

Követelmény-specifikáció készítés és ellenőrzés

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

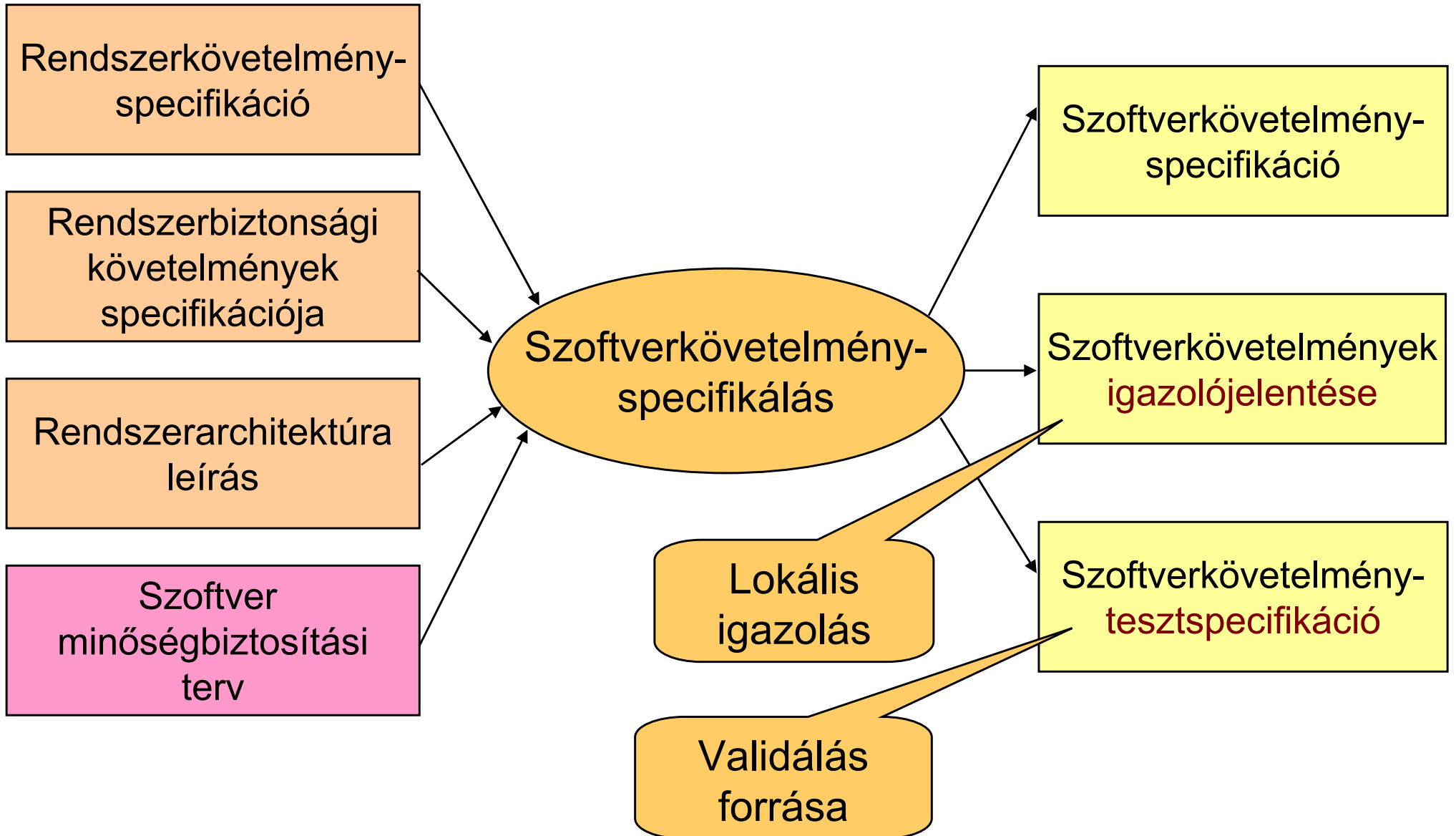
Méréstechnika és Információs Rendszerek Tanszék

<http://www.mit.bme.hu/~majzik/>

Tartalomjegyzék

- A fázis ki- és bemenetei
- A szoftverkövetelmény-specifikáció elkészítése
 - Formális nyelvek (áttekintés)
 - Félformális és strukturált technikák
 - Példa: SysML
- Ellenőrzési feladatok
 - Általános szempontok és módszerek
 - Teljesség és ellentmondás-mentesség
- Követelménykezelés
 - Feladatok
 - Automatikus eszközök

Kimenetek és bemenetek



Szoftver minőségbiztosítási terv

- **Célkitűzés:**
 - Védelem a **szisztematikus hibák** ellen
 - Az összes technikai és irányítási tevékenység meghatározása, figyelése, ellenőrzése
- **Meghatározandó pontok:**
 - Tevékenységek, be/kimeneti kritériumok
 - Számszerű **minőségi elvárások** (pl. ISO/IEC 9126)
 - **Saját korszerűsítés / felülvizsgálat**
- **Külső beszállítók ellenőrzésére szolgáló eljárások**
 - Külső szállító minőségbiztosítási tervének megfelelése
 - Szoftverek ellenőrzése, fejlesztési eszközök megfelelése
- **Probléma bejelentés és javítás rögzítése**
 - Dokumentáció és visszajelzések
 - Elemzések (okok azonosítása)
 - Felderítés és megoldás (javítás) módszerei, megelőző tevékenységek
 - Javítás ellenőrzése, ismételt igazolás és érvényesítés

Szoftverigazolási terv

- Biztonságkritikus rendszerekben külön terv
- Verifikációs tevékenységek tervezése
 - A szabvány szerinti választási lehetőségek kiaknázása
 - Kritériumok és eszközök meghatározása
- Általános igazolási célok:
 - „Lokális” helyesség: Teljesség, ellentmondás-mentesség
 - Megfelelőség: Az előző fázis követelményeinek
- Részletezés:
 - Eszközök (pl. teszt berendezés)
 - Eredmények kiértékelése (elfogadás feltételei), megbízhatósági követelmények kiértékelése
 - Résztvevők szerepe és felelőssége
 - Teszt lefedettség foka

Szoftverkövetelmény-tesztspecifikáció

- Az összes követelmény igazolására illetve a kész szoftveren végzett tesztek leírására
 - **Teszteléssel** vizsgálható követelmény
 - **Elemzéssel** vizsgálható követelmény (pl. architektúra)
- **Tesztesetek rögzítése:**
 - Bemenetek és sorrendjük
 - Elvárt kimenetek és sorrendjük
 - Teljesítési feltételek (siker / kudarc ismérve)
- **Nagy rendszereknél célszerű:**
 - Felhasználói felületek forgatókönyv alapú ellenőrzése
 - Magas kockázatú elemek esetén gyors prototípus

Szoftverkövetelmény-specifikáció

- **Követelmény (requirement):**
 - Bejövő igény, vízió, elvárás
 - Felhasználóktól (user)
 - Érdekeltektől (stakeholder: hatóság, vezetőség, operátor, ...)
 - Validáció alapja
- **Specifikáció (specification, requirement spec.):**
 - Tervezők, fejlesztők felé átalakított elvárások
 - A követelményelemzés (absztrakció, strukturálás, szűrés) eredménye
 - Sokféle típus
 - Rendszerspecifikáció, architektúra specifikáció, tervspecifikáció
 - Verifikáció alapja

Tartalomjegyzék

- A fázis ki- és bemenetei
- A szoftverkövetelmény-specifikáció elkészítése
 - Formális nyelvek (áttekintés)
 - Félformális és strukturált technikák
 - Példa: SysML
- Ellenőrzési feladatok
 - Általános szempontok és módszerek
 - Teljesség és ellentmondás-mentesség
- Követelménykezelés
 - Feladatok
 - Automatikus eszközök

Elvárások a specifikációval szemben

- A követelmények **teljes** lefedése
 - Funkcionális követelmények
 - **Extra-funkcionális** követelmények
- Megfogalmazás: Egyértelmű, igazolható, megvalósítható
- Javasolt megoldások:
 - Szigorú specifikációs nyelv (pl. formális)
 - Ellenőrzött „specifikációs minták” használata
 - Utólagos ellenőrzés
- Példa: EN 50128 szabvány által adott lehetőségek
 - **Formális** módszerek (VDM, Z, B, TL, PN, ...)
 - **Félformális** módszerek (diagram alapú technikák, UML)
 - **Strukturált** metodika (JSD, SADT, SSADM)
 - Emellett **természetes nyelvű** megadás is szükséges!

Formális nyelvek áttekintése

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)
- Logikai nyelvek (HOL, CTL*, ...)
- Konstruktív nyelvek (NUPRL, ...)
- Hibrid illetve széles spektrumú nyelvek (CPN, E-LOTOS, ...)

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek
- Processz leíró nyelvek
- Logikai nyelvek
- Konstruktív nyelvek
- Hibrid illetve széles spektrumú nyelvek
(CPN, E-LOTOS, ...)

Matematikai modell:

- Állapotok mint halmazelméleti struktúrák
- Műveletek, események (elő- és utófeltételekkel)

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)

Egy beléptető rendszer specifikációja (Event-B):

személyek:	$\text{prs} \neq 0$	(halmaz)
épületek:	$\text{bld} \neq 0$	(halmaz)
jogosultság:	$\text{aut} \in \text{prs} \leftrightarrow \text{bld}$	(bináris reláció)
tartózkodás:	$\text{sit} \in \text{prs} \rightarrow \text{bld}$	(teljes függvény)
invariáns:	$\text{sit} \subseteq \text{aut}$	

absztrakt esemény („lehetséges történés”):

```
pass = ANY p,b WHERE (p,b) ∈ aut ∧ sit(p) ≠ b
      THEN sit(p) := b END
```

Bizonyítható, hogy az esemény nem sérti az invariánst.

Bizonyítható az absztrakt események konzisztens finomítása.

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek
- Logikai nyelvek
- Konstruktív nyelvek
- Hibrid illetve széles spektrumú nyelvek (CPN, E-LOTOS, ...)

Absztrakt algebra és kategória-elmélet

- Absztrakt adattípusok: hordozó halmaz és műveletek
- Elsőrendű logika tipikus

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)

- Procedurális

Absztrakt adattípusok: hordozó halmaz és műveletek

- Logikai

```
Type Boolean is
```

```
  sorts Bool
```

```
  opns
```

```
    false, true : -> Bool
```

```
    not : Bool -> Bool
```

```
    and : Bool, Bool -> Bool
```

```
  eqns
```

```
    forall x, y: Bool
```

```
  ofsort Bool
```

```
    not(true) = false;
```

```
    not(false) = true;
```

```
    x and true = x;
```

- Korlat

- Hib

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)
- Logikai nyelvek
- Konstruktív nyelvek
- Hibrid illetve széles sp

- Processzek (szekvenciális utasítás végrehajtás)
- Operátorok a processzek között (szinkronizáció, kommunikáció)

(CSP, CCS, ...)

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)

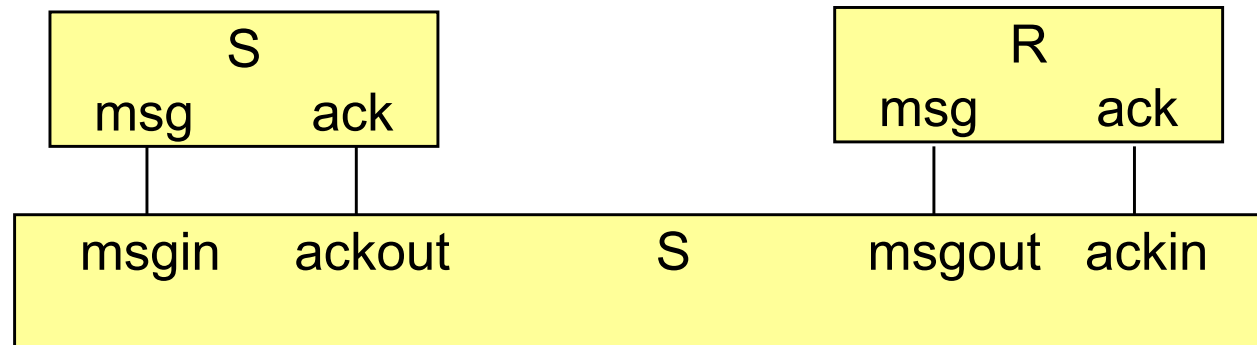
- Logika

Processz algebra (CCS):

- Kor

$$S = \overline{\text{msg}}.\text{ack}.S$$
$$R = \text{msg}.\overline{\text{ack}}.R$$

- Hib

$$M = \text{msgin}.\overline{\text{msgout}}.M + \text{ackin}.\overline{\text{ackout}}.M$$
$$P = S[\text{msgin}/\overline{\text{msg}}, \text{ackout}/\overline{\text{ack}}] \mid M \mid P[\overline{\text{msgout}}/\text{msg}, \text{ackin}/\overline{\text{ack}}]$$


Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)
- Logikai nyelvek (HOL, CTL*, ...)
- Konstruktív nyelvek
- Hibrid illetve széles spektrumú nyelvek

- Formális matematikai logika
(elsőrendű vagy magasabb rendű)
- Temporális logikák
(temporális operátorok)

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)
- Logikai nyelvek (HOL, CTL*, ...)
- Konstruktív nyelvek (NUPRL, ...)
- Hibrid illetve széles skopu

Konstruktív logikai rendszerek (hatékonyan számítható függvények): A tulajdonság bizonyítása egyben a konstrukciót (implementációt) is megadja.

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)
- Logikai nyelvek (HOL, CTL*, ...)
- Konstruktív nyelvek (NUPRL, ...)
- Hibrid

Matematika: Nem konstruktív bizonyítások

- Példa: Valami létezése bizonyítható anélkül, hogy kideríthetnénk, hogy mi az a valami:
 - Létezik $a, b \notin \mathbb{Q}$ úgy, hogy $a^b \in \mathbb{Q}$
- Szoftver specifikációhoz nem alkalmas, ezért korlátozás szükséges a konstruktív bizonyításokra, amik a specifikáció betarthatóságát (szintézist) garantálják

Formális nyelvek

- Modell-orientált nyelvek (VDM, Z, B, ...)
- Algebrai nyelvek (ADT, OBJ, ...)
- Processz leíró nyelvek (CSP, CCS, ...)
- Logikai nyelvek (HOL, CTL*, ...)
- Konstruktív nyelvek (NUPRL, ...)
- Hibrid illetve széles spektrumú nyelvek
(CPN, E-LOTOS, ...)

- Kombinálják az egyes formalizmusok előnyeit, pl.
 - LOTOS: processz algebra és ADT
 - CPN: Petri-háló és adatkezelés (ML)

Félformális nyelvek

- **Struktúra leírás:**
 - Funkcionális blokkdiagramok
- **Adatáramlás leírása:**
 - Adatáramlási diagramok
- **Vezérlés leírása**
 - Állapotátmeneti diagram, állapottérkép
 - (Üzenet) szekvencia diagram
- **Logikai feltételek leírása**
 - Igazságtáblázatok
 - Kényszer nyelvek (pl. OCL a struktúra alapján)

Strukturált metodika

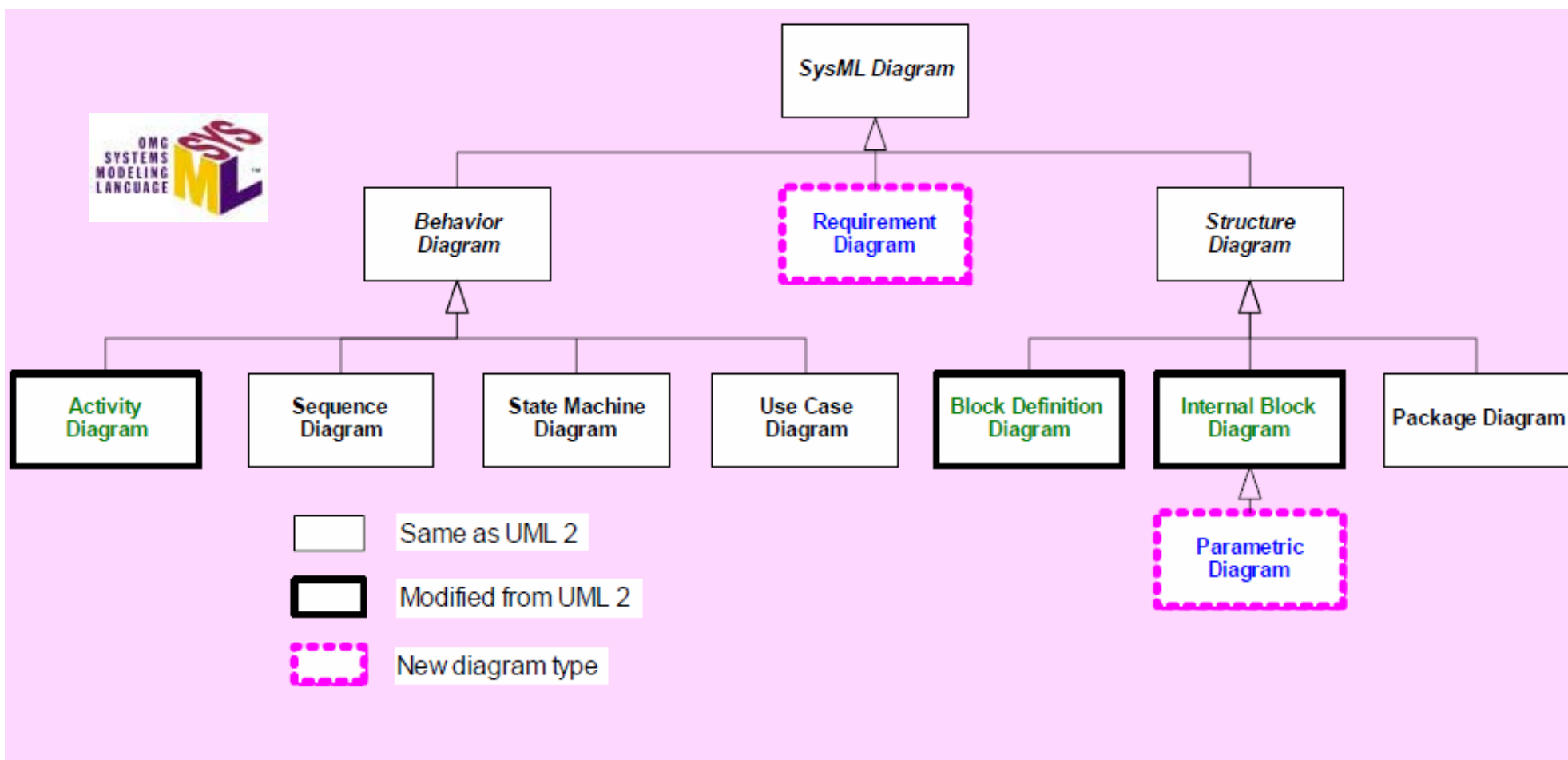
- Jellemzők:
 - Teljes rendszer + környezet leírása
 - Kezelhető részekre osztás (strukturáltság)
 - Intuitív, pragmatikus próbál lenni
 - Ellenőrző listák csatolhatók
 - Jelölésrendszer: Rendszerelemek (viselkedés általában informális)

Strukturált metodika példák

- Jackson System Development (JSD)
 - Entitás struktúra: Entitások + akciók (sorrend) + processzek
 - Hálózati: Kommunikáló szekvenciális processzek
- Valósidejű Yourdon (Ward-Mellor)
 - Alapvető: Környezet (események) + viselkedés (válasz)
 - Konstrukciós: Folyamatok (processzorokon)
- SSADM
 - Logikai adatmodell (entitás relációs diagram)
 - Adatfolyam diagram (processzek, adattárak)
 - Entitás (life history) diagram
 - Entitás hatás diagram
- Structured Analysis and Design Technique (SADT)
 - Aktivitás-faktor diagram: Tevékenységek + faktor relációk; bemenet, vezérlés, erőforrás, kimenet
- ROOM: Real-Time Object-Oriented Modeling

Félformális követelményspecifikáció: SysML

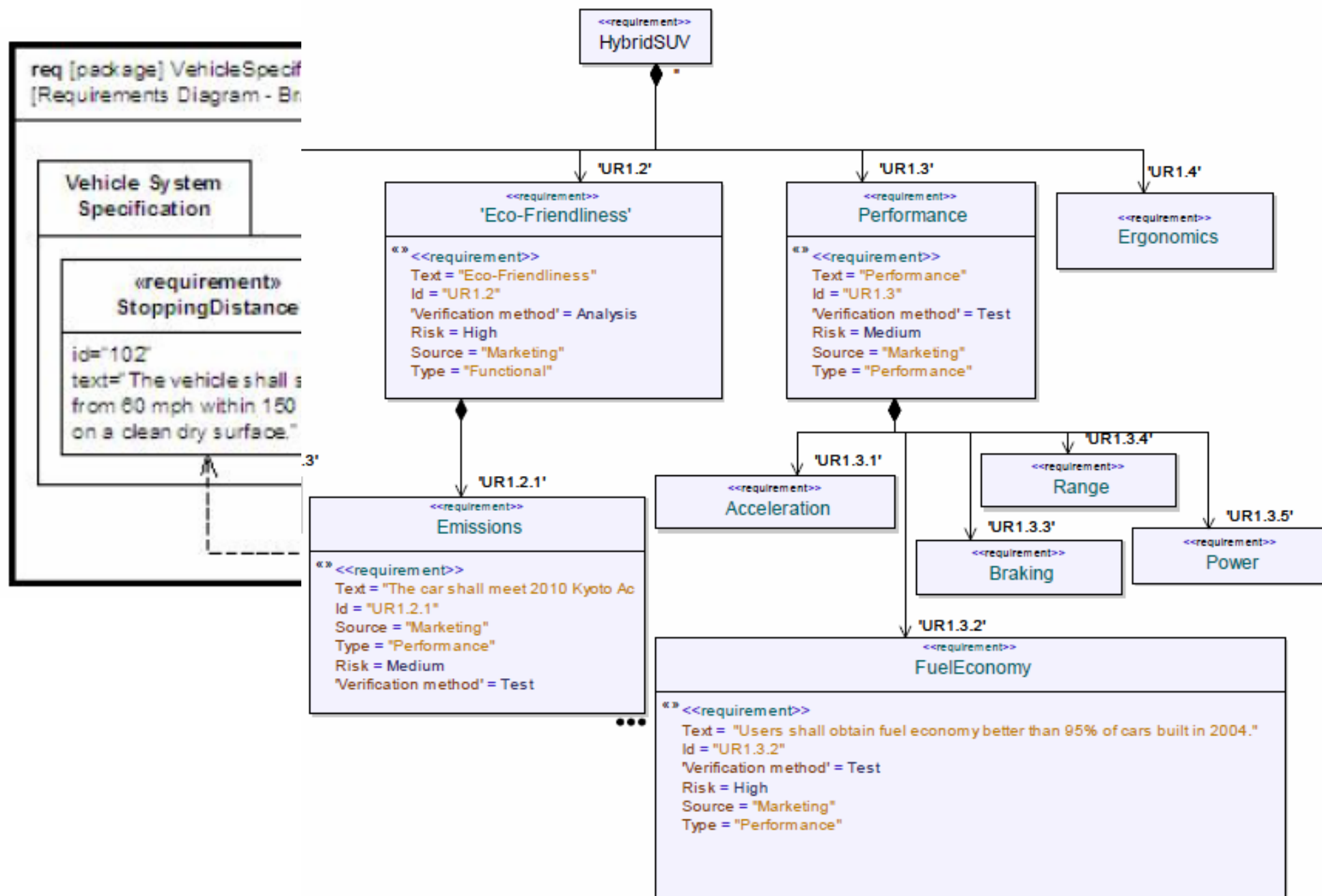
- Systems Modeling Language
 - UML részhalmoz egy kiterjesztése rendszertervezéshez
 - Fő újdonságok: Requirement és Parametric diagram



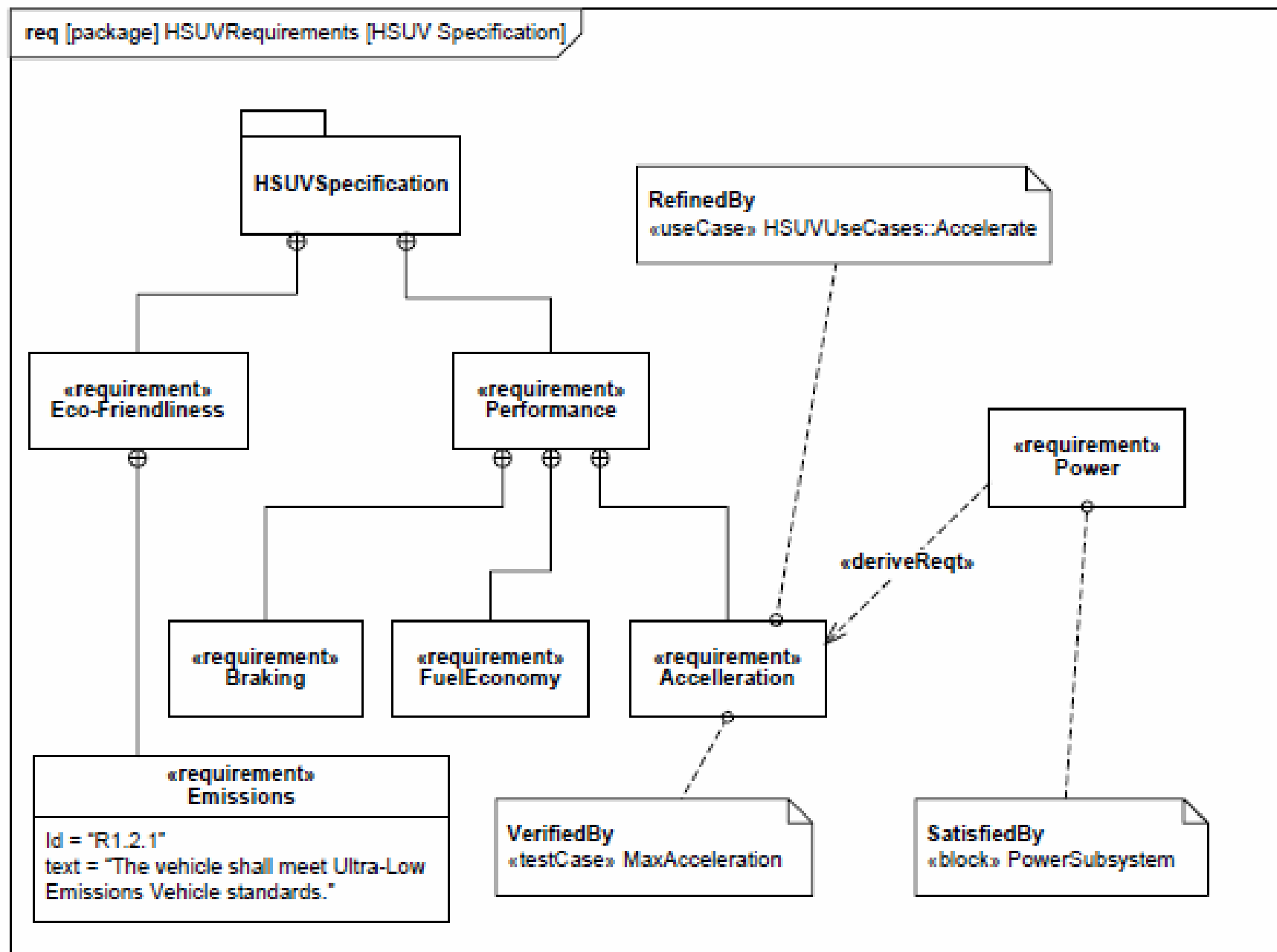
Requirements diagram

- **Követelmények (szöveges is) tárolása azonosítóval**
 - <<requirement>> stereotype
 - Id és text mezők
 - Felhasználói attribútumok: pl type, source, risk, ...
 - Táblázatos forma is támogatott
- **Követelmények hierarchikus csomagokba rendezhetők**
 - Funkcionális, teljesítmény, ... kategóriák
- **Követelmények közötti finomítás (~ subclass), kompozíció**
- **Relációk használhatók (callout: megjegyzésekben):**
 - Copy: követelmények között (master – slave)
 - Trace: követelmények között (client – supplier)
 - DeriveReq: követelmények között (forrás – származtatott)
 - Refine: követelmények és terv elemek között (pl. szövegeshez)
 - Satisfy: követelmények és terv vagy implementáció elemek között
 - Verify: követelmények és teszt elemek között

Requirements diagram példa: Struktúra

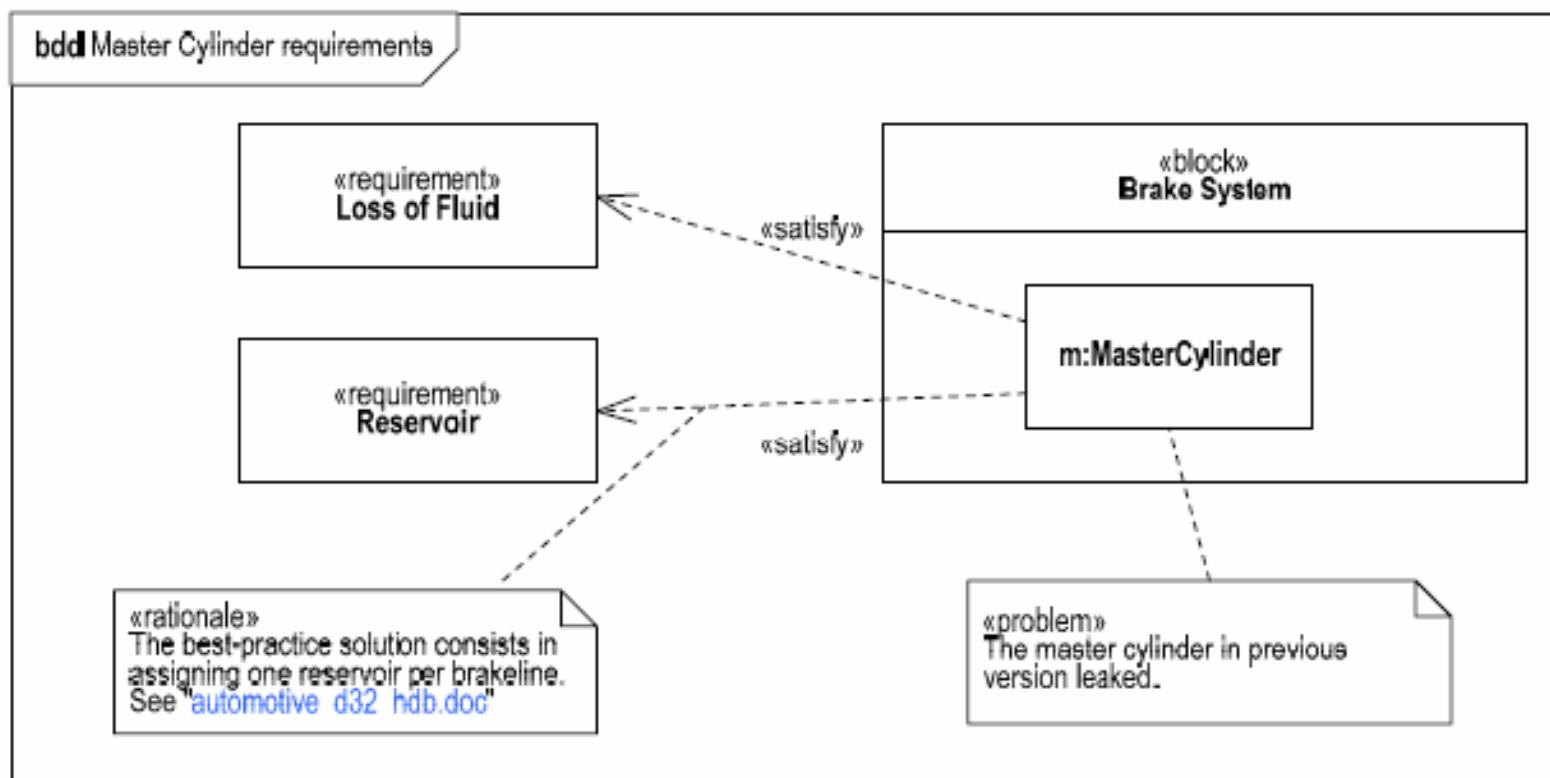


Requirements diagram példa: Relációk



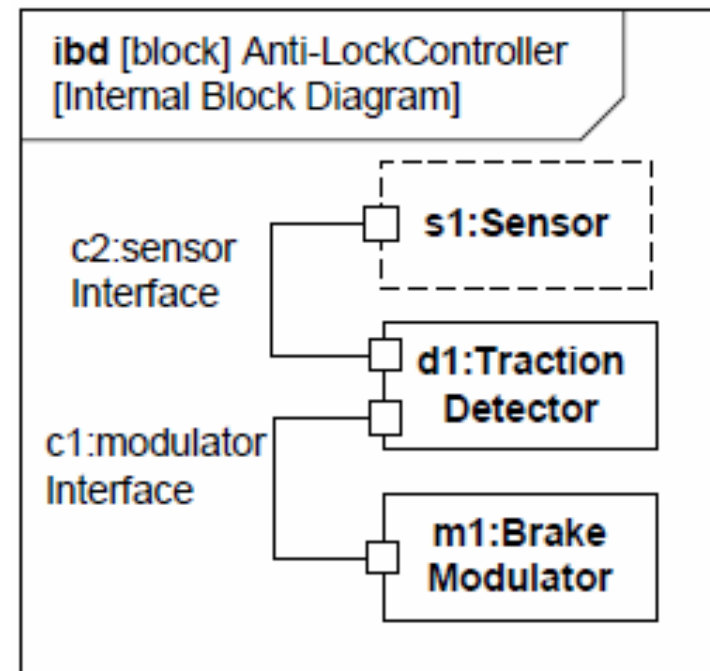
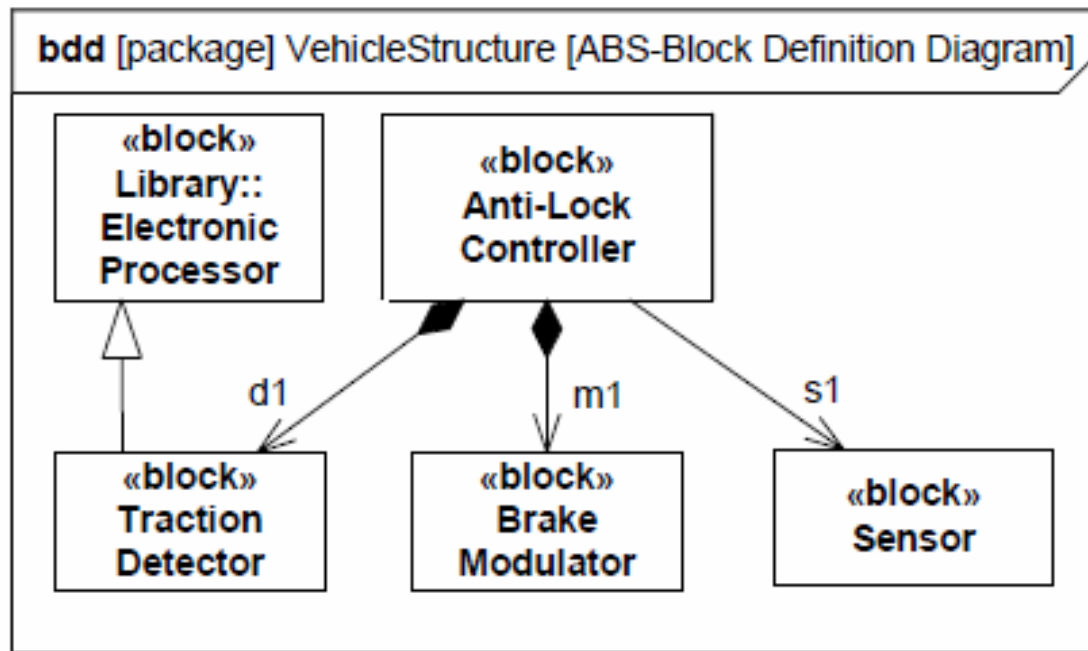
Requirements diagram: Döntések

- Tetszőleges modell elemhez köthető speciális megjegyzések (előredefiniált stereotype):
 - <<problem>>: Probléma, döntést igénylő felvetés
 - <<rationale>>: Megoldás, magyarázat



Block diagram

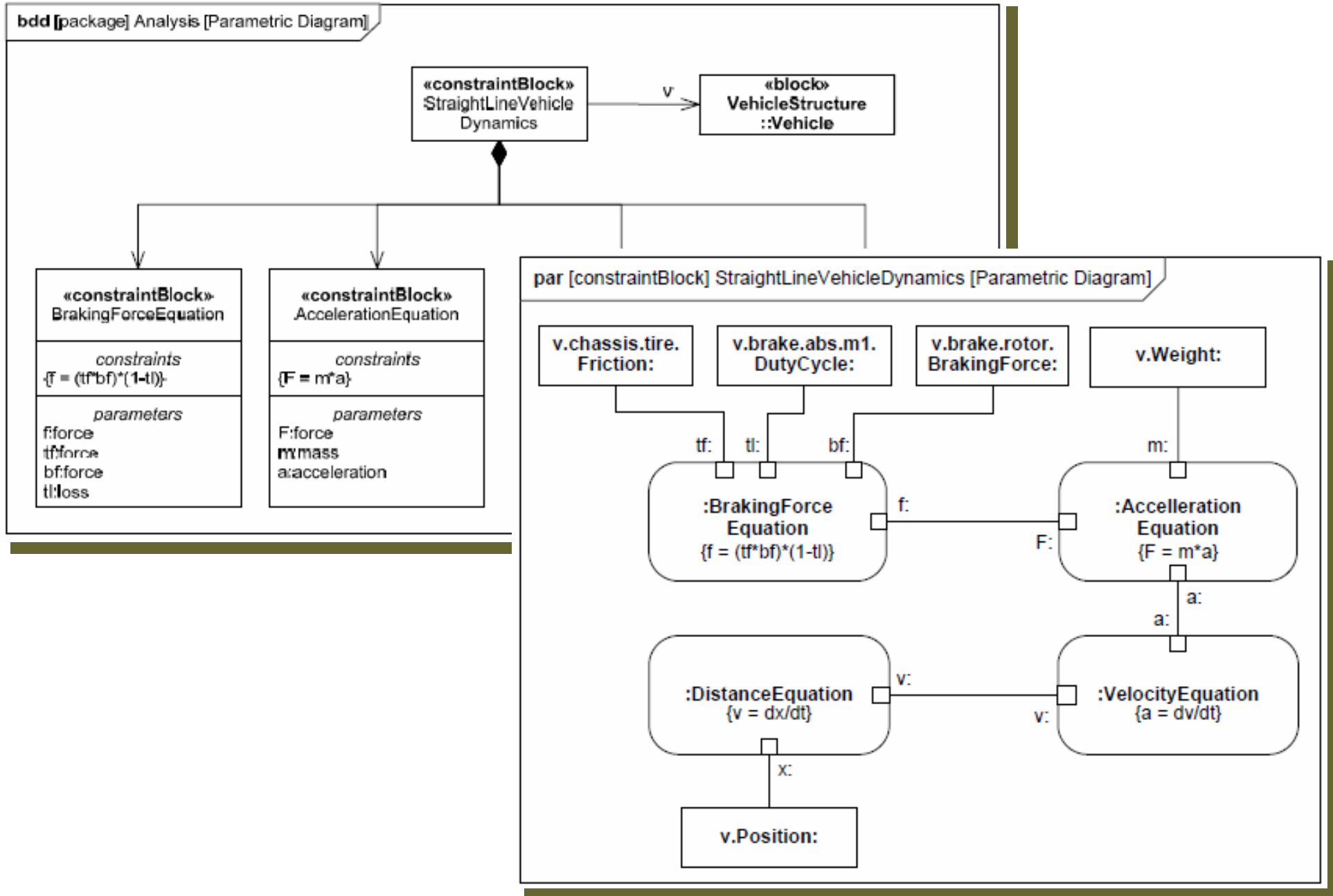
- Block definition diagram:
 - Block: A struktúra eleme (fekete / üveg doboz)
 - Komponens (nem csak szoftver)
 - A SysML-ben az UML 2.0 osztályokon alapul
- Internal block diagram:
 - Konkrét szerepek; típust a Block adja meg



Parametric diagram

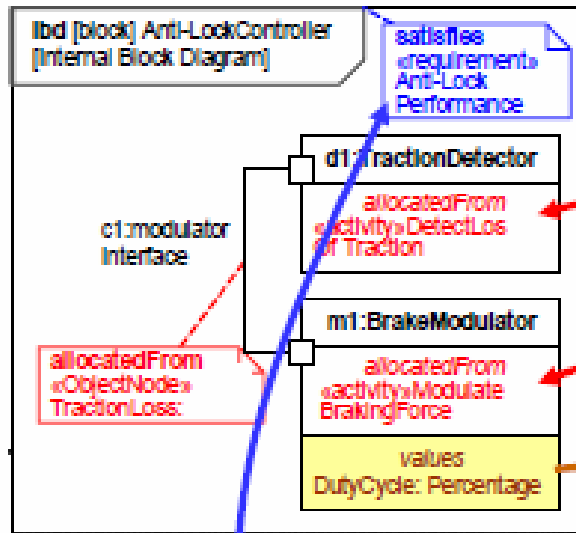
- **Cél: Ellenőrizhető számszerű követelmények (kényszerek) megfogalmazása tulajdonságokra**
 - Nem-funkcionális követelmények aspektusa
 - **Analízis** (pl. teljesítmény, megbízhatóság) támogatása
- **ConstraintBlock: Összefüggések megadása**
 - **Formális** (pl. MathML, OCL), vagy **informális** alakban
 - **Analízis eszközhöz igazítható** (nem SysML specifikus)
- **Parametric diagram: Alkalmazás**
 - Az összefüggések (Constraint block) **alkalmazása** egy adott környezetben
 - **Kötések értékek között**

Parametric diagram példa

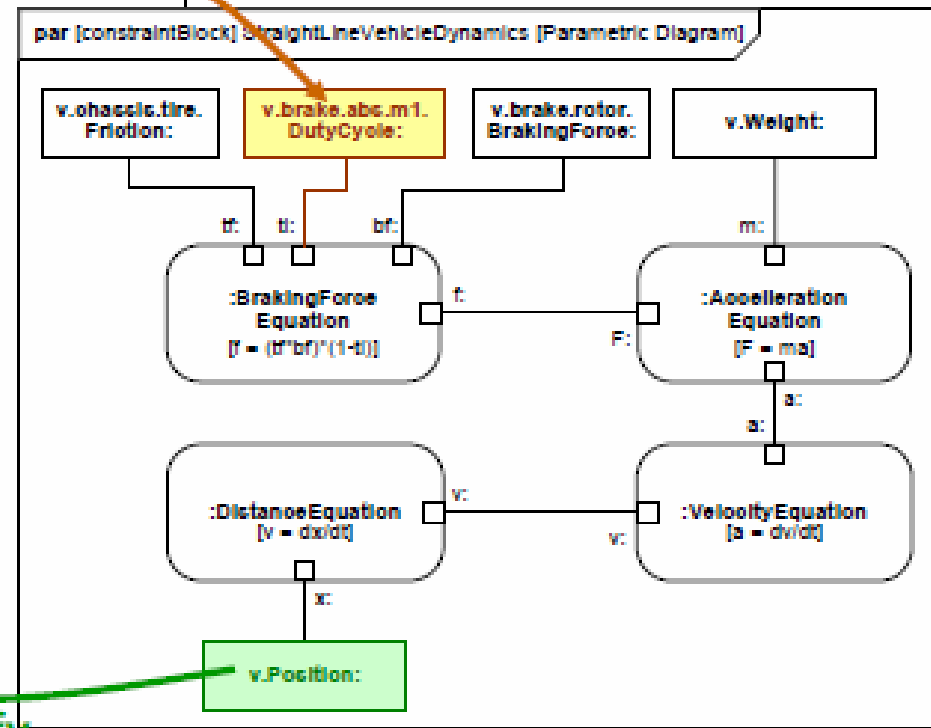
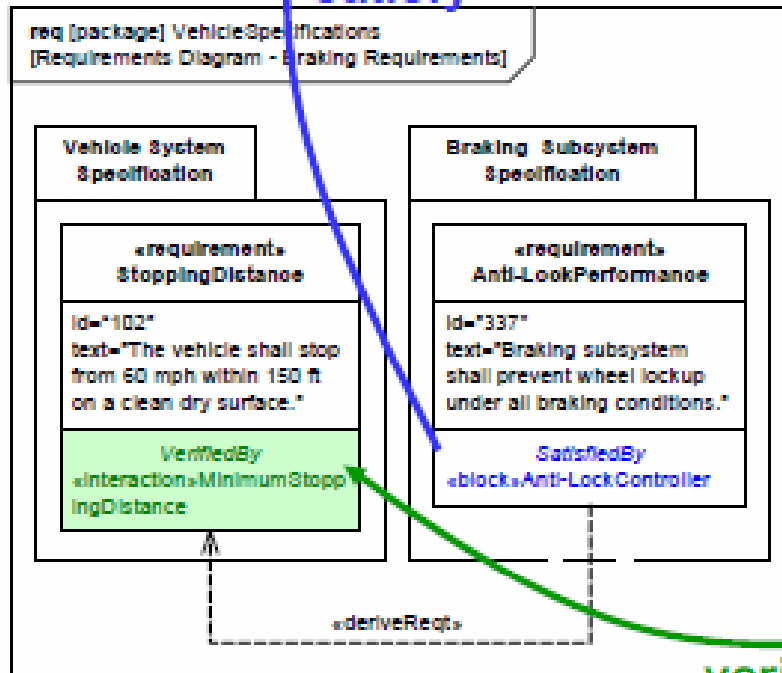
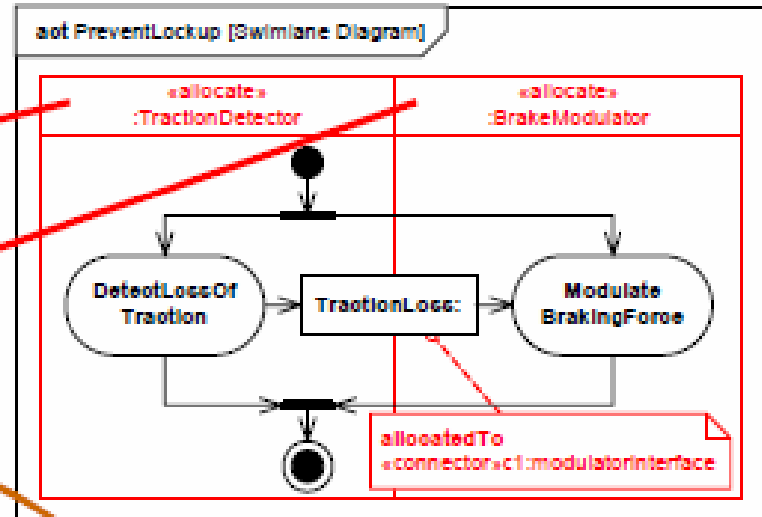


Relációk diagramok között: Követhetőség

1. Structure



2. Behavior



3. Requirements

4. Parametrics

verify

satisfy

allocate

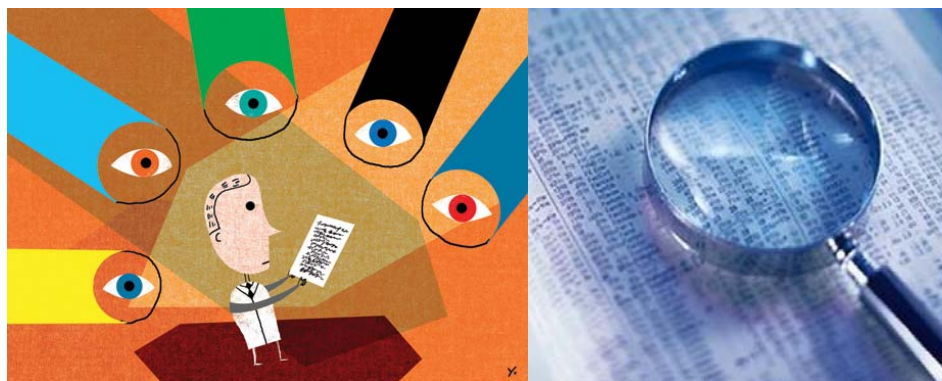
value binding

Tartalomjegyzék

- A fázis ki- és bemenetei
- A szoftverkövetelmény-specifikáció elkészítése
 - Formális nyelvek (áttekintés)
 - Félformális és strukturált technikák
 - Példa: SysML
- **Ellenőrzési feladatok**
 - Általános szempontok és módszerek
 - Teljesség és ellentmondás-mentesség
- Követelménykezelés
 - Feladatok
 - Automatikus eszközök

Verifikációs feladatok

- **Szoftverkövetelmény-specifikáció ellenőrzése**
 - Statikus analízis
 - Hiányosságok, ellentmondások kiszűrése végrehajtás nélkül
 - Analógia: Átolvasás „sorról sorra”
 - Ellenőrző listák
 - Tipikus hibák esetén hatékony (újra ne kövessük el)
 - Teljességet nem várhatunk
- **Szoftverkövetelmény-teszt-specifikáció ellenőrzése**
 - Végrehajtás (ki, mikor) és technika (hogyan) meghatározott-e?
 - Siker, kudarc ismérve adott-e?
- **Megvalósítás:**
 - Felülvizsgálat vagy egyenrangú átvizsgálás





How the customer explained it



How the Project Leader understood it



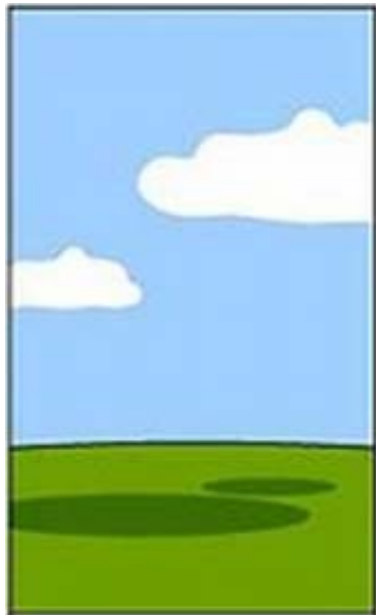
How the Analyst designed it



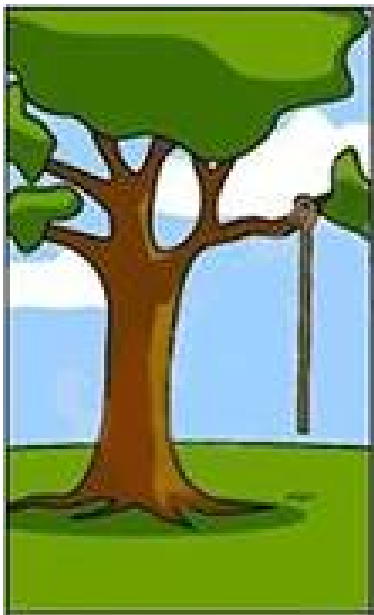
How the Programmer wrote it



How the Business Consultant described it



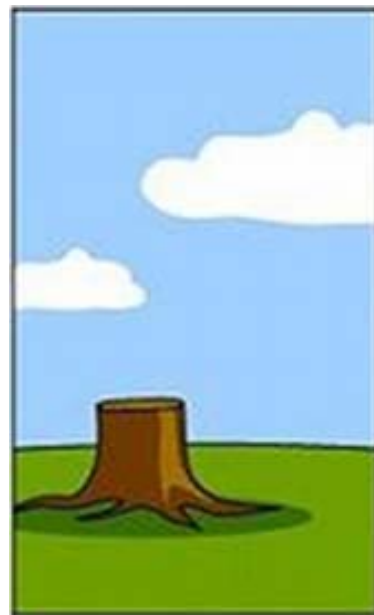
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

Általános ellenőrzési szempontok

- **Teljesség**
 - funkciók, hivatkozások, eszközök
- **Konzisztencia**
 - külső és belső
 - követhetőség
- **Megvalósíthatóság**
 - erőforrások
 - használhatóság
 - karbantarthatóság
 - kockázatok: költségbeli, technikai, környezeti
- **Tesztelhetőség**
 - specifikus
 - egyértelmű
 - számszerűsíthető

A jó specifikáció jellegzetességei: IEEE Std 830-1998

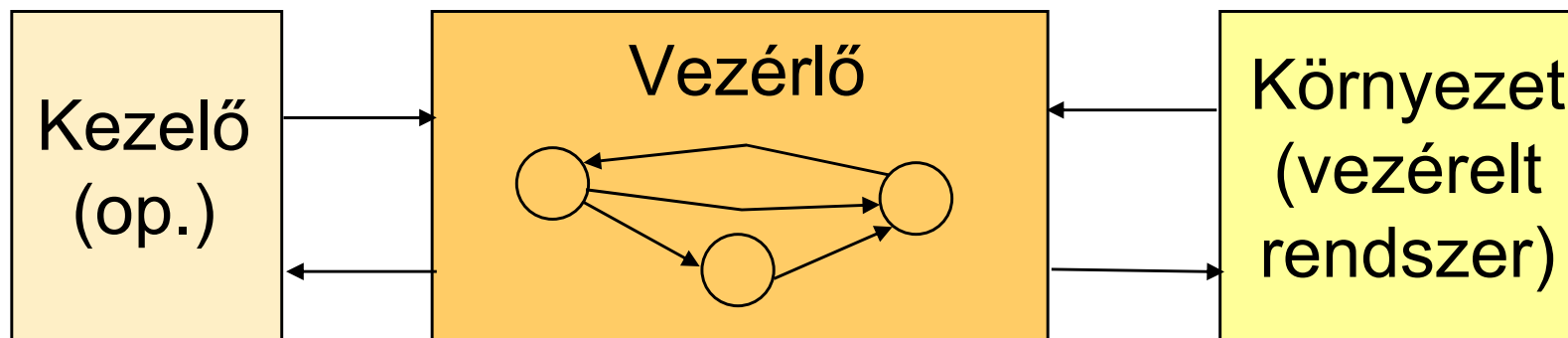
- **Helyes**
 - A szoftverre vonatkozó követelményeknek (elvárásoknak) megfelelő
 - Konzisztens a külső forrásokkal (pl. szabványok)
- **Egyértelmű**
 - Nem félreérthető, egy jelentése van
(Hasznosak a formális, félformális specifikációs nyelvek)
- **Teljes**
 - Minden (érvényes, érvénytelen) bemenetre van specifikált viselkedés
 - TBD csak indoklással és a feloldás módjával
- **Konzisztens**
 - Nincs belső ellentmondás, egységes a terminológia
- **Fontosság és stabilitás szempontjából rendezett**
 - Követelmények szükségessége, változatlansága felmérve
- **Ellenőrizhető**
 - Megállapítható egyértelműen, ha nem teljesül egy követelmény
- **Módosítható**
 - Nem redundáns, jól strukturált, jól elválasztott követelmények
- **Követhető**
 - Eredet becsatolható, további hatások hivatkozhatók

Teljesség és ellentmondás-mentesség

- Motiváció: Sok hiba visszavezethető hiányos vagy ellentmondásos specifikációra
 - Példa: Voyager és Galileo űrszondák szoftver tesztelése során felfedezett hibák statisztikája
 - 78% (149/192) specifikációs hiányosság, ebből
 - 23% veszélyes állapotban ragadás (nincs kilépés)
 - 16% időzíítési kényszerek megadásának hiánya
 - 12% nincs specifikált reakció külső eseményre
 - 10% bemeneti érték ellenőrzésének hiánya
- Megoldás lehet:
 - Szigorú specifikációs nyelv
 - Ellenőrzött tervezési minták használata
 - Specifikáció ellenőrzése (verifikáció)

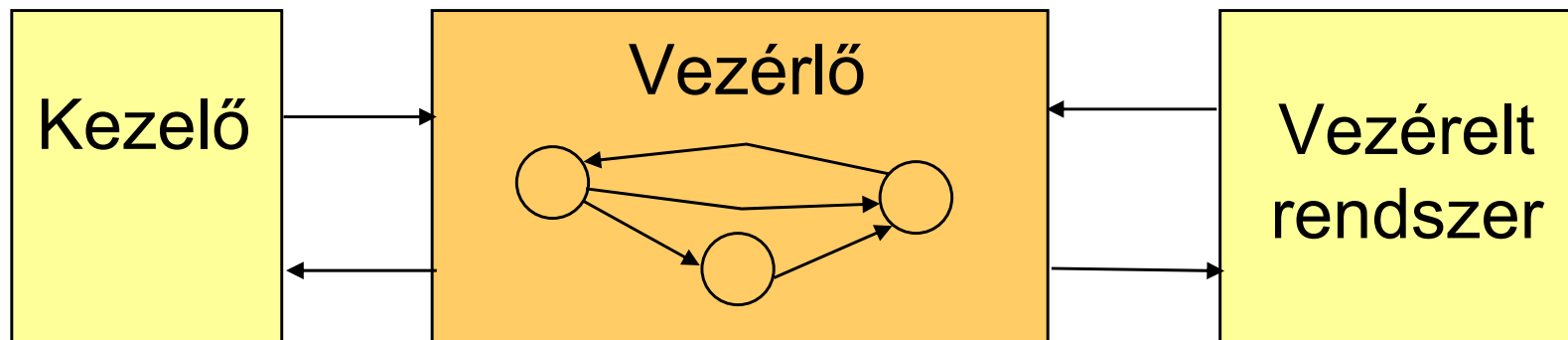
Példa: Reaktív rendszerek specifikációjának ellenőrzése (N. Leveson)

- Állapotdefiníció
- Bemenetek (események) specifikációja
- Kimenetek specifikációja
- Kimenetek és trigger kapcsolata
- Állapotátmenetek specifikációja
- Ember-gép interfész specifikációja



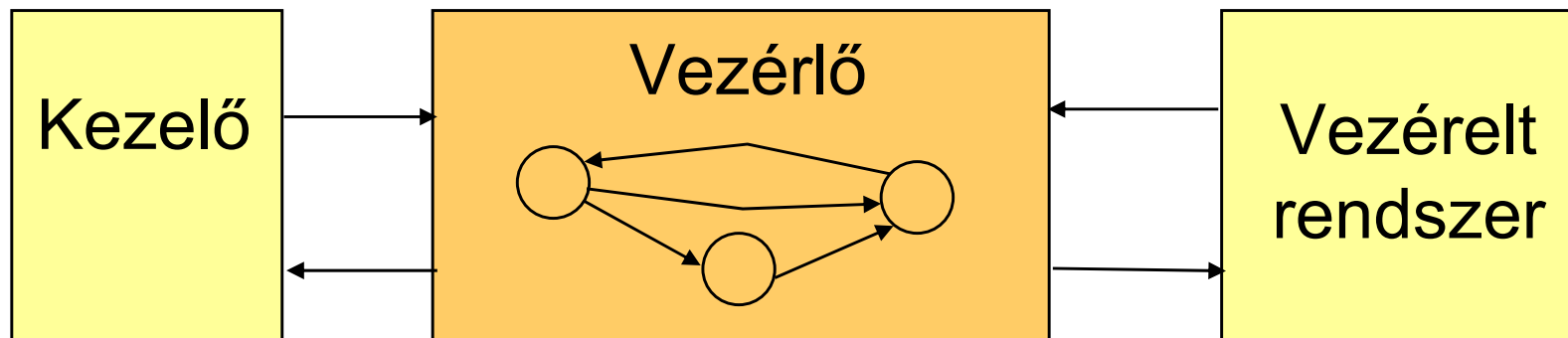
Vizsgálati szempontok

- Állapotdefiníció
- Bemeneti események
- Kimenet
- Kimenet (kimaradó bemeneti események esetén time-out és nincs vezérlés)
- Állapot
- Ember-gép interfész



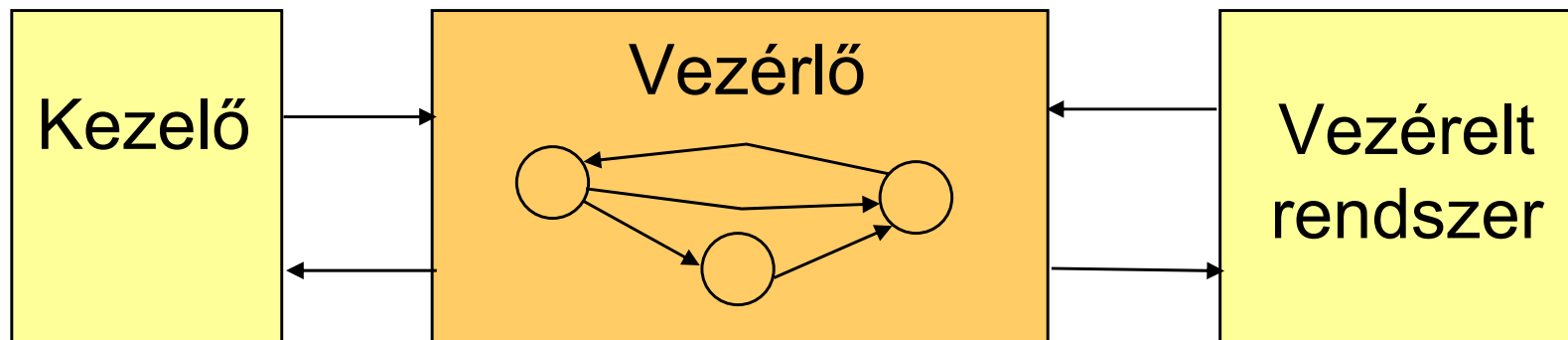
Vizsgálati szempontok

- **Állapotdefiníció**
 - **Bemenetek (események)**
 - **Kimenetek**
 - **Kime**
 - **Állap**
 - **Embe**
- Minden bemenetre van megadott reakció
 - Egyértelmű (determinisztikus) reakciók
 - Van bemeneti ellenőrzés (érték, idő)
 - Hibás bemenet kezelése
 - Megszakítások gyakorisága specifikált



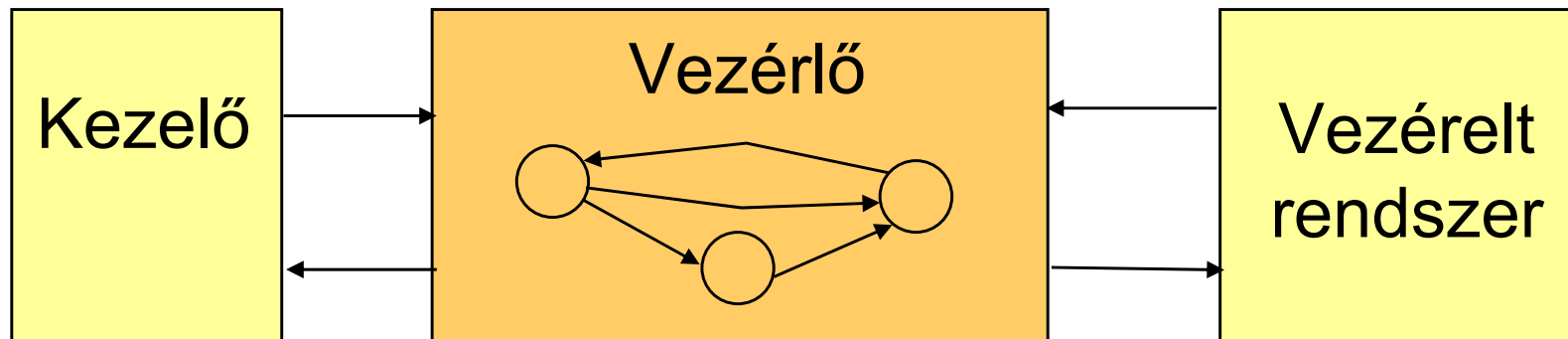
Vizsgálati szempontok

- **Állapotdefiníció**
 - **Bemenetek (események)**
 - **Kimenetek**
 - **Kimenetek**
 - **Állap**
 - **Embe**
- Hihetőség-vizsgálat kritériumai adottak
- Fel nem használt kimenetek ellenőrzése
- Környezeti feldolgozó kapacitás betartva



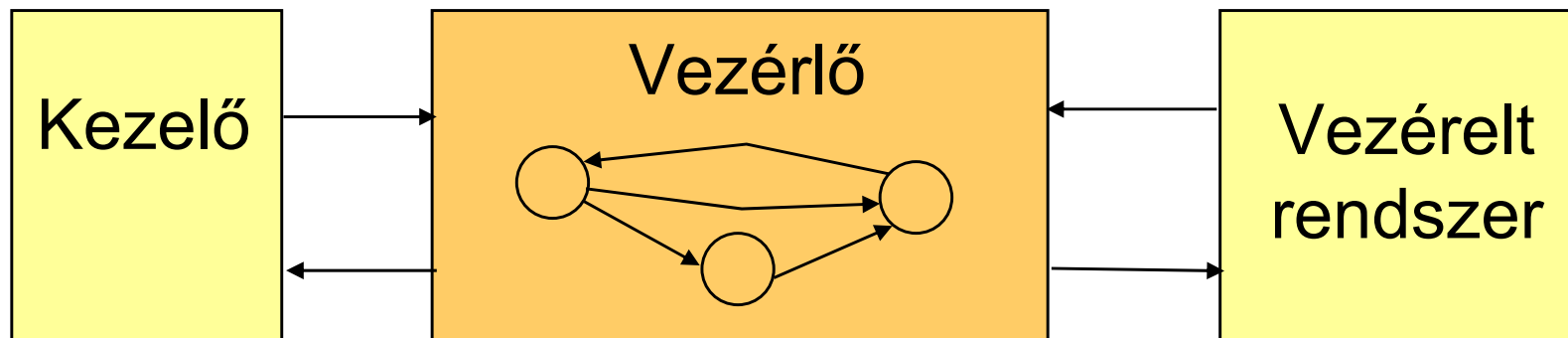
Vizsgálati szempontok

- **Állapotde** - Kimenetek hatásának ellenőrzése a bemeneteken keresztül
- **Bemenet** - Szabályzási kör stabilitása
- **Kimenetek**
- Kimenetek és trigger kapcsolata
- **Állapotátmenetek**
- **Ember-gép interfész**



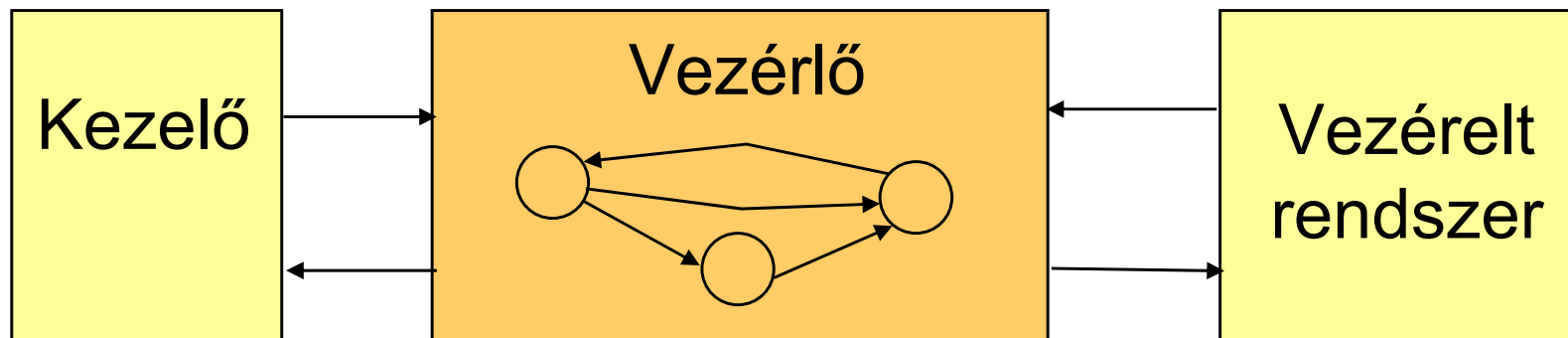
Vizsgálati szempontok

- **Áll** - Minden állapot statikusan elérhető
- **Be** - Állapotátmenetek visszafordíthatók
- **Kir** - Több átmenet veszélyes állapotból biztonságosba
- **Kin** - Megerősített átmenet biztonságosból veszélyes állapotba
- Állapotátmenetek
- Ember-gép interfész



Vizsgálati szempontok

- **Állapotátvitel**
- **Beviteli események**
- **Kimeneti események**
- **Kimeneti események specifikációja:**
 - Sorrendezés (prioritás)
 - Frissítés
 - Időbeliség (kezelő terhelhetősége)
- **Állapotátviteli események**
- Ember-gép interfész



Tartalomjegyzék

- A fázis ki- és bemenetei
- A szoftverkövetelmény-specifikáció elkészítése
 - Formális nyelvek (áttekintés)
 - Félformális és strukturált technikák
 - Példa: SysML
- Ellenőrzési feladatok
 - Általános szempontok és módszerek
 - Teljesség és ellentmondás-mentesség
- **Követelménykezelés**
 - Feladatok
 - Automatikus eszközök

A követelménykezelés feladatai (áttekintés)

- Követelmények hatékony, strukturált tárolása
 - Hierarchikus elrendezés, tulajdonságokkal
- Követelmény életciklus támogatása
 - Felvétel, törlés, változás, finomítás, kapcsolatok megjelenése
- Követelmények finomítása és kapcsolódása
 - Specifikáció -> Architektúra terv -> Modulterv -> Forráskód -> Teszt
- Analízis lehetőségek
 - **Hatás** analízis (impact analysis): változáskezelés
 - Mit befolyásol, ha a követelmény megváltozik?
 - **Eredet** analízis (derivation analysis): költség-haszon elemzés
 - Milyen követelményre vezethető vissza? Miért van itt, szükséges-e?
 - **Fedettsé**g analízis (coverage analysis): projekt követés
 - Mely követelmények nincsenek implementálva / tesztelve?

Követhetőség
(traceability)
szükséges

A követelménykezelés „kézi” módszerei

- Természetes nyelvű követelmény dokumentum
 - Strukturálás adott (fejezetek, alfejezetek)
 - Követelményazonosítók felvétele
- Követelményfinomítás: Táblázatos nyilvántartás
 - Követelményazonosítók szerepelnek
 - Különböző dokumentumokból (SRS, SA, MDS, MTS, ...)
 - Analízis makrókkal támogatható
 - Üres, többszörös, ... mezők kikeresése
- Követhetőségi mátrix: Táblázatos forma
 - Követelmények azonosítói
 - Kódrészlet azonosítók (funkció szint tipikus)
 - Teszt azonosítók
 - Sikeres/sikertelen teszteredmény bejelölése

Automatikus követelménykezelők feladatai

Követelmények nyilvántartása	Hierarchikus felépítés
Kapcsolatok nyilvántartása	Sokféle reláció: Kompozíció, származtatás, finomítás, bizonyítás, .. Követelmény – Modell – Kód – Teszt – Teszteredmény között
Követelmény változások kezelése	Időbeli struktúra, triggerek
Navigáció a kapcsolatokon	Előre: pl. hatás analízishez Vissza: pl. eredet analízishez
Fedettségi listák készítése	Lefedetlen követelményekhez Indokolatlan megvalósításhoz
Jogosultságok kezelése	Hozzáférés szerepek
Értesítési rendszer	Változások
Biztonsági megoldások	Sértetlenség

Megvalósítás: Strukturált tárolás

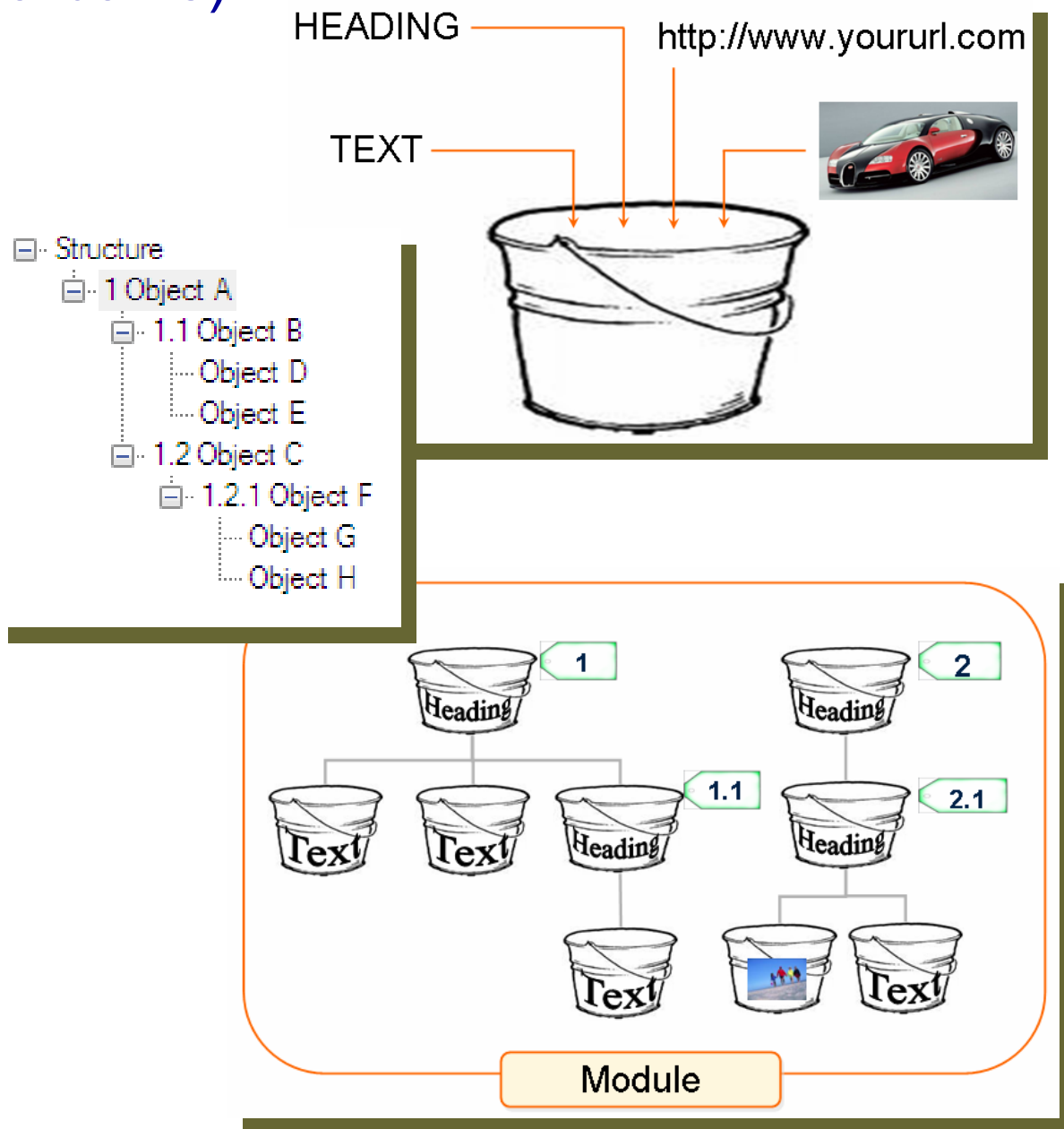
- Objektum modell (adatbázis)

- Általános „tároló”
- Egyedi azonosító
- Modulok

- Hierarchia

- Tulajdonságok

- Fejléc / szöveg
- Hozzáférési jogok
- Történet
- Attribútumok
 - Prioritás
 - Státusz
 - Költség
 - ...
- Linkek



Példa: IBM Telelogic DOORS

ID	Last Modified By	Car user requirements	Priority	Percentage cost	Comments
TRN-CSR-1	Bill Young	1 Introduction	Mandatory	0.172835	
TRN-CSR-2	Bill Young	This module contains the user requirements for a new car to be commercially available by 1 August 2006.	Mandatory		Use form text a text field.
TRN-CSR-3	Bill Young	2 User types	Desirable	1.370889	
TRN-CSR-4	Bill Young	2.1 Nationalities	Mandatory	0.642687	
TRN-CSR-5	Bill Young	The car will be used in the following countries: UK, USA, Northern Europe, Eastern Europe, Japan, Russia, Australia.	Mandatory	0.769025	
TRN-CSR-6	Bill Young	2.2 User sizes	Mand:		
TRN-CSR-7	Bill Young	People come in all shapes and sizes. The car must be suitable for people maximum and minimum sizes of fgfgf to 2 m weighing 25 kilograms to	Mand:		

Tulajdonságok

Objektum azonosító

Változás-jelző

Fejléc objektum

Szöveges objektum

Hierarchia

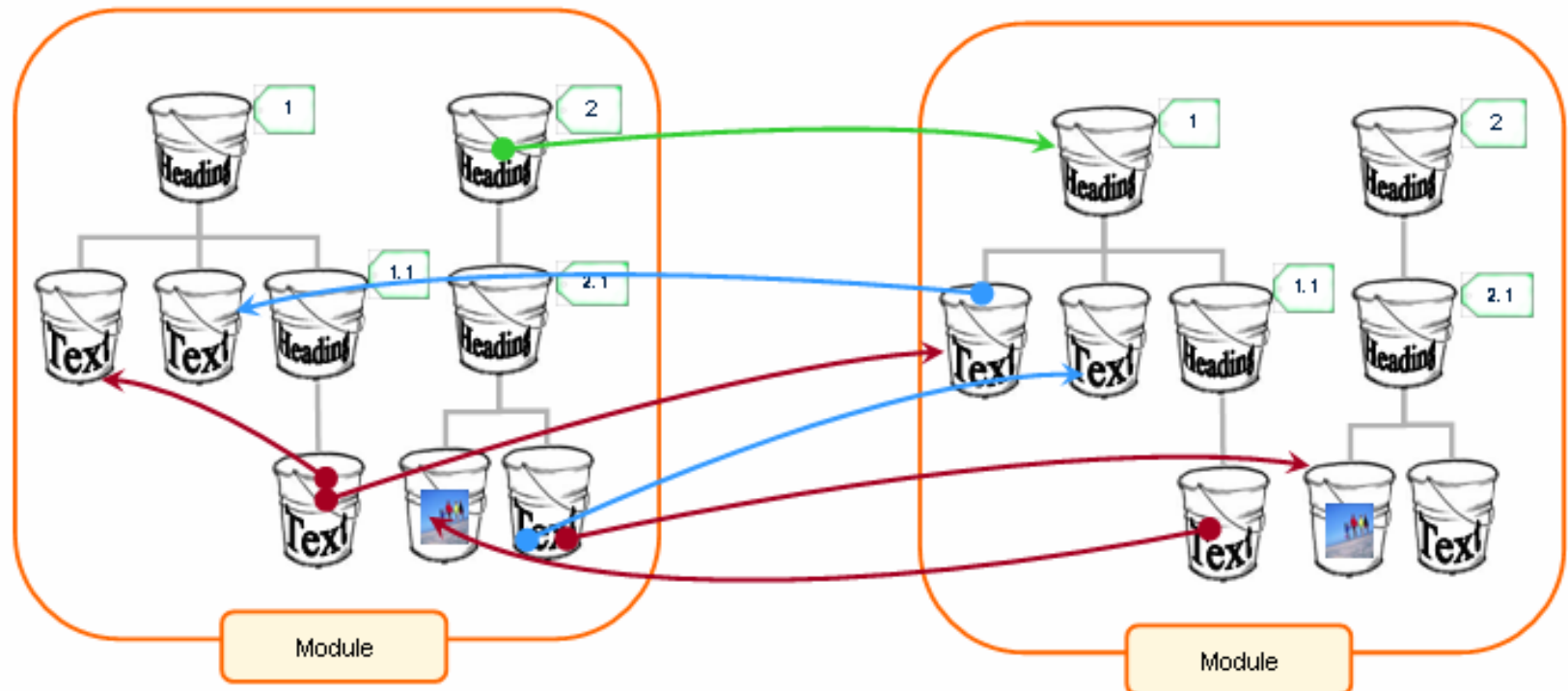
The screenshot shows the 'Structure Module' window in IBM Telelogic DOORS. On the left, a hierarchy tree is visible with '1 Heading 1' expanded to show 'Requirement 1', 'Requirement 2', and 'Requirement 3', and '2 Heading 3' expanded to show 'Requirement 5' and 'Requirement 6'. On the right, a table lists requirements with their IDs and headings:

ID	Structure Exercise
TRN-DF4-1	1 Heading 1
TRN-DF4-8	Requirement 1
TRN-DF4-9	Requirement 2
TRN-DF4-10	Requirement 3
TRN-DF4-7	2 Heading 3
TRN-DF4-6	Requirement 5
TRN-DF4-11	Requirement 6

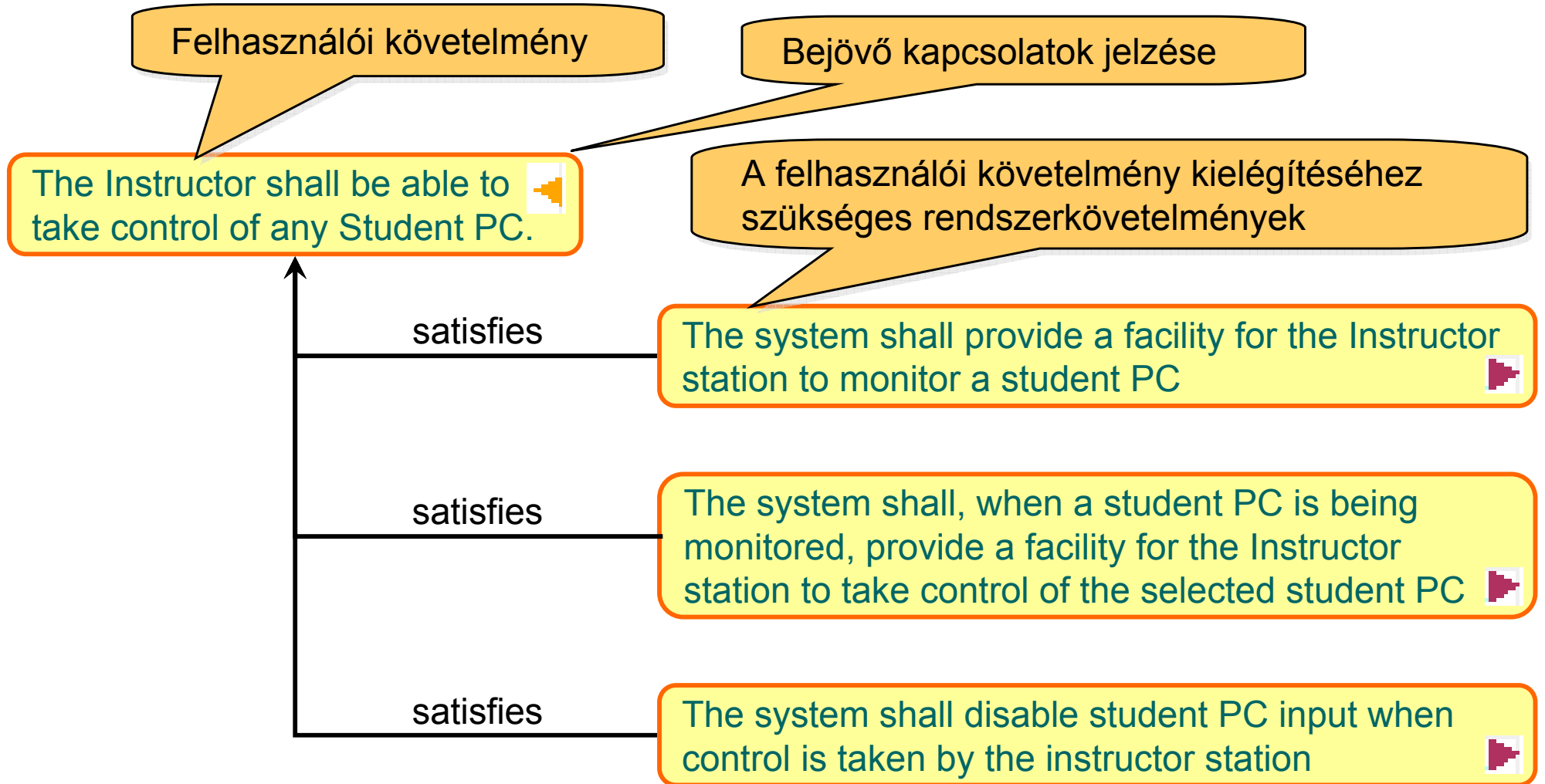
At the bottom of the window, it shows 'Username: Bill Young' and 'Exclusive edit mode'.

Megvalósítás: Kapcsolatok (linkek)

- Relációk: Rendezett párok
 - Objektumok között
 - Külső kapcsolatok
- Típusok
 - Finomítja
 - Kielégíti
 - Teszteli
 - ...



Példa: DOORS



Megvalósítás: Követelmény életciklus

- **Objektum szintű változások**

- Közvetlen szerkesztés (létrehozás, módosítás, törlés)
- Megváltozott kapcsolatok jelzése (suspect link)
- **Változtatási javaslatok** menedzselése (elkészítés, csoportosítás, felülvizsgálat, érvényre juttatás)

- **Változások mint események**

- Scriptek indítására használhatók (pl. teszt érvénytelenítés)

- **Nagyléptékű változások**

- **Baseline** definiálás (állapot rögzítése)
 - Inkrementális változások vizsgálhatók
 - Összehasonlítás lehetséges
- Követelmény-partíciók **kiajánlása**, távoli szerkesztés, szinkronizálás, visszavétel

Megvalósítás: Analízisek

- Követhetőség alapja:
Navigálás a kapcsolatokon keresztül
 - Kiterjedés (scope) és mélység kijelölhető
 - Irány kijelölhető (előre, hátra)
- Script nyelv használható
 - Bejárás, kigyűjtés
 - Tulajdonságok megváltoztatása
- Jelentések készítése
 - **Hatás** analízis: Előre navigálás alapján
 - **Eredet** analízis: Hátra navigálás alapján
 - **Fedettség** analízis
 - Szűrés: Navigálás kód objektumokig, teszt objektumokig
 - Objektumok kigyűjtése: Nincs kapcsolat adott célhalmazig
 - Pl. nincs megvalósítás, nincs sikeres teszt

Megvalósítás: Járulékos funkciók

- **Nézetek** az objektum (követelmény) listában
 - Szűrés hierarchiaszintekre, tulajdonságokra, ...
- **Űrlapok** a tulajdonságok szerkesztésére
- **Megosztott szerkesztés** (csoportmunka)
 - Objektum (követelmény) szintű zárolás
- **Dokumentáció generálás**
 - Hierarchia (=> fejezetek) alapján rendezett szöveges és egyéb objektumok
 - **Exportálás**: Strukturált külső dokumentumban (pl. Word) történt változtatások vissza is olvashatók (struktúra megőrzése)
 - **Importálás**: Előkészítés szükséges
- **Webes hivatkozások** (pl. e-mailben küldhető)
 - Adatbázis, projekt, objektum hivatkozás

Követelmény alapú verifikációs eszközláncok

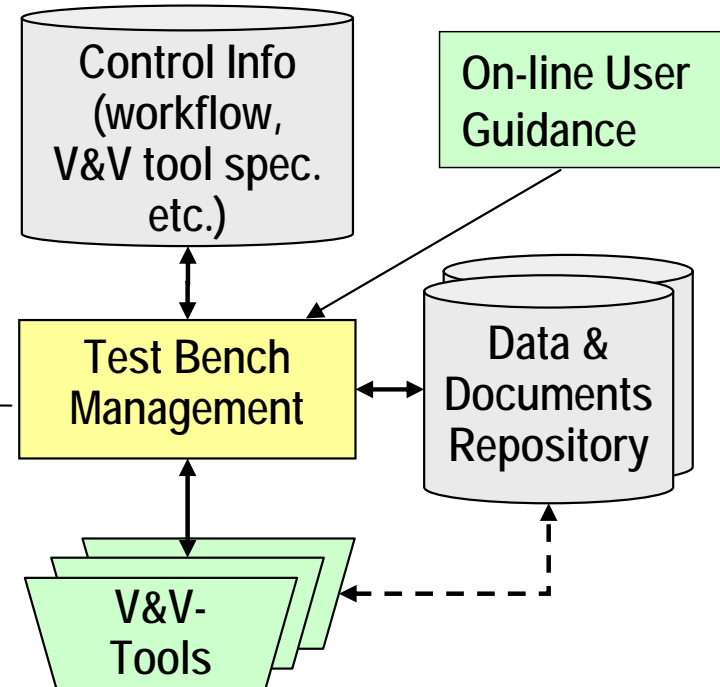
- **Verifikációs tevékenység felvétele a követelmények mellé**
 - Tervezett és rögzített verifikáció (pl. biztonságigazolás)
 - Ellenőrzések a követelmények és szabványok alapján
- **Verifikációs eszközláncok kialakítása (általában külső)**
 - **Analízis:** analízis modell generálása, analízis végrehajtása, eredmények visszacsatolása
 - **Tesztelés:** modell alapú tesztgenerálás, teszt végrehajtás, teszt eredmény kiértékelés
 - **Mérések:** mérési konfigurálása, végrehajtása, eredmények értékelése
- **Verifikációs eszközláncok indítása a követelménykezelőből**
 - Triggerek alapján; script nyelven programozható
- **Verifikáció státuszának rögzítése**
 - Ellenőrzött modell, sikeresen tesztelt követelmény
 - Eredmények tárolása: **adattár** (repository)

Verifikáció „menedzselése” a követelménykezelővel

Format module: /DECOS_TestBench/V-Plans/Components and Middleware' current 0.0 - DOORS

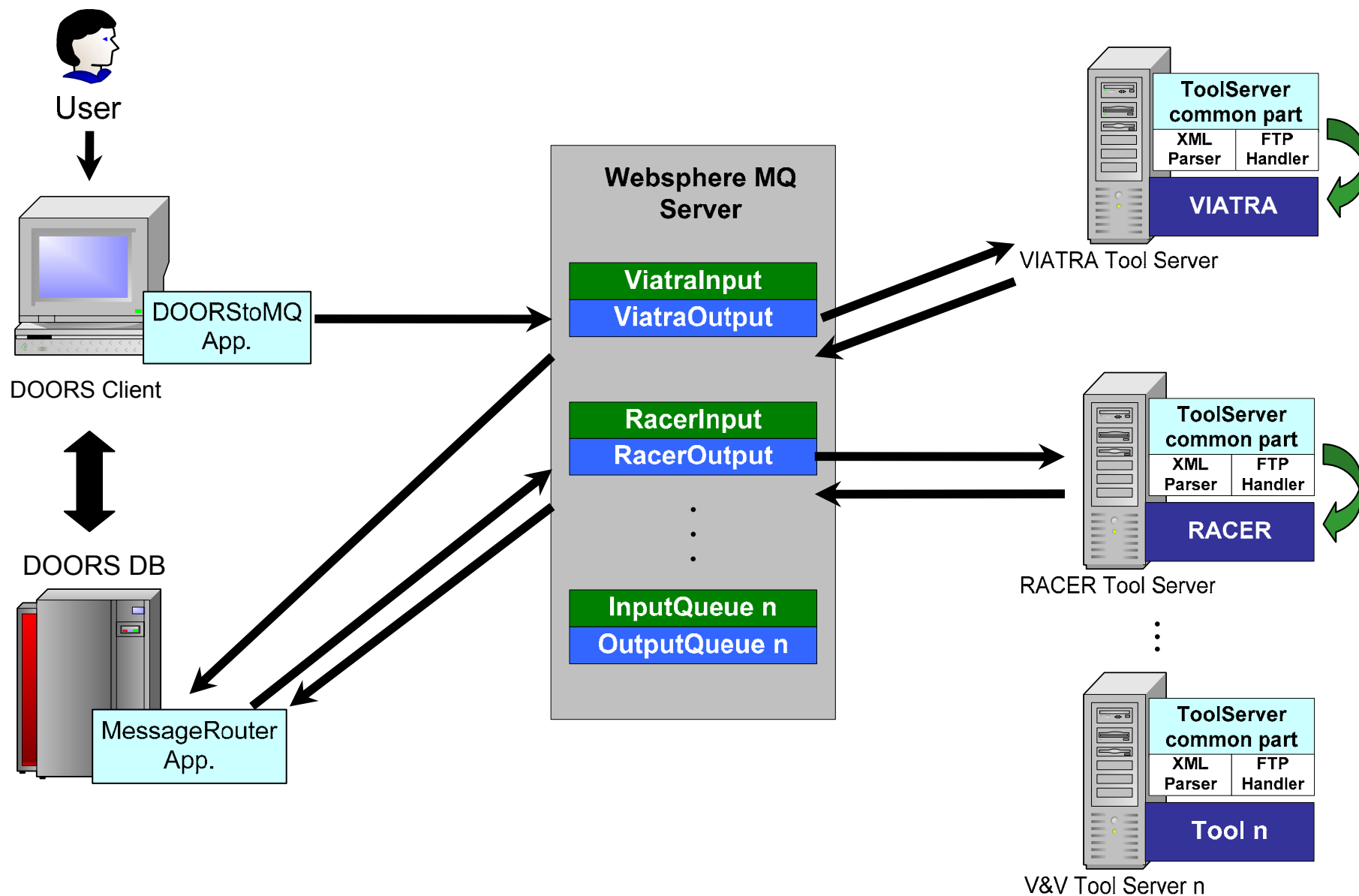
ID		VVStatus	Type	Phase	V&V-Activity
VPCM53	1 V-Plan Components and Middleware	Not ready			
VPCM1	Arch-gen-1	Not ready	Compound		
VPCM2	Arch-gen-core-1	Completed	Compound		
VPCM3	Arch-core-predictable-transport-1	Completed	Elementary		
VPCM4	Arch-core-ft-clock-sync-1	Completed	Elementary		
VPCM5	Arch-core-fault-isolation-1	Completed	Compound		
VPCM6	Arch-core-fault-hypothesis-1	Completed	Elementary		
VPCM7	Arch-core-never-give-up-1	Completed	Elementary		
VPCM8	Arch-core-transient-faults-1	Completed	Elementary		
VPCM9	Arch-core-consistent-diagnosis-1	Completed	Elementary		
VPCM10	Arch-gen-core-2	Not ready			
VPCM11	Arch-DECOS-high-level-service-1	Not ready			
VPCM12	Arch-DECOS-exec-1	Not ready			
VPCM13	Arch-DECOS-com-1	Not ready			
VPCM14	Arch-DECOS-com-2	Not ready			
VPCM15	Arch-DECOS-com-3	Not ready			

Implemented in DOORS™



- ITEM (Hazard and Risk Analysis)
- RACER (Formal Verification)
- SCADE MTC (Simulation)
- LDRA (Testing)
- PROPANE (Fault Injection)
- EMI Test Bench

Példa végrehajtás: PIM ellenőrzés RACER-rel



Összefoglalás

- A fázis ki- és bemenetei
- A szoftverkövetelmény-specifikáció elkészítése
 - Formális nyelvek (áttekintés)
 - Félformális és strukturált technikák
 - Példa: SysML
- Ellenőrzési feladatok
 - Általános szempontok és módszerek
 - Teljesség és ellentmondás-mentesség
- Követelménykezelés
 - Feladatok
 - Automatikus eszközök