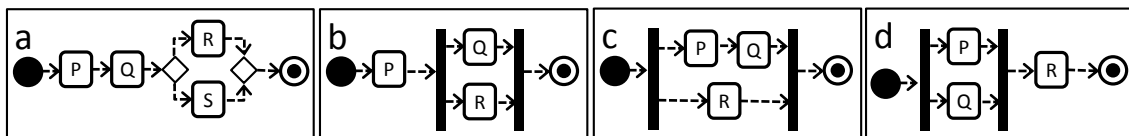


### 3. gyakorlat – Folyamatmodellek, kooperáló viselkedésmodellek

#### 1. Folyamat lefutása

Egy folyamat végrehajtása során az összes lépést megfigyeltük. A következő eseménysor bekövetkeztét észleltük:

Folyamat indul,  $P$  elkezdődik,  $P$  befejeződik,  $Q$  elkezdődik,  $R$  elkezdődik,  $Q$  befejeződik,  $R$  befejeződik, Folyamat befejeződik.



Az a, b, c, d folyamatmodellek közül melyek lehetnek helyes modelljei a rendszernek?

#### 2. Vezérlési folyam (forráskód alapján)

Tekintsük az alábbi C nyelvű függvényt.

```

1 unsigned long long f(int n)
2 {
3     if (n <= 0) {
4         return 0;
5     } else if (n == 1) {
6         return 1;
7     } else {
8         unsigned long long a = f(n - 1);
9         unsigned long long b = f(n - 2);
10        return a + b;
11    }
12 }

```

- Milyen vezérlési folyamatot határoz meg a függvény?
- Mi biztosítja azt, hogy a függvény előbb-utóbb terminál?
- Azonosítsuk az adatfüggőségeket (adatáramlást) a tevékenységek között!
- Ha a programozási nyelv vagy a futtatókörnyezet megengedi, hol van lehetőség párhuzamosításra?
- Ellenőrizzük, hogy jólstrukturált-e ez a folyamat!

#### 3. Folyamatmodell szöveges specifikáció alapján

Egy nagy szoftveralapítvány kódtára (pl. Git) számos nyílt forráskódú szoftver fejlesztésének ad otthont. A megbízható belső fejlesztőkön kívül külsősök is gyakran küldenek be hibajavításokat vagy újonnan megvalósított képességeket. Oda kell figyelni arra, hogy a kiadott szoftverben csak jogszerűen (pl. munkaadó beleegyezésével) bekerült forráskód szerepeljen.

- Ha egy fejlesztő hozzá szeretne járulni egy projekthez az általa készített forráskóddal, akkor a saját státuszától függő lépéseket kell tennie. Belső fejlesztők közvetlenül írhatnak a kódtár adott projekt részére fenntartott területére. Külsős fejlesztőknek először átvizsgálásra (*code review*) be kell nyújtaniuk a kódjukat; ezután egy belső fejlesztőnek ellenőriznie kell azt, és utána vagy elutasítania, vagy elfogadnia. Ha a kívülről érkező kód egy bizonyos küszöbértéknél rövidebb (pl. néhány soros hibajavítás), akkor az elfogadás után a készítőjének már csak egy rövid hozzájárulási nyilatkozatot kell tennie, hogy beolvasható legyen a kódtárba.

A nagyobb lélegzetű külső hozzájárulások (pl. egy teljesen új modul beépítése) esetében azonban az elfogadást követően az alapítvány jogi osztálya egy külön adminisztratív eljárásban tisztázza a változtatások szellemi tulajdonának jogállását, és csak ennek sikeres lezárása után olvashatja be a belső fejlesztő a kódot. Frissen indított, első hivatalos kiadásuk előtt álló projekteknél itt tesznek egy kivételt: az elfogadott külső hozzájárulás kódtárba beolvasztásával nem kell megvárni ezt az adminisztratív eljárást. Készítsünk folyamatmodellt az itt leírt tevékenységekből!

- b) A szoftver fejlesztési projektje abból áll, hogy újabb és újabb módosításokat végeznek a forráskódon, amíg a projekt vezetése úgy nem látja, hogy a szoftver kellően stabil egy hivatalos kiadáshoz (*release*). Amikor eljött ez a pont, akkor közzétesznek egy új stabil verziót a szoftverből, majd ismét a fejlesztésen a sor, és így tovább. Készítsünk folyamatmodellt az itt leírt tevékenységekből!
- c) Milyen viszonyban állnak egymással a fenti részfeladatokban elkészített folyamatmodellek?
- d) (Kiegészítő feladat.) Ellenőrizzük, hogy jólstruktúráltak-e a folyamataink!

## 4. Összetett rendszer modellezése

Felhő alapú adattárolást modellezünk (ld. Dropbox, Google Drive, Tresorit), egyetlen állományra szorítkozva. Az állománynak a szerveren és a kliensnél (pl. laptop) is elérhető egy-egy replikája, kezdetben azonos tartalommal. A fájl módosításai *szinkronizálás* során továbbítódnak a példányok között. Ha szinkronizáció előtt mindkét példányt módosítják, akkor *ütközés* lép fel, amelyet a felhasználónak kell feloldania a kliensen.

Lokálisan a kliensnél, illetve (pl. másik kliens tevékenységének hatására) módosulhat a szerveren is. Felhasználói utasításra, valamint időről időre spontán módon a kliens és a szerver szinkronizálhat; ilyenkor az esetleges módosítás eljut a másik példányhoz is, és újra szinkronban lesz a két másolat. Ha a legutóbbi szinkronizáció óta egymástól függetlenül mindkét replikát módosították, akkor viszont konfliktushelyzet (ütközés) áll fenn. Ilyenkor a kliens a saját és a szerverről letöltött változatot összehasonlítja, és a felhasználóra bízva az ütközés feloldását.

- a) Modellezzük először a kliens (részleges) működését állapotgéppel! A kliens kezdetben *szinkron* állapotú (a lokális fájlmásolat egyezik azzal, ami a szerveren a legutóbbi szinkronizációkor volt / lett), ám *írás* bemenet hatására a *piszkos* állapotba kerül (és további *írás* hatására is ottmarad). Az *elvetés* bemenet hatására tetszőleges állapotból újra *szinkron* állapotba kerül.
- b) A szerver lehetséges állapotai (csupán az adott klienssel való szinkronizációt vizsgálva) a *szinkron* és a *frissült*. Előfordulhat, hogy a megfelelő írási jog birtokában egy másik felhasználó (vagy ugyanazon felhasználó egy másik kliens, pl. a telefonja segítségével) frissíti a szerveren található állományt.
- c) Ha a szerver *szinkron* állapotban van, akkor a kliens a *szinkronizálj* bemenet hatására feltölti az esetleges lokális módosításokat a szerverre, és szintén *szinkron* állapotba kerül. A *szinkronizálj* bemenetet a szerver is megkapja. Hol kooperál a két automata?
- d) Ha a szerver *frissült* állapotban van, akkor a kliensnek adott *szinkronizálj* bemenet hatására a szerver *szinkron* állapotba kerül; a kliens pedig *szinkron* állapotból nem mozdul, de *piszkos* állapotból *ütközés* állapotba megy. Mit jelent ez? Mi történjen az ütközés állapotban? Hol kooperál a két automata?
- e) (Kiegészítő feladat.) A kliens időnként magától is szinkronizál a szerverrel, felhasználói bemenet nélkül. Mit jelent ez? Hol kooperál a két automata?
- f) (Kiegészítő feladat.) Fejtsük ki a teljes összetett állapotteret a vegyes szorzatban résztvevő két automata alapján.
- g) (Kiegészítő feladat.) Ebben a modellben a szerver és a kliens közvetlenül figyelembe tudják venni egymás belső állapotát, és a szinkronizáció is pillanatszerűen végbemegy közöttük. Egy valódi elosztott rendszerben azonban üzenetváltással kell a kliens és a szerver közötti kommunikációt megvalósítani; a küldés és a válasz megérkezése között pedig huzamosabb idő eltelhet. Gondoljuk végig, hogy lehetne finomítani a modellt, hogy ezeket a részleteket is tükrözze!