Általános információk

A diplomaterv szerkezete:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletesés pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő Diplomaterv sablon dokumentum tartalma. Ügyeljen a konzulens nevét és a beadás évét jelölő szövegdobozokra, mert azokra külön ki kell adni a frissítést. A mezők tartalma a sablonban a dokumentum adatlapja alapján automatikusan kerül kitöltésre.

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2,5 cm, baloldalon 1 cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel.

Minden oldalon – az első négy szerkezeti elem kivételével – szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2 táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le.

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerző és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben. Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

* a szakdolgozat készítő/diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás Karunkon, ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
* mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozat készítést, ill. diplomatervezést kívánunk!

FeladatkiÍrás

A feladatkiírást a tanszék saját előírása szerint vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.

C:\Users\szarnyasg\Downloads\bme_logo_nagy.eps

**Budapesti Műszaki és Gazdaságtudományi Egyetem**

Villamosmérnöki és Informatikai Kar

Készítette

Lucz Tamás SomaKonzulens

2016

Tartalomjegyzék

[Kivonat 5](#_Toc452026591)

[1. Bevezetés 6](#_Toc452026592)

[1.1. Forráskódanalízis 6](#_Toc452026593)

[1.1.1. Motivációk 6](#_Toc452026594)

[1.2. Statikus analízis 6](#_Toc452026595)

[1.3. Dinamikus kódanalízis 6](#_Toc452026596)

[2. JavaScript-kódanalízis 7](#_Toc452026597)

[2.1. Kódinstrumentáció 7](#_Toc452026598)

[2.2. AST-bejárás 7](#_Toc452026599)

[2.3. 7](#_Toc452026600)

[3. Összegyűjtött eszközök 8](#_Toc452026601)

[3.1. Fejlesztést közvetlenül támogató eszközök 8](#_Toc452026602)

[3.2. Statikus analízis eszközök 8](#_Toc452026603)

[3.3. Dinamikus analízis eszközök 8](#_Toc452026604)

[4. Összefoglalás és további lehetőségek 9](#_Toc452026605)

[5. Bevezetés 10](#_Toc452026606)

[5.1. Formázási tudnivalók 10](#_Toc452026607)

[5.1.1. Címsorok 10](#_Toc452026608)

[5.1.2. Képek 10](#_Toc452026609)

[5.1.3. Táblázatok 10](#_Toc452026610)

[5.1.4. Kódrészletek 10](#_Toc452026611)

[5.1.5. Irodalomjegyzék 11](#_Toc452026612)

[5.1.6. Margók 11](#_Toc452026613)

[6. Utolsó simítások 12](#_Toc452026614)

[7. Összefoglalás 13](#_Toc452026615)

[Ábrák jegyzéke 14](#_Toc452026616)

[Táblázatok jegyzéke 15](#_Toc452026617)

[Irodalomjegyzék 16](#_Toc452026618)

[Függelék 17](#_Toc452026619)

# Kivonat

A szakdolgozat magyar nyelvű kivonata. Ez egy ½–1 oldalas magyar nyelvű összefoglaló, melynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

# Bevezetés

## Forráskódanalízis, motivációk

Szoftvereink kódját emberek írják. Az emberek természetes tulajdonsága, hogy hibákat követnek el, amik a megfelelő eszköztárak hiányában felfedezetlenek maradnak. Ezen fejlesztői hibák fokozott kockázatot jelenthetnek a készülő szoftverre, hiszen a logikailag esetlegesen helytelen működés mellett olyan biztonsági réseket teremthetnek, melyek kiaknázása a szoftver nemkívánatos viselkedését idézheti elő. Ez rosszindulatú támadóknak nyújt lehetőséget arra, hogy a szoftvert számukra kedvező, a fejlesztők számára kedvezőtlen módon, de mindenképpen a szándékolttól eltérő módon futtassák.

A forráskódanalízis módszertanának kidolgozása mögött elsődleges motivációként áll, hogy fejlesztői hibákat még a futtatási idejű tesztelés folyamatának megkezdése előtt, vagyis a fejlesztési folyamat közben – akár a kód írásának idejében, valós időben – fedezzünk fel, és figyelmeztessük a kód készítőjét.

Amennyiben a fentiekre lehetőségünk nyílik, szeretnénk az elkövetett hibákat minél teljesebben feltárni, ezáltal minimalizálni a szoftverbe kerülö biztonsági kockázatokat, „bug”-okat. Azonban a forráskódanalízis önmagában csak egy ún. „best effort” tevékenység, vagyis helyesség/teljesség nem feltétlenül követelménye: a formális verifikáció foglalkozik az egyes szoftvertulajdonságok matematikai bizonyításával.

Mivel a hibák hiányát kézi teszteléssel nehéz feltárni, automatizált munkafolyamatra van szükség. Ez a folyamat szükségszerűen determinisztikus: az egyes ugyanolyan paraméterekkel, azonos kódon készített eredmények megegyeznek.

A továbbiakban a forráskódanalízis fogalmat egy automatizált számítógépes eszköz által végzett analízisként használom.[[1]](#footnote-1)

## Statikus analízis

Statikus forráskódanalízisként a forrás során a forráskód által reprezentált szoftvert nem futtatjuk. A forráskódot, mint absztrakt entitást értelmezzük, és ennek során próbálunk megadott szabályok alapján következtetéseket levonni.

A forráskódot szinte minden esetben különféle matematikai eszközökkel (pl. fák, gráfok) vizsgáljuk. Ennek követelménye, hogy a forráskód által reprezentált programot matematikailag értelmezhető struktúrákba transzformáljuk. A transzformációnak egyértelműnek és helyesnek kell lennie, hiszen helytelen strukturális reprezentációval az analízis eredménye elméletileg sem lehet helyes.

A következőkben bemutatott három struktúra terjedt el széleskörűen statikus analízisek során.[[2]](#footnote-2)

### Absztrakt szintaxisfa (Abstract Syntax Tree, AST)

Az absztrakt szintaxisfa (a továbbiakban: AST) a forráskód absztrakt szintaktikai struktúrájának fa-alapú reprezentációja. A fa minden eleme szükségszerűen egy, a forráskódban megjelenő elemet jelöl.

A forráskód a program logikai szerkezetének szempontjából egértelműen megfeleltethető egy AST-nek, és viszont. Tehát a forráskód–AST-transzformáció, illetve az AST–forráskód-transzformáció a reprezentációk programlogikai szempontból egyértelmű egymásba alakítása.

A fa abban az értelemben absztrakt, hogy nem veszi figyelembe a forráskód minden egyes részletét: pl. blokkokat csoportosító kapcsos zárójelek a program logikai struktúrájában nem játszanak szerepet, így nem kell, hogy feltétlenül szerepeljenek a fában.[[3]](#footnote-3)

Az AST a program szintaktikai szempontból történő ellenőrzését teszi lehetővé, ezzel a statikus analízisben kiemelt szerepet játszik.

### Absztrakt szemantikus gráf (Abstract Semantic Graph, ASG)

Az absztrakt szemantikus gráf (a továbbiakban: ASG) a forráskód absztrakt szintaktikai struktúrájának az AST-nél egy magasabb absztrakciós szinten történő reprezentációja.

***Definíció [term]:*** *Azon szimbólumokat, melyeket konstansokból, változókból, vagy függvényekből állítunk elő,* ***termeknek*** *nevezzük.*

Az ASG a forráskódot egy kifejezésként ábrázolja, csúcsai a kifejezés résztermjei. Általában irányított körmentes gráf (DAG), ha kört tartalmaz, az pl. rekurziót jelenthet.

### Vezérlésfolyam-gráf (Control-Flow Graph, CFG)

A vezérlésfolyam-gráf (a továbbiakban: CFG) a forráskód absztrakt reprezentációja gráf formában. Tartalmazza a program összes lefutási útvonalát.

***Definíció [vezérlésfolyam-blokk]:*** *Azon kódrészletet, amely nem tartalmaz ugrást vagy elágazást,* ***vezérlésfolyam-blokknak*** *nevezzük.*

A gráf minden csúcsa egy vezérlésfolyam-blokk, a gráf irányított élei a blokkok közötti vezérlésfolyamot reprezentálják. Az ún. belépési blokk a gráfba belépő vezérlésfolyam belépési pontja, az ún. kilépési blokk pedig a gráfot elhagyó vezérlésfolyam helye.

CFG-k használata statikus analízisek terén igen elterjedt. Többek között elérhetőségi problémákra nyújt megoldást: pl. ha egy részgráf belépési pontjának nincs bemenő éle, a részgráf elérhetetlen kódrészletet reprezentál; ha egy kilépési blokk nem elérhető a belépési blokkból, az végtelen ciklust jelenthet.

## Dinamikus kódanalízis

Dinamikus analízis során a programot futtatjuk, és a futtatás során végbemenő viselkedést vizsgáljuk. Ennek követelménye, hogy a program futtatható (tehát fordított nyelvek esetén fordítható) legyen. A dinamikus analízisra tehát úgy tekinthetünk, mint egy statikus analízis után végzett „második lépés”, mellyel más típusú, futtatási idejű teszteredményeket kaphatunk.

Dinamikus tesztelésnél fontos szerepet játszik, hogy milyen bemeneti paraméterekkel, inputokkal futtatjuk a programot, hiszen más inputokkal más viselkedést produkálhat a vizsgált szoftver.

***Definíció [instrumentáció]:*** *Azt a tevékenységet, melynek során egy forráskódot vagy programot olyan formába alakítunk, hogy a számunkra fontos tulajdonságait vizsgálni tudjuk,* ***instrumentációnak*** *nevezzük.*

Rendkívül fontos továbbá a kód instrumentációja során keletkező mellékhatások minimalizálása. Az instrumentáció semmiképpen nem változtathatja meg a kód logikai működését, de egyéb futtatási tulajdonságok (pl. futási idő, memóriahasználat) szempontjából is legfeljebb elhanyagolható mértékben befolyásolhatja a programot.

Dinamikus analízist széleskörűen alkalmaznak a szoftverfejlesztés különféle szintjein, ezt az 1.1. táblázat ábrázolja.

|  |  |
| --- | --- |
| ****Analízis szintje**** | ****Leírás**** |
| Egységtesztelés | A kód logikailag legkisebb logikai egységeinek (tipikusan osztályok) önmagában végzett tesztelése. |
| Integrációs tesztelés | Az önálló szoftveregységek (tipikusan osztályok) együttes tesztelése, melynek során az egységek egymás, és a rendszer felé nyújtott interfészeit vizsgálják. |
| Rendszertesztelés | A kész szoftvertermék tesztelése követelmények, funkcionális specifikáció, rendszerterv szempontjából. |

1.1. táblázat. Dinamikus analízis alkalmazása különféle szinteken

## Hibrid analízis

Hibrid analízis során egyszerre statikus és dinamikus eszközökkel is vizsgáljuk a kérdéses szoftvert. Ennek előnye, hogy a két analízistípus eredményeit egymással kölcsönhatásban is tudjuk értelmezni: lehetőségünk nyílik tehát statikusan, illetve dinamikusan önmagában nem elvégezhető analízisek megismerésére is, további rejtőzködő szoftverhibákat tárva fel ezzel.

Egy számomra igen ígéretes hibrid analízis-irány a dinamikus anomáliakeresés statikus predikciók alapján. Egy statikus eszközökkel kinyert vezérlésfolyam-gráf segítségével a program dinamikus analízise során rendellenes lefutási útvonalakat detektálhatunk.

# JavaScript-kódanalízis

## Analízis-eszköztárak támogatottsága a JavaScript-szcénán belül

## Kódinstrumentáció

## AST-bejárás

## CFG-generálás

## Logelemzés

# Összegyűjtött eszközök

## Fejlesztést közvetlenül támogató eszközök

## Statikus analízis eszközök

## Dinamikus analízis eszközök

# Összefoglalás és további lehetőségek

# Bevezetés

A következő fejezet pár példán keresztül bemutatja a diplomatervekben és szakdolgozatokban szokásosan előkerülő formázások megvalósítását.

## Formázási tudnivalók

A dokumentum folyószövegéhez használjuk a **Normál** (angol Word esetén Normal) stílust.

### Címsorok

A fejezetcímek esetén a **Címsor 1–4** (Heading 1–4) stílusokat használjuk.

### Képek

A képhez használjuk a **Kép** stílust. Képaláírást a képen jobb gombbal kattintva a **Képaláírás beszúrása…** opcióval adhatunk hozzá, így az automatikusan **Képaláírás** (Caption) stílusú lesz.



1.. ábra. Példa képaláírásra

Képek hivatkozásához jelöljük ki a képaláírásban a sorszámot (pl. „1.1.”), majd kattintsunk a **Könyvjelző** gombra, majd hozzunk létre egy könyvjelzőt (pl. „bmelogo” névvel). Ezután a **Kereszthivatkozás** gombra kattintva a **Hivatkozástípus**t állítsuk **Könyvjelző**re és válasszuk ki a **bmelogo** könyvjelzőt. Így ehhez hasonló hivatkozásokat készíthetünk: lásd az XXX. ábrán.

### Táblázatok

A dolgozatban szereplő táblázatokat az 1.1. táblázat mintájára érdemes elkészíteni.

|  |  |  |
| --- | --- | --- |
| ****Fejléc**** | ****Opció A**** | ****Opció B**** |
| 1. sor |  |  |
| 2. sor |  |  |

1.. táblázat. Példa táblázat feliratára

### Kódrészletek

Kódrészletek beillesztése esetén használjuk a **Kód** stílust.

using System;

namespace MyApp

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Szia Világ!");

}

}

}

### Irodalomjegyzék

Az irodalomjegyzék kezelése többféleképpen is megoldható, az alábbiakban két egyszerű módszert ismertetünk.

#### Kereszthivatkozásokkal

A kereszthivatkozásokkal történő irodalomjegyzék egy megfelelően formázott felsorolás, melynek egyes elemeire (bekezdéseire) mutatnak hivatkozások. Jelen dokumentum ezt a megközelítést alkalmazza.

Az Irodalomjegyzékben szereplő hivatkozásokat **Irodalomjegyzék bejegyzés** stílussal formázzuk, a címüket pedig a **Kiemelés** stílussal emeljük ki.

A szövegbe a hivatkozásokat a **Kereszthivatkozás beszúrása** (Insert cross-reference) funkcióval helyezzük el (példa egy így beszúrt hivatkozásra: [1]), így azok később frissíthetők a hivatkozások átrendezése esetén (lásd 2. fejezet).

#### Források kezelése

A kereszthivatkozások alternatívája, hogy a hivatkozott műveket először felvesszük a szerkesztőprogram adatbázisába. Új műveket a **Hivatkozás** fülön a **Források kezelése** alatt az **Új…** gombbal vehetünk fel. A szerzőket érdemes a **Szerző** mező mellett található **Szerkesztés** gomb használatával felvenni.

Az irodalomjegyzéket az **Irodalomjegyzék** gomb alatt az **Irodalomjegyzék** opcióval szúrhatjuk be a dokumentumba. A hivatkozások stílusa a **Stílus** gomb alatt állítható be, a javasolt stílus az **IEEE**.

### Margók

Az **Oldalbeállítás** menüben ellenőrizzük a **Kötésmargó** beállítását. Amennyiben a dolgozat kétoldalas nyomtatással készül, a **Több oldal** beállításnál válasszuk a **Margók tükrözése** opciót.

# Utolsó simítások

Miután elkészültünk a dokumentációval, ne felejtsük el a következő lépéseket:

* Kereszthivatkozások frissítése: miután kijelöltük a teljes szöveget (Ctrl + A), nyomjuk meg az F9 billentyűt, és a Word frissíti az összes kereszthivatkozást. Ilyenkor ellenőrizzük, hogy nem jelent-e meg valahol a „Hiba! A könyvjelző nem létezik.” szöveg.
* Dokumentum tulajdonságok megadása: a dokumentumhoz tartozó metaadatok kitöltése (szerző, cím, kulcsszavak stb.). Ez Word 2013 alatt a **Fájl** | **Információ** | **Tulajdonságok** | **Dokumentumpanel megjelenítése** gombra kattintva érhető el.
* Kinézet ellenőrzése PDF-ben: a dokumentum elkészítése után feltétlenül ellenőrizzük a kapott PDF dokumentumot is.

# Összefoglalás

A diplomaterv összefoglalása.

# Ábrák jegyzéke

[1.1. ábra. Példa képaláírásra 8](#_Toc396824939)

# Táblázatok jegyzéke

[1.1. táblázat. Példa táblázat feliratára 8](#_Toc396824940)

# Irodalomjegyzék

1. Jeney Gábor, Hogyan néz ki egy igényes dokumentum? Néhány szóban az alapvető tipográﬁai szabályokról, Budapesti Műszaki és Gazdaságtudományi Egyetem, Híradástechnikai Tanszék, Budapest, 2007. május 9., online: <http://mcl.hu/~jeneyg/foliak.pdf>
2. William Strunk Jr., E. B. White, The Elements of Style, Fourth Edition, Longman, 4th edition, 1999.
3. Levendovszky, J., Jereb, L., Elek, Zs., Vesztergombi, Gy., Adaptive statistical algorithms in network reliability analysis, Performance Evaluation – Elsevier, Vol. 48, 2002, pp. 225-236
4. National Instruments, LabVIEW grafikus fejlesztői környezet leírása, <http://www.ni.com/> (2014. aug.)
5. Fowler, M., UML Distilled, 3rd edition, ISBN 0-321-19368-7, Addison-Wesley, 2004

# Függelék

A függelék szövege.

1. vö. humán analízis, forráskódértelmezés, code review, software walkthrough [↑](#footnote-ref-1)
2. Elsősorban meglévő eszközökre, eszköztárakra hagyatkoztam a félév során, nem volt feladatom saját forráskód-reprezentációt kidolgozni. [↑](#footnote-ref-2)
3. vö. Concrete Syntax Tree (CST): A hagyományosan Parse Tree-nek is nevezett, tipikusan fordítók által készített reprezentáció a forráskód minden egyes elemét – a whitespace-ektől eltekintve – egyértelműen reprezentálja, a forráskód–CST-transzformáció után a CST–forráskód-transzformációval az eredetivel megegyező kódot kapunk vissza. [↑](#footnote-ref-3)