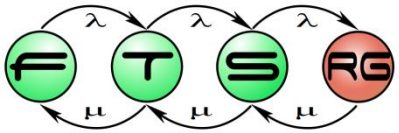
# Folyamatmodellezés

**Budapesti Műszaki és Gazdaságtudományi Egyetem**
**Hibatűrő Rendszerek Kutatócsoport**



# Tartalom

- Ismétlés, kitekintés
- Folyamatmodellezés célja
- Folyamatmodellek
- Vezérlési folyam
- Megvalósítás

# Tartalom

# Ismétlés: felépítési vs. viselkedési modellek

- Felépítési (*structural*) modellek
  - Statikus
  - Rész és egész, összetevők
  - Kapcsolatok, összeköttetések
- Viselkedési (*behavioral*) modellek
  - Dinamikus
  - Időbeli lefolyás
  - Állapot, folyamat
  - Reakciók a külvilágra
- Nem fed le mindent, nem válik élesen szét…

Az autóban van kamera és kormányvezérlő

A kamera jeleket küld a sáv elhagyásáról (mennyit? mikor?)

A sávtartó rendszerben a kamera jeleit fogadva a kormányvezérlő beavatkozik (mikor/hogyan?)

# Viselkedésmodellek fő kérdései

- Mit „csinál” a rendszer?

Esemény alapú modell

**Folyamat alapú modell**

…

- Most „milyen”, és hogyan változik a rendszer?

Állapot alapú modell

# Folyamat

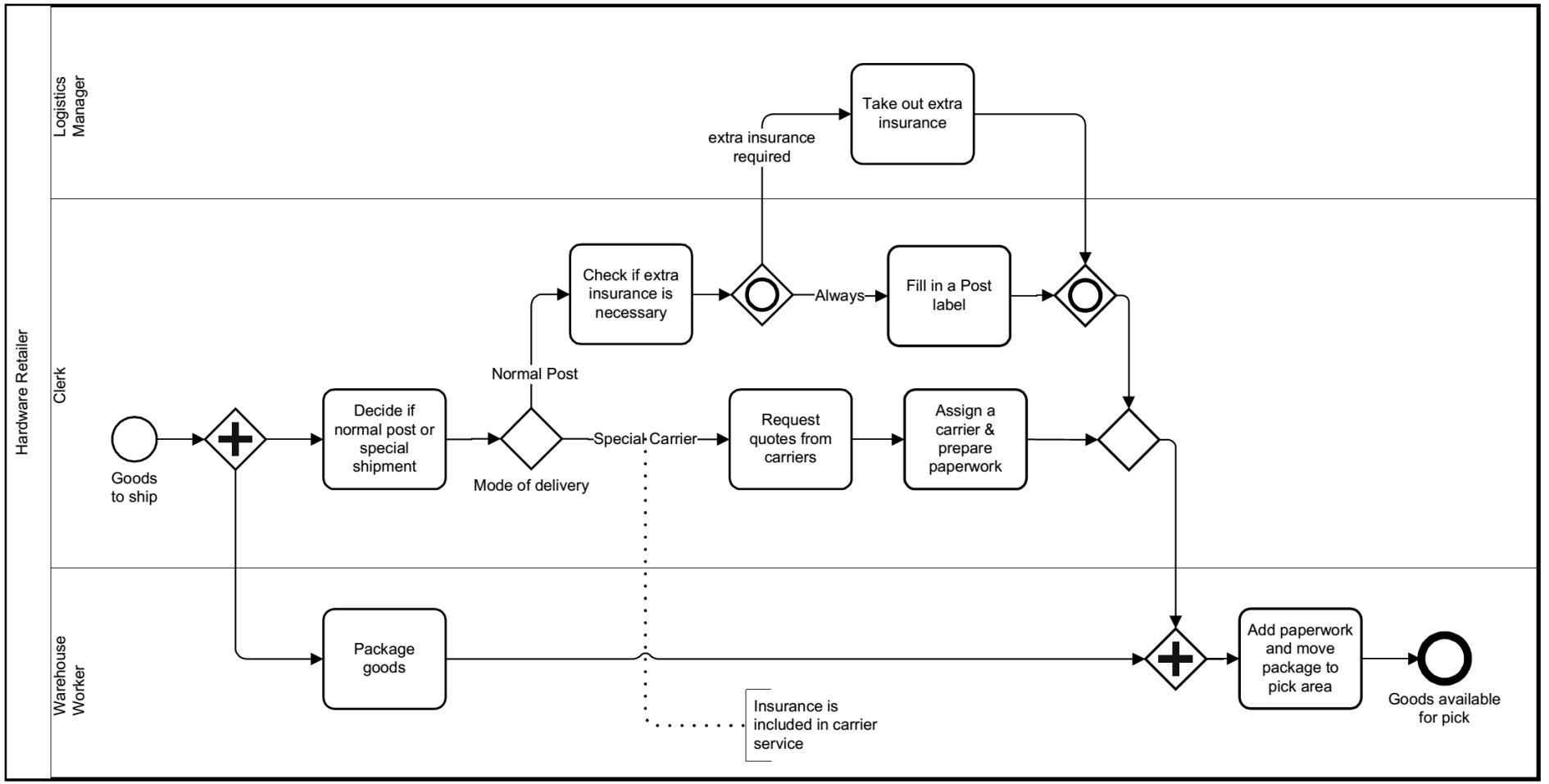
**Folyamat**: lépések sorozata, melyek sorrendben történő végrehajtása valamilyen célra vezet.

# Tartalom

# Folyamatmodellezés célja

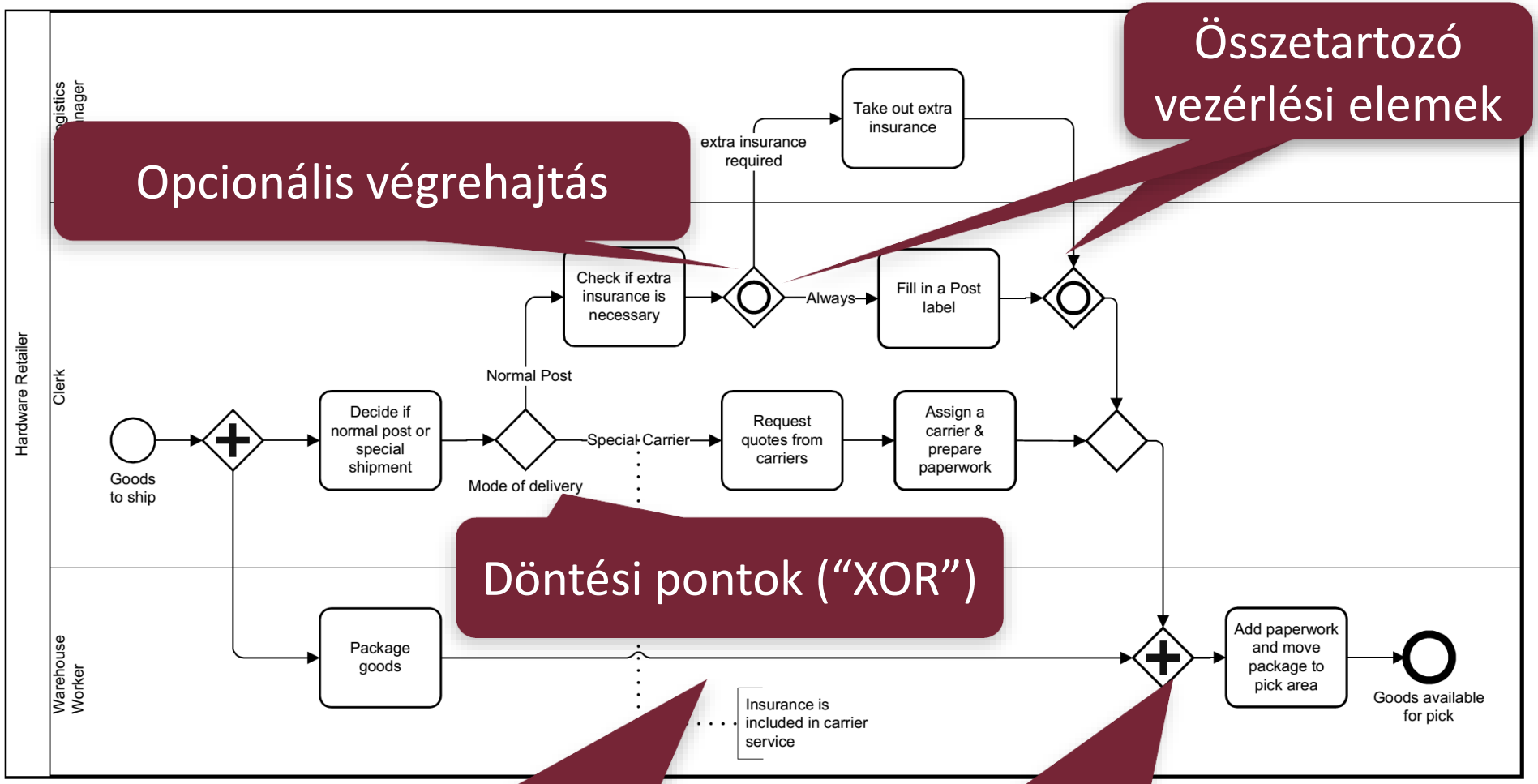
- Dokumentáció
- Implementáció
  - Végrehajtható modellek
  - Kódgenerálás
- Modell szintű ellenőrzés (Verifikáció)
  - Szimuláció
  - Monitorozás
  - Automatizált modellellenőrzés

# Példa: HW rendelés kiszállítása




omg.org, BPMN 2.0 by Example

# Példa: HW rendelés kiszállítása




Összetartozó vezérlési elemek

Opcionális végrehajtás

Döntési pontok (“XOR”)

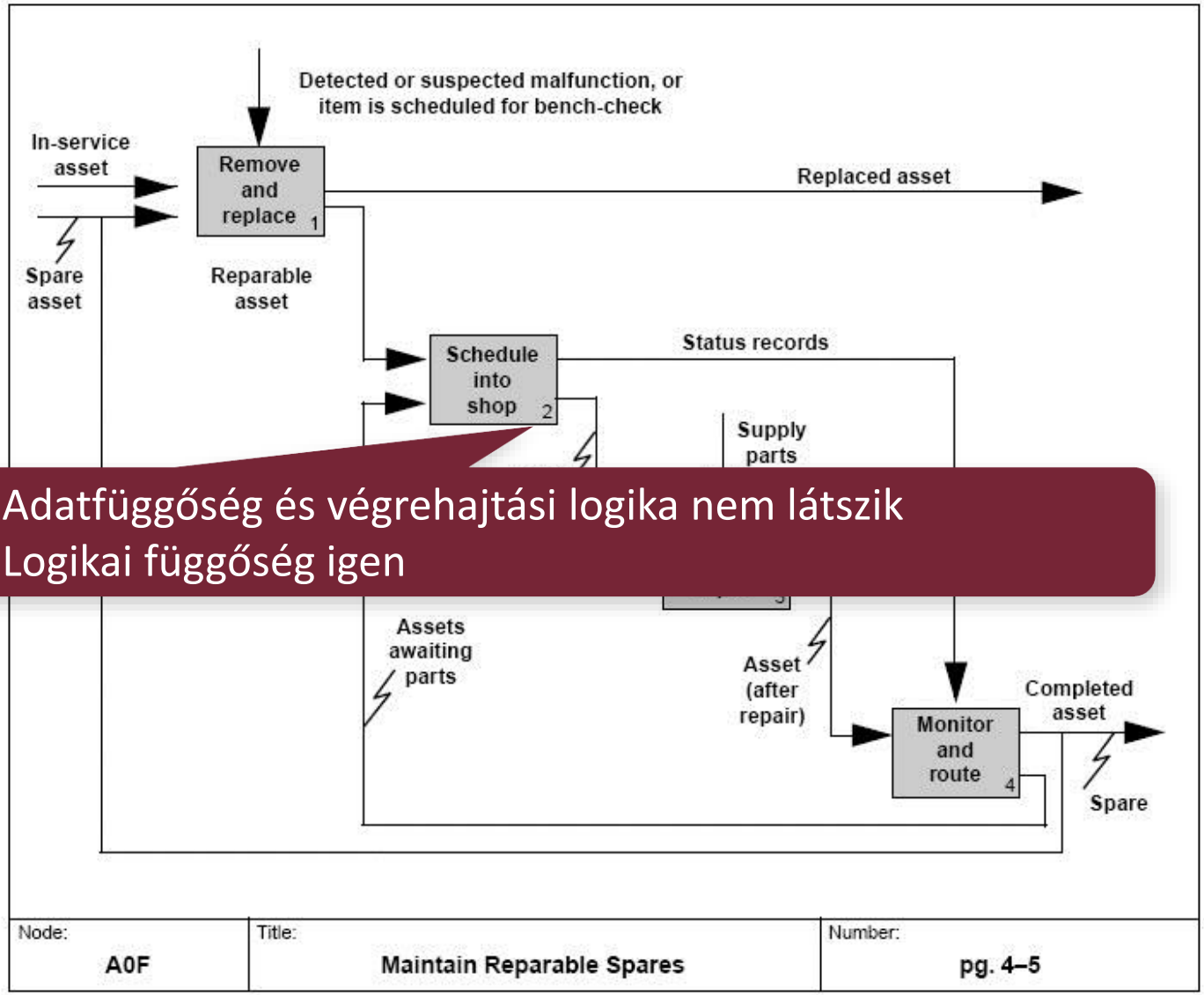
Lépések végrehajtási sorrendje

“Párhuzamos” (független) végrehajtás (“AND”)

# Mire épül?

- Történeti előzmények
  - Programok vezérlési szerkezete
  - Ütemezés (pl. GANTT diagramok)
  - Gyártási/irodai folyamatok modellezése
  - IDEF-0: 1980-as évek, US AirForce
  - Logisztikai folyamatok leírása
  - Üzemeltetés: “runbook”
- Közös elemek
  - Vannak elemi lépések
  - Köztük függőségek (idő? adat? sorrend?)
  - Döntési pontok
  - → általános célú folyamatmodellezési nyelvek (pl. BPMN)

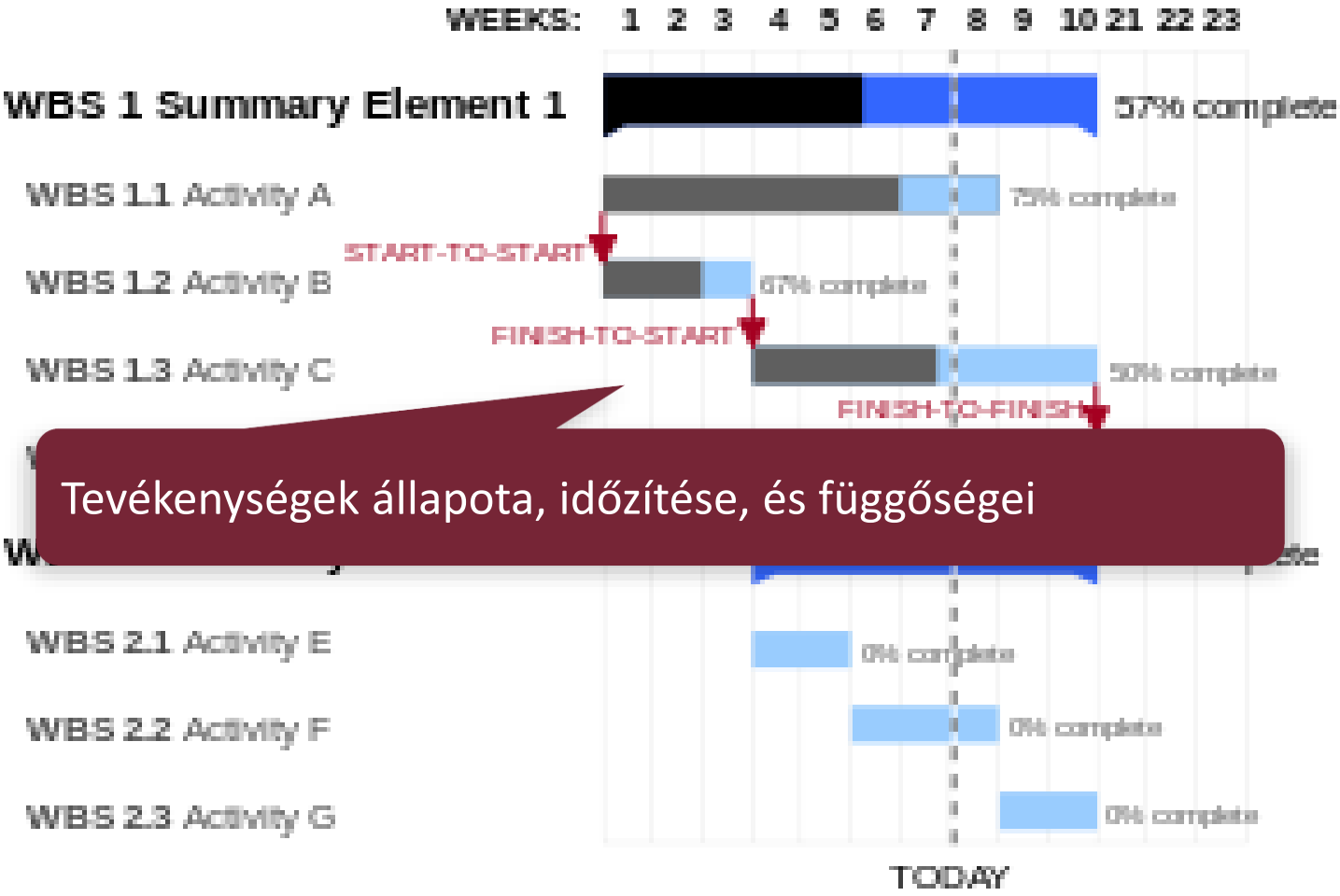
# Példa: IDEF-0




Adatfüggőség és végrehajtási logika nem látszik
Logikai függőség igen

Defense Acquisition University - *Systems Engineering Fundamentals.* Defense Acquisition University Press, 2001

# Példa: GANTT




Tevékenységek állapota, időzítése, és függőségei

wikipedia.org

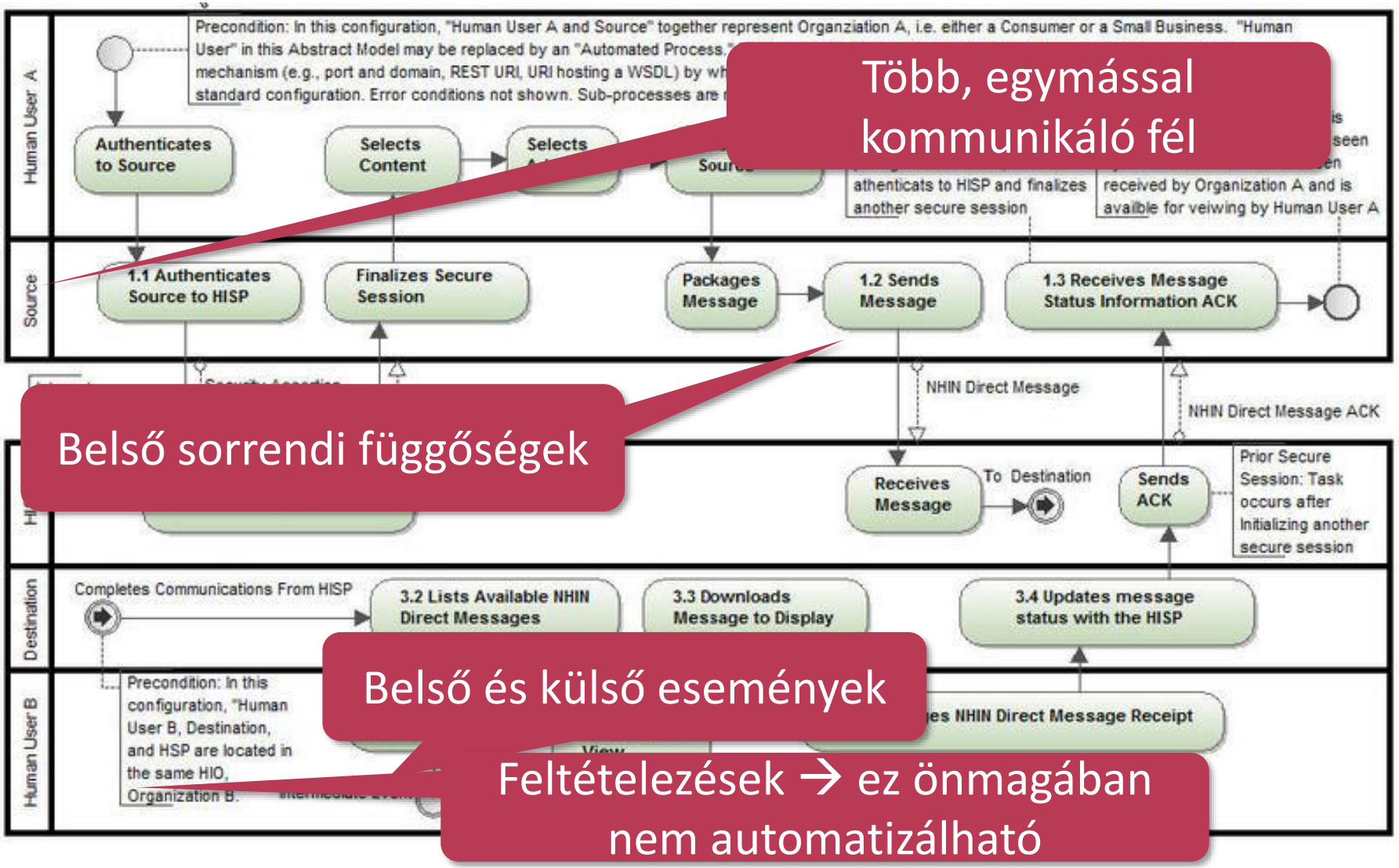
# Mit használ fel?

- Ötlet rendszer/szoftvertervezésben:
  - használjunk fel meglévő elemeket
  - Írjuk le az összetett rendszer működését
- Alapelem lehet sokféle
  - webform validáció, email küldés, adatbázisművelet, távoli webszolgáltatás, emberi interakció, SMS küldés, diagram kirajzolás, stb.
- Mire “fordul” a vezérlési logika?
  - Lehet közvetlen kód (C/C++, C#, Java, …)
  - Lehet egy végrehajtó környezet bemenete
    - “Csinálj nekem ilyen folyamatot”

# Hol használnak még folyamatmodelleket?

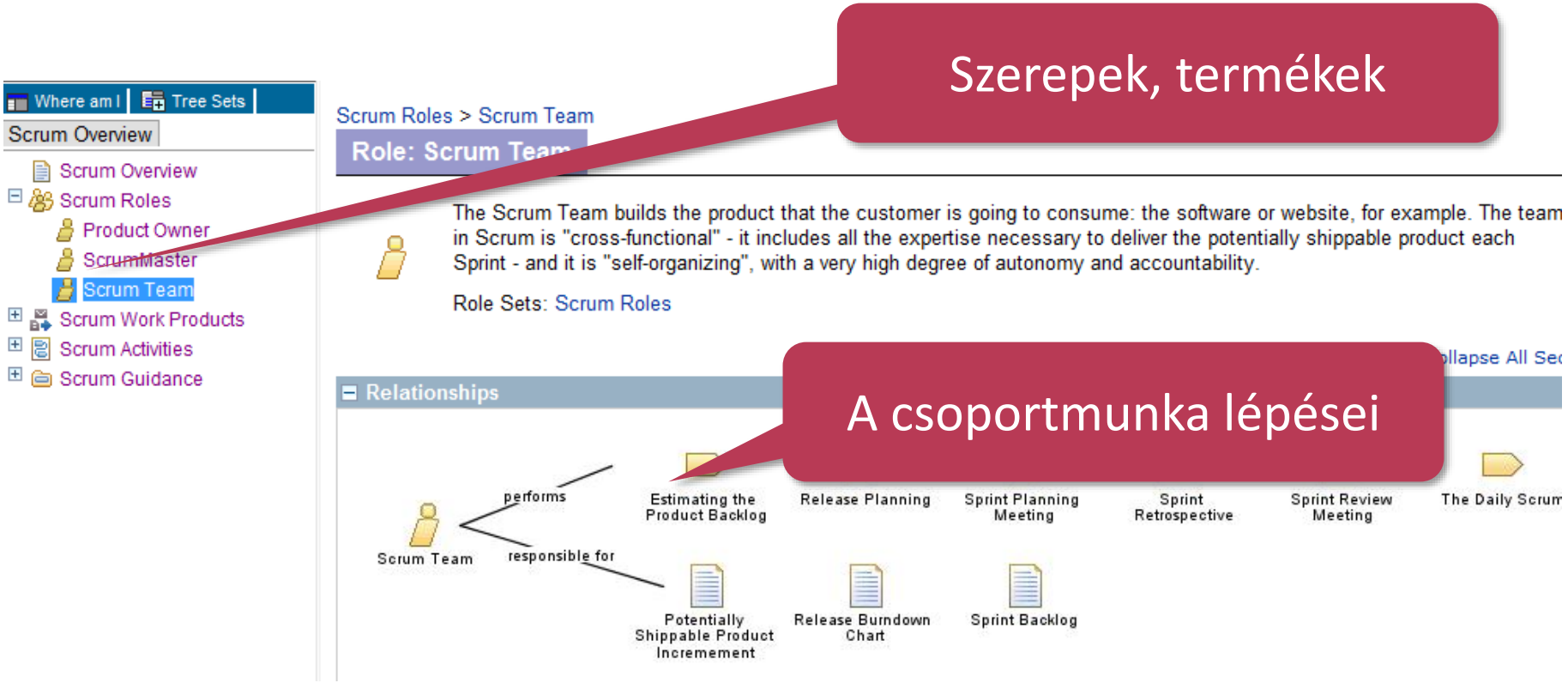
- Informatikai rendszerek működtetése
  - ITIL, UK Gov. kezdeményezés
- Protokoll specifikáció
  - Összetett rendszer részei hogy működnek együtt
  - Melyik komponensnek mi a szerepe
- Végrehajtható folyamatok tervezése
  - Rendeléskiértékelés, hitelbírálat előkészítése, …
- Adatfeldolgozási/elemzési folyamatok

# Példa: egészségügyi adatok kezelése




http://wiki.directproject.org/Abstract+Model+Examples

# Példa: agilis fejlesztés, mint folyamat

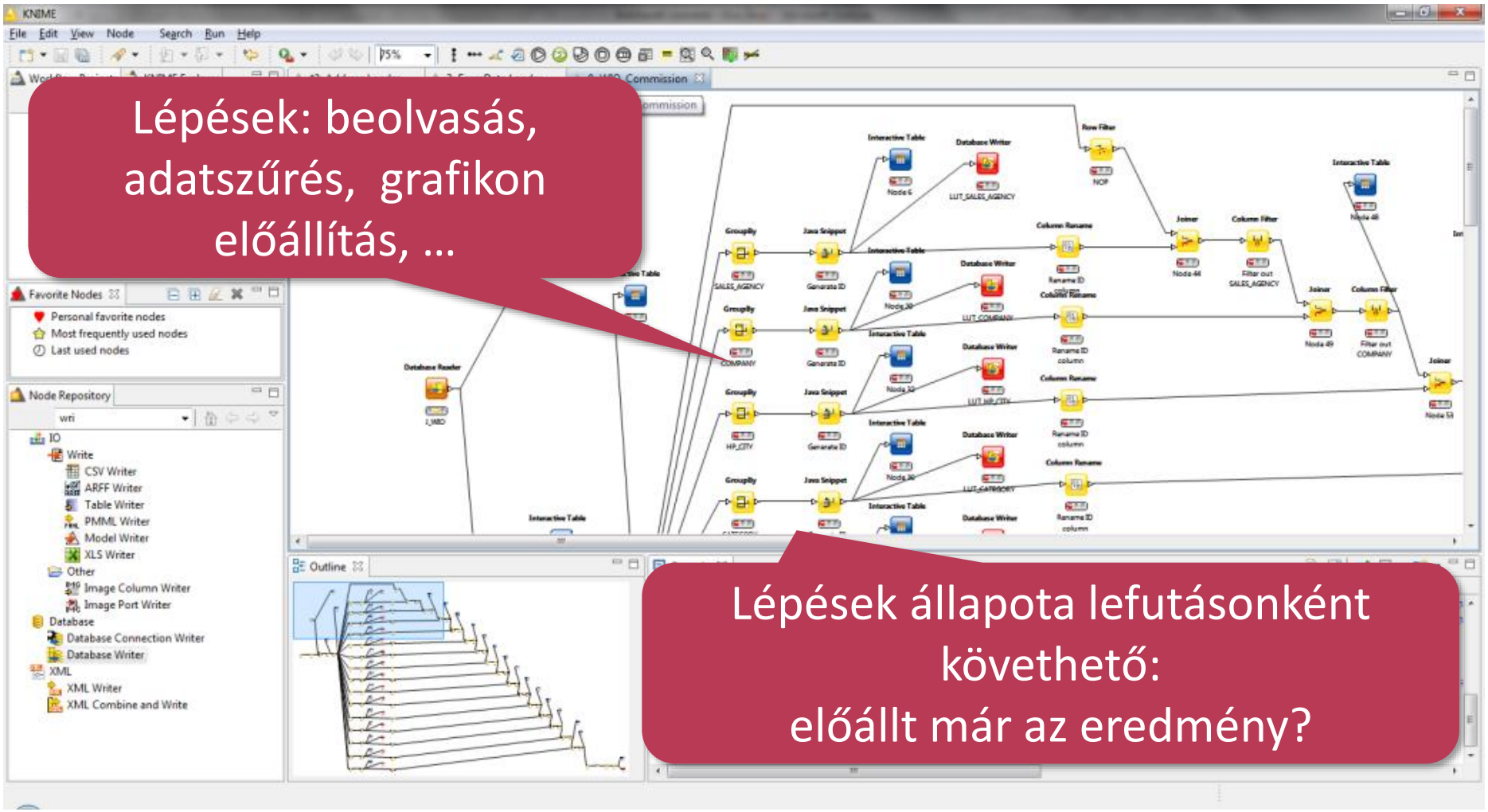



http://www.eclipse.org/epf/

# Példák

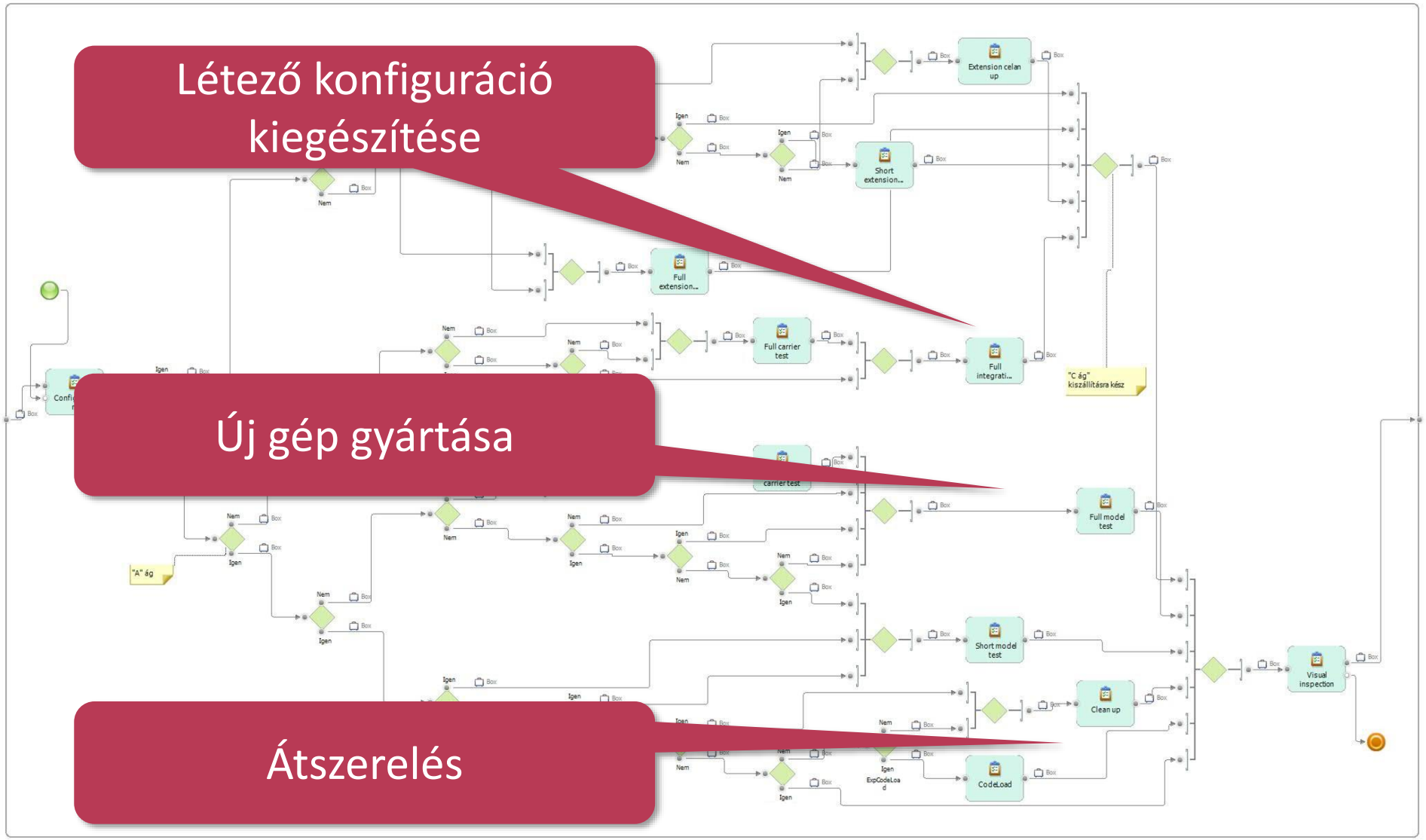
- Banki folyamatok modellezése
  - Milyen tevékenységek tudnak le “záráskor”?
  - Át tud-e állni a bank a napi többszöri utalásra?
- Gyártási folyamat modellezése
  - Optimális gyártásütemezés: átszereljünk vagy újat gyártsunk?
  - Mi történik a gyárban?
  - (ld. Szimuláció előadás)
- Üzletkötési folyamatok modellezése
  - Hol vannak ismétlődő kommunikációs minták?
  - Adatfeldolgozás modell alapon

# Példa: adatfeldolgozási folyamat




Eszköz: pl. KNIME

# Példa: gyári tesztelés, mint folyamat

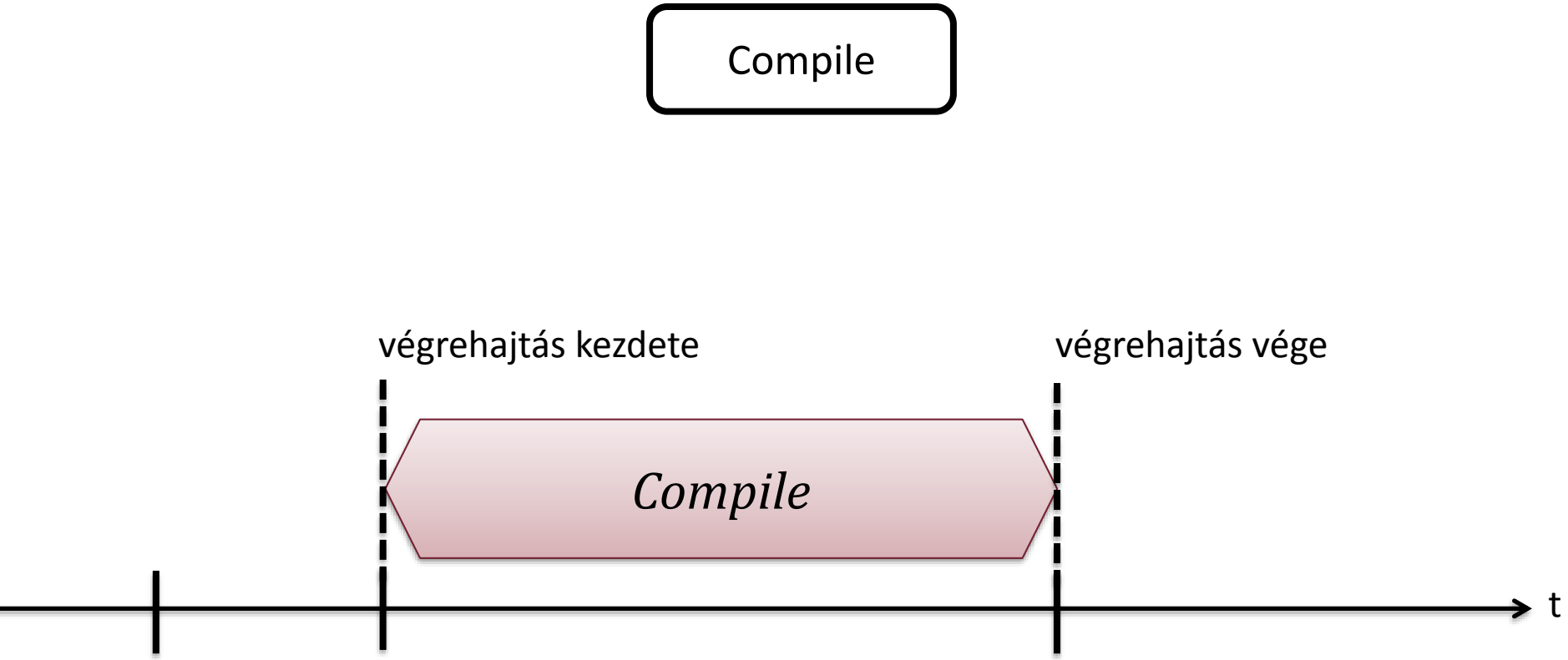


# Folyamatok tervezésének alapfogalmai

- Folyamat leíró nyelv
  - BPMN, jPDL, XPDL, BPEL, UML AD, …
  - Vezérlés, adatáramlás
  - Adatstruktúrák kapcsolhatóak hozzá
  - Végrehajtandó lépések definíciója
  - Időzítések, erőforrások
- Folyamat minta (template)
  - Pl. “Jegyrendelés” folyamat
  - Verziózás…
- Folyamat példány (instance)
  - „Gönczy László jegyet rendel Lisszabonba”

# Tartalom

# Elemi tevékenység



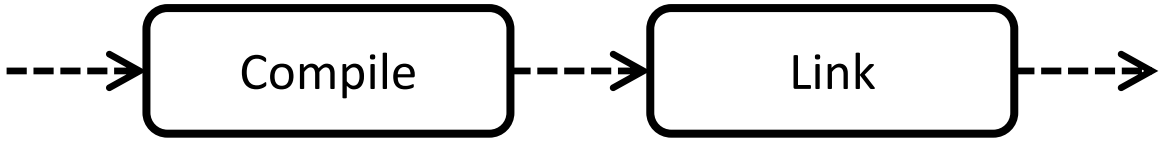
# Definíció: Elemi tevékenység

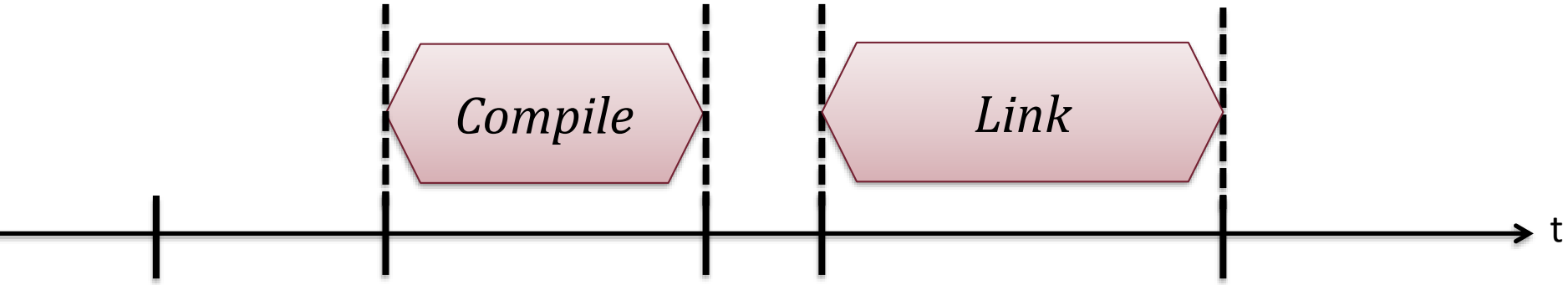
Az **elemi tevékenység** olyan

- időbeli kiterjedéssel rendelkező tevékenység,
- amelynek a megkezdésén és befejezésén túl további részleteit nem modellezzük.

Compile

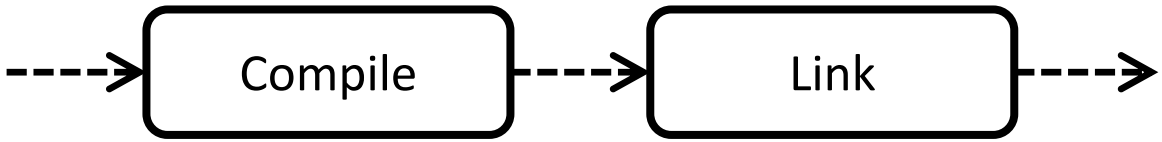
# Szekvencia, vezérlésfolyam



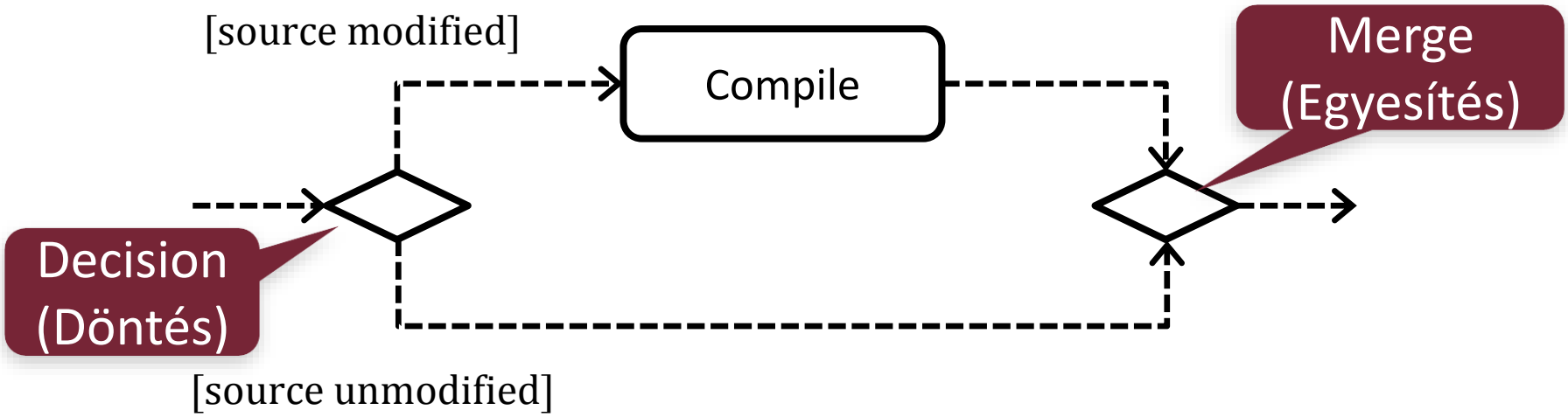



# Definíció: Szekvencia

A **Szekvencia** tevékenységek végrehajtási sorrendjét definiálja.



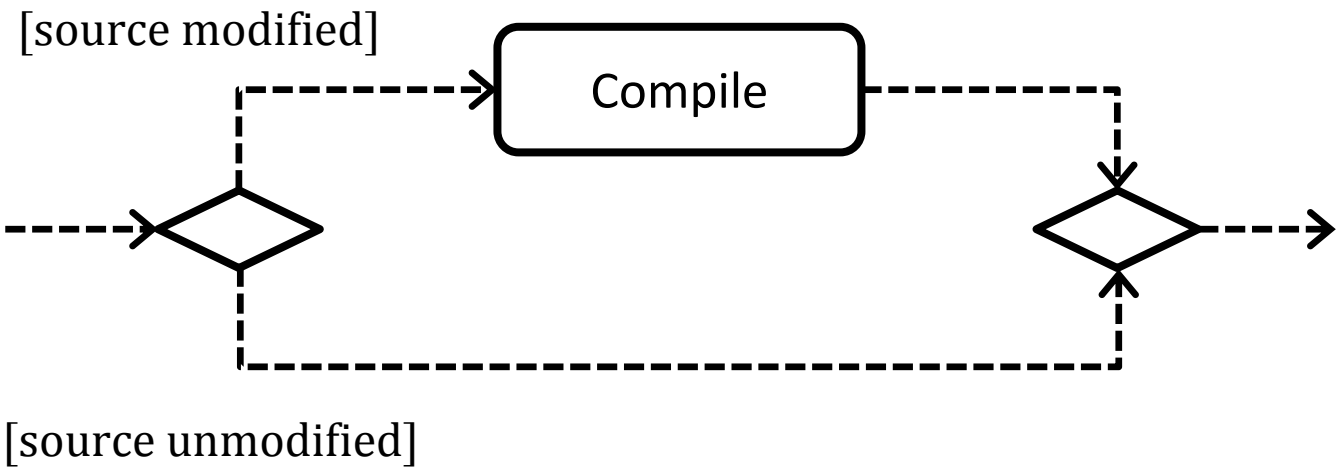
# Őrfeltételek, elágazás




- Szemantika:
  - Csak az egyik ág hajtódik végre
  - Nemdeterminizmus lehetséges
    - Átlapolódó őrfeltételek
    - Vagy egyszerűen őrfeltételek nélkül

# Definíció: Vezérlési elem

A vezérlési elem olyan csomópont a folyamatban, mely a folyamatmodell tevékenységei közül választ ki egyet vagy többet végrehajtásra.



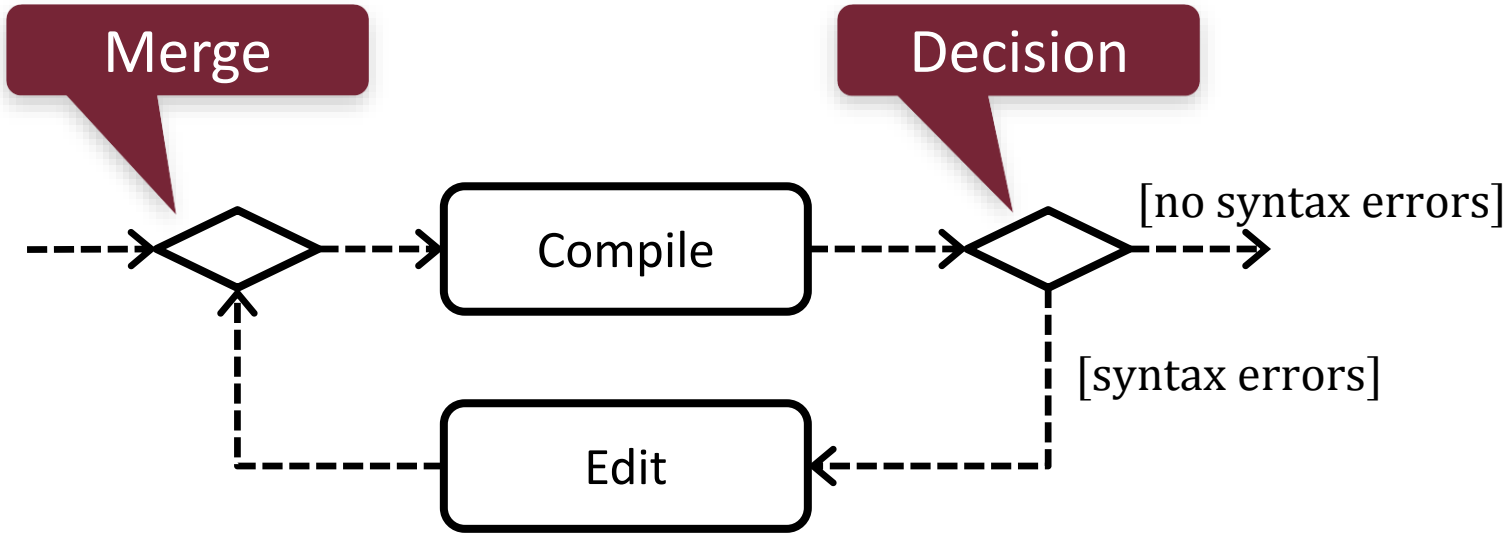
# Definíció: Elágazás

Az **elágazás** olyan vezérlési szerkezet, mely

- áll egy **Döntés (Decision)** és egy **Egyesítés (Merge)** vezérlési elemből, ahol
- a döntési csomópontnak van legalább kettő **kimenete**, melyek közül a kimenetekhez tartozó **őrfeltételek** kiértékelése alapján választunk (oda kerül a vezérlési token),
- a kiválasztott kiement (döntési ág) tetszőleges számú elemet tartalmazhat,
- az összes döntési ág az Egyesítés csompontba fut be.

- Itt az elágazást mindig kizáró döntés (XOR kapu) értelemben használjuk, vagyis egy kiértékelés során csak az elágazás (Decision) egyik kimenete lehet kiválasztva (aktív).
- Egy elágazás lehet többszörös vagy bináris, a tárgyban alapvetően bináris (két kimenetű) elágazást használunk

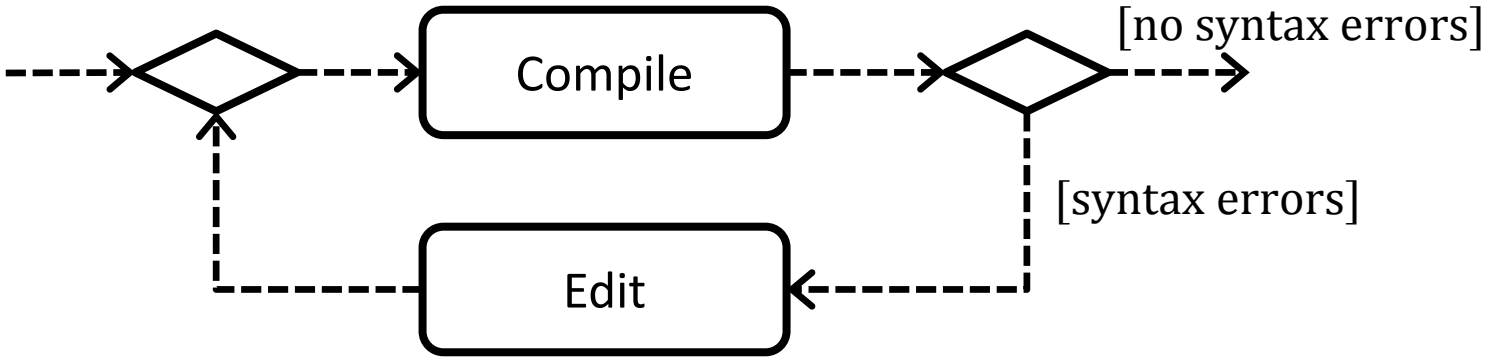
# Ciklus


Merge
Decision
Compile
[no syntax errors]
[syntax errors]
Edit

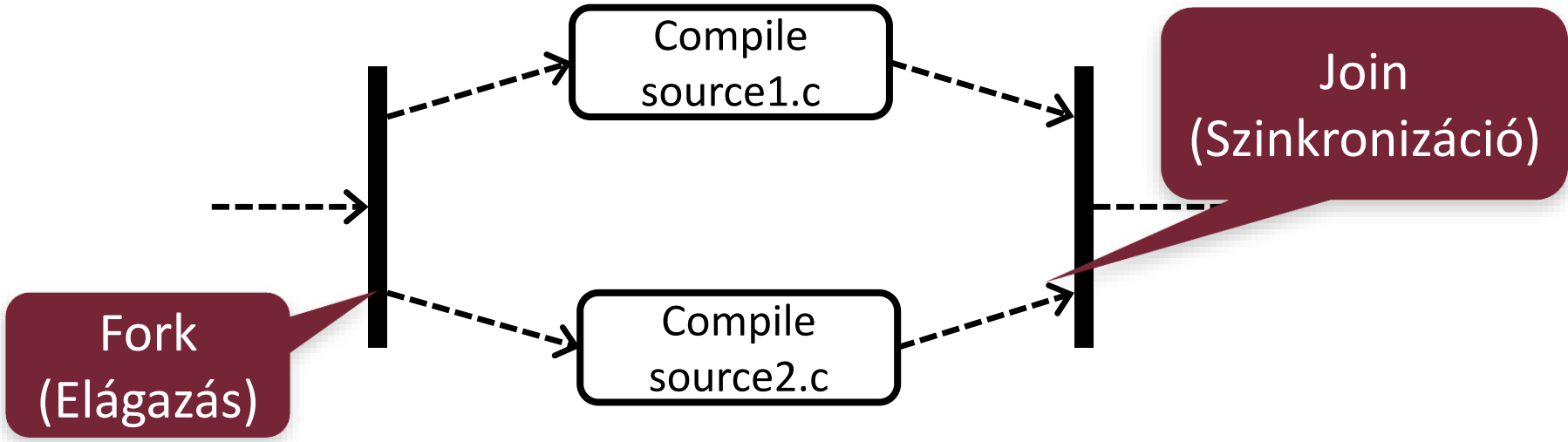
# Definíció: ciklus

A **ciklus** olyan vezérlési szerkezet, mely többszörös végrehajtást definiál. A ciklus

- áll egy **Egyesítés (Merge)** és egy azt *szekvenciálisan követő* **Döntés (Decision)** vezérlési elemből, ahol
- a döntési csomópont egyik kimenete (döntési ága) az egyesítés (merge) vezérlési csomópontba fut vissza.



# Fork / Join




- Szemantika:
  - Nem meghatározott végrehajtási sorrend
  - Párhuzamos / átlapolt végrehajtás is lehet
- Lásd: SzGArch tárgy

# Definíció: párhuzamos végrehajtás

A **párhuzamos végrehajtás**

- áll egy **Párhuzamos elágazás (Fork)** és egy **Összeillesztés/ Szinkronizáció (Join)** vezérlési elemből, ahol
- az elágazásnak tetszőleges számú kimenete lehet (ágak),
- az egyes ágak egymással **konkurrens** módon hajthatóak végre,
- az összes ág a szinkronizáció csomópontba fut be,
- a párhuzamos végrehajtás akkor fejeződik be, ha az összes ág végrehajtása befejeződött.

Két tevékenység **konkurrens**, ha végrehajtási sorrendjükre nézve nincs megkötés.

- Megj: tipikusan kettő párhuzamos ággal fogunk dolgozni.
- **NEM azonos a döntési elágazással!**

# Flow begin / flow end

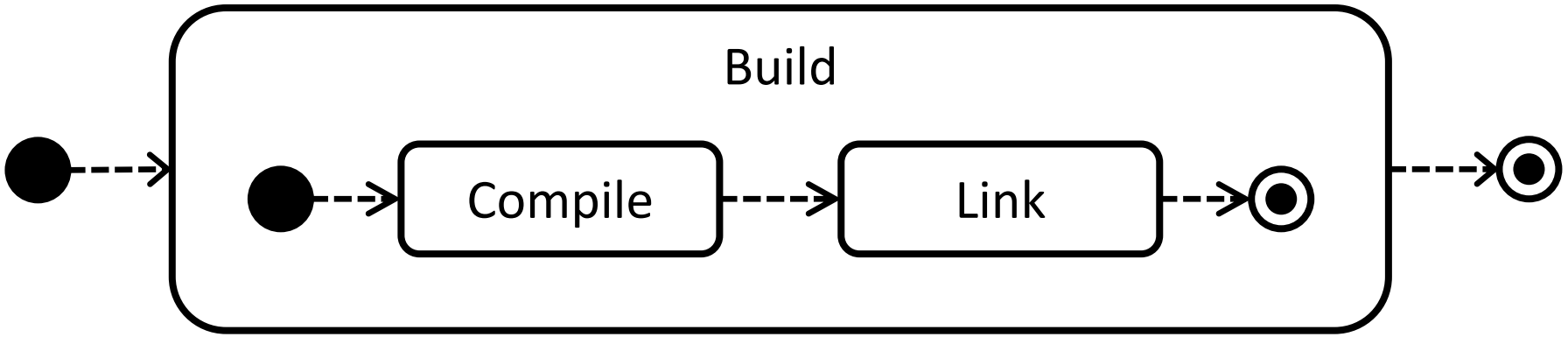
# Definíció: folyamat indítása/befejezése

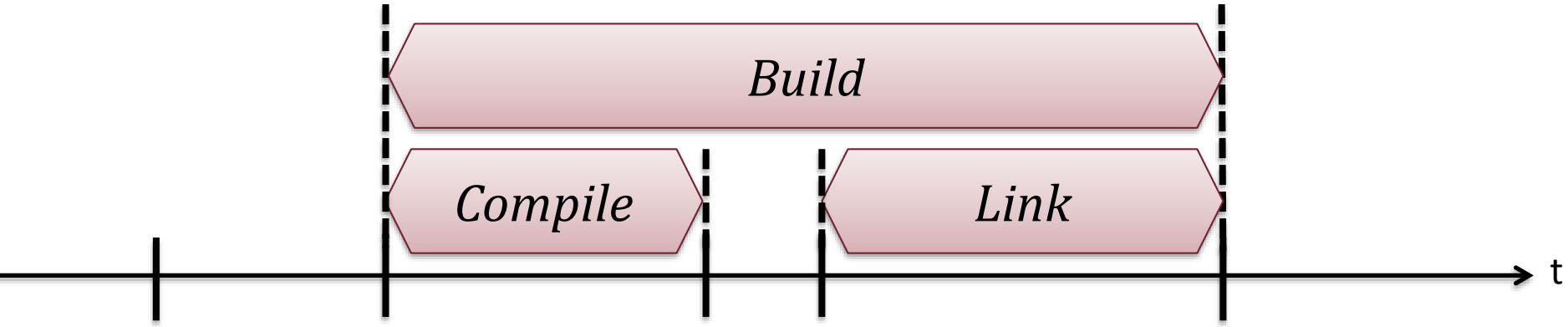
Minden folyamat egy indítási (Flow Begin) vezérlési elemmel indul, és egy befejezési (Flow End) elemmel fejeződik be.

- Az **indítási csomópont** a csomópont legelső eleme, melynek pontosan egy kimenete van.
- A **befejezési csomópont** a folyamat utolsó eleme, melynek pontosan egy bemenete van.

- Megj: itt nem modellezzük külön, minek a hatására indul el a folyamat.

# Hierarchia



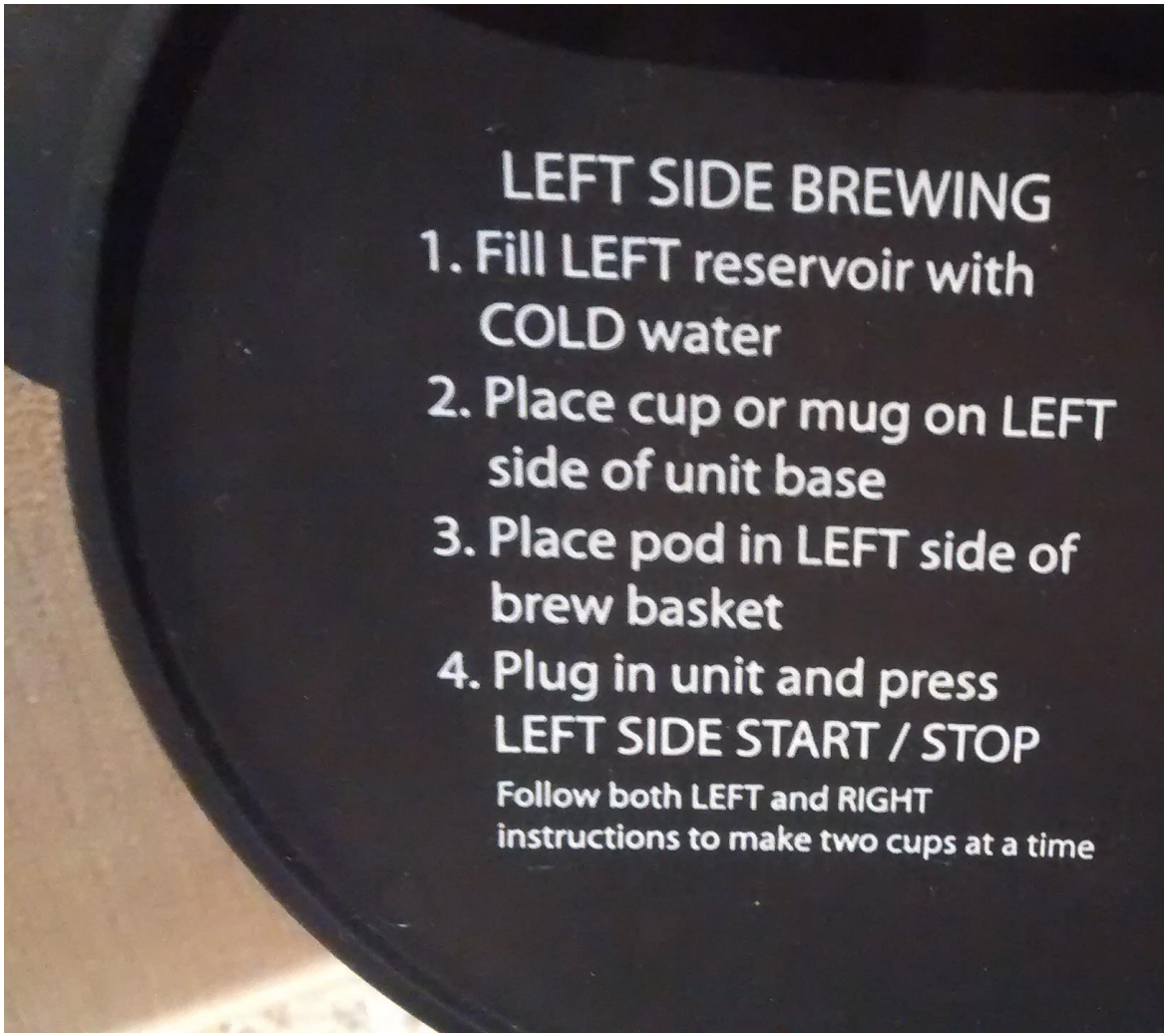



# Definíció: hierarchia

Hierarchikus folyamatmodell:

- Elemi tevékenység helyett tartalmaz folyamatmodellel leírt részmodellt (hierarchikus finomítás)

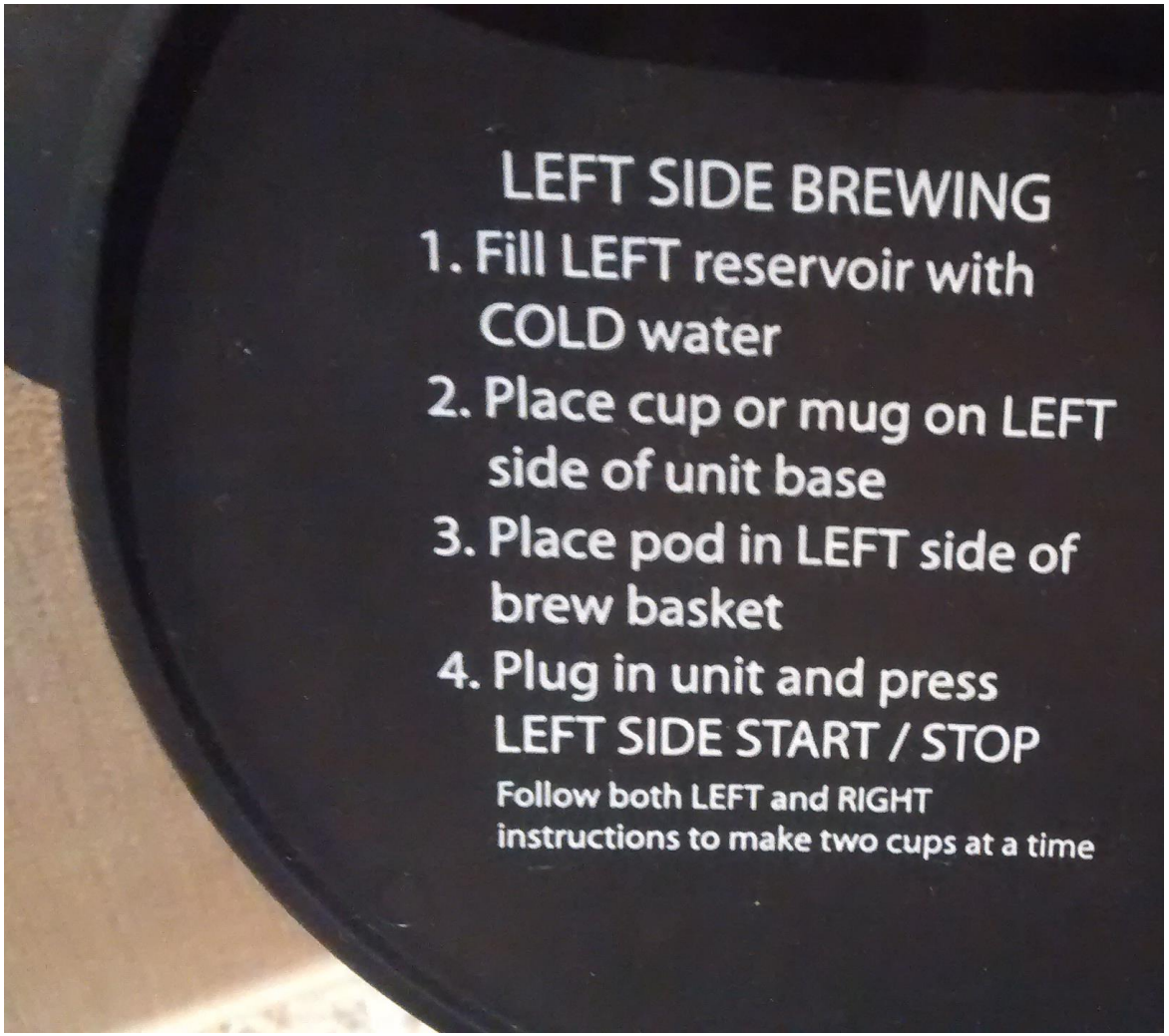
# Példa: kávéfőzés

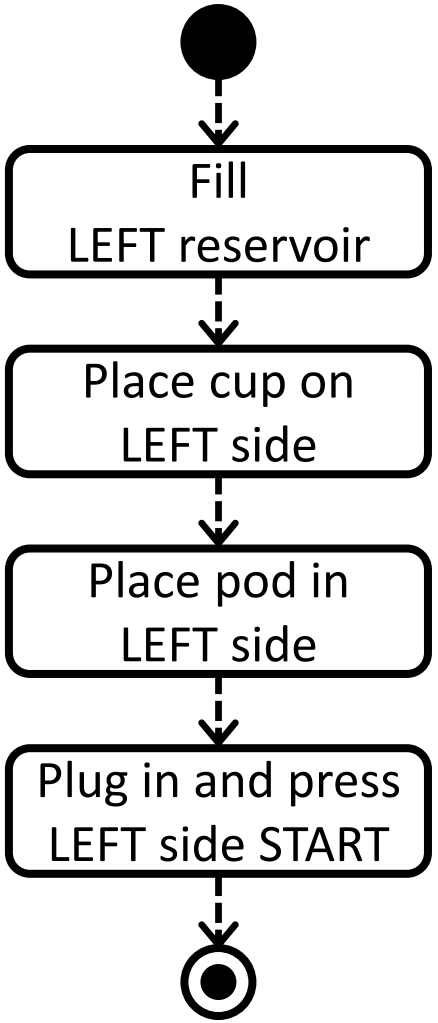



1. Töltse meg a BAL tartályt hideg vízzel
2. Tegyen egy bögrét vagy poharat a BAL pohártartóra
3. Tegyen be egy kávépárnát a BAL oldali tartóba
4. Dugja be a kábelt és nyomja meg a BAL OLDAL START/STOP gombot

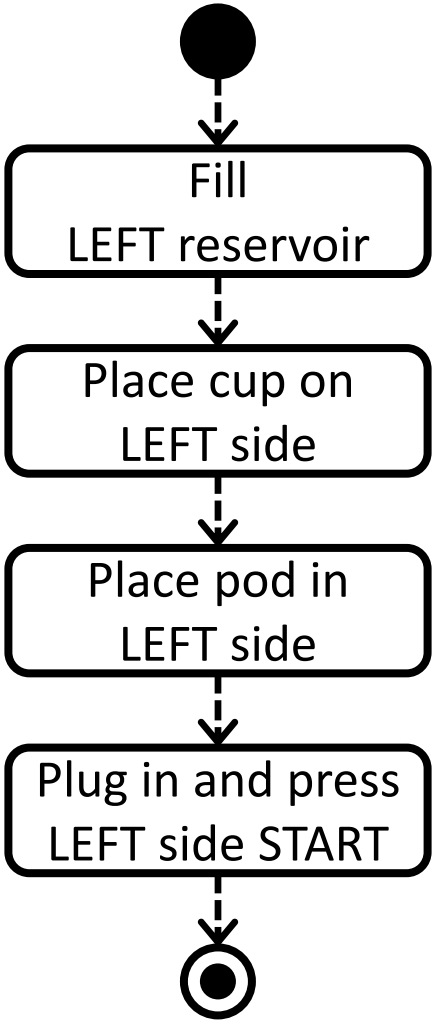
Ha egyszerre két kávét akar főzni, egyszerre kövesse a BAL és JOBB utasításokat

# Példa: kávéfőzés

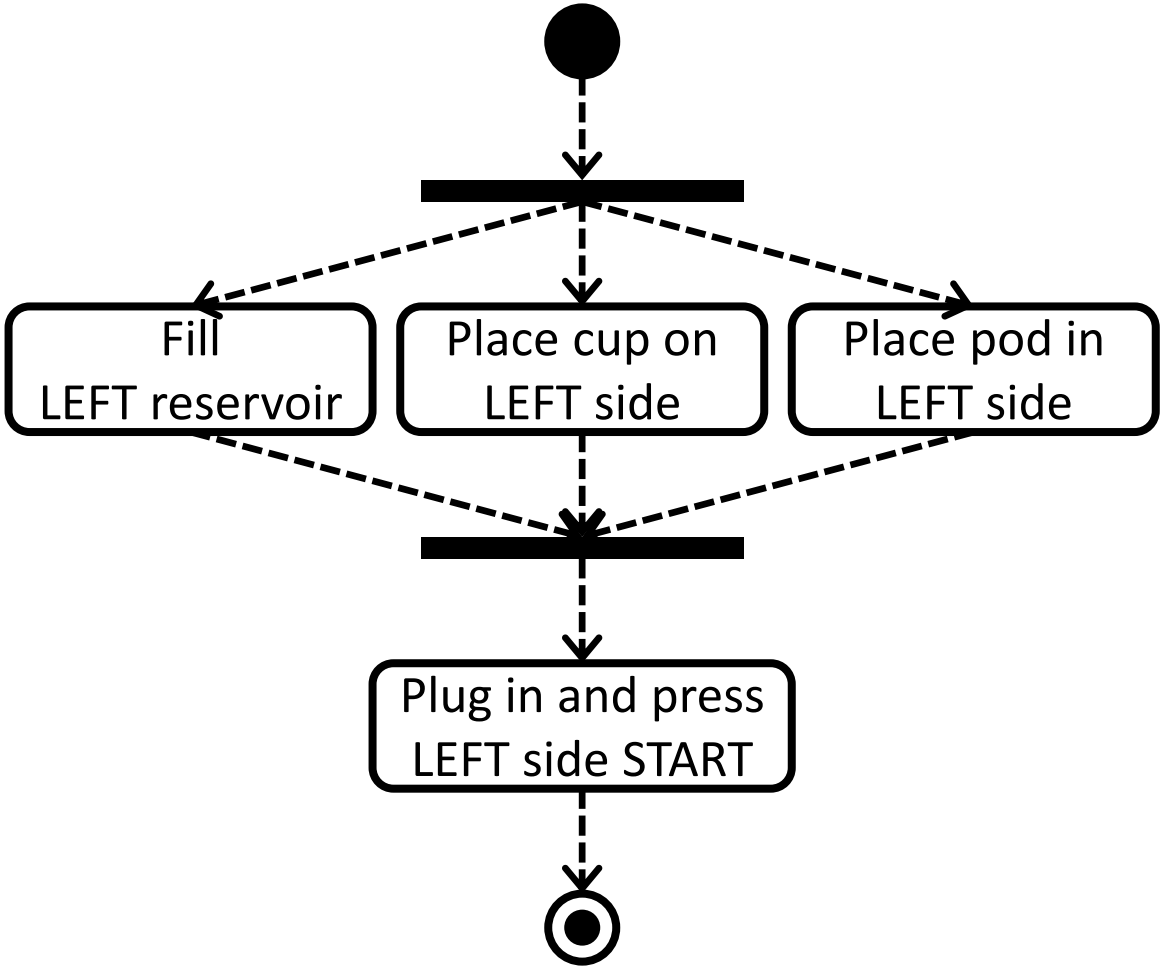


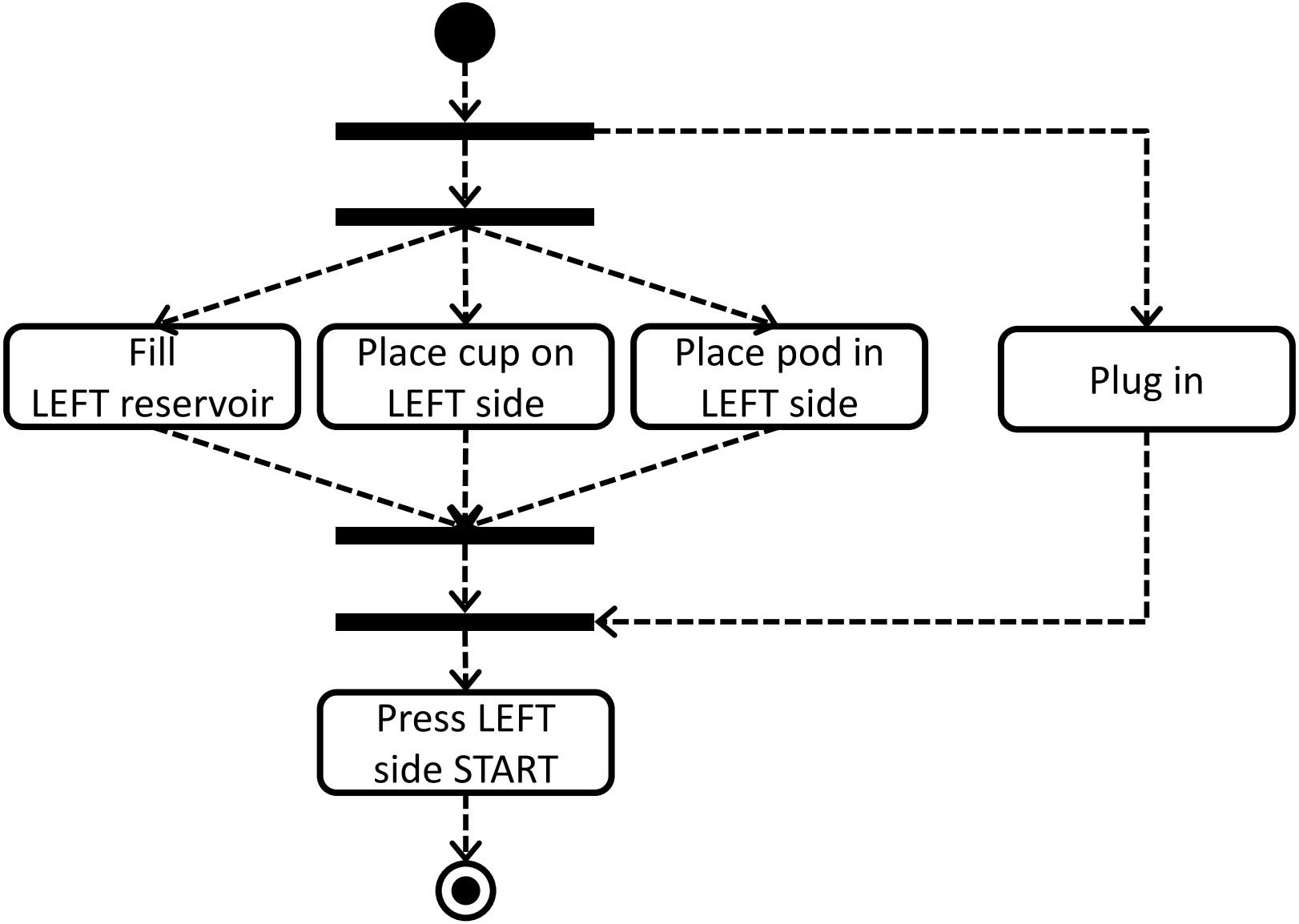
# Kávéfőzés


Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Plug in and press
LEFT side START

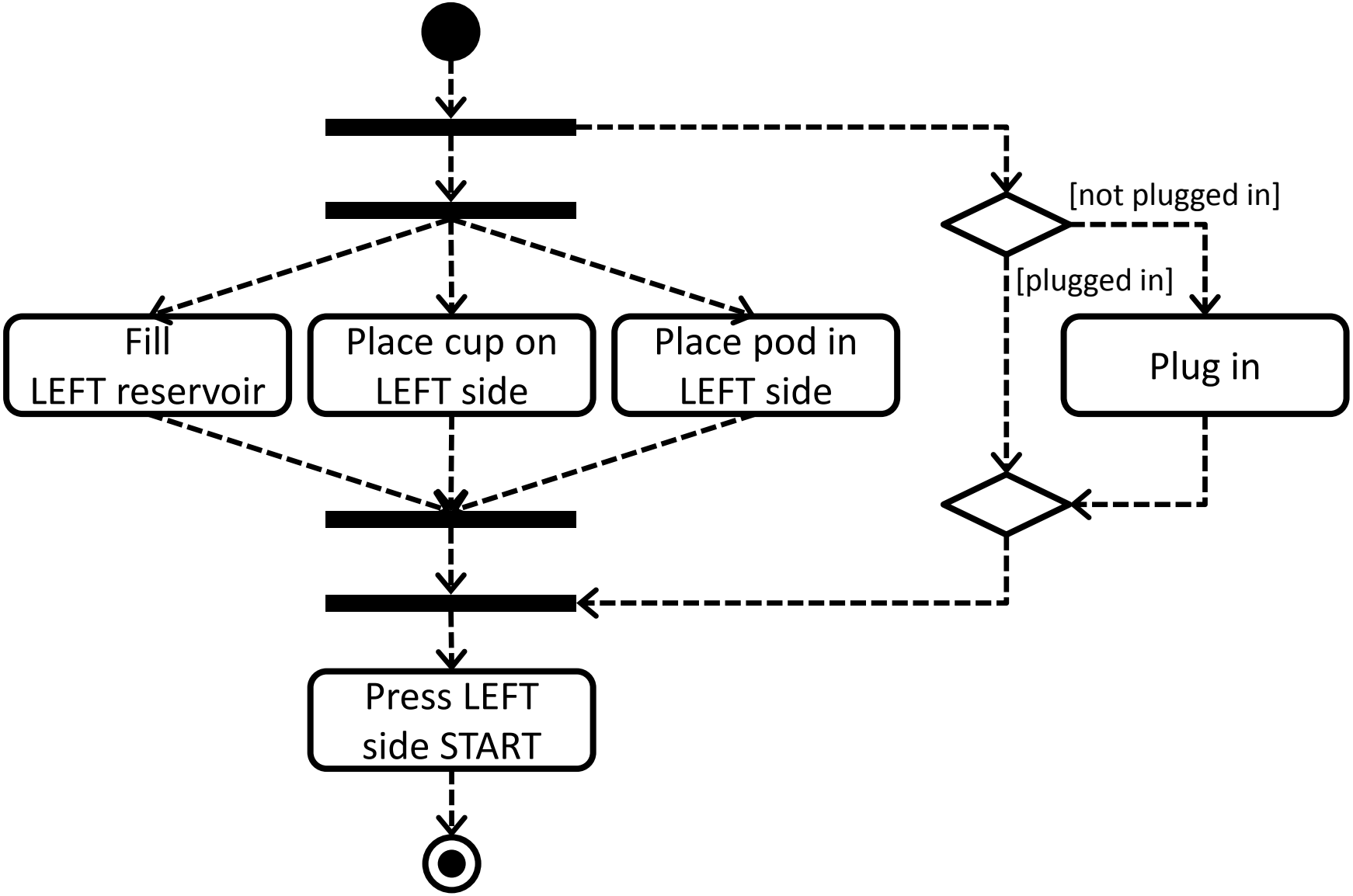
# Kávéfőzés


Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Plug in and press
LEFT side START

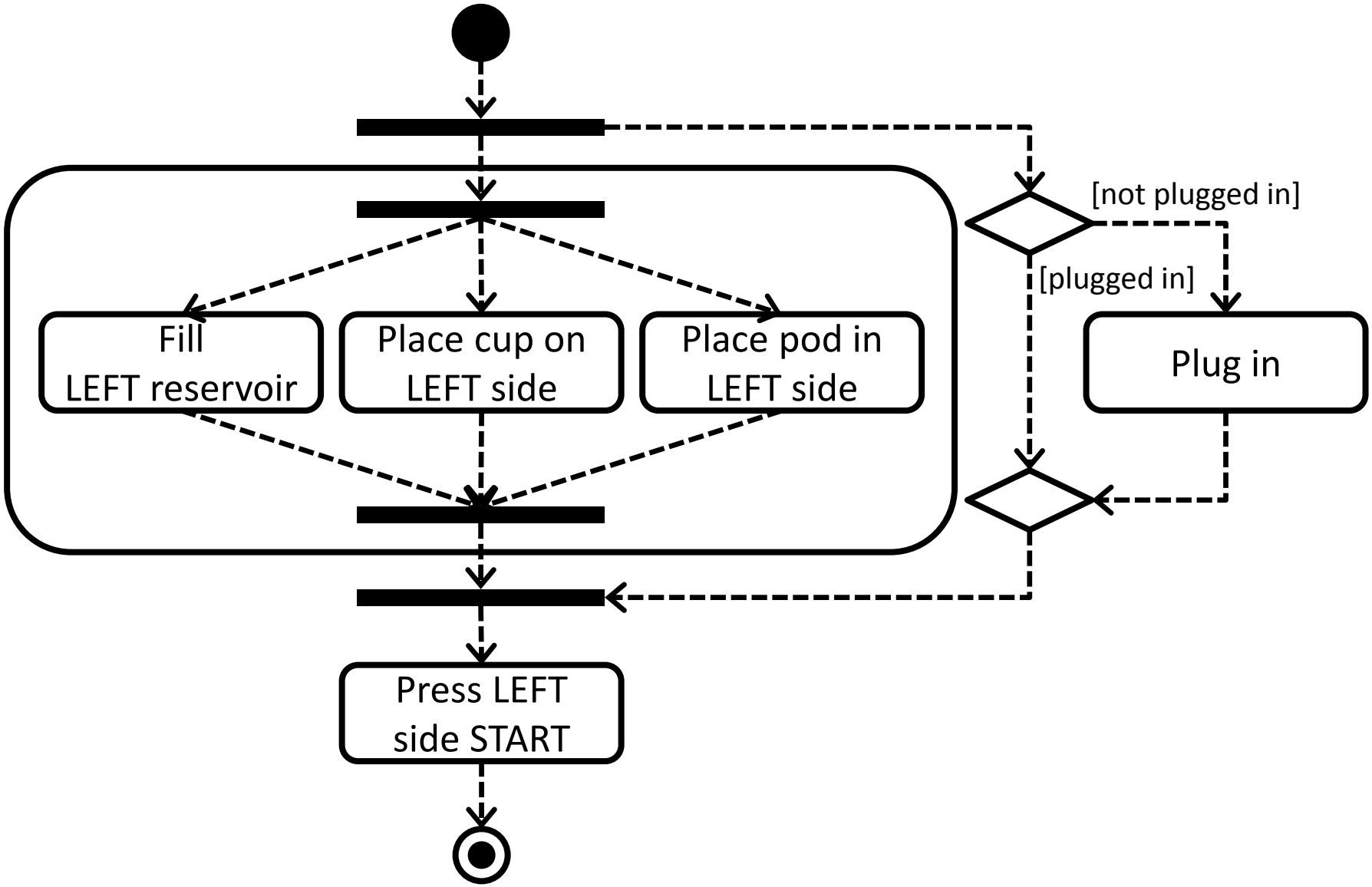
# Kávéfőzés


Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Plug in
Press LEFT
side START

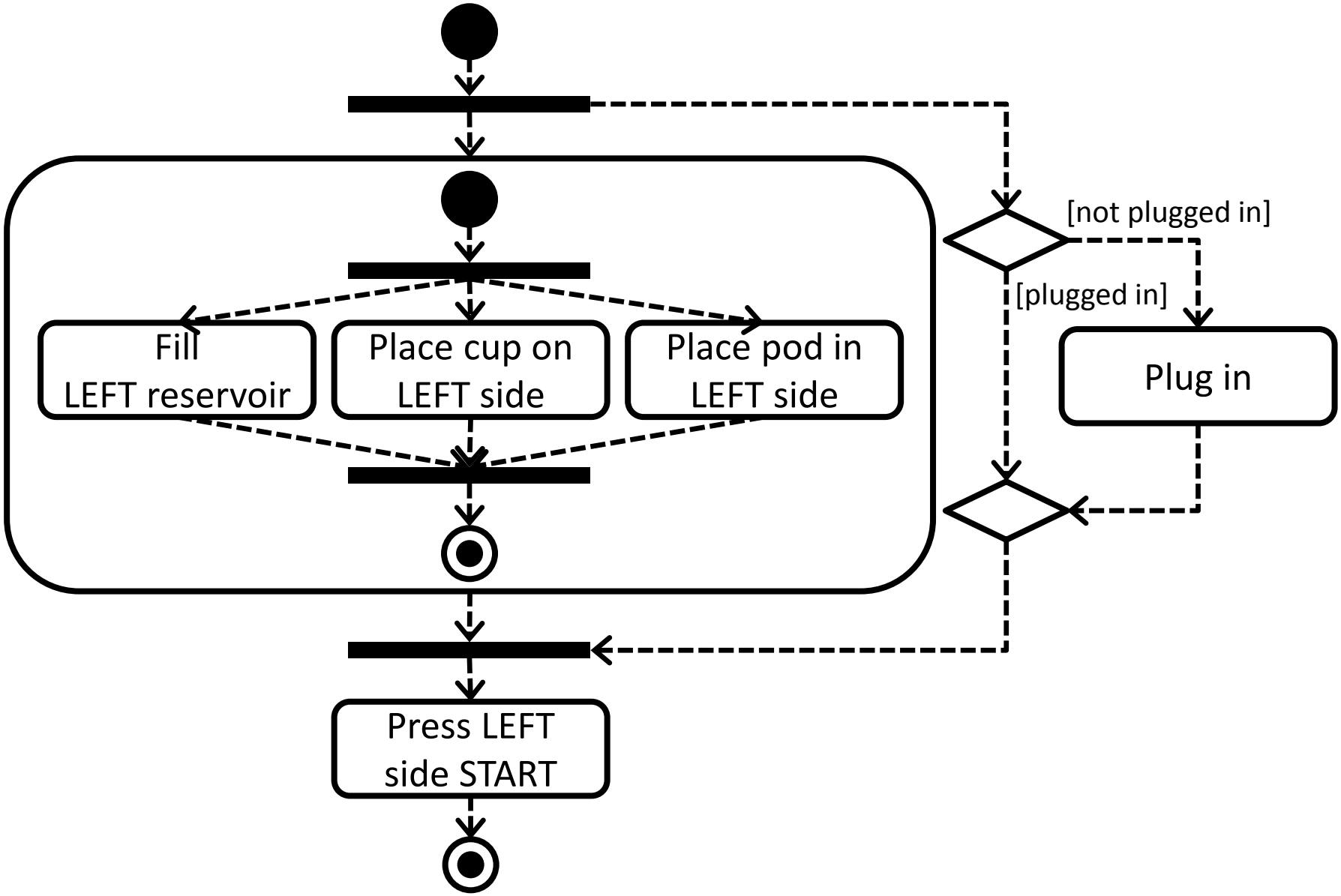
# Kávéfőzés


[not plugged in]
[plugged in]
Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Plug in
Press LEFT
side START

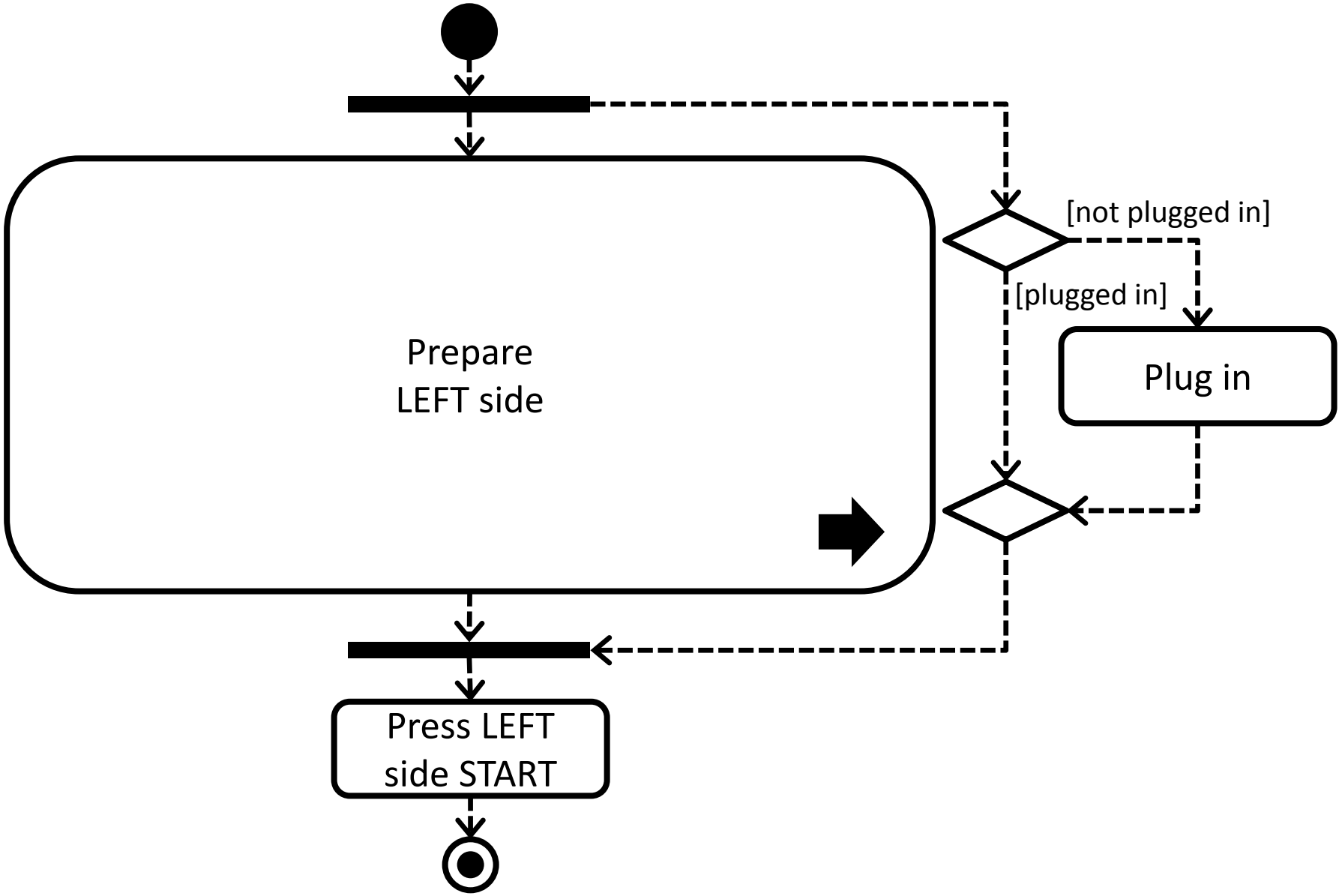
# Kávéfőzés


[not plugged in]
[plugged in]
Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Plug in
Press LEFT
side START

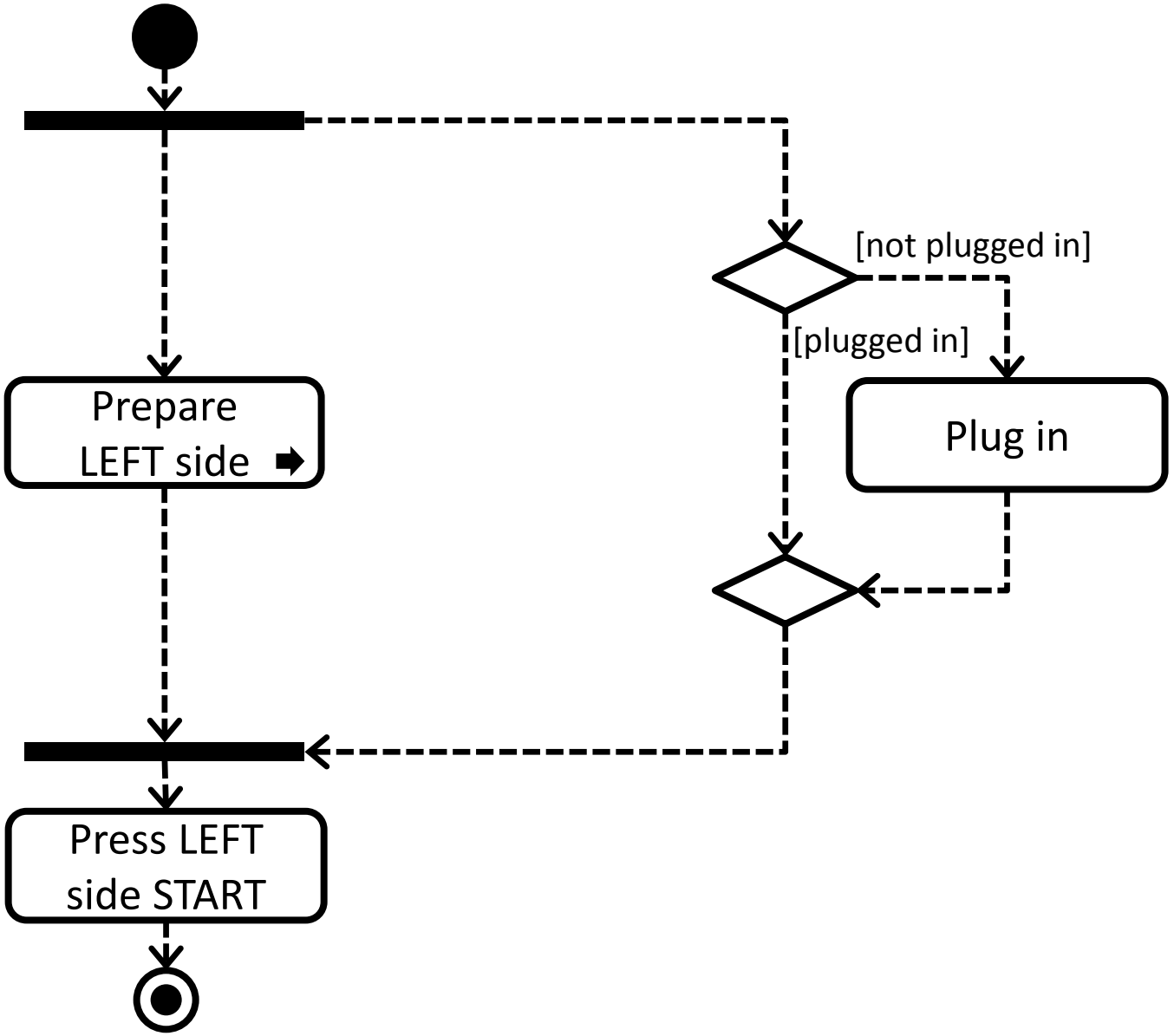
# Kávéfőzés


[not plugged in]
[plugged in]
Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Plug in
Press LEFT
side START

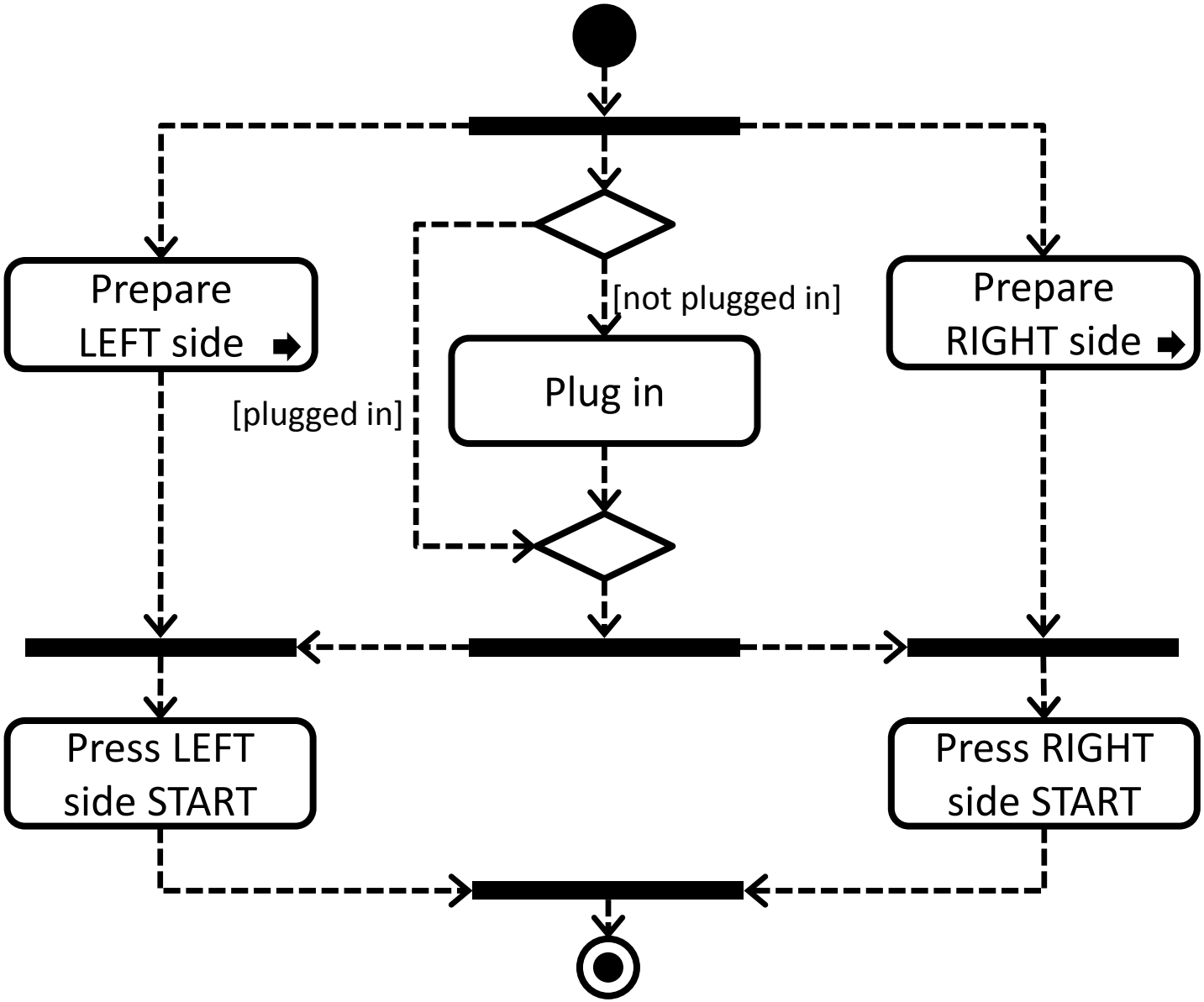
# Kávéfőzés


[not plugged in]
[plugged in]
Prepare
LEFT side
Plug in
Press LEFT
side START

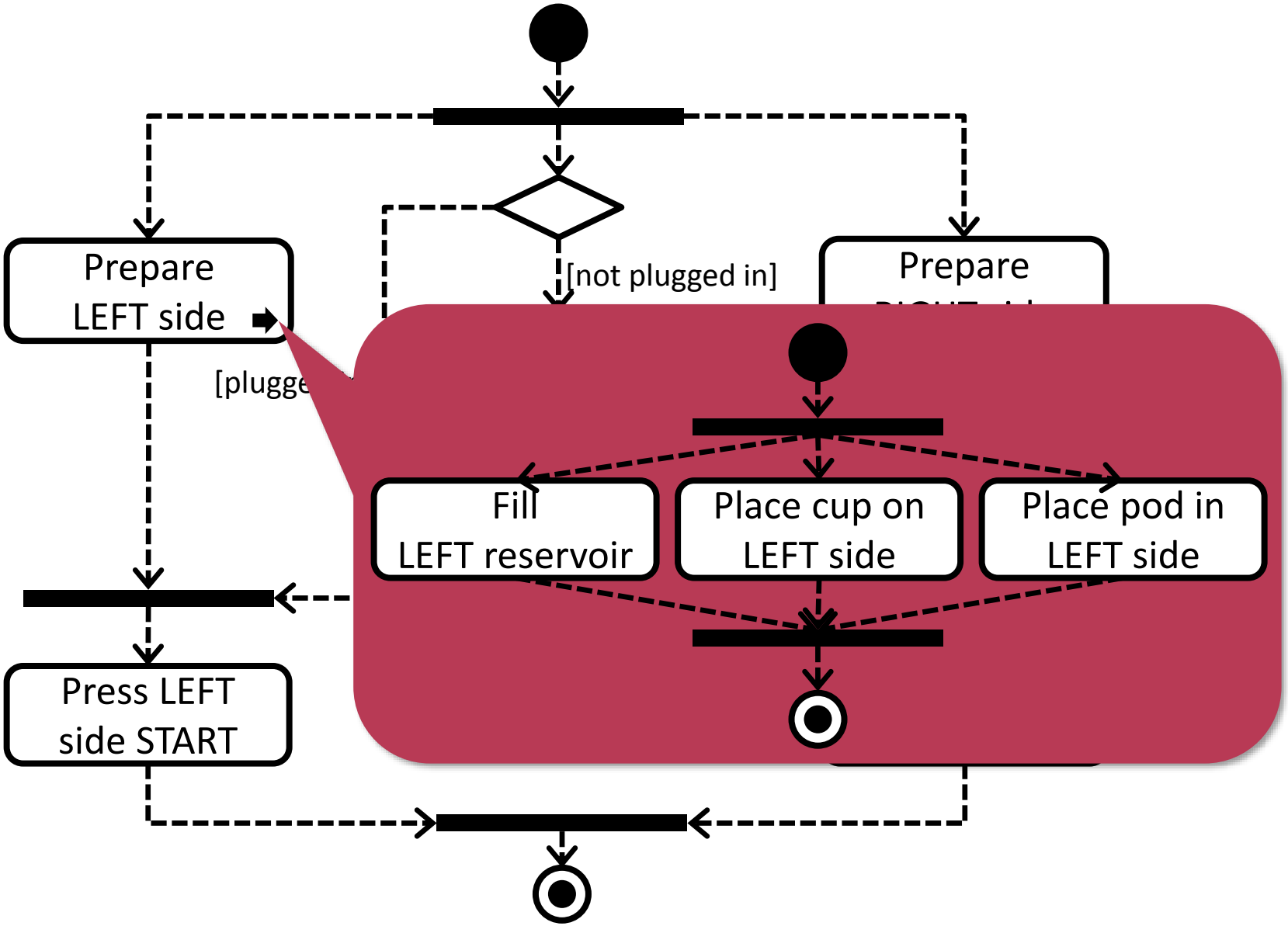
# Kávéfőzés


[not plugged in]
[plugged in]
Prepare
LEFT side
Plug in
Press LEFT
side START

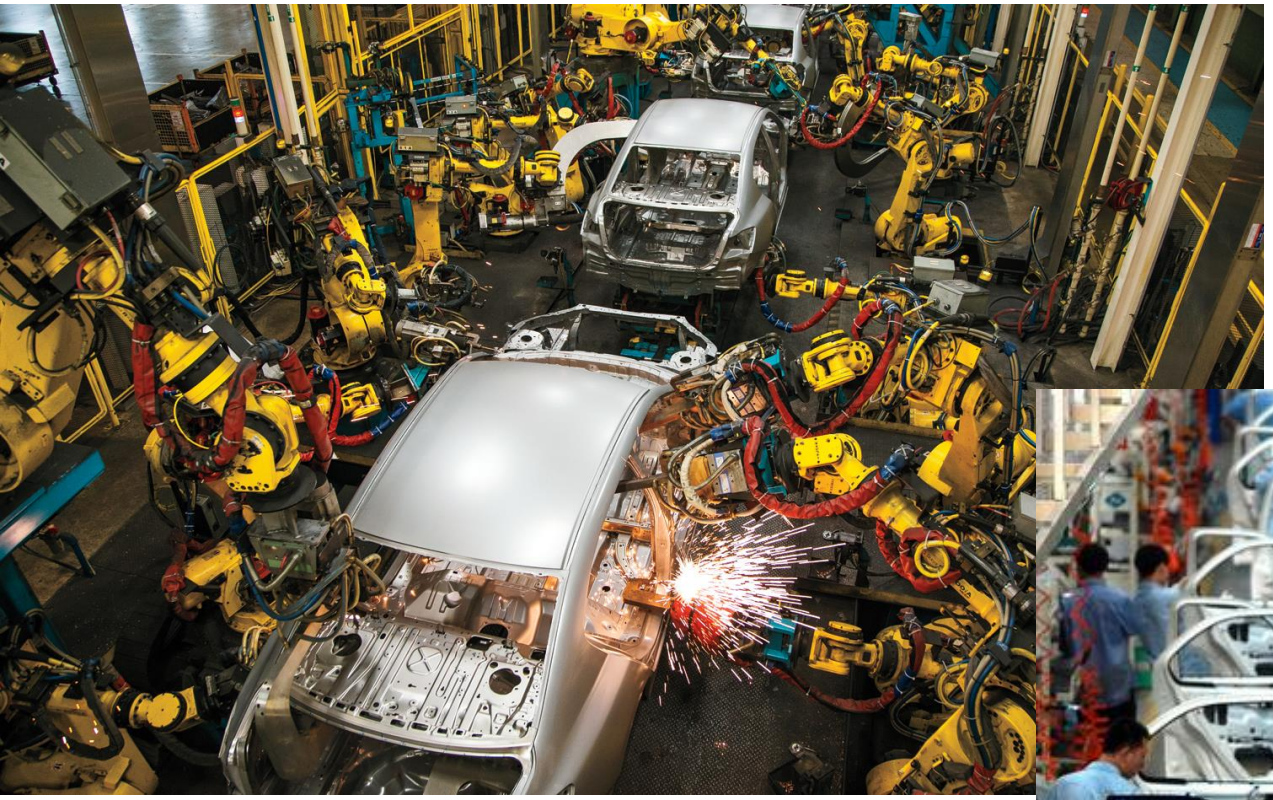
# Kávéfőzés


Prepare
LEFT side
[plugged in]
[not plugged in]
Plug in
Prepare
RIGHT side
Press LEFT
side START
Press RIGHT
side START

# Kávéfőzés

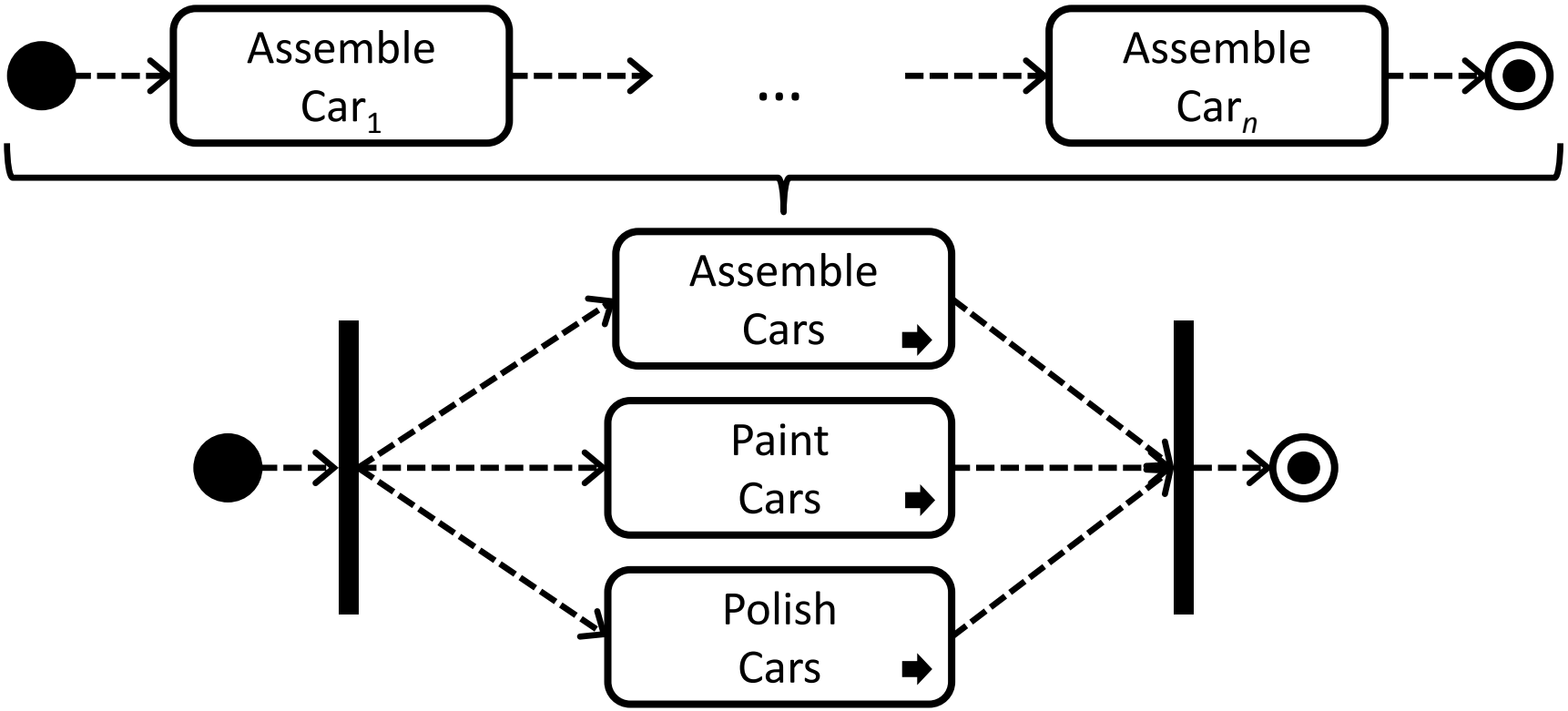

Prepare
LEFT side
[not plugged in]
Prepare
[plugge
Fill
LEFT reservoir
Place cup on
LEFT side
Place pod in
LEFT side
Press LEFT
side START

# Különböző szempontok szerinti modellezés

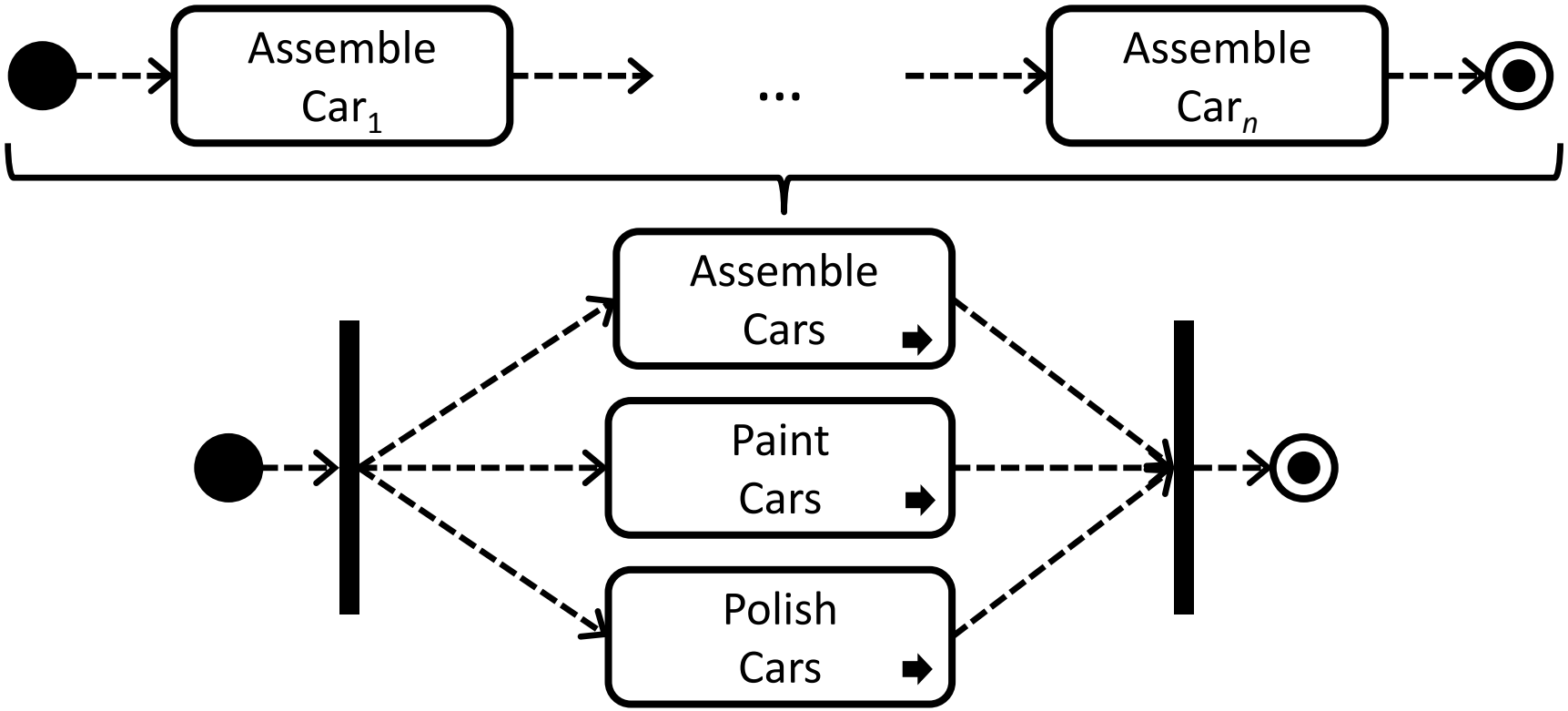


# Mi történik egy autóval?

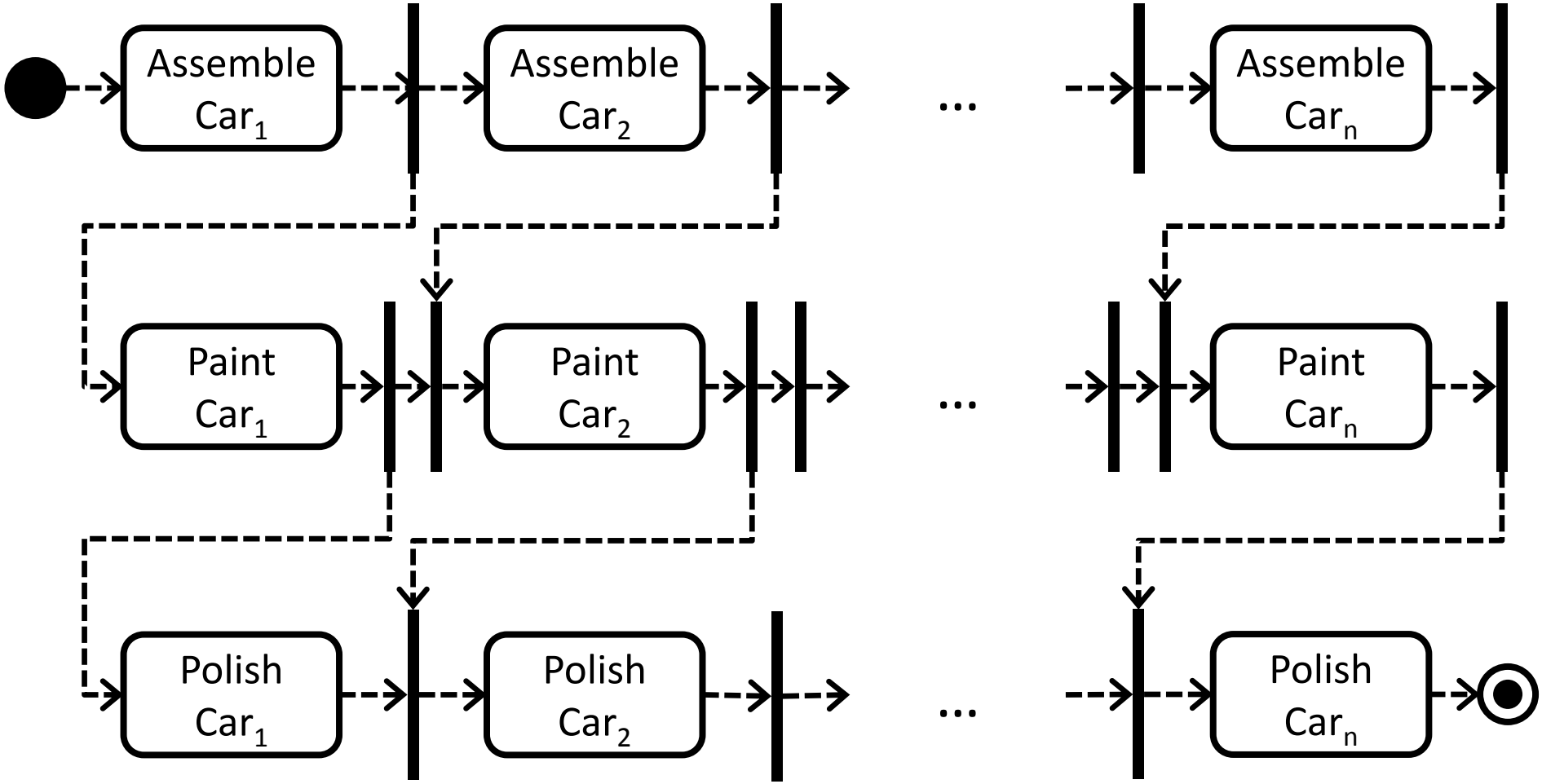
# Mi történik a gyártósoron?



# Különböző szempontok szerinti modellezés

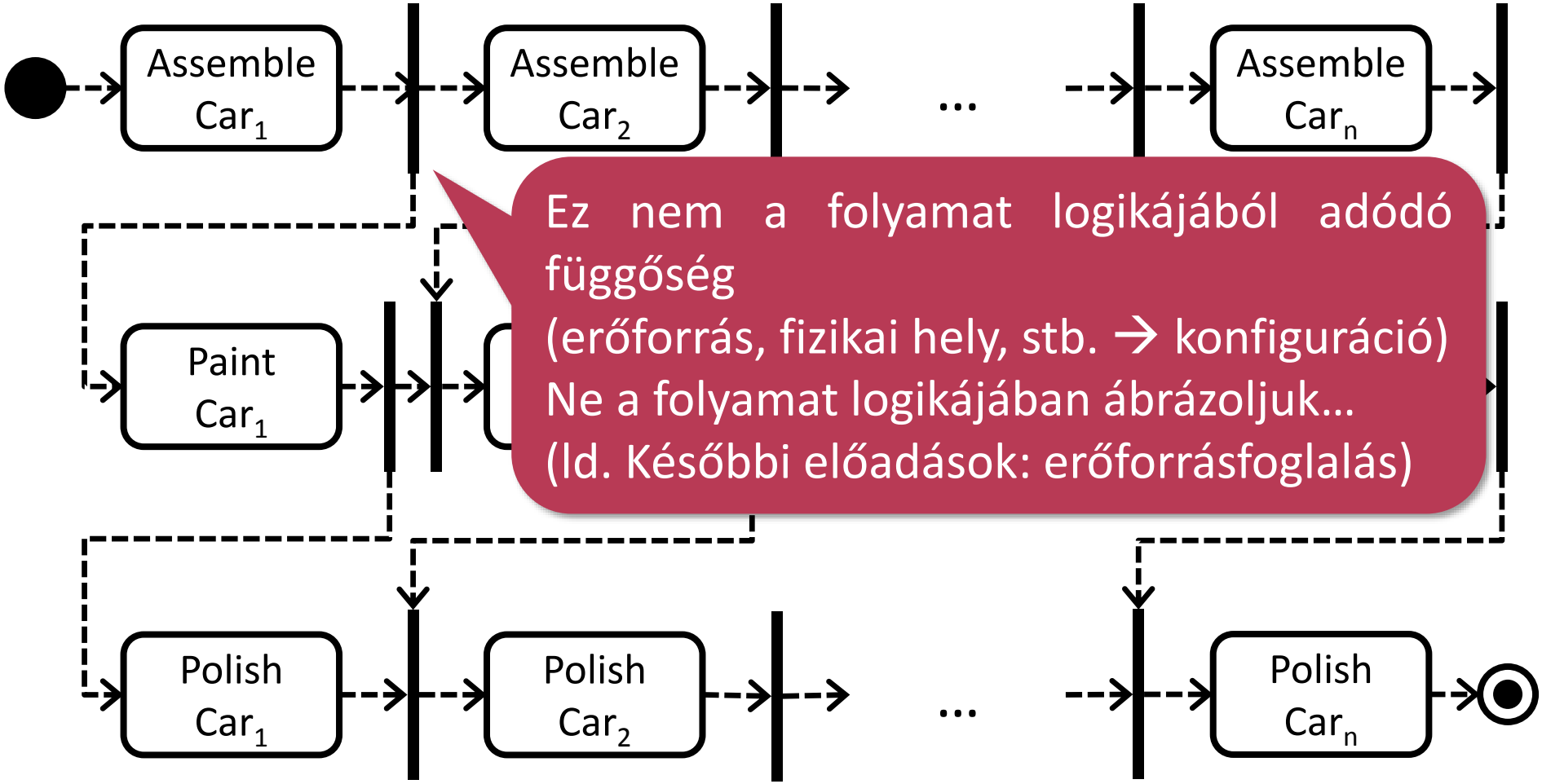



# Együttes nézet




- Mindent tartalmaz, de nem túl praktikus

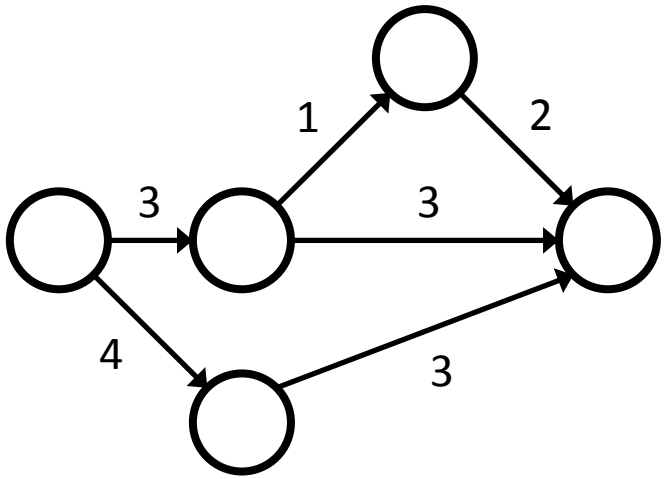
# Együttes nézet



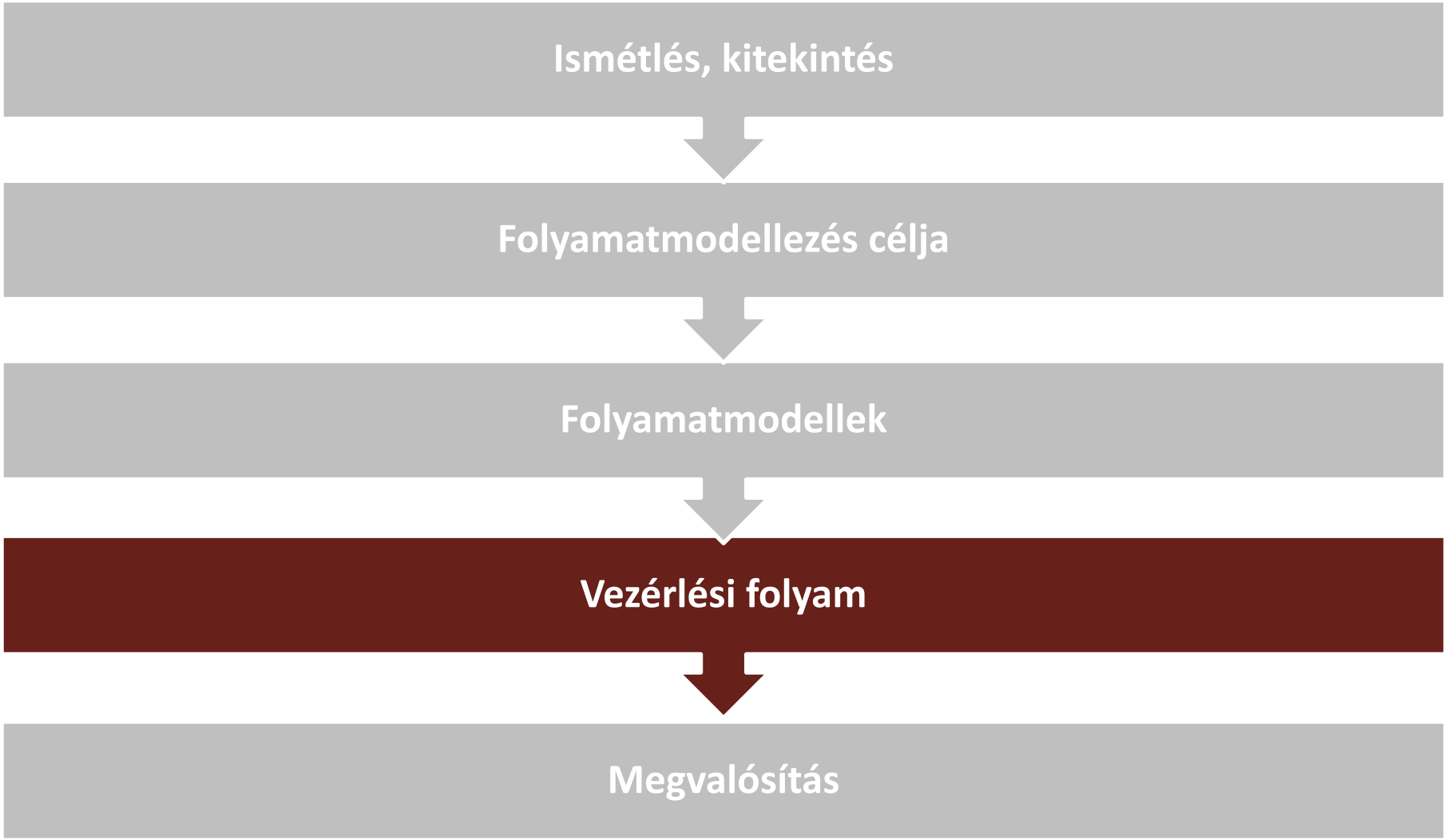

- Mindent tartalmaz, de nem túl praktikus

# Együttes nézet

- A 2D fork-join háló nem túl praktikus
  - Aspektusokra (autó és gép életútja) külön folyamat
- Sok fork-join tömörebben? → PERT chart
  - Program Evaluation and Review Technique
    - Végrehajtási idő analízisére való, ld. BSz 2
  - (Ez viszont elágazást nem tud)

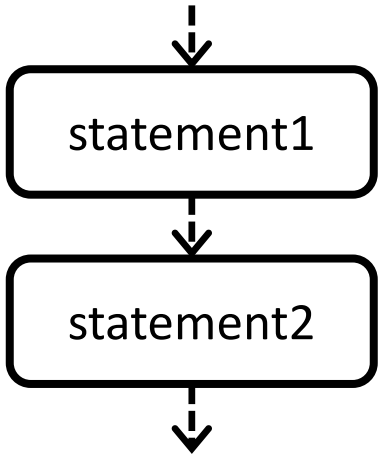


# Tartalom


Ismétlés, kitekintés
Folyamatmodellezés célja
Folyamatmodellek
Vezérlési folyam
Megvalósítás

# Vezérlési folyam

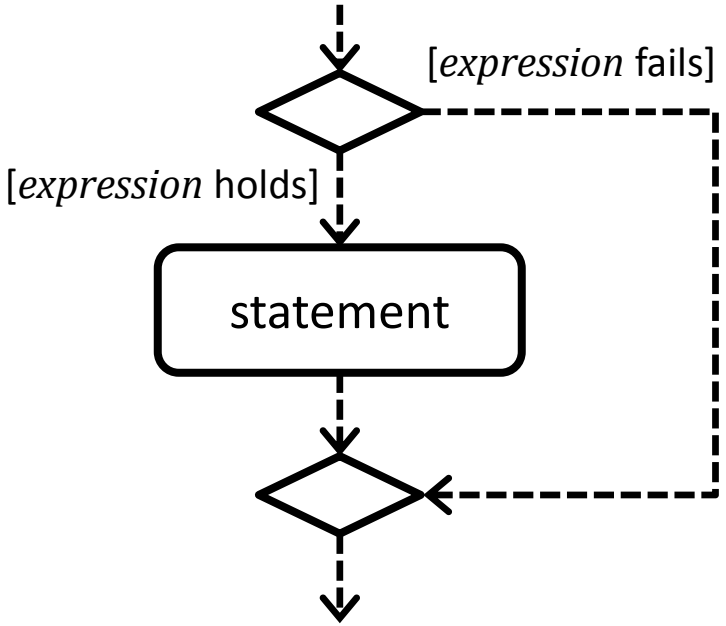
*<statement1>*
*<statement2>*



# Vezérlési folyam

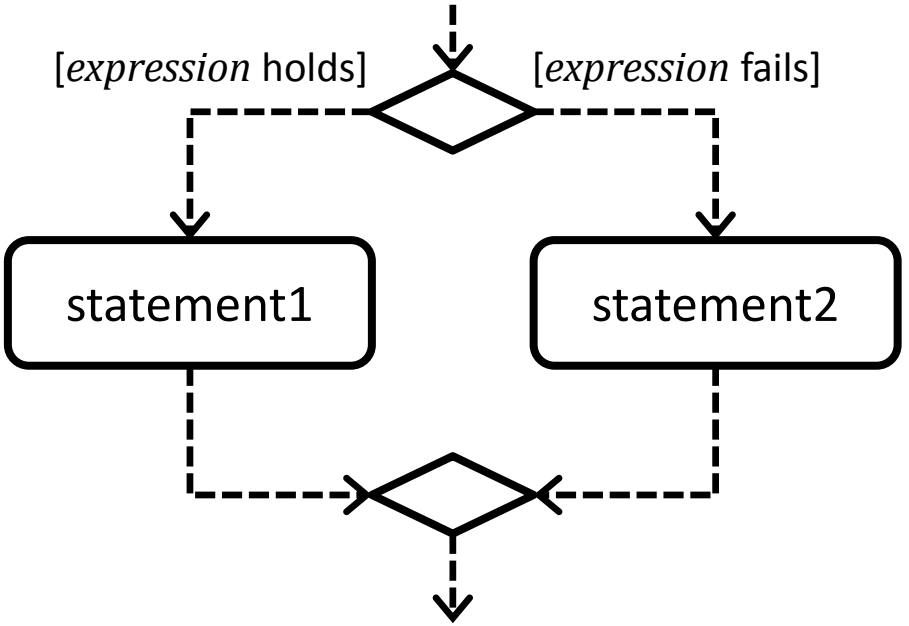
**`if`** (*<expression>*)

*<statement>*



# Vezérlési folyam

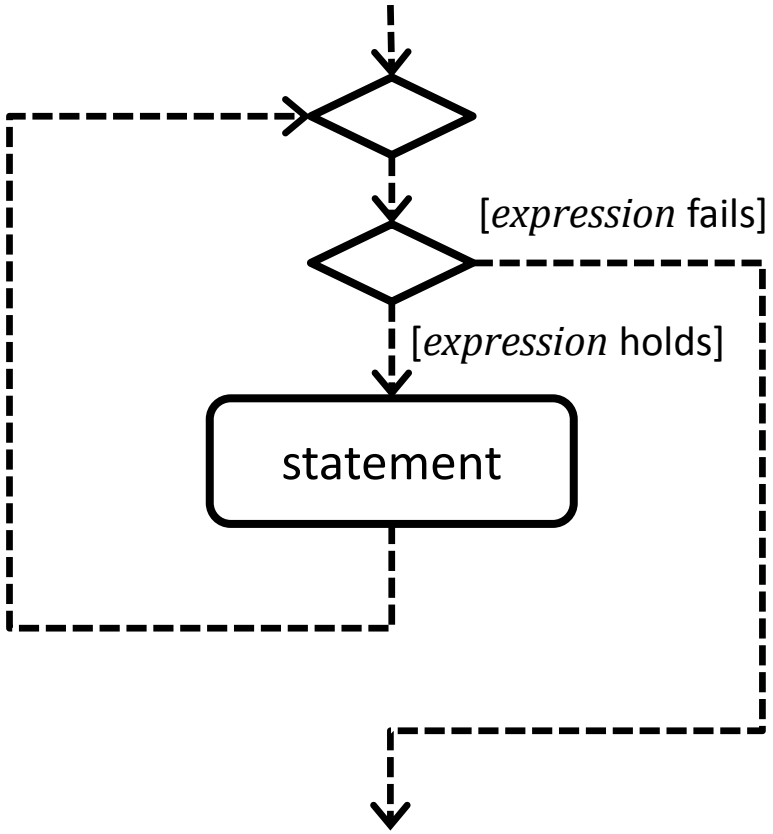
```
if (<expression>)
    <statement1>
else
    <statement2>
```



# Vezérlési folyam

**while** (*<expression>*)

*<statement>*

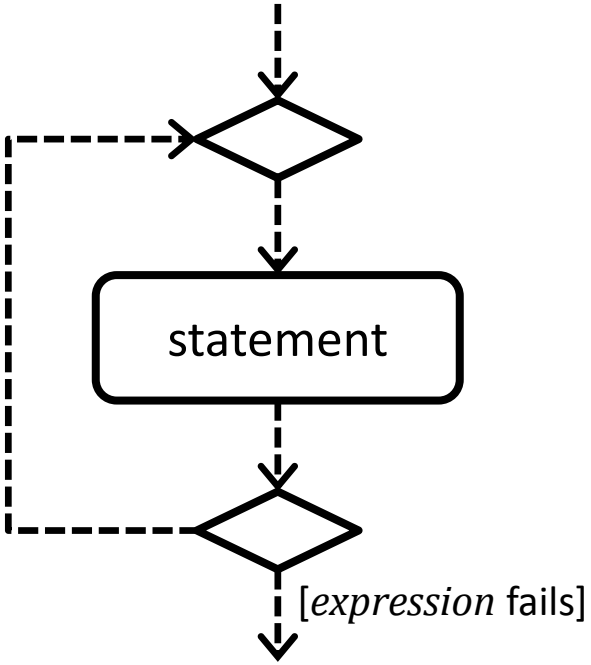


# Vezérlési folyam

**do**

*<statement>*

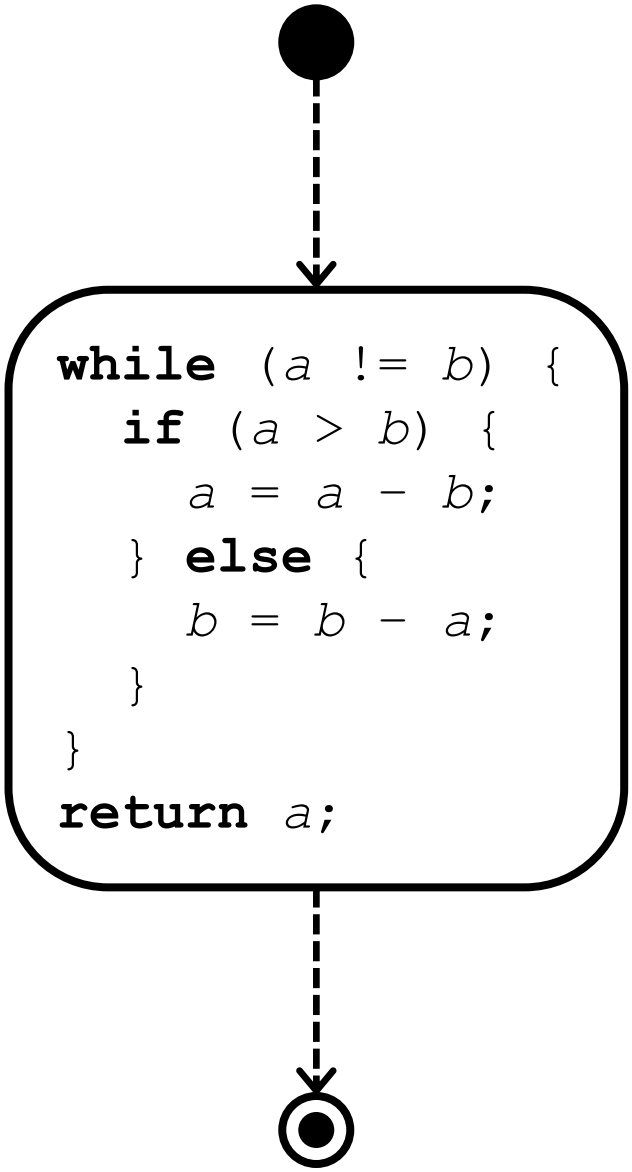
**while** (*<expression>*)



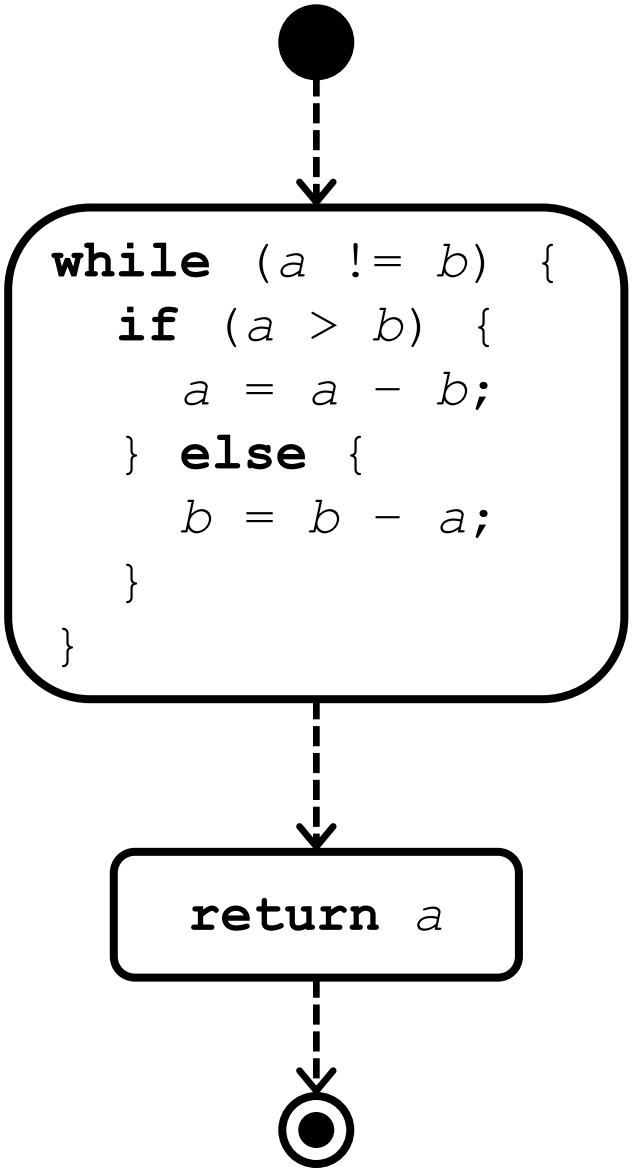
# Vezérlési folyam - példa

```
while (a != b) {
   if (a > b) {
      a = a - b;
   } else {
      b = b - a;
   }
}
return a;
```

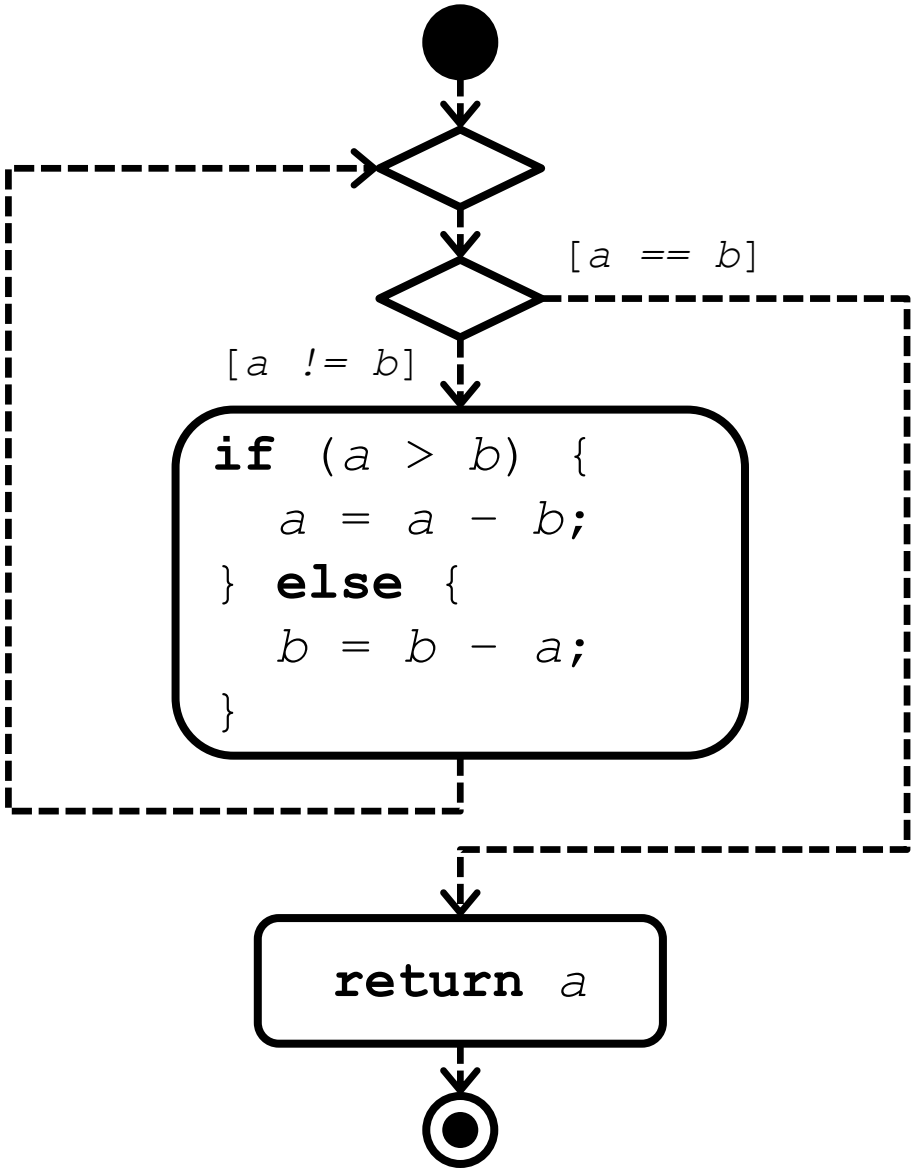
# Vezérlési folyam - példa



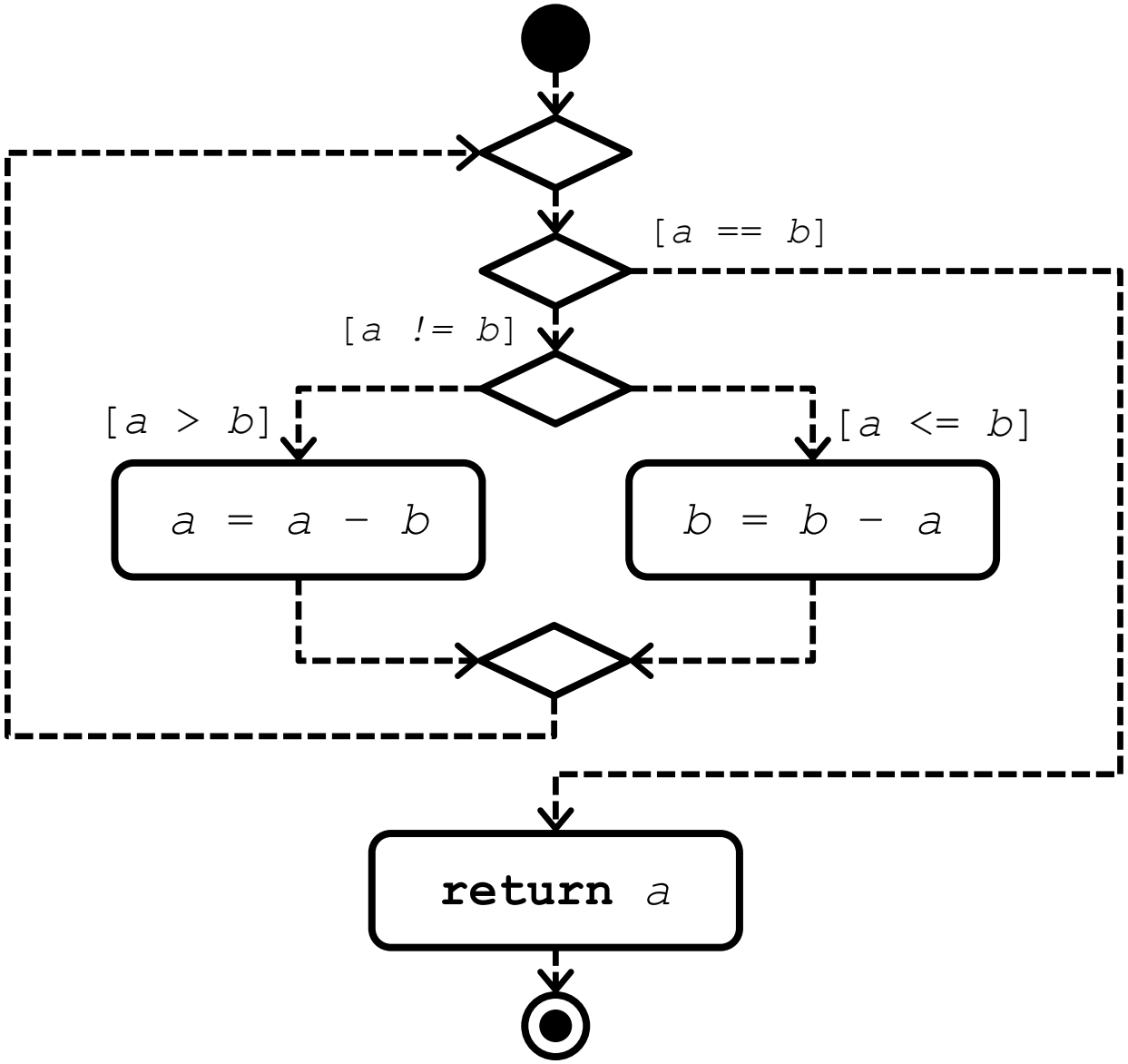
# Vezérlési folyam - példa



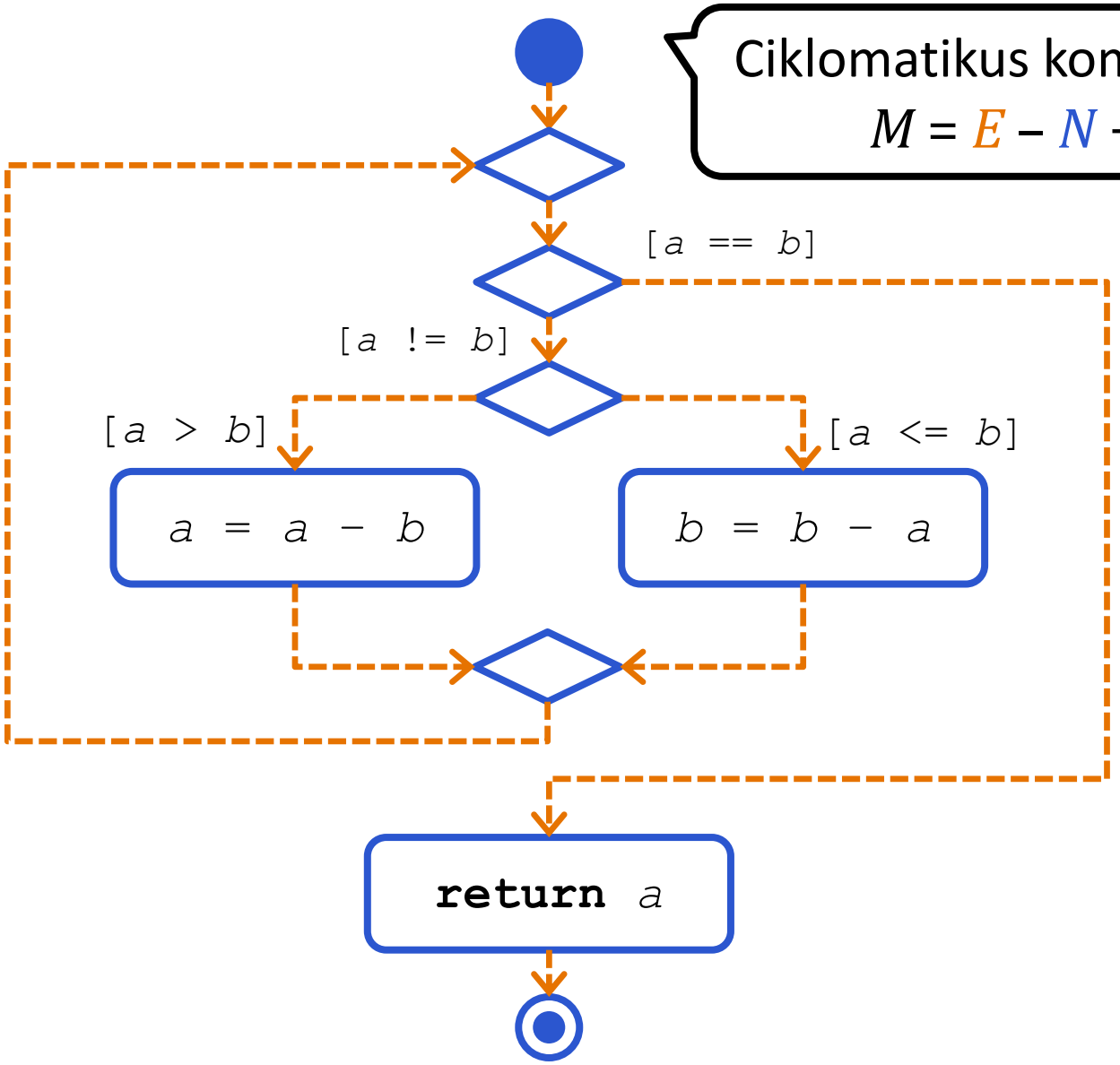
# Vezérlési folyam - példa



# Vezérlési folyam - példa



# Vezérlési folyam - komplexitás



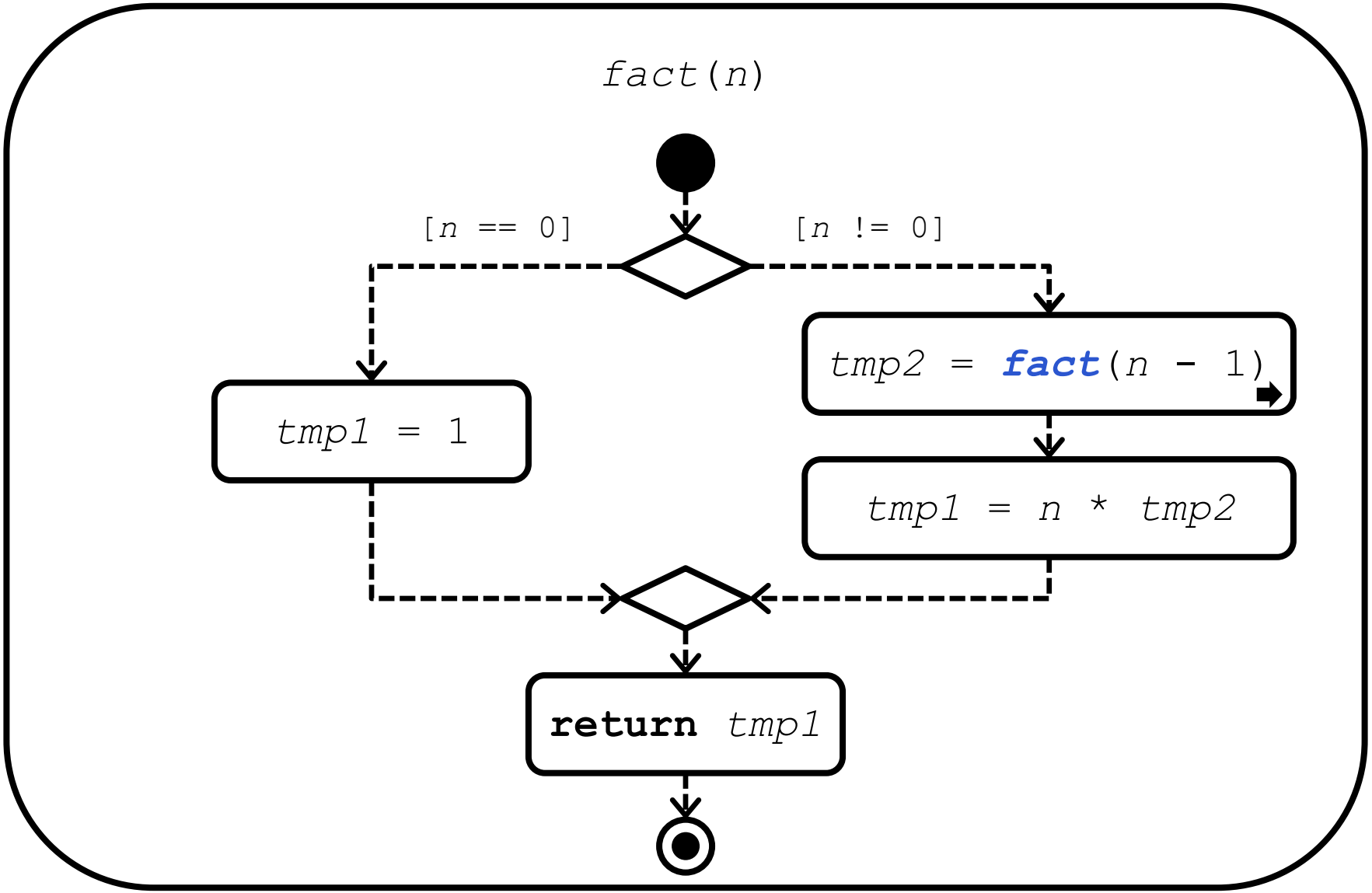
# Vezérlési folyam - rekurzió

```
int fact(int n) {
   return
     (n == 0) ? 1 : n * fact(n - 1);
}
```

# Vezérlési folyam - rekurzió

```
int fact(int n) {
   int tmp1;
   if (n == 0) {
      tmp1 = 1;
   } else {
      int tmp2 = fact(n - 1);
      tmp1 = n * tmp2;
   }
   return tmp1;
}
```

# Vezérlési folyam - rekurzió



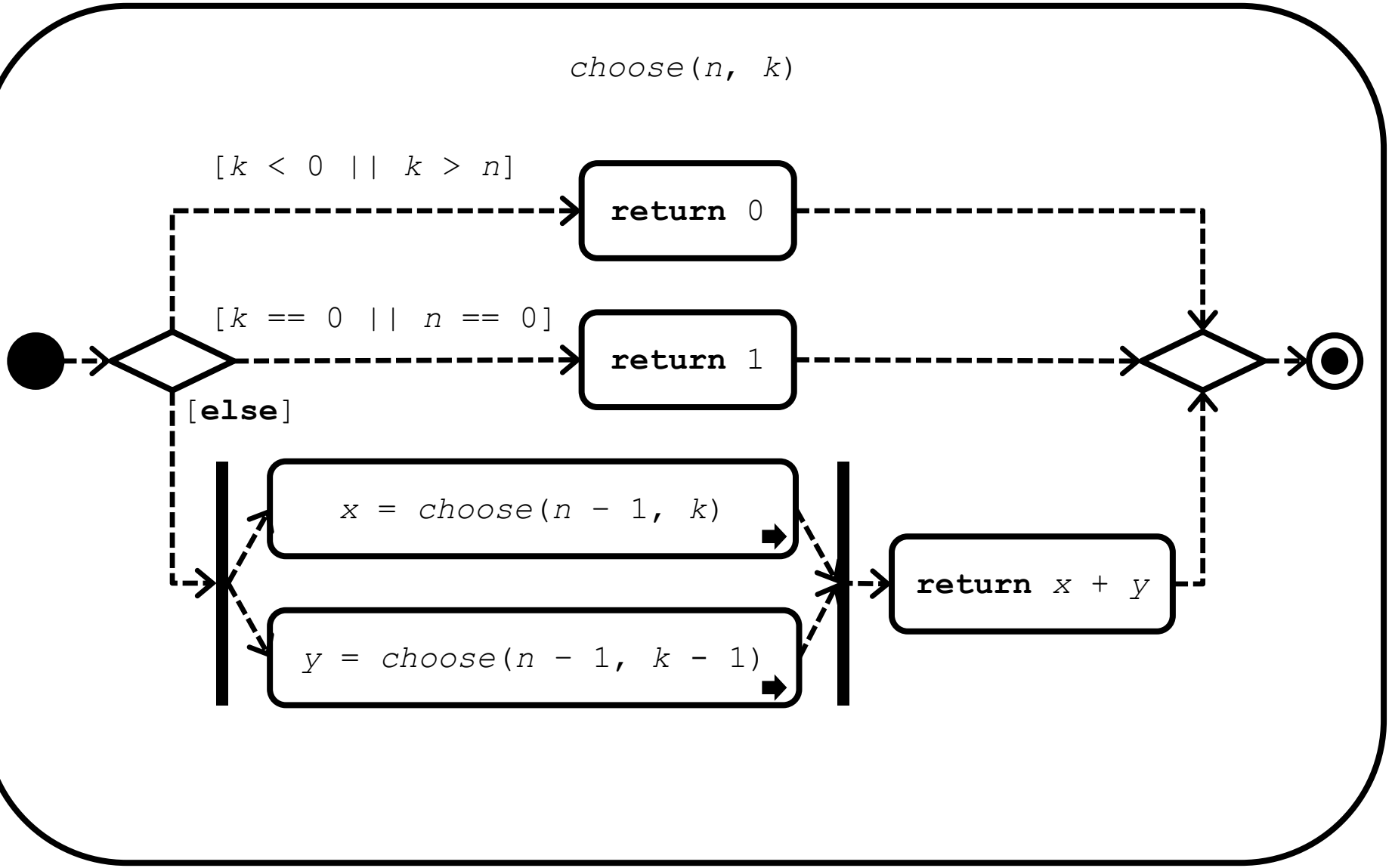
# Példa: *n* alatt a *k*

```
int choose(int n, int k) {
  if (k < 0 || k > n) {
    return 0;
  } else if (k == 0 && n == 0) {
    return 1;
  } else {
    int x = spawn choose(n - 1, k);
    int y = spawn choose(n - 1, k - 1);
    sync;
    return x + y;
  }
}
```

$\binom{0}{0}=1$

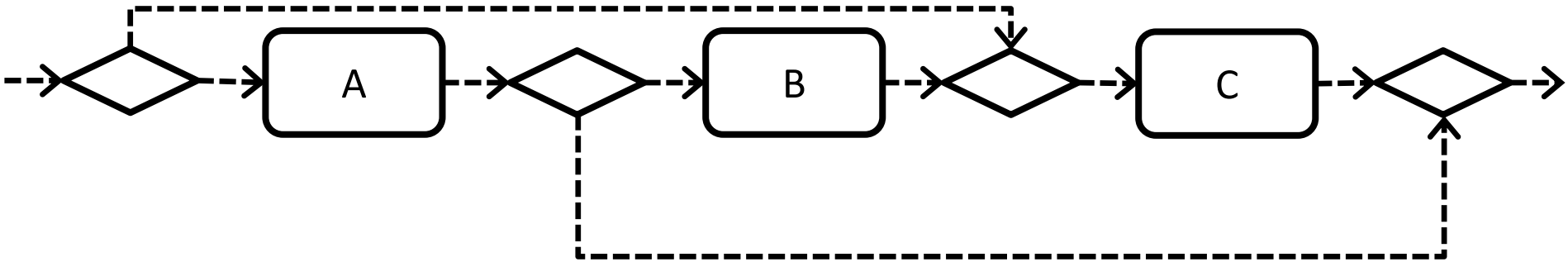
$\binom{n}{k}=\binom{n-1}{k}+\binom{n-1}{k-1}$

# Példa: *n* alatt a *k*



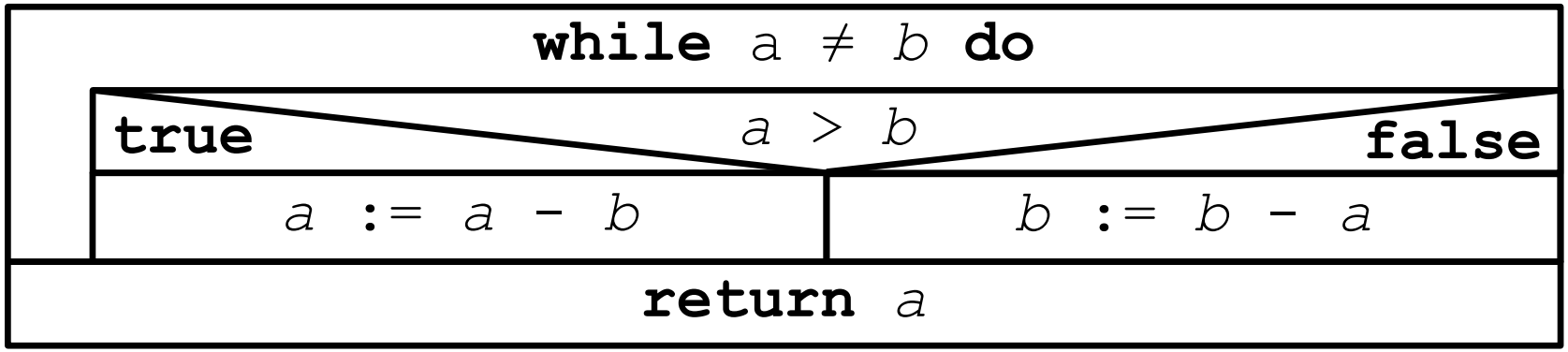
# Jólstrukturált folyamatok

- Vezérlési blokkokból építkezünk
  - **Egy bemenet, egy kimenet, közte jól strukturált blokk**
  - Szekvencia, decision-merge és fork-join blokk, ciklus
  - (üres vezérlési szakasz)
- Analógia: strukturált programozás (`goto` helyett vezérlési szerkezetek)
- Nem jólstrukturált folyamatra példa:



# Jólstrukturált folyamatok

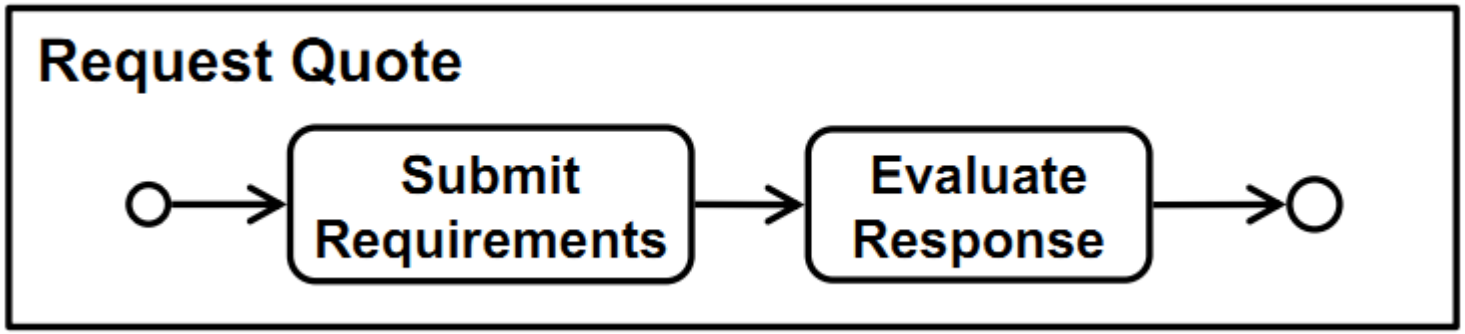
- Bizonyos formalizmusok kikényszerítik
  - pl. BPEL (üzleti folyamatok webszolgáltatások fölött)
  - Pl. Struktogram (Nassi-Shneiderman)



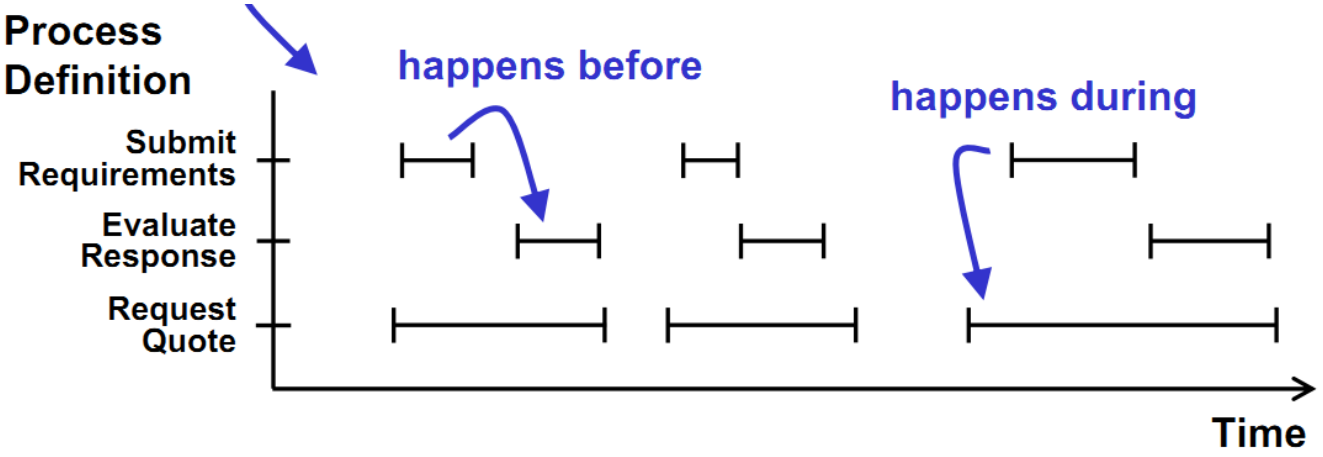
# ÜZLETI FOLYAMATOK VÉGREHAJTÁSA

# Folyamatok szemantikája

- Modellezés szempontjából

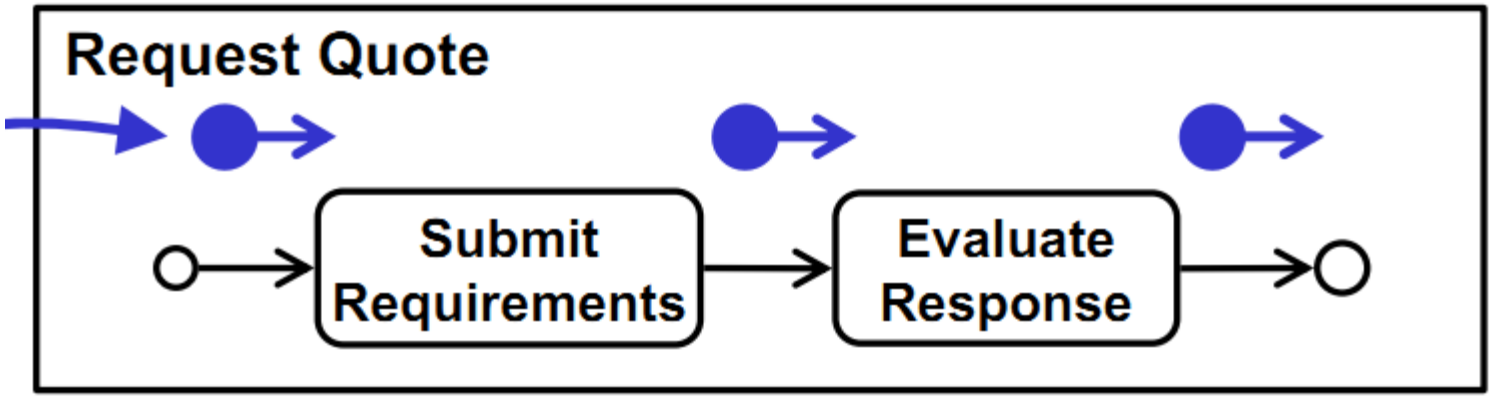



- Az elvárt működés

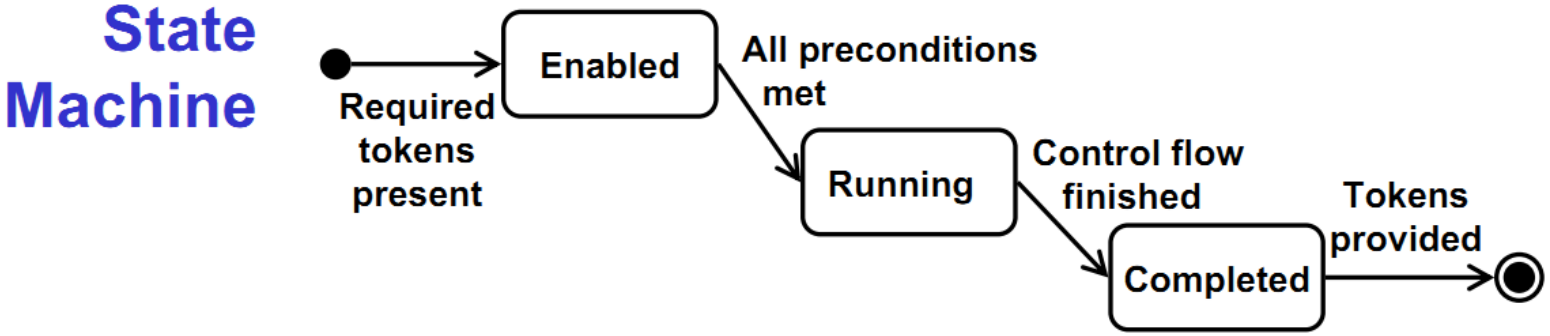


# Folyamat végrehajtása

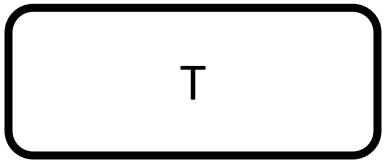
- Tokenáramlás

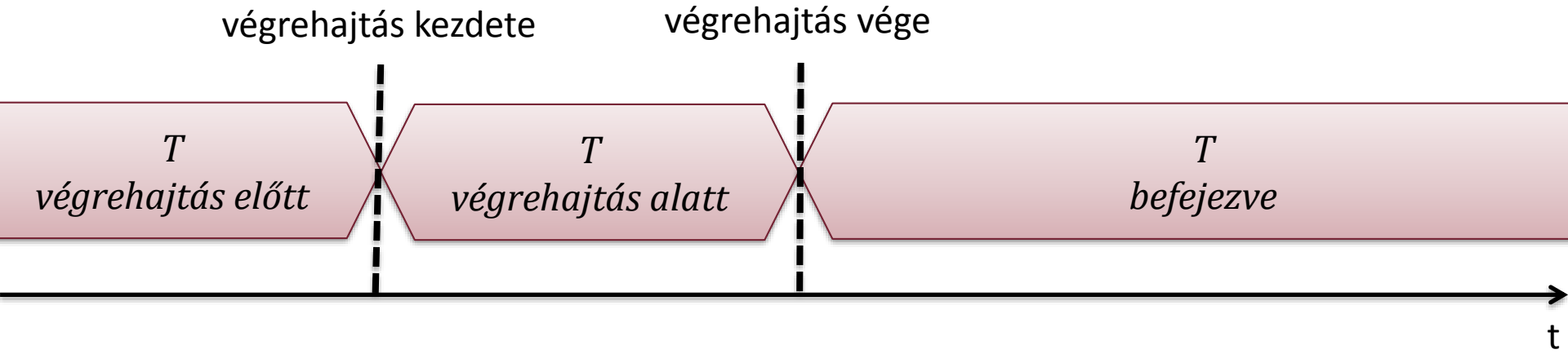



- A folyamat állapota

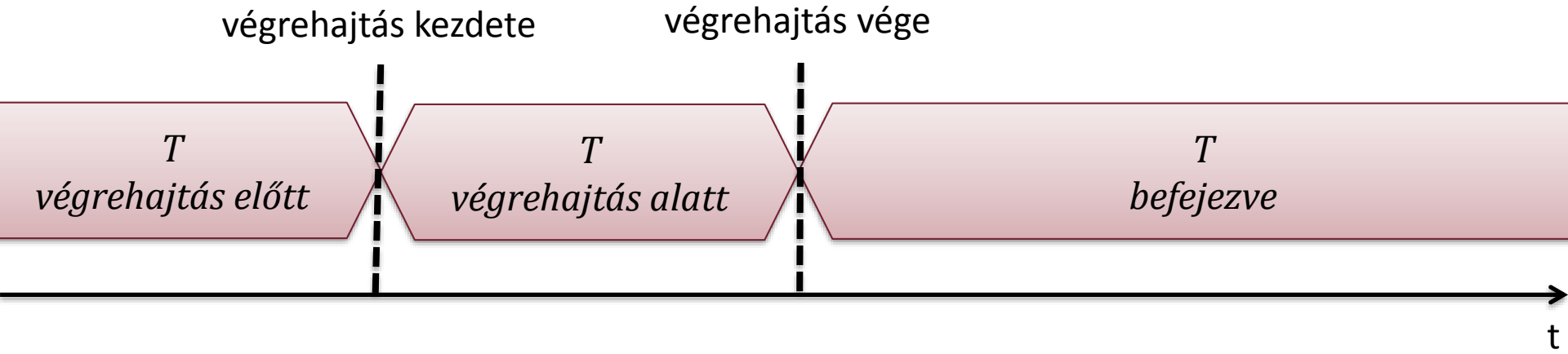


# Elemi tevékenység állapotai

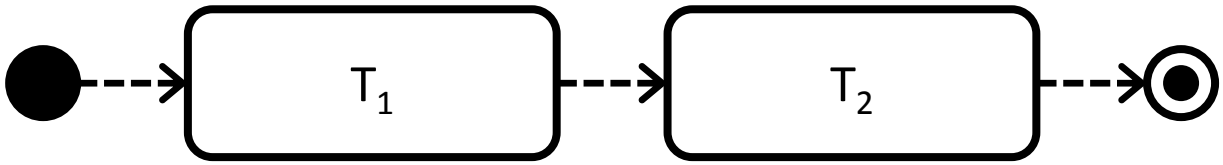


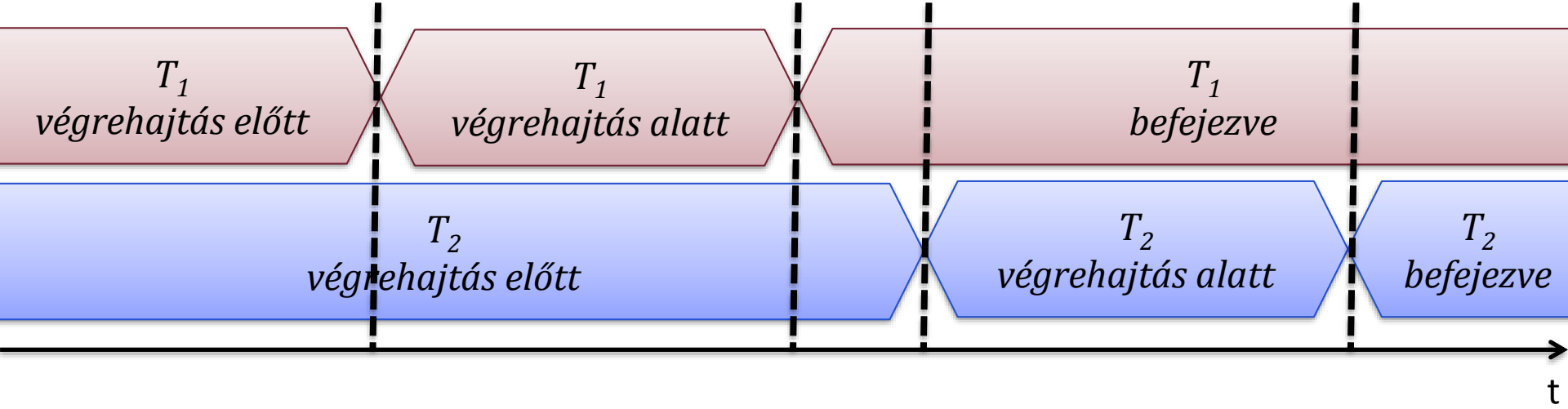



# Elemi tevékenység állapotai

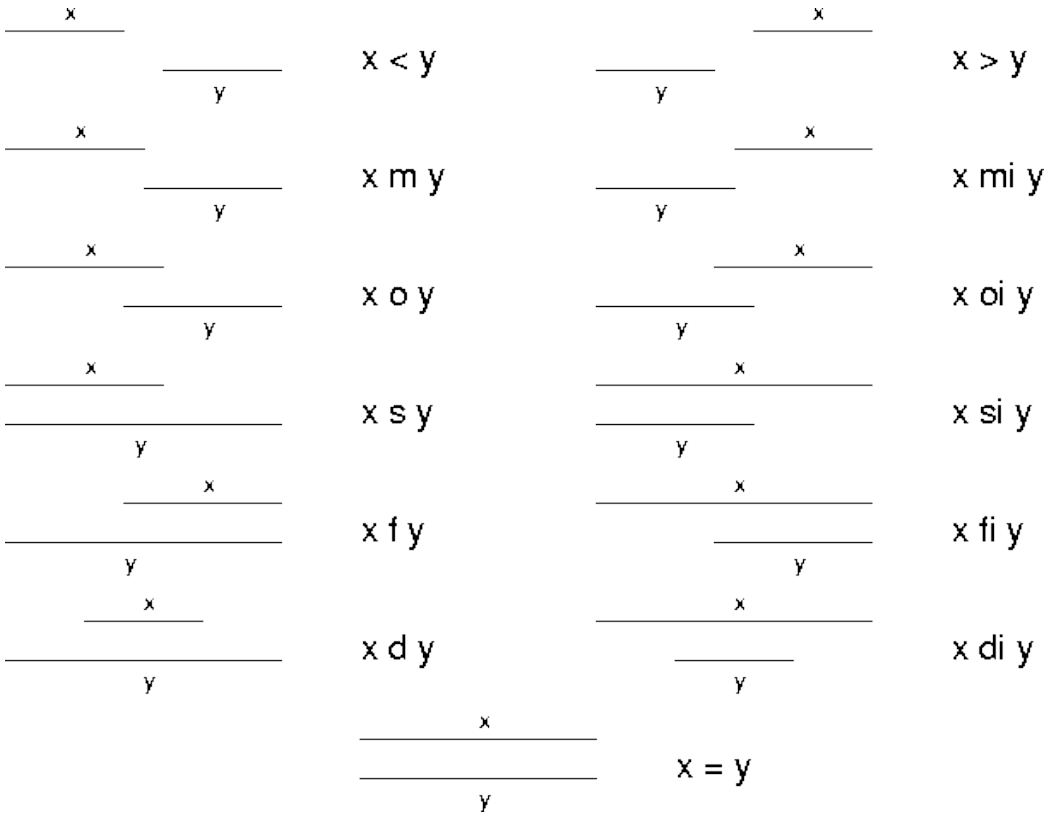
# Folyamat állapotai



# Háttér: matematikai modell

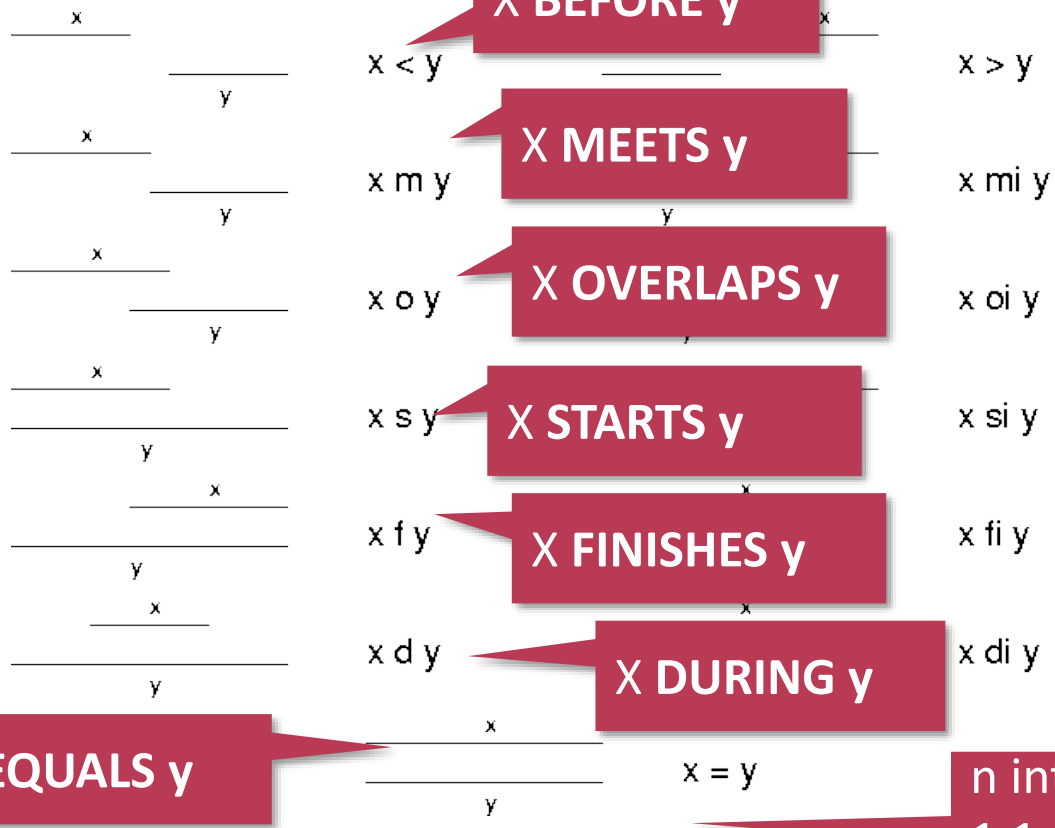
- Allen-féle intervallum logika (1983)
  - Pl. tesztelésnél használják, 13 (6 + 1 + 6) eset




James F. Allen: *Maintaining knowledge about temporal intervals*.
In: *Communications of the ACM*. 26 November 1983. ACM Press. pp. 832–843, ISSN 0001-0782

# Háttér: matematikai modell

- Allen-féle intervallum logika (1983)
  - Pl. tesztelésnél ha[illegible]13 (6 + 1 + 6) eset




n intervallum:
1,1,13,409, 23917… eset

James F. Allen: *Maintaining knowledge about temporal intervals*.
In: *Communications of the ACM*. 26 November 1983. ACM Press. pp. 832–843, ISSN 0001-0782

# Mit lehet ellenőrizni?

- Pl. a végrehajtás nem folyamat alapon történt
  - Megfelelt-e az elvárásoknak (sorrend, függetlenség)?
- Mi lehetett a “folyamat” a rendszer mögött?
  - Workflow mining
- Pl. a futtatókörnyezet megengedő
  - Lépés kihagyható
  - Ilyenkor is teljesülnek az elvárások?
- Eszköz: formális módszerek
  - Logika, Petri-hálók, modellellenőrzés, stb.