**1. Követelmények**

54 – Override

Konzulens:

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 |

2011.02.16.

# Követelmények leírása

## A követelmények leírása

### Digitális rendszer modellezése

Egy digitális rendszer (áramkör) alapvető építőelemekből áll. Az építőelemek be- és/vagy kimenetekkel rendelkeznek. A kimenetek értéke a bemenetek értéke alapján alakul ki. Egy kimenetre tetszőleges számú bemenet csatlakoztatható, amely hatására az adott bemenetek felveszik a kimenet által adott logikai (igaz vagy hamis) szintet. Több kimenet összekapcsolása nem megengedett. Üresen hagyott (semmihez sem kapcsolódó) bemenet nem megengedett.

Az alapvető építőelemek:

* INVERTER - az inverter egy bemenettel és egy kimenettel rendelkezik, a kimenet értéke a bemenet logikai negáltja.
* ÉS kapu - az és kapu két vagy több bemenettel és egy kimenettel rendelkezik. A kimenet értéke a bemenetek logikai és kapcsolata.
* VAGY - a vagy kapu két vagy több bemenettel és egy kimenettel rendelkezik. A kimenet értéke a bemenetek logikai vagy kapcsolata.
* Kapcsoló - a kapcsoló jelforrás, egy kimenete van. "Be" állásban a kimenete igaz, "ki" állásban pedig hamis.
* Jelgenerátor - a jelgenerátor jelforrás, egy kimenettel. A kimenet előre definiált jelsorozatot ismétel ciklikusan.
* Kijelző - a kijelző a jel értékét megjelenítő elem, egy bemenete van. Bemenetének igaz estén világít, hamis esetében pedig nem világít.

Későbbiekben új építőelemek megjelenése is várható.

Az építőelemekből tetszőleges logikai hálózat felépíthető, amely a jelforrások, mint bemeneti értékek hatására működést produkál, amely a kijelzőkön keresztül megfigyelhető.

# Követelmény, projekt funkcionalitás

## Követelmény definíció

### 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.

### A fejlesztőkörnyezet

A fejlesztéshez NetBeans 6.9.1 szoftvert választottuk. Modellezéshez a Visual Paradigm for UML nevű alkalmazást használjuk, mely képes az osztály-diagramból Java forráskódot generálni és vica versa. Fejlesztés szem előtt tartjuk, hogy a program kompatibilis legyen az Oracle által gondozott Java 1.6-os verziójával. Természetesen a cél az, hogy a modellező program a Hallgatói Számítógép Központban rendszeresített JDK és JRE alatt fordítható és futtatható legyen. A dokumentumokat Microsoft Office és Open Office programokkal, DOCX formátumban készítjük. A unit-tesztekre a JUnit csomagot fogjuk használni.

### 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.

### A felhasználói felület

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

### 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.

### 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.

### A kibocsátás

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

## Projekt terv

### Fejlesztői csapat

|  |  |
| --- | --- |
| **Csapattag neve** | **Feladatköre** |
| Kriván Bálint | csapatvezető, kód, dokumentáció, szervezés |
| Jákli Gábor | kód, dokumentáció, ticket koordinálás |
| Dévényi Attila | kód, dokumentáció |
| Apagyi Gábor | kód, dokumentáció |
| Péter Tamás Pál | kód, dokumentáció |

### 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.

### 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 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**: 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.

### 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.

### 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 |

## Feladatleírás

Az általunk készített alkalmazás segítségével a felhasználó egy 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.
* A Kijelző komponenssel a felhasználó a bementre kötött jelet vizuális formában tudja megjeleníteni. A megjelenítő színt piros, kék, sárga közül választhat.
* 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.

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 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 vizsgálat bármikor megszakítható majd újraindítható, illetve a felkínált digitális áramkörök közül választhat másikat, hogy azt szimulálja.

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

## Szótár

[A szótár a részletes leírás alapján készítendő dokumentum. Egy szótári bejegyzés definiálásához csak más szótári bejegyzések és köznapi – a feladattól független – fogalmak használhatók fel. A szótár mérete kb. 1-2 oldal legyen.]

## Essential use-case-ek

### Use-case diagram



1. ábra

### Use-case leírások

**Use Case:** Szimuláció kezelés

**Actor:** Felhasználó

**Leírás:** A felhasználó új szimulációt indít vagy leállít

**Use Case:** Digitális áramkör definiálása

**Actor:** Felhasználó

**Leírás:** A felhasználó digitális áramkört betölti, szerkeszti

## 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? |
| 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. | Projekt terv bővítése, formázása, javítása (2.2) |
| 2011.02.17. 16:00 | 1 óra | Jákli G.  Kriván B.  Dévényi A. | 2.1-es és 2.2 alfejezet átdolgozása |
| 2011.02.17. 19:15 | 45 perc | Jákli G.  Kriván B.  Dévényi A. | Értekezlet. Döntések:   * A 2.3-as alfejezet főbb gondolatait megfogalmaztuk, és meghatároztuk, hogy mik legyenek a mindenképpen lejegyezendő dolgok. |
| 2011.02.17. 20:00 | 50 perc | Jákli G. | A megbeszéltek alapján elkezdte a 2.3-as alfejezet megírását. |