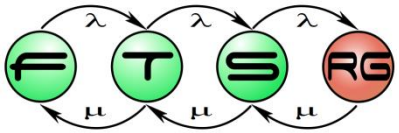
Modellek fejlesztése

Molnár Vince, **Dr. Pataricza András**

Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport



Tartalom

- Motiváció
- Fejlesztési folyamat
- Modellépítés

Motiváció

Fejlesztési folyamat

Modellépítés

MOTIVÁCIÓ

Hogyan modellezzünk?

Út a követelményektől a megvalósulásig

Motiváció

- Modellek fejlesztése
  - Hasonlít a szoftverfejlesztésre
  - Hasonló követelmények
    - Felülről lefelé tervezés
      → fokozatos finomítás
    - Minőségbiztosítás
    - Újrafelhasználhatóság
    - Stb.

- Eszköz: **modellek finomítása**
  - Tervezői döntések fokozatos beépítése a modellekbe
    → egyre konkrétabb tervek

Motiváció

Fejlesztési folyamat

Modellépítés

FEJLESZTÉSI FOLYAMAT

Fejlesztési folyamat

- A (szoftver)fejlesztési folyamat
  - különálló szakaszokra bontása
  - a hatékonyabb tervezés és menedzsment érdekében.
- Szakaszonként meghatározott feladatok
  - Specifikáció
  - Tervezés
  - Implementáció
  - Ellenőrzés
  - Karbantartás
- Különböző stratégiák léteznek

Fejlesztési folyamat: a vízesés modell

Követelmények, specifikáció

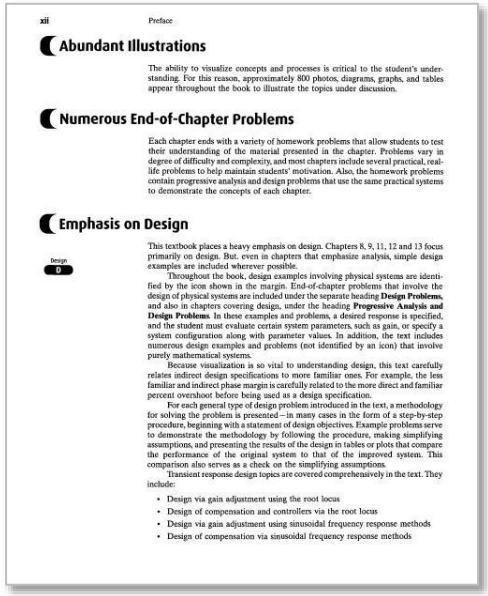
Tervezés

Implementáció

Tesztelés

Karbantartás

Szöveges leírás, félformális modellek, **absztrakt modellek**



xii Preface

Abundant Illustrations

The ability to visualize concepts and processes is critical to the student's understanding. For this reason, approximately 800 photos, diagrams, graphs, and tables appear throughout the book to illustrate the topics under discussion.

Numerous End-of-Chapter Problems

Each chapter ends with a variety of homework problems that allow students to test their understanding of the material presented in the chapter. Problems vary in degree of difficulty and complexity, and most chapters include several practical, real-life problems to help maintain students' motivation. Also, the homework problems contain progressive analysis and design problems that use the same practical systems to demonstrate the concepts of each chapter.

Emphasis on Design

Design

This textbook places a heavy emphasis on design. Chapters 8, 9, 11, 12 and 13 focus primarily on design. But, even in chapters that emphasize analysis, simple design examples are included wherever possible.

Throughout the book, design examples involving physical systems are identified by the icon shown in the margin. End-of-chapter problems that involve the design of physical systems are included under the separate heading **Design Problems**, and also in chapters covering design, under the heading **Progressive Analysis and Design Problems**. In these examples and problems, a desired response is specified, and the student must evaluate certain system parameters, such as gain, or specify a system configuration along with parameter values. In addition, the text includes numerous design examples and problems (not identified by an icon) that involve purely mathematical systems.

Because visualization is so vital to understanding design, this text carefully relates indirect design specifications to more familiar ones. For example, the less familiar and indirect phase margin is carefully related to the more direct and familiar percent overshoot before being used as a design specification.

For each general type of design problem introduced in the text, a methodology for solving the problem is presented – in many cases in the form of a step-by-step procedure, beginning with a statement of design objectives. Example problems serve to demonstrate the methodology by following the procedure, making simplifying assumptions, and presenting the results of the design in tables or plots that compare the performance of the original system to that of the improved system. This comparison also serves as a check on the simplifying assumptions.

Transient response design topics are covered comprehensively in the text. They include:

- Design via gain adjustment using the root locus
- Design of compensation and controllers via the root locus
- Design via gain adjustment using sinusoidal frequency response methods
- Design of compensation via sinusoidal frequency response methods

Fejlesztési folyamat: a vízesés modell

Követelmények, specifikáció

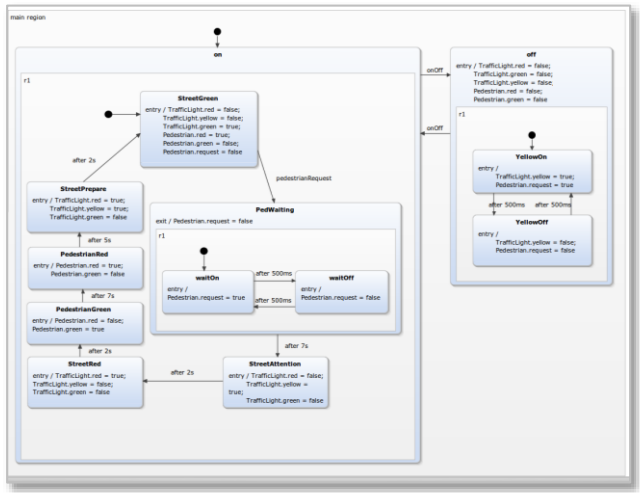
Tervezés

Implementáció

Tesztelés

Karbantartás

Részletes, precíz, finomabb modellek



Fejlesztési folyamat: a vízesés modell

Követelmények, specifikáció

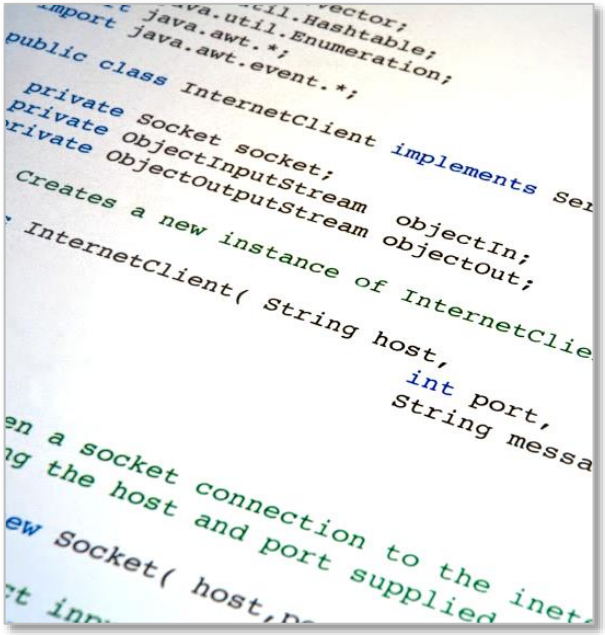
Tervezés

Implementáció

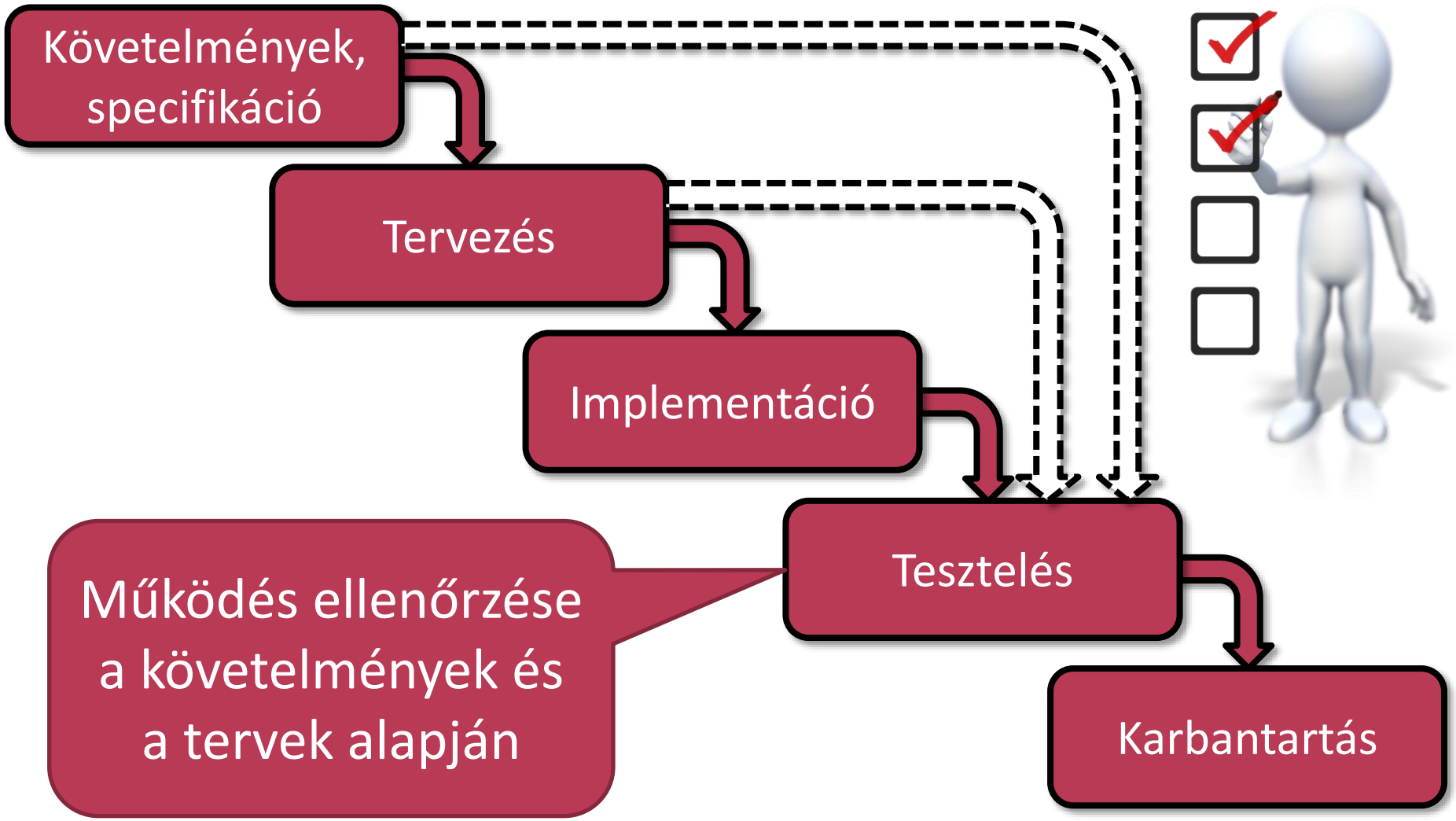
Tesztelés

Karbantartás

Programkód



Fejlesztési folyamat: a vízesés modell



Fejlesztési folyamat: a vízesés modell

Követelmények, specifikáció

Tervezés

Implementáció

Tesztelés

Karbantartás

Hibák javítása, továbbfejlesztés, monitorozás

Modellek életciklusa

Modellek fejlesztése

Szoftver-fejlesztés

Követelmények, specifikáció

Kezdeti modellek

Részletes modellek

Ellenőrzés

Karbantartás

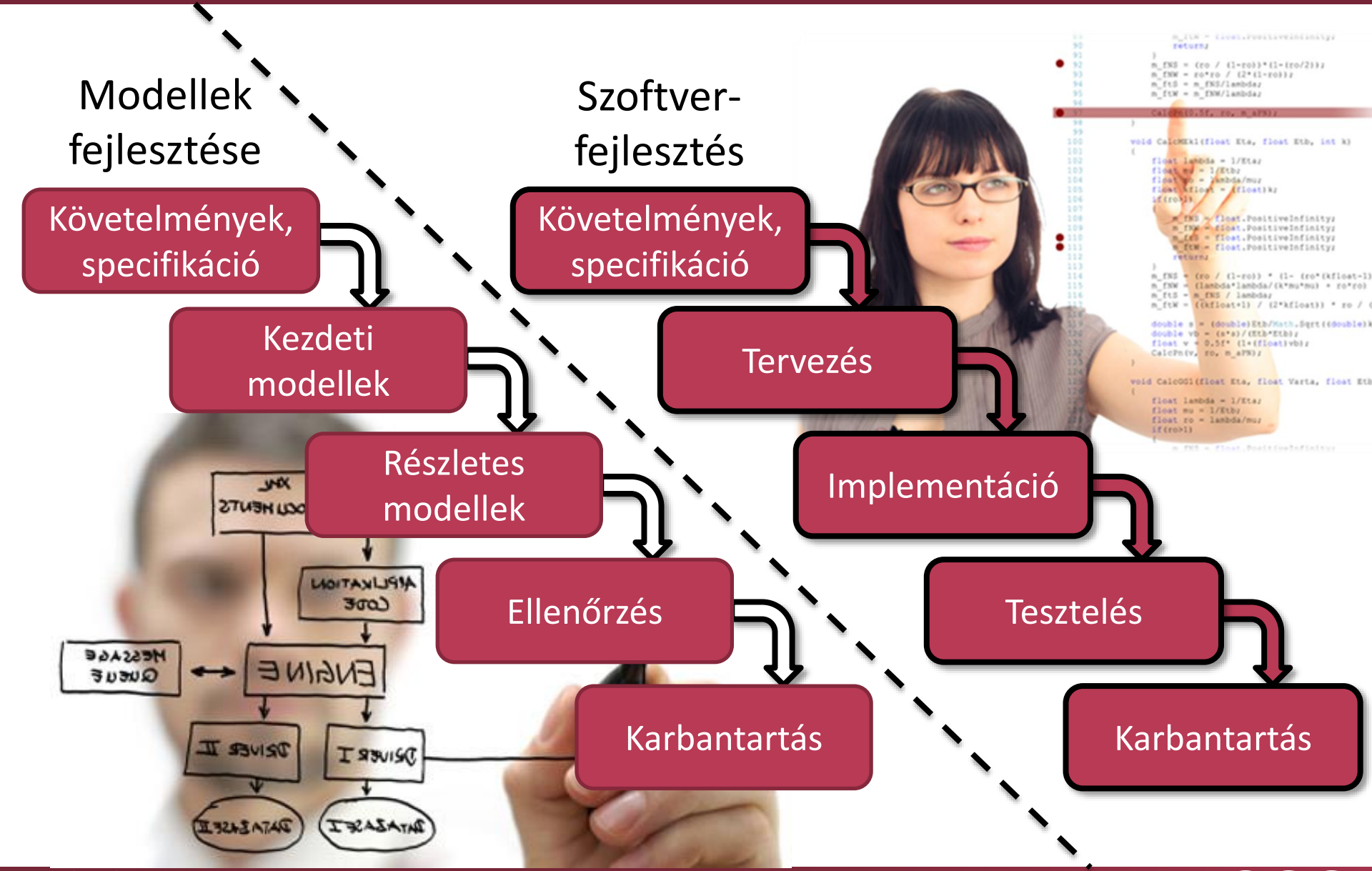
Követelmények, specifikáció

Tervezés

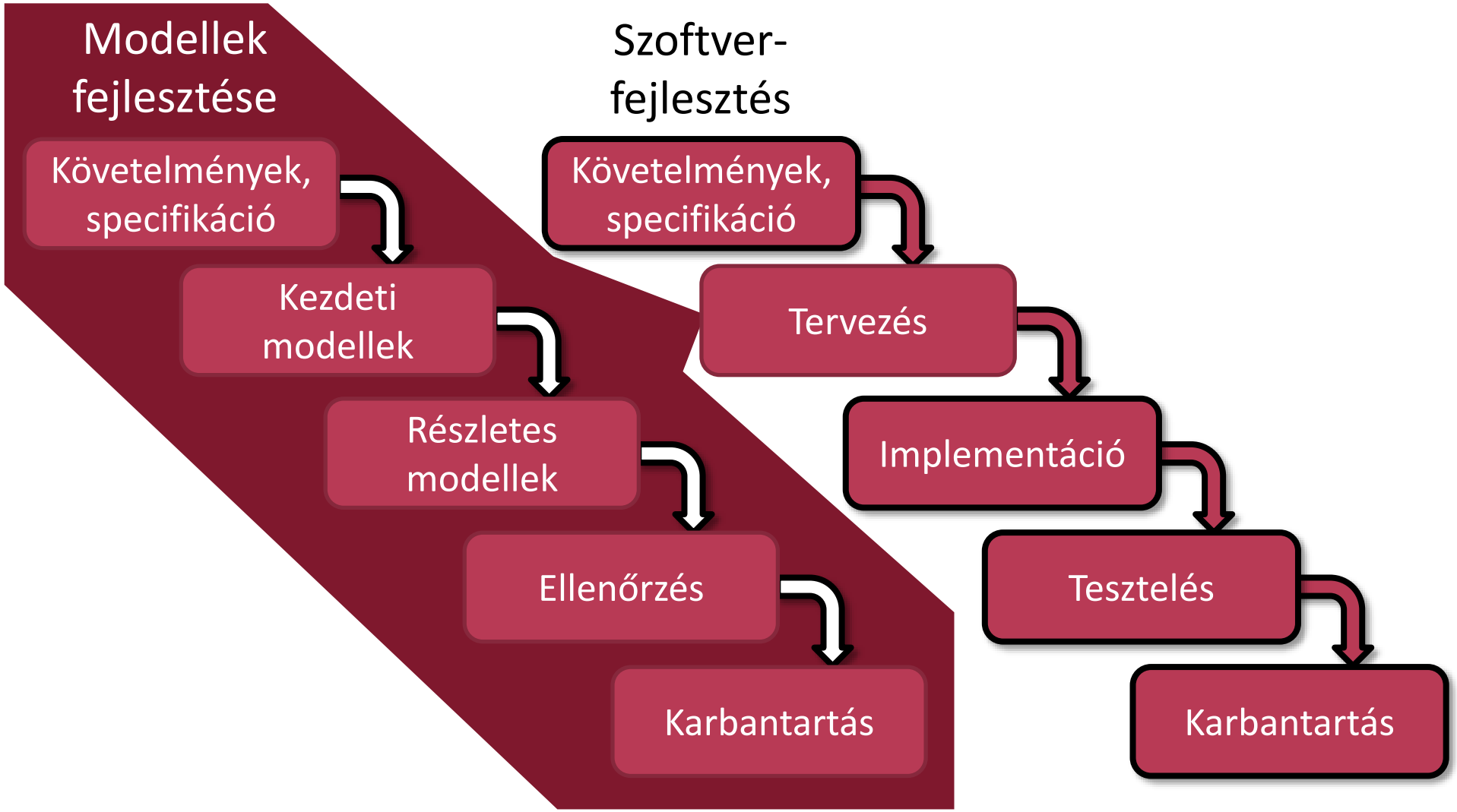
Implementáció

Tesztelés

Karbantartás



Modellek a tervezésben



Modellalapú szoftverfejlesztés

Modellek fejlesztése

Szoftver-fejlesztés

Követelmények, specifikáció

Kezdeti modellek

Részletes modellek

Ellenőrzés

Karbantartás

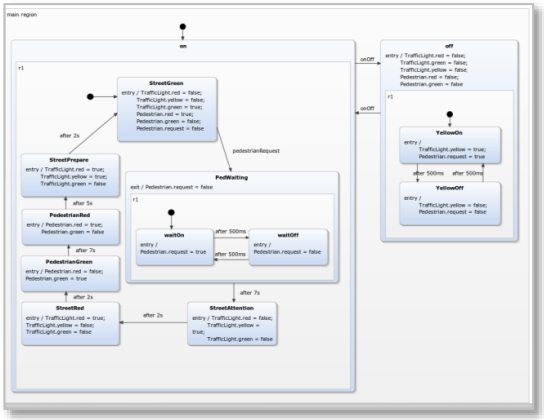
Követelmények, specifikáció

Tervezés

Implementáció

Tesztelés

Karbantartás



Automatikus kódgenerálás

Modellek fejlesztése

Szoftver-fejlesztés

Követelmények, specifikáció

Kezdeti modellek

Részletes modellek

Ellenőrzés

Karbantartás

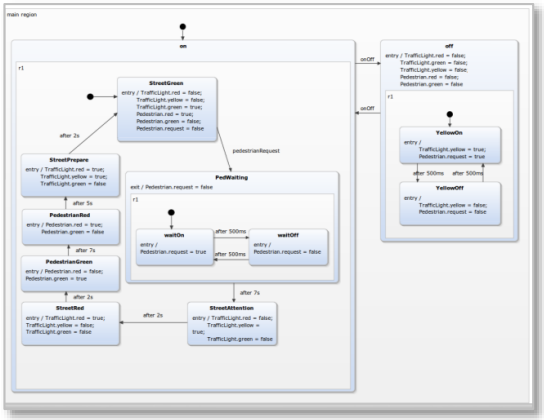
Követelmények, specifikáció

Tervezés

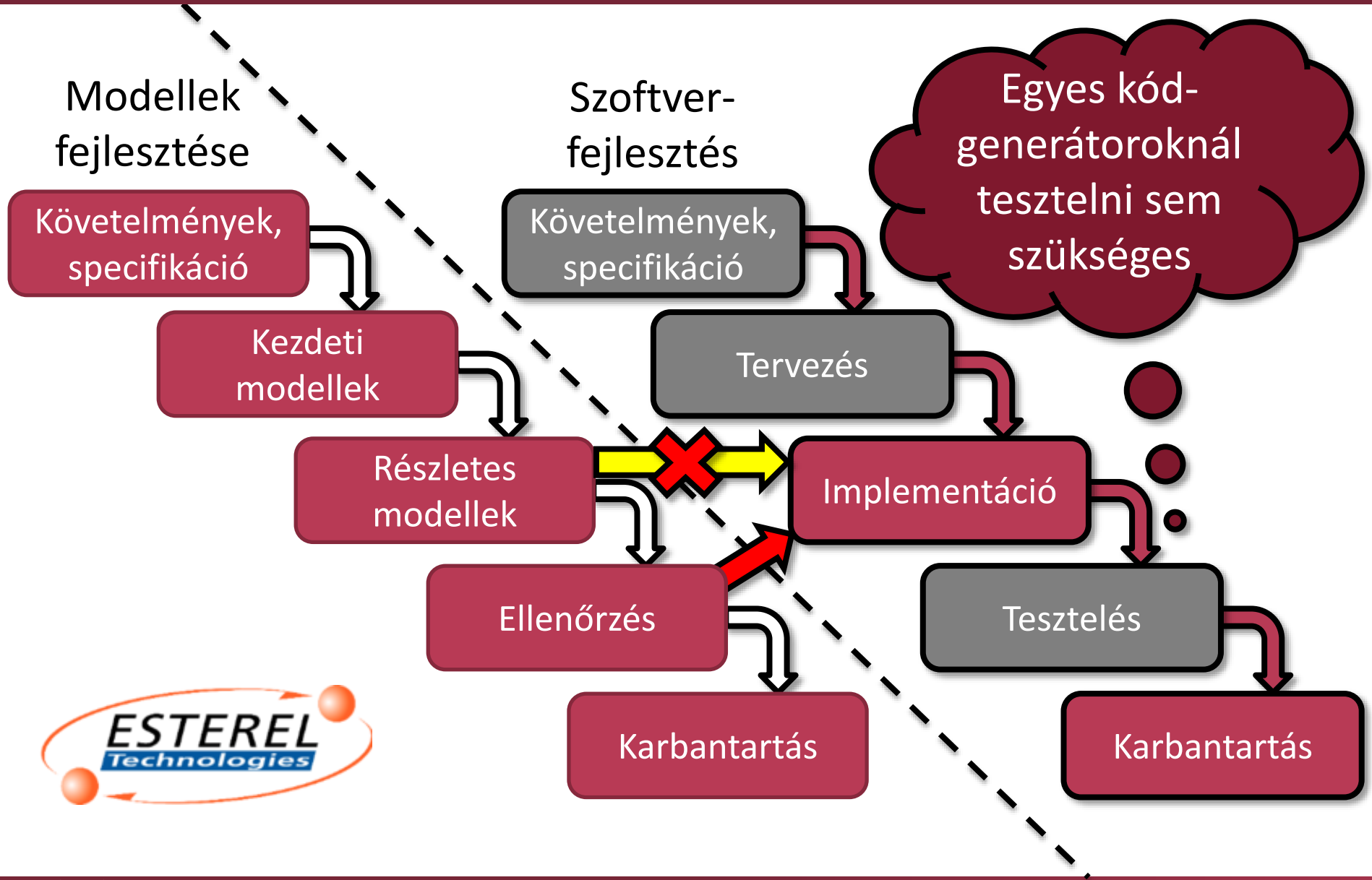
Implementáció

Tesztelés

Karbantartás



Hitelesített kódgenerátor



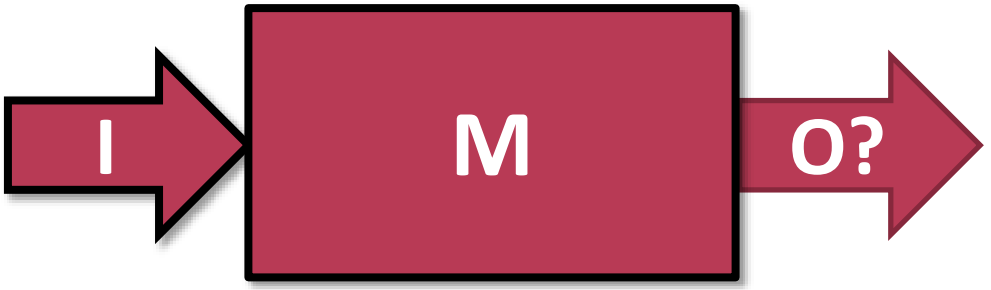
Modellek és feladatok

- Szintézis:

  Specifikációnak megfelelő modell?

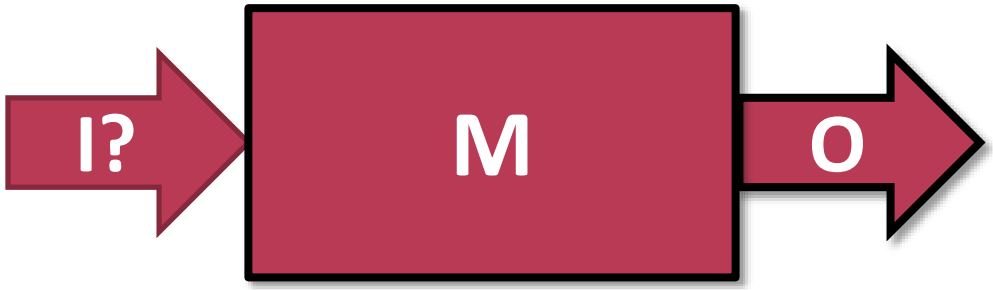
- Analízis:

  Modell viselkedése?

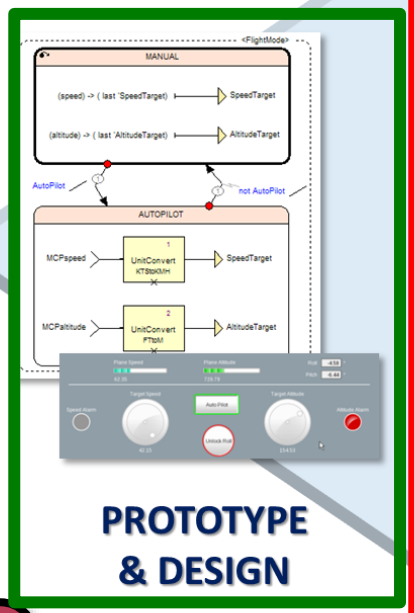


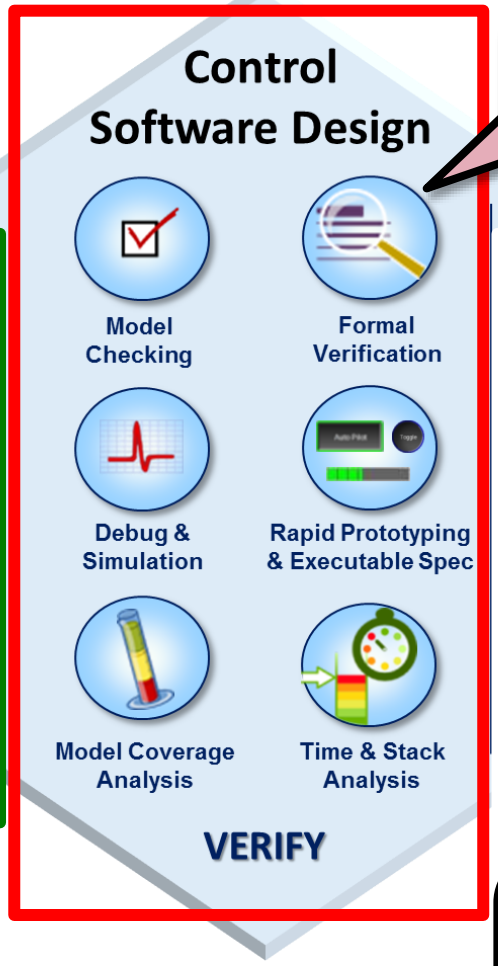
- Vezérlés:

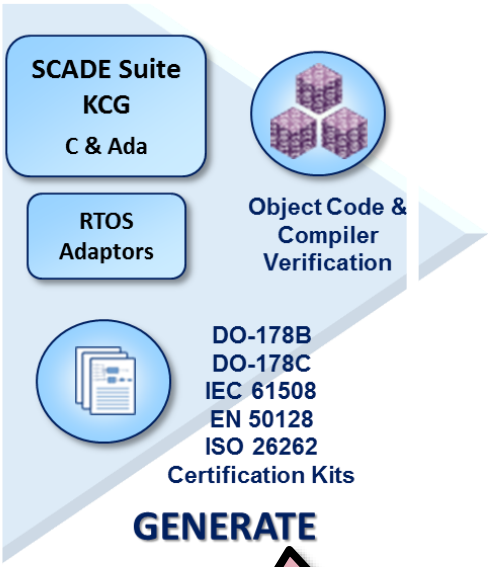
  Kívánt állapot hogy érhető el?



Példa: Esterel SCADE







Lásd: modellek ellenőrzése

Specifikáció

Modellek

Ellenőrzés

Generálás

Repülőgép, kritikus ES, autó

Előnyök

- Átlagosan 10 ellenőrzött kódsor/nap (kritikus rendszer)
  - Kézi kódolásnál csak 5 → dupla termelékenység!
- Kódolás, ellenőrzés és tesztelés költsége
  - 70-90%-os csökkenés
- Módosításhoz szükséges idő (software update)
  - 65-75%-os csökkenés
- Automatikus modellellenőrzés
  - Nincsenek se kódolási hibák, se alacsony szintű tesztek

SPECIFIKÁCIÓ

Hogyan biztosítjuk azt, hogy ne rossz rendszert építsünk jól?

Specifikáció

- Dokumentált követelmények halmaza, amelynek egy adott terméknek, dizájnnak vagy szolgáltatásnak meg kell felelnie.
- Fajtái:
  - Informális vs. Formális
  - Deklaratív vs. Végrehajtható
  - Funkcionális vs. Nemfunkcionális

Specifikáció vs. implementáció

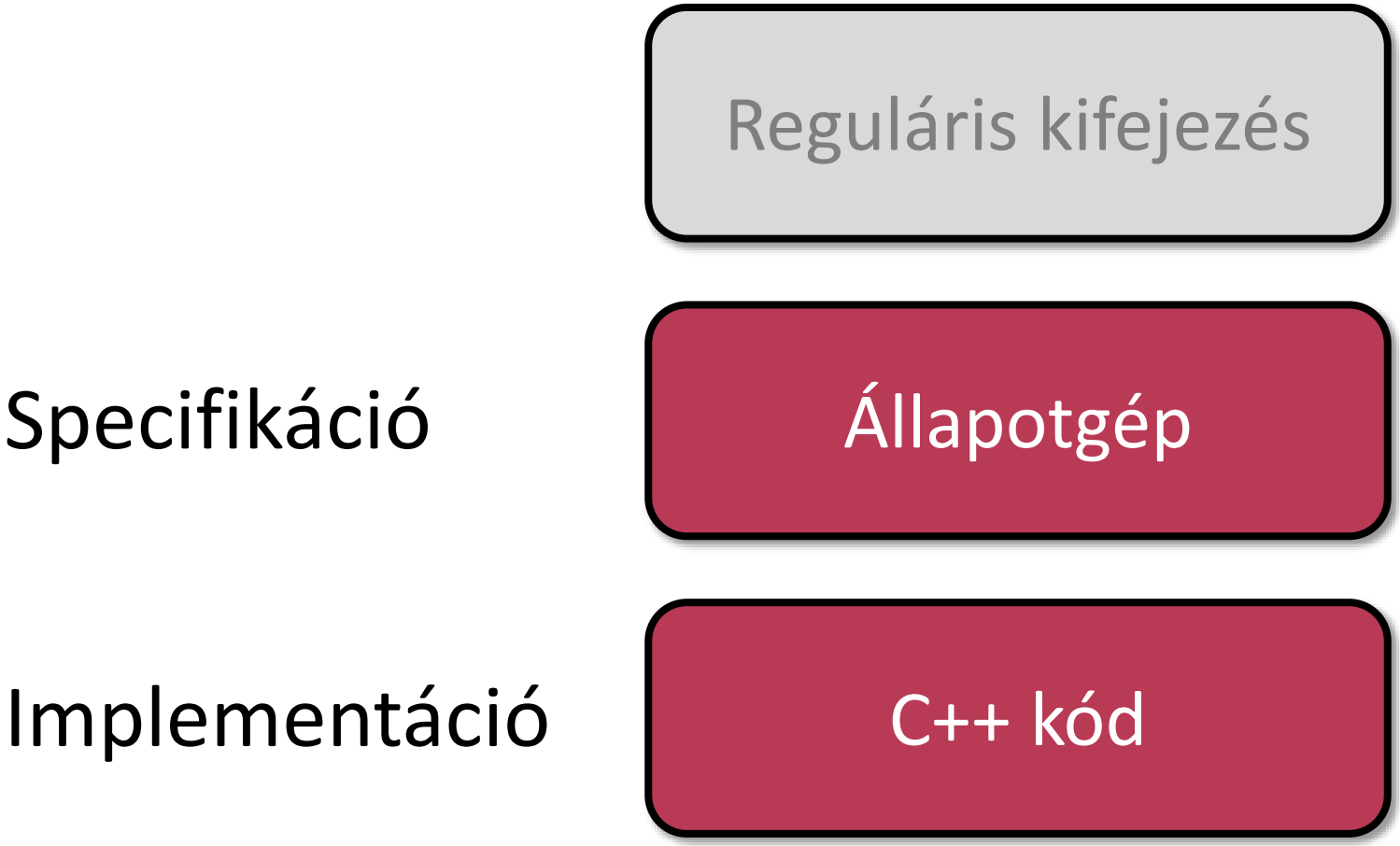
- Specifikáció és implementáció: a folyamat két vége

Specifikáció — Reguláris kifejezés

Implementáció — Állapotgép

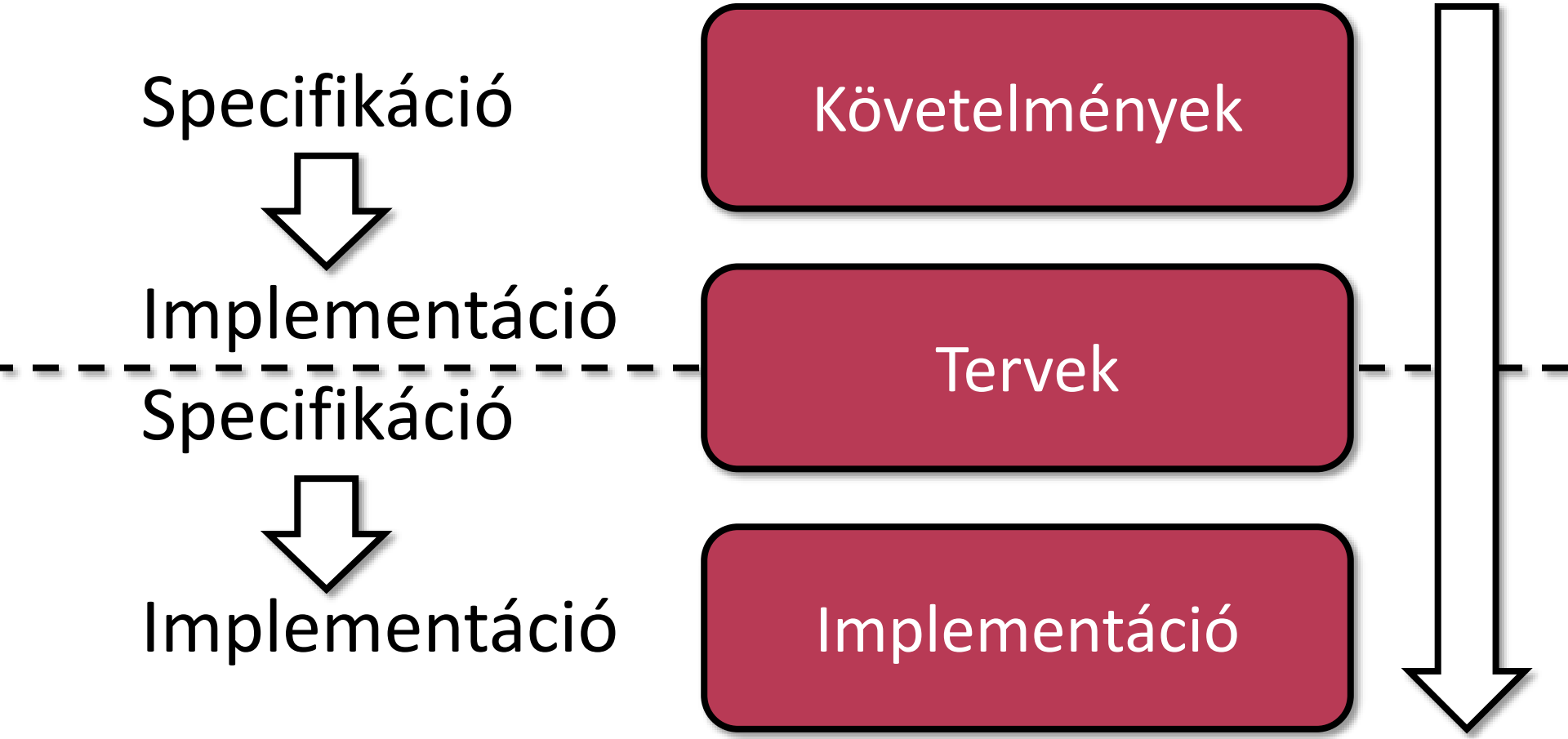
Specifikáció vs. implementáció

- Specifikáció és implementáció: a folyamat két vége



Specifikáció vs. implementáció

- Specifikáció és implementáció: a folyamat két vége

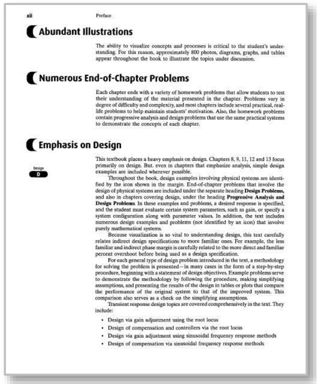


Specifikáció vs. implementáció

- Specifikáció és implementáció: a folyamat két vége
- Specifikációból implementáció:
  - Szintézis/finomítás
  - Nemdeterminizmus csökken
    - Végleges implementáció: célszerűen determinisztikus
- Végső implementáció: végrehajtó környezettől függ
  - Szimuláció
  - CPU gépi kódja
  - Java esetén:
    - Java virtuális gép
    - Java processzor (hardveres implementáció)

Informális és formális specifikáció

- Informális specifikáció
  - Leginkább kommunikációra használt
  - Jellemzően a fejlesztési folyamat elején
  - Többféle értelmezés, pontosítás szükséges
  - Pl. szöveges leírás, folyamatábra, pszeudokód, stb.



xii Preface

Abundant Illustrations

The ability to visualize concepts and processes is critical to the student's understanding. For this reason, approximately 800 photos, diagrams, graphs, and tables appear throughout the book to illustrate the topics under discussion.

Numerous End-of-Chapter Problems

Each chapter ends with a variety of homework problems that allow students to test their understanding of the material presented in the chapter. Problems vary in degree of difficulty and complexity, and most chapters include several practical, real-life problems to help maintain students' motivation. Also, the homework problems contain progressive analysis and design problems that use the same practical systems to demonstrate the concepts of each chapter.

Emphasis on Design

Design D

This textbook places a heavy emphasis on design. Chapters 8, 9, 11, 12 and 13 focus primarily on design. But, even in chapters that emphasize analysis, simple design examples are included wherever possible.

Throughout the book, design examples involving physical systems are identified by the icon shown in the margin. End-of-chapter problems that involve the design of physical systems are included under the separate heading **Design Problems,** and also in chapters covering design, under the heading **Progressive Analysis and Design Problems**. In these examples and problems, a desired response is specified, and the student must evaluate certain system parameters, such as gain, or specify a system configuration along with parameter values. In addition, the text includes numerous design examples and problems (not identified by an icon) that involve purely mathematical systems.

Because visualization is so vital to understanding design, this text carefully relates indirect design specifications to more familiar ones. For example, the less familiar and indirect phase margin is carefully related to the more direct and familiar percent overshoot before being used as a design specification.

For each general type of design problem introduced in the text, a methodology for solving the problem is presented—in many cases in the form of a step-by-step procedure, beginning with a statement of design objectives. Example problems serve to demonstrate the methodology by following the procedure, making simplifying assumptions, and presenting the results of the design in tables or plots that compare the performance of the original system to that of the improved system. This comparison also serves as a check on the simplifying assumptions.

Transient response design topics are covered comprehensively in the text. They include:

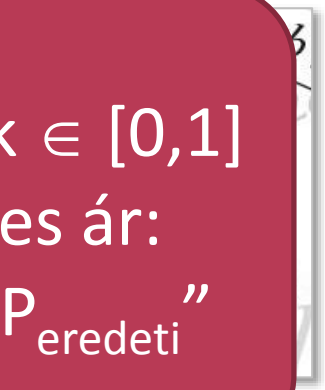
- Design via gain adjustment using the root locus
- Design of compensation and controllers via the root locus
- Design via gain adjustment using sinusoidal frequency response methods
- Design of compensation via sinusoidal frequency response methods

- Formális specifikáció
  - Pontos matematikai szemantika
  - Egyetlen lehetséges értelmezés
  - Ellenőrzés és kódgenerálás alapja lehet
  - Pl. Mealy automata, BPMN, logika, stb.

Informális és formális specifikáció

- Informális specifikáció
  - Leginkább kommuni…
  - Jellemzően a fejlesztési fo…
  - Többféle értelmezés, ponto…
  - Pl. szöveges leírás, folyama…
- Formális specifikáció
  - Pontos matematika…
  - Egyetlen lehetséges értel…
  - Ellenőrzés és kódgenerálás…
  - Pl. Mealy automata, BPMN…

„A szoftver számítsa ki az árat a százalékos kedvezmény(ek) alapján.”



„Kedvezmény: $k \in [0,1]$
Kedvezményes ár:
$P_{kedv} = (1 - \sum k)\, P_{eredeti}$”

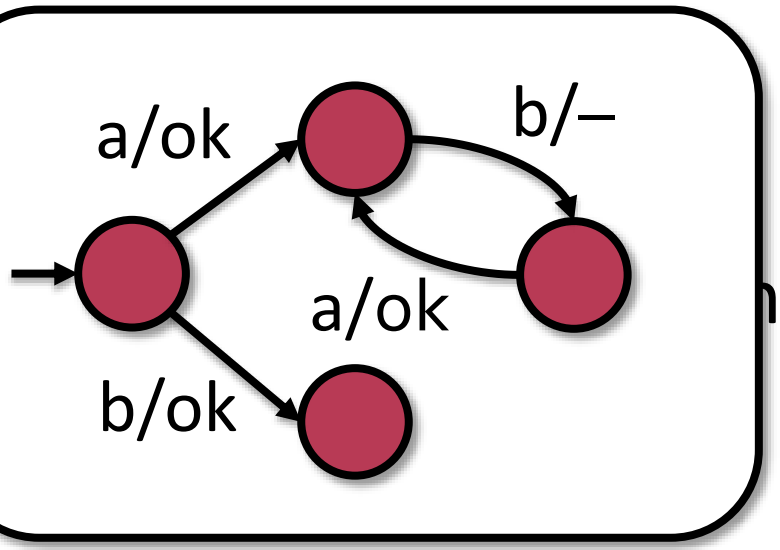
Deklaratív és végrehajtható specifikáció

- Deklaratív specifikáció
  - Az elvárt végeredményt írja le, de nem mond semmit az előállításáról
  - Pl. logika, reguláris kifejezés, stb.
- Végrehajtható specifikáció
  - Legalább „ennyit kell tudnia” az implementációnak
  - Pl. állapotgép, folyamatmodell, adatfolyamháló, stb.
- Sokszor generálható deklaratív leírásból ekvivalens végrehajtható modell
  - Futási idejű ellenőrzés

Deklaratív és végrehajtható specifikáció

- Deklaratív specifikáció
  - Az elvárt végeredmény az előállításáról
  - Pl. logika, reguláris kifejezé
- Végrehajtható specifikáció
  - Legalább „ennyit k
  - Pl. állapotgép, folyamat
- Sokszor generálható dekla végrehajtható modell
  - Futási idejű ellenőrzés

Elfogadott bemenetek:
(a(ba)*)|b
(pl. „a ba ba ba”)



Funkcionális és nemfunkcionális követelmény

- Funkcionális követelmény
  - **Mit** kell csinálnia a rendszernek?
  - Pl. Autót lehessen vezetni
- Nemfunkcionális követelmény
  - **Hogyan** csinálja a rendszer?
  - Pl. Autó legyen gyors

Többnyire a rendszer **egy részére** vonatkozik

Többnyire a **teljes** rendszerre vonatkozik

Tipikus nemfunkcionális követelmények

- Időzítés (Timeliness)
- Átbocsátóképesség (Throughput)
- Teljesítmény (Performance)
- Megbízhatóság (Reliability)
- Rendelkezésre állás (Availability)
- Védelem ártó szándék ellen (Security)
- Biztonságosság (Safety)
- Karbantarthatóság (Maintainability)
- Használhatóság (Usability)









Szintézis vs. transzformáció

- Szintézis/finomítás:
  - A specifikációt teljesítő implementációk felderítése
    - Tervezési tér bejárása
  - Választás a lehetséges megoldások közül
    - Valamilyen szempont szerint (pl. hatékonyság)
    - Tervezői döntés $\Rightarrow$ felelősség!
  - **Plusz információ** kerül a tervekbe

- Fordítás/generálás:
  - Jellemzően különböző formalizmusok között
  - **Nincs új információ**, automatizálható
  - Pl. fordító C++ kódból gépi kódot generál

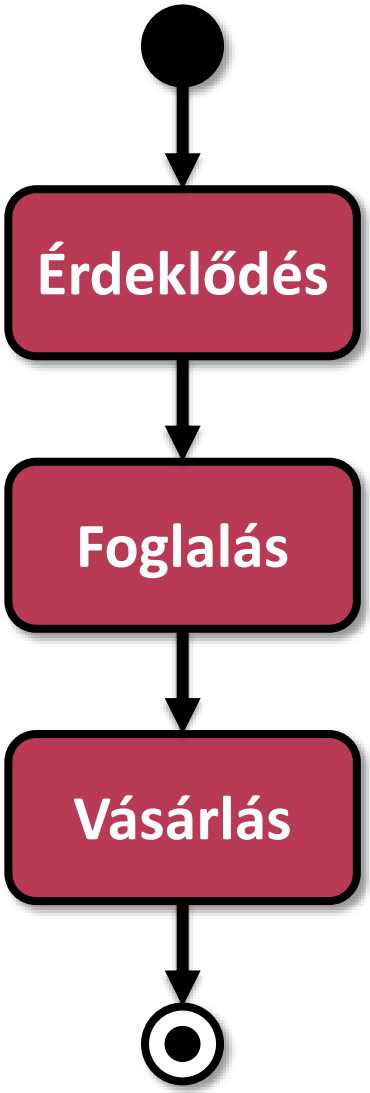
Motiváció

Fejlesztési folyamat

Modellépítés

FINOMÍTÁS, ABSZTRAKCIÓ

Példa: elektronikus utazási iroda



Az iroda ügyfelei

1) érdeklődés után
2) foglalhatnak, majd
3) a vásárlással véglegesíthetik a folyamatot.”

Követelménymodellezés

- Cél: egyértelmű, kezelehető specifikáció
- Terminológiai egységesítés
- Nyomonkövethetőség (traceability):
  - A követelménytől a kódig (+tesztig) legyen követhető
  - Melyik követelményből származik?
  - Komment/annotáció
- Szabvány:
  OMG ReqIF (Requirements Interchange Format)
- Eszközök:
  - (Excel)
  - IBM DOORS
  - ECLIPSE RMF
  - FormalMind

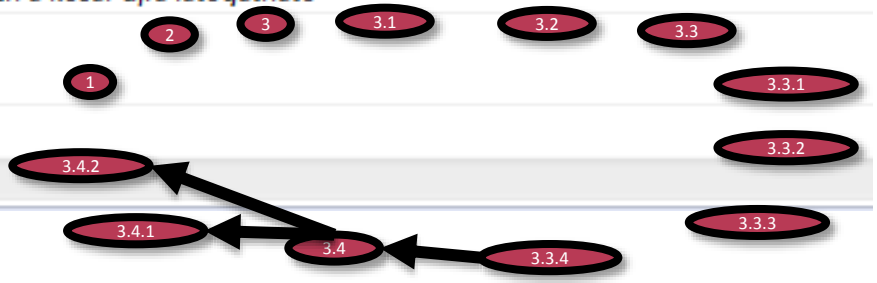
Követelmény modellezés - FormalMind

	Description	
1	Az iroda ügyfeleinek lehetősége van érdeklődni	
2	Az iroda ügyfeleinek lehetősége van utazást foglalni	
3	Az iroda ügyfeleinek lehetősége van vásárlás véglegesíteni a folyamatot.	
3.1	Lehetőség van a kosár tartalmának ellenőrzésére.	
3.2	Lehetőség van a számlázási adatok megadására	
3.3	Lehetőség van fizetni a bank honalpján	
3.3.1	Kérés küldhető a banknak	
3.3.2	Minden tranzakciót naplózni kell	
3.3.3	Az ügyfélnek fizetnie kell	
3.3.4	A folyamat a bank válaszával ér véget	0 ▷ R ▷ 1
	▷	A banki tranzakciónak két lehetséges kimenete van: sikeres tranzakció és egyéb
3.4	A banki tranzakciónak két lehetséges kimenete van: sikeres tranzakció és egyéb	1 ▷ R ▷ 2
	▷	Sikeres tranzakció esetén a megrendelés összeqzésre kerül
	▷	Egyéb esetben a kosár újra látogatható
3.4.1	R Sik… …nzakció esetén a megrendelés összeg… …ül	1 ▷ R ▷ 0
3.4.2	R Egyéb esetb… …újra látogatható	1 ▷ R ▷ 0

Hierarchikus követelmények

Függőségek

- Hány követelményre hivatkozik?
- Hány hivatkozik rá?

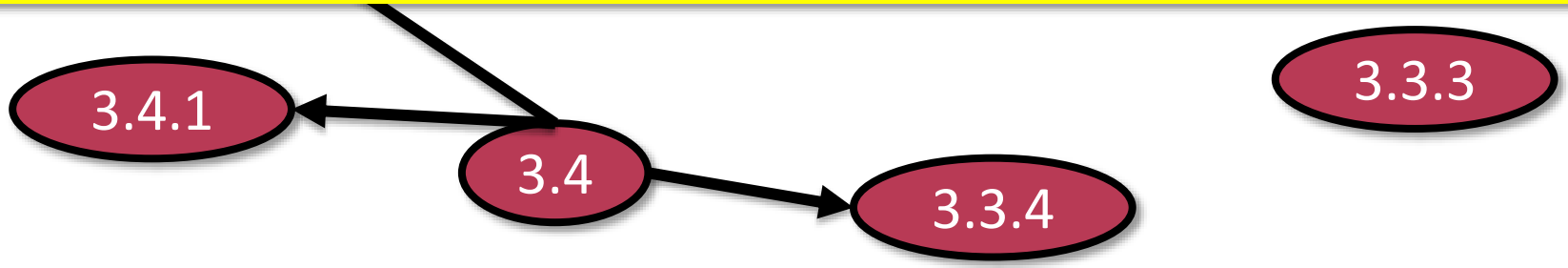


Azonosító

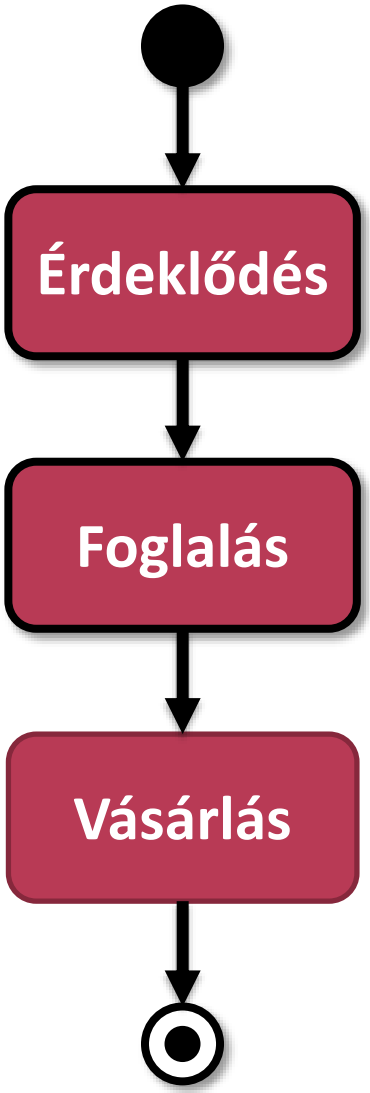
Követelmények függése - gráf

Kérdések:

- **Fokozatos SW fejlesztés:**
  Egy verzióban mik az egyszerre implementálandó követelmények?
- **Karbantartás:**
  Mit kell változtatni, ha változik egy követelmény?



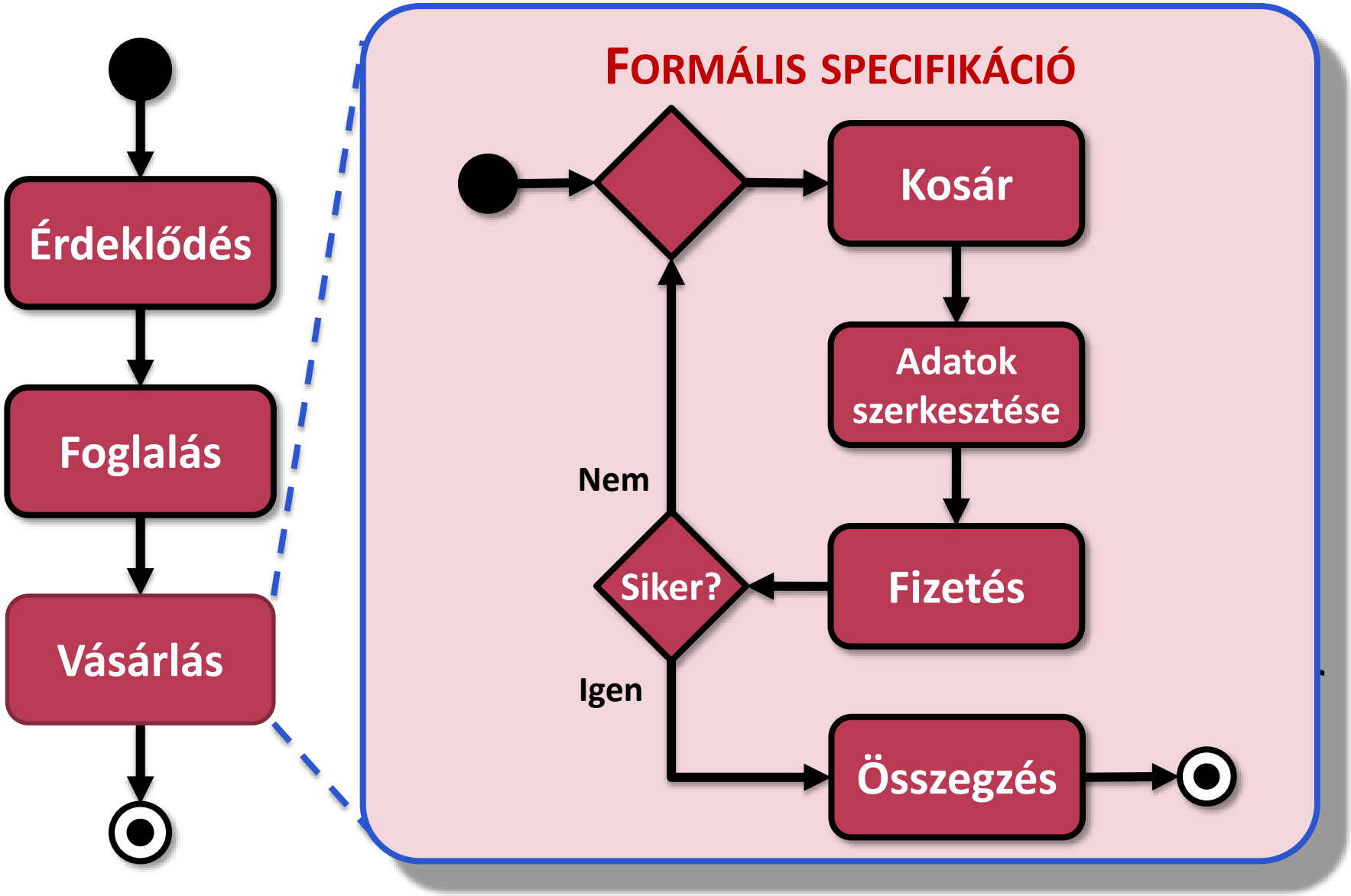
Vásárlási lépés menete



SZÖVEGES SPECIFIKÁCIÓ

1) Kosár tartalmának ellenőrzése
2) Számlázási és szállítási adatok megadása
3) Fizetés a bank honlapján
4) Tranzakció:
   a) Sikeres: megrendelés összegzése
   b) Egyébként: újra a kosár látható

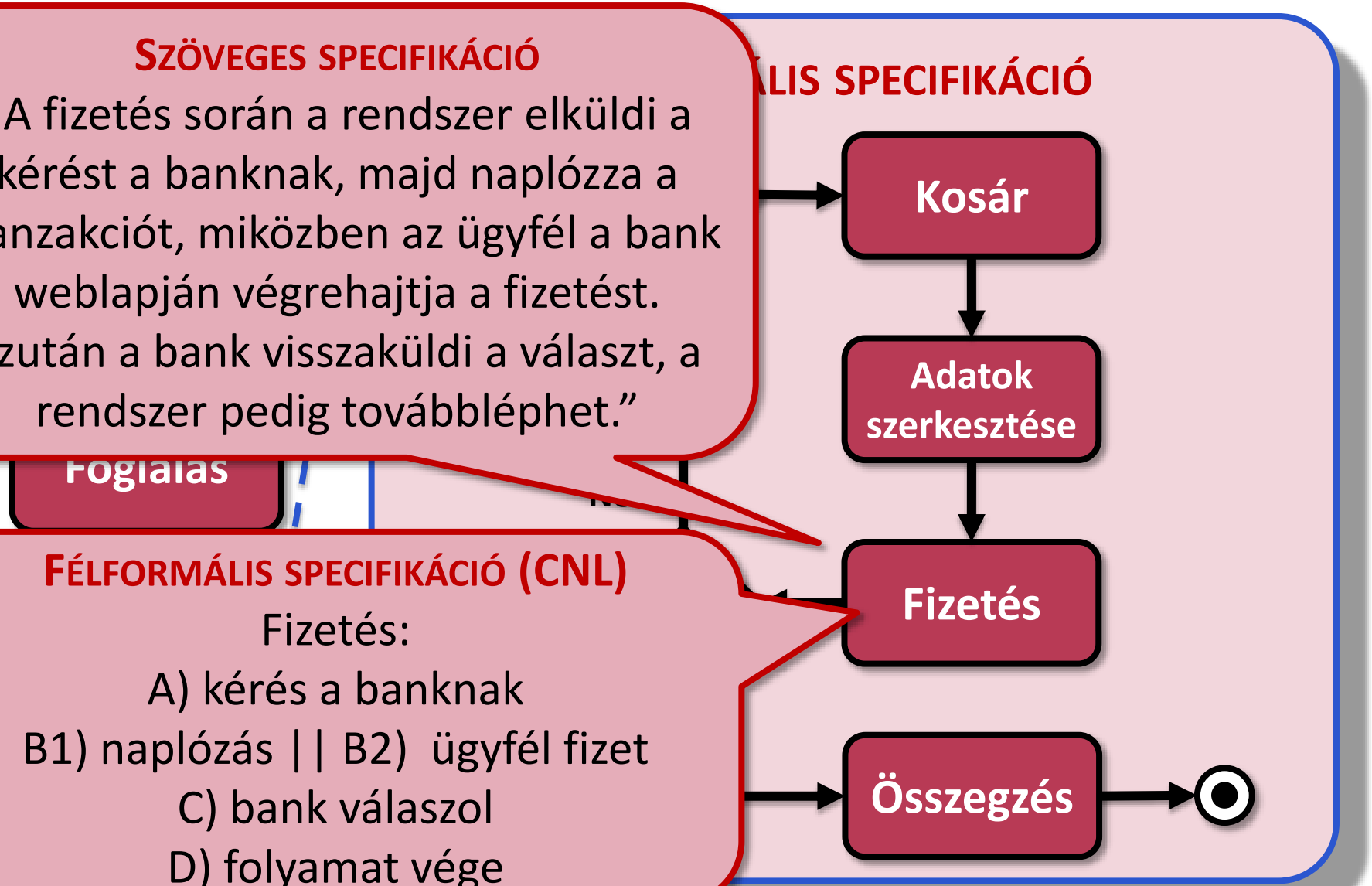
Vásárlási lépés menete kibontva



Fizetés lépés menete

SZÖVEGES SPECIFIKÁCIÓ

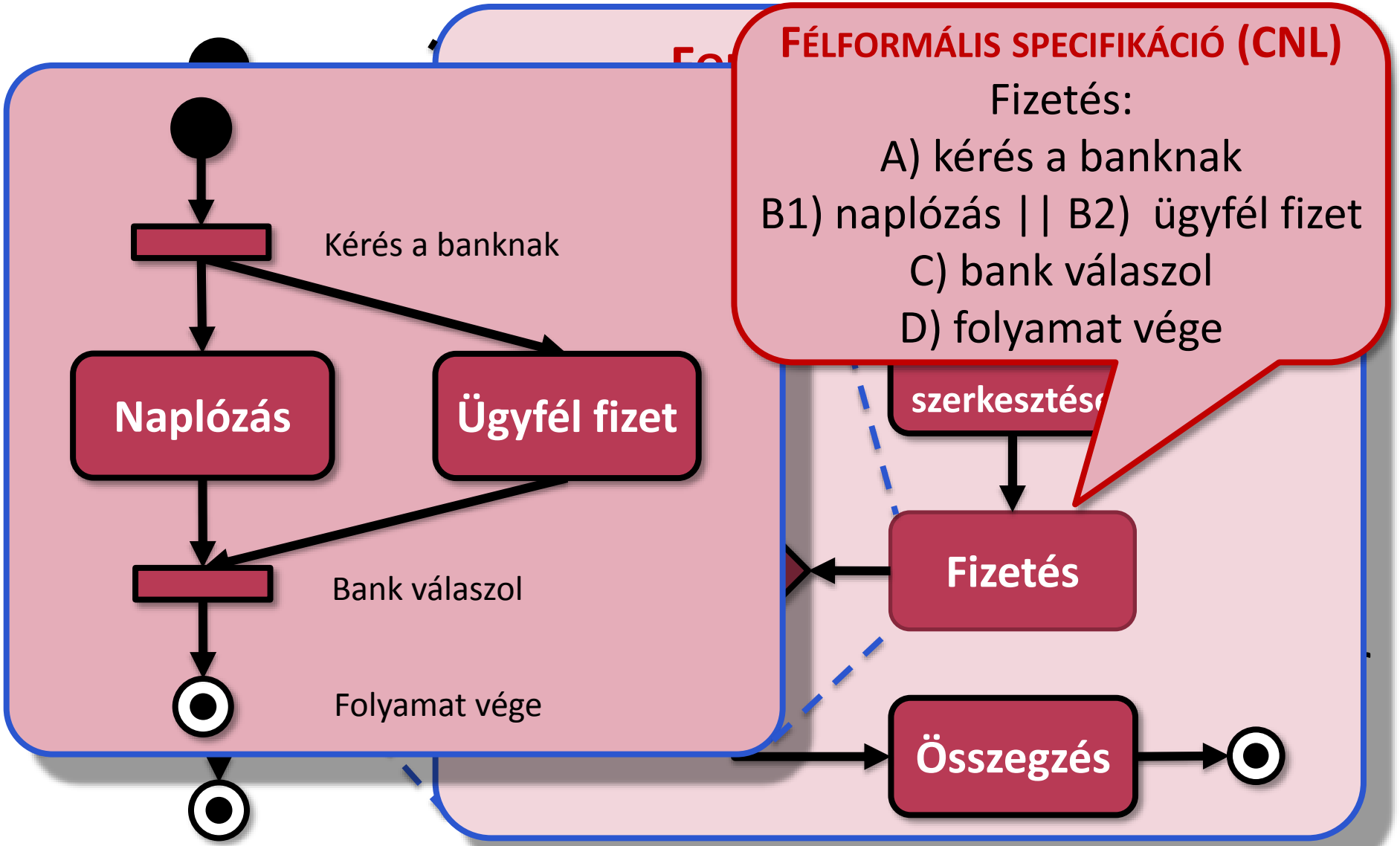
„A fizetés során a rendszer elküldi a kérést a banknak, majd naplózza a tranzakciót, miközben az ügyfél a bank weblapján végrehajtja a fizetést. Ezután a bank visszaküldi a választ, a rendszer pedig továbbléphet.”



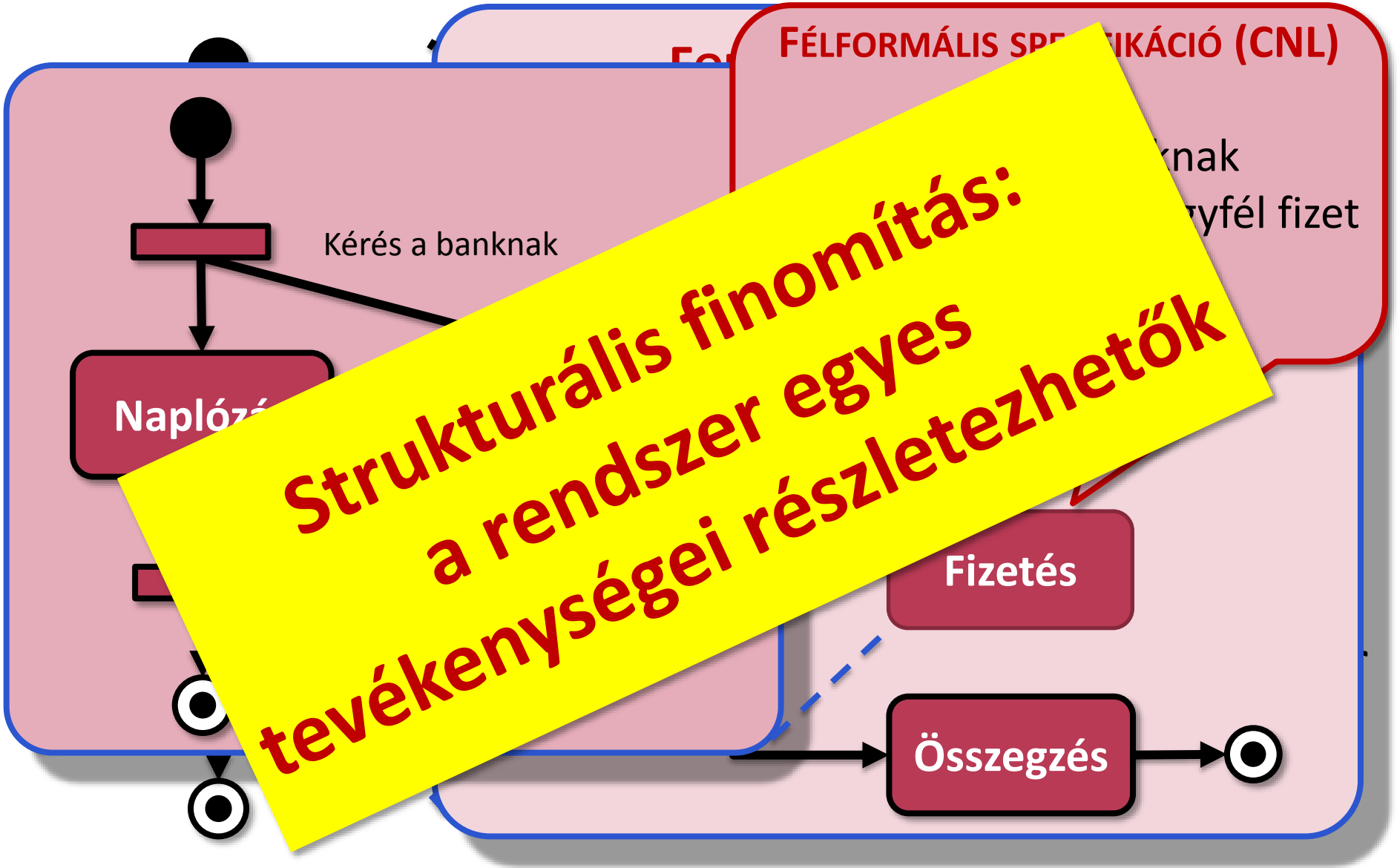
FÉLFORMÁLIS SPECIFIKÁCIÓ (CNL)

Fizetés:
A) kérés a banknak
B1) naplózás || B2) ügyfél fizet
C) bank válaszol
D) folyamat vége

Fizetés lépés menete kifejtve

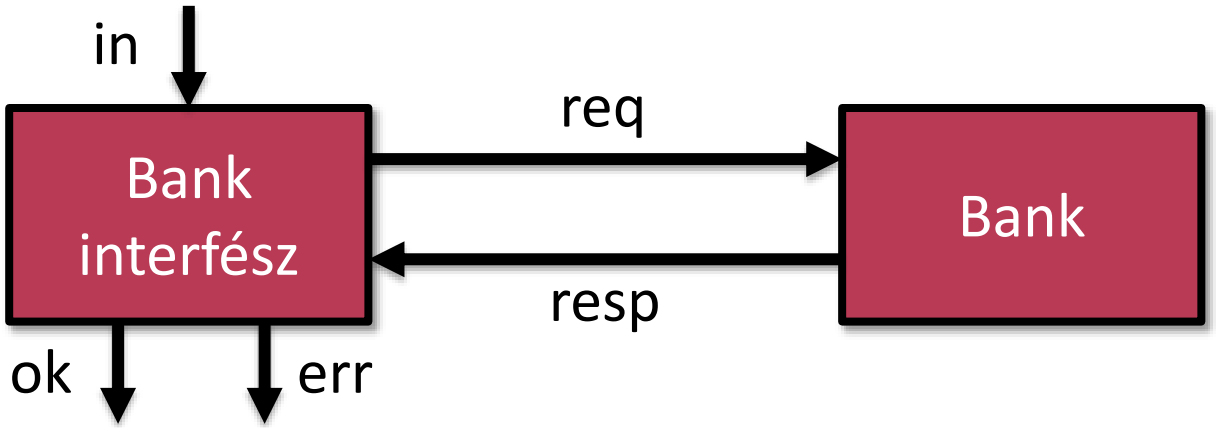


Fizetés finomítása

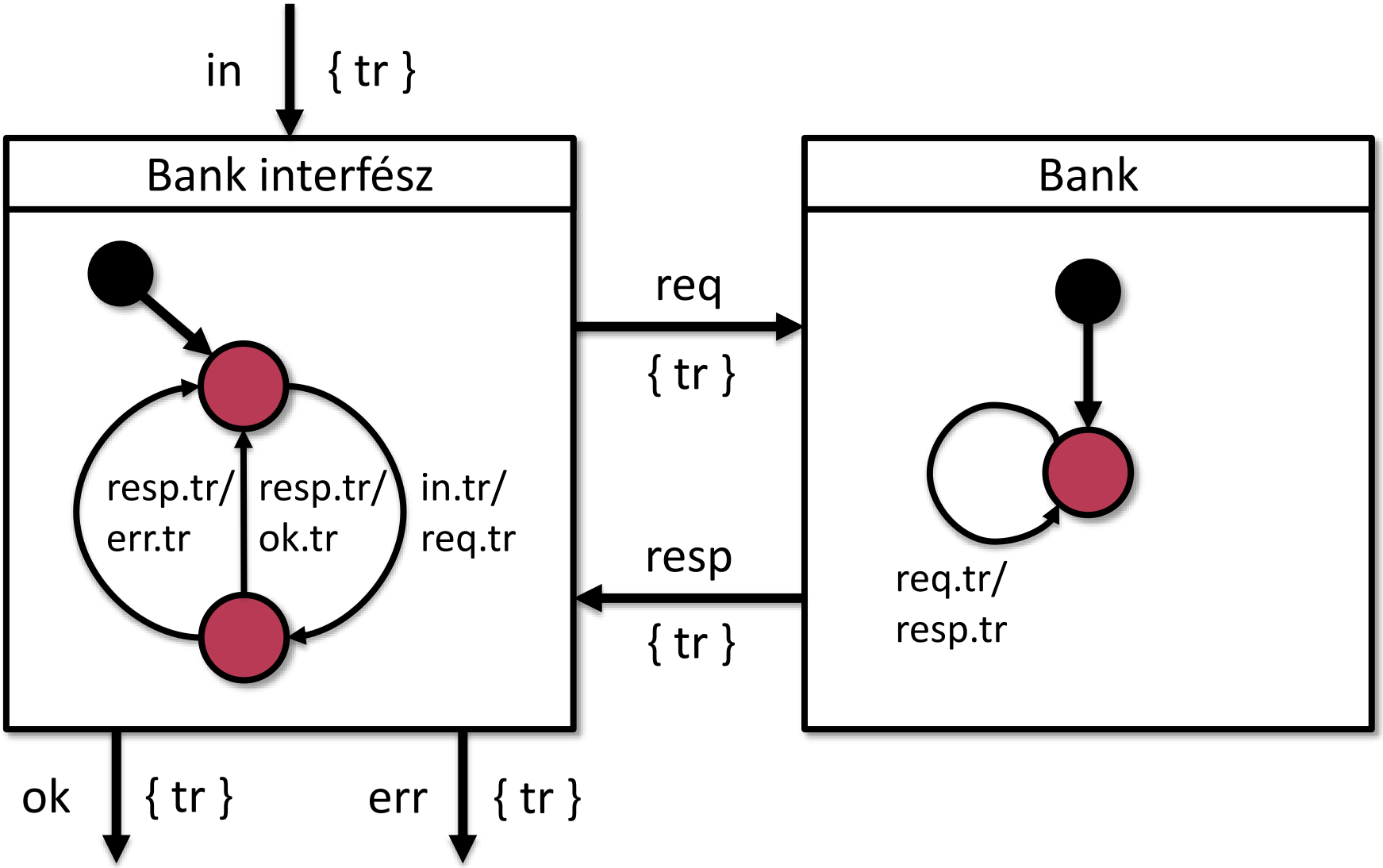


Kommunikáció tervezése

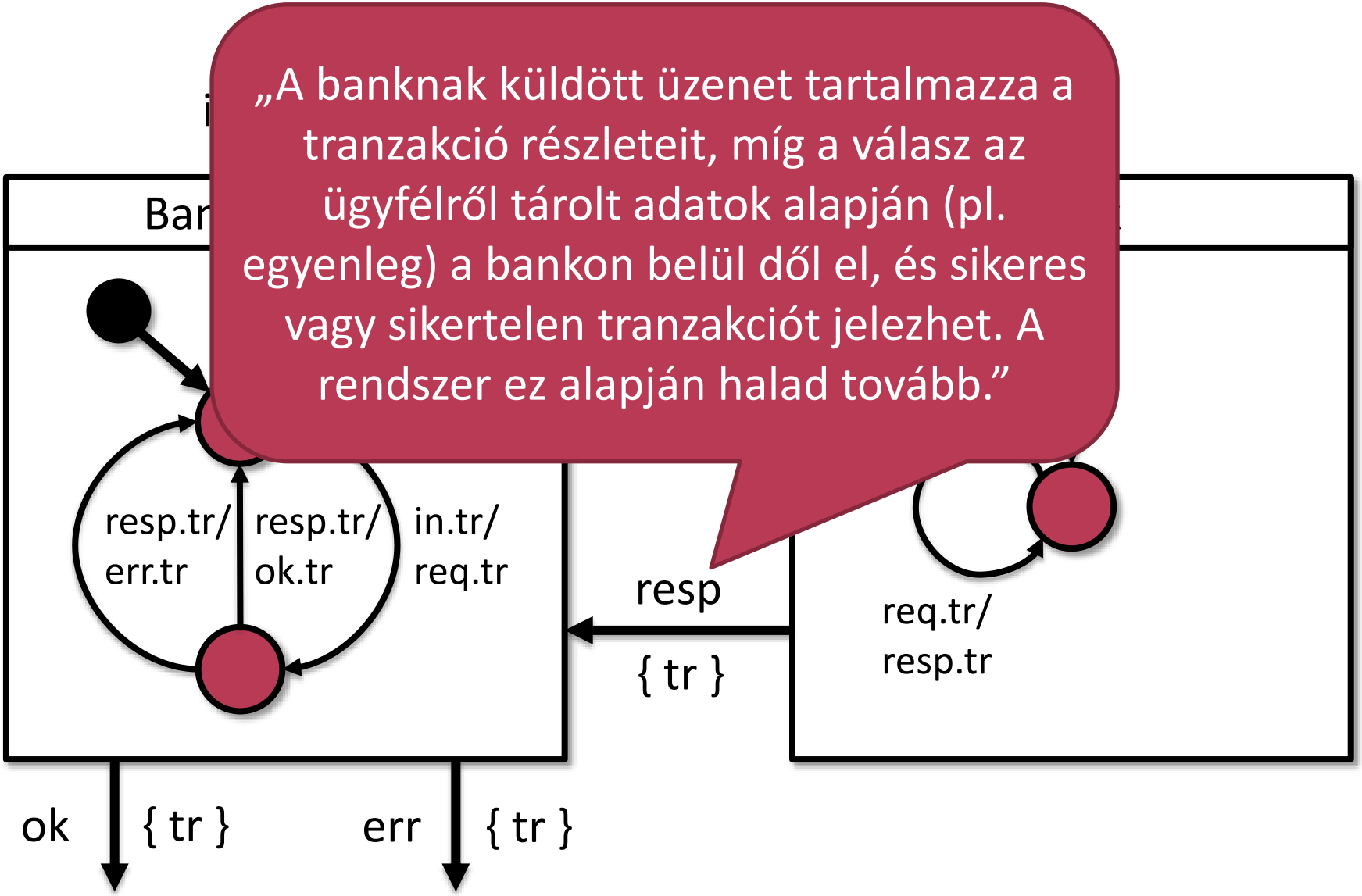
„A bank és a rendszer kommunikációja során először a rendszer üzenetet küld a banknak, aminek hatására az feldolgozza a tranzakciót és választ küld a rendszernek. Ezután a rendszer vagy a teljesült, vagy a hibás tranzakcióhoz tartozó (már definiált) úton halad tovább.”



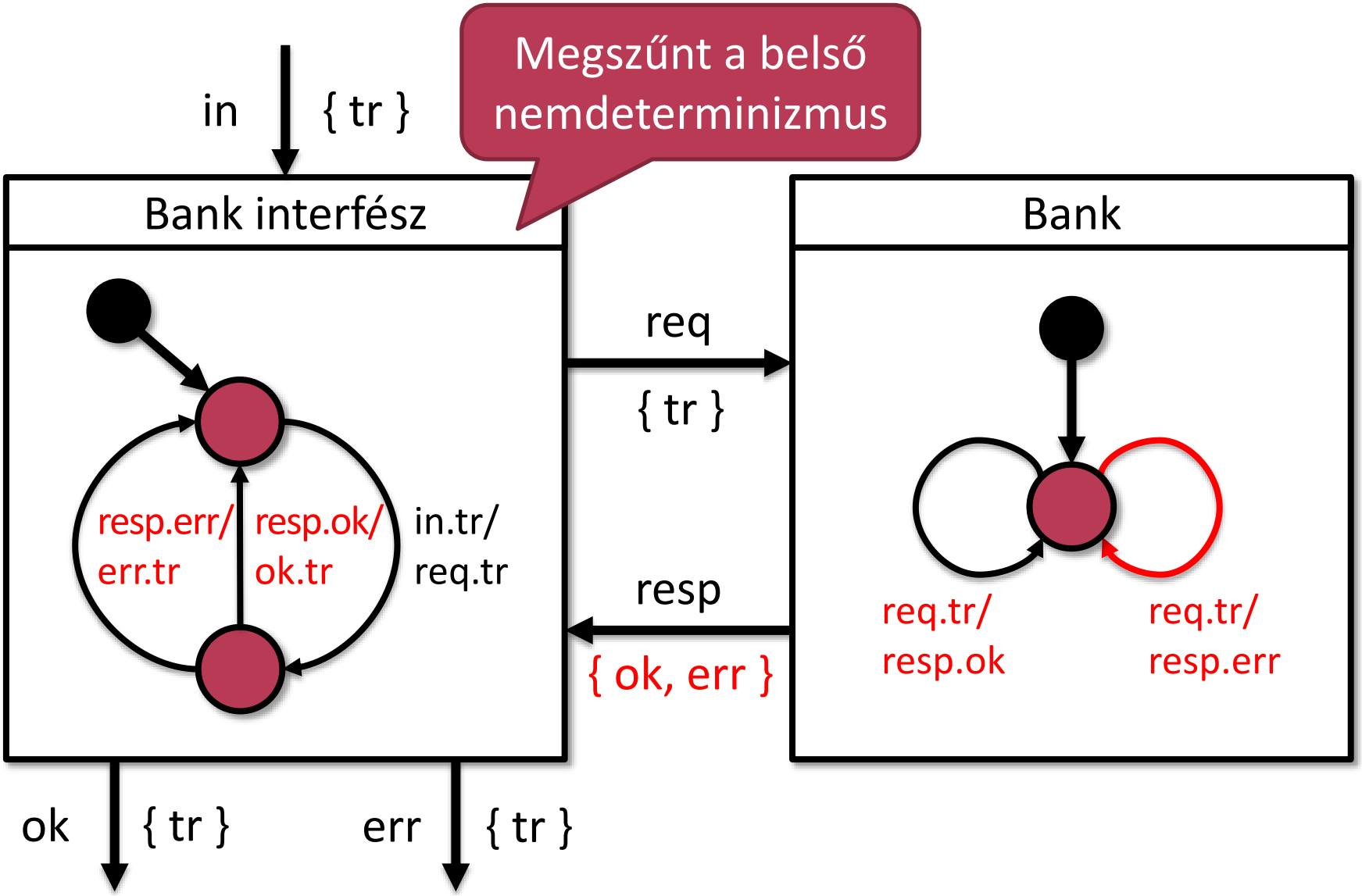
Kommunikáció formális modell



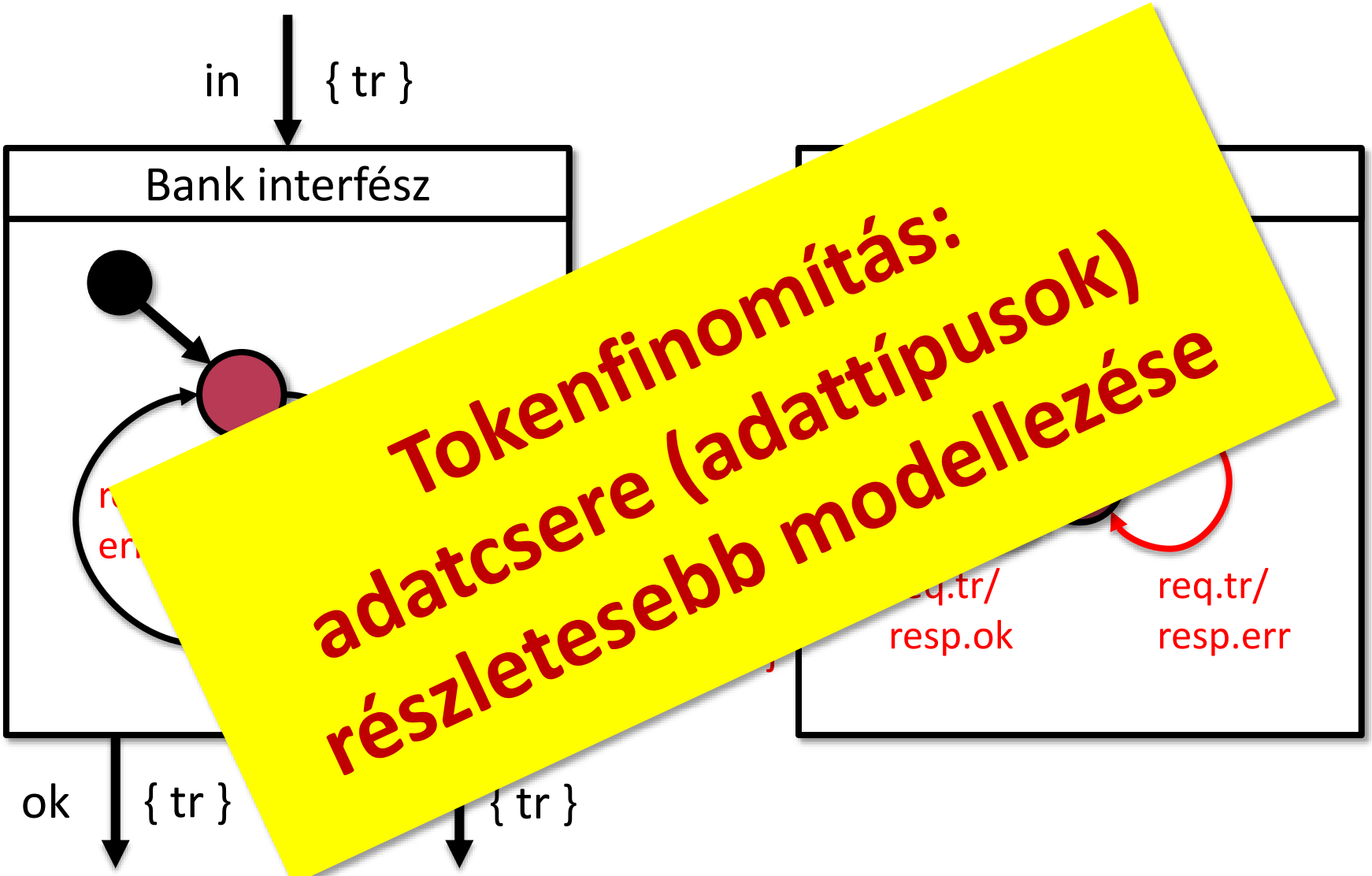
Választípusok kezelése



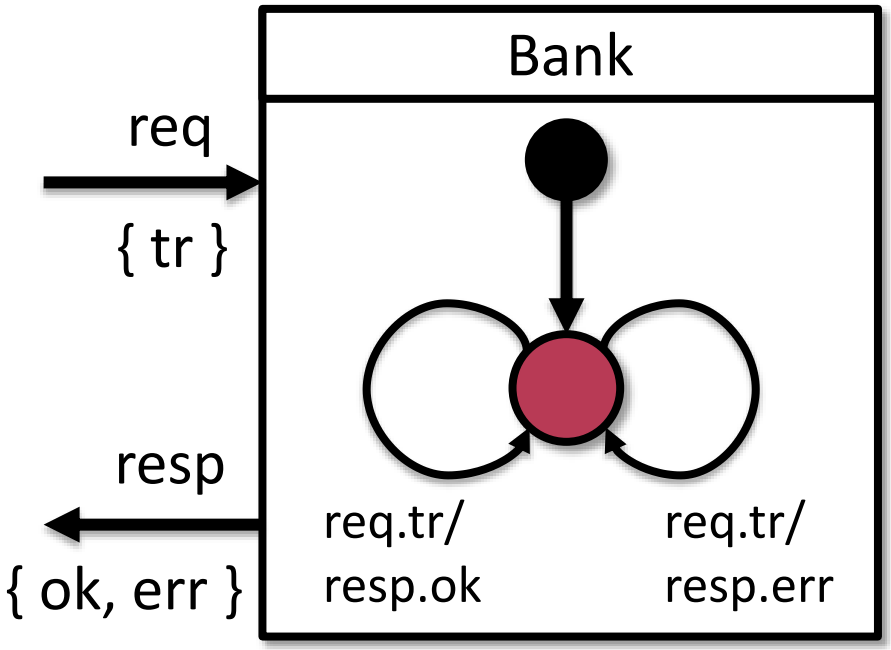
Tokenfinomítás: adattípusok



Tokenfinomítás használata

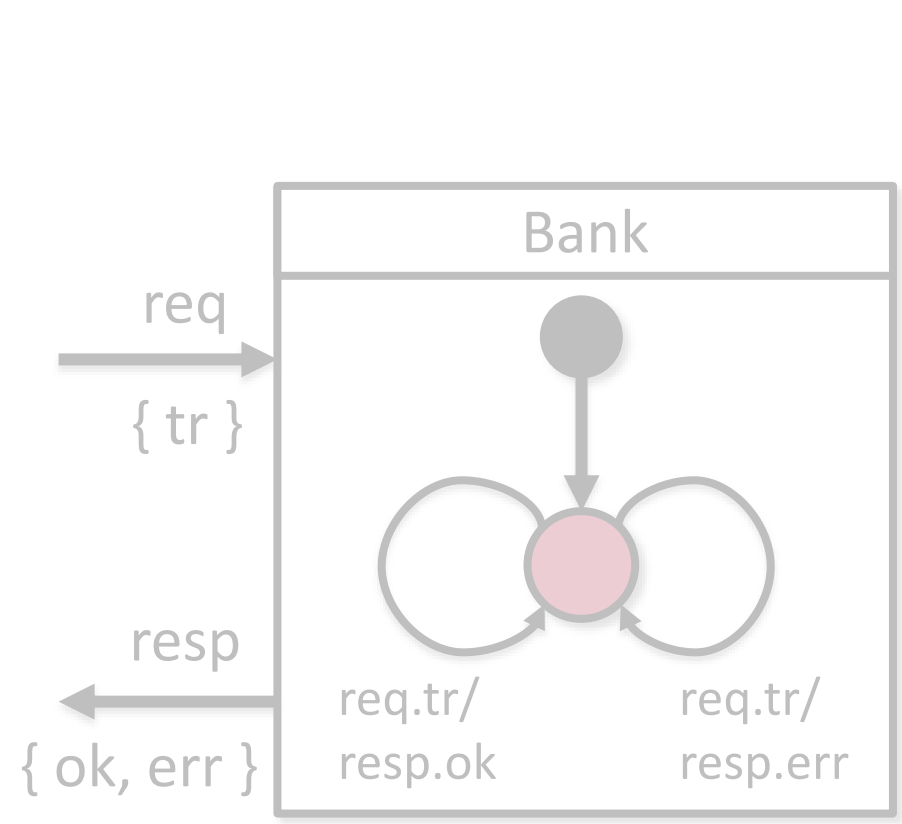


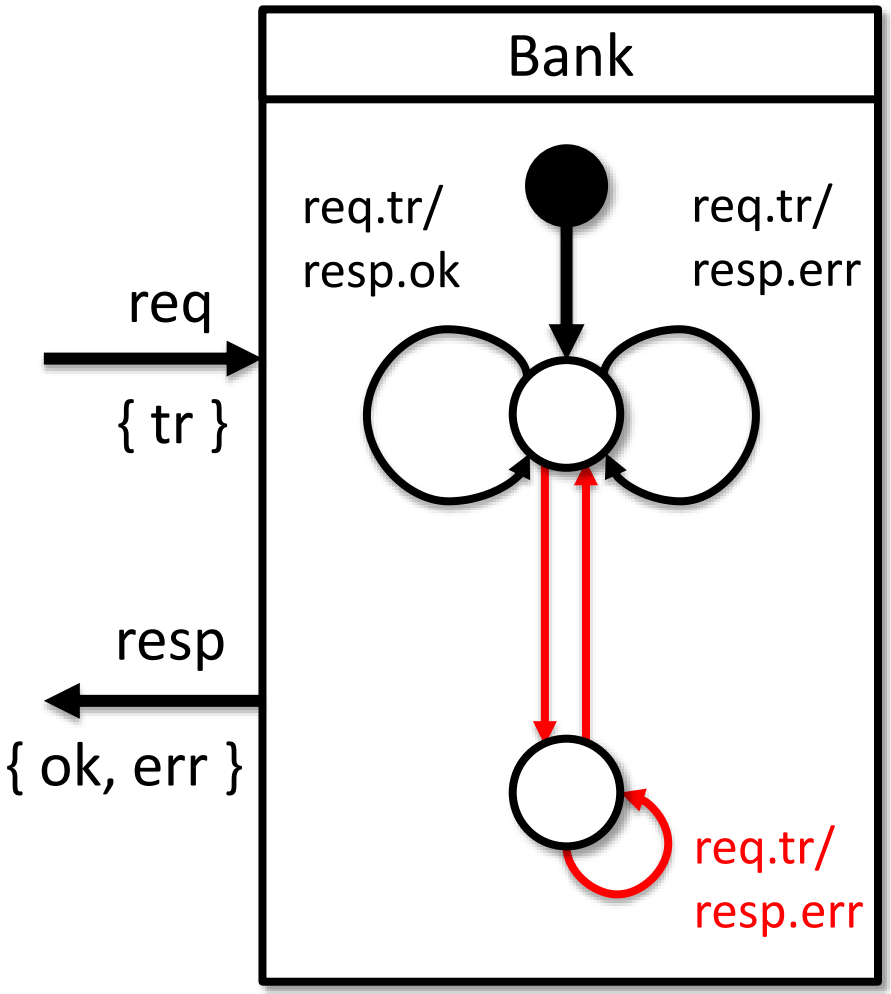
Nemfunkcionális követelmény: hibakezelés



„Időnként előfordulhat, hogy a bank belső hiba miatt képtelen teljesíteni a kéréseket. Ilyenkor a kérésekre mindig hibaüzenet érkezik.”

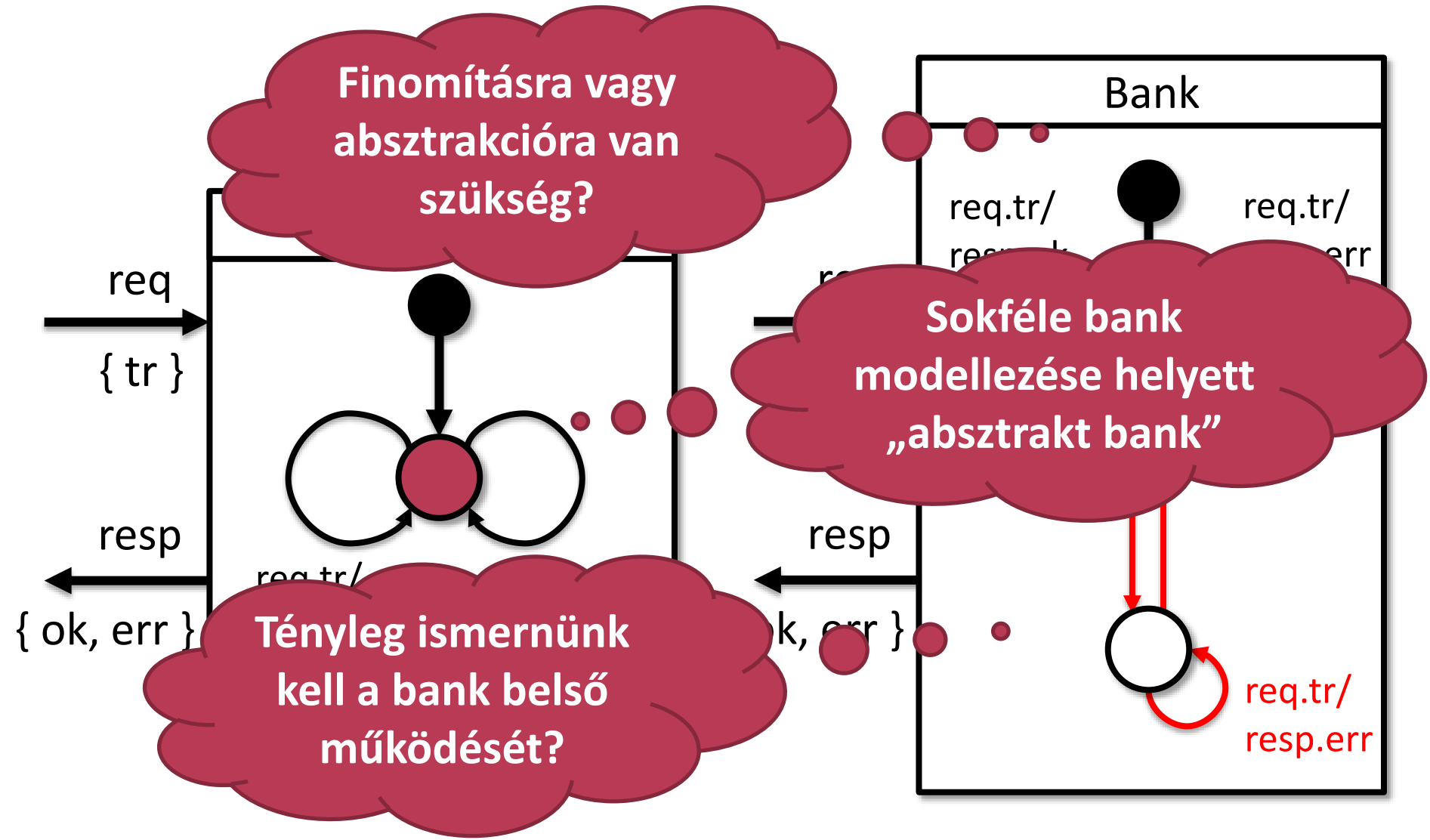
Hibamodellezés



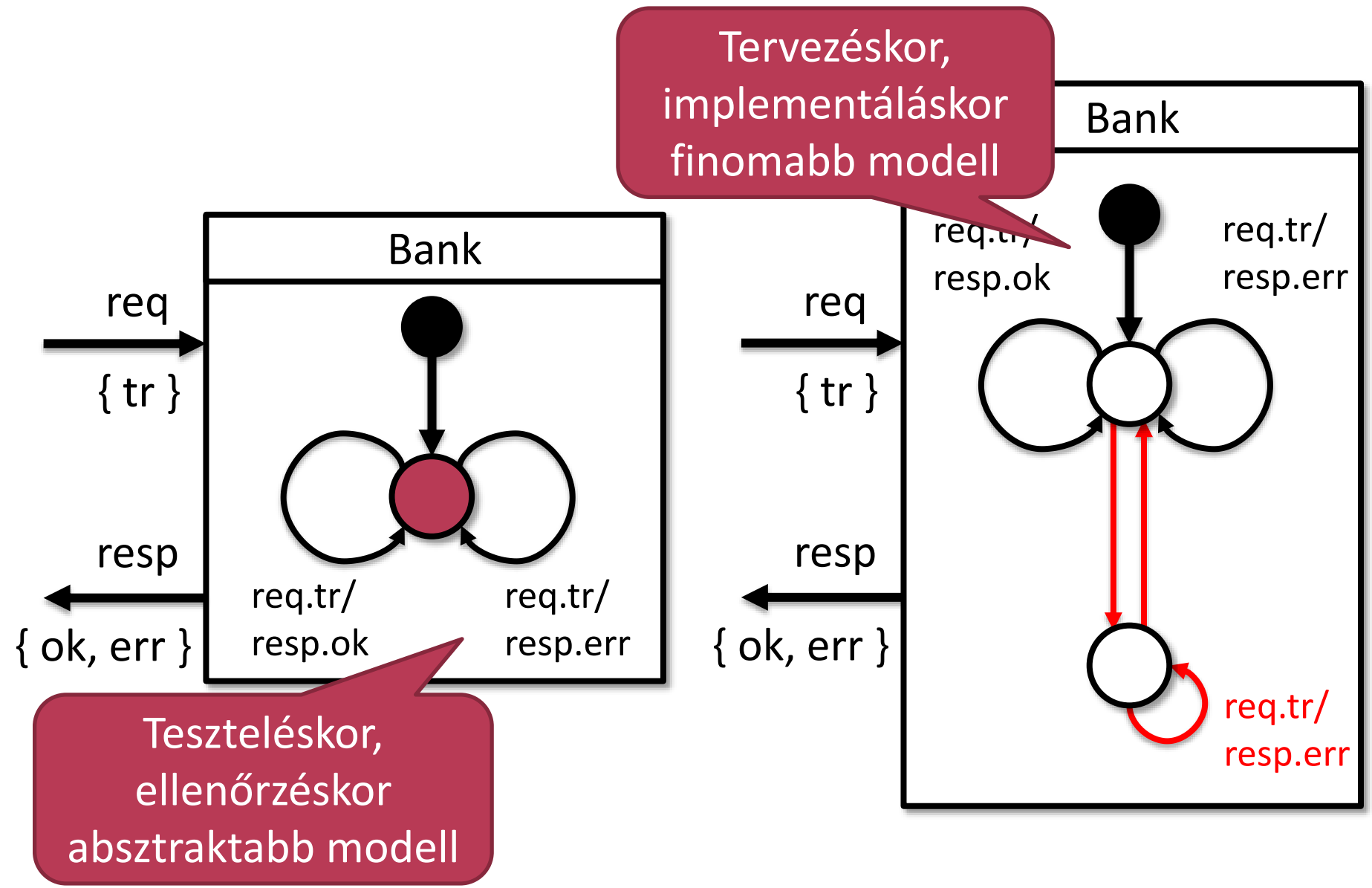


Állapotfinomítás használata

Modell részletessége



Modell részletessége



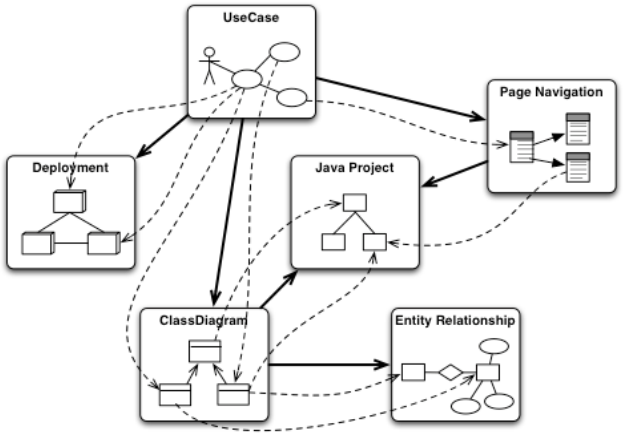
Összefoglalás: finomítás, absztrakció

- Finomítás:
  - **Új információ** beépítése
  - A lehetséges megvalósítások száma csökken
  - Tervezéskor, specializációkor, együttes vizsgálatkor

- Absztrakció:
  - Információ **elvesztésével** jár
  - A lehetséges megvalósítások száma nő
  - Elemzéskor, teszteléskor, dekomponáláskor

Kitekintés: deployment

Tisztán logikai modellek



Informatikai erőforrások

- Informatikai erőforrások:
  - Pl. szerver, processzor, memória, hálózat, adat, stb.
- Hol fognak futni a logikai folyamatok?
  - **Erőforrások** rendelése **tevékenységhez**, **csomóponthoz**

Kitekintés: erőforrásfoglalás

- Erőforráskészlet:
  - Adott típusú erőforrásból elérhető példányok
  - Pl. szerverfürtben lévő gépek
- Folyamatmodell kiegészítése
  - Minden tevékenységhez
    - Használt erőforrás típusa
    - Foglaltsági idő
  - Erőforrásfoglalás
    - Van-e elérhető példány?
    - **Atomi** foglalás (reserve)
    - Felszabadítás (release)