5. Szkeleton tervezése

54 – *Override*

Konzulens:
dr. László Zoltán

Csapattagok:

Kriván Bálint	CBVOEN	balint@krivan.hu
Jákli Gábor	ONZ5G1	j_gab666@hotmail.com
Dévényi Attila	L1YRH0	devenyiat@gmail.com
Apagyi Gábor	X8SG3T	apagyi.gabooo@gmail.com
Péter Tamás Pál	N5ZLEG	falconsaglevlist@gmail.com

2011. március 13.

Tartalomjegyzék

5 Szkeleton tervezése **4**

5.1. Errata . . . 4
5.1.1. Objektumleírás: **Wire** . . . 4
5.1.2. Objektumleírás: **Node** . . . 4
5.1.3. Osztályleírás: **AbstractComponent** . . . 4
5.1.4. Osztályleírás: **Node** . . . 5
5.1.5. Osztályleírás: **Wire** . . . 5
5.1.6. Statikus struktúra diagramok . . . 6
5.2. A szkeleton modell valóságos use-case-ei . . . 7
5.2.1. Use-case diagram . . . 7
5.2.2. Use-case leírások . . . 7
5.3. Architektúra . . . 11
5.4. A szkeleton kezelői felületének terve, dialógusok . . . 11
5.5. Szekvencia diagramok a belső működésre . . . 12
5.6. Napló . . . 16

Ábrák jegyzéke

5.1. Statikus struktúra nézet 6
5.2. A szkeleton modell valóságos use-case-ei 7
5.3. Áramkör inicializálása 13
5.4. Kapcsoló és Led 14
5.5. 5-ös 15

5. Szkeleton tervezése

5.1. Errata

Az előző fejezetben leírtak egy apró részletben megváltoztak. Az elemeket már nem közvetlenül kötjük össze, hanem *vezeték*ek segítségével, melyeket egymással *csomópont*okkal lehet összekötni, ha szükséges. Így javítottuk a láthatósággal kapcsolatosan felmerült problémákat, ehhez fel kellett venni 2 új osztályt (Wire, Node), illetve az AbstractComponent módosítani, ezekhez tartozó objektum és osztályleírások alább olvashatóak, valamint mellékeltük a módosított statikus osztálydiagramot is. (Egy-két egyéb objektumleírás is módosult, de csak azért mert a kiértékelés logikája változott – nem hátulról megyünk, hanem az összes kiértékeli magát, ez nem szükséges a jelen fejezethez, hiszen magától értetődő)

5.1.1. Objektumleírás: **Wire**

Vezeték, mely az áramköri komponensek ki és bemeneteit köti össze. Egy vezeték egy darab kimenetet és egy darab bemenetet köt össze. A rajta lévő értéket le lehet tőle kérdezni, illetve be lehet azt állítani.

5.1.2. Objektumleírás: **Node**

Csomópont, mely a bemenetén lévő értéket a kimeneteire adja. Segítségével lehet egy vezetéket „szétágaztatni”.

5.1.3. Osztályleírás: **AbstractComponent**

Absztrakt osztály.

- Felelősség
  Egy komponens absztrakt megvalósítása, ebből származik az összes többi komponens. A közös logikát valósítja meg. A gyakran használt feladatokra ad alapértelmezett implementációt (pl. vezetékek bekötése). Tudja magáról, hogy a legutóbbi két kiértékelés között változtak-e a kimenetei.
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Attribútumok
  - `protected Wire[] inputs`: Bemeneteire kötött vezetékek.
  - `protected Wire[] outputs`: Kimeneteire kötött vezetékek.
- Metódusok
  - `addTo(Circuit c)`: Meghívja az áramkör `add(AbstractComponent ac)` metódusát.
  - `void evaluate()`: Komponens kimenetein lévő értékek kiszámolása a bemenetek alapján.
  - `boolean isChanged()`: Visszaadja, hogy a legutóbbi két kiértékelés között változtak-e a kimenetek.
  - `void setInput(int inputPin, Wire wire)`: Az adott bemeneti lábára rákötjük a megadott vezetéket.
  - `void setOutput(int outputPin, Wire wire)`: Az adott kimeneti lábára rákötjük a megadott vezetéket.

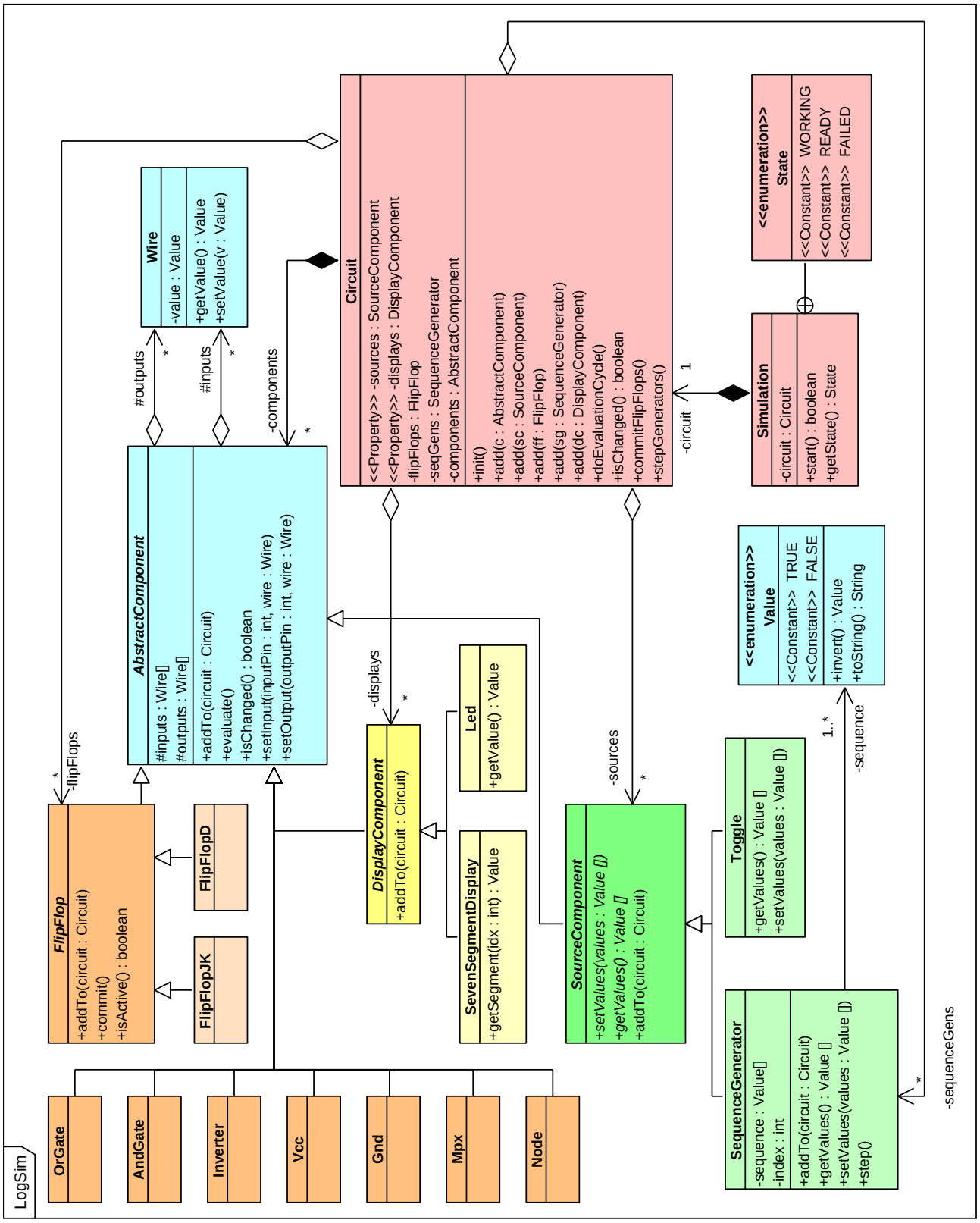
5.1.4. Osztályleírás: **Node**

- Felelősség
  Csomópont, mely a bemenetén lévő értéket a kimeneteire adja. Segítségével lehet egy vezetéket „szétágaztatni”.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

5.1.5. Osztályleírás: **Wire**

- Felelősség
  Vezeték, mely az áramköri komponensek ki és bemeneteit köti össze. Egy vezeték egy darab kimenetet és egy darab bemenetet köt össze. A rajta lévő értéket le lehet tőle kérdezni, illetve be lehet azt állítani.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)
- Attribútumok
  - `private Value value`: Vezetéken lévő érték
- Metódusok
  - `Value getValue()`: Visszaadja a vezetéken lévő értéket.
  - `void settValue(Value v)`: Beállítja a vezetéken lévő értéket.

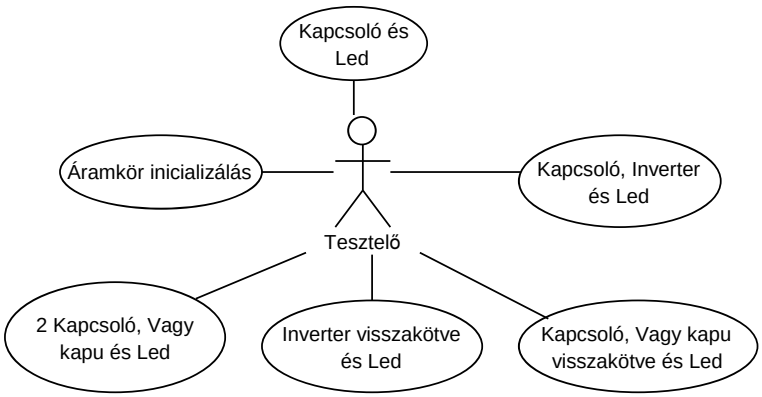
5.1.6. Statikus struktúra diagramok



5.1. ábra. Statikus struktúra nézet

5.2. A szkeleton modell valóságos use-case-ei

5.2.1. Use-case diagram



5.2. ábra. A szkeleton modell valóságos use-case-ei

5.2.2. Use-case leírások

Use-case neve	**Áramkör inicializálása**
Rövid leírás	Ez a usecase egy áramkör és a hozzá tartozó szimuláció inicializálását mutatja be, hogyan jönnek létre a komponensek és a közöttük lévő összeköttetés. Jelen példa egy Kapcsoló és egy Led összeköttetését prezentálja.
Aktorok	Tesztelő
Forgatókönyv	• szimuláció létrehozása • áramkör létrehozása • áramkör beregisztrálása a szimulációba • áramkör inicializálása – kapcsoló létrehozása – vezeték létrehozása – kapcsoló kimenetére vezeték kötése – led létrehozása – led bemenetére vezeték kötése – kapcsoló áramkörbe regisztrálása – led áramkörbe regisztrálása

Use-case neve	**Kapcsoló és Led**
Rövid leírás	Ez a usecase egy olyan áramkör tesztelését mutatja be, amely egy kapcsolóból és rá kötött ledből áll.
Aktorok	Tesztelő

Forgatókönyv	• Áramkör és komponensek létrehozása • kapcsoló értékének beállítása (**megkérdezi a tesztelőt**) • szimuláció indítása – hálózat kiértékelés indítása ∗ kapcsoló kiértékelése (állapotának kijelzése) ∗ led kiértékelése (világít/nem világít kijelzése) – áramkör változásának vizsgálata – stacionárius állapot, szimuláció vége

Use-case neve	**Kapcsoló, Inverter és Led**
Rövid leírás	Ez a usecase egy olyan áramkör tesztelését mutatja be, amely egy kapcsolóból egy rá kötött inverterből és egy arra kötött ledből áll.
Aktorok	Tesztelő
Forgatókönyv	• Áramkör és komponensek létrehozása • kapcsoló értékének beállítása (**megkérdezi a tesztelőt**) • szimuláció indítása – hálózat kiértékelés indítása (2x) ∗ kapcsoló kiértékelése (állapotának kijelzése) ∗ inverter kiértékelése · bemenetén lévő érték lekérése · kimenetére kötött érték kiszámolása és kiadása ∗ led kiértékelése (világít/nem világít kijelzése) – áramkör változásának vizsgálata – két lépés alatt stacionárius állapot[1], szimuláció vége

Use-case neve	**2 Kapcsoló, Vagy kapu és Led**
Rövid leírás	Ez a usecase egy olyan áramkör tesztelését mutatja be, amely egy vagy kapura kötött két kapcsolóból és a vagy kapu kimenetére kötött ledből áll.
Aktorok	Tesztelő

[1] amennyiben a kapcsoló logikai igazra van állítva, akkor egy lépés is elég, de két lépés biztosan, így ezt ábrázoljuk diagramon

Forgatókönyv	• Áramkör és komponensek létrehozása • egyik kapcsoló értékének beállítása (**megkérdezi a tesztelőt**) • másik kapcsoló értékének beállítása (**megkérdezi a tesztelőt**) • szimuláció indítása – hálózat kiértékelés indítása (2x) ∗ egyik kapcsoló kiértékelése (állapotának kijelzése) ∗ másik kapcsoló kiértékelése (állapotának kijelzése) ∗ VAGY kapu kiértékelése · bemenetén lévő értékek lekérése · kimenetére kötött érték kiszámolása és kiadása ∗ led kiértékelése (világít/nem világít kijelzése) – áramkör változásának vizsgálata – második lépés után stacionárius állapot[2], szimuláció vége

[2]amennyiben mindkét kapcsoló 0-ás állapotban van, egy lépés alatt stabil lesz a hálózat, hiszen a VAGY kapu végig hamis állapotot ad ki, itt és a szekvencia diagramon úgy vesszük, mintha legalább az egyik kapcsoló 1-esbe lenne állítva.

Use-case neve	**Inverter visszakötve és Led**
Rövid leírás	Ez a usecase egy olyan áramkör tesztelését mutatja be, amely egy inverterből, amelynek kimenete egy ledbe illetve saját bemenetére van kötve. Oszcillálni fog, ezért a szimuláció rövid időn belül leáll.
Aktorok	Tesztelő
Forgatókönyv	• Áramkör és komponensek létrehozása • szimuláció indítása – hálózat kiértékelés indítása (3x) ∗ inverter kiértékelése · bemenetén lévő értékek lekérése · kimenetére kötött érték kiszámolása és kiadása ∗ csomópont kiértékelése · bemenetén lévő érték lekérése · kimeneteire az érték kiadása ∗ led kiértékelése (világít/nem világít kijelzése) – áramkör változásának vizsgálata – harmadik lépés után sincs stacionárius állapot, szimuláció vége

Use-case neve	**Kapcsoló, Vagy kapu visszakötve és Led**
Rövid leírás	Ez a usecase egy olyan áramkör tesztelését mutatja be, amely egy kapcsolóból, egy VAGY kapuból, melynek egyik bemenetére a kapcsoló, másik bemenetére a saját kimenete van kötve és egy ledből, melyre szintén a VAGY kapu kimenetét kötöttük. Ez egy olyan visszakötéses hálózat, mely stabil állapotban van.
Aktorok	Tesztelő

Forgatókönyv	• Áramkör és komponensek létrehozása • szimuláció indítása – hálózat kiértékelés indítása (2x) ∗ kapcsoló kiértékelése (állapotának kijelzése) ∗ VAGY kapu kiértékelése · bemenetén lévő értékek lekérése · kimenetére kötött érték kiszámolása és kiadása ∗ csomópont kiértékelése · bemenetén lévő érték lekérése · kimeneteire az érték kiadása ∗ led kiértékelése (világít/nem világít kijelzése) – áramkör változásának vizsgálata – második lépés után stacionárius állapot[3], szimuláció vége

5.3. Architektúra

5.4. A szkeleton kezelői felületének terve, dialógusok

Az általunk elkészített szkeleton egy program váz melynek felülete egy egyszerű konzolos megjelenítési felület, amely alkalmas arra, hogy a use case-k által leírt teszteseteket bemutassuk. Az egyes tesztesetek a neki megfelelő use case sorszámával van elnevezve, így program indítás után egy szám bevitelét követően a kiválasztott teszteset lefut. A teszteset futása közben kiír minden objektumot amin metódust hív, illetve kiírja a metódus nevét a paraméterekkel együtt, majd a visszatérési értéket. Ez azért lehetséges, mert a szkeleton már tartalmazza az elkészítendő szoftver összes fontos osztályát és metódusát, azonban az üzleti logikát még nem. Így könnyen eldönthető, hogy a use case-nek megfelelően viselkedik a program és továbbiakban képes lesz-e megfelelően működni. A tesztelési folyamat során döntési helyzet léphet fel. Ilyenkor a program felteszi a kérdést, majd a kapott válasz alapján folytatja a további futást. Ezzel csökkentjük a tesztesetek számát, anélkül, hogy bizonyos esetek kimaradnának a tesztelés alól. Futás közben megjegyzés formájában a program tájékoztat néhány elem belső állapotáról (például kapcsoló értéke) vagy bizonyos fontosabb lépésekről (például inicializálás). Az elvárás, hogy a szkeleton a szekvenciadiagramok által leírt működést mutassa. A program egyszerű és könnyen összehasonlítható formában írja ki a működését, amelyet könnyen összevethetjük a szekvencia diagrammokkal.

Egy metódushívás és visszatérés esetén kiírt adatok a következők:

- Metódushívás esetén a CALL szót, konstruktorhívás esetén CREATE szót, míg visszatéréskor a RETURN szót
- Objektum neve
- A metódus neve és a metódus paramétereinek értékét
- Visszatérés esetén a visszatérési értéket

Egy döntési helyzetben a kiírt adatok a következő:

- QUESTION szó

[3] ha a kapcsoló 0-ás állapotban van, akkor egy lépés alatt bekövetkezik, de érdekesebb szituáció, amikor 1-es állapotban van, ezt ábrázoljuk diagramon

- objektum neve
- Egy rövid magyarázó szöveg
- Szögletes zárójelben a lehetséges válaszok

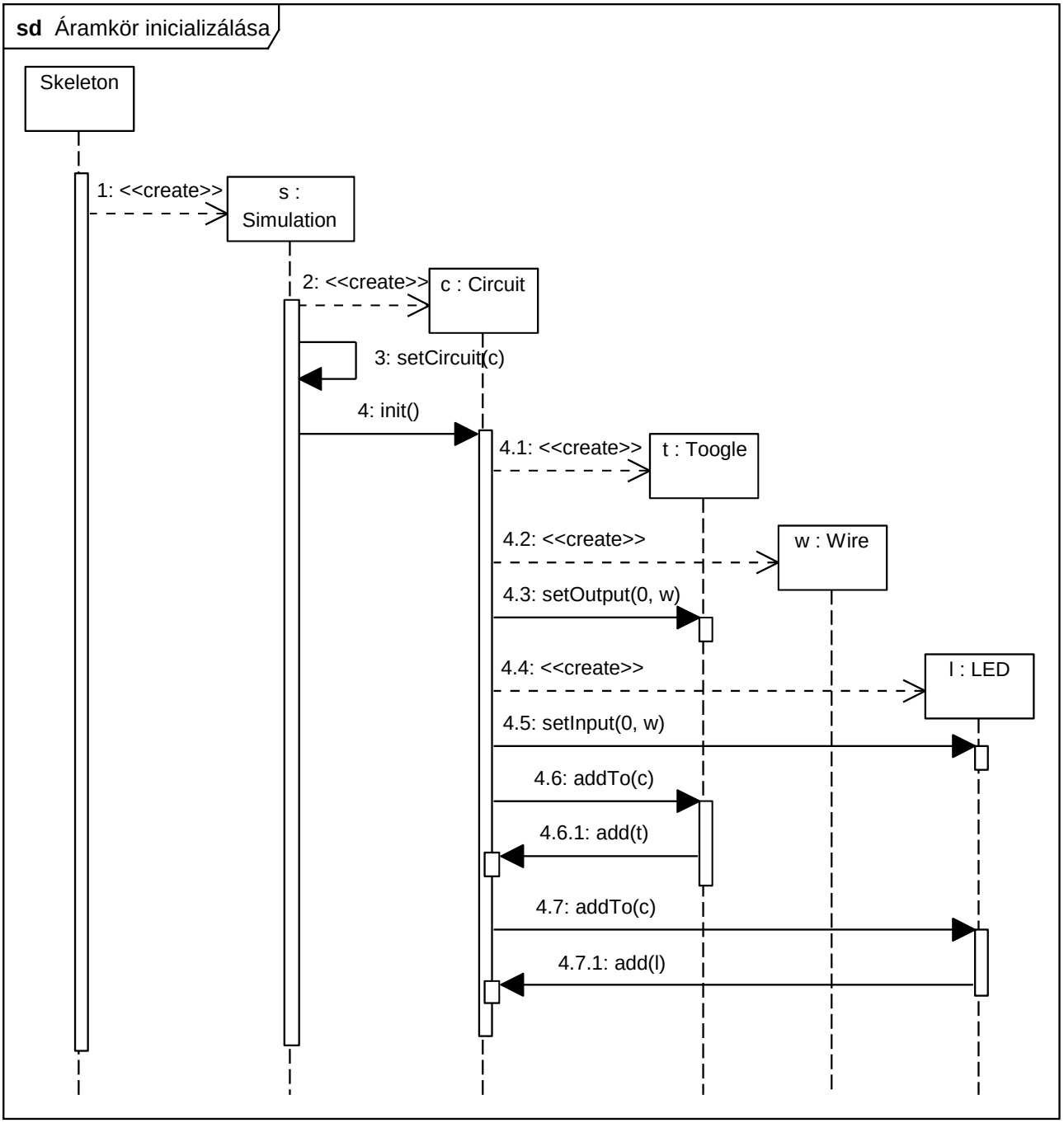
Formátumra példa:

```
CALL simulation.start()
  CALL circuit.doEvaluationCycle()
    CALL toggle.evaluate()
      QUESTION toggle állapot? [0/1]
1
      CALL toggle_to_inv.setValue(Value.TRUE)
      RETURN
    RETURN
    CALL inv.evaluate()
      CALL toggle_to_inv.getValue()
        QUESTION toggle_to_inv vezetéken lévő érték? [0/1]
1
      RETURN Value.TRUE
      CALL inv_to_led.setValue(Value.FALSE)
      RETURN
    RETURN
    CALL led.evaluate()
      CALL inv_to_led.getValue()
        QUESTION inv_to_led vezetéken lévő érték? [0/1]
0
      RETURN Value.FALSE
      # nem világít
    RETURN
  RETURN
  CALL circuit.doEvaluationCycle()
    CALL toggle.evaluate()
      QUESTION toggle állapot? [0/1]
1
      CALL toggle_to_inv.setValue(Value.TRUE)
      RETURN
    RETURN
    CALL inv.evaluate()
      CALL toggle_to_inv.getValue()
        QUESTION toggle_to_inv vezetéken lévő érték? [0/1]
1
      RETURN Value.TRUE
      CALL inv_to_led.setValue(Value.FALSE)
      RETURN
    RETURN
    CALL led.evaluate()
      CALL inv_to_led.getValue()
        QUESTION inv_to_led vezetéken lévő érték? [0/1]
0
      RETURN Value.FALSE
```

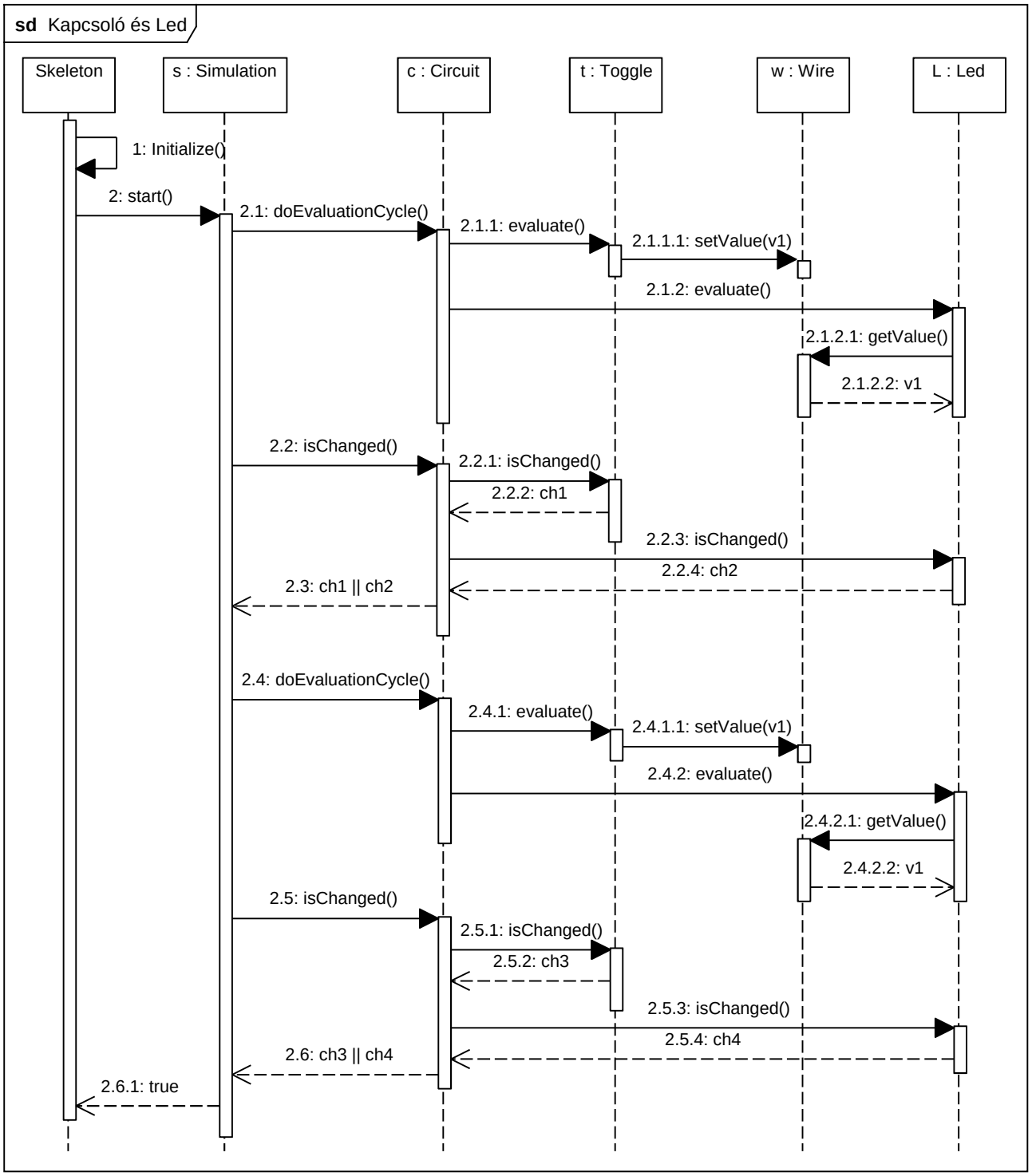
```
      # nem világít
    RETURN
  RETURN
  CALL circuit.isChanged()
    CALL toggle.isChanged()
      QUESTION toggle változott? [0/1]
0
    RETURN false
    CALL inv.isChanged()
      QUESTION inv változott? [0/1]
0
    RETURN false
    CALL led.isChanged()
      QUESTION led változott? [0/1]
0
    RETURN false
  RETURN false
RETURN true
```

5.5. Szekvencia diagramok a belső működésre

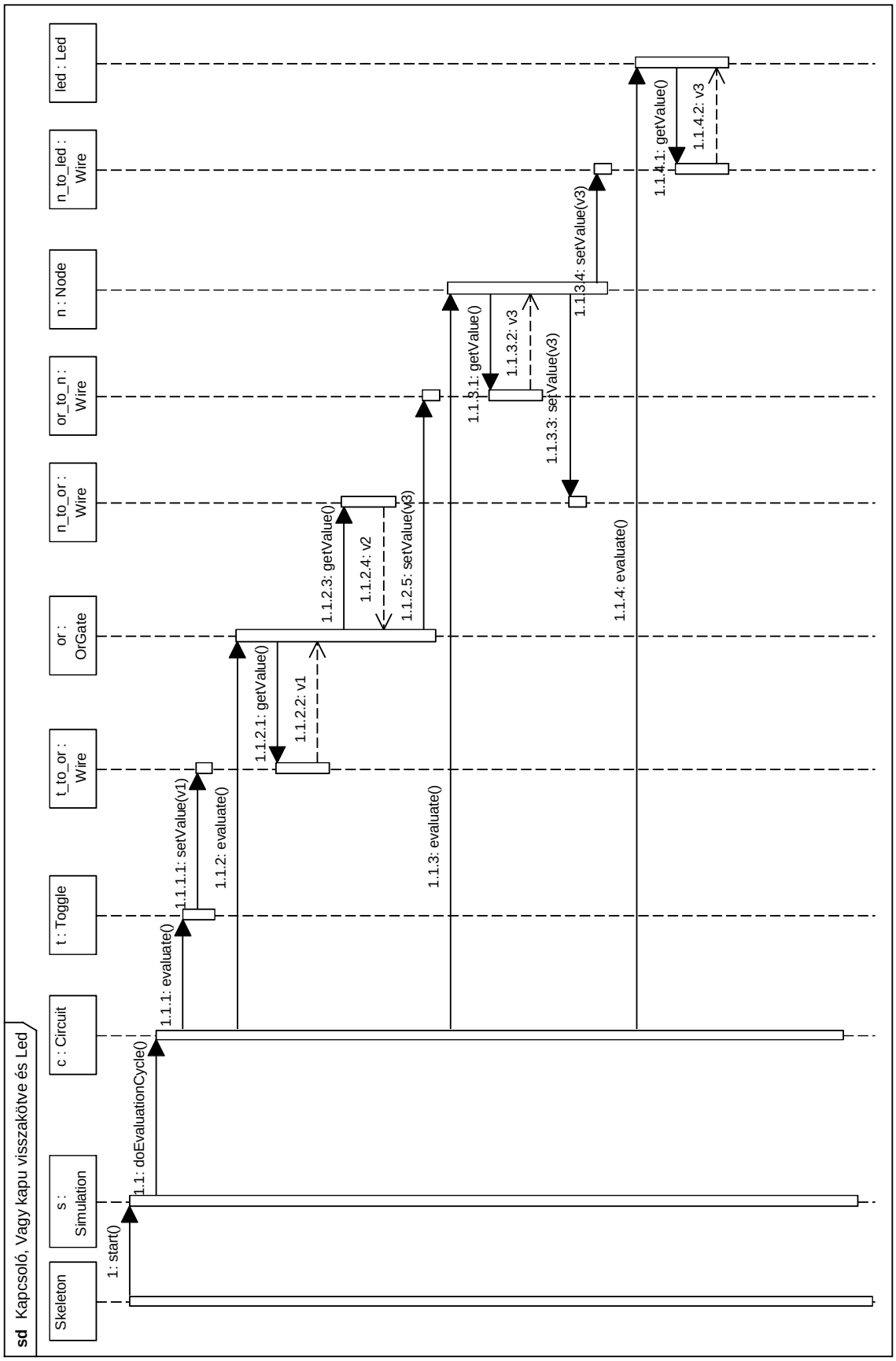
[A szkeletonban implementált szekvenciadiagramok. Tipikusan egy use-case egy diagram. Ezek megegyezhetnek a korábban specifikált diagramokkal, de az egyes életvonalakat (lifeline) egyértelműen a szkeletonban példányosított objektumokhoz kell tudni kötni. Azt kell megjeleníteni, hogy a szkeletonban létrehozott objektumok egymással hogyan fognak kommunikálni.]



5.3. ábra. Áramkör inicializálása



5.4. ábra. Kapcsoló és Led


sd Kapcsoló, Vagy kapu visszakötve és Led
Skeleton
s : Simulation
c : Circuit
t : Toggle
t_to_or : Wire
or : OrGate
n_to_or : Wire
or_to_n : Wire
n : Node
n_to_led : Wire
led : Led
1: start()
1.1: doEvaluationCycle()
1.1.1: evaluate()
1.1.1.1: setValue(v1)
1.1.2: evaluate()
1.1.2.1: getValue()
1.1.2.2: v1
1.1.2.3: getValue()
1.1.2.4: v2
1.1.2.5: setValue(v3)
1.1.3: evaluate()
1.1.3.1: getValue()
1.1.3.2: v3
1.1.3.3: setValue(v3)
1.1.3.4: setValue(v3)
1.1.4: evaluate()
1.1.4.1: getValue()
1.1.4.2: v3

5.5. ábra. 5-ös

5.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2010.03.12. 14:00	1,5 óra	**Kriván B.**	Javasolt módosítások elvégzése az előző fejezetben, rövid errate készítése jelen fejezet elé.
2010.03.13. 00:00	2 óra	**Péter T.**	Use-casek leírása szöveges formátumban
2010.03.13. 09:30	30 perc	**Kriván B.**	Use-case diagram megrajzolása
2010.03.13. 10:00	2 óra	**Kriván B.**	Use-casek leírásának LaTeX formátumra való alakítása, apróbb finomítások
...	...	**...**	...