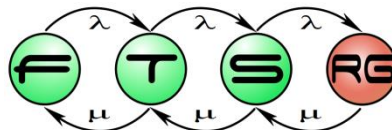


Folyamatmodellezés

Rendszermodellezés 2018.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport



Tartalom

Ismétlés, kitekintés



Folyamatmodellezés célja



Folyamatmodellek



Vezérlési folyamat



Megvalósítás

Tartalom

Ismétlés, kitekintés



Folyamatmodellezés célja



Folyamatmodellek



Vezérlési folyamat



Megvalósítás

Ismétlés: felépítési vs. viselkedési modellek

■ Felépítési (*structural*) modellek

- Statikus
- Rész és egész, összetevők
- Kapcsolatok, összeköttetések

Az autóban van kamera és kormányvezérlő

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

■ Viselkedési (*behavioral*) modellek

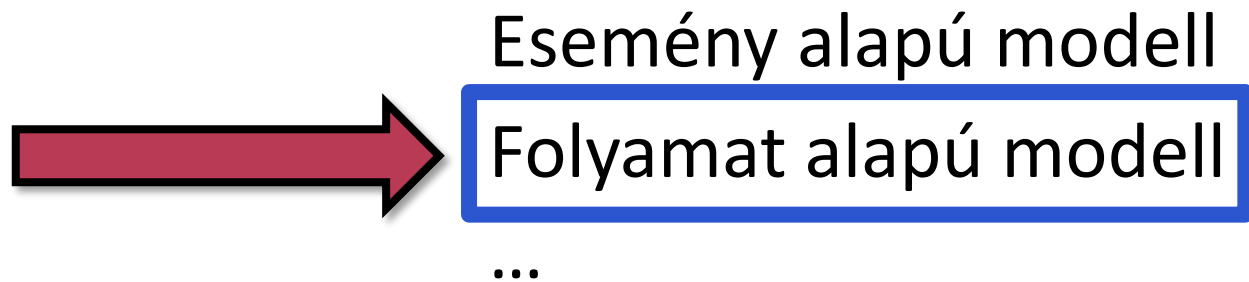
- Dinamikus
- Időbeli lefolyás
- Állapot, folyamat
- Reakciók a külvilágra

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

■ Nem fed le mindent, nem válik élesen szét...

Viselkedésmodellek fő kérdései

- Mit „csinál” a rendszer?



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



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

Tartalom

Ismétlés, kitekintés



Folyamatmodellezés célja



Folyamatmodellek



Vezérlési folyamat

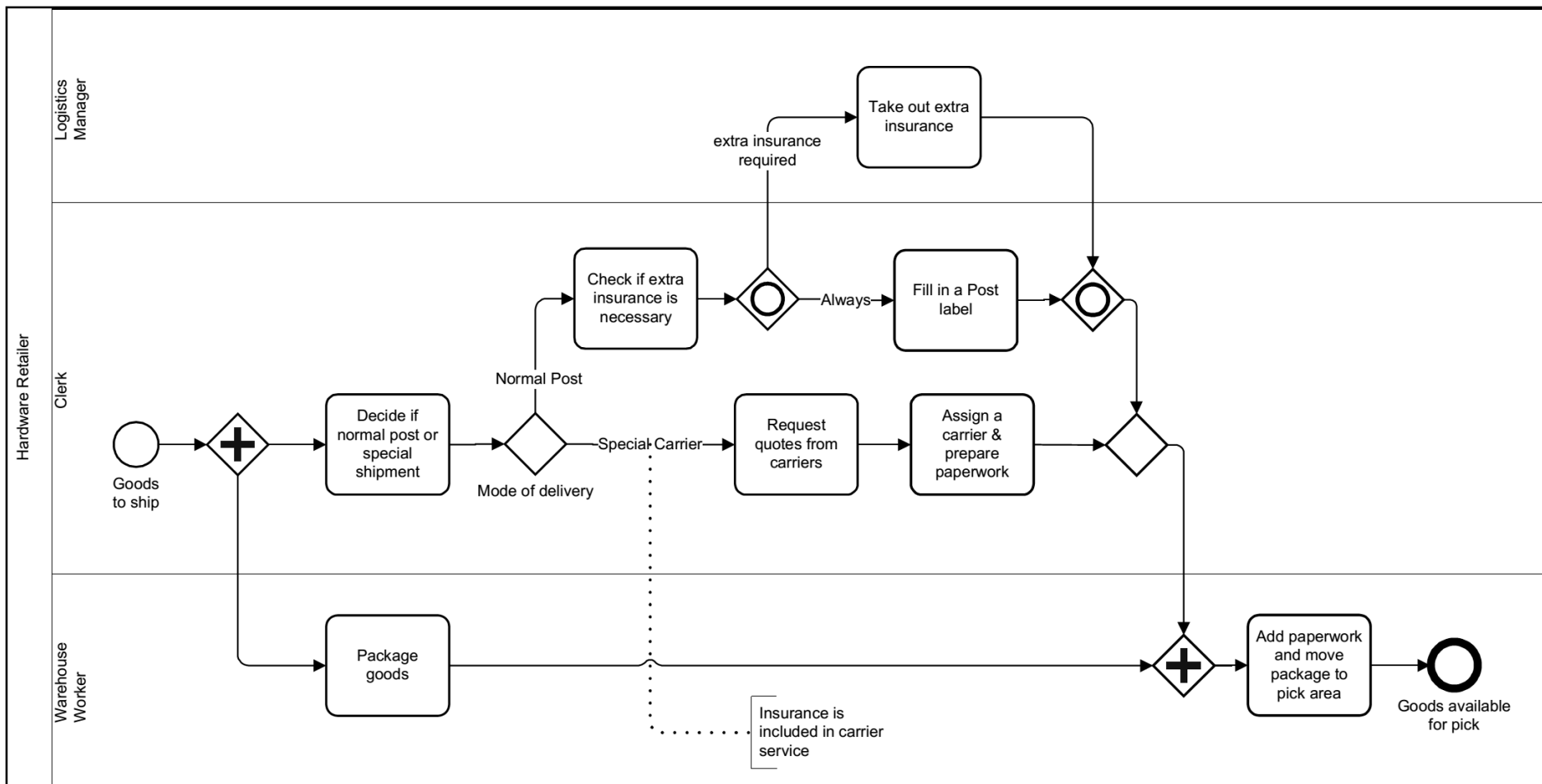


Megvalósítás

Folyamatmodellezés célja

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

Példa: HW rendelés kiszállítása



Példa: HW rendelés kiszállítása

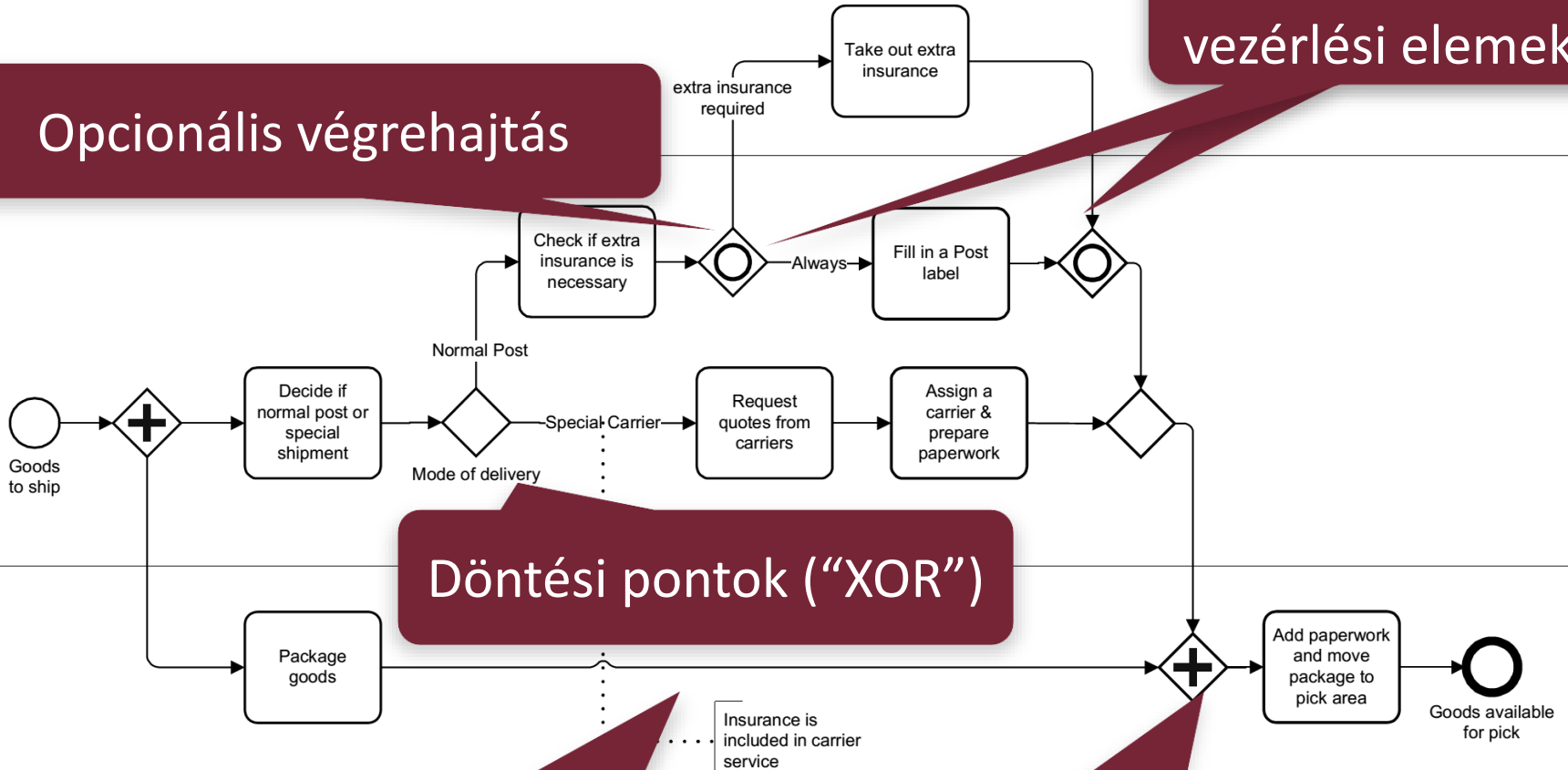
Opcionális végrehajtás

Összetartozó vezérlési elemek

Döntési pontok ("XOR")

Lépések végrehajtási sorrendje

"Párhuzamos" (független) végrehajtás ("AND")



Mire épül?

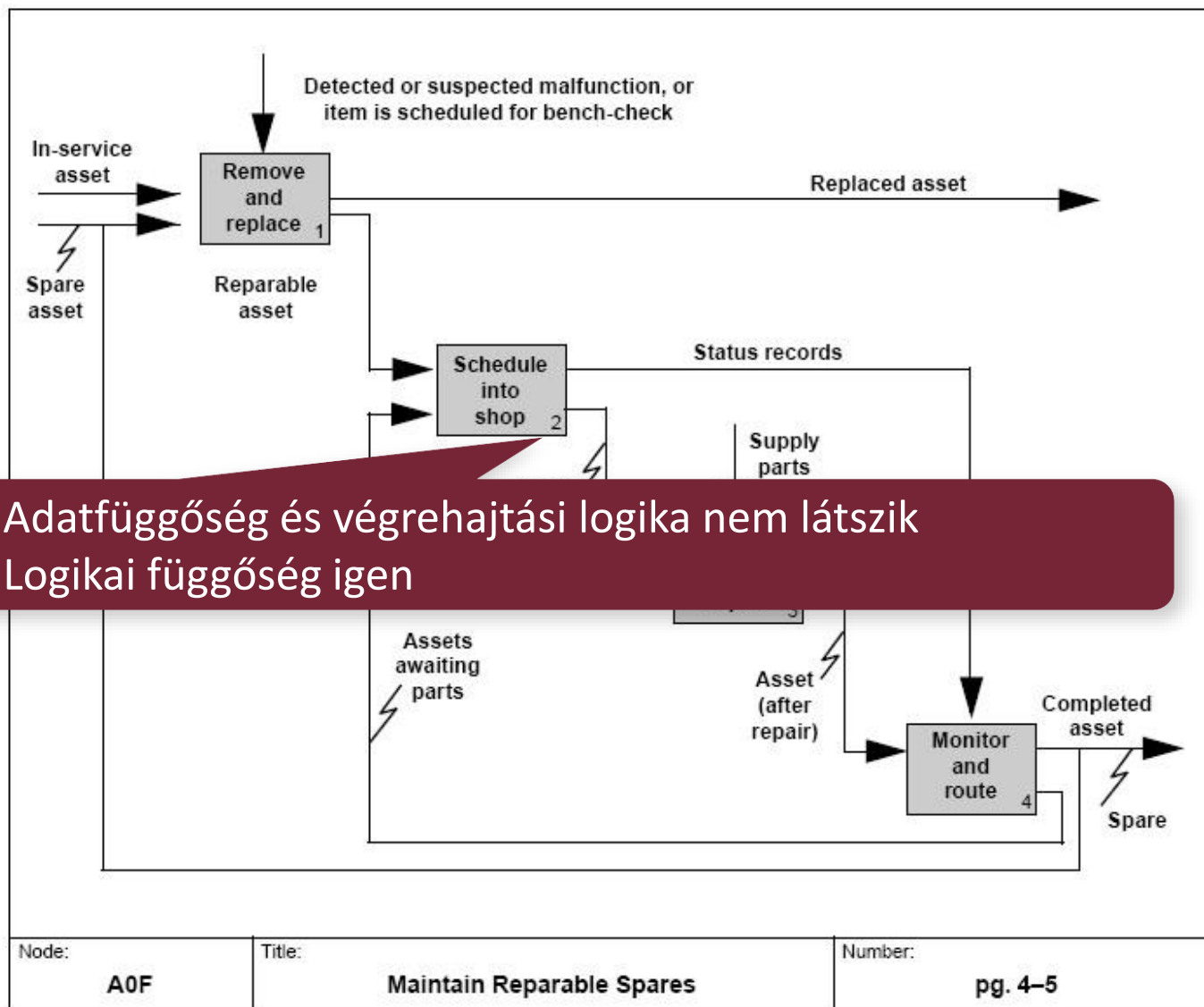
■ Történeti előzmények

- 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

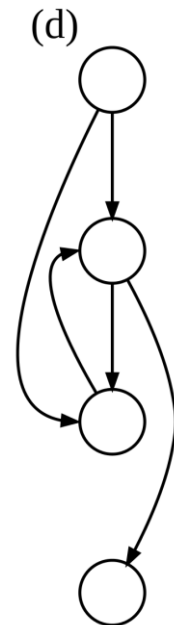
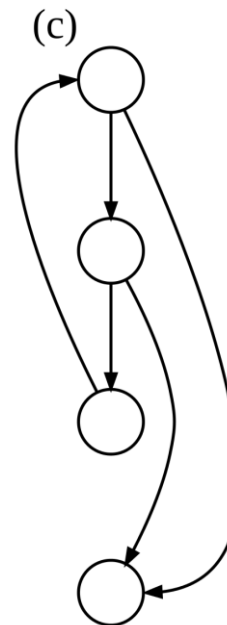
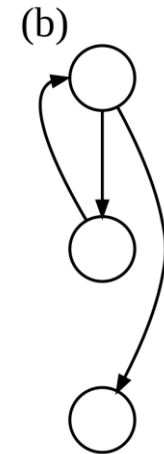
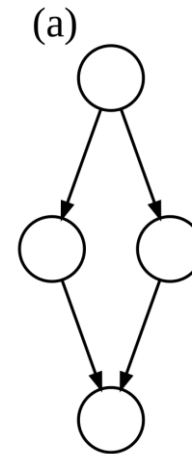
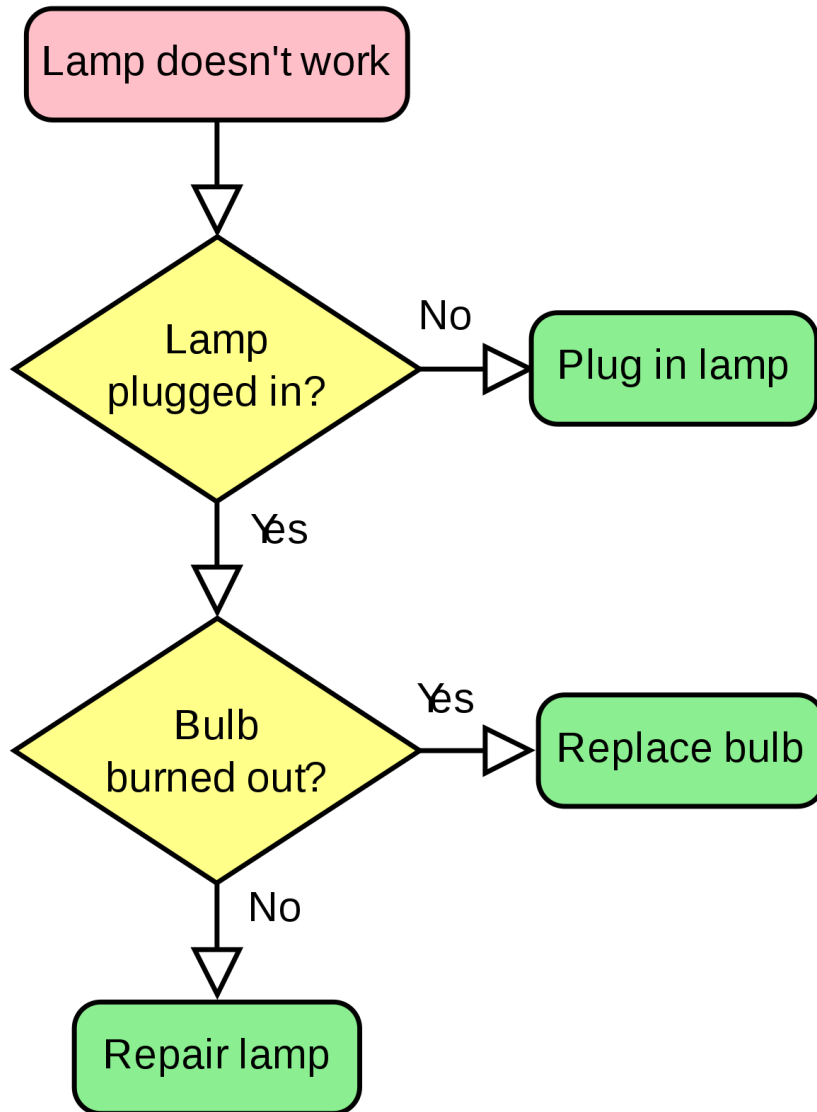
- Vannak elemi lépések
- Köztük függőségek (idő? adat? sorrend?)
- Döntési pontok
- → általános célú folyamatmodellezési nyelvek (pl. BPMN)

Példa: IDEF-0

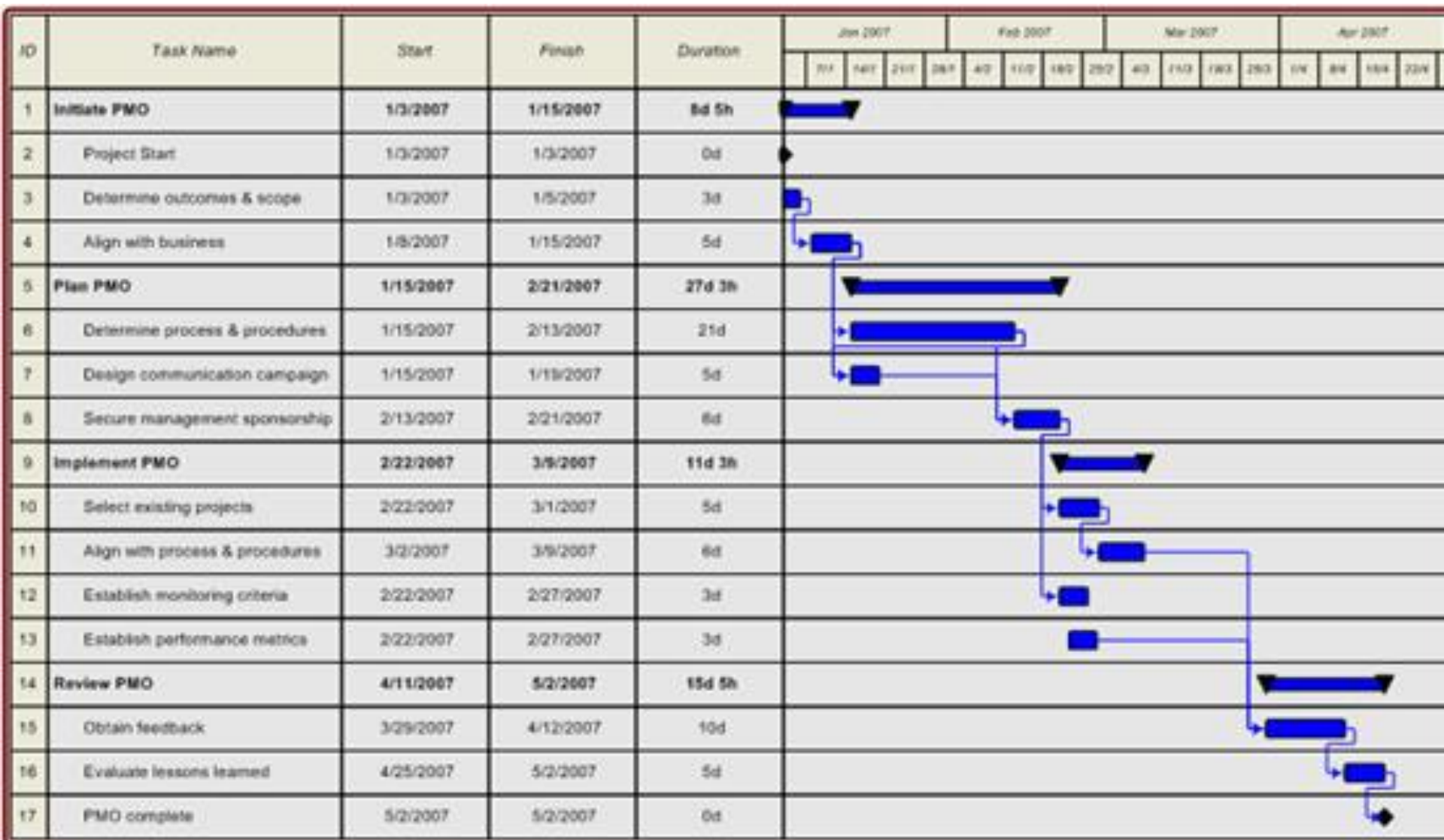


Adatfüggőség és végrehajtási logika nem látszik
Logikai függőség igen

flow



Gantt



Példa: GANTT

201x

GANTT CHARTS

MONTHLY PROJECT MANAGEMENT



Completed



To be completed



Not started

Shutterstock.com

This work is licensed under a [Creative Commons Attribution](#)

SOFTWARE DEVELOPMENT PLAN



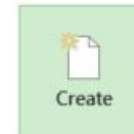
Software Development Plan

Provided by: [Microsoft Corporation](#)

Scope, analyze, design and develop a software product with this template.

Download size: 378 KB

Start Date:



Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
	Software Development	95.75 days	Wed 6/20/11	Thu 11/1/11		
	Scope	3.5 days	Wed 6/20/11	Tue 6/26/12		
	Analysis/Software Requirements	14 days	Tue 6/26/12	Mon 7/16/12		
	Design	14.5 days	Mon 7/16/11	Fri 8/3/12		
	Development	21.75 days	Fri 8/3/12	Tue 9/4/12		
	Testing	48.75 days	Fri 8/3/12	Thu 10/11/11		
	Training	45.75 days	Fri 8/3/12	Mon 10/8/11		
	Documentation	30.5 days	Fri 8/3/12	Mon 9/17/11		
	Pilot	70.25 days	Mon 7/16/11	Mon 10/22/11		



Wedding planner

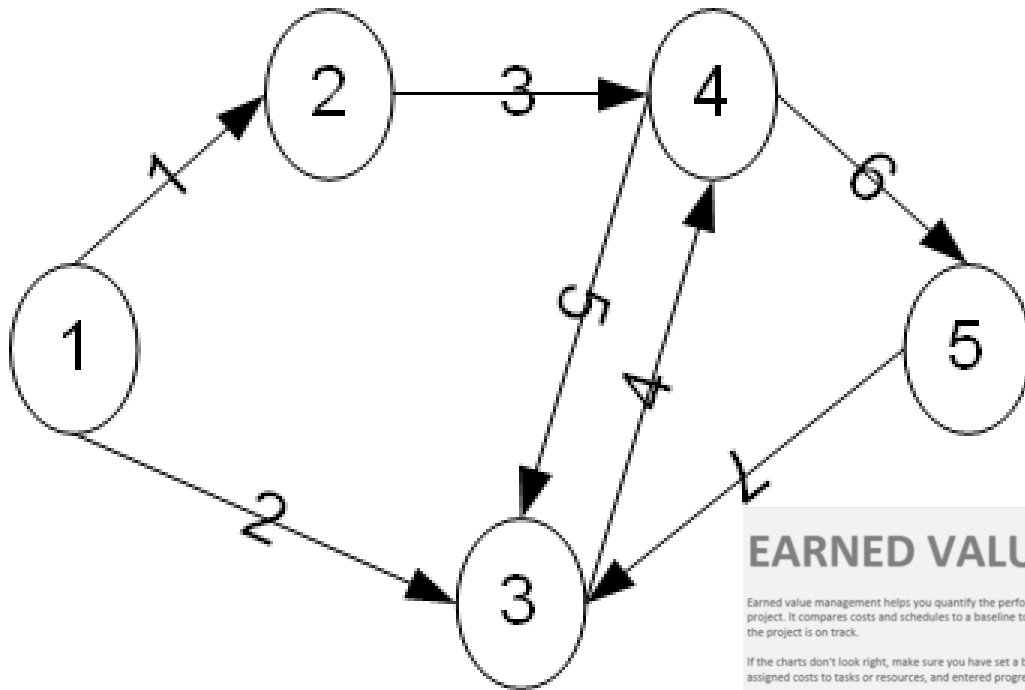
Provided by: [Microsoft Corporation](#)

This template assists the bride and groom-to-be with the organization of all the details that are involved in planning and executing a wedding.

Download size: 401 KB

Start Date:

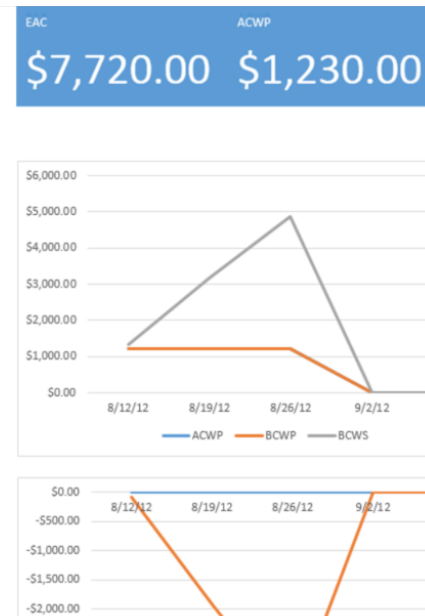




EARNED VALUE

Earned value management helps you quantify the performance of a project. It compares costs and schedules to a baseline to determine if the project is on track.

If the charts don't look right, make sure you have set a baseline, assigned costs to tasks or resources, and entered progress.



Earned value

Provided by: [Microsoft Corporation](#)

Learn how you can track project progress by using earned value management with this template.

Download size: 355 KB

Start Date:



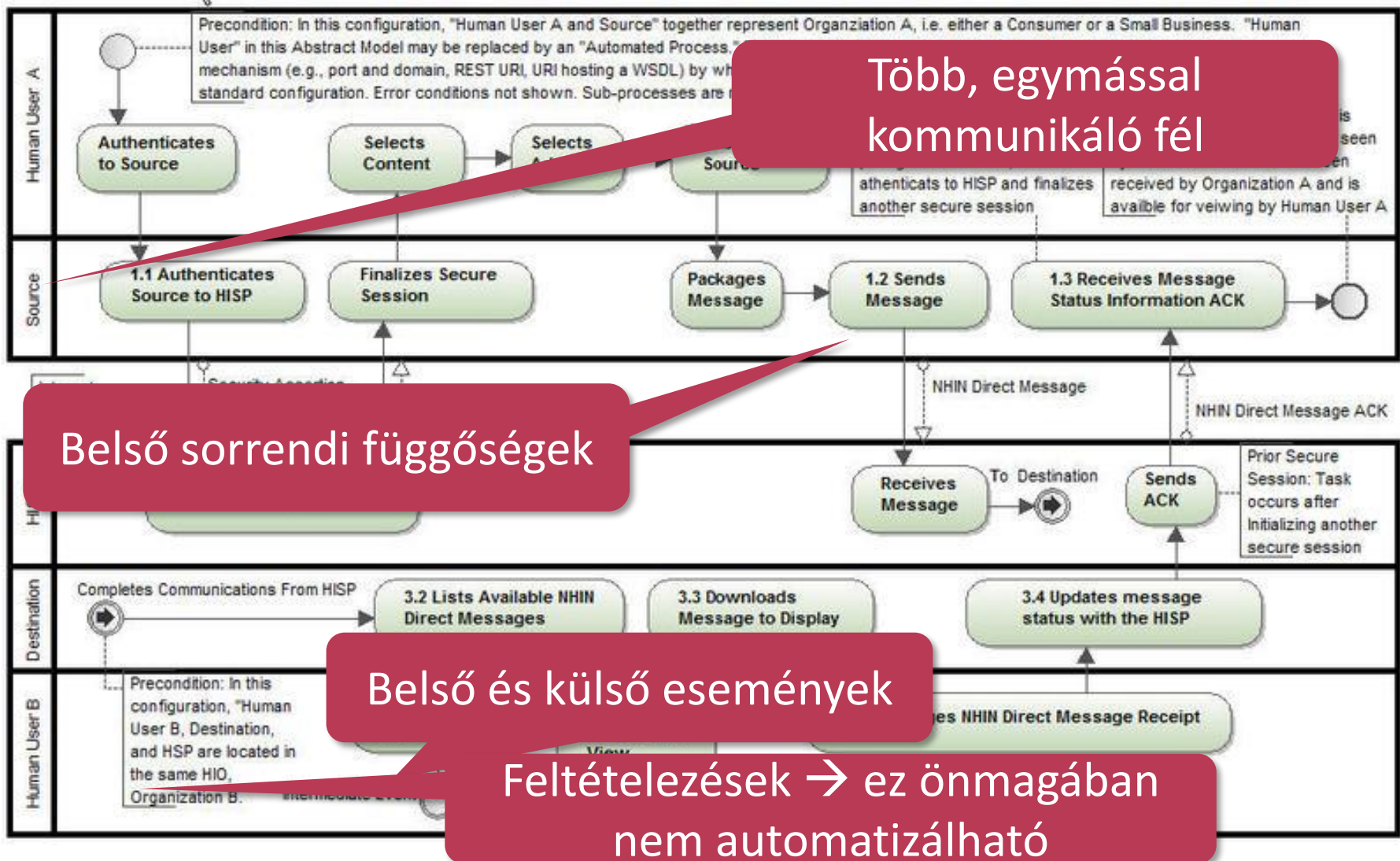
Mit használ fel?

- 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, hitelbírálathoz előkészítése, ...
- Adatfeldolgozási/elemezési folyamatok

Példa: egészségügyi adatok kezelése



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

Példa: agilis fejlesztés, mint folyamat

Szerepek, termékek

Scrum Roles > Scrum Team

Role: Scrum Team



The Scrum Team builds the product that the customer is going to consume: the software or website, for example. The team in Scrum is "cross-functional" - it includes all the expertise necessary to deliver the potentially shippable product each Sprint - and it is "self-organizing", with a very high degree of autonomy and accountability.

Role Sets: Scrum Roles

A csoportmunka lépései

Relationships



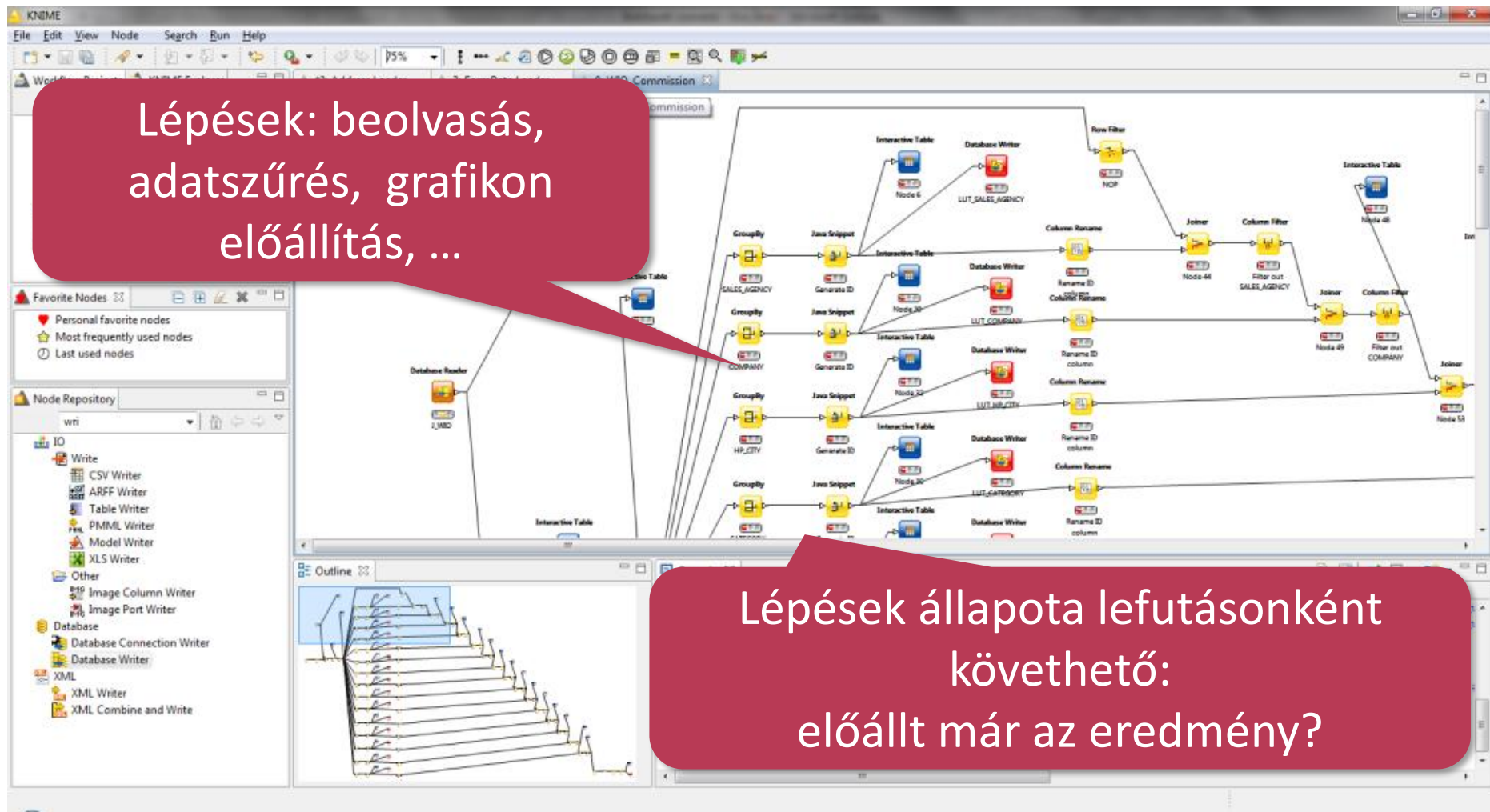
<http://www.eclipse.org/epf/>

Példák

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

Példa: adatfeldolgozási folyamat

Lépések: beolvasás,
adatszűrés, grafikon
előállítás, ...



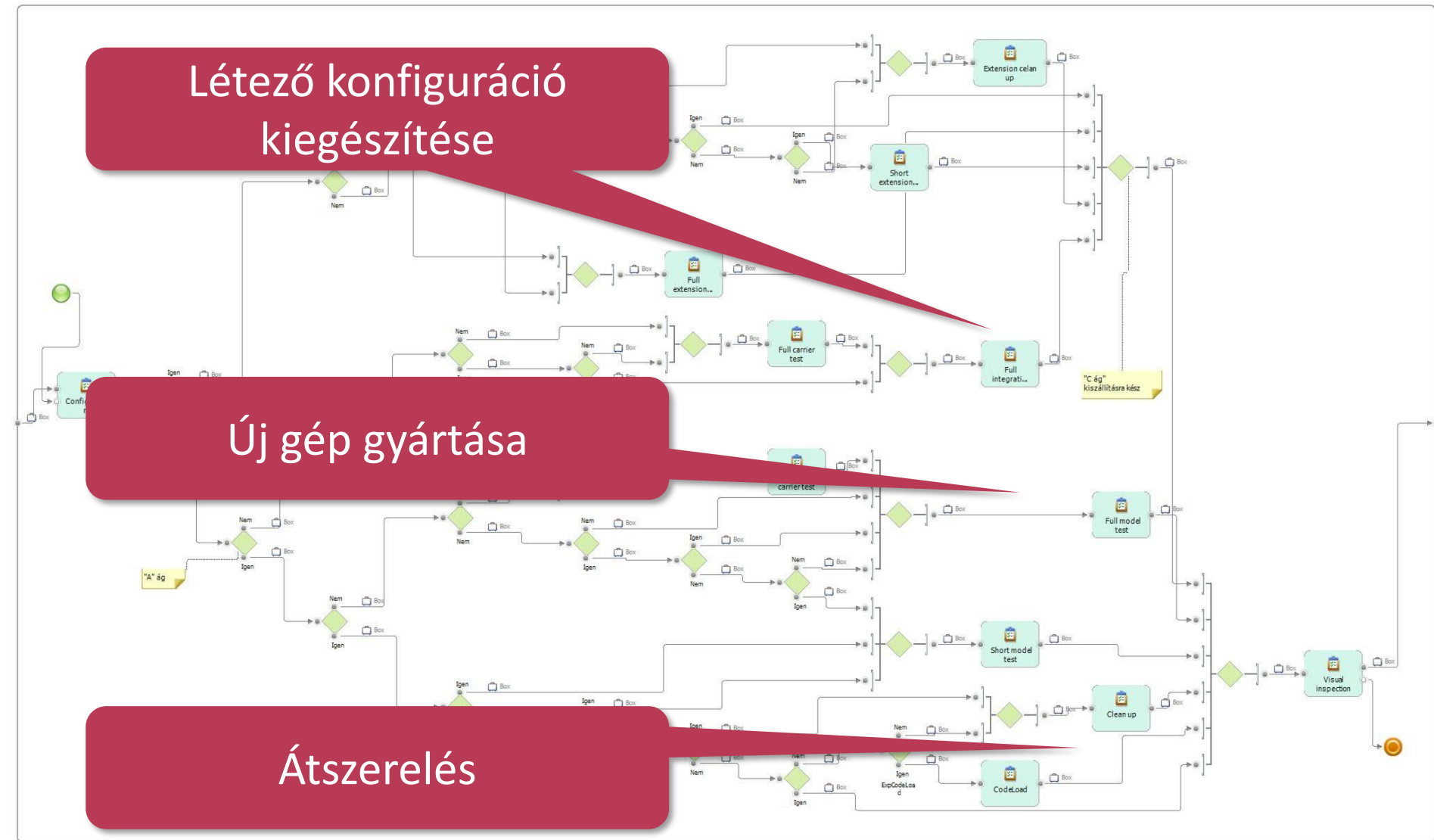
Eszköz: pl. KNIME

Példa: gyári tesztelés, mint folyamat

Létező konfiguráció
kiegészítése

Új gép gyártása

Átszerelés



Folyamatok tervezésének alapfogalmai

- Folyamat leíró nyelv
 - BPMN, jPDL, XPD, 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

Ismétlés, kitekintés



Folyamatmodellezés célja



Folyamatmodellek



Vezérlési folyamat



Megvalósítás

Elemi tevékenység

Compile

végrehajtás kezdete

végrehajtás vége

Compile

t

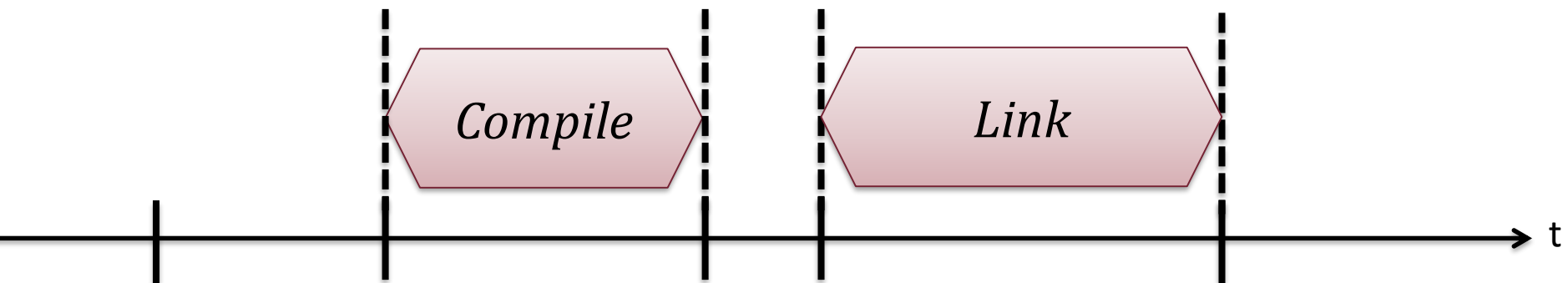
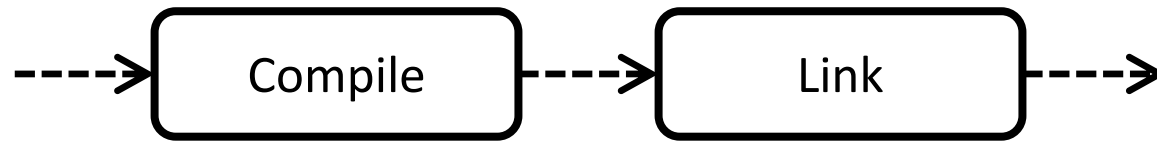
Definíció: Elemi tevékenység

Az **elemi tevékenység** olyan

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

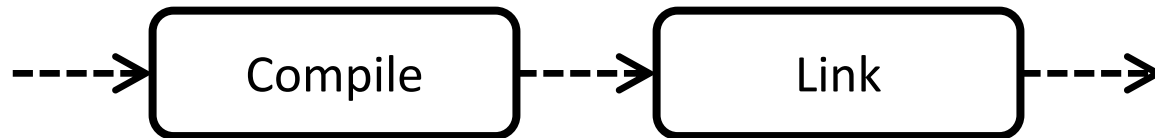
Compile

Szekvencia, vezérlésfolyam

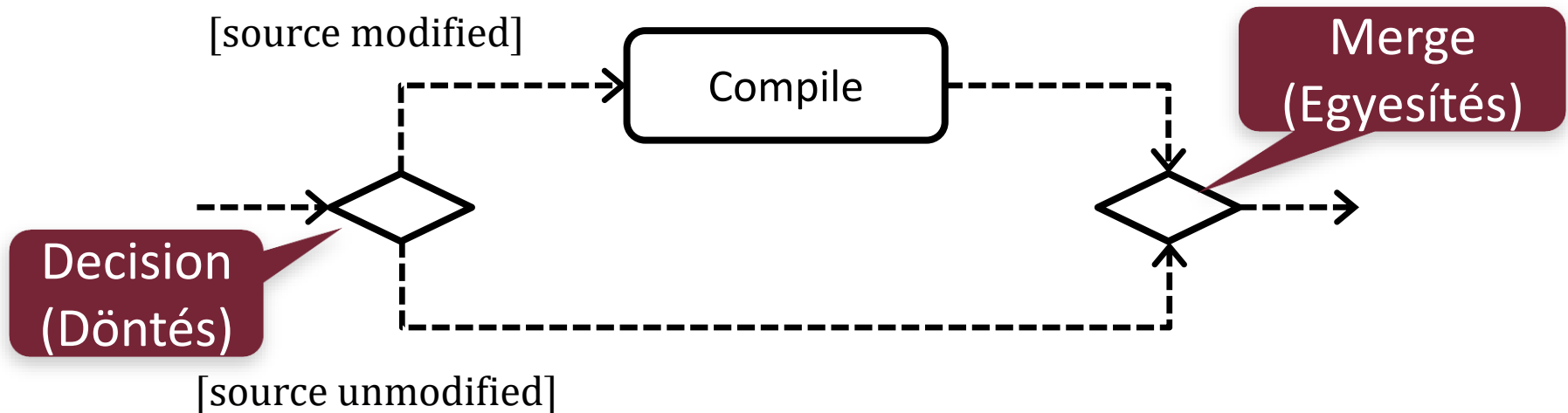


Definíció: Szekvencia

A **Szekvencia** tevékenységek végrehajtási sorrendjét definiálja.



Őrfeltételek, elágazás

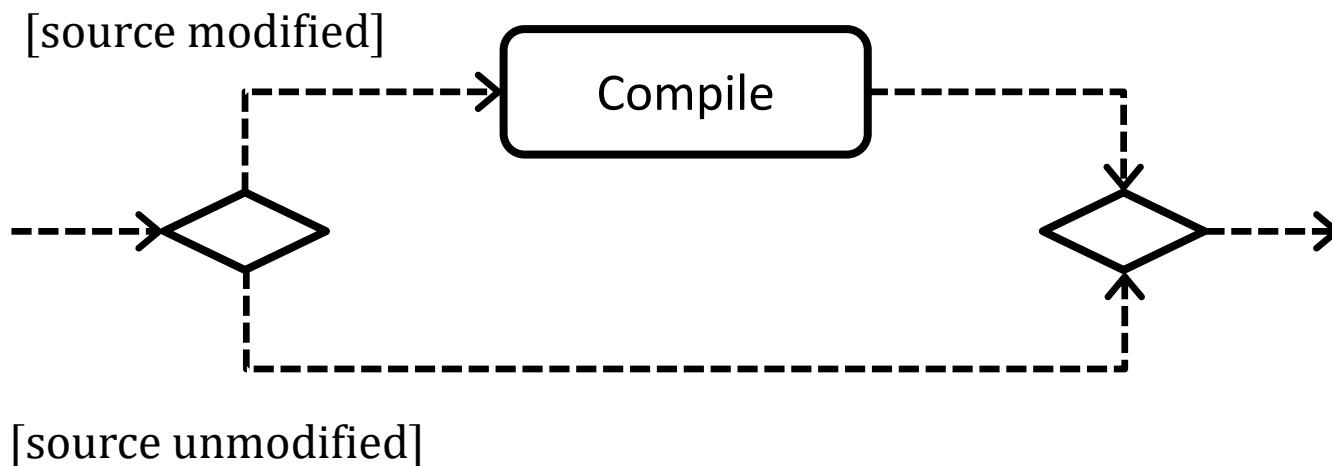


■ Szemantika:

- 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.

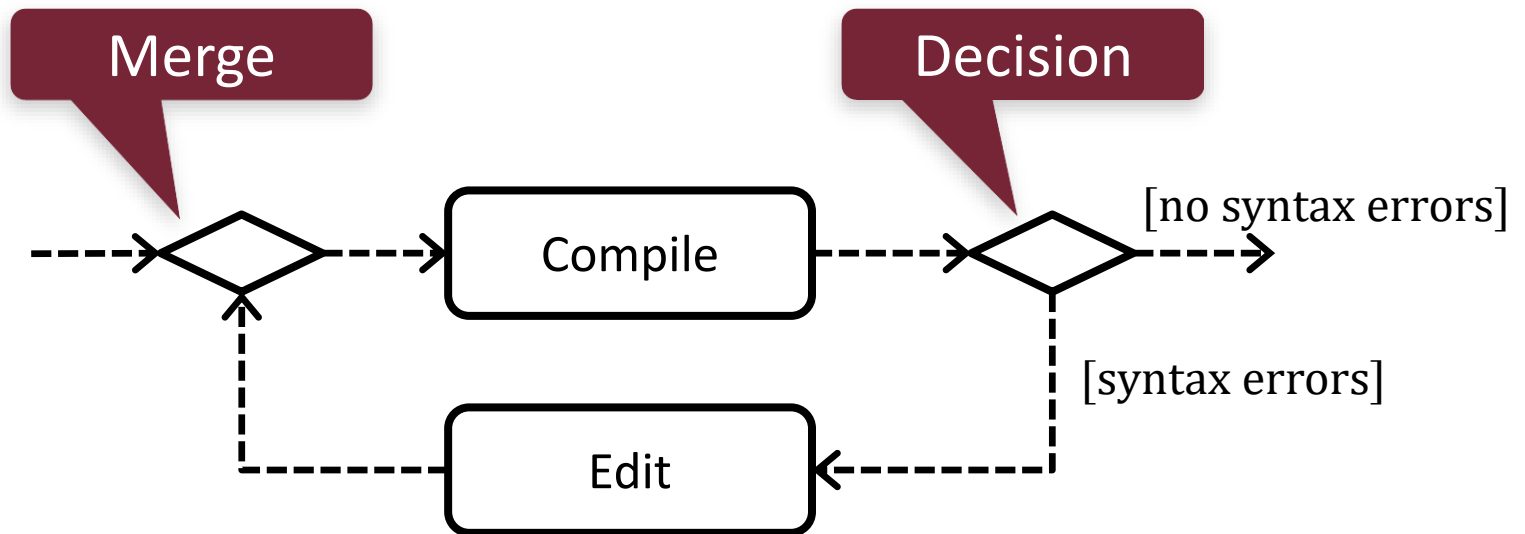


Definíció: Elágazás

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 kimenet (döntési ág) tetszőleges számú elemet tartalmazhat,
 - az összes döntési ág az Egyesítés csompontra 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

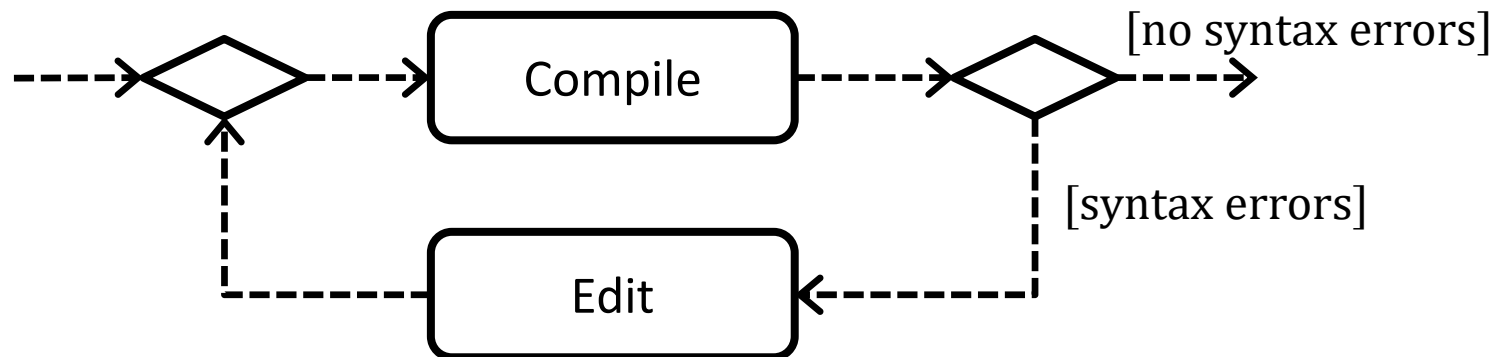
Ciklus



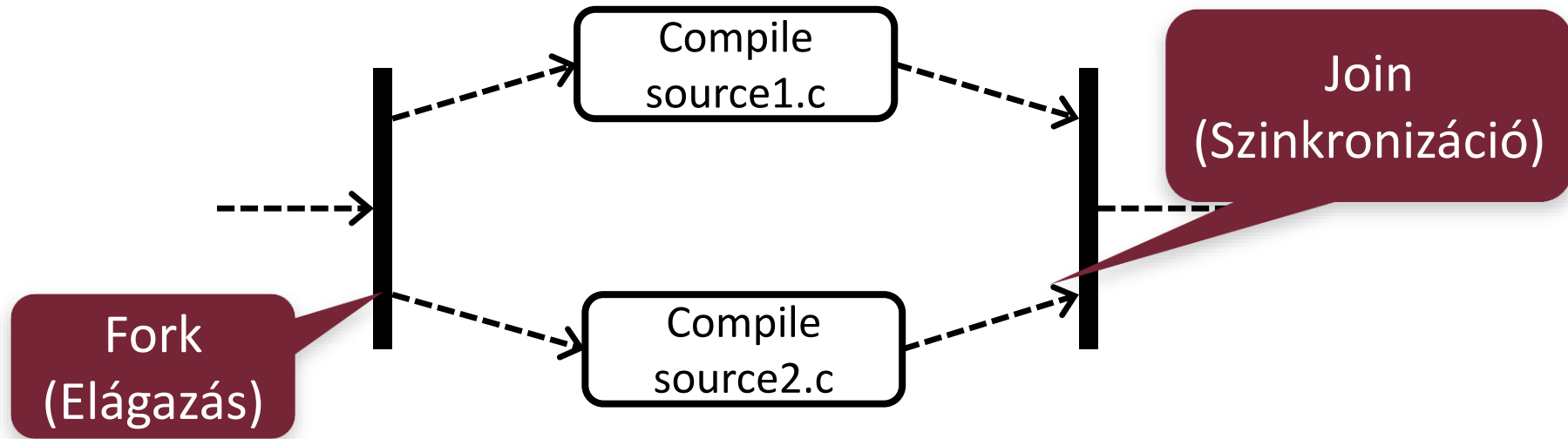
Definíció: ciklus

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

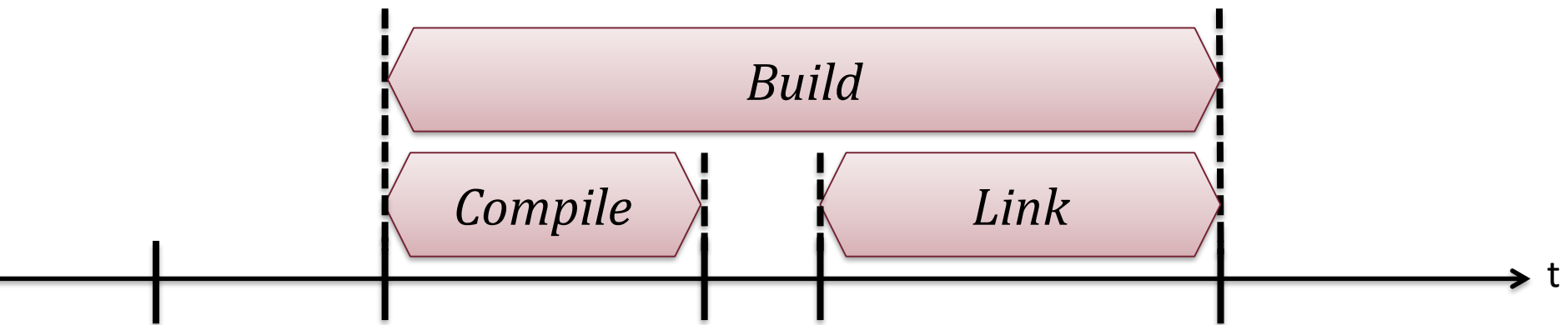
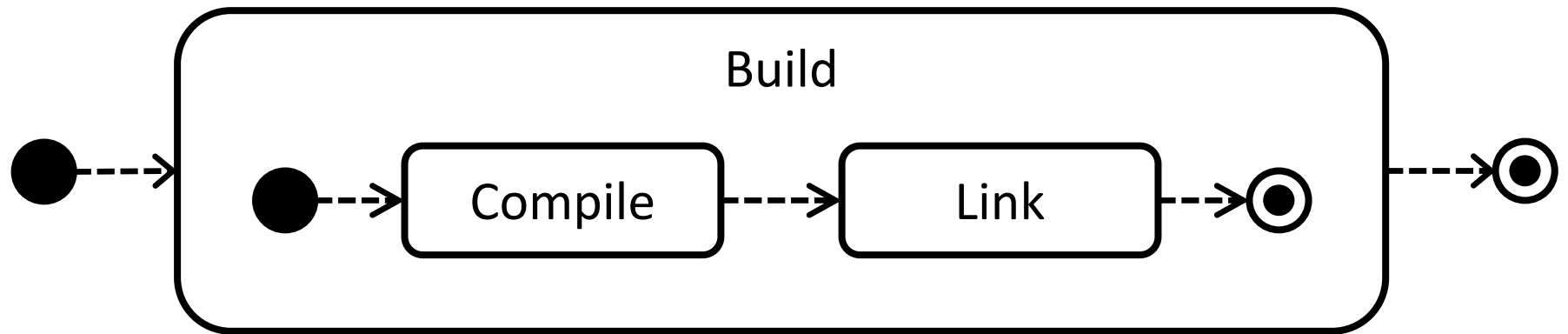


Definíció: folyamat indítása/befejezése

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

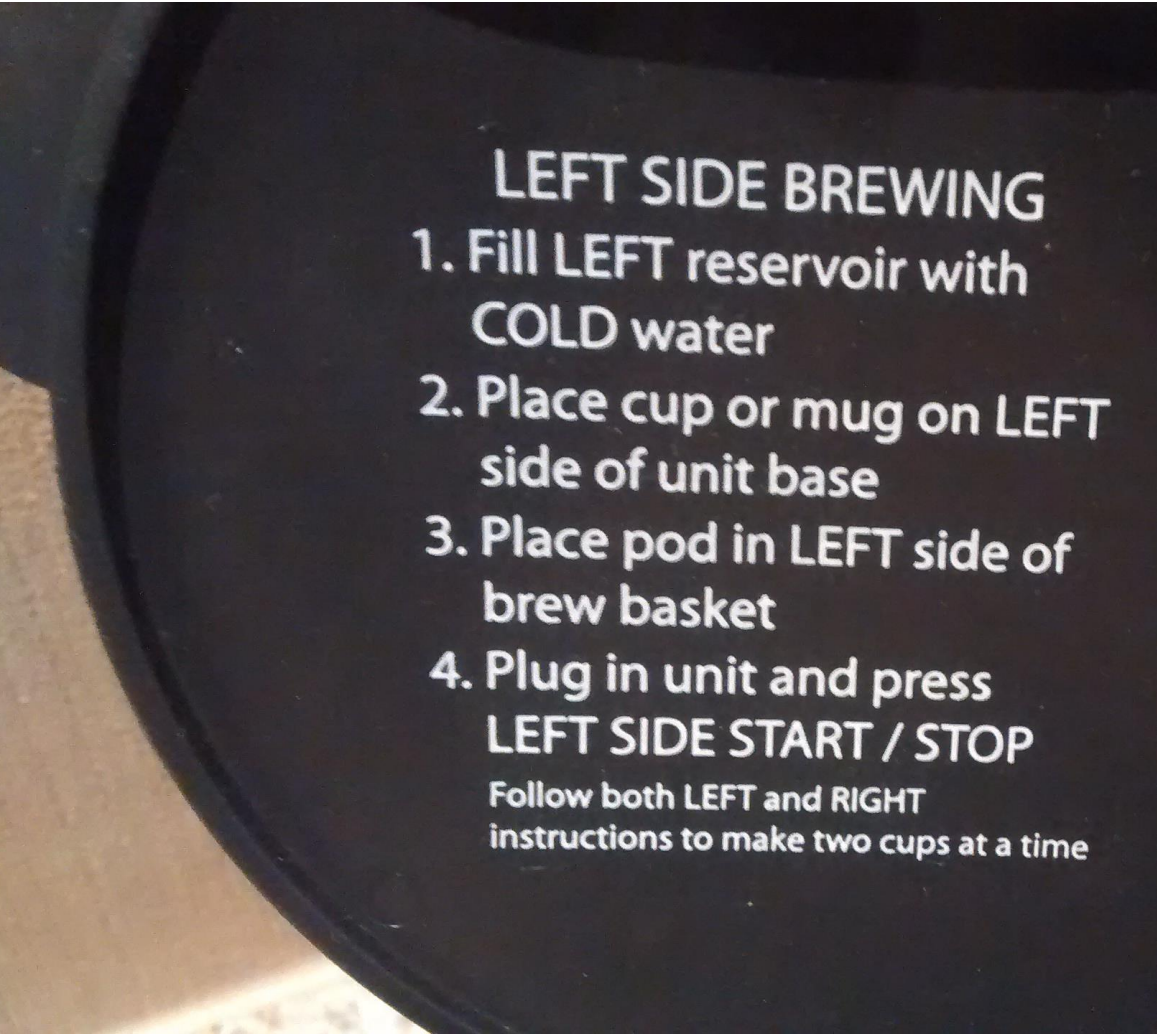


Definíció: hierarchia

Hierarchikus folyamatmodell:

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

Példa: kávéfőzés



LEFT SIDE BREWING

1. Fill LEFT reservoir with COLD water
2. Place cup or mug on LEFT side of unit base
3. Place pod in LEFT side of brew basket
4. Plug in unit and press LEFT SIDE START / STOP

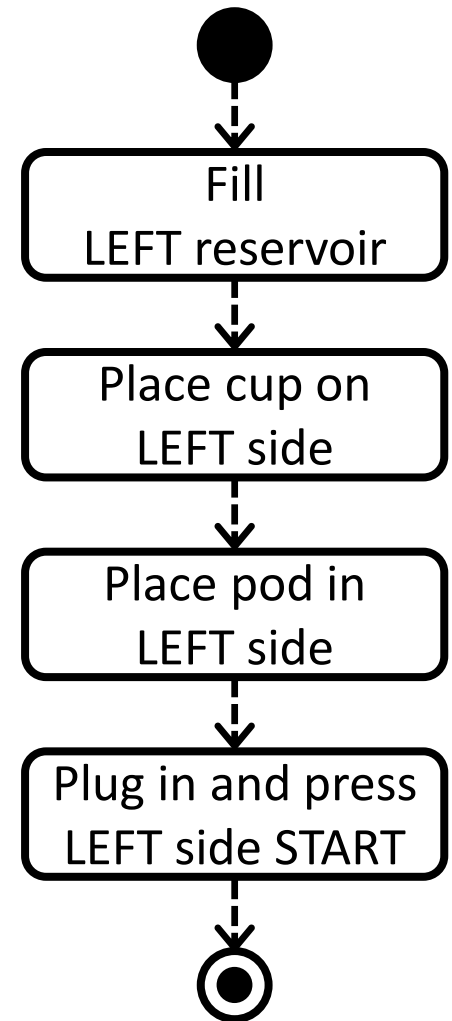
Follow both LEFT and RIGHT instructions to make two cups at a time

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

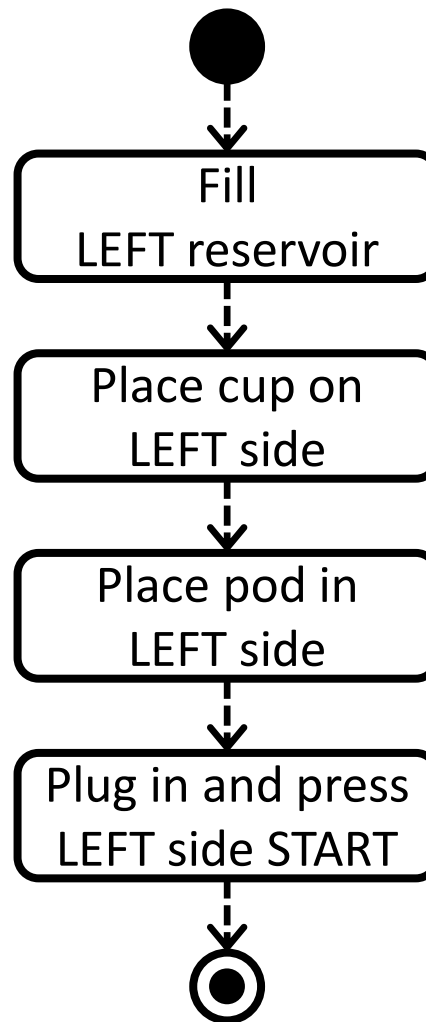
Ha egyszerre két kávéat akar főzni, egyszerre kövesse a BAL és JOBB utasításokat

Példa: kávéfőzés

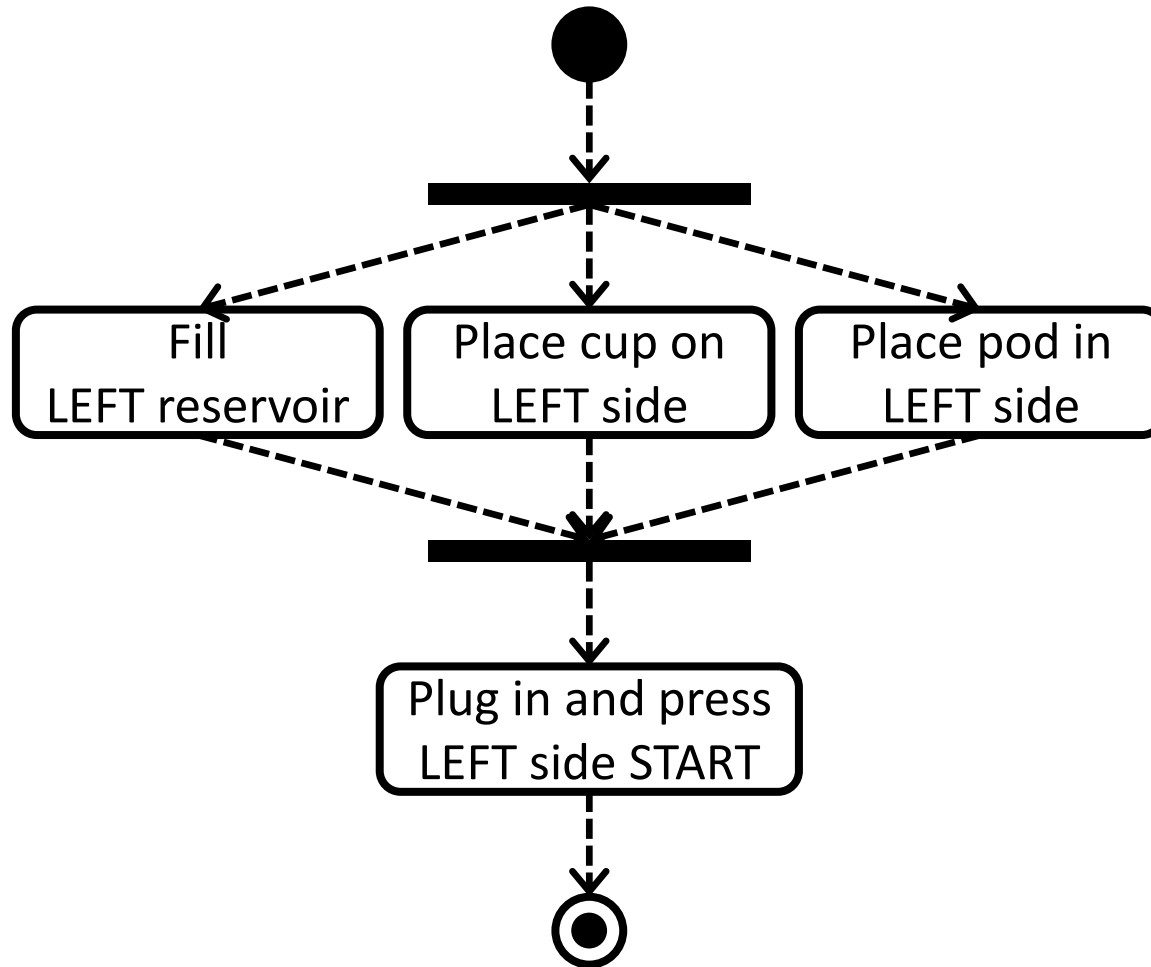
- LEFT SIDE BREWING**
1. Fill LEFT reservoir with COLD water
 2. Place cup or mug on LEFT side of unit base
 3. Place pod in LEFT side of brew basket
 4. Plug in unit and press LEFT SIDE START / STOP
- Follow both LEFT and RIGHT instructions to make two cups at a time



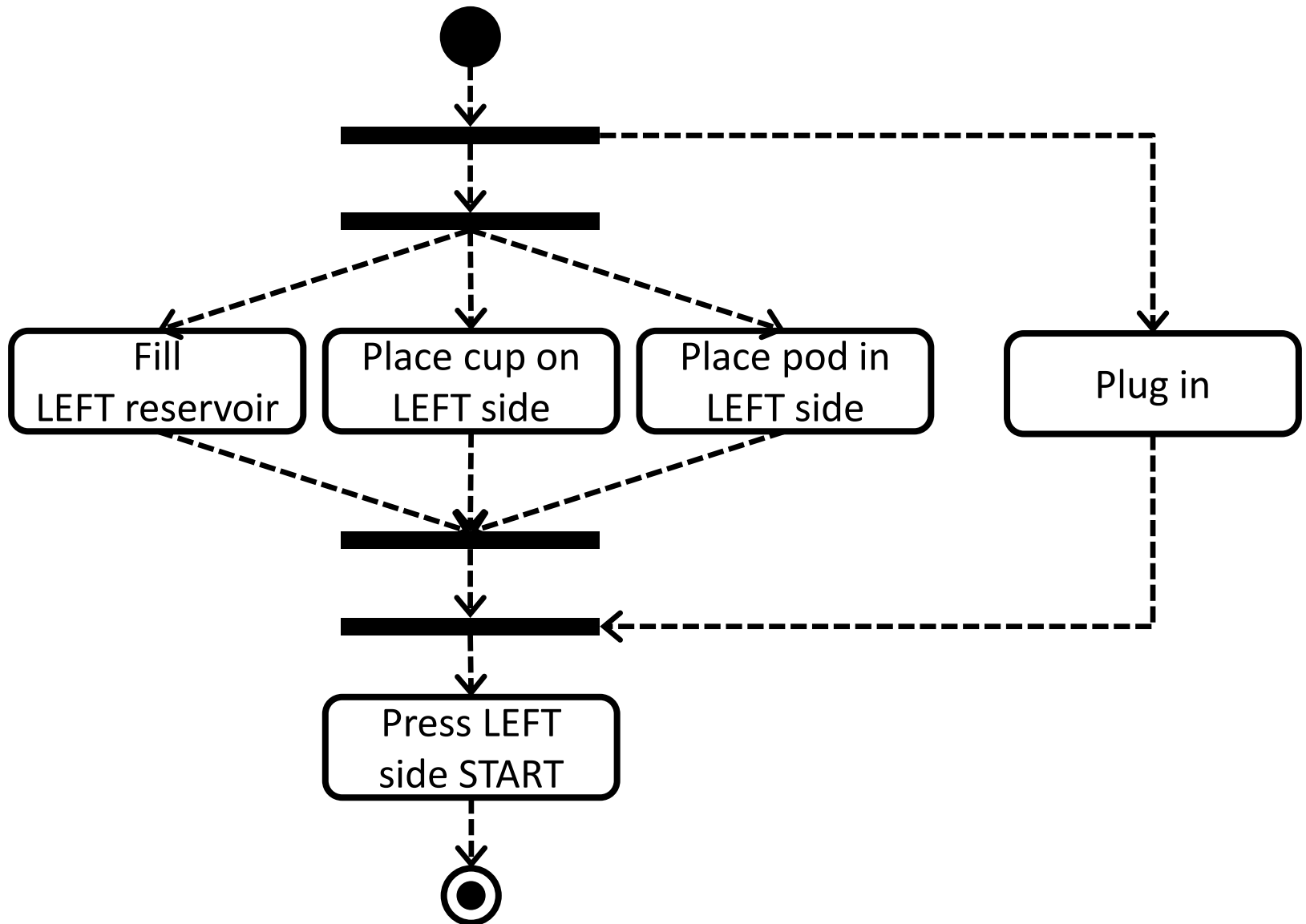
Kávéfőzés



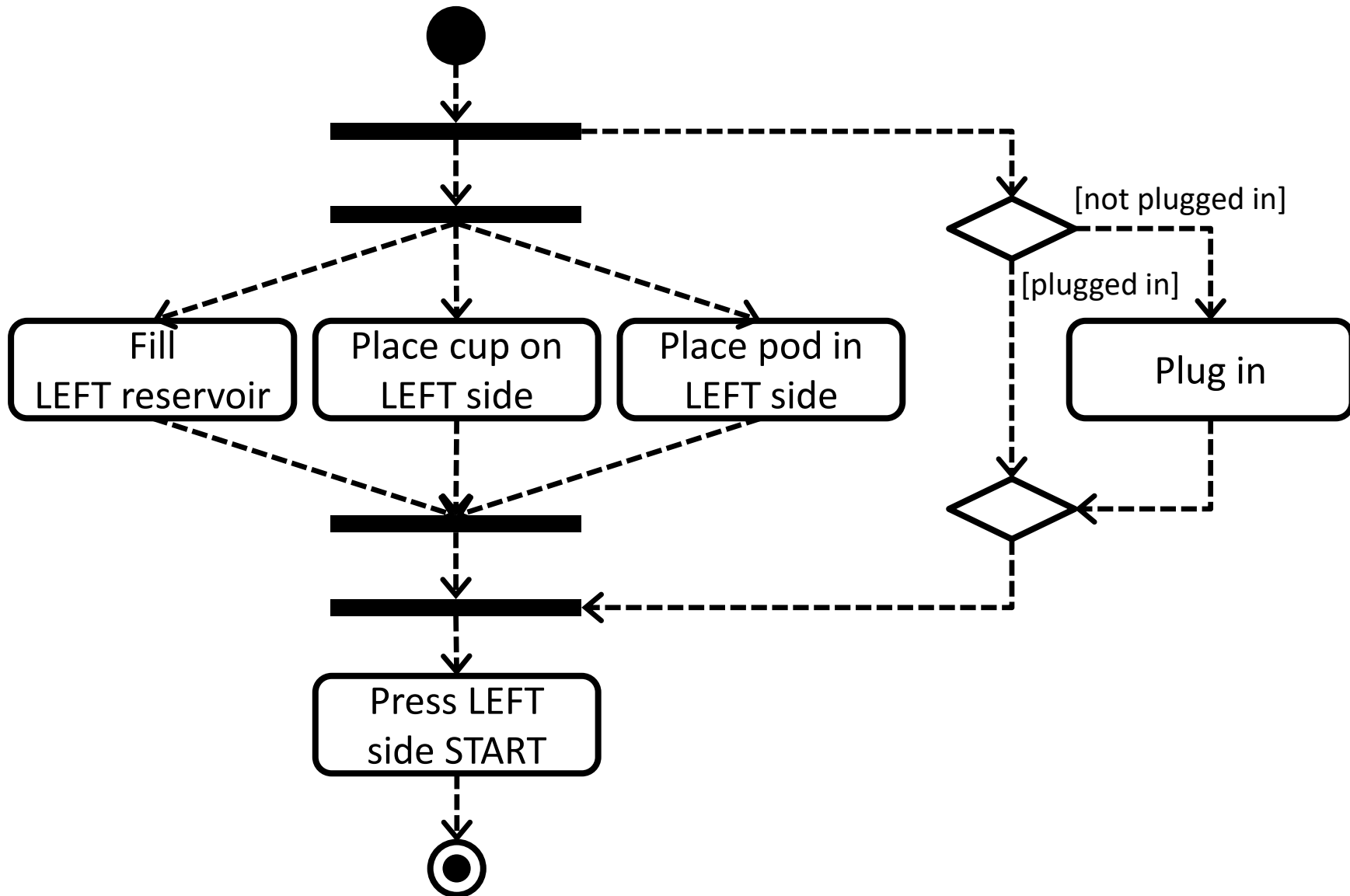
Kávéfőzés



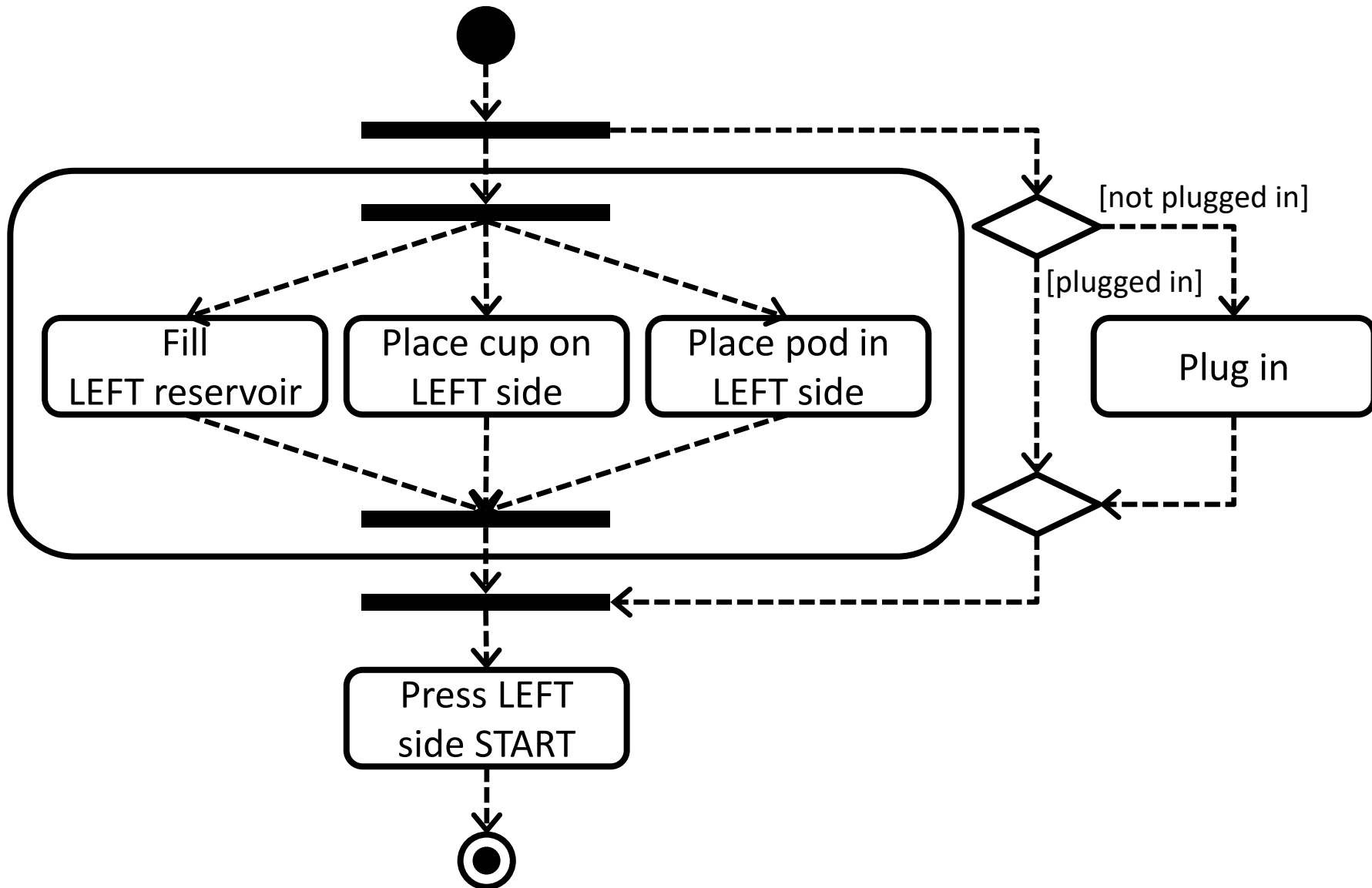
Kávéfőzés



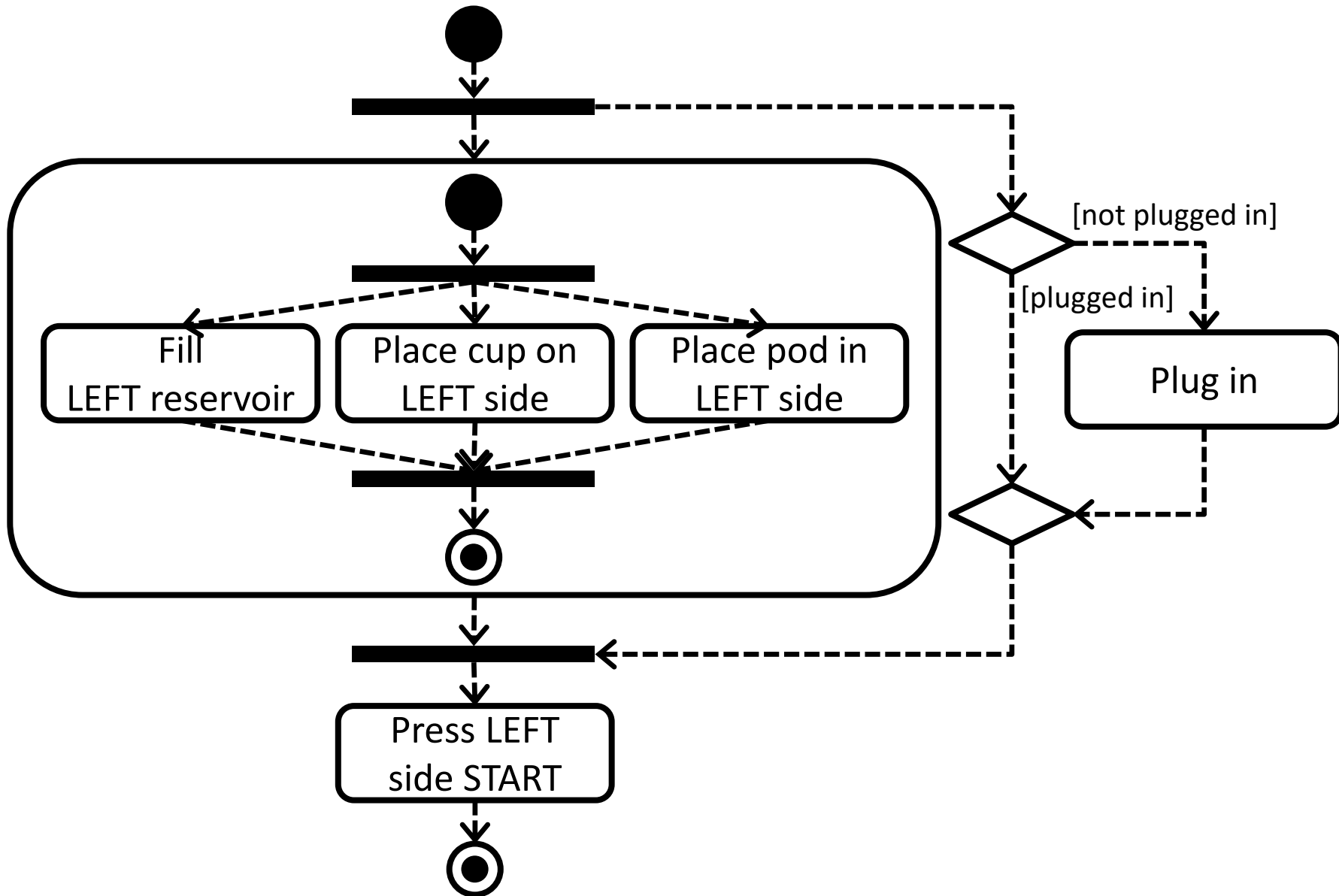
Kávéfőzés



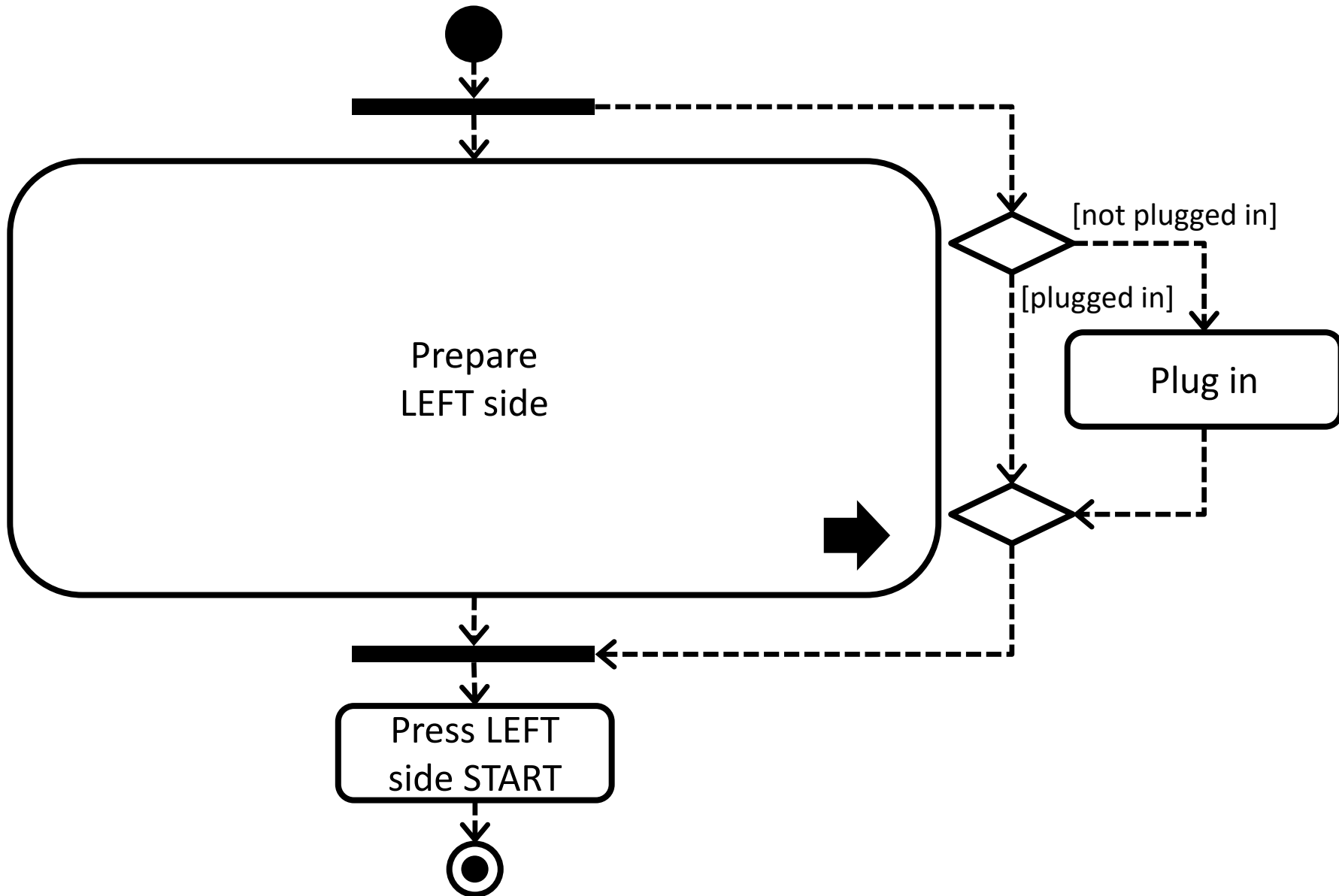
Kávéfőzés



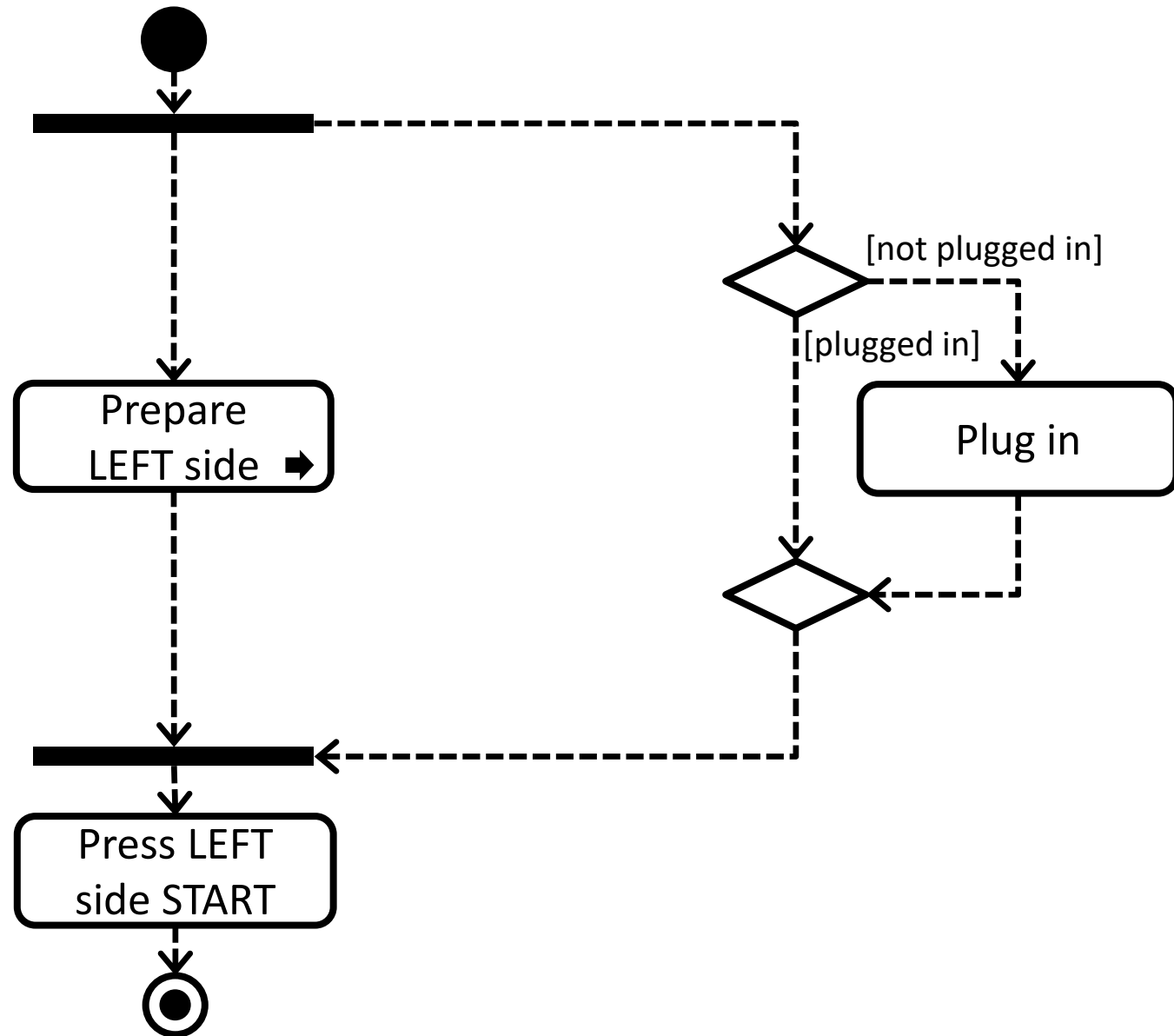
Kávéfőzés



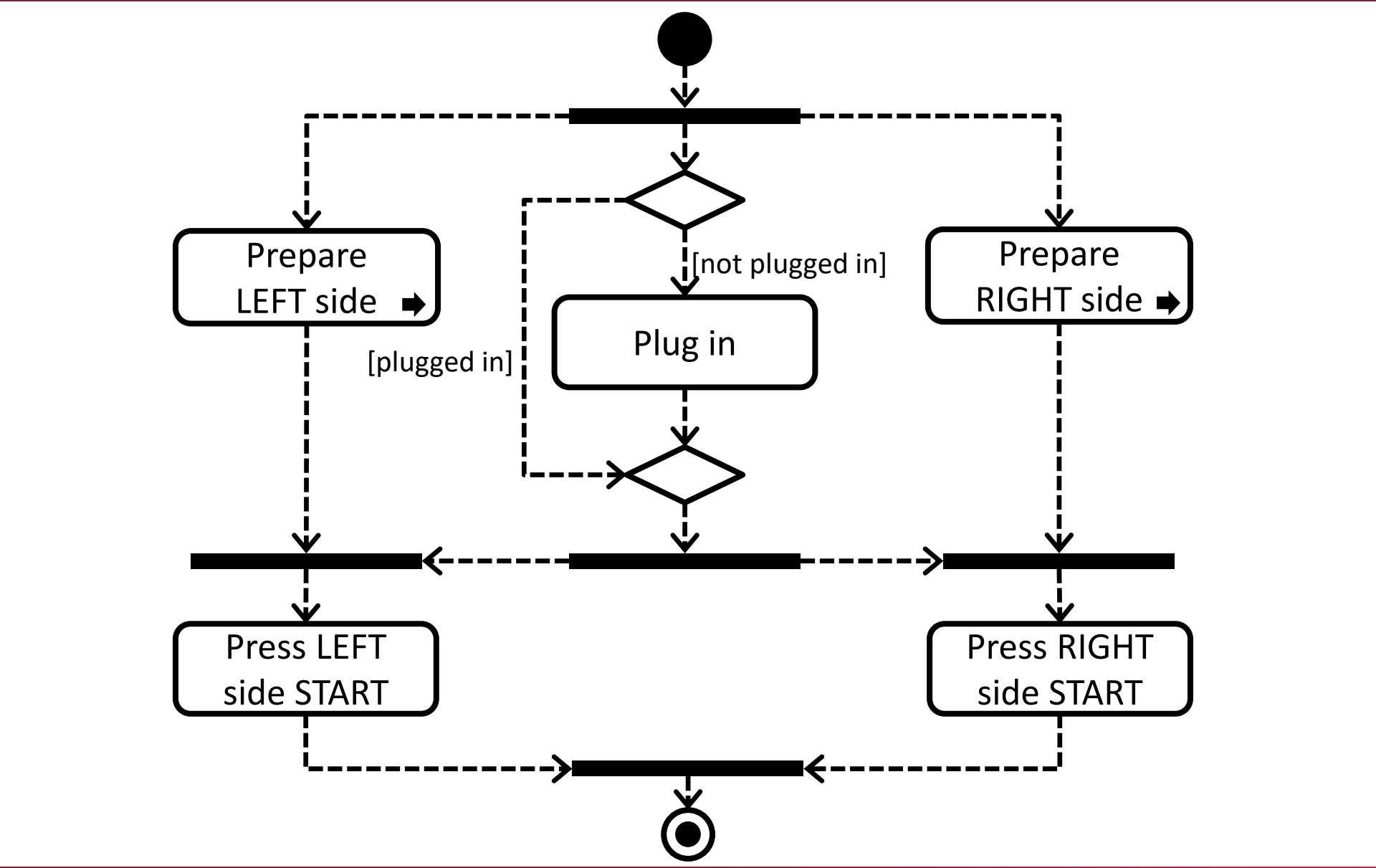
Kávéfőzés



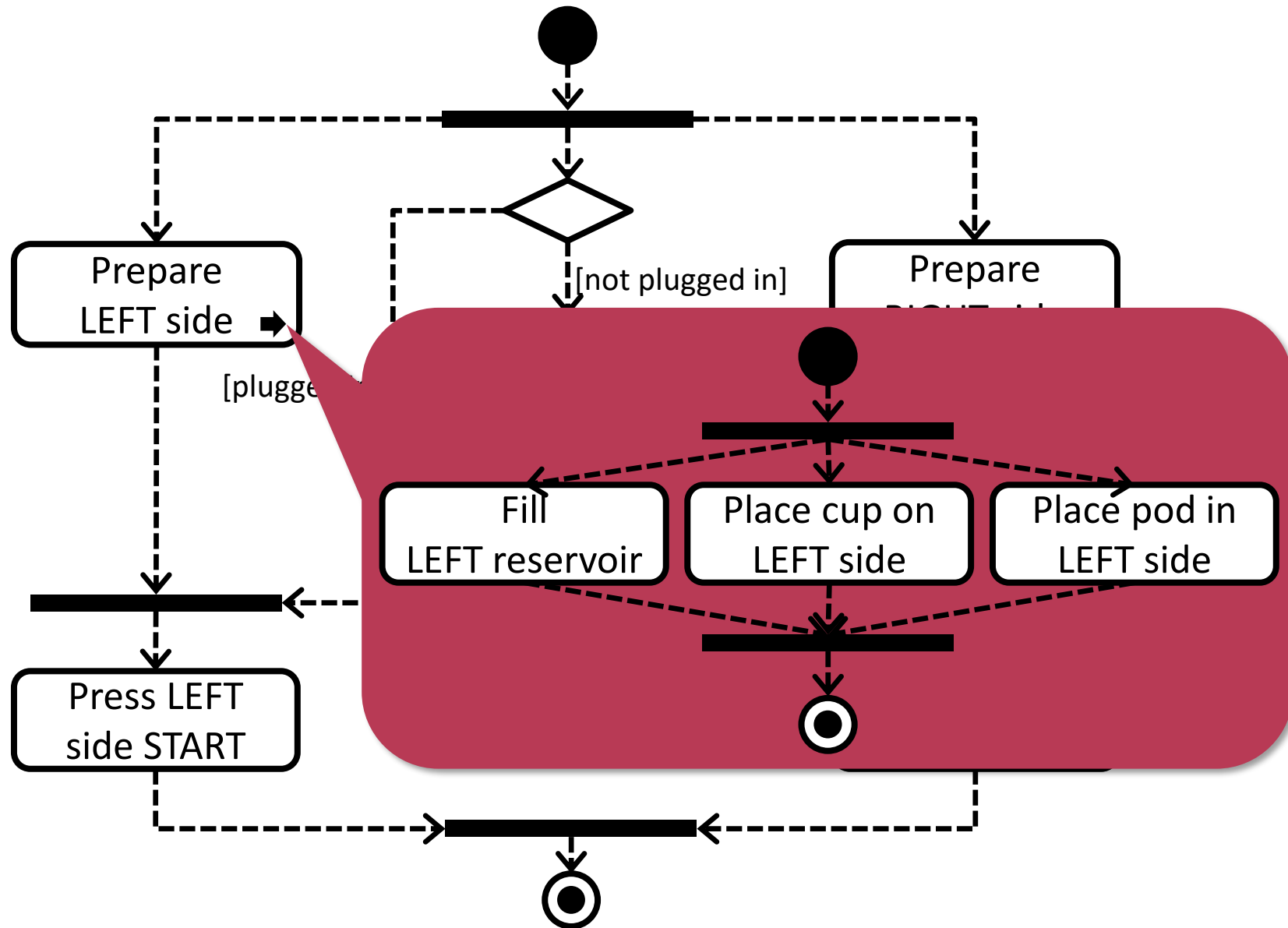
Kávéfőzés



Kávéfőzés



Kávéfőzés



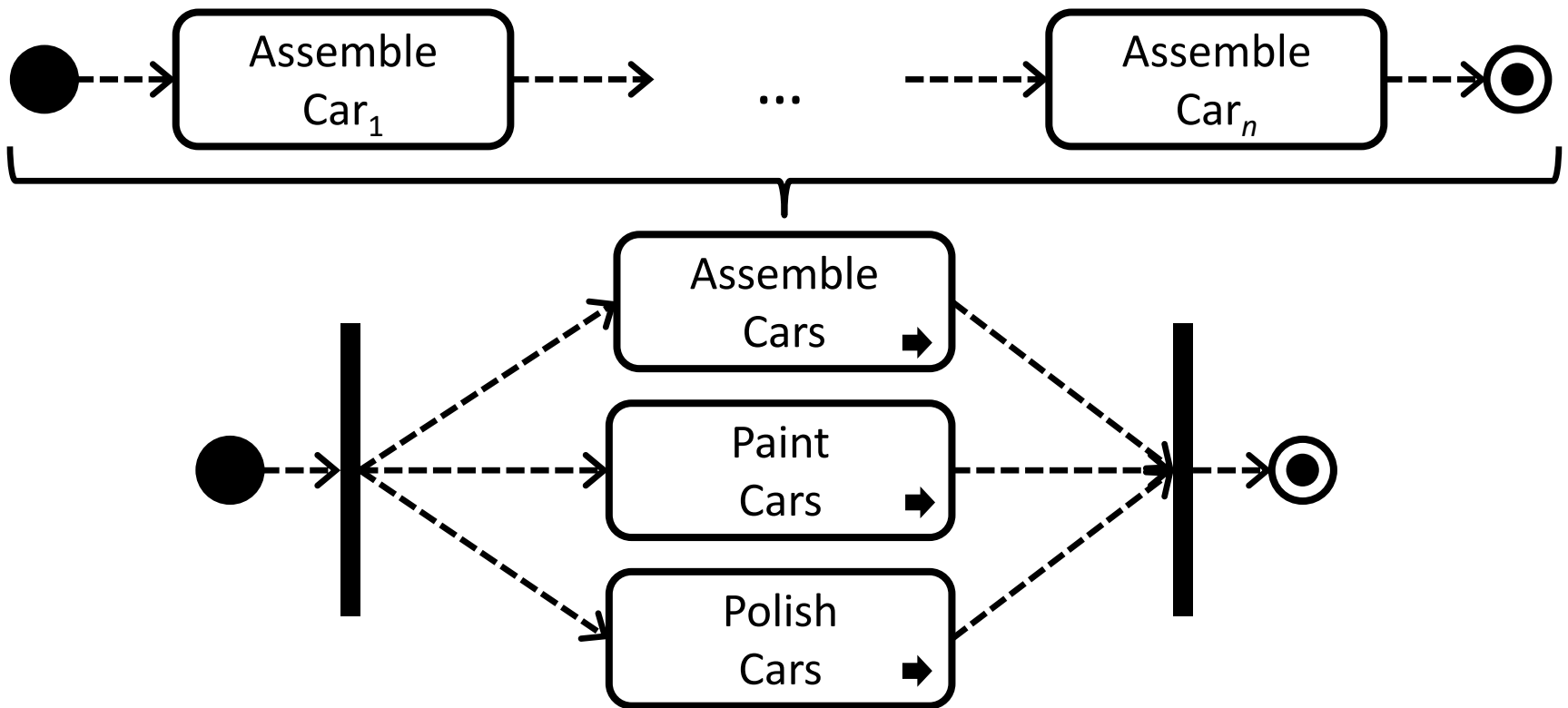
Különböző szempontok szerinti modellezés



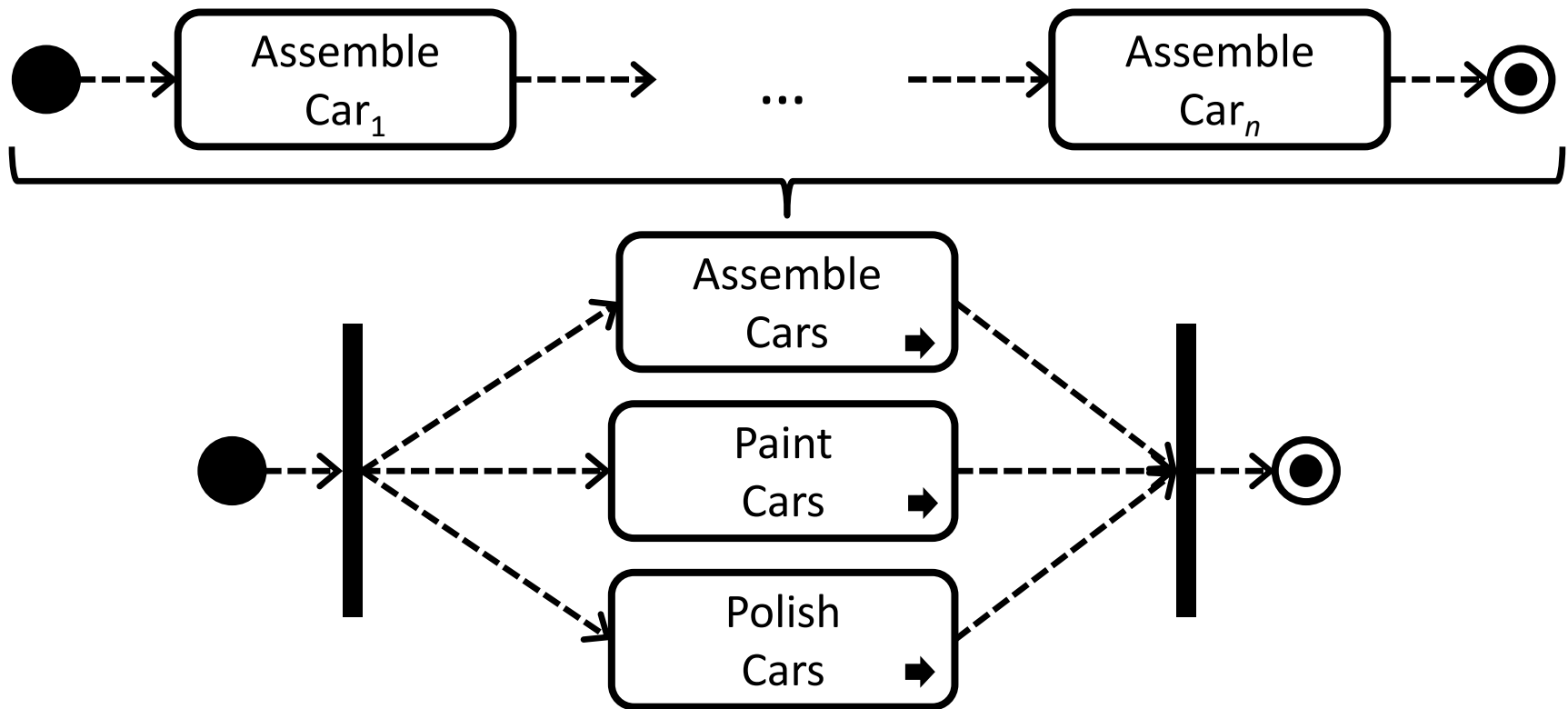
Mi történik egy autóval?



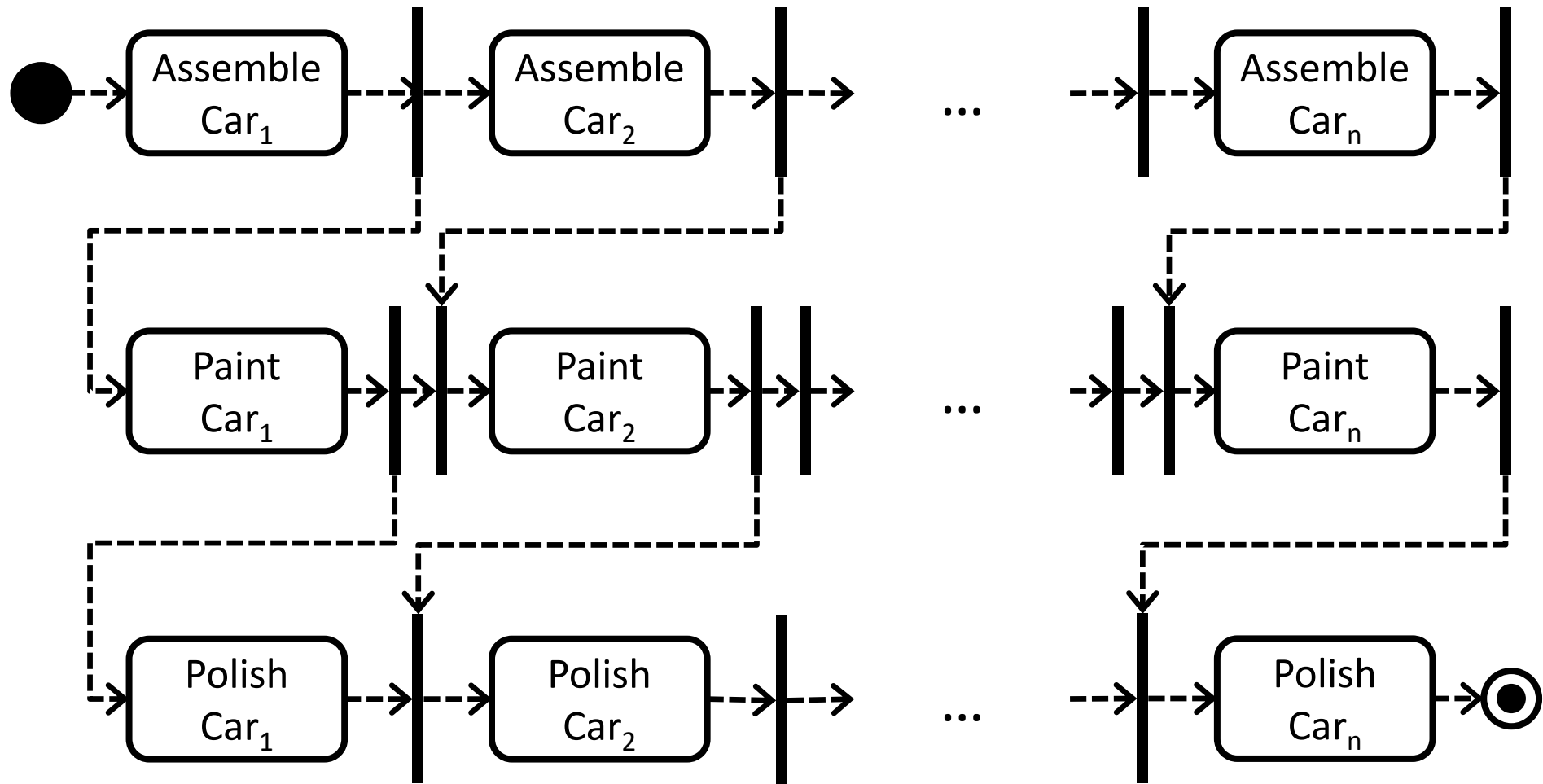
Mi történik a gyártósoron?



Különböző szempontok szerinti modellezés

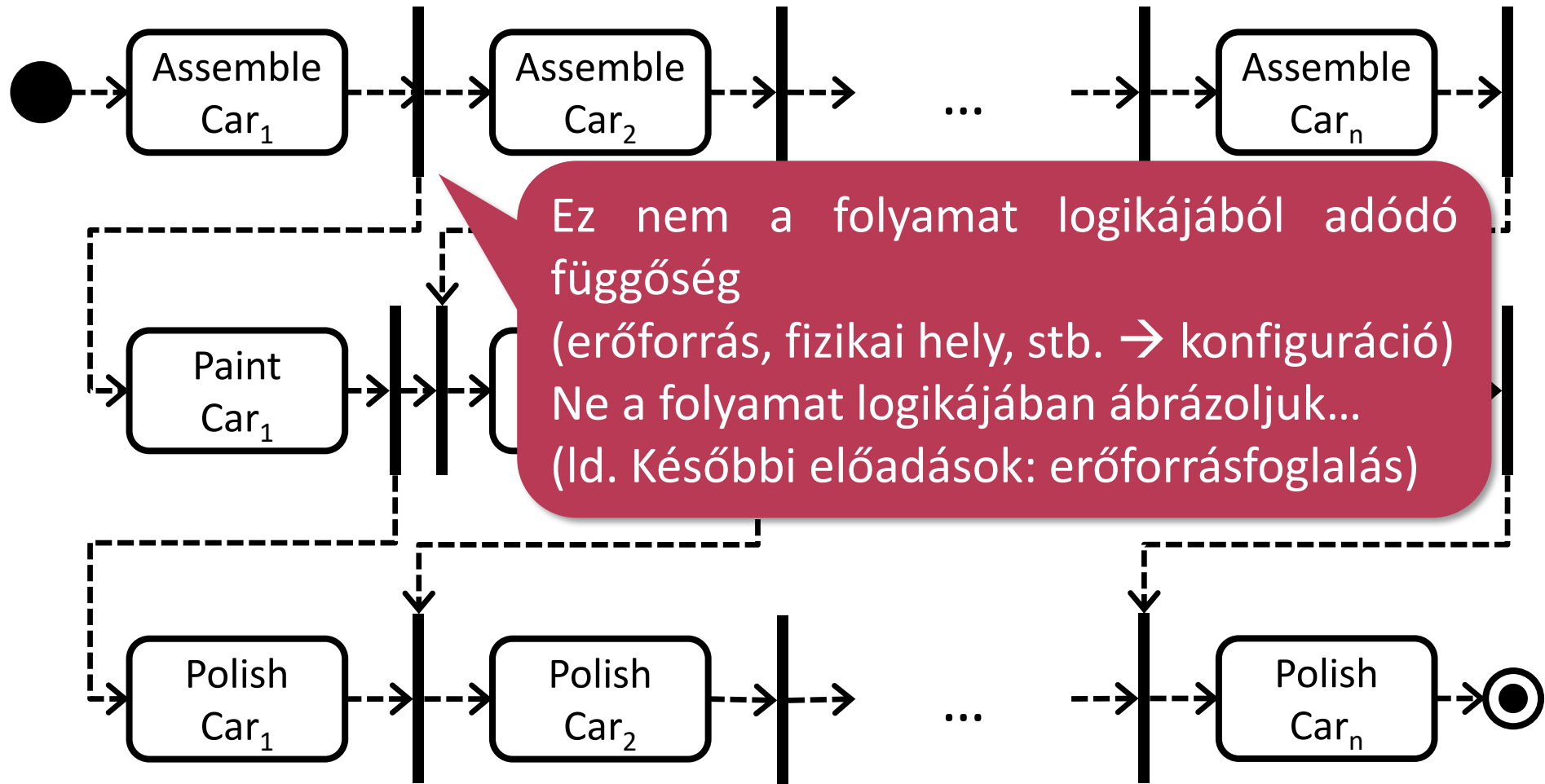


Együttes nézet



- Mindent tartalmaz, de nem túl praktikus

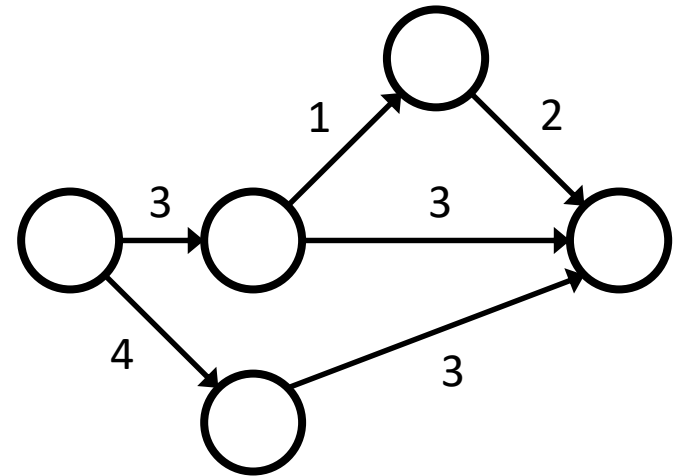
Együttes nézet



- Mindent tartalmaz, de nem túl praktikus

Együttes nézet

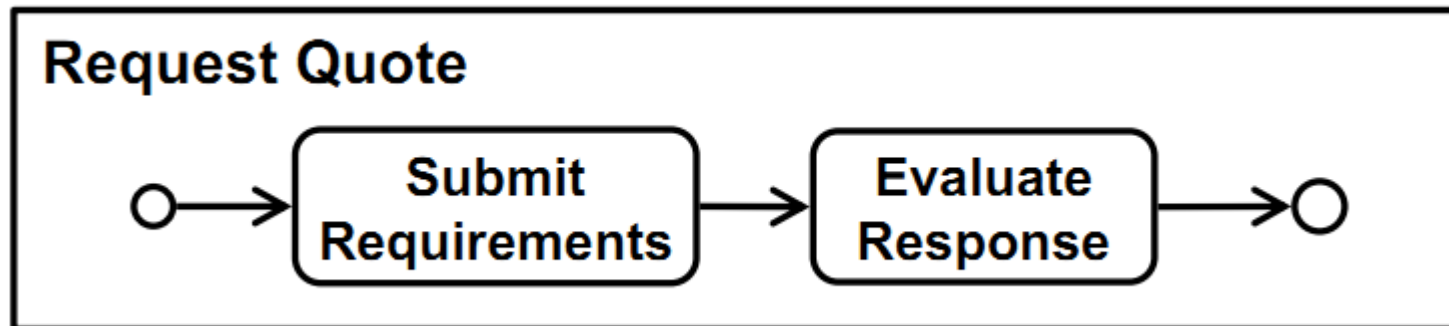
- A 2D fork-join háló nem túl praktikus
 - Aspektusokra (autó és gép életútja) külön folyamat
- Sok fork-join tömörebben? → PERT chart
 - Program Evaluation and Review Technique
 - Végrehajtási idő analízisére való, ld. BSz 2
 - (Ez viszont elágazást nem tud)



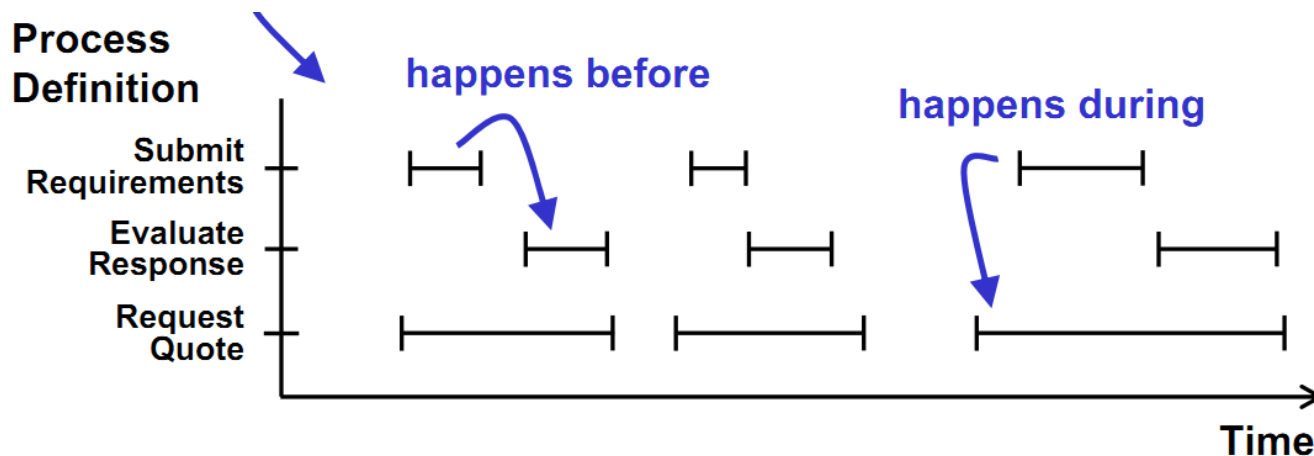
ÜZLETI FOLYAMATOK VÉGREHAJTÁSA

Folyamatok szemantikája

- Modellezés szempontjából

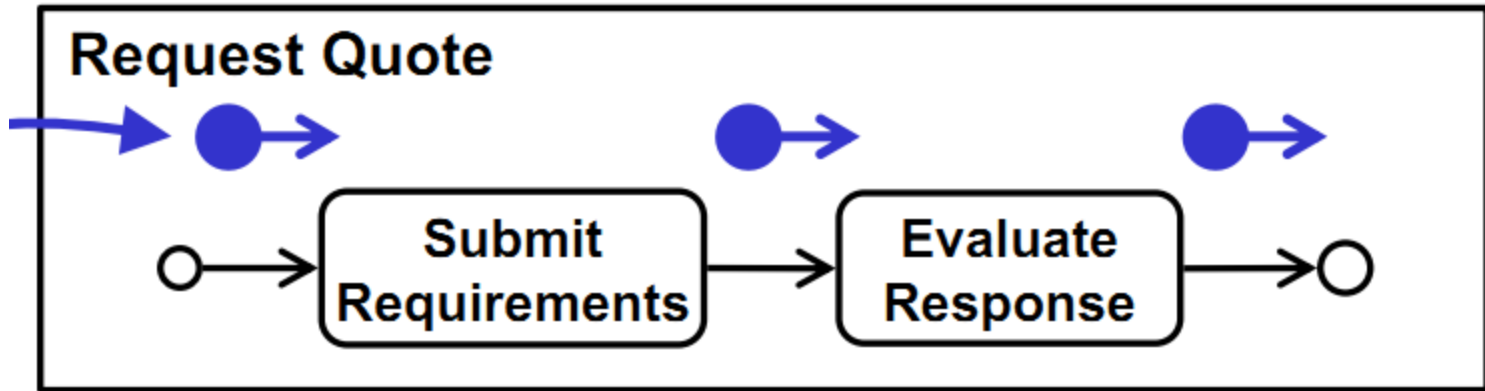


- Az elvárt működés



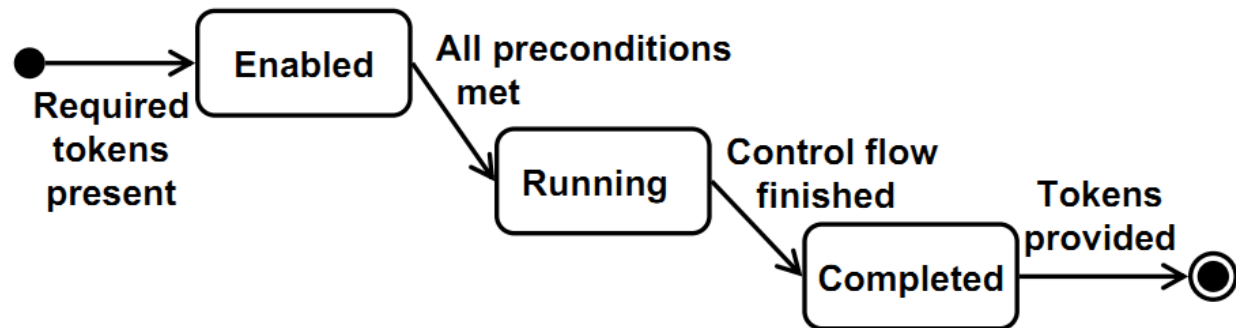
Folyamat végrehajtása

■ Tokenáramlás

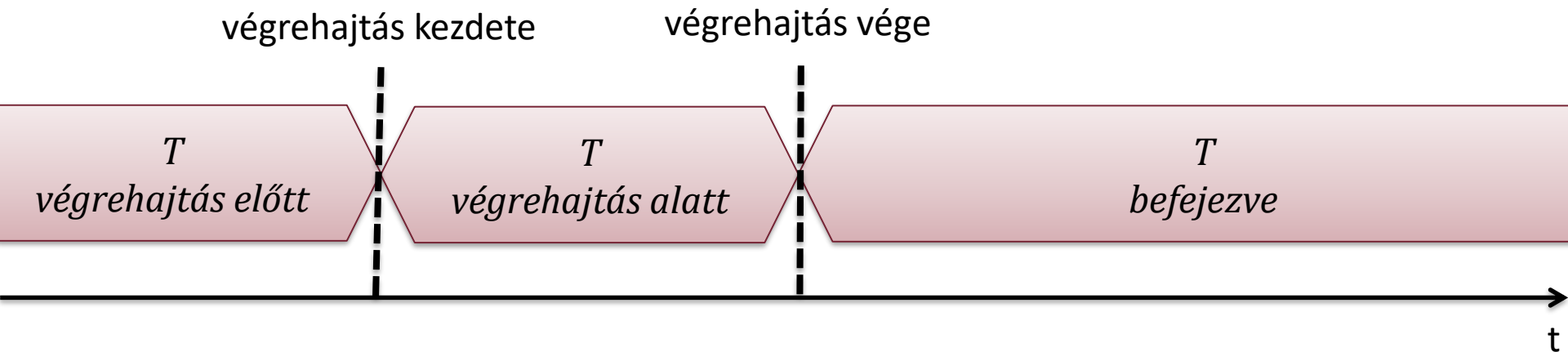


■ A folyamat állapota

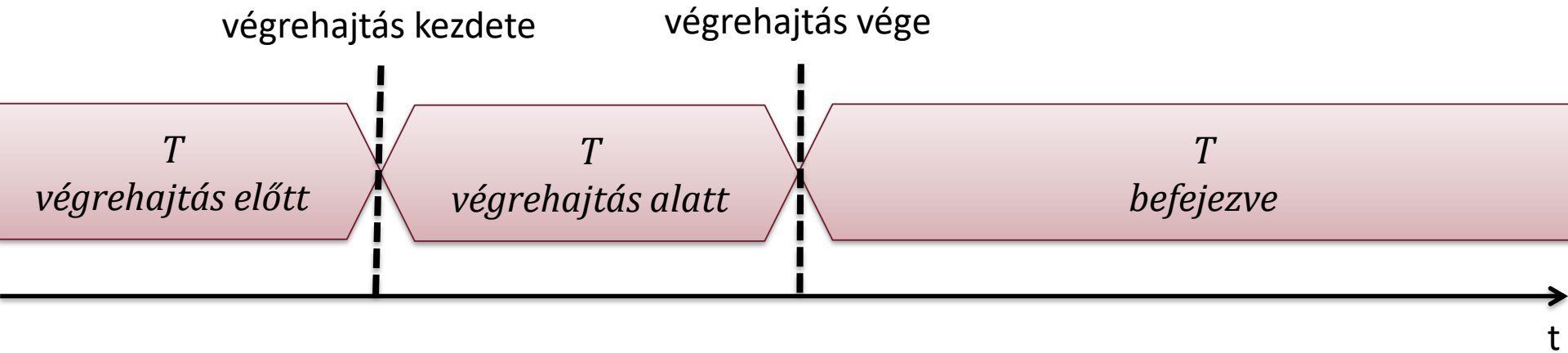
**State
Machine**



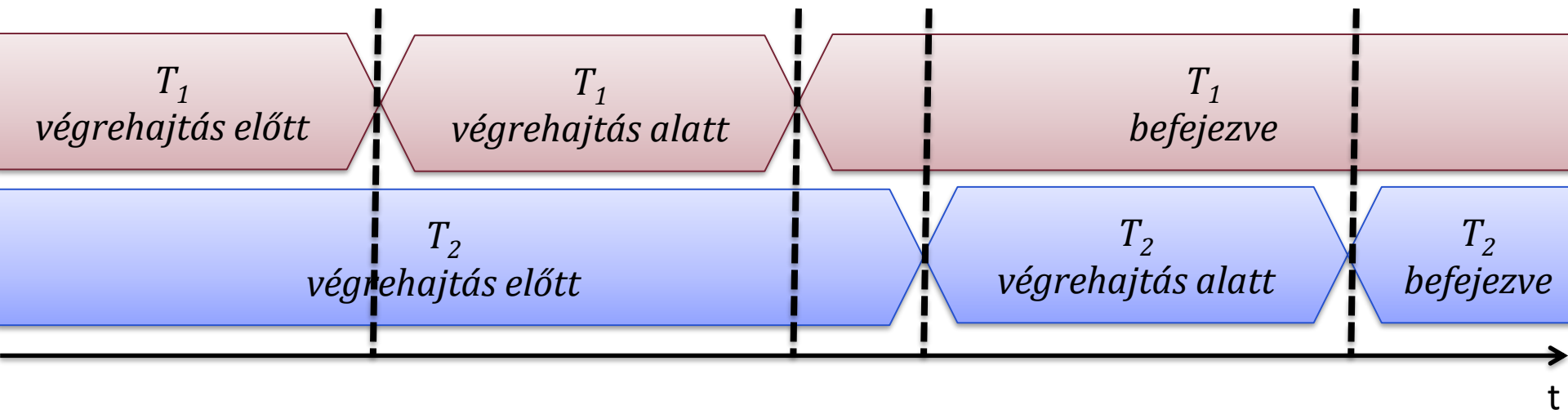
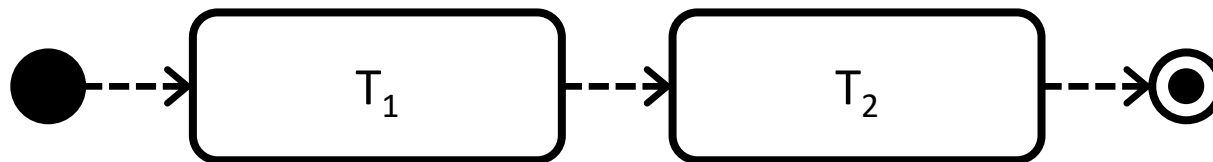
Elemi tevékenység állapotai



Elemi tevékenység állapotai

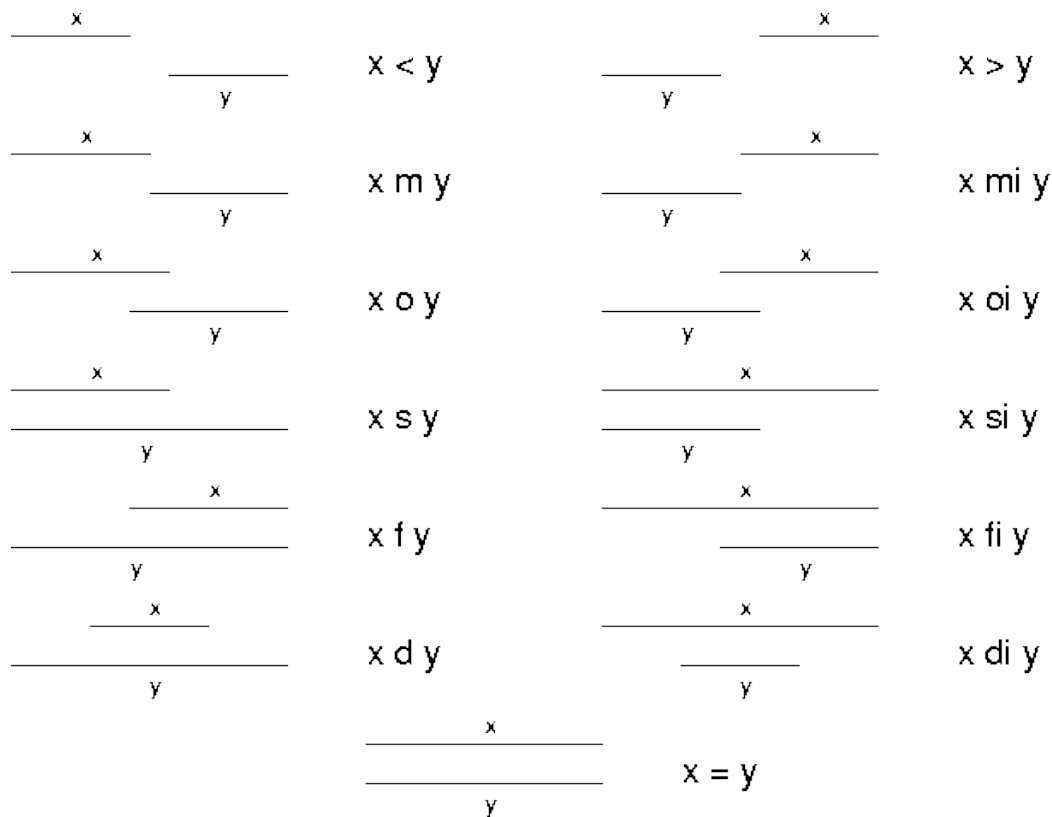


Folyamat állapotai



Háttér: matematikai modell

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



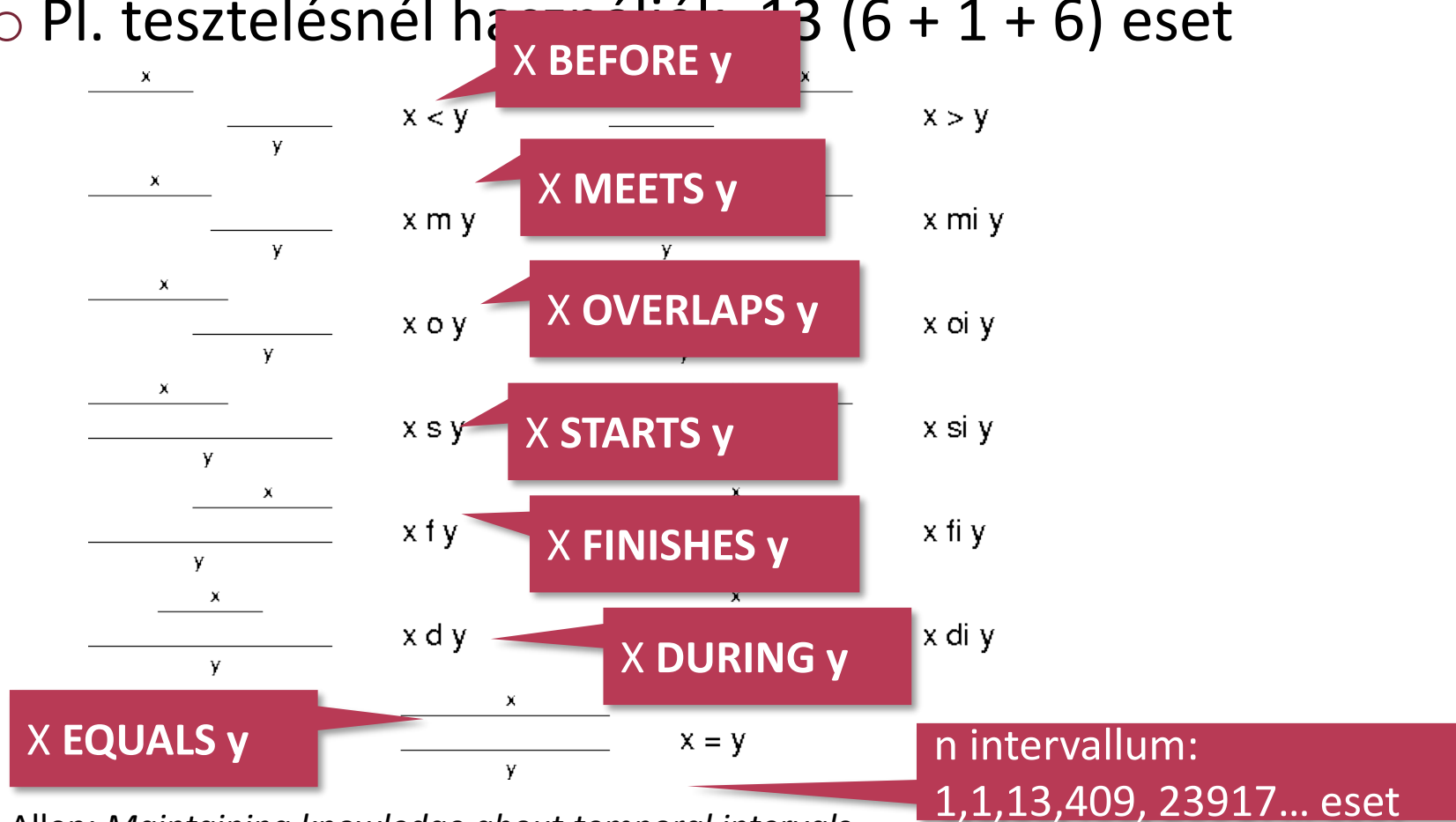
James F. Allen: *Maintaining knowledge about temporal intervals.*

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 használható 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

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.

Tartalom

Ismétlés, kitekintés



Folyamatmodellezés célja



Folyamatmodellek



Vezérlési folyamat

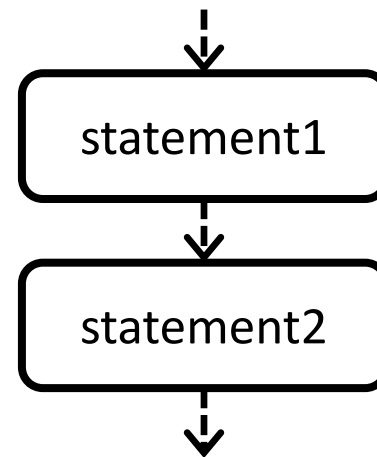


Megvalósítás

Vezérlési folyamat

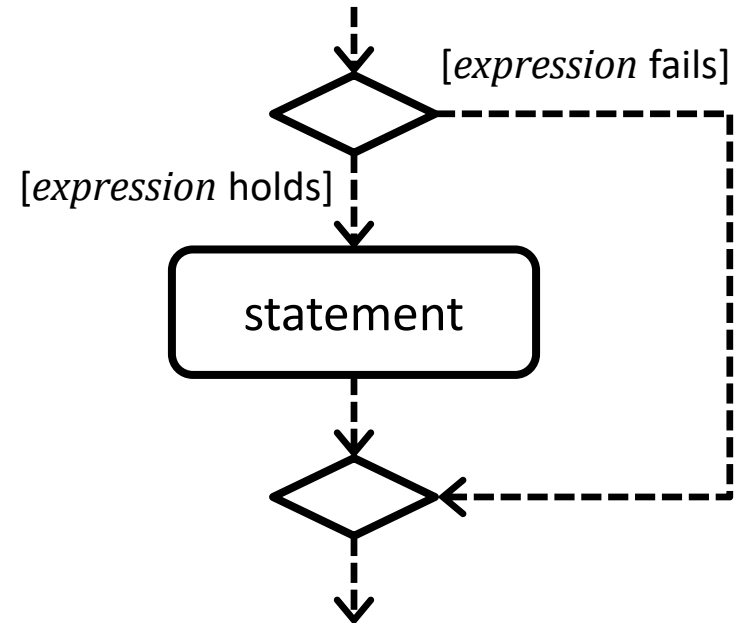
<statement1>

<statement2>



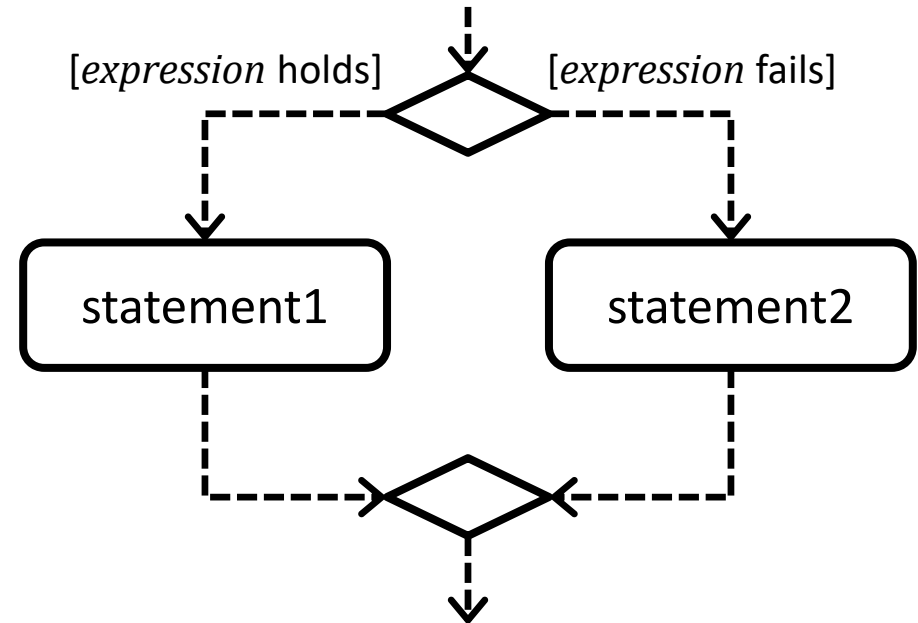
Vezérlési folyamat

if (*<expression>*)
 <statement>



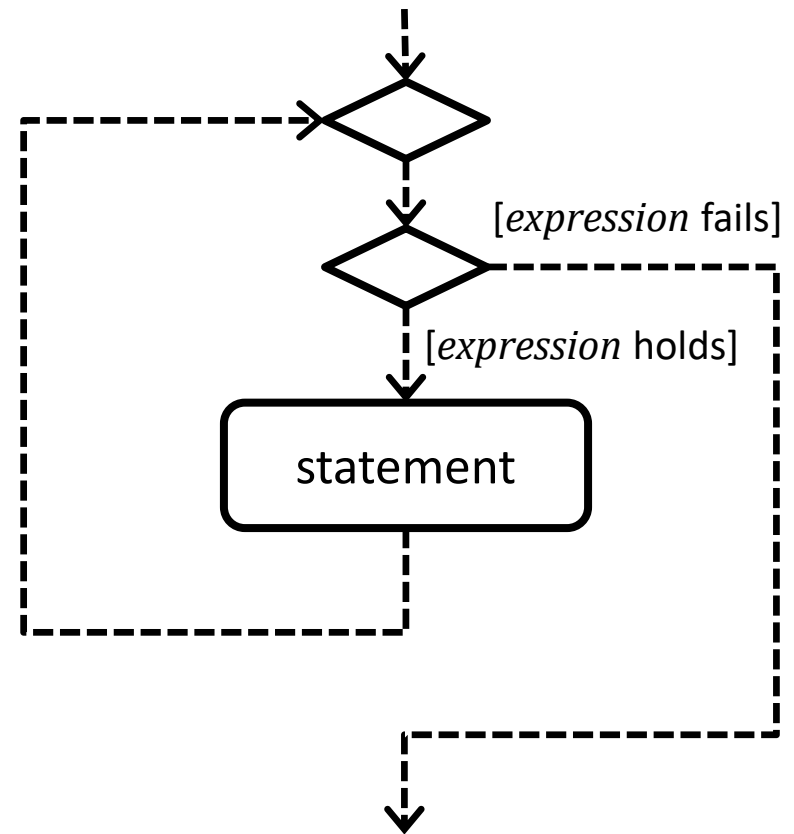
Vezérlési folyamat

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



Vezérlési folyamat

while (*<expression>*)
<statement>

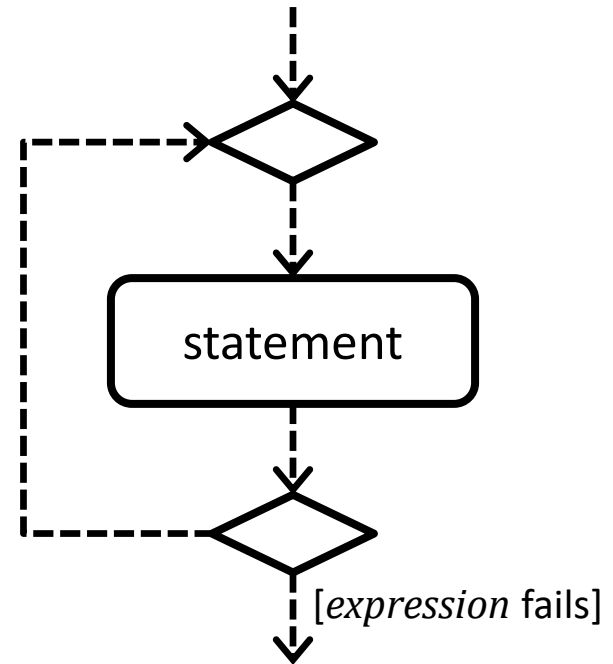


Vezérlési folyamat

do

<statement>

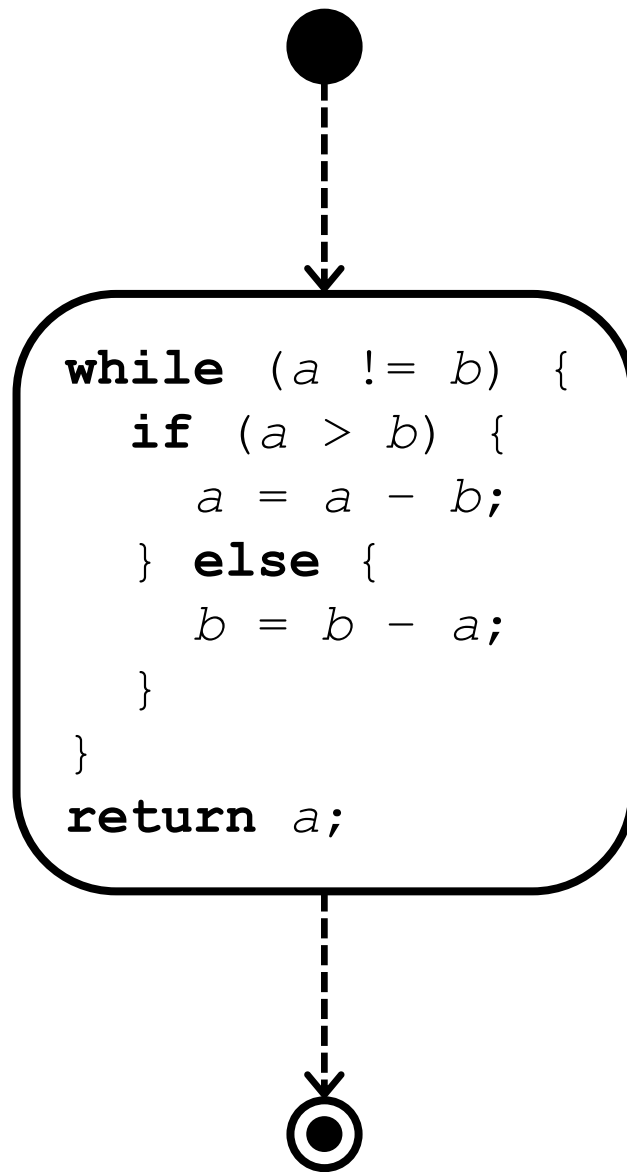
while (*<expression>*)



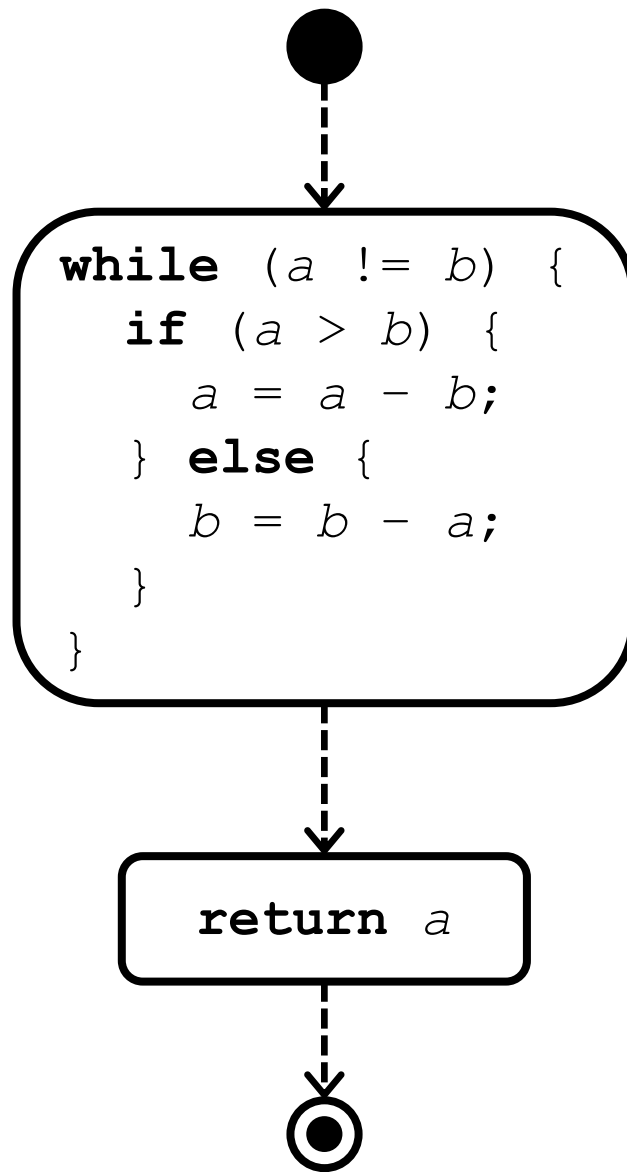
Vezérlési folyamat - példa

```
while ( $a \neq b$ ) {  
    if ( $a > b$ ) {  
         $a = a - b$ ;  
    } else {  
         $b = b - a$ ;  
    }  
}  
return  $a$ ;
```

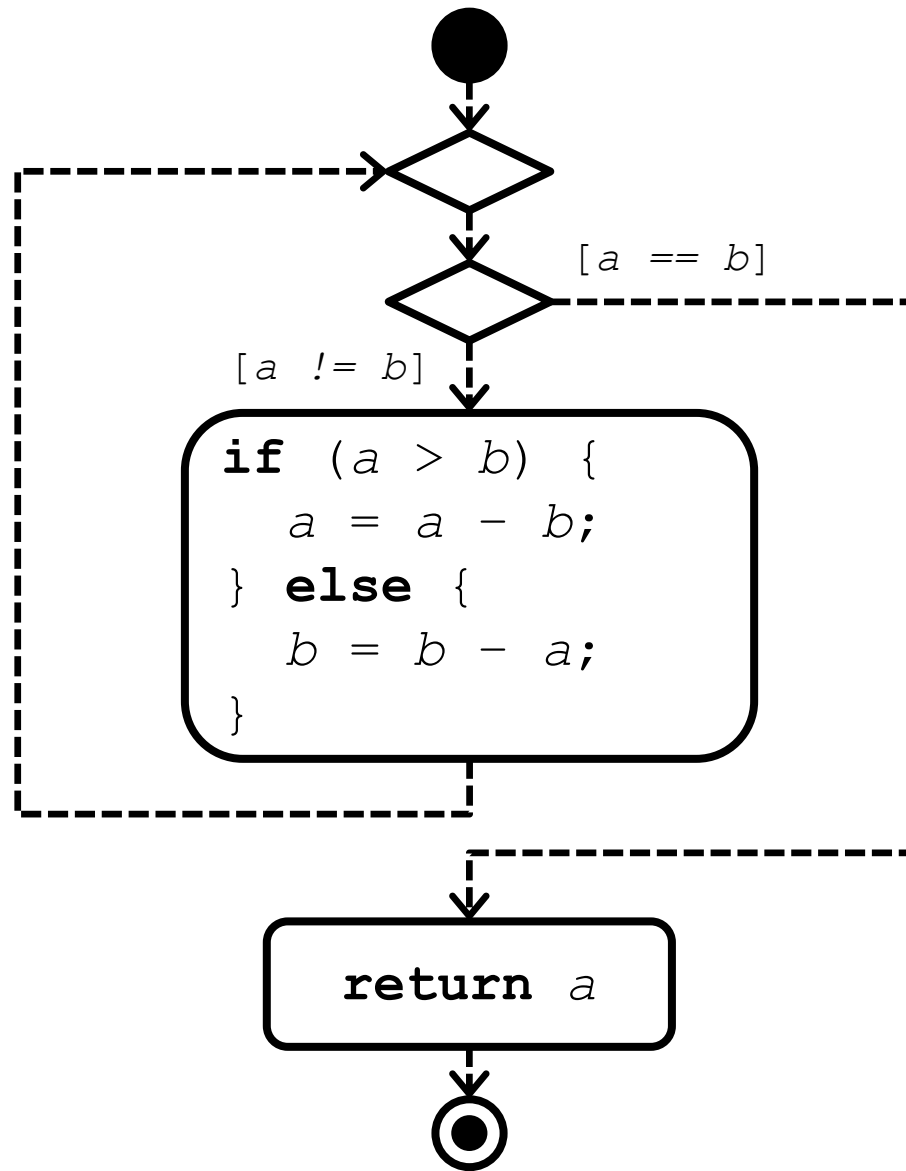
Vezérlési folyamat - példa



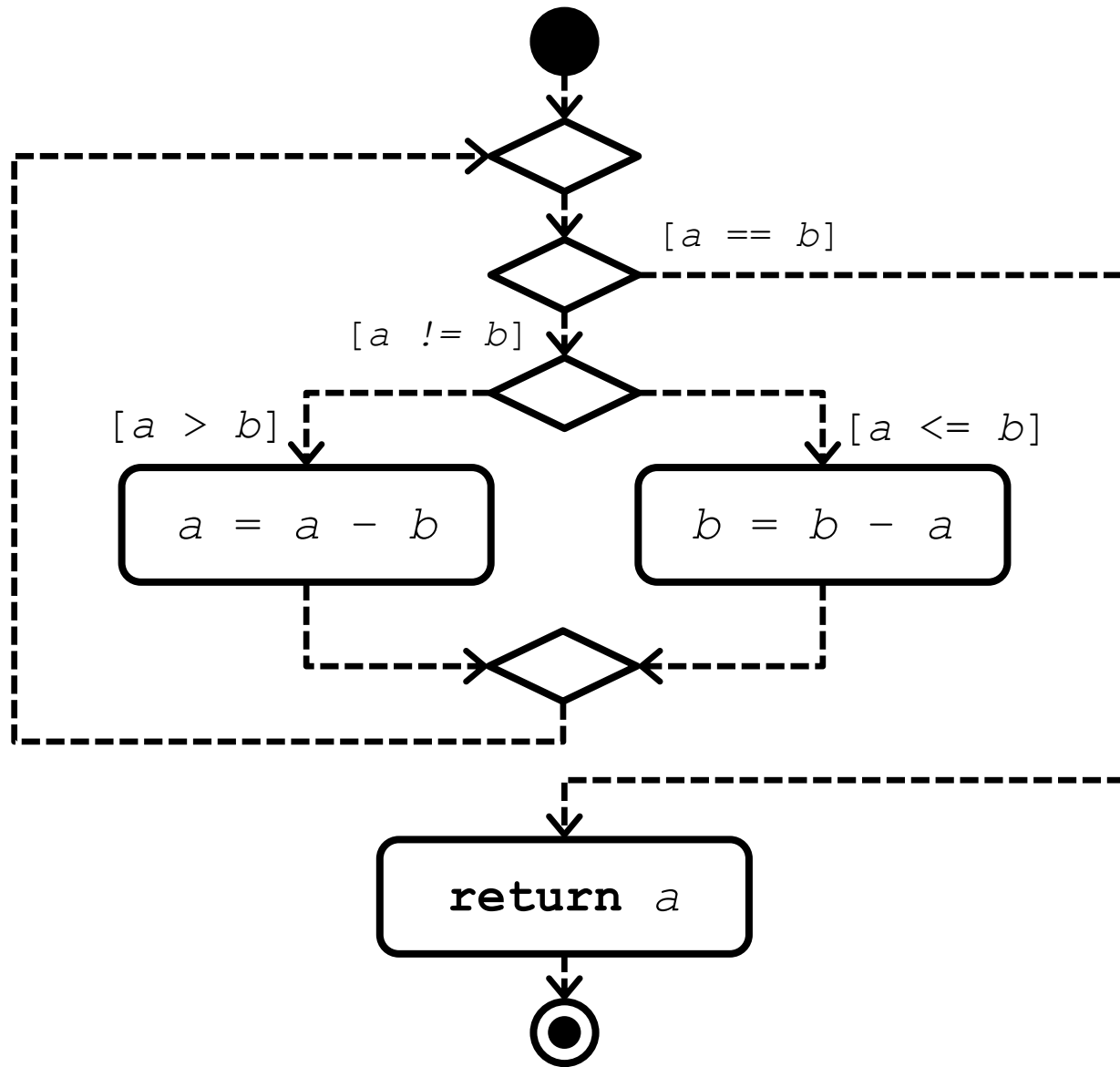
Vezérlési folyamat - példa



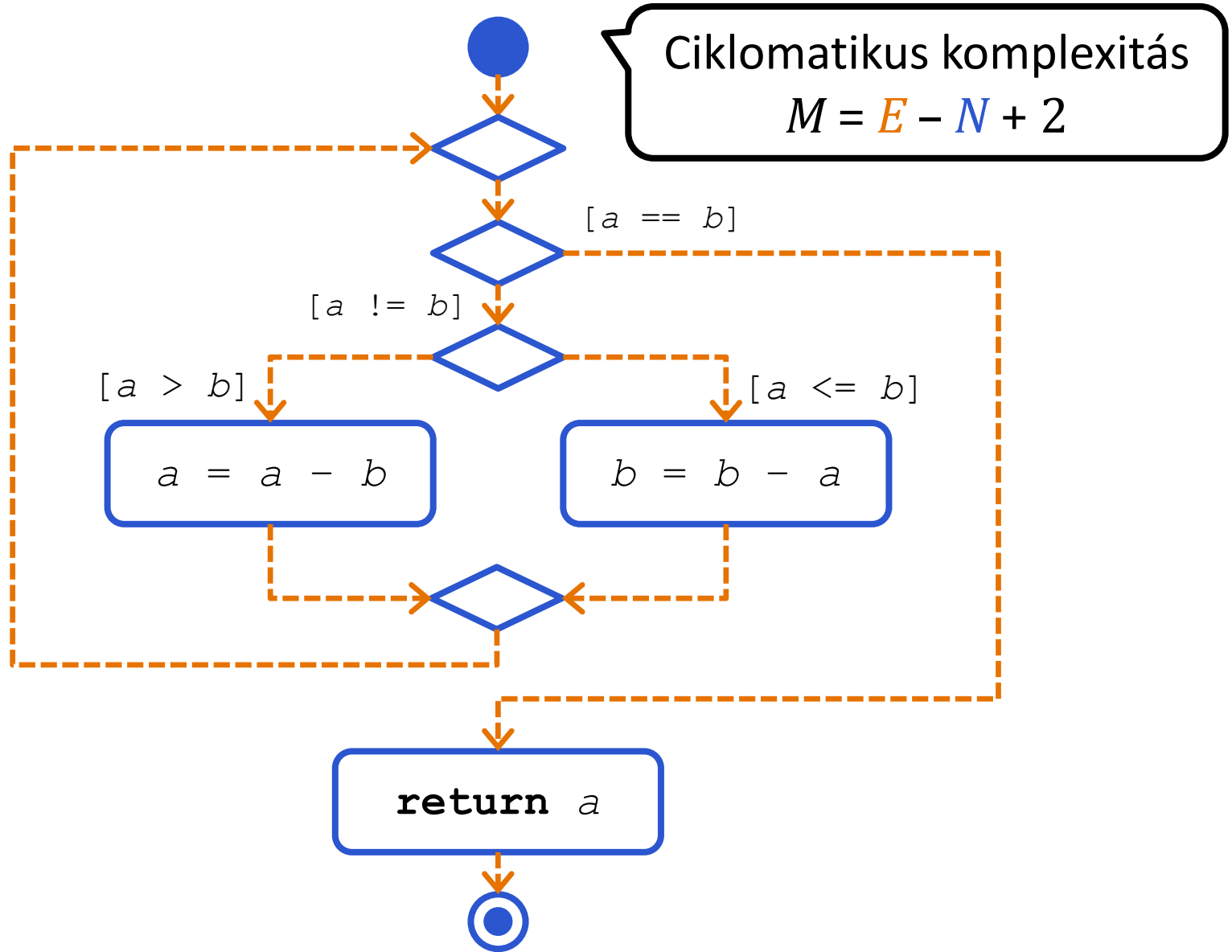
Vezérlési folyamat - példa



Vezérlési folyamat - példa



Vezérlési folyamat - komplexitás



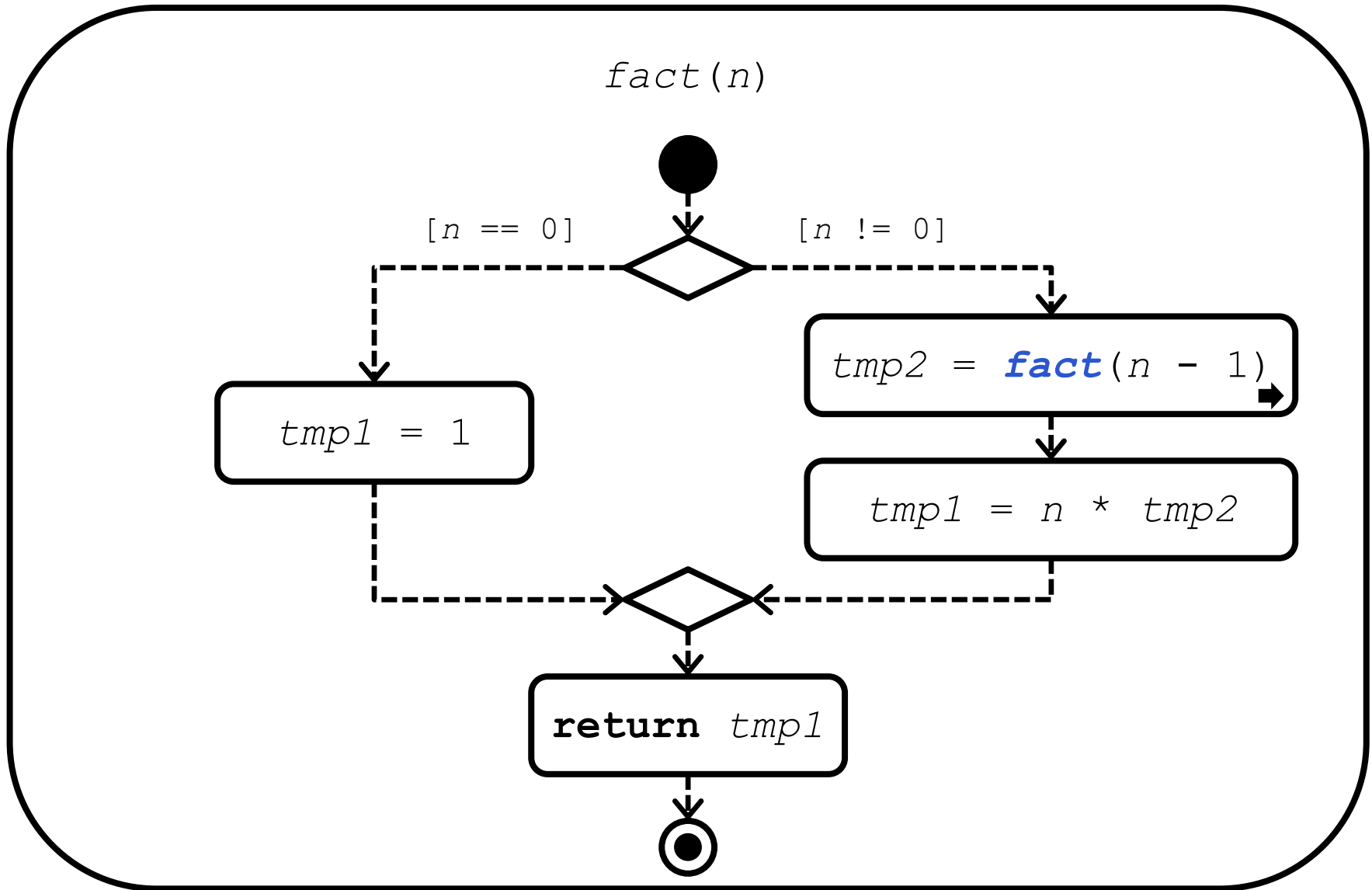
Vezérlési folyamat - rekurzió

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

Vezérlési folyamat - rekurzió

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

Vezérlési folyamat - rekurzió



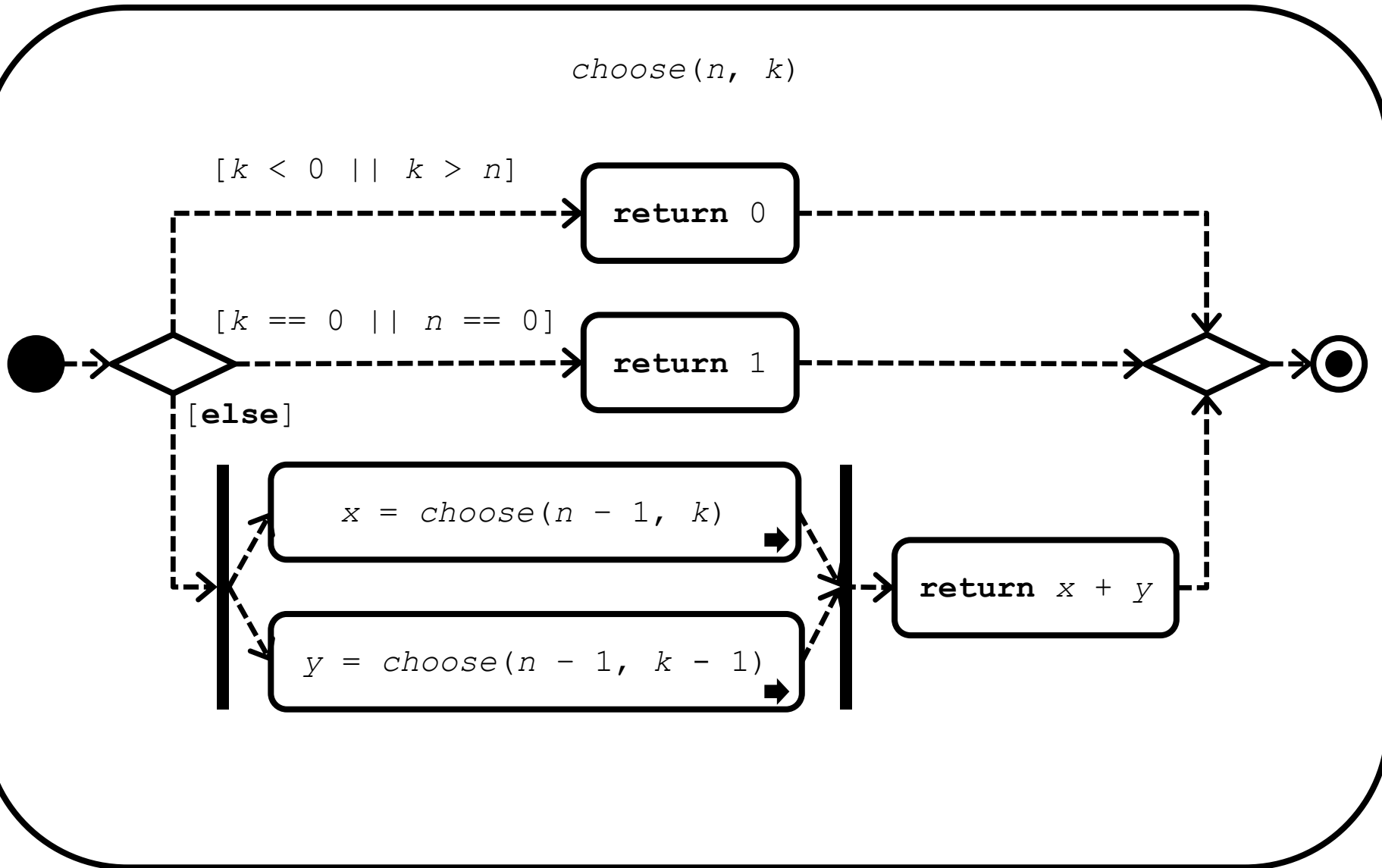
Példa: n alatt a k

```
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;  
    }  
}
```

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

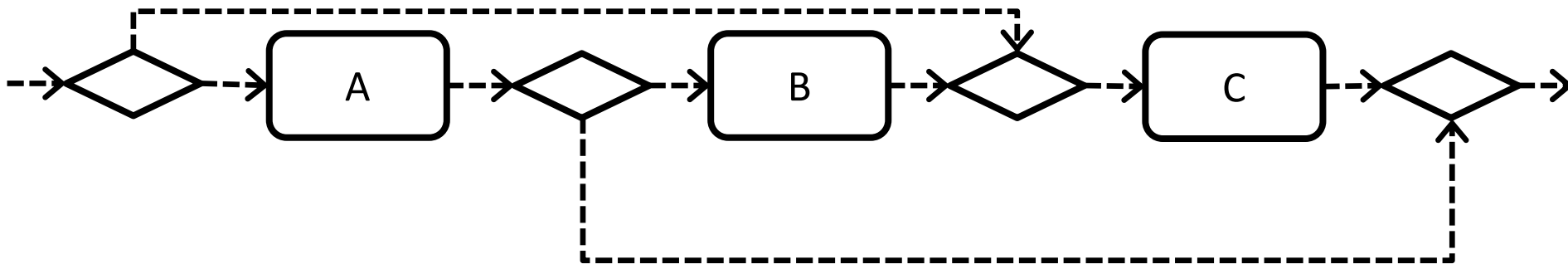
$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

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)

