Kurs 1794 „Methodische Entwicklung von Benutzungsschnittstellen für Webapplikationen“

Musterlösungen zur Klausur am 4.3.2006

Aufgabe 1 *Sessions (2 + 3 +1 = 6 Punkte)*

a) *(2 Punkte)*

Welches Problem lösen Sessions?

HTTP ist zustandslos, (Web-)Applikationen sind es in der Regel nicht. Der Zustand einer Webapplikation bezieht sich im Allgemeinen auf den jeweiligen Benutzer. Zur Lösung dieses Problems wird das Konzept der Sitzung (Session) verwendet. Eine Session ist eine logische, dauerhafte Verbindung zwischen Client und Server. Um die Session zu identifizieren, wird die Session-ID verwendet.

b) *(3 Punkte)*

Wie werden Session-IDs in Requests eingebunden? Nennen Sie die technischen Lösungen und erläutern Sie in einem Satz deren Vor- und Nachteile.

Es gibt drei Möglichkeiten, um die Session-ID in jede Anfrage einzubinden: Verwendung von versteckten Formularfeldern, URL-Rewriting und Cookies.
Versteckte Formularfelder sind einfach zu erstellen, ihre Anwendbarkeit ist allerdings auf Formulare beschränkt. URL-Rewriting kann immer verwendet werden, ist aber technisch aufwendiger, da wirklich jede (applikationsinterne) URL bei der Ausgabe behandelt werden muss. Für den Entwickler sind daher Cookies am einfachsten zu behandeln, allerdings kann der Client Cookies verbieten, so dass man sich nicht auf diese Technik verlassen kann.

c) *(1 Punkt)*

Woher kennt der Client die Session-ID?

Die Session-ID wird mit der vorhergehenden Response mitgeliefert. Dies geschieht, in dem entweder ein Cookie beim Client angelegt

Autor: Jens von Pilgrim
Stand: 2006-02-02

wird oder einfach durch Hinzufügen der Session-ID in den HTML-Code.

Aufgabe 2 *Scopes (1 + 4 =5 Punkte)*

a) *(1 Punkt)*
Was sind „Scopes" und wozu werden sie verwendet? (Ein Satz genügt!)

Scopes sind Behälter mit definierter Lebensdauer zum Ablegen temporärer Daten.

b) *(4 Punkte)*
Welche Scopes sind in der Servlet-Spezifikation vorgesehen? Nennen Sie die Scopes und beschreiben Sie sie kurz.

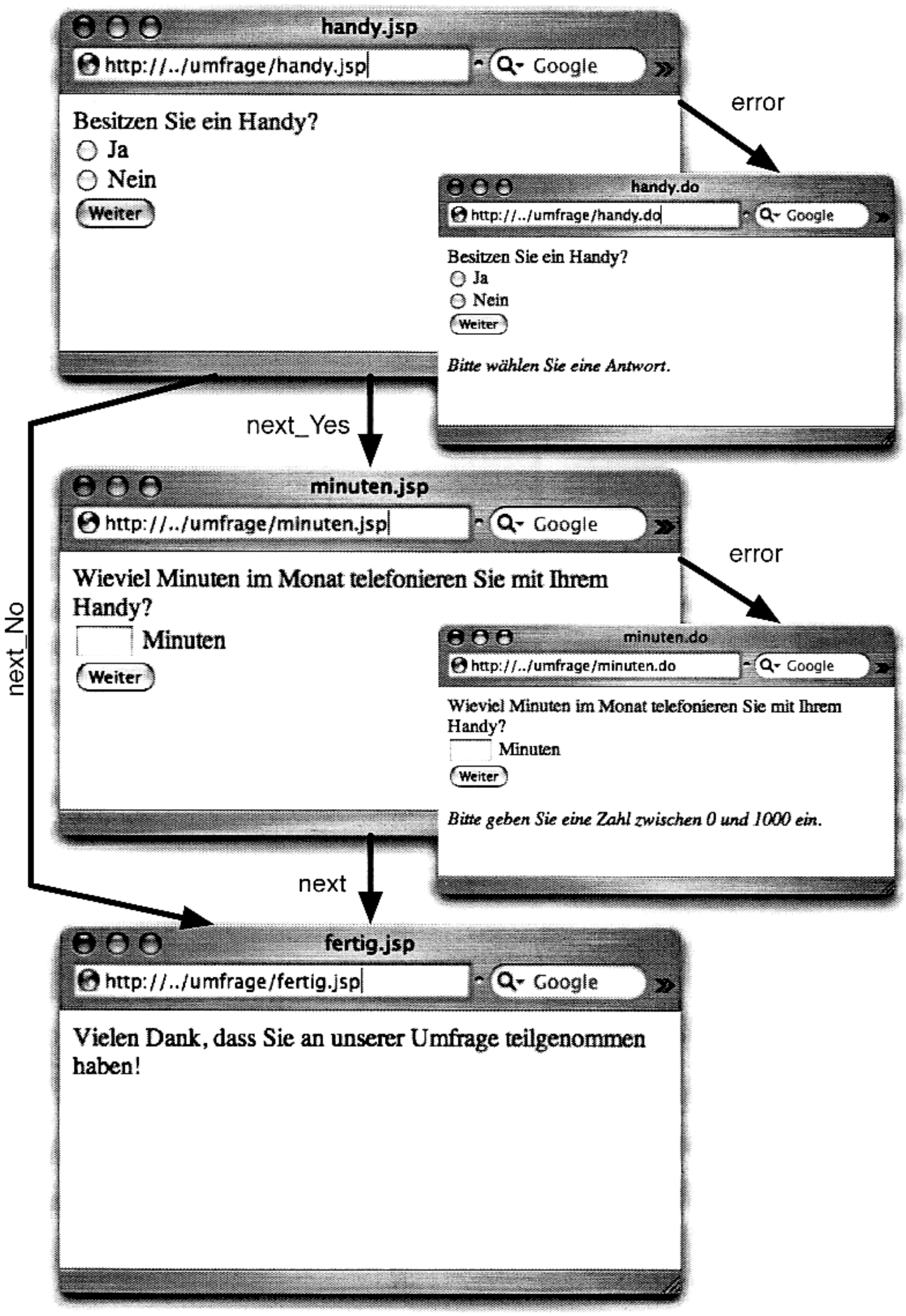
Es sind vier Scopes vorgesehen: Request-, Session-, Application- und Page-Scope. Der Request-Scope ist während eines Request-Durchlaufs aktiv und wird etwa verwendet, um Formulardaten zu übermitteln. Der Session-Scope ist gültig für die Dauer einer Session und speichert nutzerspezifische Informationen, die nur für eine Session von Belang sind oder auf die während einer Session schnell zugegriffen werden können muss. Der Application-Scope ist während der gesamten Laufzeit einer Webapplikation aktiv, hier werden Daten gespeichert, die allen Komponenten der Webapplikation zu Verfügung stehen sollen. Der Page-Scope ist nur für JSP-Seiten relevant und ist nur während der Erzeugung der JSP-Seite aktiv. Er wird verwendet, damit JSP-Tags untereinander Daten austauschen können.

Kurs 1794 „Methodische Entwicklung von Benutzungsschnittstellen für Webapplikationen“

Musterlösungen zur Klausur am 4.3.2006

Aufgabe 3 *Anforderungsorientiertes Aktivitätsdiagramm (5 +5 = 10 Punkte)*

Angegeben sei folgende Webapplikation in Form von Screenshots:

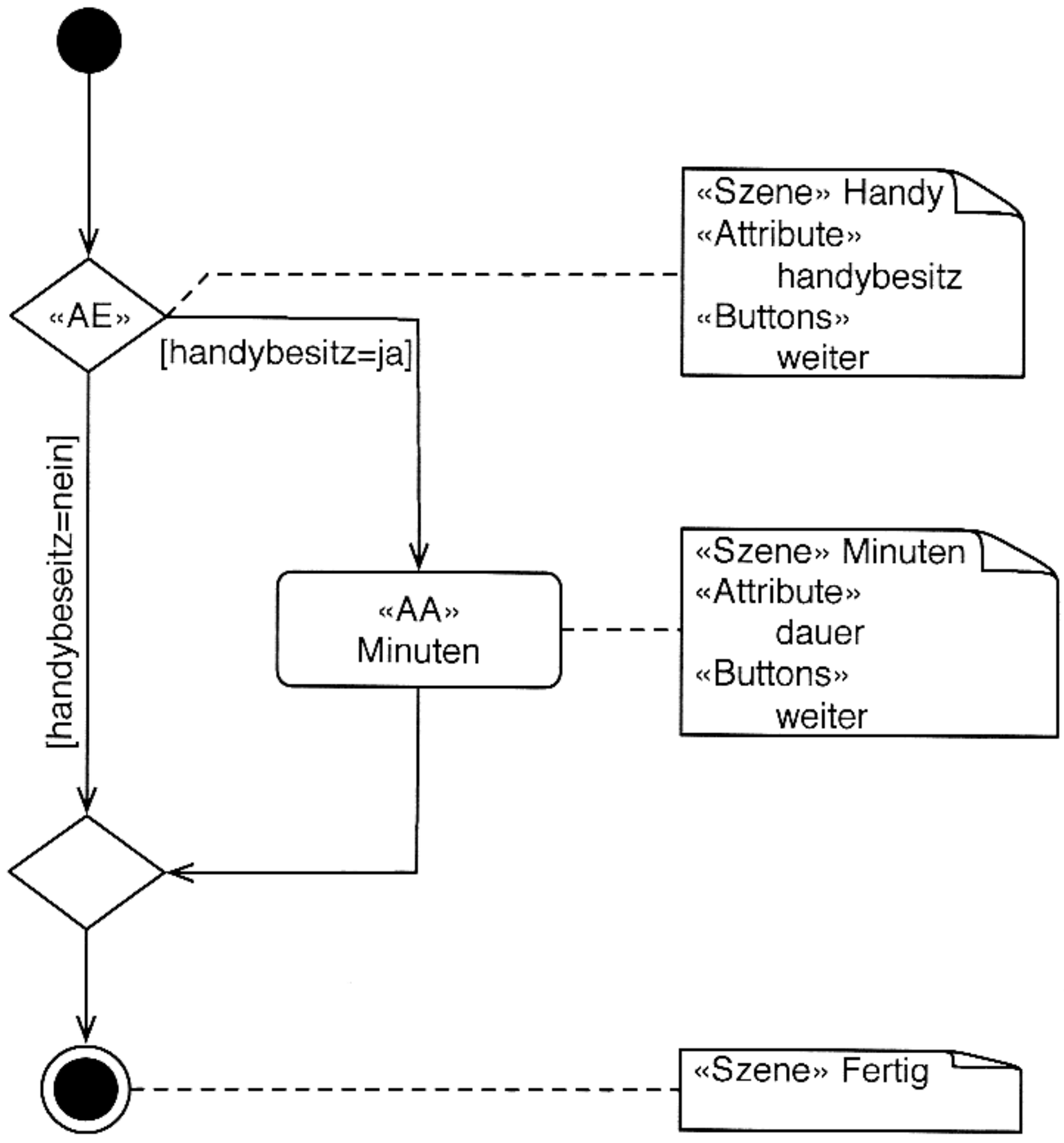


Kurs 1794 – Software Engineering II

Musterlösungen zur Klausur am 4.3.2006

a) *(5 Punkte)*

Erstellen Sie ein anforderungsorientiertes Aktivitätsdiagramm, das obigen Ablauf modelliert. Die Fehlerseiten (verkleinerte Seiten auf der rechten Hälfte) müssen in dieser Teilaufgabe nicht berücksichtigt werden.



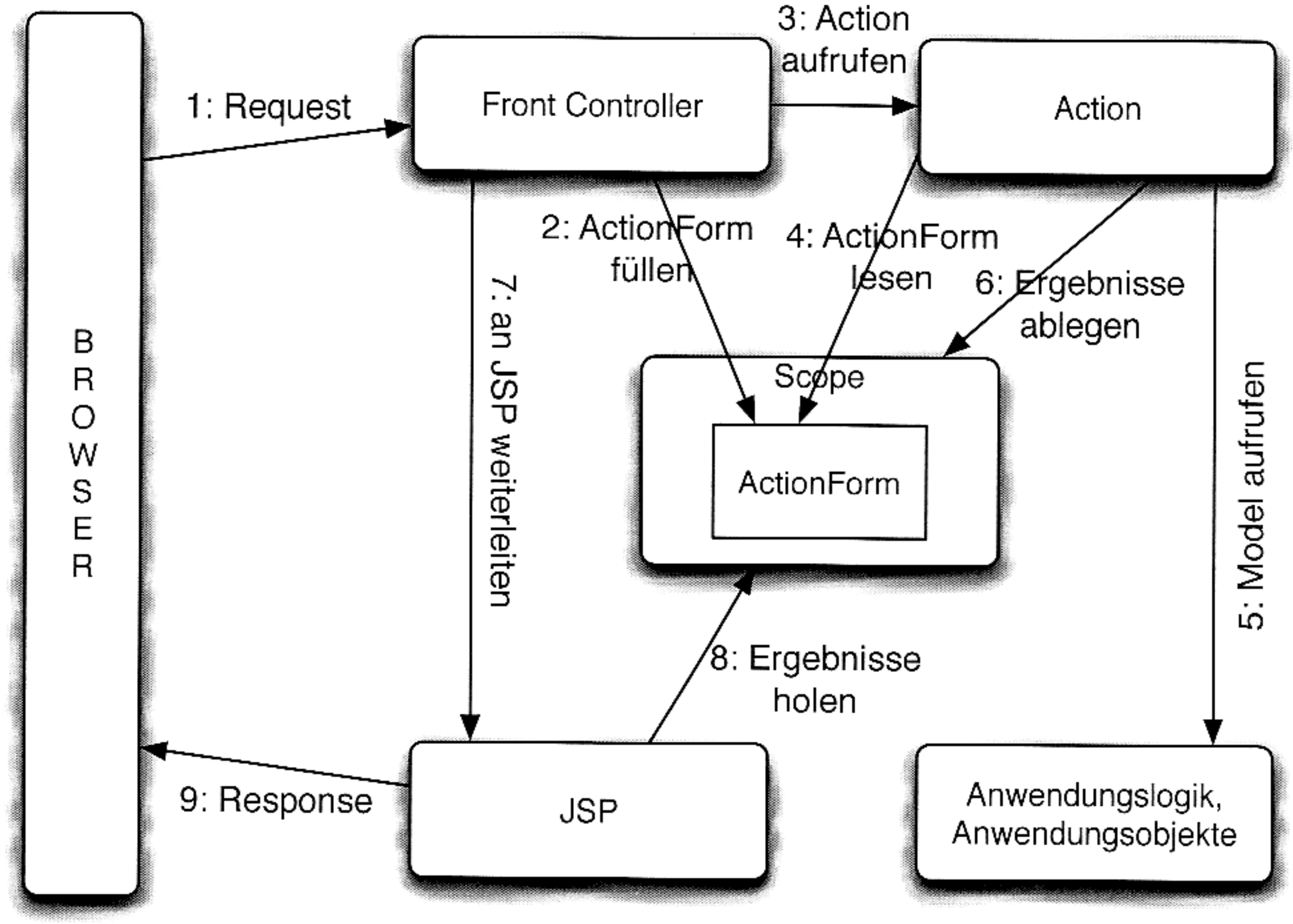
b) *(5 Punkte)*

Erstellen Sie eine Struts-Konfigurationsdatei, die die angegebene Webapplikation umsetzt. Nehmen Sie dabei an, dass alle Eingabedaten mittels ActionsForms behandelt werden. Falls Ihnen Informationen fehlen, setzen Sie selbst gewählte Angaben ein und kommentieren Sie diese Fälle am Ende kurz.

```
<struts-config>

     <form-beans>
          <form-bean name="handyAF" type="HandyAF" />
          <form-bean name="minutenAF" type="MinutenAF" />
     </form-beans>

     <action-mappings>
          <action   path="/handy" type="HandyA"
                    name="HandyAF" validate="true"
                    input="handy.jsp">
               <forward name="next_Yes" path="minuten.jsp" />
               <forward name="next_No" path="fertig.jsp" />
          </action>
          <action   path="/minuten" type="MinuntenA"
                    name="MinutenAF" validate="true"
                    input="minuten.jsp">
               <forward name="next" path="fertig.jsp" />
          </action>
     </action-mappings>

</struts-config>
```

Aufgabe 4 *Request-Ablauf (9 Punkte)*

Beschreiben Sie den typischen Request-Ablauf unter Einsatz von Struts. Vorgefüllte Formularfelder sollen dabei nicht berücksichtigt werden. Zeichnen Sie dabei die jeweiligen Aufrufe in das vorgegebene Diagramm ein.



Beschreibung:

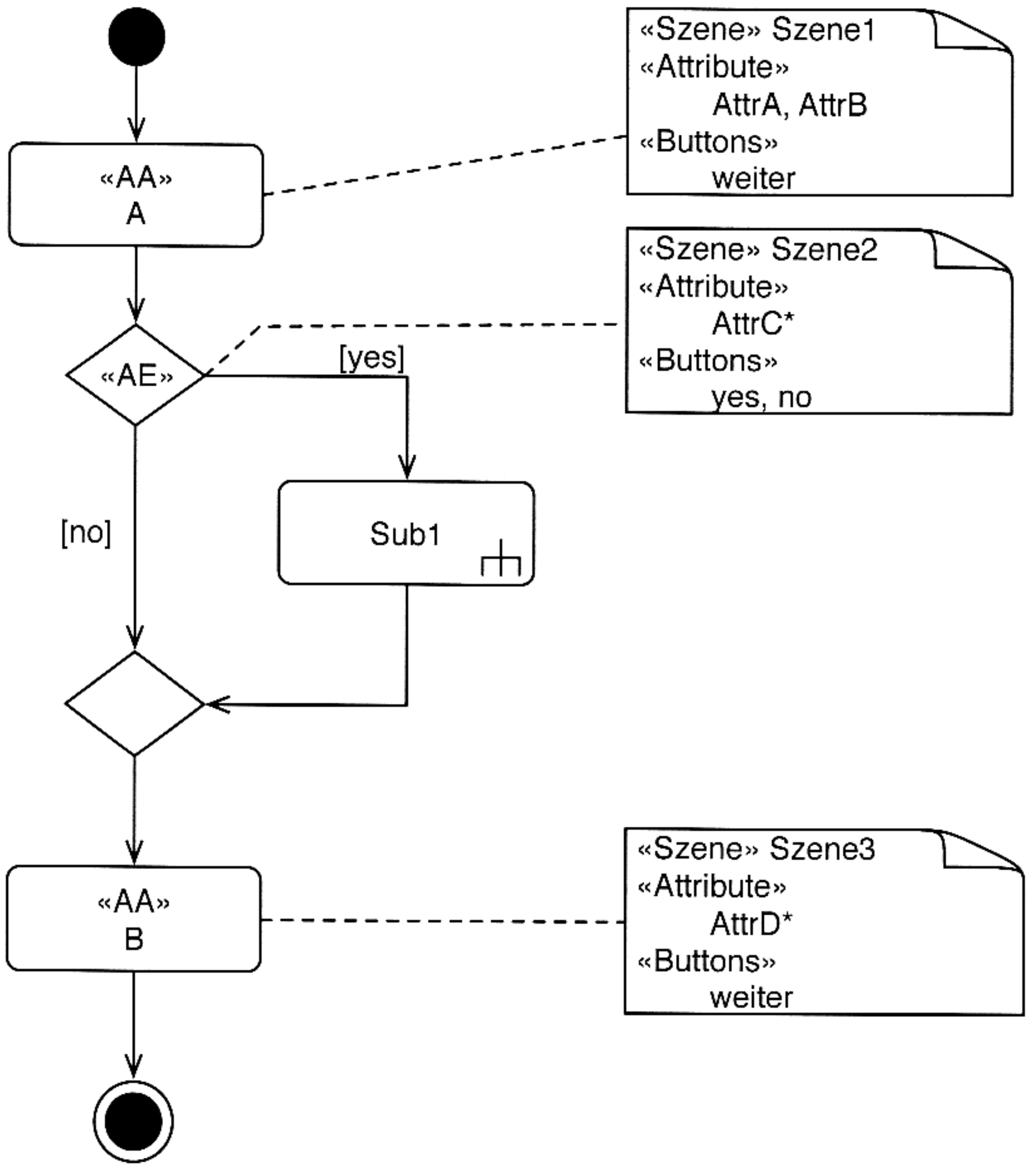
Der eingehende Request erreicht den Front Controller (1). Dieser bestimmt das zugehörige ActionForm und füllt es mit den vom Benutzer gesendeten Daten (2). Das ActionForm führt einfache Eingabeüberprüfungen durch. Verläuft die Überprüfung erfolgreich, bestimmt der Front Controller die nachfolgende Action und ruft sie auf (3). Die Action holt sich die Benutzereingaben aus dem ActionForm (4) und ruft das Model auf (5). Die Action speichert anschließend die vom Model gelieferten Ergebnisse in einem passenden Scope (6). Dann leitet der Front Controller an die passende JSP weiter (7). Zur Darstellung der Ergebnisse greift die JSP auf den Scope zu (8) und übergibt abschließend das erstellte HTML-Dokument als Response an den Browser (9).

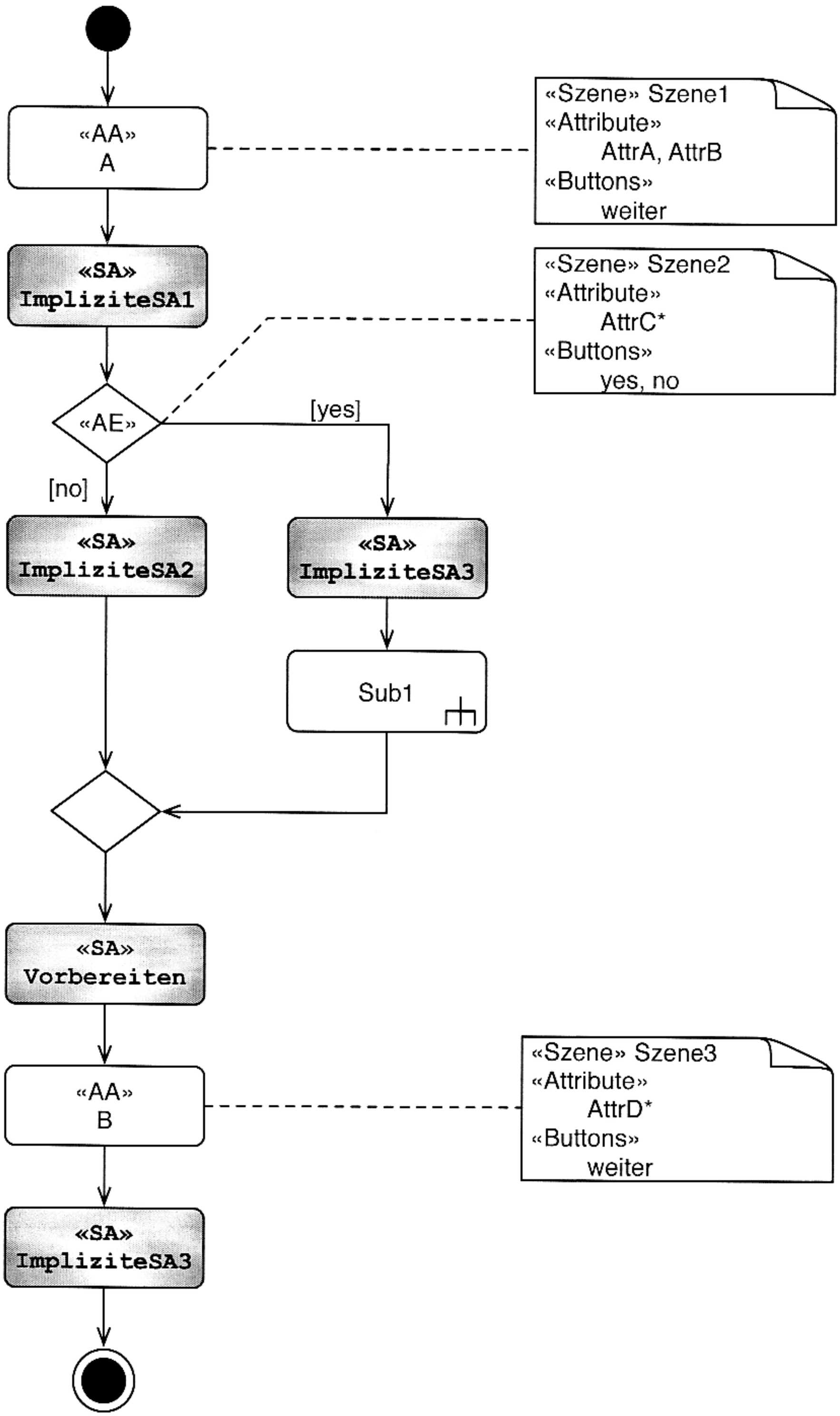
Kurs 1794 „Methodische Entwicklung von Benutzungsschnittstellen für Webapplikationen“

Musterlösungen zur Klausur am 4.3.2006

Aufgabe 5 *Entwurfsorientiertes Aktivitätsdiagramm (6 Punkte)*

Wandeln Sie folgendes Aktivitätsdiagramm in ein entwurfsorientsorientiertes Aktivitätsdiagramm um und erläutern Sie kurz Ihr Vorgehen.




«AA»
A
«Szene» Szene1
«Attribute»
AttrA, AttrB
«Buttons»
weiter
«SA»
ImpliziteSA1
«Szene» Szene2
«Attribute»
AttrC*
«Buttons»
yes, no
«AE»
[yes]
[no]
«SA»
ImpliziteSA2
«SA»
ImpliziteSA3
Sub1
«SA»
Vorbereiten
«AA»
B
«Szene» Szene3
«Attribute»
AttrD*
«Buttons»
weiter
«SA»
ImpliziteSA3

Kurs 1794 „Methodische Entwicklung von Benutzungsschnittstellen für Webapplikationen“

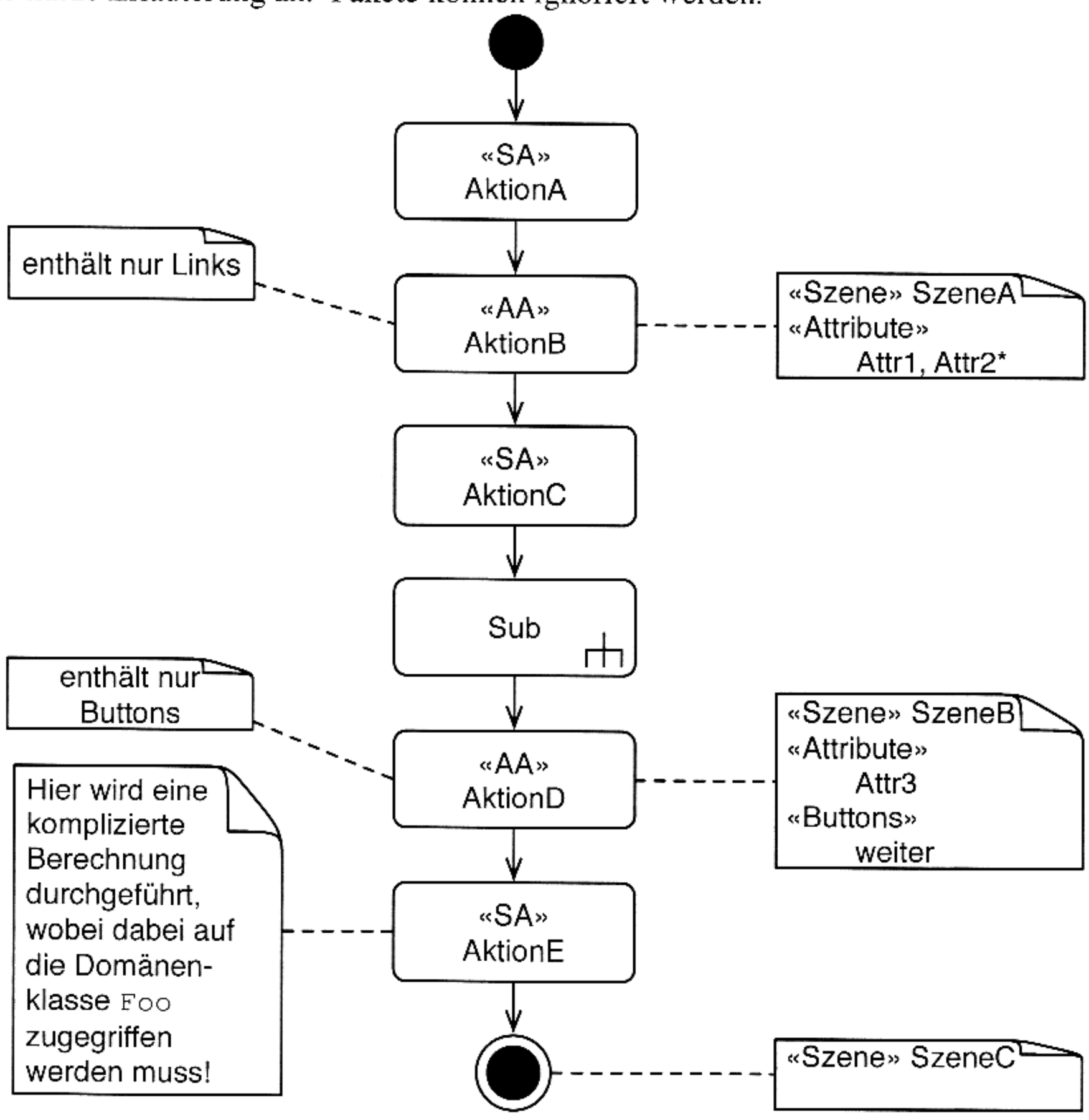
Musterlösungen zur Klausur am 4.3.2006

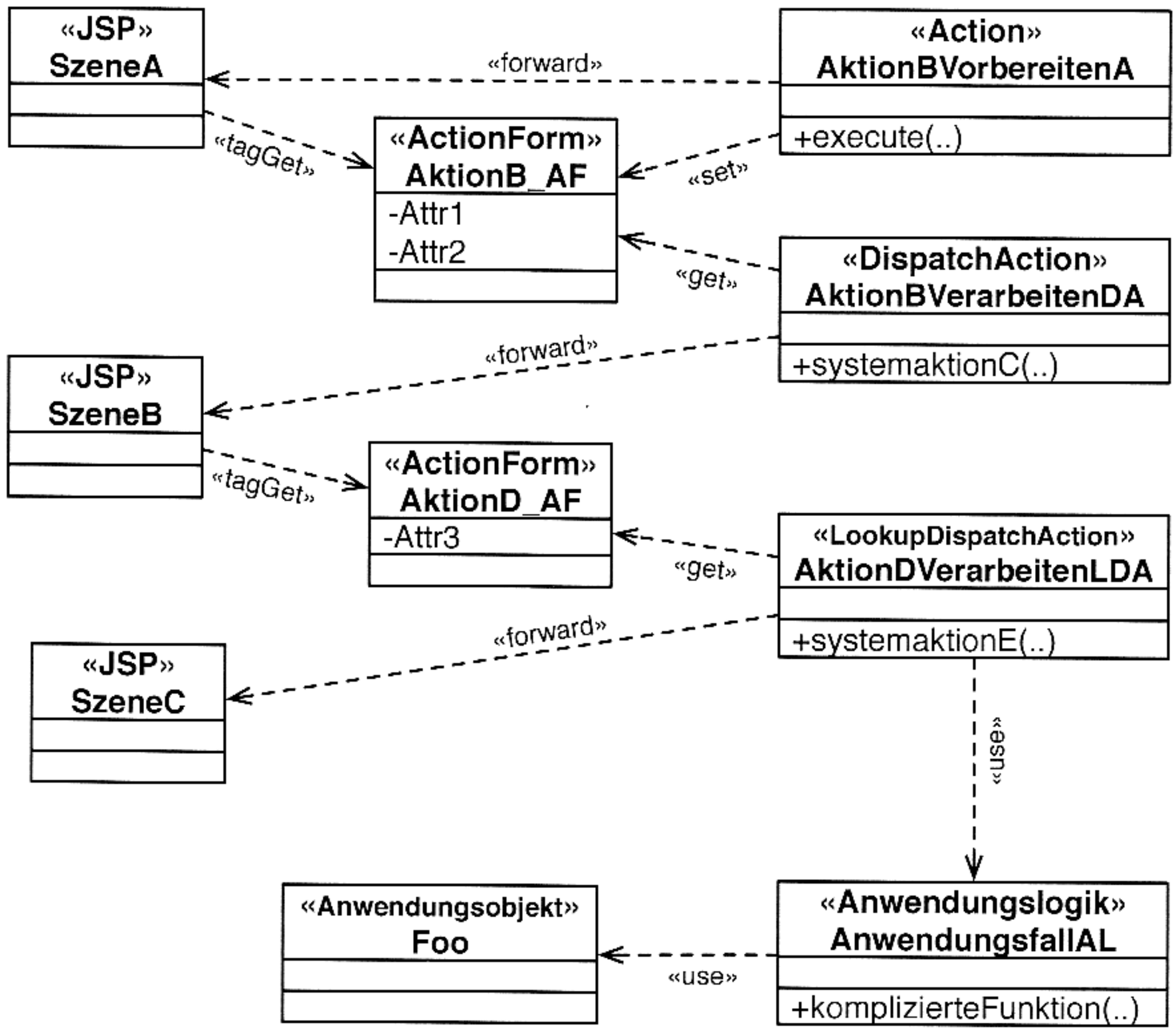
Erläuterungen:

Die neu eingefügten Systemaktionen sind hervorgehoben. Jede zu einer Akteurtätigkeit gehörende implizite Systemaktion wird nun explizit dargestellt - diese Systemaktionen sind mit „Implizite“ gekennzeichnet. Vor der Subaktivität muss eine Systemaktion stehen, um den Aufruf der Subaktivität vorzubereiten - eine solche Aktion ist bereits im Schritt vorher eingefügt worden („«SA» ImpliziteSA3“). Da die Akteuraktion „«AA» B“ gefüllte Daten erwartet („AttrD*“), muss hier eine Systemaktion zur Bereitstellung der Daten eingefügt werden („«SA» Vorbereiten“). Da die erste Akteuraktion („«AA» A“) keine gefüllten Daten erwartet, ist dieser Schritt hier nicht erforderlich.

Aufgabe 6 *Entwurfsklassendiagramm (10 Punkte)*

Wandeln Sie folgendes Aktivitätsdiagramm in ein Entwurfsklassendiagramm um und fügen Sie bitte eine kurze Erläuterung an. Pakete können ignoriert werden.





Erläuterungen:

Zunächst wird die Anwendungslogikklasse „AnwendungsfallAL" erstellt. Für die Szenen werden jeweils JSPs erstellt („SzeneA", „SzeneB" und „SzeneC). Die ersten beiden Szenen verwenden Attribute, daher werden passende ActionForms mit den jeweiligen Attributen erstellt („ActionB_AF" und „ActionD_AF"). Die zur Akteurtätigkeit „AktionB" gehörende Szene „SzeneA" verwendet keine Buttons, daher wird hier eine DispatchAction angelegt („AktionBVerarbeitenDA"), bei der anderen Akteuraktion benötigen wir eine LookupDispatchAction („AktionDVerarbeitenLDA"). Die Systemtätigkeiten „AktionC" und „AktionE" werden als entsprechende Methoden in diesen beiden Actions modelliert. Die Systemaktion „ActionA" wird als normale Action umgesetzt. Diese dient dem Befüllen des ActionForms und wird entsprechend benannt („AktionBVorbereitenA"). Die komplizierte Berechnung wird nach dem Prinzip des „Separation of Concerns" in der Anwendungslogik durchgeführt und dort als entsprechende Methode eingetragen. Die Abhängigkeiten werden entsprechend eingetragen.