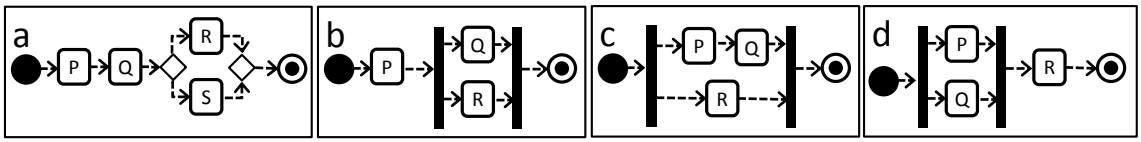
3. Übung – Modellierung kooperierender Komponenten

1 Ablauf des Prozesses

Wir haben bei der Ausführung eines Verfahrens alle Schritte beobachtet. Wir haben die folgende Ereignisreihe erfahren:

> Der Prozess beginnt, P fängt an, P beendet sich, Q fängt an, R fängt an, Q beendet sich, R beendet sich, der Prozess wird beendet.



Welches Prozessmodell von a, b, c, d kann ein richtiges Modell des Systems sein?

2 Steuerungsprozess (aufgrund von Quellcode)

Betrachten wir die folgende C-Funktion.

```
1  unsigned long long f(int n){
2     if (n <= 0) {
3        return 0;
4     } else if (n == 1) {
5        return 1;
6     } else {
7        unsigned long long a = f(n - 1);
8        unsigned long long b = f(n - 2);
9        return a + b;
10    }
11 }
```

a. Was für einen Kontrollfluss hat diese Funktion?
b. Was versichert es, dass die Funktion früher oder später sicher terminieren wird?
c. Identifizieren Sie die Datenabhängigkeiten (-fluss) zwischen den einzelnen Aktivitäten!
d. Falls die Programmiersprache oder die Ausführungsumbgebung es erlaubt, wo gibt es eine Möglichkeit zur Parallelisierung der Ausführung?
e. Überprüfen Sie, ob dieses Prozess wohlstrukturiert ist.

3 Prozessmodellierung aufgrund von schriftliche Spezifikation

Die Code-Bibliothek einer großen Software-Stiftung (z.B. in Git) unterstützt die Entwicklung von zahlreichen Open-Source-Software. In diesen Softwareprojekten wird die Entwicklung öffentlich gemacht, in dem neben den zuverlässigen, internen Entwicklern auch häufig fremde Entwickler Korrekturen oder neu implementierte Fähigkeiten einsenden dürfen. Es muss aufgepasst werden, dass in der ausgegebenen Software nur rechtsgemäß (z.B. mit der Zustimmung des Arbeitgebers) eingereichter Quellcode erscheint.

a. Falls ein Entwickler zu einem Projekt mit dem selbst produzierten Quellcode beitragen möchte, soll er von seinem Status abhängig verschiedene Schritte tun. Interne Entwickler dürfen direkt auf den für das gegebene Projekt reservierten Bereich der Code-Bibliothek schreiben. Externe Entwickler sollen ihren Code zuerst zur Revision (*code review*) einreichen; danach soll der Code von einem internen Entwickler überprüft und dann entweder akzeptiert oder abgewiesen werden. Falls dieser externe Code kürzer, als der Schwellenwert ist (zB. Fehlerausbesserung mit nur ein paar Reihen), dann muss der Ersteller nach der Akzeptanz nur eine kurze Äußerung schreiben, das es ins Code eingeführt werden kann.
   Nach der Akzeptanz des größeren Codes wird von der Rechtsabteilung der Stiftung in einem Verwaltungsverfahren geprüft, ob der geisteseigentumrechtliche Status der Änderungen geklärt werden kann. Nach erfolgreichem Abschluss dieses Verfahrens darf der interne Entwickler den

Code integrieren. Bei den neuen, vor ihrer ersten offiziellen Veröffentlichung stehenden Projekten wird eine Ausnahme gemacht: mit der Integration des Codes von dem internen Entwickler in die Code-Bibliothek soll der Abschluss des Verwaltungsverfahrens nicht abgewartet werden. Erstellen Sie ein Prozessmodell der oben beschriebenen Tätigkeiten.

b. Das Entwicklungsprojekt einer Software besteht daraus, dass immer neuere und neuere Modifikationen des Quellcodes durchgeführt werden, solange das Projektmanagement nicht so sieht, dass die Software ausreichend stabil zu einer offiziellen Ausgabe ist (*release*). Zu diesem Zeitpunkt wird eine neue stabile Version der Software ausgegeben, dann kommt wieder die Entwicklung an die Reihe, usw. Erstellen Sie ein Prozessmodell der beschriebenen Tätigkeiten.
c. In was für Verhältnis stehen miteinander die in den obigen Teilaufgaben erstellte Prozessmodelle?
d. (Optionale Aufgabe) Überprüfen Sie, ob die Prozessmodelle wohlstrukturiert sind.

4 Modellierung eines zusammengesetzten Systems

In dieser Aufgabe betrachten wir einen Cloud-basierten Datenspeicherung-Dienst (siehe Dropbox, Google Drive, Tresorit), wir beschränken uns aber dabei zuerst auf eine einzige Datei. Je eine Kopie der Datei befindet sich auf dem Server und bei dem Klienten (z.B. Laptop), am Anfang sind die zwei Kopien synchronisiert, sie haben also den gleichen Inhalt. Die Modifikationen der Datei werden während der *Synchronisation* zwischen den Kopien weitergeleitet. Falls vor der Synchronisation beide Kopien modifiziert werden, dann sprechen wir über eine *Kollision*, die von dem Benutzer auf Klientenseite aufgelöst werden soll.
Es kann sich lokal beim Klienten, oder (zB. wegen Betätigungen von anderen Klienten) beim Server ändern. Sie können sich nach der Angabe vom Benutzer, oder spontan synchronisieren. Der eine bekommt hier die Änderung, und so werden die zwei Kopien wieder in Synchron sein. Falls man seit der letzten Synchronisierung beide Kopien geändert hat, passiert ein Konflikt (Kollision). In diesem Fall vergleicht der Klient die zwei Varianten, und überlässt die Lösung der Kollision den Benutzer.

a. Mit Hilfe einer Zustandsmaschine modellieren Sie zuerst (teilweise) die Funktionalität des Laptop-Klienten. Am Anfang ist der Klient in einem *synchron*en Zustand (der Inhalt der lokalen Dateikopie übereinstimmt mit dem Inhalt der Kopie am Server zu dem Zeitpunkt der letzten Synchronisation), aber nach einem *Schreiben* geht er in einen Zustand *verschmutzt* über (und nach weiteren Schreiben bleibt er da). Nach einem *Verwerfen* der Änderungen geht der Klient aus einem beliebigen Zustand in den *synchron*en Zustand über.
b. Die mögliche Zustände des Servers (wenn wir nur die Synchronität mit dem gegebenen Klient betrachten) sind die Zustände *synchron* und *neu*. Es kann passieren, dass ein anderer Benutzer mit den entsprechenden Schreibzugriffrechten (oder sogar derselbe Benutzer mit der Hilfe eines anderen Klienten, z.B. mit einem Handy), die auf dem Server befindlichen Datei aktualisiert. Modellieren Sie das diesbezügliche Verhalten des Servers.
c. Falls der Server im *synchron*en Zustand ist, lädt der Klient auf die Eingabe *Synchronisieren* die eventuelle lokale Modifikationen der Datei auf den Server hoch, und geht dabei in den *synchron*en Zustand über. Die *Synchronisieren* Eingabe bekommt auch der Server. Beim Synchronisieren tauschen der Klient und der Server Informationen – wie kooperieren die beiden Automaten?
d. Falls der Server in dem Zustand *neu* ist, geht er auf die Eingabe *Synchronisieren* in den *synchron*en Zustand über; der Klient bewegt sich aus dem *synchron*en Zustand nicht, aber aus dem *verschmutzt*en Zustand geht er in den Zustand *Kollision*. Was bedeutet es? Was soll im Zustand *Kollision* geschehen? Wie kooperieren die beiden Automaten?
e. (Optionale Aufgabe) Der Klient synchronisiert mit dem Server auch ohne Benutzereingabe periodisch. Was bedeutet es? Wie kooperieren die beiden Automaten?
f. (Optionale Aufgabe) Ausführung des vollständigen komplexen Zustandsraumes aufgrund den im gemischten Produkt teilnehmenden beiden Automaten.
g. (Optionale Aufgabe) Der Server und der Klient können in diesem Modell die innere Zustände vom anderen unmittelbar betrachten, und die Synchronisation passiert auch in einem Moment zwischen ihnen. In einem realen geteilten System muss man die Kommunikation zwischen dem Server und den Klienten mit Nachrichtenaustausch verwirklichen; und zwischen der Zeit vom Senden und Empfangen kann eine längere Zeit vergehen. Denken wir durch we man das Modell verfeinern könnte, dass es auch diese Detailen zeigt!