2. Követelmény, projekt, funkcionalitás

54 – *Override*

Konzulens:
dr. László Zoltán

Csapattagok:

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

2011. február 19.

Tartalomjegyzék

Ábrák jegyzéke

2. Követelmény, projekt, funkcionalitás

[TODO ide a követelmény leírása??? vagy hagyjuk a fenébe???]

2.1. Követelmény definíció

2.1.1. A program célja, alapvető feladata

Az általunk kifejlesztett program célja egy előre megadott digitális áramkör szimulációja és annak megjelenítése, grafikus mindenki számára könnyen kezelhető, átlátható formában. Az alkalmazás az áramköri elemekből felépített digitális hálózat működését szemlélteti úgy, hogy felhasználói interakciók során a rendszer bemenetei átkonfigurálhatóak.

2.1.2. A fejlesztőkörnyezet

A fejlesztéshez NetBeans 6.9.1 szoftvert választottuk. Az UML diagramok elkészítéséhez a Visual Paradigm for UML nevű alkalmazást használjuk, mely képes az osztály-diagramból Java forráskódot generálni és vica versa. Fejlesztés során szem előtt tartjuk, hogy a program kompatibilis legyen az Oracle által gondozott Java 1.6-os verziójával. Természetesen a cél az, hogy a digitális áramkört modellező program a Hallgatói Számítógép Központban rendszeresített JDK és JRE alatt fordítható és futtatható legyen. A dokumentumokat LaTeX-hel készítjük el a Texmaker nevű alkalmazás segítségével, melyet PDF-be fordítunk le. A unit-tesztekre a JUnit csomagot fogjuk használni.

Mivel a fentebb felsoroltak közül mindegyik alkalmazás fut mind Windows, mind Linux operációs rendszeren, így az egész fejlesztés mindkét platformon megvalósítható.

2.1.3. A futtatáshoz szükséges környezet

Java Runtime Environment 1.6-os verziója, illetve az a számítógép, mely ezt futtatni képes. A modellező alkalmazás használatához billentyűzet, grafikus képernyő és egér szükséges.

2.1.4. A felhasználói felület

A program végső változata grafikus felhasználói felülettel rendelkezik. A programot a felhasználó az egér és a billentyűzet segítségével vezérelheti.

2.1.5. Minőségi tényezők

Teljesítmény: A cél az, hogy a digitális rendszermodellező szoftver használható legyen a fentebb meghatározott minimális rendszeren. A grafikus felületnél törekedni fogunk a folyamatos szimuláció megjelenítése.
Újrafelhasználhatóság: A cél az, hogy a grafikus felhasználói felületet a program többi részétől teljesen különválasszuk, így lehetővé téve azt, hogy később a grafikus felület egyszerűen és gyorsan változtatható legyen.
Rugalmasság: A rugalmasságot a fejlesztőkörnyezet biztosítja, a modellező szoftvernek ugyanis minden olyan környezetben futtathatónak kell lennie, melyben létezik megfelelő Java futtatókörnyezet.
Felhasználhatóság: A használat különösebb tanítást nem igényel, alapfokú számítástechnikai tudással akár a felhasználói kézikönyv elolvasása nélkül is használható.

2.1.6. A software minősítése

A kifejlesztett software akkor megfelelő, ha minél pontosabban megegyezik a fentebb leírtakkal. Ezt ellenőrizni lehet a program futtatásával és kipróbálásával, illetve a forráskód és a modell összevetésével, valamint a funkcionális tesztek futtatásával.

2.1.7. A kibocsátás

A program kibocsátása először a forráskóddal együtt a konzulens felé fog történni.

2.2. Projekt terv

2.2.1. A fejlesztői csapat

Csapattag neve	feladatköre
Kriván Bálint	csapatvezető, kód, dokumentáció, szervezés
Jákli Gábor	kód, dokumentáció, ticket-koordinátor
Dévényi Attila	kód, dokumentáció, GUI-felelős
Apagyi Gábor	kód, dokumentáció
Péter Tamás Pál	kód, dokumentáció

2.2.2. Életciklus modell

A feladat először a program megtervezése, mely a dinamikus és objektum modelleket foglalja magába. Ha ez készen van, elkezdhető a skeleton implementálása. Ez a lépés már teljesen meghatározott, nem merülhet fel semmilyen komplikáció, ha a modellek megfelelőek voltak. A következő feladat a prototípus elkészítése. A programnak ebben az állapotban könnyen tesztelhetőnek kell lennie, hogy a programozási és funkcionalitási logikai hibák könnyen felismerhetők legyenek. Ha már a prototípus is megfelelő, akkor kezdődhet a grafikus felület megvalósítása. Itt is fontosa tesztelés és a kiértékelés, mert a jó megjelenés sokat számít a modellező használhatóságában. Ha ennek kifejlesztése is sikeres, készen van a program első teljes változata. A kötelező feladat csak eddig tart. Ezt a változatot kell leadni a dokumentációval és a forráskóddal együtt.

2.2.3. Szervezési struktúra

A csapat öt emberből áll. A feladat szempontjából a tudásunk nem azonos, mindenki más-más területet érez a magáénak, illetve a feladat eltérő részeinek megoldásához van nagyobb kedvünk. Azt a felépítést választottuk, hogy mindenki az érdeklődésének és tudásának legmegfelelőbb részt kapja az egész feladatból. A feladatok szétosztását találkozókon, illetve az alább meghatározott kommunikációs csatornákon egyeztetjük, ahol az egyéni kívánságok mellett ügyelünk arra, hogy minden feladat kiosztásra kerüljön, valamint a csapattagok az egész feladat megoldásából nagyjából egyenlő mértékben vegyék ki a részüket. A találkozók keretében, mivel a szétosztott feladatok nagy mértékben függnek egymástól, javaslatokat teszünk egymásnak a feladat megoldásának körülményeit és a határidőt illetően.

A forráskódot és minden a fejlesztés során elkészülő dokumentációt, illetve a projekthez tartozó egyéb fájlokat megegyezés alapján egy Git központi tárolóban tároljuk, melyhez a Codaset (`http://codaset.com`) nevű ingyenes szolgáltatását használjuk és erről mindenki egy saját klónt készít.

A kiosztott feladatokat a tulajdonosuk elvégzi a megbeszélt határidőig, de ha ez megváltozott funkcionalitást takar, akkor az adott csapattag köteles a megfelelő teszteseteket megírni, és azok sikeres lefutásáról meggyőződni. Abban az esetben, ha az alkalmazás nem fordul le, vagy valamelyik teszteset nem fut le sikeresen, az adott commit visszaállításra kerülhet annak kijavításáig, melyet a ticket-rendszerben jelezzük a másik felé.

Hogy a fejlesztés minél hatékonyabb és zökkenőmentesebb legyen, a következő eszközöket, technológiákat alkalmazzuk:

E-mail Az egymás számára fontos anyagokat, melyeket a találkozókon előzetesen megbeszéltünk, levélben küldjük el.

Msn Felvettük egymást a Microsoft Messenger-be, hogy szükség esetén egymástól is segítséget tudjunk kérni kisebb technikai problémák megoldásában. Természetesen ezek a feladat lényegét, a projektről hozott döntéseket nem érinthetik, de kivételes helyzetben akár az Interneten is tarthatunk találkozót.

Git tároló A feladatok megoldása közben keletkezett anyagokat egy – kizárólag a csapat tagjai által hozzáférhető – helyen tároljuk (lásd fentebb). Így mindig elérhető a fejlesztések legfrissebb változata.

Ticket-rendszer A fejlesztés során felmerülő problémákat, kérdéseket ticket formájában megírjuk egymásnak, amit később a kijelölt felelős személy megold, ha szükséges, akkor együtt konzultálunk a megoldás módjáról, menetéről.

2.2.4. Fejlesztési ütemterv

A program fejlesztésének három fő lépcsőfoka van. Ezek a következők:
Skeleton: A cél az, hogy mind a dinamikus, mind az objektum modell jól legyen kitalálva. Ha ezek elkészültek, akkor a fejlesztés szempontjából sikeresen leraktuk az alapokat.
Prototípus: Ez már szinte a teljes változat, csak a grafikus felület elemei hiányoznak. Ez a változat tökéletesen megfelelő arra, hogy az objektumok, rutinok, függvények szemantikai helyességét vizsgáljuk.
Grafikus változat: A program teljes változata. Tulajdonképpen a prototípus a grafikus felülettel kiegészítve, esetleg kismértékben továbbfejlesztve.

2.2.5. Határidők

febr. 11.	Csapat regisztráció
febr. 21.	Követelmény, projekt, funkcionalitás
febr. 28.	Analízis modell kidolgozása 1.
márc. 7.	Analízis modell kidolgozása 2.
márc. 14.	Szkeleton tervezése
márc. 21.	Szkeleton
márc. 28.	Prototípus koncepciója
ápr. 4.	Részletes tervek
ápr. 18.	Prototípus
ápr. 26.	Grafikus felület specifikációja
máj. 9.	Grafikus változat
máj. 13.	Összefoglalás

2.3. Feladatleírás

Az általunk készített alkalmazás segítségével a felhasználó egy előre elkészített áramköri listából kiválasztott digitális áramkör szimulációját végezheti el grafikus megjelenítéssel. A program az alábbi alkatrészeket támogatja áramköri elemként: ÉS kapu, VAGY kapu, INVERTER, Kijelző, Jelgenerátor, Kapcsoló. Ezek mindegyike egy vagy több ki- és/vagy bemenettel rendelkezik.

A komponensek (alkatrészek) részletezése:

- Az ÉS kapu kettő vagy több bemenettel és egy kimenettel rendelkezik. A kimenetre a bemenetre kötött jelek logikai ÉS kapcsolata jelenik meg.
- A VAGY kapu kettő vagy több bemenettel és egy kimenettel rendelkezik. A kimenetre a bemenetre kötött jelek logikai VAGY kapcsolata jelenik meg.
- Az INVERTER egyetlen kimenetén az egyetlen bemenetére kötött jel logikai negáltja jelenik meg.

Ezen elemeket általános elemként kezeljük. Működésük általánosan annyiból áll, hogy a bemenetükre érkező logikai értéket, a megfelelő logikai művelet elvégzése után a kimenetükön továbbítják.

- A Kijelző komponenssel a felhasználó a bementre kötött jelet vizuális formában tudja megjeleníteni. A megjelenítő színt piros, kék, sárga közül választhat.

A Kijelző alkotja a komponensek második csoportját, mely a későbbiekben akár bővülhet más elemekkel (pl. hétszegmenses kijelző). Az ide tartozó elemek feladata a logikai értékek vizualizálása.

- Jelgenerátor segítségével egy bitsorozatot tárolhatunk el, amelyet a szimuláció során az egyetlen kimenetén ciklikusan kiad.
- A Kapcsolónak egy kimenete van, melynek értéke a kapcsoló állásától függ. „Be" állásban igaz, „Ki" állásban hamis értékű.

A harmadik csoport elemei jelforrások, amiket a felhasználó érdemben tud módosítani, vagyis a változtatások kihatnak az áramkör viselkedésére.

Az összes alkatrészre igaz, hogy nem lehet olyan bemenetük, amelyek szabadok, vagyis sehova sincsenek bekötve, ellenkező esetben a szimulációt nem lehet elindítani és ezt jelzi a felhasználó felé.

A felhasználó a fentebb említett alkatrészekből összeállított digitális áramkör szimulációját végezheti. Az alkatrészek és azok egymáshoz kötéseik (összeköttetések) grafikus formában kerül megjelenítésre.

A szimuláció során bármelyik komponens pillanatnyi értékeit a felhasználó lekérdezheti az alkatrészre való kattintással, ezzel egyidejűleg a szimulációt szünetelteti. Az áramkör vizsgálata közben a Kapcsolók értékeit szabadon változtathatja, melyek hatása valós időben megjelenik. Szimuláció elkezdésekor az összes áramköri elem kimenete hamis értéket vesz fel. Ha a vizsgálandó áramkör bizonyos idő alatt nem áll be stacionárius állapotba változatlan bemenetek mellett, akkor ez jelzésre kerül a felhasználó számára és a szimuláció automatikusan leáll. A szimuláció bármikor megszakítható majd újraindítható, illetve átválthat egy másik digitális áramkör vizsgálatára (az előre elkészített digitális áramkörök közül választva), amennyiben a jelenlegi áramkört nem kívánja tovább használni.

A szimuláció sebessége a felhasználó által konfigurálható, ezáltal a Jelgenerátor kimenetein kiadott jelek ciklusideje változtatható.

A Kapcsolók, illetve a Jelgenerátorok gyors és egyszerű alap állapotba helyezése érdekében lehet törölni minden addig elvégzett beállítást egyetlen paranccsal, majd elölről kezdeni az egyes elemek konfigurálását. Lehetőség lesz továbbá a jelforrások beállításainak elmentésére illetve későbbi visszatöltésére is. A konfigurációs fájl sikeres betöltődéséhez teljesülnie kell annak a feltételnek, hogy abban az összes meghatározott elem szerepeljen az áramkörben, amire használni szeretnénk a beállításokat. Amennyiben ez nem áll fent, akkor a nem specifikált jelforrások alapállapotban lesznek. Előfordulhat még, hogy a beállításokat tartalmazó fájlban olyan elemek (is) szerepelnek, amelyek az áramkörünkben nem, ekkor a program hibaüzenetet jelenít meg, és megszakítja a betöltés folyamatát.

2.4. Szótár

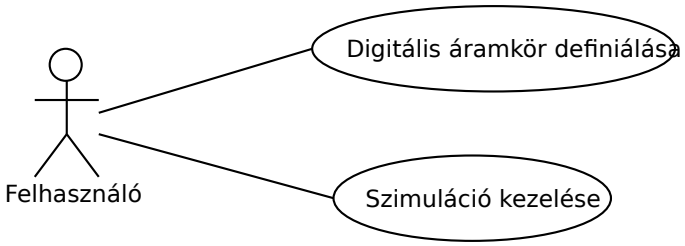
Előre elkészített lista	Áramköröket tartalmazó előre elkészített gyűjtemény.
Áramkör	A komponensek egymáshoz kötéséből létrejövő rendszer.
Komponensek egymáshoz kötése	Egy olyan folyamat, amely során 2 komponenst oly módon kapcsolunk össze, hogy az egyik komponens bemenetére a másik komponens kimenetének logikai jelét kapja meg.
Logikai jel	Az áramkörben levő komponensek által továbbított információ a többi komponens számára, mely igaz, és hamis értéket veheti fel.

Szimuláció	Olyan folyamat, mely során minden alkatrész kimenetének logikai jel értékét kiszámoljuk a bemenetére érkező logikai jelekből.
Grafikus	Az áramkör felhasználó számára felfogható, érzékelhető megjelenítése.
Alkatrész	komponens szinonimája
ÉS kapu	Logikai jeleken végez olyan műveletet mely során ha a bemenetére érkező logikai jelek közt található hamis érték, akkor kimenetén hamis értéket továbbít.
VAGY kapu	Logikai jeleken végez olyan műveletet mely során ha a bemenetére érkező logikai jelek közt található igaz érték, akkor kimenetén igaz értéket továbbít.
INVERTER	Bemenetére érkező logikai jel negáltját továbbítja a kimenetén.
Kijelző	Bementére érkező logikai jelet a felhasználó számára érzékelhető módon szemlélteti.
Jelgenerátor	Olyan egység, mely előre megadott bitsorozatot ad ki ciklikusan a kimenetén.
Bitsorozat	Logikai jelekből létrejövő olyan sorozat, melynél az igaz értéket '1' szimbólum reprezentál, míg a hamis értéket '0'.
Kapcsoló	Olyan egység, mely felhasználói interakció hatására kimenetén igaz, vagy hamis értéket továbbít.
Felhasználói interakció	Olyan folyamat, mely során a felhasználó egyes komponensek működését tudja befolyásolni
Kimenet	A komponens olyan része, melyen keresztül logikai jeleket tud továbbítani más komponenseknek.
Jel továbbítás	A logikai jel egyik komponenstől másik komponensig való áramlása.
Bemenet	A komponensek olyan része, melyen keresztül fel tudják használni maguk szimulációjához szükséges logikai jeleket.
Komponens szimulációja	Egy alkatrészre vonatkoztatott szimuláció.
Logikai negált	Logikai jel érték változást jelent, oly módon, hogy igaz értékből hamis lesz, és hamis értékből igaz.
Igaz érték	Logikai jel egysége.
Hamis érték	Logika jel egysége.
Komponens	Egy komponens 3 típusú lehet. Sima komponensek logikai jeleken végeznek műveleteket, és azt továbbítják további komponens/komponensek számára. Kijelző típusú források logikai jeleket szemléltetik a felhasználó számára érzékelhető módon. Jelforrás típusú komponensek közvetlen bemenet nélkül generálnak kimenetet és továbbítják további komponens/komponensek számára azt.
Közvetlen bemenet	Egy komponensnek közvetlen bemenete van, ha a bemenetére a logikai jeleket más komponens továbbítja számára
Jelforrás	Jelgenerátor szinonimája
Valós időben	A felhasználó számára észrevehetetlen idő alatt
Stacionárius	A rendszer egy stabil állapota, melyben olyan értékek jelennek meg a komponensek kimenetein, amelyek hatására (visszacsatolás esetén sem) változiknak a rendszer komponenseinek kimeneti értékei (Jelgenerátor esetén nincs stacionárius állapot)

Szimuláció sebessége	Az a sebesség, ami a szimuláció egyes állapotai közötti váltás sebességét mutatja
Ciklusidő	A szimuláció két állapotváltása között eltelt idő
Vizsgálat megszakítás	*[TODO]*
Állapot	Az áramkör komponenseinek aktuális tulajdonságainak (kimeneti/bemeneti értékek) összessége
Alap állapot	Az a kiindulási állapot, mely az áramkör betöltése után keletkezik

2.5. Essential use-case-ek

2.5.1. Use-case diagram



2.1. ábra. Essential use-case diagram

2.5.2. Use-case leírások

Use-case neve	**Szimuláció kezelés**
Rövid leírás	A felhasználó új szimulációt indít vagy leállít
Aktorok	Felhasználó
Forgatókönyv	*[TODO]*

Use-case neve	**Digitális áramkör definiálása**
Rövid leírás	A felhasználó digitális áramkört betölti, szerkeszti
Aktorok	Felhasználó
Forgatókönyv	*[TODO]*

2.6. Napló

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2011.02.15. 21:30	2 óra	**Kriván B.** **Dévényi A.** **Jákli G.**	Értekezlet. Döntések: Megegyeztünk a feladat értelmezését illetően. Milyen kérdéseket teszünk fel a konzulensnek az első konzultáción?

Kezdet	Időtartam	Résztvevők	Leírás
2011.02.16. 09:00	2 óra	**Kriván B.** **Dévényi A.** **Apagyi G.** **Péter T.**	Értekezlet. Döntések: • Fejlesztői környezetben megállapodtunk (2.1.2) • Projekt szervezési struktúráját rögzítettük (2.2.3) • A felmerülő algoritmusokról is konzultáltunk.
2011.02.16. 15:00	1 óra	**Jákli G.**	Tevékenység: Projekt terv bővítése, formázása, javítása (2.2)
2011.02.17. 16:00	1 óra	**Jákli G.** **Kriván B.** **Dévényi A.**	Tevékenység: a 2.1 és a 2.2 alfejezet átdolgozása
2011.02.17. 19:15	45 perc	**Jákli G.** **Kriván B.** **Dévényi A.**	Értekezlet. Döntések: A 2.3 alfejezet főbb gondolatait megfogalmaztuk, és meghatároztuk, hogy mik legyenek a mindenképpen lejegyezendő dolgok.
2011.02.17. 20:00	50 perc	**Jákli G.**	Tevékenység: A megbeszéltek alapján elkezdte a 2.3 alfejezet megírását.