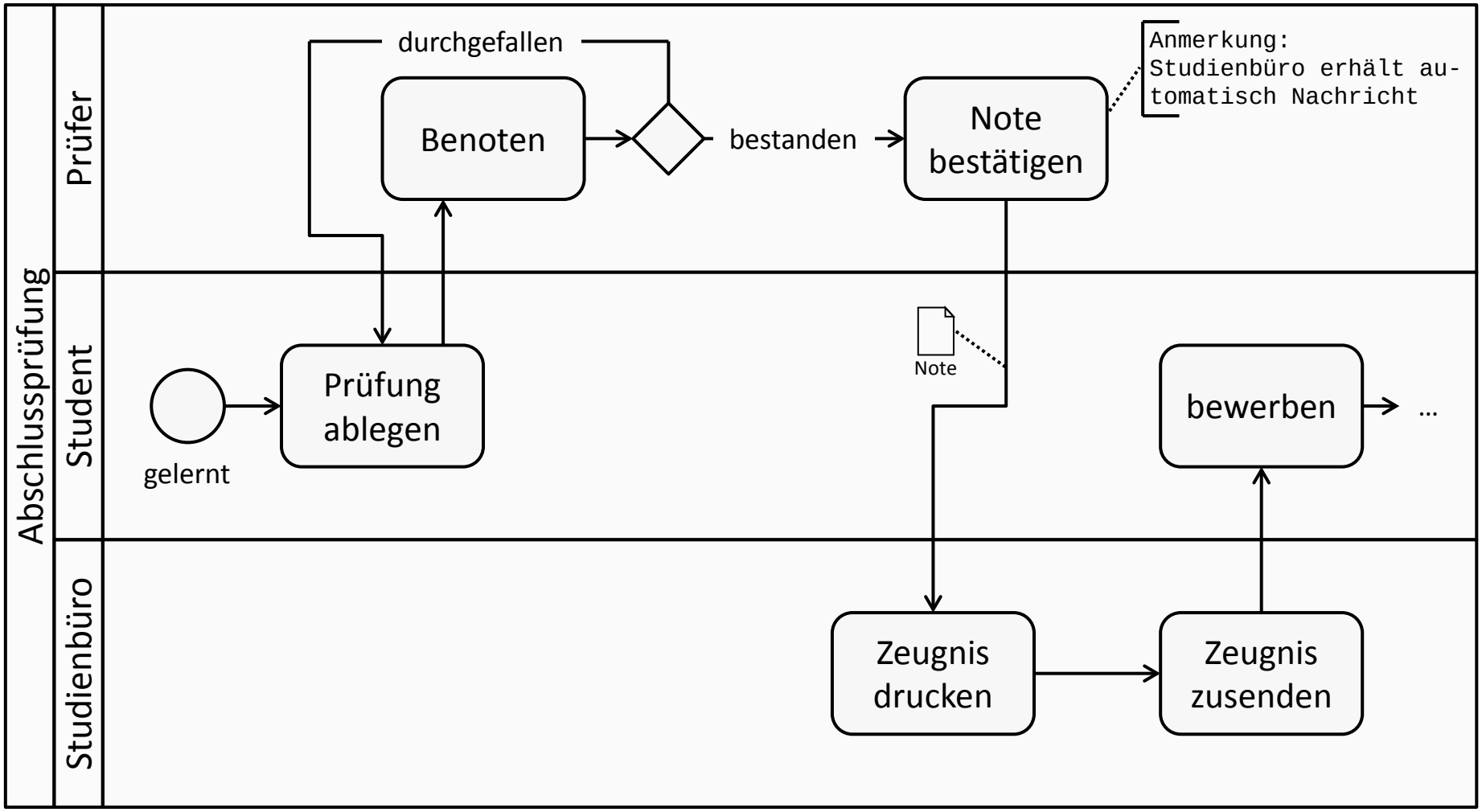
- Personen sind Akteure, die Aufgaben in einem Prozess ausführen, z.B. „Prof. K. Böhm“
- Personen sind meist Rollen zugeordnet; so hat das Individuum „K. Böhm“ die Rolle „Professor“ inne

Attribute

- Personen haben Attribute (z.B. Alter)
- Wir nutzen eine vereinfachte Sicht auf die Daten
- Syntax: Person.attribut
  - z.B. `Professor.age` für das Alter
- Für die Aufgaben nötige Attribute werden vorgegeben
- Im Zweifelsfall einfach etwas Passendes ausdenken

1. Einführung und Motivation
Beispiel eines Geschäftsprozesses

Abschlussprüfung an der Uni (mit Zeugnisvergabe)



1. Einführung und Motivation (3)
Datenobjekte

- Normalerweise haben Personen, die Datenobjekte benötigen, entsprechende Zugriffsrechte
  - z.B. Hausarzt darf auf Patientenakte zugreifen
  - z.B. Betreuer bei Versicherung darf Kundendaten einsehen
- Zugriffe auf Datenobjekte und die damit verbundenen Rechte sind im Prozess-Diagramm ausmodelliert und an Aufgaben gekoppelt

- Es gibt allerdings auch Ausnahmesituationen, in denen Personen, die normalerweise keine Zugriffsrechte haben, ausnahmsweise auf die Datenobjekte zugreifen müssen
  - z.B. Arzt in der Notaufnahme braucht Patientenakte
  - z.B. Urlaubsvertretung darf Kundendaten einsehen

1. Einführung und Motivation (4)
Ausnahme-Datenzugriffe

- Außerplanmäßige Datenzugriffe sind i.d.R. zum Modellierungszeitpunkt bekannt und müssen vom Prozessmodellierer durch die Spezifikation geeigneter Zugriffsrechte umgesetzt werden
- Die direkte Modellierung dieser Ausnahmen in BPMN-Diagrammen wäre zwar möglich, allerdings auch
  - aufwändig
  - fehleranfällig
  - schnell ausufernd

⇒ Somit Suche nach einem eleganteren Weg zur Ausnahmebehandlung

⇒ BTG: Break the Glass als Lösung

1. Einführung und Motivation (5)
Break the Glass (veranschaulicht)

- Datenobjekte befinden sich hinter Glasscheibe
  - Nutzer weiß, dass sie da sind
  - Nutzer darf allerdings nicht darauf zugreifen
- Glasscheibe zerbrechen, um an Daten zu gelangen
  1. Wunsch, an Datenobjekt zu gelangen
  2. Genehmigung nötig
  3. Randbedingungen (Hammer dabei? Alarmanlage aus?)
  4. Die Konsequenzen auf sich nehmen

⇒ Diese Schritte zum Zerbrechen der Glasscheibe müssen modelliert werden

1. Einführung und Motivation (6)
Modellierung

- Modellierung mit Hilfe einer dafür entwickelten Annotationssprache zur Umsetzung von BTG
- Kommentar-Funktion von BPMN wird genutzt, um Annotationen zu „coden“
- Für jeden außerplanmäßigen Datenzugriff ist also eine entsprechende BTG-Annotation zu modellieren
  - Dazu wird ein Kommentar an die Aufgabe, nach der das Brechen des Glases angefordert wird, angefügt
  - Vorgehen analog zu klassischen Sicherheitsaspekten von Workflows (siehe Vorlesung WfMS)
  - `<<BTG: ..... >>` kennzeichnet eine solche Annotation

⇒ Prozessmodellierer „coden“ also die Ausnahmen

1. Einführung und Motivation (7)
Umsetzung

- Prozess-Fragmente sind die „ausmodellierte“ BPMN-Version der BTG-Annotationen
- Ein Tool bildet die BTG-Annotationen auf derartige Prozess-Fragmente ab
- Die Fragmente werden vor der Ausführung an die entsprechenden Stellen des BPMN-Diagramms eingefügt

⇒ Spätere Nutzung der Sprache im Rahmen eines BPMN-Modellierungs-Tools

⇒ Die Umwandlung (Transformation) der Annotationen erfolgt für den Endanwender unsichtbar

Beispiel: Abschlussprüfung

- Ein Student erbringt seine Prüfungsleistung. Diese wird vom Prüfer (Professor, wiss. Mitarbeiter, …) benotet. Ist der Student durchgefallen wird die Prüfung wiederholt, hat er bestanden trägt der Prüfer die Note ein. Das Studienbüro wird automatisch benachrichtigt, druckt ein Zeugnis und schickt dieses an den Studenten.

- *Wir betrachten nun folgende Ausnahmesituation:*
- Ein Student möchte sich möglichst sofort nach Erbringen seiner Prüfleistung bewerben. Dafür braucht er eine Vorab-Bescheinigung seiner Note. Diese kann vom Prüfer ausgestellt und dem Studenten gegeben werden. Nach Aushändigung der Vorab-Note muss der Prüfer eine E-Mail ans Studienbüro senden.

Aufgabe 1
BP um BTG-Annotation ergänzen

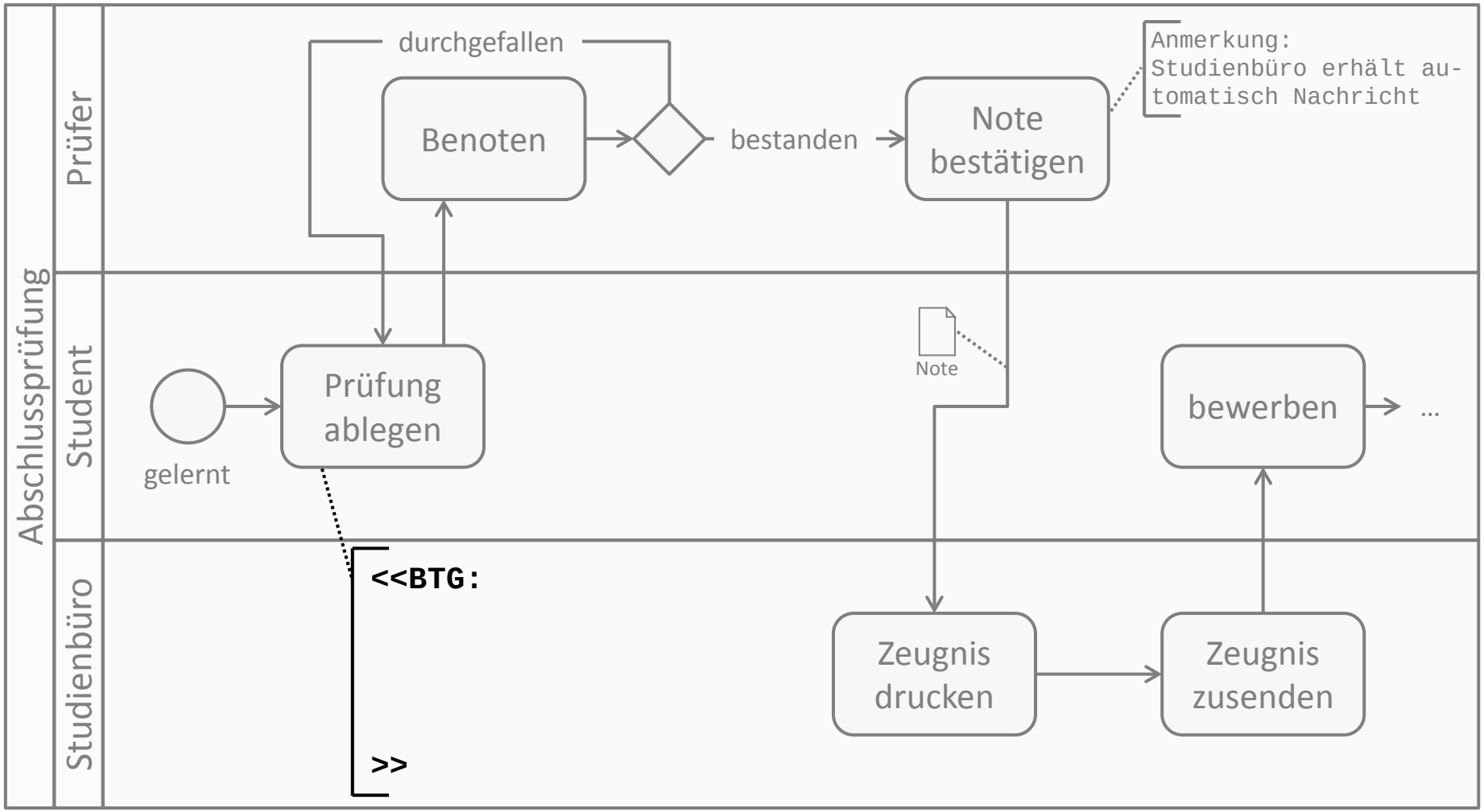
- *Angenommen, der Student möchte sich unmittelbar nach Abschluss der Prüfungen bewerben*
- *Er braucht seine Noten für diese Bewerbung*
- *Er möchte also eine inoffizielle Vorab-Note seiner Prüfung für seine Bewerbung haben*

❖ **Wo wäre hier das Glas zu brechen?**

- ✓ Fügen Sie eine Annotation an der entsprechenden Stelle des vorliegenden Geschäftsprozesses ein
- ✓ Machen Sie deutlich, dass es sich dabei um eine BTG-Annotation handelt

1. Einführung und Motivation (2)
Beispiel eines Geschäftsprozesses

Abschlussprüfung an der Uni (mit Zeugnisvergabe)



AGENDA

1. Einführung und Motivation
2. **BTG-Konzept**
3. Rollenverteilung
4. BTG-Annotationen
5. Obligations
6. BPCC
7. Weitere Aufgaben
8. Fragebogen

2. Gesamtmodell

- Die BTG-Annotationssprache besteht im Wesentlichen aus drei Elementen
  - Break The Glass Annotationen
  - Einbeziehung des Prozesskontext, z.B. Bedingungen an Ausführung
  - Zugehörige Folgeaufgaben, d.h. Konsequenzen (Verpflichtungen)

- Genauere Vorstellung dieser Elemente im Folgenden

2. Gesamtmodell (2)
Break The Glass (BTG)

- In BTG-Annotationen wird spezifiziert

  - Wer greift auf die Daten zu?
  - Wer gibt den Zugriff auf die Daten frei?
  - Authentifizierung dieser Akteure

  - Die Datenobjekte, auf die zugegriffen wird
  - Rechte zur Nutzung der Daten (R/W)

  - zusätzliche Bedingungen
  - Auflistung zugehöriger Folgeaufgaben: Obligations

2. Gesamtmodell (3)
Obligations (OG)

- Folgeaufgaben, die aus BTG resultieren (Verpflichtungen)
- Anschaulich: Beseitigung der Schäden, die durch Brechen des Glases entstanden sind
  - z.B. Scherben wegfegen
  - z.B. neue Glasscheibe kaufen
  - z.B. Hausmeister informieren
- Legen (in vorgegebenen Mustern) fest, was als Folge des Glas-Brechens zu tun ist (und wer es tun muss)

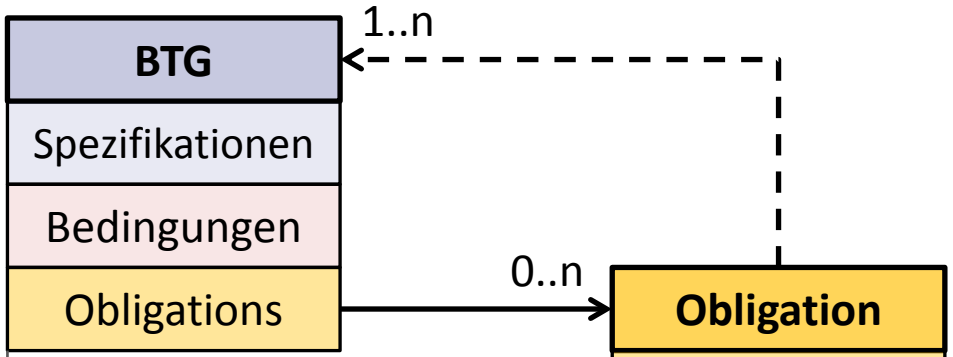
- Separate Modellierung (in eigenen Annotationen)
- Referenzierung von BTG aus

2. Gesamtmodell (4)
Business Process Context Constraints (BPCC)

- Bezug zum zugrundeliegenden Geschäftsprozesses
- Ist BTG überhaupt möglich?
  - z.B. Altersbeschränkung
  - z.B. Zeitverzug bis zum Start
- Zwei Typen von Bedingungen
  - müssen zu einem konkreten Zeitpunkt einmalig erfüllt sein
  - müssen zu einem beliebigen Zeitpunkt erfüllt sein
- Definition von Ausdrücken mit Bezug zum Kontext des Geschäftsprozesses

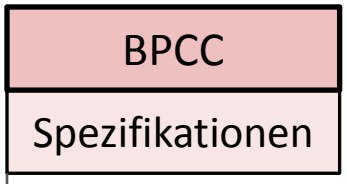
- BPCC finden sich sowohl innerhalb von BTG-Anweisungen als auch innerhalb von Obligations wieder
- Keine separate Modellierung in eigenen Annotationen

2. Gesamtmodell (5)
Übersicht



- Spezifikationen
  - Rollenzuweisungen
  - Datenobjekte
  - Rechte
- Bedingungen
  - Einschränkungen über BPCC
- Obligations
  - Resultierende Folgeaufgaben
  - Zuweisung zu entspr. Obligations

- Spezifikationen
  - Rollenzuweisungen
  - Festlegung auf eine Aufgabe
  - Parameter
- BP-CC
  - Einschränkungen

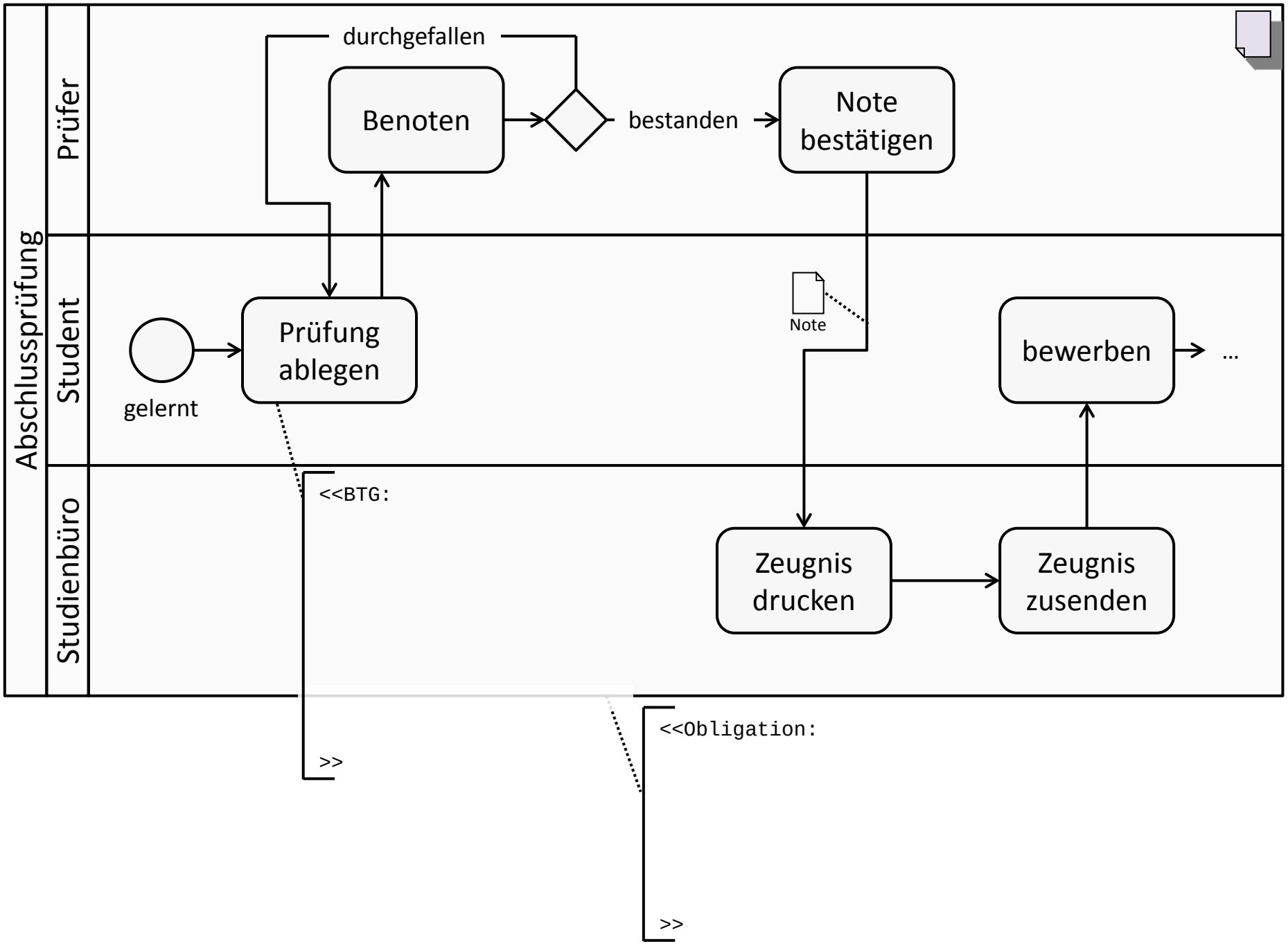


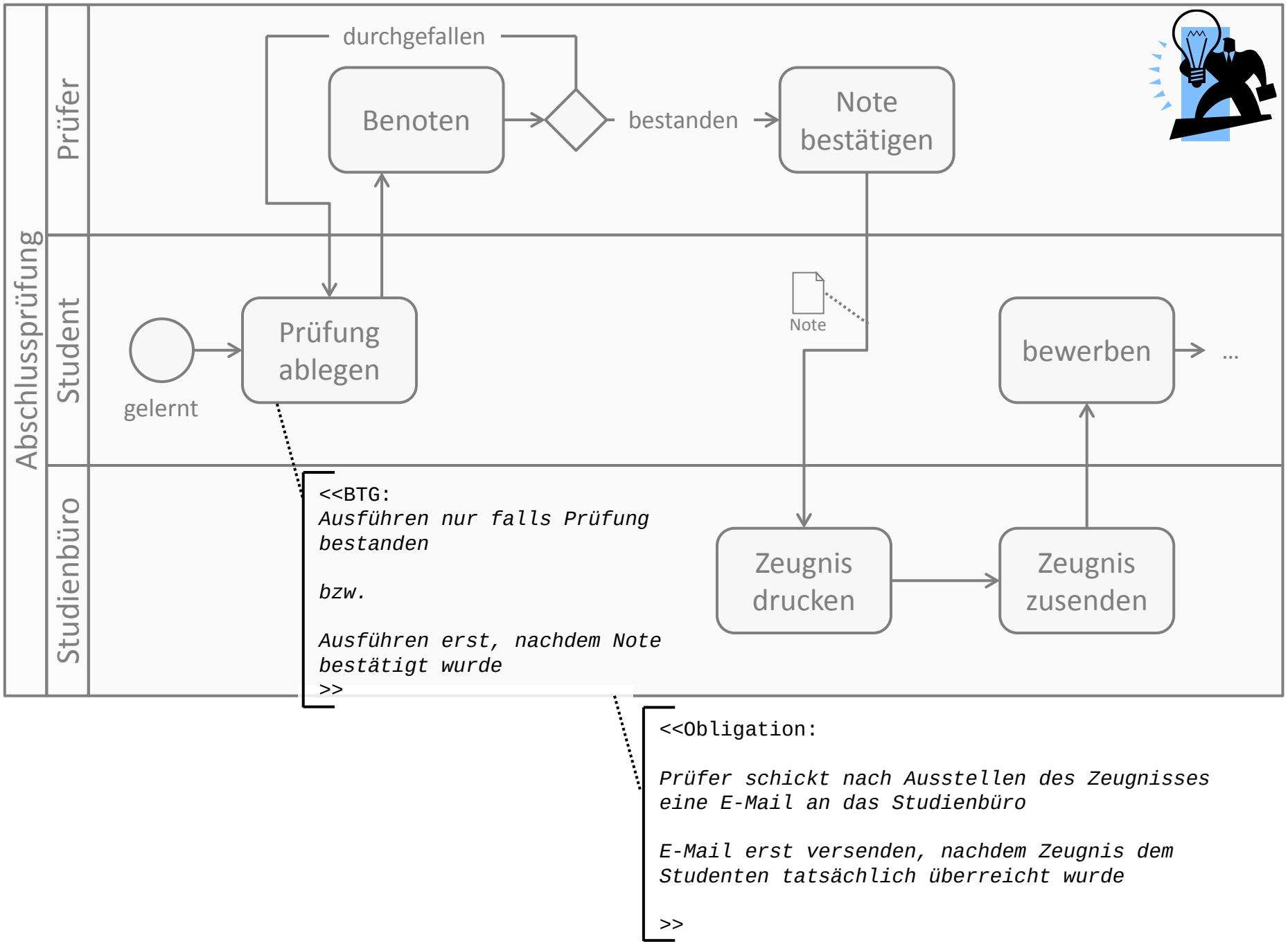
- Kontext des annotierten Geschäftsprozesses
- Beispielsweise
  - Zeitbeschränkung
  - Altersbeschränkung
  - Zeitlicher Verzug
- In BTG/ Obligation modelliert; kein separates Objekt

Aufgabe 2
Obligation und BPCC beschreiben

- **Gegeben das vorliegende Szenario und die entsprechende BTG-Annotation, beschreiben Sie verbal die nötige Obligation.**
- **Geben Sie außerdem verbal sinnvolle Rahmenbedingungen (BPCC) an**
  - ✓ Für die BTG-Annotation
  - ✓ Für die Obligation

- *Schreiben Sie Ihre Anmerkungen einfach in die entsprechenden Annotationen*


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Note
Prüfung ablegen
gelernt
bewerben
...
<<BTG:
>>
Zeugnis drucken
Zeugnis zusenden
<<Obligation:
>>


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
Ausführen nur falls Prüfung bestanden
bzw.
Ausführen erst, nachdem Note bestätigt wurde
>>
<<Obligation:
Prüfer schickt nach Ausstellen des Zeugnisses eine E-Mail an das Studienbüro
E-Mail erst versenden, nachdem Zeugnis dem Studenten tatsächlich überreicht wurde
>>

AGENDA

1. Einführung und Motivation
2. BTG-Konzept
3. **Rollenverteilung**
4. BTG-Annotationen
5. Obligations
6. BPCC
7. Weitere Aufgaben
8. Fragebogen

3. Rollen
Konzept

- Bei BTG gibt es drei verschiedene Rollen
- Rollen beschreiben die Personen, die an BTG beteiligt sein können

- Es müssen nicht immer alle Rollen definiert werden
- Eine Person kann mehrere Rollen belegen

- Die BTG-Rollen werden in BTG und OG vergeben

- Jeder BTG-Rolle wird eine Rolle des Prozesses zugewiesen, z.B. der Student im vorliegenden Beispiel

3. Rollen
Definition

- **Accessor**
  - Person, die auf Daten zugreifen will
  - fordert BTG an, damit also zugleich Initiator
  - in BTG-Annotation spezifiziert

- **Activator**
  - Person, die Ausnahme-Zugriff genehmigt oder verweigert
  - i.d.R. nicht Person, die Ausnahme-Zugriff anfordert
  - in BTG-Annotation spezifiziert

- **Compensator**
  - zuständig für Folgen der Ausnahme (Reparatur)
  - in Obligation definiert

3. Rollen
Notwendigkeit / Default

- Accessor
  - Default: Ausführender der BTG-annotierten Aktivität
- Activator
  - Default: kein Activator
- Compensator
  - Default: Automatisierte, nicht-menschliche Ausführung
  - Gehören zu einer BTG-Annotation mehre Obligations, so kann es für jede Obligation einen eigenen Compensator geben

3. Rollen
Variable BTG-Rollen

- Es ist möglich, mehrere Rollen für Activator oder Accessor oder Compensator zu spezifizieren (durch Auflistung)
- Dies entspricht einem logischen „oder“; jeder von ihnen kann also die Rolle im Rahmen eines BTG-Zugriffes einnehmen
- eine Rangfolge wird so nicht festgelegt

- Dass mehrere Personen gleichzeitig/gemeinsam eine der BTG-Rollen ausführen ist nicht möglich

Aufgabe 3
Rollen vergeben

- **Bitte benennen Sie im vorliegenden Beispiel der Abschlussprüfung die drei involvierten Rollen**

✓ Accessor:
✓ Activator:
✓ Compensator:

Aufgabe 3
Rollen vergeben

- **Bitte benennen Sie im vorliegenden Beispiel der Abschlussprüfung die drei involvierten Rollen**

✓ Accessor:	**Student** *ODER* *weglassen (default)*
✓ Activator:	**Prüfer**
✓ Compensator:	**Prüfer** *(versendet E-Mail)*

AGENDA

1. Einführung und Motivation
2. BTG-Konzept
3. Rollenverteilung
4. **BTG-Annotationen**
5. Obligations
6. BPCC
7. Weitere Aufgaben
8. Fragebogen

4. Break The Glass
Aufbau

Pflichtelemente

- Beginn und Ende mit doppelten spitzen Klammern
- Am Anfang zusätzlich BTG:
- Datenobjekte und Rechte

```
<<BTG:
{Roles}
objects: …
rights: …
{Conditions}
{Obligations}
>>
```

Optionale Elemente

- Spezifikation der BTG-Rollen (Accessor, Activator)
- Angabe möglicher Einschränkungen (BPCC)
- Auflistung zugehöriger Obligations

4. Break The Glass
Einschub zur Terminologie

- `something: list(x)`
  auf den Folien bedeutet
  `something: a, b, c, …`

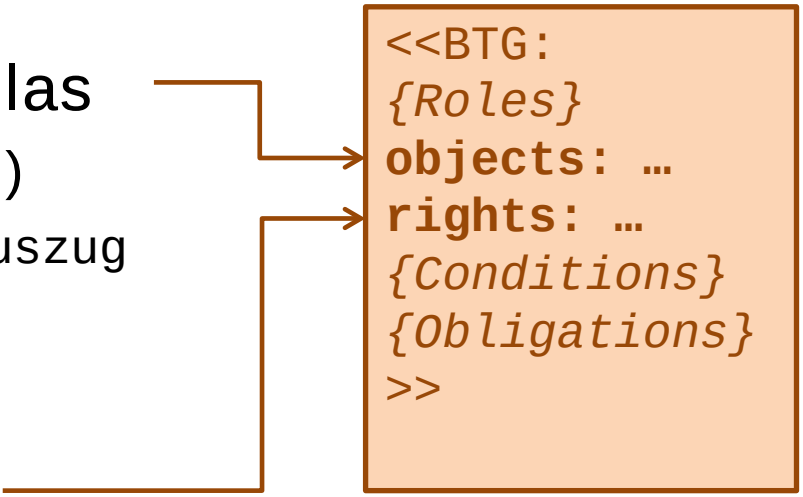
- `something: list([x,y])`
  auf den Folien bedeutet
  `something: [a,b], [c,d], …`

- Variablen werden grundsätzlich mit
  `$Variable`
  gekennzeichnet (z.B. für Datenobjekte)

- Strings in Anführungszeichen: `„Stringtext“`

4. Break The Glass Elemente

- Datenobjekt(e) hinter dem Glas
  - Ein Objekt oder mehrere (a,b,...)
  - z.B.: `objects: $Student.notenauszug`
- Rechte auf dem Datenobjekt
  - `read` ODER `write` ODER beides
  - read und write sind disjunkt!
  - delete, create o.ä. sind aktuell nicht vorgesehen
  - z.B.: `rights: read,write`
- Diese Rechte gelten, nachdem das Glas gebrochen wurde



4. Break The Glass Elemente (2)

- Zuweisung der Rollen
  - Accessor
  - Activator
  - ~~Compensator~~ (erst mit Obligations)

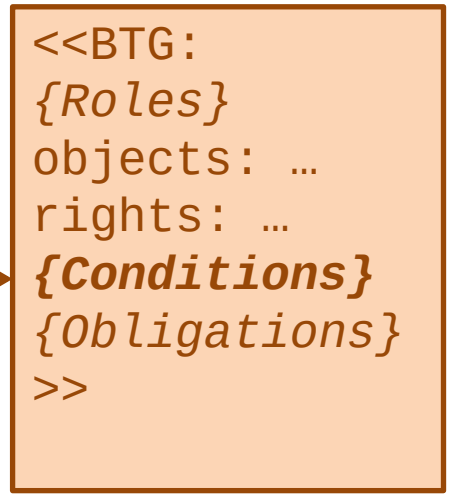
```
<<BTG:
{Roles}
objects: …
rights: …
{Conditions}
{Obligations}
>>
```

- z.B.: `activator.role: Student`
- In der Praxis meist Authentifizierung gewünscht
  - Auflistung als Tripel
    `activator.authn: [$what,$how,$where]`
    - `$what` spezifiziert die Quelle, z.B. Studiausweis
    - `$how` spezifiziert was von der Quelle gelesen wird, z.B. Matrikelnummer
    - `$where` spezifiziert optional den Identitätsprovider, z.B. idp.kit.edu
  - z.B.: `activator.authn: [Studiausweis,MN,ipd.kit.edu]`

BTGAccessor analog!

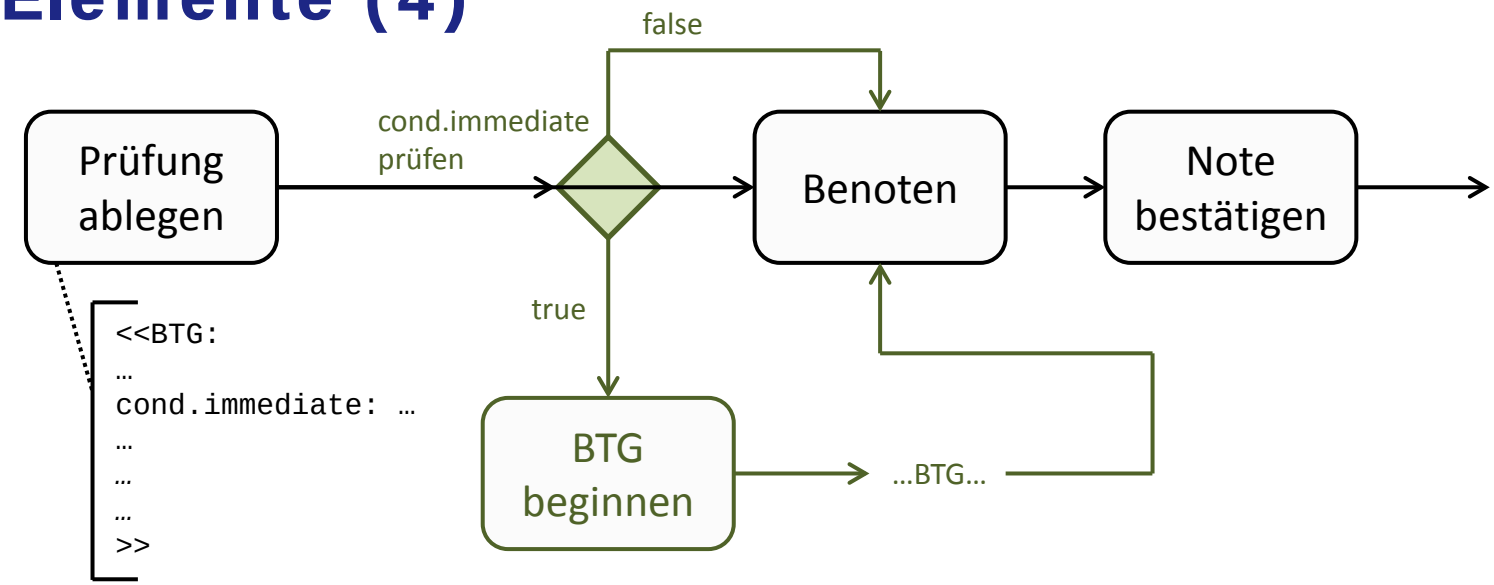
4. Break The Glass
Elemente (3)

- Business Process Context Constraints
  - Einschränkungen bzgl. Prozesskontext
  - Bezug zum zugrundeliegenden BP
  - Syntax und Semantik später



- Zwei Arten von Bedingungen
  - Einmalig zu prüfende Bedingungen:
    `cond.immediate: …`
    müssen bei der Ausführung von BTG gültig sein
    - z.B. Alter einer Person >= 18
  - Wiederholt zu prüfende Bedingungen:
    `cond.anytime: …`
    müssen zu einem beliebigen Zeitpunkt gültig sein, damit BTG gestartet werden kann
    - z.B. Aktivität T wurde ausgeführt
  - Syntax und Semantik mittels BPCC (später)

4. Break The Glass Elemente (4)

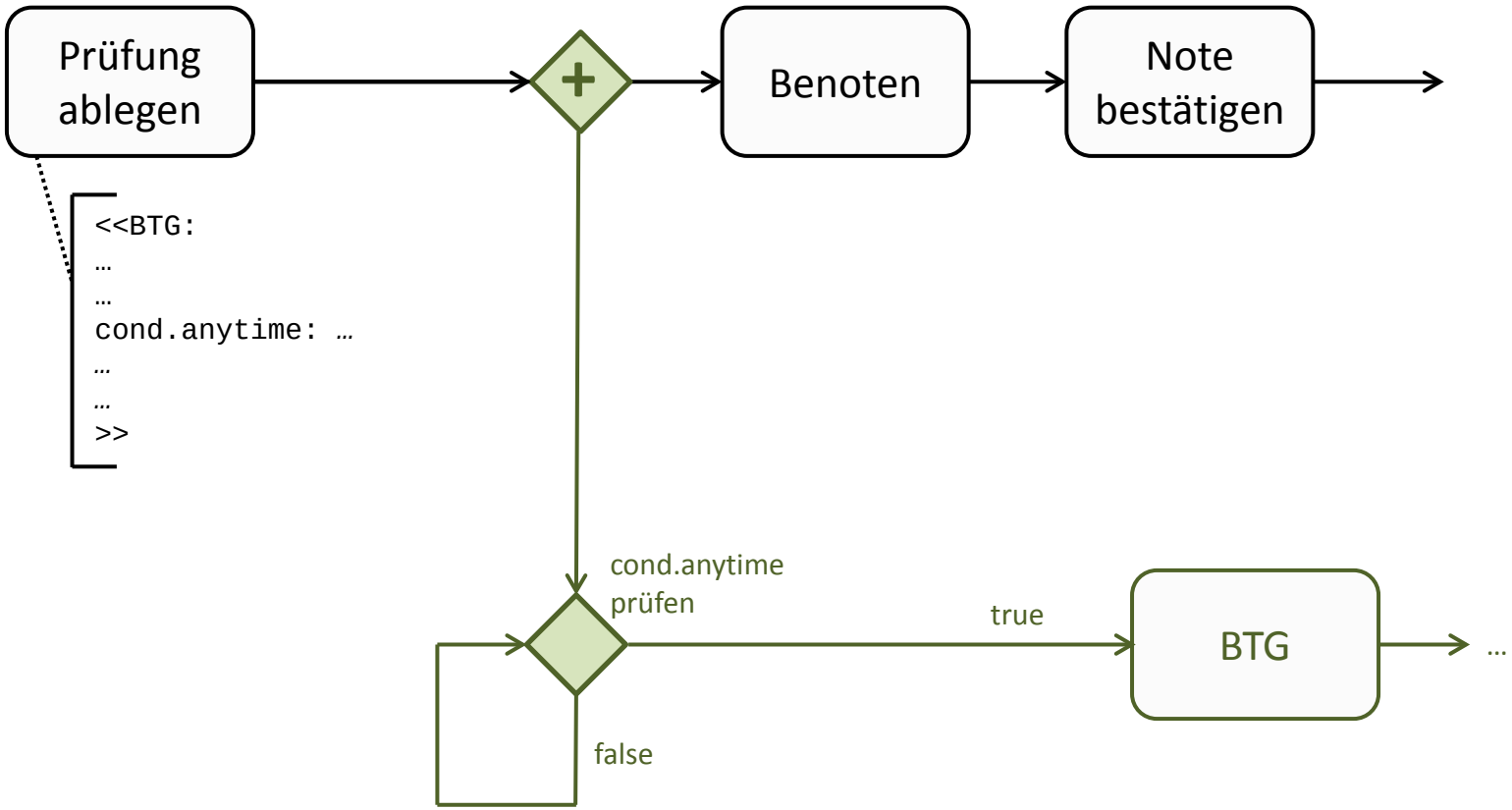



```
<<BTG:
{Roles}
objects: …
rights: …
{Conditions}
{Obligations}
>>
```

vereinfachte Darstellung der BTG-Bedingungen

4. Break The Glass Elemente (4)

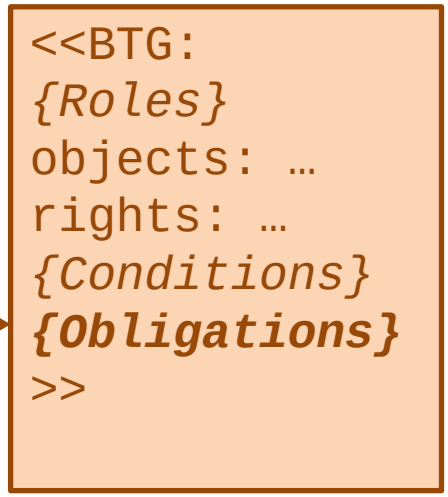
```
<<BTG:
{Roles}
objects: …
rights: …
{Conditions}
{Obligations}
>>
```



vereinfachte Darstellung der BTG-Bedingungen

4. Break The Glass Elemente (5)

- Folgeaufgaben zur Reparatur des gebrochenen Glases
- Separate Modellierung
- In BTG-Annotation nur Auflistung aller Obligations



- `obligations: list($obligation-ID)`
  - z.B.: `obligations: O1,O5,O6`

4. Break The Glass
Zusammenfassung

Aufbau

- `<<BTG: … … … … >>`
- Datenobjekte und Rechte sind Pflichtangaben
- Rollenspezifikationen optional
- Obligations werden nur referenziert
- BPCC werden innerhalb von BTG-Annotation ausformuliert (anytime und immediate)

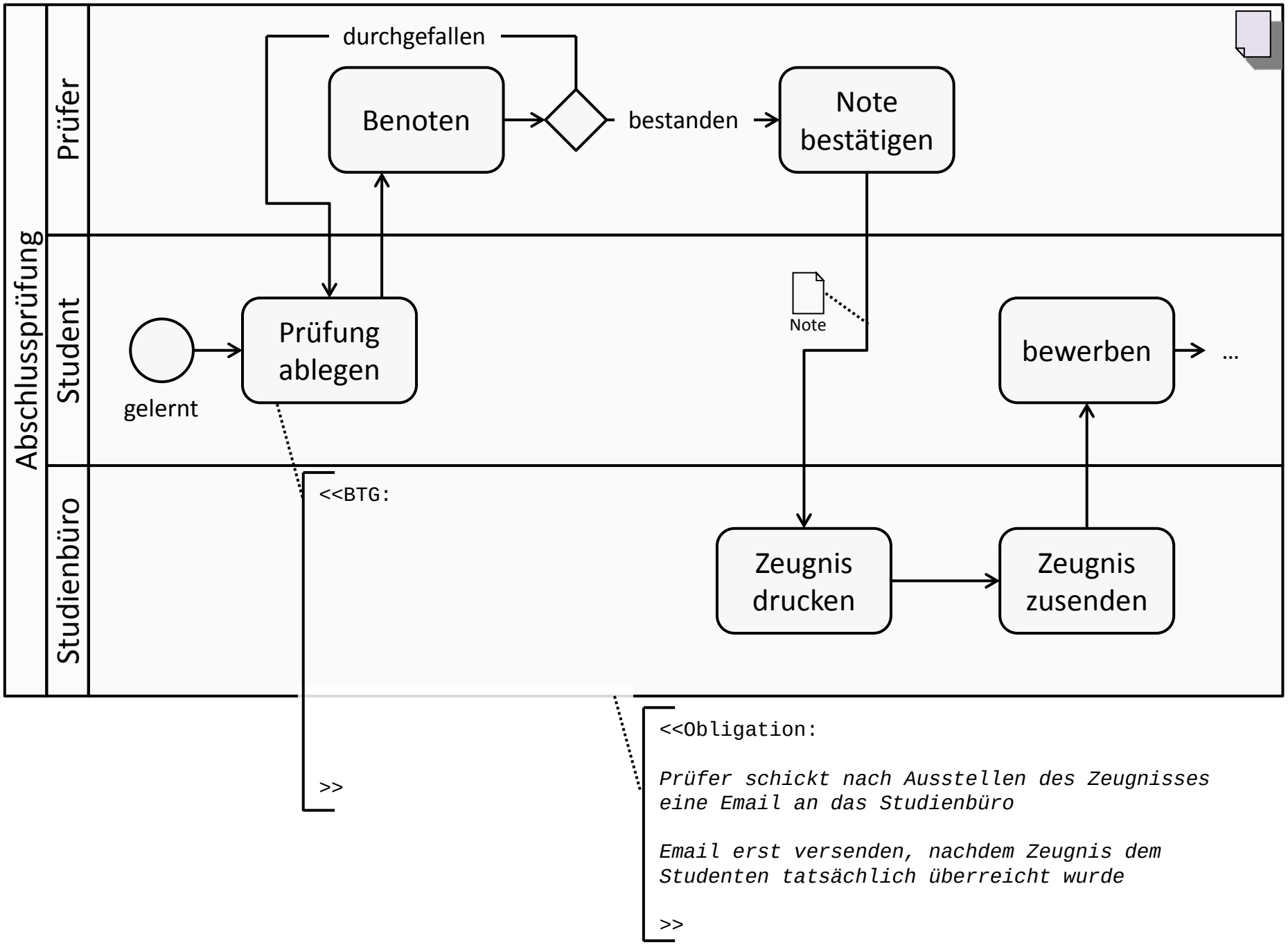
Syntax allgemein

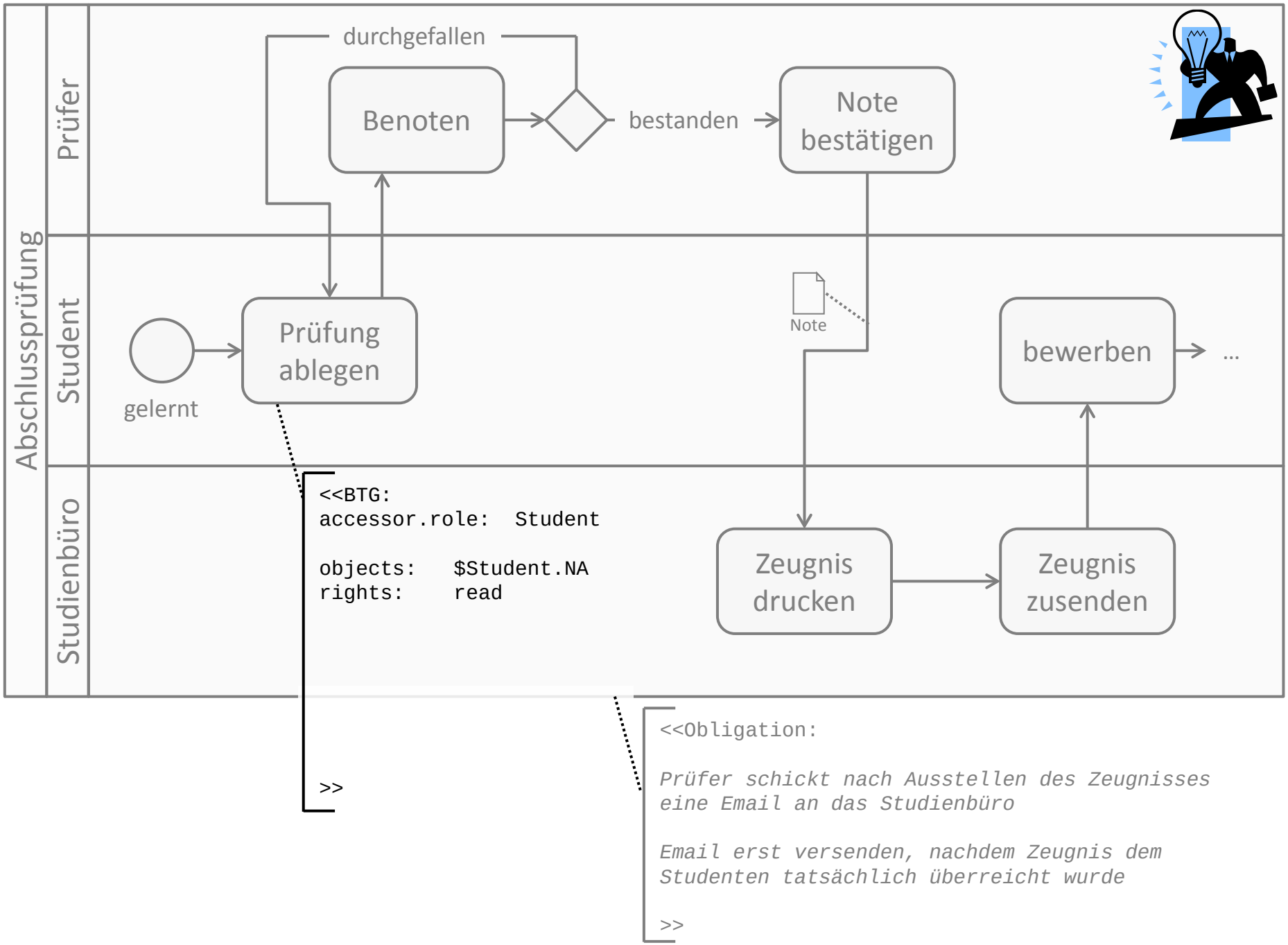
- `something: $whatever`
- Auflistungen mittels Kommata
- Tupel/Tripel in eckigen Klammern
- Kein Semikolon

Aufgabe 4.1
BTG – Grundlagen

- **Modellieren Sie die BTG-Annotation im vorliegenden Notenbeispiel. Modellieren Sie dabei Folgendes:**
  - ✓ Den Studenten als Accessor ohne Authentifizierung
  - ✓ Das Datenobjekt, nämlich den Notenauszug des Studenten ($Student.NA)
  - ✓ Die darauf benötigten Leserechte des Studenten

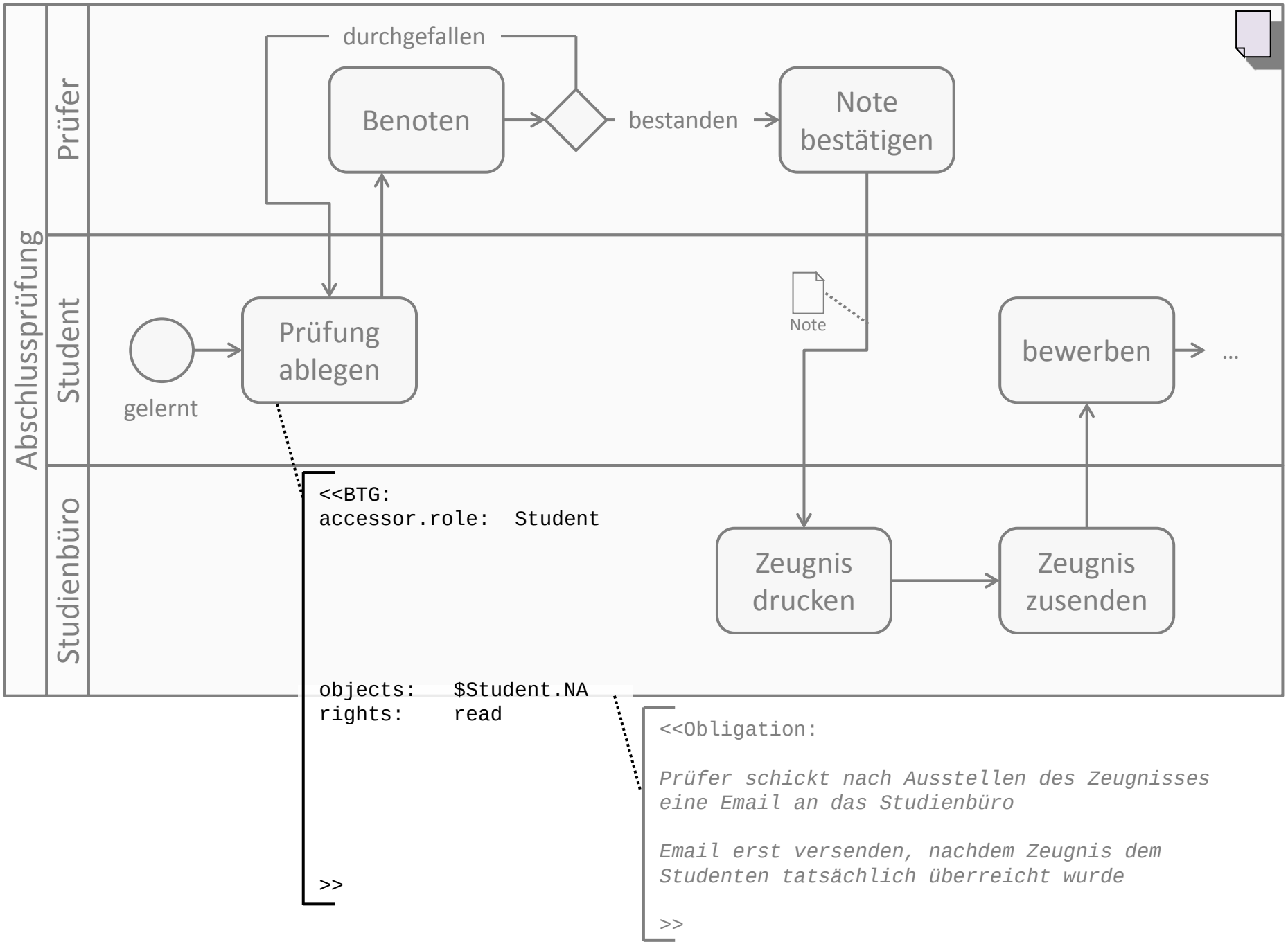
- **Im Rahmen dieser Aufgabe sind nicht zu formulieren:**
  - ✓ BPCC
  - ✓ Obligations
  - ✓ Activator

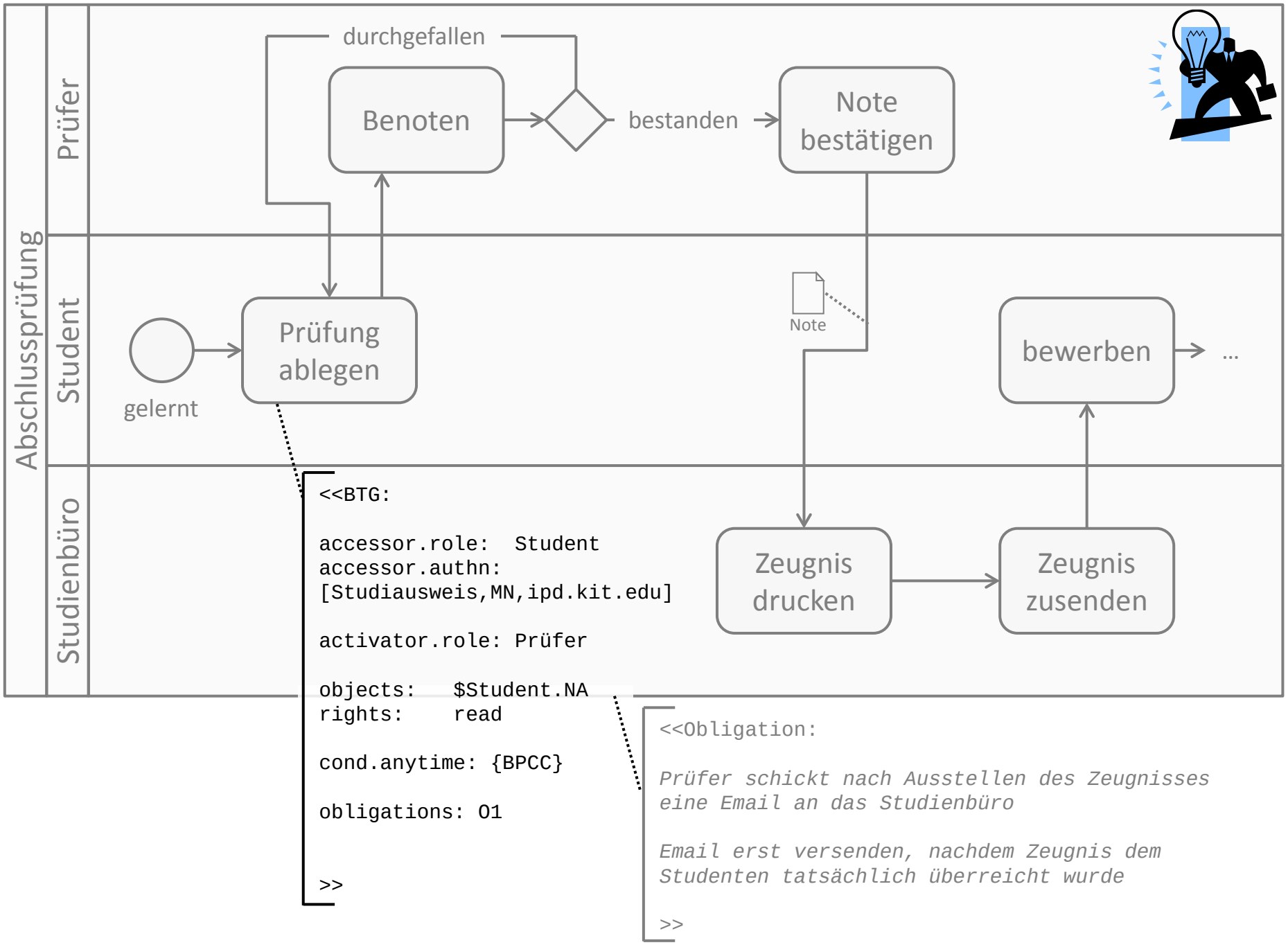

Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
<<BTG:
>>
Zeugnis drucken
Zeugnis zusenden
<<Obligation:
Prüfer schickt nach Ausstellen des Zeugnisses eine Email an das Studienbüro
Email erst versenden, nachdem Zeugnis dem Studenten tatsächlich überreicht wurde
>>


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
objects: $Student.NA
rights: read
>>
<<Obligation:
Prüfer schickt nach Ausstellen des Zeugnisses eine Email an das Studienbüro
Email erst versenden, nachdem Zeugnis dem Studenten tatsächlich überreicht wurde
>>

Aufgabe 4.2
BTG für Fortgeschrittene

- **Erweitern Sie die Modellierung aus dem ersten Aufgabenteil um die bisher fehlenden Aspekte:**
  - ✓ Den entsprechenden Activator
  - ✓ Der Student muss sich authentifizieren
    - *Die Matrikelnummer (MN) seines Studiausweises wird geprüft*
    - *Authentifizierungsdaten werden bereitgestellt von ipd.kit.edu*
  - ✓ Eine wiederholt zu prüfende Bedingung (anytime) an den Beginn der Ausführung
    - *BTG soll möglich sein, sobald eine BPCC erfüllt ist*
    - *Verwenden Sie für jegliche BPCC bitte „{BPCC}“ als Platzhalter*
  - ✓ Die Verknüpfung zur zugehörigen Obligation
    - *zum Versenden einer E-Mail*
    - *mit der ID O1*


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
objects: $Student.NA
rights: read
>>
<<Obligation:
Prüfer schickt nach Ausstellen des Zeugnisses eine Email an das Studienbüro
Email erst versenden, nachdem Zeugnis dem Studenten tatsächlich überreicht wurde
>>


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn:
[Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.anytime: {BPCC}
obligations: O1
>>
<<Obligation:
Prüfer schickt nach Ausstellen des Zeugnisses eine Email an das Studienbüro
Email erst versenden, nachdem Zeugnis dem Studenten tatsächlich überreicht wurde
>>

AGENDA

1. Einführung und Motivation
2. BTG-Konzept
3. Rollenverteilung
4. BTG-Annotationen
5. **Obligationen**
6. BPCC
7. Weitere Aufgaben
8. Fragebogen

5. Obligationen
Einführung

- Syntaktischer und semantischer Aufbau verlaufen nach dem gleichen Muster wie BTG-Annotationen
- Annotationen analog gekennzeichnet
- BPCC analog verwendet

- Formulierung im Rahmen eines Musters (pattern)
- Muster definieren die allgemeine Obligations-Aufgabe und die dafür festzulegenden Parameter
  - z.B.: E-Mail versenden, Parameter wie Sender/Empfänger

5. Obligationen
Aufbau

```
<<OG:
id: …
{Role}
pattern: …
{parameters}
{Conditions}
>>
```

Pflichtelemente

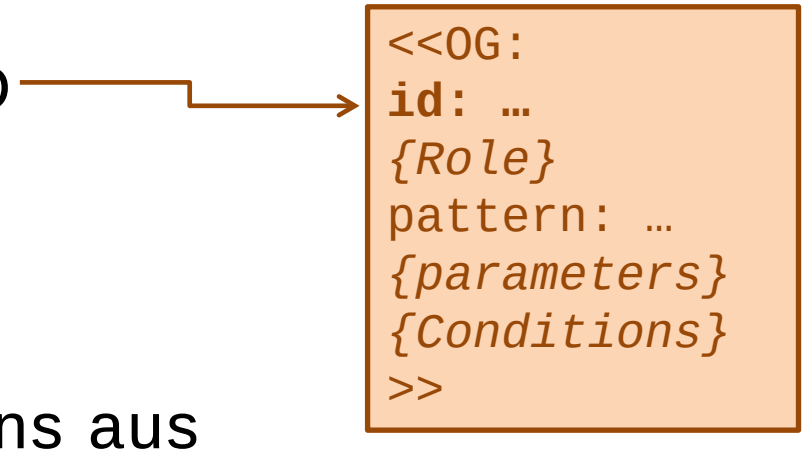
- Beginn und Ende mit doppelten spitzen Klammern
- Am Anfang zusätzlich `OG:`
- Vergabe einer id (numerisch)
- Festlegung auf ein Muster (Aufgabentyp)

Optionale Elemente

- Definition der Compensator-Rolle
- Auflistung der Parameter für das Pattern (verpflichtend, falls das Pattern Parameter verlangt)
- Angabe möglicher Einschränkungen (BPCC)

5. Obligationen
Elemente

- Vergabe einer eindeutigen ID
  - Ox, wobei x eine Zahl ist
  - `id: 043`



- Referenzierung der Obligations aus BTG-Annotationen heraus über diese Identifikationsnummer

5. Obligationen
Elemente (2)

- Definition der Compensator-Rolle
  - optional
  - nur nötig, falls Obligation von Menschen (und nicht automatisiert) erfolgt

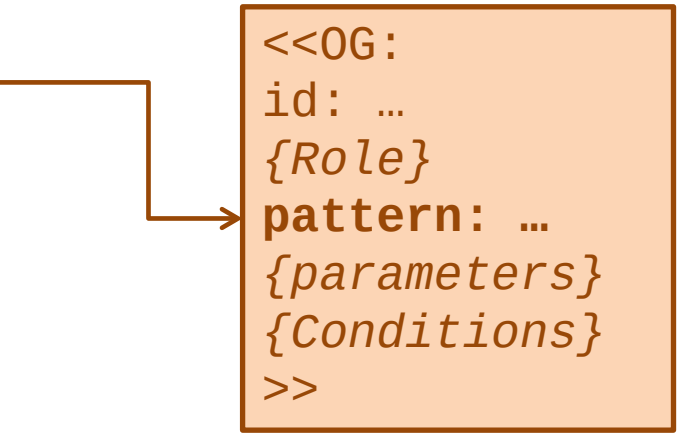
```
<<OG:
id: …
{Role}
pattern: …
{parameters}
{Conditions}
>>
```

- `compensator.role: Arzt`
- In der Praxis meist Authentifizierung gewünscht
  - Auflistung aller Tripel, z.B.
    `compensator.authn:`
    `[Account, Passwort, idp.krankenhaus.intern]`

Accessor und Activator existieren hier nicht!

5. Obligationen
Elemente (3)

- Festlegung der in der Obligation durchzuführenden Aufgabe
- Dazu: Festlegung auf ein Muster
- Muster sind vorgegeben, aktuell:
  - `SendEmail`
  - `AuditAccess`

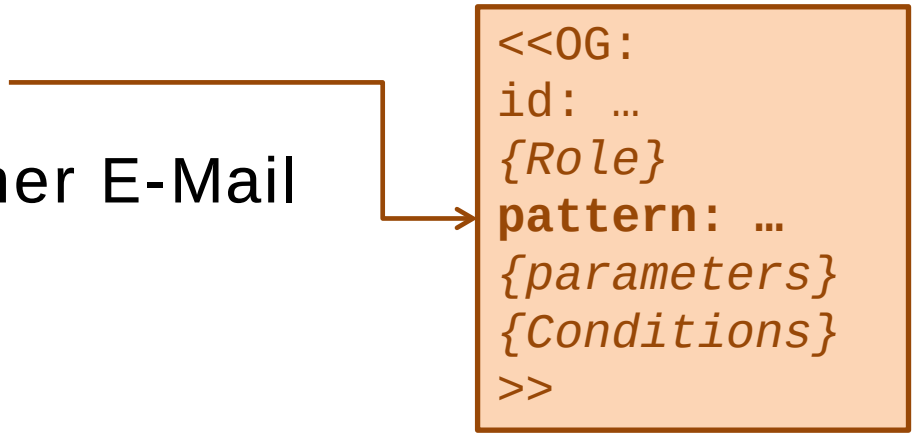


- Pro Obligation wird genau ein Muster gewählt
- Gegebenenfalls sind Parameter zu definieren

- Zusätzliche Muster können vom Prozessmodellierer in BPMN einmalig modelliert und dann im Rahmen von Obligations immer wieder verwendet werden

5. Obligationen
Elemente (4)

- pattern: SendEmail
- Dient dem Versenden einer E-Mail



Parameter

- from Sender
- to Empfänger
- subject Betreffzeile
- body Inhalt der E-Mail
- attachment Dateianhang

5. Obligationen
Elemente (6)

- `pattern: AuditAccess`
- Nachvollziehen eines Datenzugriffs
- Als Folge eines BTG-Zugriffs

```
<<OG:
id: …
{Role}
pattern: …
{parameters}
{Conditions}
>>
```

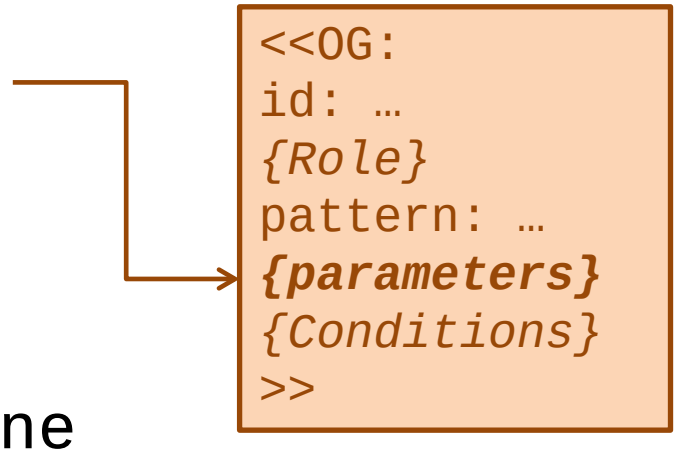
Parameter

- auditpolicy — Definition einer Richtlinie, in der Objekte festgehalten werden sollen
- start — Zeitpunkt, zu dem Audit beginnt
- end — Zeitpunkt, zu dem Audit endet

Heute außen vor ☺

5. Obligationen Elemente (7)

- Spezifikation der Parameter für das gewählte Muster
- festzulegende Parameter hängen vom Muster ab
- Falls für das gewählte Muster keine Parameter nötig sind entfällt `parameters`

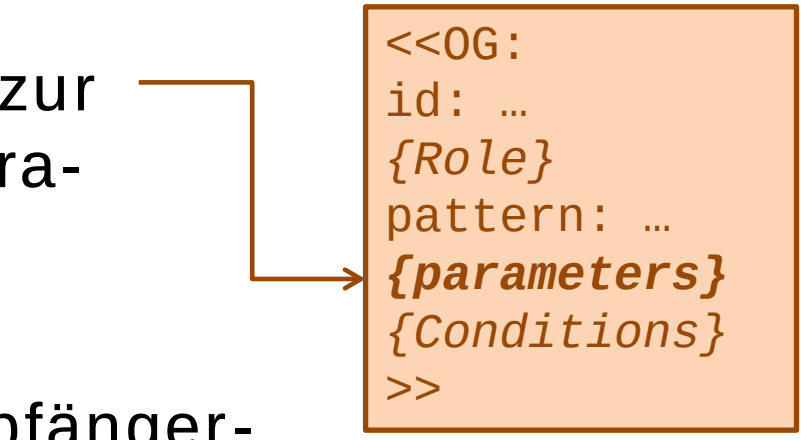


- parameters: list([name,value])
  - `parameters: [from,Arzt],[to,Patient],[subject,“Ihr Arztbesuch“],[body,“Hallo…“]`

- Falls ein einzelner Parameter nicht nötig ist wird dieser weggelassen (z.B. Dateianhang)

5. Obligationen Elemente (8)

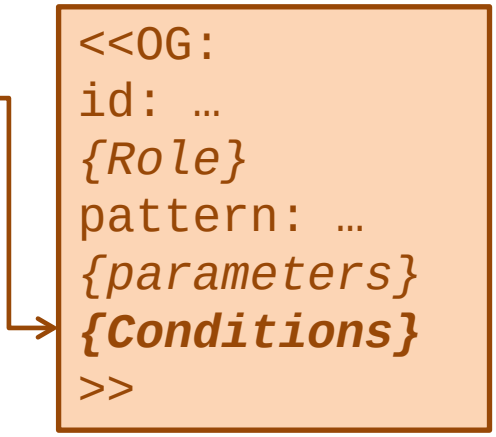
- Verwendung von BPCC auch zur Festlegung der Werte von Parametern



- Es sei beispielsweise der Empfänger-Parameter des „SendEmail“-Patterns gegeben
- Empfänger könnte nun genau die Person sein, die die Aufgabe „Task4“ ausgeführt hat
- Die Referenzierung dieser Person wäre hier über einen BPCC möglich
  - `[to,{BPCC}]`

5. Obligationen
Elemente (9)

- Business Process Context Constraints
  - Einschränkungen bzgl. Kontext
  - Bezug zum zugrundeliegenden BP
  - Syntax und Semantik später



- Zwei mögliche Bedingungen
  - analog zu BTG
  - `cond.immediate: …`
  - `cond.anytime: …`

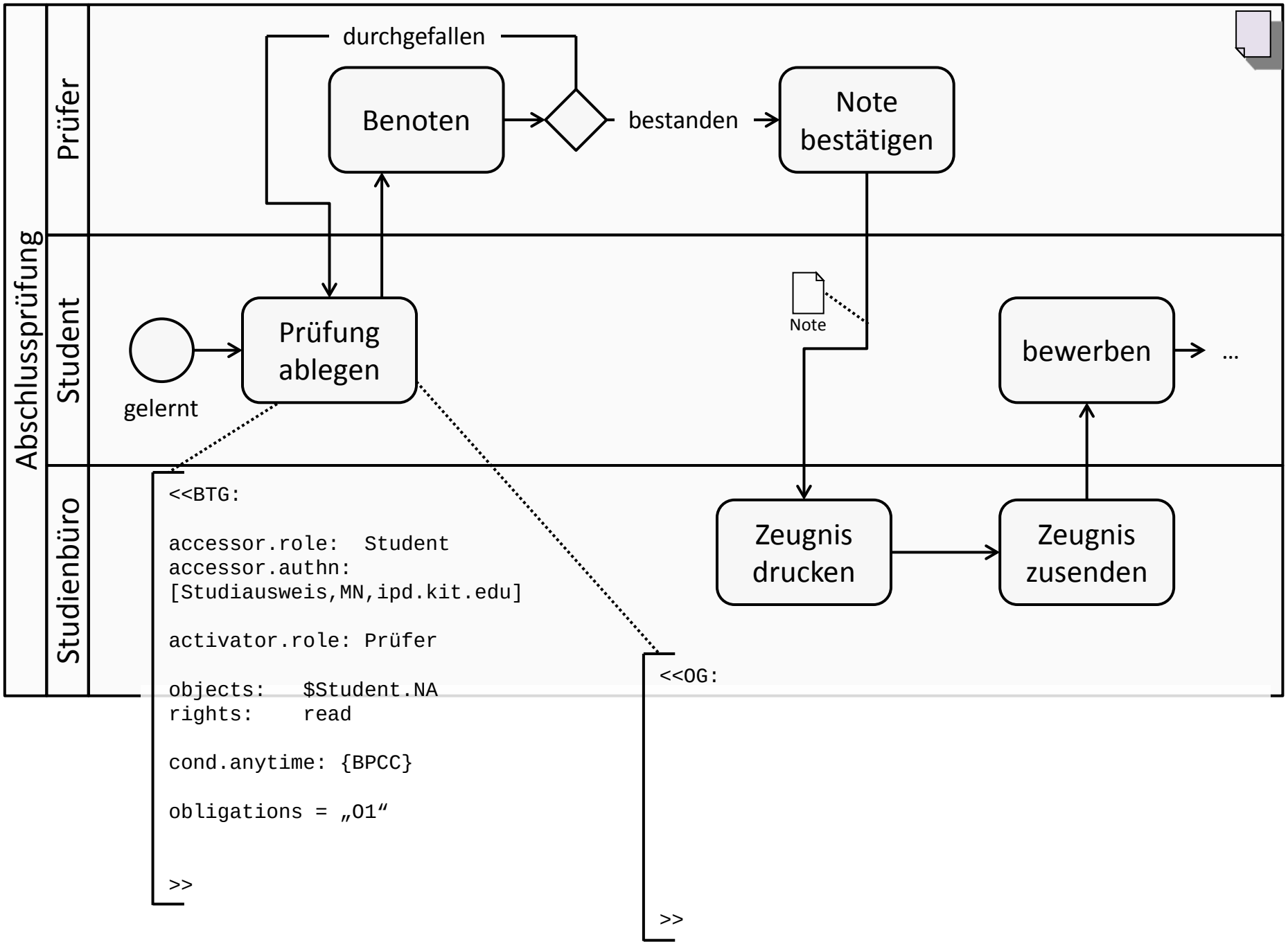
- Default
  - Obligations werden automatisch direkt nach BTG ausgeführt; hierfür ist also kein zusätzlicher BPCC nötig

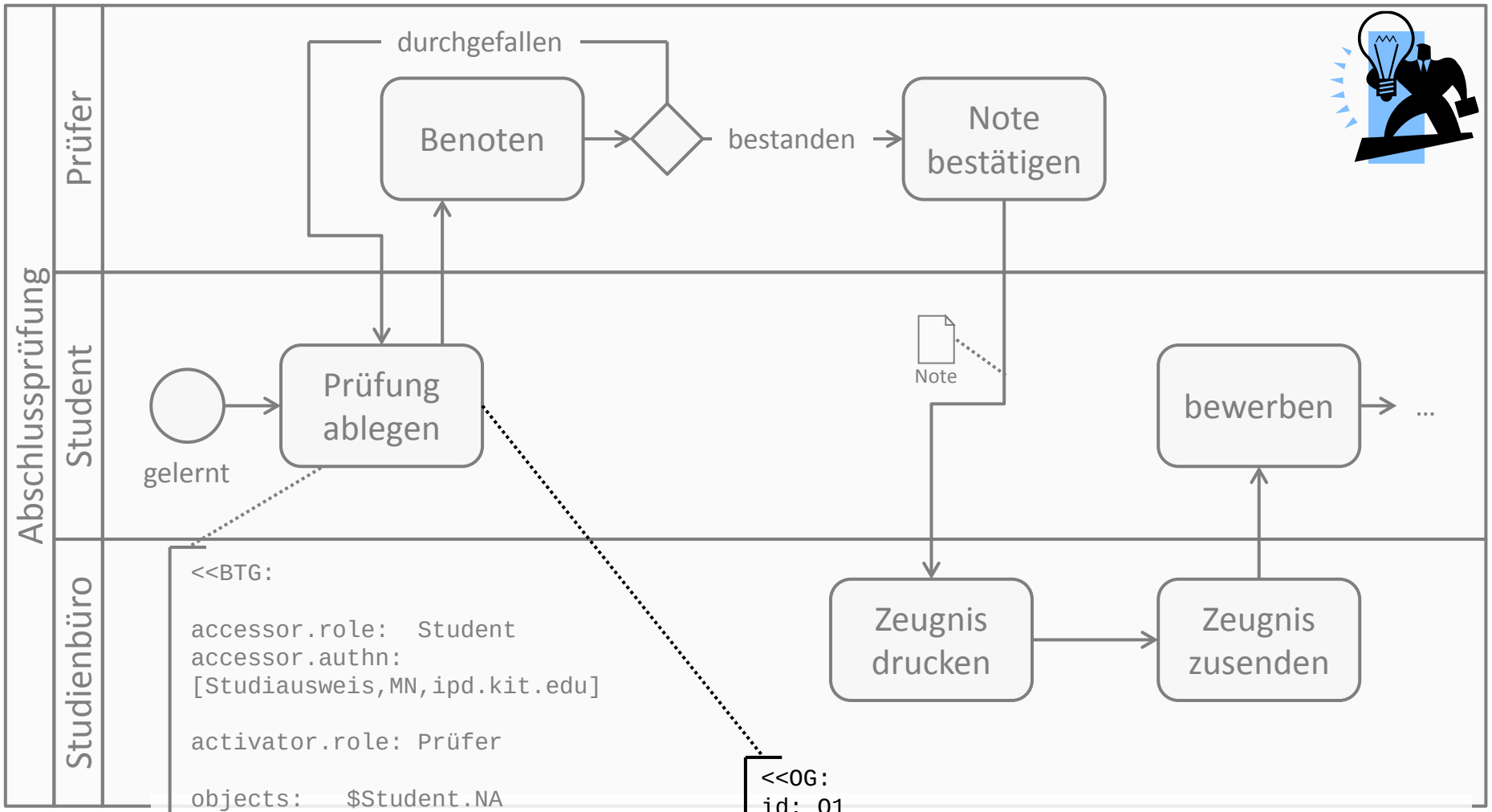
5. Obligationen
Zusammenfassung

- Aufbau ähnlich zu BTG-Annotationen
- Festlegung einer ID
- ggf. Compensator-Rolle spezifizieren
- Festlegung auf eine Aufgabe: pattern
- Definition der Parameter für gewähltes Muster
- Rahmenbedingungen mittels BPCC festlegen

Aufgabe 5
Modellierung einer Obligation

- **Modellieren Sie die Obligation O1 zum Versenden einer E-Mail (nach der BTG-Ausführung) durch den Prüfer für das Notenvergabebeispiel**
- **Verwenden Sie für die Parameter folgende Werte:**
  - ✓ Absender: Prüfer
  - ✓ Empfänger: Studienbüro
  - ✓ Betreff: „Note“
  - ✓ Inhalt: „wurde bekanntgegeben“
  - ✓ Anhang: *keiner*


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn:
[Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.anytime: {BPCC}
obligations = „O1“
>>
<<OG:
>>


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn:
[Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.anytime: {BPCC}
obligations = „O1“
>>
<<OG:
id: O1
compensator.role: Prüfer
pattern: SendEmail
parameters: [from,Prüfer],
[to,Studienbüro],
[subject,“Note“],
[body,“wurde bekanntgegeben“]
>>

AGENDA

1. Einführung und Motivation
2. BTG-Konzept
3. Rollenverteilung
4. BTG-Annotationen
5. Obligations
6. **BPCC**
7. Weitere Aufgaben
8. Fragebogen

6. BPCC
Einführung

- Werden innerhalb von BTG-Annotationen oder Obligations modelliert
- Modellierung von durch den zugrundeliegenden BP gegebenen Bedingungen
  - `cond.immediate: …` => sequentiell
  - `cond.anytime : …` => parallel
- Durch Einbettung in BTG/OG keine syntaktische Kennzeichnung von Anfang und Ende nötig

- Finden ebenfalls Verwendung in `parameters` der Obligations

6. BPCC
Konzept

- Funktionen über
  - Datenobjekte
  - Aktivitäten
  - Akteure
  - sonstige BPMN-Elemente
- werden mittels logischer Operatoren abgeglichen mit
  - Werten/Mengen
  - anderen Funktionen

6. BPCC
Funktionen

- Funktionen über
  - Datenobjekte, Aktivitäten, Akteure und andere Elemente
- Optional: Angabe einer Zahl für
  - Anzahl der zu betrachtenden Datenobjekte
  - Anzahl der zu betrachtenden Ausführungen der Aktivität
  - Anzahl der zu betrachtenden Akteure
- Schachtelungen von Funktionen sind zulässig, wenn der Rückgabewert der inneren Funktion zum Argument der äußeren passt

`owner($Student.notenauszug)`	Betroffener des Notenauszugs
`end-time ($lesezugriff,3)`	Ende der Ausführung von insgesamt 3 Lesezugriffen

6. BPCC
Einschub

- BPCC benötigen immer eine Funktion (Vorgabe)
- Außerdem benötigt man oftmals Informationen über einen konkreten Akteur, nicht jedoch über die Rolle im Allgemeinen (z.B. Notenschnitt dieses bestimmten Studenten)
- Ein Konstrukt wie `Accessor.Notenschnitt` gibt es nicht
- Daher sind oftmals Funktionen zum Zugriff auf Personen und Rollen nötig

- daher beispielsweise:
  - `performer(Patient behandeln).Gehalt`

    statt
  - `Arzt.Gehalt`

6. BPCC
Funktionen (2)

Funktionsname	Argumenttyp	Rückgabewert
`data-user`	Datenobjekt{, Recht}	Akteure, die auf dieses Datenobjekt zugreifen
`owner`	Datenobjekt	Betroffener, dessen persönliche Daten im Datenobjekt stehen
`performer`	Aktivität	Akteur, der diese Aufgabe (Aktivität) ausführt
`start-time`	Datenobjekt{, Recht}	Zeitpunkt, zu dem Zugriff auf dieses Datenobjekt beginnt
	Aktivität	Zeitpunkt, zu dem Ausführung dieser Aktivität beginnt
`end-time`	Datenobjekt{, Recht}	Zeitpunkt, zu dem Zugriff auf dieses Datenobjekt beendet ist
	Aktivität	Zeitpunkt, zu dem Ausführung dieser Aktivität beendet ist
`data-object`	Aktivität{, Recht}	Alle Datenobjekte, auf die diese Aktivität zugreift
`tasks`	Rolle	Aktivitäten, die von dieser Rolle bereits ausgeführt wurden
	Akteur	Aktivitäten, die von diesem Akteur bereits ausgeführt wurden

6. BPCC Funktionen (3)

Funktionsname	Argumenttyp	Rückgabewert
duration	Aktivität	Dauer der Durchführung einer Aktivität
frequency	Datenobjekt{, Recht}	Häufigkeit der Nutzungen des Datenobjekts
	{, Group}	zusätzlich: Gruppierung der betrachteten Aktivitäten
fulfilled	Gateway, Bedingung	erfüllt, falls die genannte Bedingung bei Gateway „true“ war
	Ereignis	erfüllt, falls ein Ereignis eingetreten ist
	Nachricht	erfüllt, falls Nachrichtenfluss ankommt
executed	list(Aktivität)	erfüllt, falls Aktivitäten ausgeführt wurden
owned-objects	Akteur	Datenobjekte, die persönliche Daten des Betroffenen enthalten
used-objects	Akteur{, Recht}	Datenobjekte, auf die der Akteur zugegriffen hat
role	Aktivität	Rolle, die die Aktivität ausführt
delay	Typ, Zeiteinheit, Dauer	zeitlicher Verzug bis zum Start

6. BPCC
Einfache Operatoren

- Abgleich von Funktionsrückgabe-werten
- Abgleich mit einem Objekt/Wert oder einer Menge davon

Operator	Bedeutung
>	Größer
<	Kleiner
>=	Größer oder gleich
<=	Kleiner oder gleich
==	Gleichheit
≠	Ungleichheit
∈	Element der Menge
∉	Kein Element der Menge

- Beispiele:

- `owner($student.notenauszug) == Diplomer`
- `owner($student.notenauszug) ∈ Studenten`

- `start-time (lesen).hours > 50`
- `start-time (lesen) ∉ „abends“`

6. BPCC
Komplexe Operatoren

Operator	Bedeutung
∧	UND
∨	ODER
==	Gleichheit
≠	Ungleichheit

- Logische Verknüpfung von zwei Funktionen bzw. Funktionen mit Funktionsoperatoren

  ODER

- Vergleich der Rückgabewerte zweier Funktionen

- Beispiele:

- `owner($student.notenauszug) == Diplomer ∧ start-time (lesen) > 50`

- `owner($student.notenauszug) == performer (lesen)`

6. BPCC
Einbettung

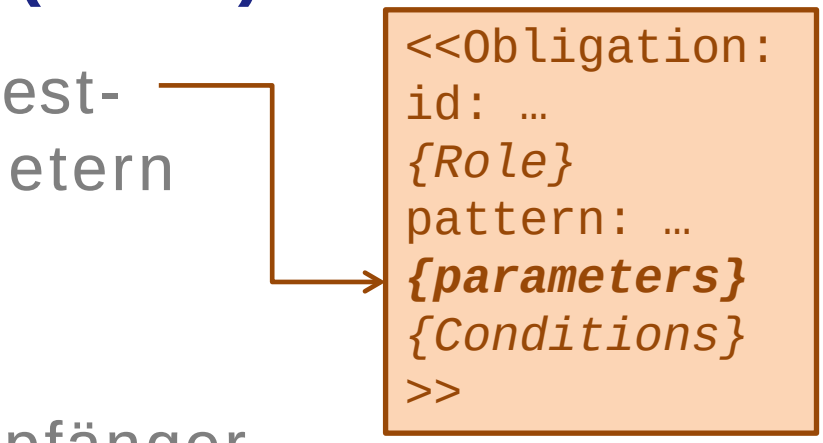
- Nutzung in BTG/OG
- Bedingungen an die Ausführung
  - sequentiell: `cond.immediate: {BPCC}`
  - parallel: `cond.anytime: {BPCC}`

```
<<BTG:
… … …
cond.immediate: owner($student.notenauszug) == Diplomer
                ∧ end-time (lesen) < start-time (schreiben)
cond.anytime:   frequency($student.notenauszug,read) > 3
… … …
>>
```

- *einmalig: Es handelt sich um den Notenauszug des Diplomers*
- *einmalig: „Lesen“ muss vor Beginn von „Schreiben“ abgeschlossen sein*
- *Falls beides erfüllt: So lange warten, bis der Notenauszug mindestens 3 Mal gelesen wurde*

6. BPCC
Verwendung in Obligations (Wdh.)

- Verwendung von BPCC zur Festlegung der Werte von Parametern

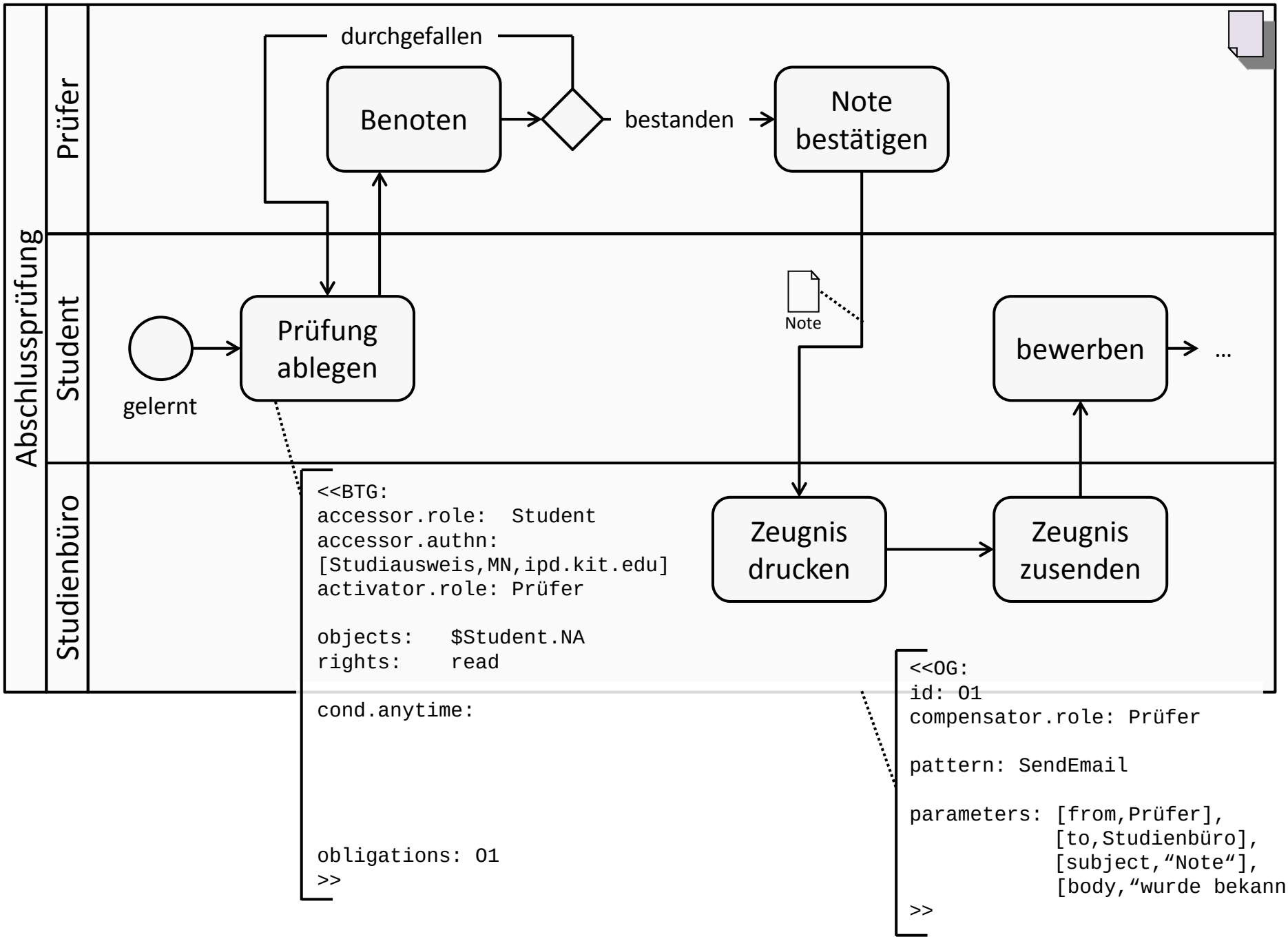


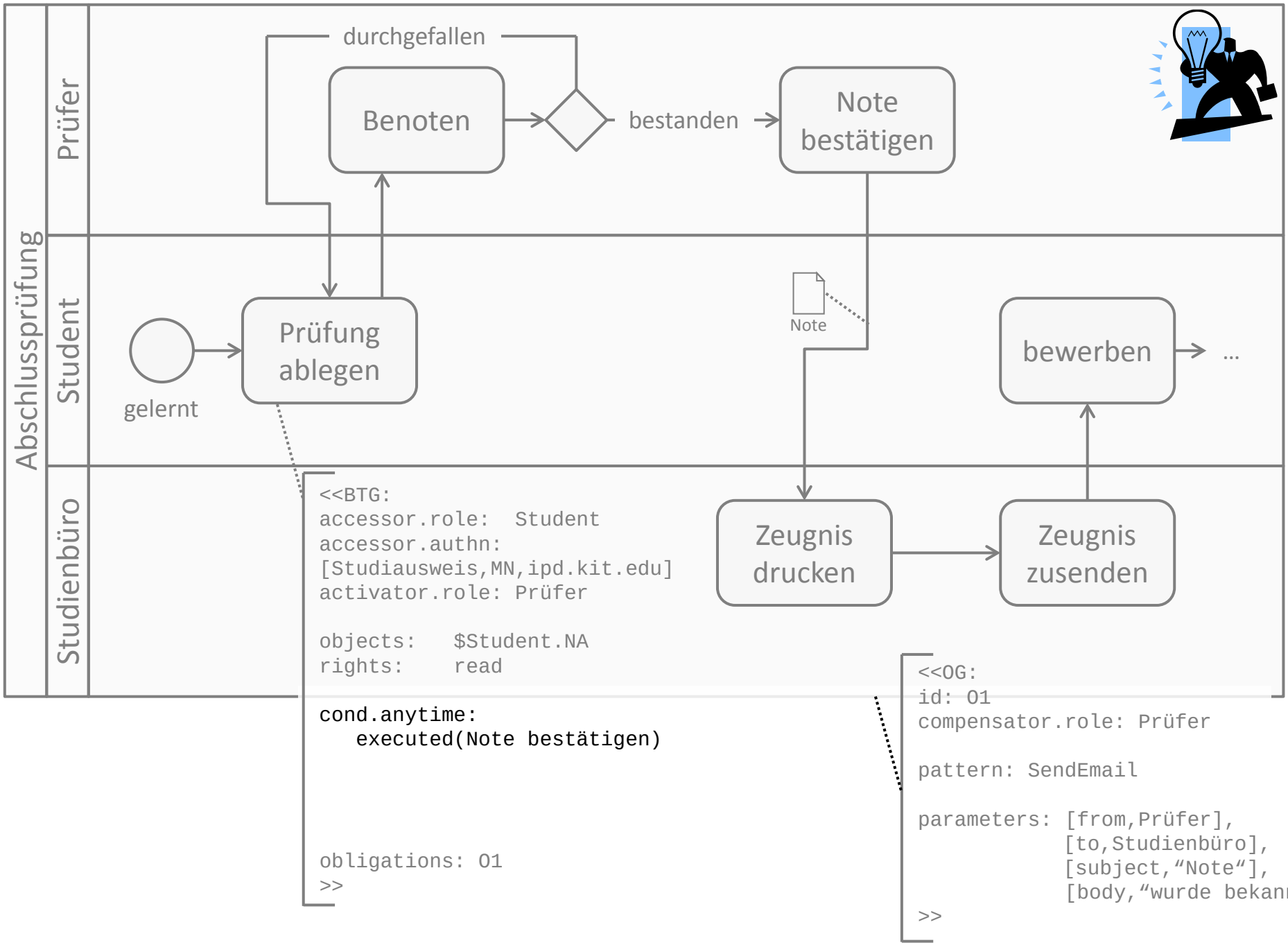
- Es sei beispielsweise der Empfänger-Parameter des „SendEmail“-Patterns gegeben
- Empfänger könnte nun die Person sein, die eine bestimmte Aufgabe ausgeführt hat
- Die Referenzierung dieser Person wäre hier über einen BPCC möglich
  - **[to,performer(lesen)]**

Aufgabe 6.1
Simpler BPCC

- Gegeben sei die BTG-Annotation aus Aufgabe 4. Modellieren Sie nun die dort fehlenden Rahmen-bedingungen zur Ausführung
- Verwenden Sie dabei BPCC

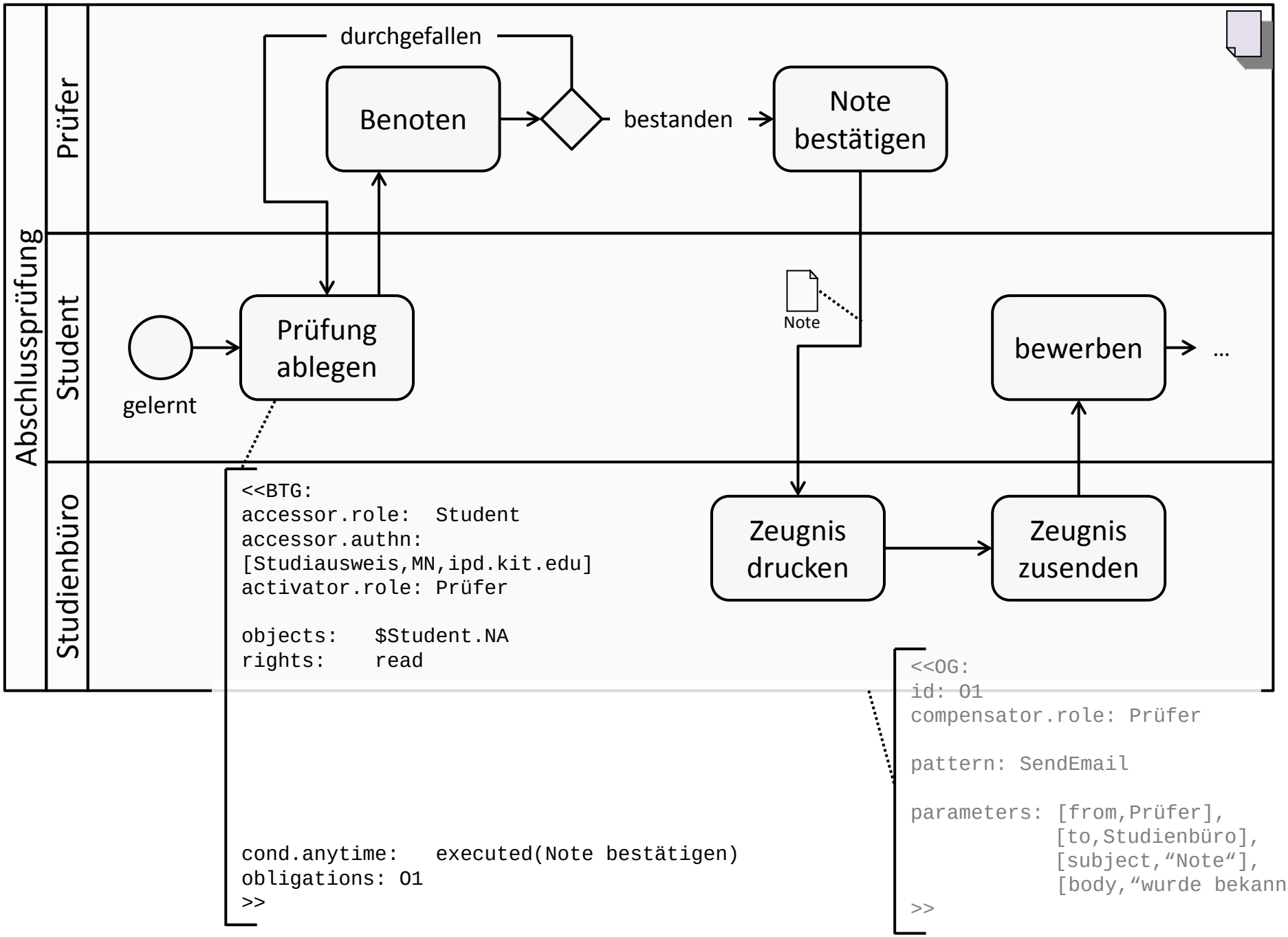
- **Beginnen Sie mit der Modellierung folgender Bedingung:**
  - ✓ Der Student erhält sein Vorab-Zeugnis erst (sprich das Glas darf erst dann gebrochen werden), wenn die Note bestätigt wurde

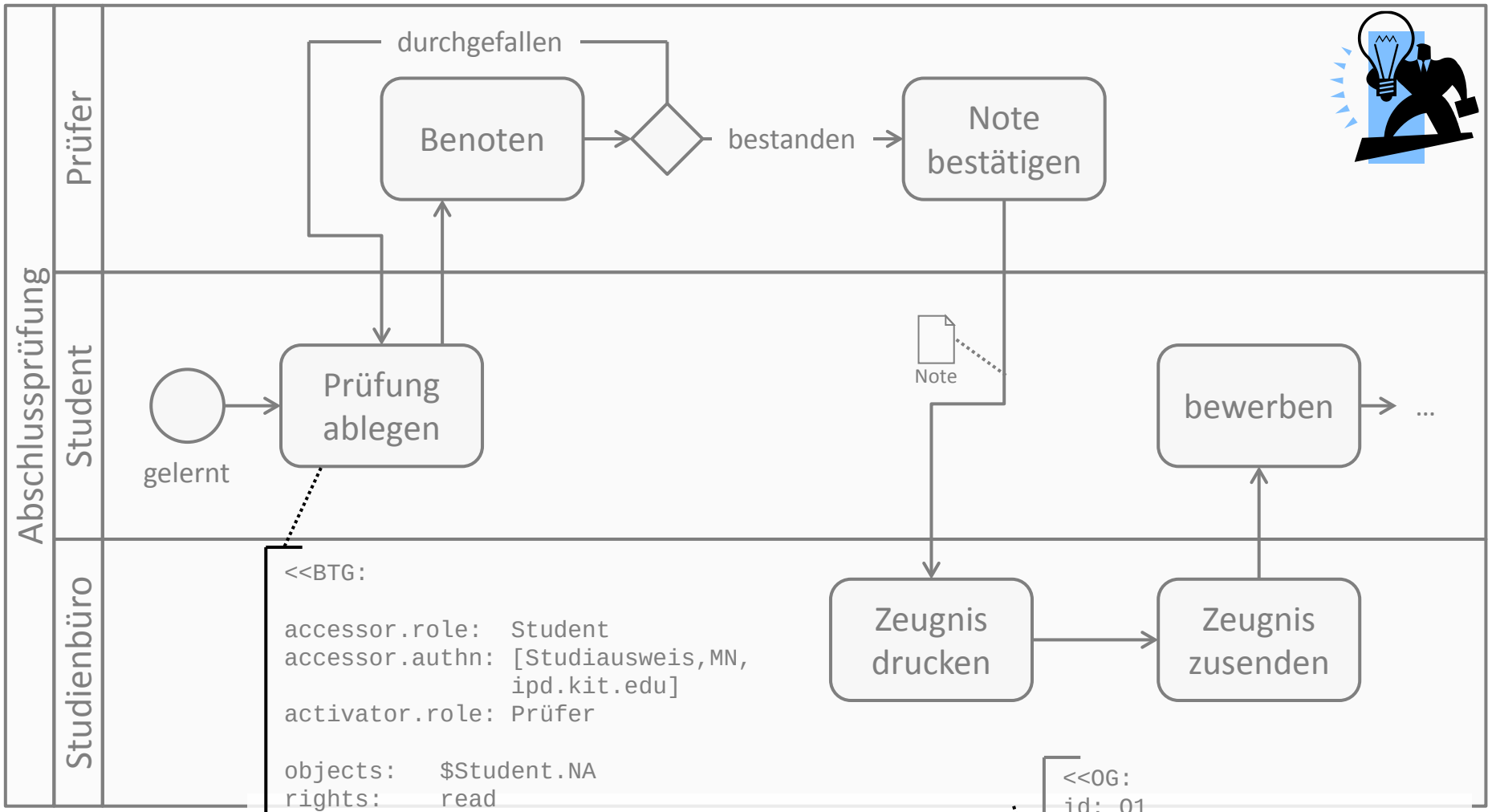

Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn:
[Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.anytime:
obligations: O1
>>
<<OG:
id: O1
compensator.role: Prüfer
pattern: SendEmail
parameters: [from,Prüfer],
[to,Studienbüro],
[subject,“Note“],
[body,“wurde bekannt
>>

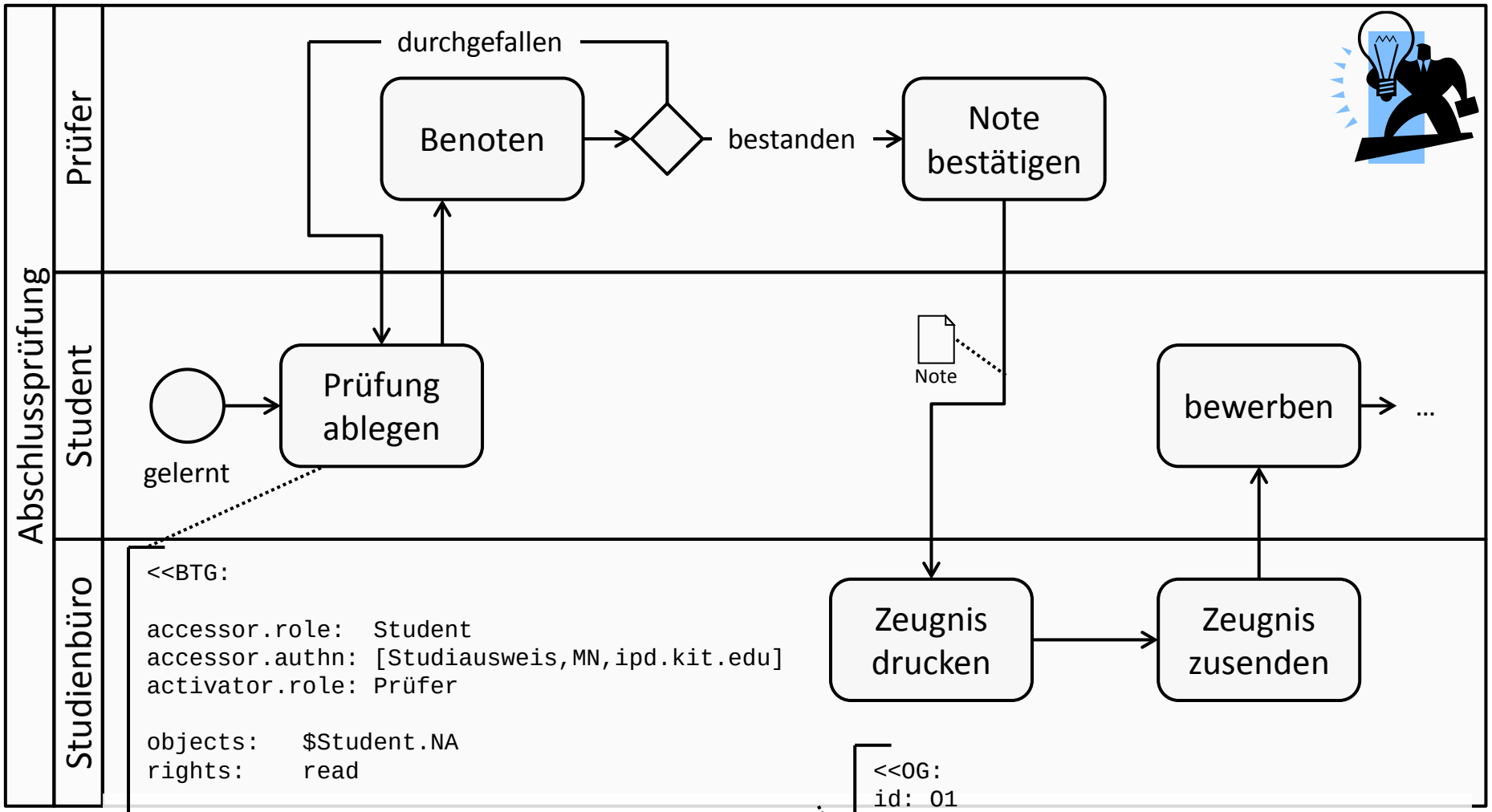

Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn:
[Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.anytime:
executed(Note bestätigen)
obligations: O1
>>
<<OG:
id: O1
compensator.role: Prüfer
pattern: SendEmail
parameters: [from,Prüfer],
[to,Studienbüro],
[subject,“Note“],
[body,“wurde bekann
>>

Aufgabe 6.2
Fortgeschrittener BPCC

- **Neuen Regelungen des Prüfungsausschusses zufolge darf ein Vorab-Zeugnis nur dann ausgestellt werden, wenn**
  - ✓ Der Student mindestens die Note 2,3 hat (die Note ist hier ein Attribut des Studenten)
  - ✓ Der Prüfer Professor ist
- **Erweitern Sie den BPCC aus Aufgabe 6.1 entsprechend**


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn:
[Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.anytime: executed(Note bestätigen)
obligations: O1
>>
<<OG:
id: O1
compensator.role: Prüfer
pattern: SendEmail
parameters: [from,Prüfer],
[to,Studienbüro],
[subject,“Note“],
[body,“wurde bekann
>>


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Note
Prüfung ablegen
gelernt
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn: [Studiausweis,MN,
ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.immediate: performer(Prüfung ablegen).
Note <= 2.3 ∧
role(Note bestätigen) ==
Professor
cond.anytime: executed(Note bestätigen)
obligations: O1
>>
<<OG:
id: O1
compensator.role: Prüfer
pattern: SendEmail
parameters: [from,Prüfer],
[to,Studienbüro],
[subject,“Note“],
[body,“wurde bekann
>>


Abschlussprüfung
Prüfer
Student
Studienbüro
durchgefallen
Benoten
bestanden
Note bestätigen
Prüfung ablegen
gelernt
Note
bewerben
...
Zeugnis drucken
Zeugnis zusenden
<<BTG:
accessor.role: Student
accessor.authn: [Studiausweis,MN,ipd.kit.edu]
activator.role: Prüfer
objects: $Student.NA
rights: read
cond.immediate: performer(Prüfung ablegen).
Note <= 2.3 ∧
role(Note bestätigen) ==
Professor
cond.anytime: executed(Note bestätigen)
obligations: O1
>>
<<OG:
id: O1
compensator.role: Prüfer
pattern: SendEmail
parameters: [from,Prüfer],
[to,Studienbüro],
[subject,“Note“],
[body,“wurde bekanntgegeben“]
>>

AGENDA

1. Einführung und Motivation
2. BTG-Konzept
3. Rollenverteilung
4. BTG-Annotationen
5. Obligations
6. BPCC
7. **Weitere Aufgaben**
8. Fragebogen

7. Weitere Aufgaben
Einschub: Nötige Zusätze

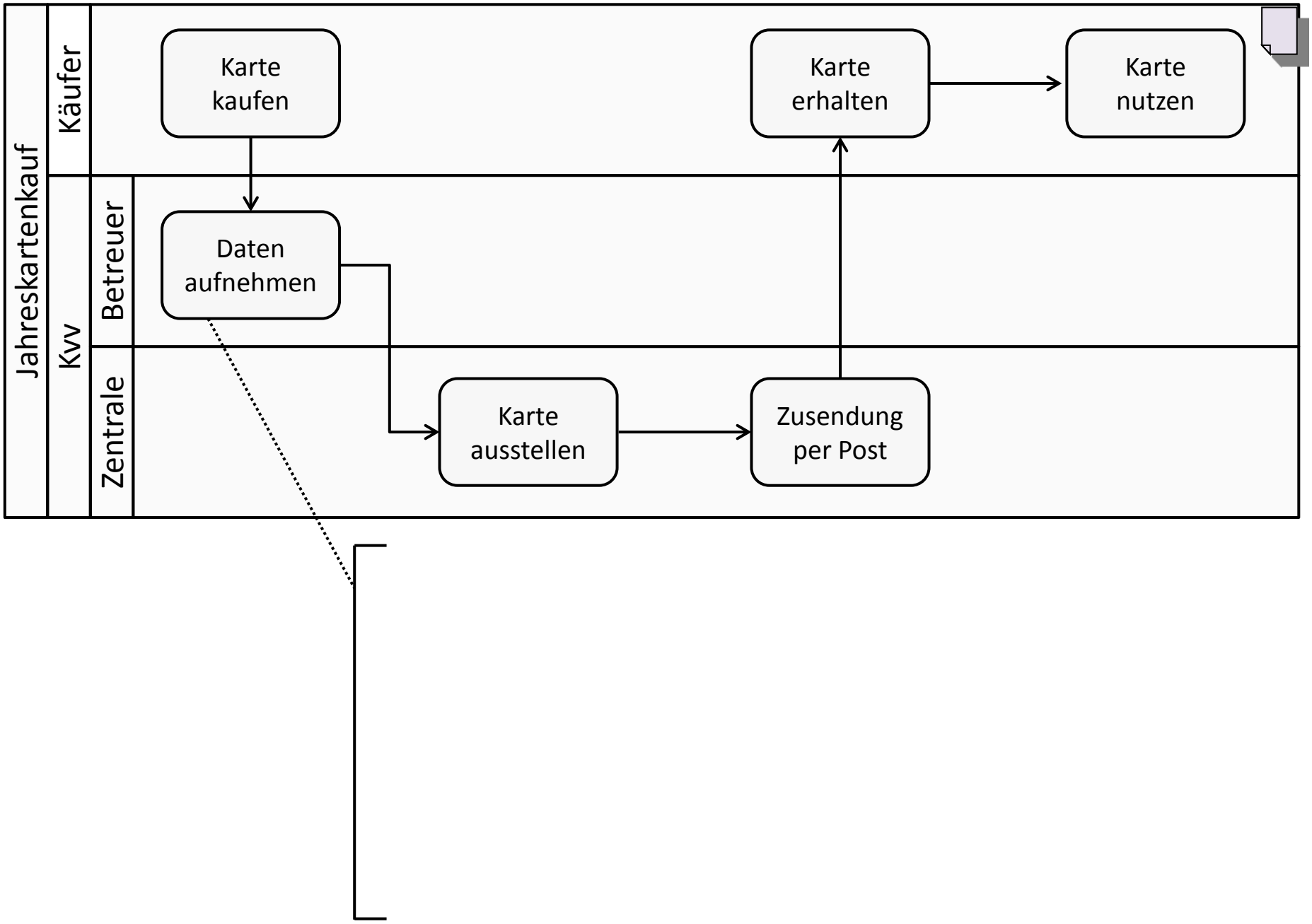
- Auf einzelne Attribute eines Zeitwerts kann mittels „years“, „months“, „days“, „hours“, „minutes“, „seconds“ zugegriffen werden
  - `duration(AufgabeX).hours`

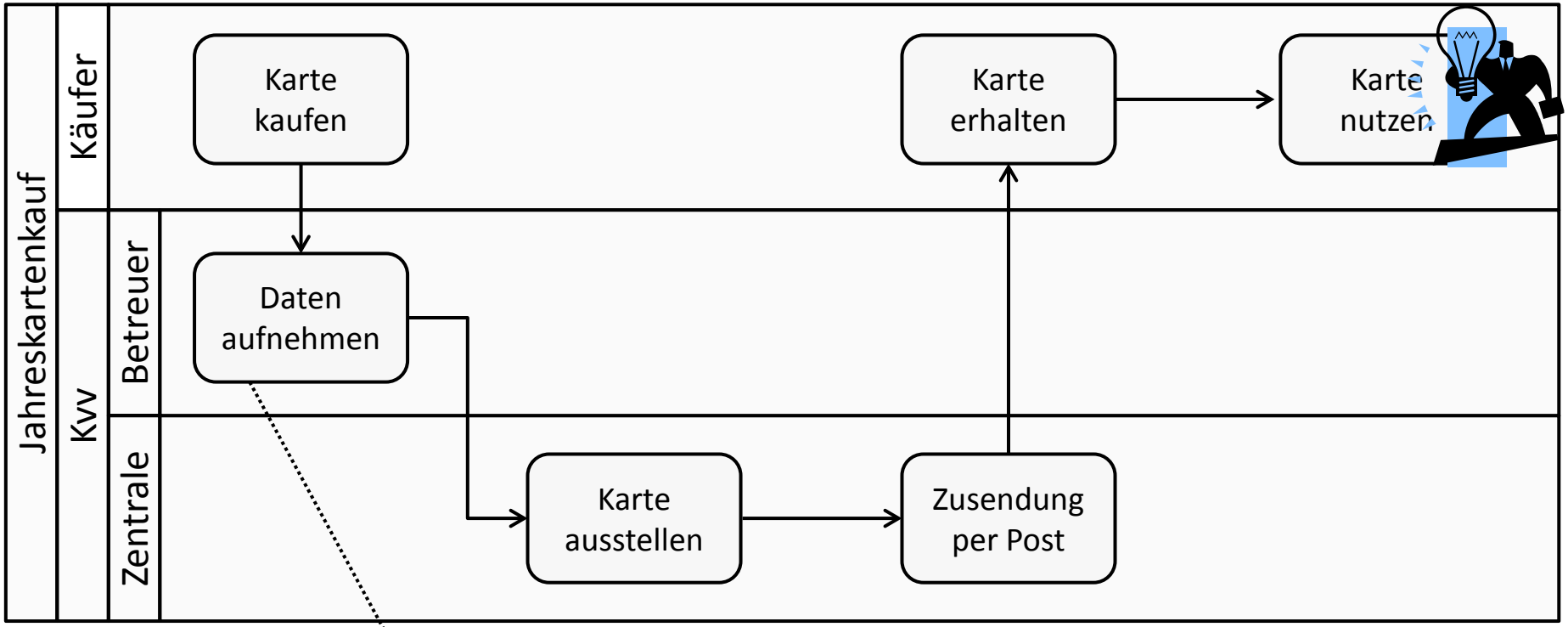
Aufgabe 7.1
KVV-Jahreskarte

- Ein Kunde kauft eine KVV-Jahreskarte am Schalter beim KVV-Kundenbetreuer
- Im Normalfall wird diese nach einigen Tagen von der KVV-Zentrale ausgestellt und dem Kunden per Post zugesandt
- Unter gewissen Umständen besteht allerdings die Möglichkeit, eine Vorab-Karte direkt am Schalter zu erhalten, wenn…
  - der Kauf an einem der ersten 5 Tage des Monats stattfindet
  - der Kunde erwachsen ist
- Zum Ausstellen der Vorab-Karte muss der Betreuer des KVV auf die Datenbank der Kundendaten zugreifen, auf die ihm sonst der Zugriff verwehrt bliebe (im Normalfall hat nur die Zentrale die entsprechenden Zugriffsrechte)

❖ **Modellieren Sie diese Ausnahme mittels BTG**

- ✓ Accessor ist der Kundenbetreuer des KVV
- ✓ Dieser authentifiziert sich mittels seines Accounts und des zugehörigen Passworts; abgeglichen wird dieses mit den Daten in ipd.kvv.de
- ✓ Es ist kein Activator nötig
- ✓ Der Zugriff auf die Kundendaten des Käufers erfolgt lesend und schreibend
- ✓ einmalig zu prüfende Bedingungen: Kauf findet an einem der ersten 5 Tage des Monats statt; Kunde ist erwachsen
- ✓ Verpflichtungen (Obligations) entstehen nicht


Jahreskartenkauf
Käufer
Kvv
Betreuer
Zentrale
Karte kaufen
Daten aufnehmen
Karte ausstellen
Zusendung per Post
Karte erhalten
Karte nutzen




```
<<BTG:

accessor.role:  Betreuer
accessor.authn: [account,password,idp.kvv.de]

objects: $Käufer.Kundendaten
rights:  read, write

cond.immediate: start-time(Daten aufnehmen).days <= 5
                ∧ performer(Karte kaufen).age >= 18

>>
```

Aufgabe 7.2
Bibliothekszenario: Verliehenes Buch

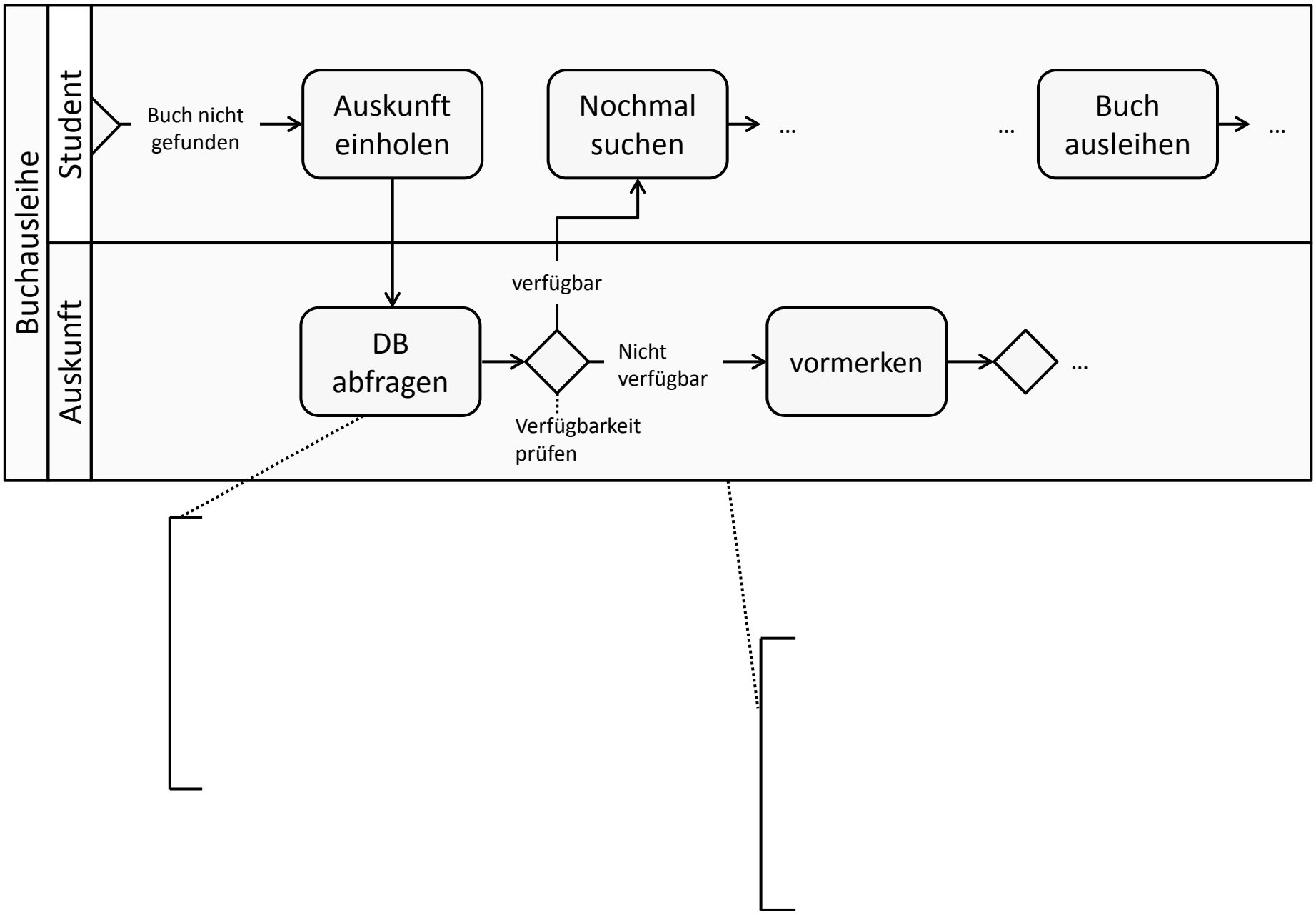
- Student: Suche nach „Buch“
  - Buch ist bereits verliehen (an irgendwen; für Student unbekannt)
- Student möchte Kontaktdaten des Ausleihers wissen, um diesen zu kontaktieren
  - Bücherei-Auskunft kennt diese Kontaktdaten
  - Der Ausleiher muss der Herausgabe seiner Daten zustimmen
- Im Nachgang muss die Bücherei-Auskunft eine E-Mail an den „Leiter“ der Bücherei schicken
  - Dies ist nur zu tun, falls der Ausleiher die Rolle Student innehat

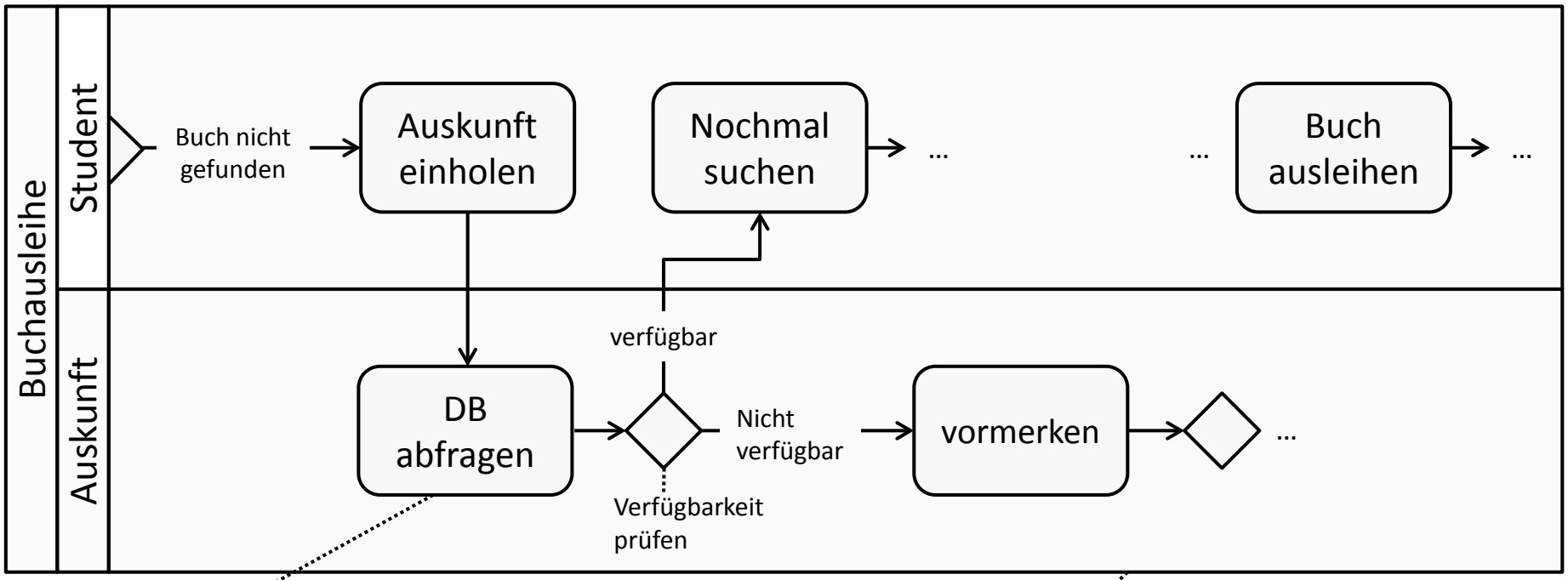
❖ **Modellieren Sie diese Situation mittels BTG**

- ✓ Accessor: Student (Auszuweisen mittels Matrikelnummer MN; Überprüfung durch idp.kit.edu)
- ✓ Activator: Ausleiher (keine Authentifizierung nötig)
- ✓ Leserechte für die Kontaktdaten des Ausleihers
- ✓ BTG sollte natürlich nur dann Anwendung finden, wenn wirklich kein Buch mehr in der Bücherei vorrätig ist

❖ **Modellieren Sie auch eine entsprechende Obligation**

- ✓ Compensator: Die Auskunft
- ✓ Vergessen Sie die zusätzliche Bedingung nicht


Buchausleihe
Student
Auskunft
Buch nicht gefunden
Auskunft einholen
Nochmal suchen
...
...
Buch ausleihen
...
verfügbar
DB abfragen
Nicht verfügbar
vormerken
...
Verfügbarkeit prüfen



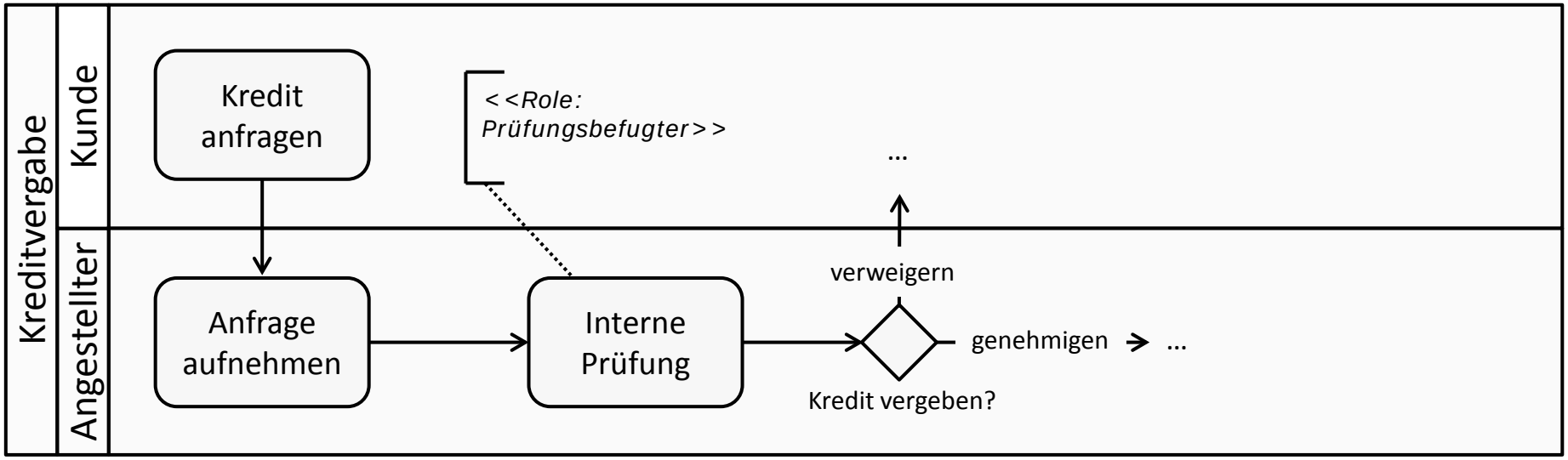

```
<<BTG:
accessor.role:  Student
accessor.authn: [Studiausweis,MN,idp.kit.edu]
activator.role: Ausleiher

objects: $Ausleiher.Kontaktdaten
rights: read

cond.immediate: fulfilled(Verfügbarkeit
prüfen, Nicht verfügbar)

obligations: O1
>>
```

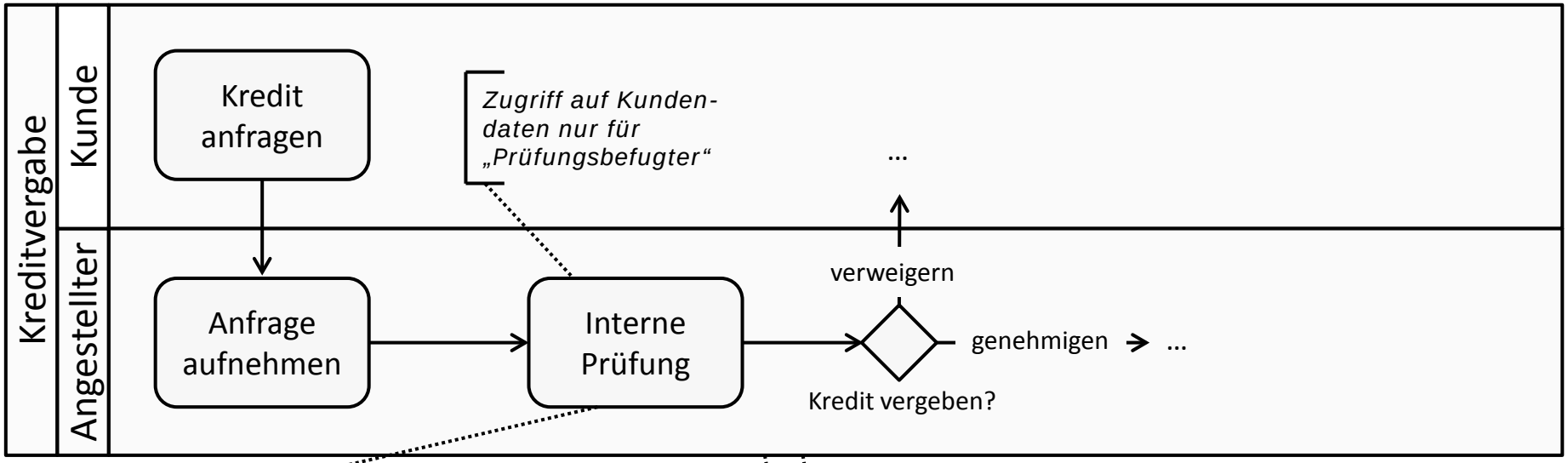
```
<<OG:
id: O1
compensator.role: Auskunft
pattern: SendEMail
parameters: [from,Auskunft],
     [to,Leiter], [subject,“Daten“],
     [body,“wurden herausgegeben“]
cond.immediate: role(owner(
     $Ausleiher.Kontaktdaten)) == Student
>>
```

Aufgabe 7.3
Kreditvergabe einer Bank

- Ein Schritt im Prozessablauf ist die Aufgabe „Interne Prüfung“
- Diese darf normalerweise nur von Angestellten mit der Rolle „Prüfungsbefugter“ durchgeführt werden, da nur diese auf die Kundendaten zugreifen dürfen
- Dauert die Aufgabe „Interne Prüfung“ mindestens 7 Tage an (sprich es passiert nichts), so dürfen auch normale Bankangestellte die „Interne Prüfung“ durchführen und dabei vor allem auf die Kundendaten zugreifen
- Als Konsequenz muss eine E-Mail an den Abteilungsleiter (Rolle: „Abteilungsleiter“) geschickt werden; diese Ausgabe muss nicht von einem Menschen ausgeführt werden
  - Im Text der E-Mail muss der Mitarbeiter stehen, der die Prüfung durchgeführt hat
- Außerdem muss der Abteilungsleiter die Entscheidung zur Kreditvergabe, die vom normalen Angestellten durchgeführt wurde, im Nachgang überprüfen

- **Modellieren Sie eine entsprechende BTG-Annotation für das vorliegende Beispiel**
- **Modellieren Sie das Versenden der E-Mail an den Abteilungsleiter**
- **Die Überprüfung durch den Abteilungsleiter lässt sich mit dem bisher Vorgestellten nicht direkt modellieren**
  - ✓ Beschreiben Sie, wie Sie vorgehen würden
  - ✓ Notieren Sie die entsprechende Obligation gegebenenfalls


Kreditvergabe
Kunde
Angestellter
Kredit anfragen
<<Role: Prüfungsbefugter>>
...
verweigern
Anfrage aufnehmen
Interne Prüfung
genehmigen
...
Kredit vergeben?




```
<<BTG:
accessor.role: Angestellter
objects: $Kunde.Daten
rights:  read

cond.anytime: duration(Interne
              Prüfung).days >= 7
obligations:  O1,O2
>>
```

```
<<OG:
id: O1
pattern: SendEMail
parameters: [to,Abteilungsleiter],
[subject,“$Kreditantrag“], [body,“geprüft von
performer(Interne Prüfung)“]
>>
```

```
<<OG:
id: O2
compensator.role: Abteilungsleiter
pattern: EntscheidungPrüfen
parameters: [gateway,Kredit vergeben], [when,
14 days]
>>
```

Für die Prüfung muss ein neues Obligation-Pattern erstellt werden. Dieses wird vom Prozessmodellierer als Prozessfragment in BPMN ausmodelliert und für die weitere Verwendung als Obligation definiert.
Beispielhaft hier: Fragment zur Überprüfung der Entscheidung an einem per Parameter „what“ spezifizierten Gateway binnen des per Parameter „when“ definierten Zeitraums.

AGENDA

1. Einführung und Motivation
2. BTG-Konzept
3. Rollenverteilung
4. BTG-Annotationen
5. Obligations
6. BPCC
7. Weitere Aufgaben
8. **Fragebogen**

8. Fragebogen

- Danke für die Teilnahme an unserer Nutzerstudie!

- Damit wir die BTG-Annotationssprache für Business Processes noch weiter verbessern können möchten wir Sie bitten, zum Abschluss noch einen kurzen Fragebogen auszufüllen
  - Was hat Ihnen gut gefallen?
  - Was hat Sie gestört?
  - Auf welche Herausforderungen sind Sie gestoßen?
  - Haben Sie konkrete Verbesserungsvorschläge?

8. Fragebogen

- Bitte antworten Sie ehrlich und Ihrem Gefühl folgend!
- Beschönigungen helfen uns nicht bei der Fehlerfindung

  ...
- Die krampfhafte Suche nach Fehlern allerdings ebensowenig
- Teilen Sie uns einfach mit, was Sie denken!

8. Fragebogen

- Bitte handschriftlich ausfüllen

- Lieber zu viel als zu wenig in die Freitextanmerkungen schreiben ☺

http://btg.grabatin.com/index.php/826811/lang-de

bzw.

http://goo.gl/2qy65