Óbudai Egyetem

Neumann János Informatikai Kar

Alkalmazott Informatikai Intézet

SZAKDOLGOZAT

FELADATLAP

Hallgató neve: **Lovas István**

Törzskönyvi száma: T/002145/FI12904/N

A dolgozat címe:

**Szoftver követelmények feltárását, analizálását és modellezést támogató intelligens eszköz**

**Intelligent Tool to Support Software Requirements Elicitation, Analysis and Modelling**

Intézményi konzulens: Dr. Tick József

Külső konzulens:

Beadási határidő: 2015. január 7.

A záróvizsga tárgyai: Számítógép architektúrák

Szoftverfejlesztés

**A feladat**

Készítsen szakdolgozatot, melyben bemutatja a szoftver követelménytervezés, illetve a követelmények modellezésének általános megközelítéseit. A szakirodalom alapján ismertesse az alkalmazott elveket, megoldásokat és azon eszközöket, melyek ezt a tevékenységet támogatják. Tervezzen meg, és valósítson meg egy olyan elosztott szoftver-eszközt, mely segíti az akár csoportos követelmény feltárást, követelmény analízist, a követelmények modellezését, összeköti a specifikáció szöveges részeit, a modellreprezentációk különböző absztrakcióival. Az eszköz könnyítse meg az ezek közti navigációt, illetve nyújtson intelligens módon segítséget a modellek elkészítésében, a szövegekből kinyert információk alapján. Az eszköz fejlesztése során alkalmazza a szoftvertechnológia modern megoldásait.

**A dolgozatnak tartalmaznia kell**:

* A feladat részletes leírását.
* Követelménytervezés elterjedt megközelítéseinek bemutatását a szakirodalom alapján.
* A kapcsolódó szövegbányászati eljárások szakirodalom segítségével történő bemutatását.
* A rendszer tervét, UML alkalmazásával.
* Az eszköz megvalósításának leírását
* A rendszer tesztelését, és annak eredményeit.
* Az eszköz továbbfejleszthetőségi lehetőségeit.

Ph.

.....……………….

Dr. Galántai Aurél

mb. intézetigazgató

A szakdolgozat elévülésének határideje: **2017. január 7.**

(OE TVSz 32.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

|  |  |
| --- | --- |
| ……………….. | ..……………………. |
| külső konzulens | intézményi konzulens |

# Tartalomjegyzék

[**Szoftver követelmények feltárását, analizálását és modellezést támogató intelligens eszköz** 1](#_Toc401598676)

[1 Tartalomjegyzék 3](#_Toc401598677)

[2 Ábrajegyzék 7](#_Toc401598678)

[3 Célkitűzések 8](#_Toc401598679)

[4 Irodalomkutatás 9](#_Toc401598680)

[4.1 Bevezetés 9](#_Toc401598681)

[4.2 A Követelménytervezés alapfogalmai 9](#_Toc401598682)

[4.2.1 Szoftverkövetelmény 9](#_Toc401598683)

[4.2.2 Követelmények csoportosításai 10](#_Toc401598684)

[4.2.3 Felhasználói követelmények 10](#_Toc401598685)

[4.2.4 Rendszerkövetelmények 10](#_Toc401598686)

[4.2.5 Rendszer 10](#_Toc401598687)

[4.2.6 Funkcionális követelmények 11](#_Toc401598688)

[4.2.7 Nemfunkcionális követelmények 11](#_Toc401598689)

[4.2.8 Szakterületi követelmény 12](#_Toc401598690)

[4.2.9 Szükséges és javasolt követelmények 12](#_Toc401598691)

[4.2.10 A követelménytervezés folyamata 12](#_Toc401598692)

[4.2.11 Szoftverkövetelmények dokumentuma 12](#_Toc401598693)

[4.2.12 Fogalomszótár 13](#_Toc401598694)

[4.3 A követelmények fontossága 13](#_Toc401598695)

[4.4 Követelménytervezés folyamata 13](#_Toc401598696)

[4.4.1 Megvalósíthatósági tanulmány elkészítése 14](#_Toc401598697)

[4.4.2 Követelmény feltárás és elemzés 14](#_Toc401598698)

[4.4.3 Követelmény specifikációs folyamat 14](#_Toc401598699)

[4.4.4 Követelmény validáció 14](#_Toc401598700)

[4.4.5 Követelmények kezelése 14](#_Toc401598701)

[4.5 A követelmény feltárás, és kezelés problematikái 14](#_Toc401598702)

[4.5.1 A természetes nyelv problémái 14](#_Toc401598703)

[4.5.2 A természetes nyelv problémáit kiküszöbölendő megfontolások 15](#_Toc401598704)

[4.6 Követelmény formátum sablonok és ajánlások 16](#_Toc401598705)

[4.6.1 Felhasználói követelmény sablon 16](#_Toc401598706)

[4.6.2 Rendszerkövetelmény sablon 16](#_Toc401598707)

[4.7 Követelménymodellezés 16](#_Toc401598708)

[4.7.1 Az UML bemutatása 16](#_Toc401598709)

[4.8 Követelmény modellezés eszközei 16](#_Toc401598710)

[4.8.1 Use Case modell 16](#_Toc401598711)

[4.9 Az agilis követelménykezelés 16](#_Toc401598712)

[4.9.1 User Stories: A használati esetek agilis megközelítése? 16](#_Toc401598713)

[4.10 Szakirány vezéreltség 16](#_Toc401598714)

[4.10.1 A TDD, DDD, BDD hármas értelmezése 16](#_Toc401598715)

[4.10.2 Szakirány vezérelt tervezés (Domain Driven Design) 16](#_Toc401598716)

[4.10.3 Girkin nyelven leírt követelmények 16](#_Toc401598717)

[4.11 Egyes szoftveréletciklus modellek és a követelmények kapcsolata 16](#_Toc401598718)

[4.11.1 Vízesés modell 17](#_Toc401598719)

[4.11.2 RUP 17](#_Toc401598720)

[4.11.3 Agilis módszerek 17](#_Toc401598721)

[4.12 A CASE eszközök 17](#_Toc401598722)

[4.13 Szövegbányászat 18](#_Toc401598723)

[4.13.1 Bevezető gondolatok a szövegbányászatról 18](#_Toc401598724)

[4.13.2 Szövegbányászati alapfogalmak 18](#_Toc401598725)

[4.13.3 A szövegbányászat és az adatbányászat összehasonlítása 18](#_Toc401598726)

[4.13.4 Szövegbányászati előfeldolgozás 18](#_Toc401598727)

[4.13.5 Szövegbányászat főbb feladattípusai 19](#_Toc401598728)

[4.14 A legismertebb ingyenes szövegbányászati csomagok és rövid bemutatásuk. 19](#_Toc401598729)

[4.15 Alternatív szövegelemzési megfontolások 19](#_Toc401598730)

[4.16 Az irodalomkutatás alapján leszűrt tanulságok 20](#_Toc401598731)

[4.16.1 Milyen lehet egy jó és hasznos követelményelemzést támogató eszköz. 20](#_Toc401598732)

[4.16.2 Az irodalomkutatás alapján milyen egy jó, és modern CASE eszköz 20](#_Toc401598733)

[4.16.3 A megismert szövegbányászati feladattípusok közül melyek, és milyen formában lehetnek relevánsak egy CASE eszközben 20](#_Toc401598734)

[5 A rendszerrel szemben támasztott követelmények 20](#_Toc401598735)

[5.1 Vázlatos felhasználói követelmények, illetve igények. 20](#_Toc401598736)

[5.2 A rendszerben megjelenő felhasználói szerepkörök 22](#_Toc401598737)

[5.2.1 A rendszer felhasználóinak használati eset nézete 22](#_Toc401598738)

[5.2.2 A rendszerben megjelenő, előre definiált felhasználói szerepkörök jellemzése 22](#_Toc401598739)

[5.3 Funkcionális követelmények 24](#_Toc401598740)

[5.3.1 A rendszer főbb felhasználói funkcióinak áttekintő használati eset nézete 24](#_Toc401598741)

[5.4 Nem funkcionális követelmények 25](#_Toc401598742)

[6 A rendszer kialakítása során felhasználható technológiák és jellemzésük 26](#_Toc401598743)

[6.1 Szerver oldali technológiák 26](#_Toc401598744)

[6.1.1 ASP.NET 26](#_Toc401598745)

[6.1.2 Java 26](#_Toc401598746)

[6.2 A kliens oldali megjelenítés kezelése 26](#_Toc401598747)

[6.2.1 HTML5 26](#_Toc401598748)

[6.2.2 CSS 26](#_Toc401598749)

[6.3 A kliens oldali dinamika kezelése 26](#_Toc401598750)

[6.3.1 JavaScript, és a TypeScript 26](#_Toc401598751)

[6.4 Kommunikációs technológiák 26](#_Toc401598752)

[6.5 Szövegbányászati csomagok 26](#_Toc401598753)

[7 Követelményanalízis 27](#_Toc401598754)

[7.1 A rendszer statikus modelljének elemzése 27](#_Toc401598755)

[7.1.1 Szerkezet áttekintő nézet 27](#_Toc401598756)

[7.1.2 Szerkezeti nézet 28](#_Toc401598757)

[7.2 A rendszer dinamikájának és folyamatainak elemzése 28](#_Toc401598758)

[7.3 Képernyő vázlatok 29](#_Toc401598759)

[8 Rendszerterv 29](#_Toc401598760)

[8.1 Képernyő tervek 29](#_Toc401598761)

[8.2 Csomagáttekintő nézet 29](#_Toc401598762)

[8.3 Részletes komponens leírások 30](#_Toc401598763)

[8.4 Részletes dinamikus nézetek 36](#_Toc401598764)

[8.5 Telepítési nézet és rendszerkörnyezet 40](#_Toc401598765)

[9 Az implementáció részletei 41](#_Toc401598766)

[10 Tesztelés 47](#_Toc401598767)

[10.1 Statikus tesztelés 47](#_Toc401598768)

[10.2 Dinamikus tesztelés 48](#_Toc401598769)

[10.2.1 Egység teszt 48](#_Toc401598770)

[10.2.2 Komponens teszt 49](#_Toc401598771)

[10.2.3 Integrációs teszt 50](#_Toc401598772)

[10.2.4 Rendszer teszt 51](#_Toc401598773)

[10.2.5 Átadási teszt 52](#_Toc401598774)

[11 Továbbfejlesztési lehetőségek 53](#_Toc401598775)

[12 A szakdolgozat tartalmi összefoglalója 54](#_Toc401598776)

[13 Irodalomjegyzék 56](#_Toc401598777)

[14 Mellékletek 57](#_Toc401598778)

# Ábrajegyzék

# Célkitűzések

Célunk egy olyan eszköz kifejlesztése, ami képes segíteni a szoftverfejlesztés egyes magas absztrakciós tevékenységeit. Gondolok itt főképp felhasználók által elkészítendő szoftver-rendszer, követelményeinek specifikációjával, analízisével, illetve modellezésével kapcsolatos teendőkre. A készítendő rendszernek célja továbbá, hogy támogassa a csoportos munkát, lehetőleg úgy, hogy a különböző felhasználók valós időben láthassák egymás munkáit, és közben képesek legyenek kommunikálni is egymással, a rendszert felhasználva. Végül, de nem utolsó sorban a rendszer legyen a lehető legkülönbözőbb környezetekből, és platformokról elérhető, és használható.

# Irodalomkutatás

## Bevezetés

Az elkövetkezendő szakaszokban, a feldolgozott irodalmak alapján bemutatom a szoftverkövetelményeket, a követelménytervezés folyamatát, kitérek a követelmények modellezésére, a szakirány központú szemléletekre, és az agilis követelménykezelésre. Majd röviden írok az egyes szoftver életciklus modellekről, illetve a CASE eszközökről. Ezután a szövegbányászat alapvető koncepcióit mutatom be, és kitérek egyes alternatív, egyszerű szövegkeresés alapú információkinyerési megközelítésekre.

## A Követelménytervezés alapfogalmai

Ebben a szakaszban igyekszem vázolni, a követelményekkel kapcsolatos összes fogalmat. Egyes fontosabb fogalmakra a későbbiekben külön fejezetben részletesebben kitérek. Nem foglalkozok ellenben a kritikus rendszerek speciális igényeivel, illetve a formális specifikációk mélyebb elemzésével, mert ezek a dolgozat hatókörén kívül esnek.

### Célkitűzések

Érdemes a project megkezdése elején egy rövid áttekintést adni, a készítendő szoftver rendszer alapvető céljairól, betöltendő szerepéről. Ezek sokkal inkább alapvető iránymutatásként szolgálnak, mint tényleges követelményként. Ezek lehetnek általános víziók a szoftver hasznosságáról, főbb feladatairól, vázolhatja a főbb használati szituációkat, és környezetet. Ez a legáltalánosabb vázlat, amit a rendszerről készítünk. Érdemes lehet itt megemlíteni, hogy a felhasználók munkáját a termékünk miképpen fogja megkönnyíteni, milyen fontosabb értékeket hordoz a megrendelőknek.

### Szoftverkövetelmény

A rendszer követelményei lehetnek a rendszer funkcionalitására vonatkozó elvárások magas és-vagy alacsonyabb szintű, részletesebb megfogalmazása, illetve a rendszerrel szemben támasztott megszorítások. Megmondja, hogy a rendszer a megrendelő problémáit milyen szolgáltatások révén oldja meg, és megoldás közben milyen általános és probléma centrikus megkötésekre kell odafigyelni. Részletesség szerint két csoportját érdemes megkülönböztetni, a felhasználói-, illetve a rendszerkövetelményeket. Ezeket a csoportokat szoktuk vegyíteni is tehát léteznek a funkcionális felhasználói követelmények, a nem-funkcionális felhasználói követelmények, és a rendszerkövetelmények is lehetnek funkcionálisak, illetve nem-funkcionálisak is. Ezeket az eltérő követelményeket fontos lehet elkülöníteni egymástól, akár jelöléssel, akár úgy, hogy a követelmény dokumentáció külön alfejezeteit alkotják az imént vázolt csoportok, ellenben jó ötlet lehet, az összekapcsolódó, de különböző csoportba sorolt követelmények gyors összevetésére is módot adni, de legalább egy hivatkozást elhelyezni, a kapcsolódó követelményre.

### Követelmények csoportosításai

Követelmények csoportosítása történhet a megfogalmazásuk mélysége szerint. Ugyanis a gyakorlatban fontos szétválasztani a követelmények azon szintjét, mely a megrendelőkkel, illetve a felhasználókkal történő egyeztetést segítik, ezek a felhasználói követelmények, és azt a szintet, melyben a rendszer követelményeit részletezzük, ezeket nevezzük rendszerkövetelményeknek. A követelmények egy másik csoportosítása történhet az alapján, hogy az adott követelmény egy a rendszertől elvárt szolgáltatást részletez, azaz funkcionális követelmény, vagy egy a rendszertől, esetleg annak egyes szolgáltatásaitól elvárt tulajdonságot ír le, tehát nem-funkcionális követelmény.

### Felhasználói követelmények

Absztrakt módon leírja a rendszertől elvárt szolgáltatásokat, külső viselkedését, és azok működési megszorításait. Közérthető, természetes nyelvű leírás, melyet a rendszer felhasználójának, a megrendelőnek, és ezek megbízottjainak szánnak. Esetenként a könnyebb érthetőség kedvéért kiegészítik magas szintű, vázlatos használati eset diagramokkal.

A természetes nyelven megfogalmazott követelmények, az olyan egyértelmű előnyeik mellett, mint az egyszerűségük, olyan problémákat is felvetnek, mint az egyértelműség nehéz biztosítása, ezekkel: A követelmény feltárás, és kezelés problematikái című részben bővebben foglalkozom.

### Rendszerkövetelmények

A rendszer szolgáltatásait, funkcióit, működési feltételeit és megszorításait részletezi. A felhasználói követelményeket fejtik ki bővebben. Érdemes megjegyezni, hogy a rendszerkövetelmények dokumentumát szokták specifikációnak is hívni.

Lényeges a specifikáció pontossága, hisz meghatározza, hogy mit kell implementálni, de ezen a szinten már kivédhetetlenül megjelenhet a hogyan is, emellett azért is fontos a teljes körű és precíz megfogalmazás mert a rendszerspecifikáció gyakran része a szerződésnek.

A rendszerspecifikációból lehetetlen kizárni minden a tervezéshez köthető információt, mert gyakran a követelményspecifikáció strukturálásához segítséget nyújt egy kiindulási architektúra elkészítése. A gyakorlatban emellett gyakran szükséges, hogy együtt működjünk más rendszerekkel, és ez bekorlátolja a tervezést és ezek a korlátok szükségszerűen új követelményekként is megjelenhetnek. Esetenként egy-egy nemfunkcionális követelmény is előírhat egy konkrét tervezési megközelítést, vagy egy konkrét fejlesztési platformot, így ilyenkor is megjelennek tervezési részletek, a rendszerkövetelmények szintjén.

A rendszerkövetelmények tehát megalapozzák a tervezést, illetve az implementációt, így a szoftver megalkotásában résztvevők a legfőbb olvasói. Ezért míg a felhasználói követelményeknél elengedhetetlen a természetes nyelv, és a külön képzettség nélküli érthetőség, addig a rendszerkövetelmények szintjén, az egyszerű formális szövegek mellett speciális jelöléseket és modelleket is alkalmazhatunk. Ilyenek lehetnek a stilizált, formázott és strukturált természetes nyelv, vagy a követelmények grafikus modelljei, mint a részletezett használati esetek vagy akár a matematikai formális nyelvek.

### Rendszer

Rendszer alatt olyan egymáshoz kapcsolódó komponensek, működési egységelemek, halmazát értjük, melyek egy közös cél érdekében dolgoznak együtt. Ezek a rendszerkomponensek munkájuk elérése érdekében, kommunikálnak a többi komponenssel és felhasználhatják azokat. A rendszer tartalmazhat szoftveres, mechanikus, és elektronikus egységeket is. A rendszer célja elérése közben emberi beavatkozást is igényelhet, de lehet teljesen automatikus is.

### Funkcionális követelmények

Leírja a rendszertől elvárt szolgáltatásokat. Az iménti fejezetekből világosan látható, hogy két szintje lehetséges, a funkcionális felhasználói követelmények magas szintű állítások, amik megfogalmazzák a rendszertől elvárt funkcionális viselkedést, míg a funkcionális rendszerkövetelmények, e funkcionalitás részletezései.

### Nemfunkcionális követelmények

Az egész rendszerre vonatkozó tulajdonságok. Olyan eredő rendszertulajdonságokra vonatkozhat, mint a megbízhatóság, a válaszidő vagy a tárfoglalás. Lehet megszorítás egyes használt technológiákra, eljárásokra, vagy szabványokra vonatkozólag. Nem csak a teljes rendszer, de akár annak egy-egy összetevőjére is vonatkozhatnak. A míg, egy funkcionális követelmény nem megfelelő támogatásával a rendszer egyes részei megkerülendővé válhatnak, addig a nemfunkcionális követelmények nem teljesítése akár a teljes rendszert is használhatatlanná teheti. A nemfunkcionális nem csupán magára a rendszerre vonatkozhat, de magára a kifejlesztésének folyamatára is vonatkozhatnak. Megszabhatja tehát a kifejlesztés közben használt metodikát, különböző minőségszabványok betartását írhatja elő, vagy akár megszabhatja a fejlesztés alatt használatos CASE eszközök sorát.

Az ide sorolható változatos követelmények, különböző forrásokból is származhatnak, mint például a szoftver termék kívánt jellemzőiből, a fejlesztő vállalattól, vagy valamilyen külső féltől, mely többnyire a megrendelő, de lehet akár valamilyen előírás is.

Lehetséges, hogy egy adott nemfunkcionális követelmény megléte, szükségessé teszi más, funkcionális követelmények felvételét. Például egy információbiztonsággal kapcsolatos követelmény, megkövetelheti különböző authentikációs és authorizációs funkciók felvételét.

A nemfunkcionális követelmények egyik problematikája, hogy a megrendelők gyakran csak általános célokként fogalmazzák meg. Ilyen nehezen mérhető jelzőket használnak, mint gyors, kisméretű, egyszerűen kezelhető, hordozható, megbízható, stb. Ezek helyett célszerű a követelményekben különböző objektív metrikákat használni. A gyors, jelző kifejezhető inkább a másodpercenkénti tranzakciók számával, egy eseményre adott átlagos válaszidővel, vagy akár a képernyő frissítési sebességével. A méret leírható elfoglalt tárterület alapján, mint például nem lehet nagyobb 64 kilobájtnál, vagy a maximális dinamikus memóriafogyasztással. A könnyű használatot leírhatjuk a betartott tervezési ajánlásokkal, a képzési idővel, vagy a segédképernyők számával. A megbízhatóságot a rendelkezésre állás, a hibák közt eltelt átlagos idő, és a hiba bekövetkeztének valószínűsége írhatja le precízen. A hordozhatóságot jellemezheti például a célrendszertől független utasítások aránya.

### Szakterületi követelmény

A szakterületi, vagy más néven környezeti követelmények, a rendszer alkalmazási területéről származnak. Ezek a követelmények nem éppen felhasználói igények, inkább a szoftver alkalmazásának szakterületéből adódó funkcionalitások, vagy megszorítások.

Ezek teljes értékű funkcionális, vagy nemfunkcionális követelmények, de mind a nyelvezetük, mind fogalomrendszerük az adott szakterülethez igazodik. Ha funkcionális követelmény, akkor előírhatja, hogy az adott feladatot hogyan kell végrehajtani, hogy az helyes legyen, illetve illeszkedjen az adott szakterület bevett gyakorlatához. Lehet akár egy szabvány, de lehet akár egy képlet is, amivel az adott szakterületen számolnak.

Ezen követelmények nagy jelentőségűek, míg az adott szakterület képviselőinek egyértelműek addig a rendszer tervezőinek némiképp idegenek lehetnek a nyelvezetük idegensége és a mögöttes implicit háttértudás hiányában.

### Szükséges és javasolt követelmények

A javaslat, nem mindig követelmény, lehet, hogy csupán egy igény a rendszerre vonatkozólag. Esetleg egy teoretikus elképzelés, ami jelenleg nem fontos, vagy kritikus és nem döntő fontosságú a rendszer szempontjából. Érdemes lehet ezeket is rögzíteni, hisz nyerhetünk belőlük újabb jó ötleteket vagy az is lehet, hogy egy átpriorizálás közben felértékelődik, és szükségessé nyilváníthatják.

A javasolt követelmények nyelvezetében érdemes, ilyen kifejezéseket használni, mint: „jó lenne ha”, vagy „érdemes lenne”, „javasolt lenne”, „javallott” stb.

Míg a szükséges követelmények a „kell”, a „szükséges”, a „fontos”, a „kívánt” az „elvárt”, és hasonló szavak használata különíti el az előző csoporttól.

### A követelménytervezés folyamata

Ebben a folyamatban megértjük és definiáljuk a rendszer által biztosítandó szolgáltatásokat, illetve a fejlesztési és az üzemeltetési megszorításokat. A folyamat végeredménye a követelménydokumentum, mely rend szerint külön tárolja a felhasználói illetve a rendszerkövetelményeket.

### Szoftverkövetelmények dokumentuma

A szoftverkövetelmények dokumentuma, amit szoktak szoftverkövetelmény specifikációnak is hívni, az a dokumentum, mely a követelmény specifikáció folyamata során jön létre. A specifikáció tehát a követelmények egy szabványos formába való szedésének és leírásának folyamata. Maga a dokumentum tartalmazza a felhasználói-, illetve a rendszer követelményeket. Szöveges dokumentum, így a megrendelő külön előképzettség nélkül is olvashatja, de a kötött szerkezete elősegíti a lehető legteljesebb információreprezentálást.

A funkcionális és a nemfunkcionális követelményeket célszerű megkülönböztetni a dokumentumon belül. Ez lehet elszeparálás, ilyenkor a külön elhelyezkedő követelmények között nehezebb megtalálni az esetleges összefüggéseket, viszont az egyes követelmények nem mosódnak egybe, és jól elkülöníthetőek a funkcionális, illetve a nemfunkcionális megfontolások. A megkülönböztetés történhet esetleg más vizuális jellemzőkkel is.

### Fogalomszótár

A követelmények között, vagy a különböző modelleken megjelenhetnek olyan szavak, melyeket érdemes lehet definiálni. Ez több célt is szolgálhat, növeli a szövegek precizitását, megóvhat az egyes félreértésektől, kiküszöböli a kétértelműségeket. A szakterületi kifejezések leírása segítheti a fogalmak megértését, és ezáltal segítheti a hatékonyabb fejlesztést. A rendszerben használt rövidítéseket is érdemes lehet ide felvenni.

## A követelmények fontossága

A követelmények meghatározása, elemzése és karbantartása a megvalósítandó komplex szoftverrendszerek életciklusában egy kritikus fontosságú feladat, hisz ez a szakasz segít megérteni, és hogy a megrendelő mit is vár el a készítendő rendszertől. A feltárt követelmények alapján döntünk arról, hogy a rendszer megvalósítható-e és hogy megközelítőleg mennyibe fog kerülni a megrendelőnek. A szoftver komplexitása is becsülhető általa. Általában a szoftver specifikáció része a szerződésnek.

Hagyományosan a specifikáció köti össze a felhasználó igényeit és a fejlesztőket, így a kommunikációs szerepe is jelentős. Amennyiben a rendszer megrendelői mi magunk vagyunk, vagy a kommunikáció igen jó és gyakori a megrendelővel, illetve egyes metodikákat alkalmazva, a követelmények kifejtése a fejlesztés során több részletben, iteratívan történik.

## Követelménytervezés folyamata

Az a folyamat, melyben felderítjük, elemezzük, dokumentáljuk és ellenőrizzük a szoftverkövetelményeket. Ez a szoftverspecifikáció megalkotásának folyamata, itt készítjük el és tartjuk karban a rendszerkövetelmények dokumentumát.

A folyamat négy nagy tevékenységre bontható. Ezek a részfolyamatok a megvalósíthatósági tanulmány elkészítése, a követelmények feltárása és elemzése, a követelmények validálása, illetve azok kezelése és követése.

### Megvalósíthatósági tanulmány elkészítése

Megvizsgálja és becslést ad arról, hogy a rendszerrel kapcsolatos elvárások kielégíthetőek-e az adott szoftveres és hardveres technológiák segítségével. Eldönti, hogy a rendszer költséghatékony-e az adott üzleti szempontokat figyelembe véve, illetve hogy a költségvetési megszorítások mellett kivitelezhető-e. Lehetőség szerint minél inkább gyors, és olcsó folyamatnak kell lennie. A tanulmány elkészítése során kiszámításra kerül egy ROI (Return Of Investment) érték, mely megadja, hogy milyen mértékben és mikor térül meg az elkészítendő rendszer, és mekkora annak az üzleti haszna. A megvalósíthatósági tanulmány információt biztosít a rendszer elkészítésének költséghatékonyságáról.

A tanulmány elkészítésének végeztével döntés születik a fejlesztés folytatásáról. A megvalósíthatósági jelentés a részfolyamat kimenő dokumentuma, mely tartalmazza a lépésben összegyűjtött fontosabb információkat és döntéseket. Ez után a lépés után következhet a követelmények feltárása részfolyamat.

### Követelmény feltárás és elemzés

A folyamat során a potenciális felhasználókkal és megrendelőkkel történő megbeszélések, és egyeztetések során, illetve az esetleges már működő rendszerek és folyamatok megfigyelése által az elemzők feltérképezik és megértik a készítendő szoftver követelményeit. Ez a folyamat magában foglalhatja egyes rendszermodellek, illetve prototípusok elkészítését, melyek elősegítheti a követelmények pontosabb megértését.

### Követelmény specifikációs folyamat

Az elemzési tevékenység során összegyűjtött információk egységes dokumentummá történő szerkesztésének folyamata.

### Követelmény validáció

A követelmények valószerűségét, konzisztenciáját és teljességét ellenőrző tevékenység.

### Követelmények kezelése

## A követelmény feltárás, és kezelés problematikái

### A természetes nyelv problémái

A természetesen nyelven írt dokumentumok, mint amilyen lehet a felhasználói követelmények is, számos problémával rendelkeznek. Az egyik ilyen probléma az egyértelműség hiánya. Olykor nehéz a nyelvet pontosan használni, egy dolgot többféleképpen is leírhatunk, és a szavainknak is lehet számos jelentése. A szövegeink emellett terjengőssé is válhatnak, ezzel megnehezítve a lényeges információk kiszűrését. A folyó szövegben összemosódhatnak az egyes követelmények, illetve a különböző típusú, funkcionális és nemfunkcionális követelmények keveredhetnek így nehezebb ezeket elválasztani egymástól.

### A természetes nyelv problémáit kiküszöbölendő megfontolások

A követelmények megfogalmazásakor érdemes lehet minden követelményhez egy egyszerű magyarázatot fűzni, ami kifejti, hogy miért került be a megjegyzés, mi a szerepe annak. Ez nem csak érthetőbbé teszi az adott követelményt, de változásakor is segítséget nyújthat.

Érdemes egy szabványos követelményformátum elkésztése. Ami megadja, milyen formai és tartalmi szerkezettel adjuk meg az egyes követelményeket. Egyes ajánlásokban például a követelményeket kártyákra vették fel, minden egyes ilyen kártya egy darab követelményt tartalmaz, illetve tartalmazza az adott követelmény magyarázatát, más követelményekkel való kapcsolatát, azoktól való függését, illetve a követelmény forrását, azaz azt a személyt, akitől a követelmény ered. Így könnyen megtalálható az, akivel a követelmény változásakor érdemes lehet konzultálni. A formátum előírhatja továbbá a követelmény félkövér típussal való kiemelését, esetleg azt, hogy a felhasználói szintű követelmény tartalmazzon egy linket az azt részletező rendszerkövetelményre. Az ilyen kötöttebb, szabványos formátum megelőzheti az esetleges véletlen elhagyásokat, és könnyebben ellenőrizhetővé teszi magukat a követelményeket is.

Mint már egy korábbi szakaszban említettük érdemes lehet szétválasztani a szükséges illetve a javasolt követelményeket, például a „kell” vagy a „javallott” szavak felhasználásával és ezek következetes használatával.

A követelmény kulcsfontosságú részeit érdemes lehet kiemelni, félkövér, dőlt, vagy egyéb szövegformázási eszközökkel, ezzel elősegítve a követelmény későbbi olvasását, és az fontosabb információk hangsúlyozását.

Elkerülendőek a számítástechnikai szakzsargonok, de a rendszer felhasználási területének felhasználása elkerülhetetlen. Érdemes lehet az egyértelműség kedvéért fogalomszótárat használni.

Teljesség, ellentmondás mentesség, konzisztencia.

Analízis paralízis.

Változás.

Nyelvezet.

Elkülönítés, csoportosítás.

## Követelmény formátum sablonok és ajánlások

### Felhasználói követelmény sablon

### Rendszerkövetelmény sablon

## Követelménymodellezés

### Az UML bemutatása

## Követelmény modellezés eszközei

### Use Case modell

## Az agilis követelménykezelés

### User Stories: A használati esetek agilis megközelítése?

Scenario

## Szakirány vezéreltség

### A TDD, DDD, BDD hármas értelmezése

### Szakirány vezérelt tervezés (Domain Driven Design)

### Girkin nyelven leírt követelmények

## Egyes szoftveréletciklus modellek és a követelmények kapcsolata

A szoftverfejlesztési módszertanok olyan módszerek, melyek meghatározzák a fejlesztés menetét és lépéseit. Keretet biztosítanak ahhoz, hogy milyen sorrendben végezzük az egyes a rendszer kifejezését elősegítő lépéseket. A módszertanok tartalmazhatnak különböző eszközöket a modellezésre, jelölési konvenciókat és ajánlásokat. Tanácsokat, esetlegesen betartandó szabványokat, tartalmazhatnak specifikációra, a tervezésre, illetve a fejlesztésre vonatkozólag.

### Vízesés modell

### RUP

### Agilis módszerek

## A CASE eszközök

A CASE (Computer-Aided Software Engineering) rövidítés számítógéppel támogatott szoftvertervezést jelent. Ellenben ezen eszközök jelenleg koránt sem csak a tervezés folyamatát támogatják.

A CASE rövidítés (Computer-Aided Software Engineering) számítógéppel támogatott szoftver tervezést jelent, de ezen eszközök jelenleg koránt sem csak a tervezés folyamatát támogatják.

A CASE eszközök olyan szoftverek vagy szoftver rendszerek, melyek a szoftverfejlesztés különböző fázisait és/vagy ezek közti kapcsolat megteremtését segítik elő, lehetőleg minél több automatikus eszköz és eljárás segítségével. A CASE eszközök többnyire egy fejlesztési metodikához illeszkednek, és ennek a szoftverfolyamatnak a tevékenységeit, vagy azok egy részét támogatják. Persze léteznek általánosan használható CASE eszközök is, melyek nem kötődnek szorosan egy adott metodikához. A szoftverfolyamat tevékenységei közül támogatják például a követelményelemzést, a követelmények változásainak kezelését, a rendszermodellezést és a modellek esetleges ellenőrzését, a tervezést, a fejlesztés nyomkövetését, és akár a tesztelést. CASE eszközök segíthetik a követelmény specifikációs, és a szoftvertervezésnél használt modellek és diagramok előállítását. Egyes CASE eszközök a tervezés elemeiről adatszótárakat tartalmaznak, amik leírják az adott elemeket és azok kapcsolatait. Segíthetik a felhasználói interfész vázlatszerű, vagy teljes értékű elkészítését.

A CASE eszközök, a rutinfeladatok automatizálásával képesek a szoftver minőségén javítani, illetve a fejlesztés sebességét fokozni.

Bár nagymértékben megkönnyítik a szoftverek kifejlesztését, a CASE eszközök nem voltak képesek oly mértékben segíteni a munkát, mint amennyire azt a 1980-as, '90-es években prognosztizálták. Ennek fő okai közül az egyik, hogy a szoftverek kifejlesztése egy egyedi, és kreatív folyamat, melyet, akár mesterséges intelligencia bevonásával is, csak nehezen, vagy egyáltalán nem lehet teljesen automatizálni. A másik jelentős probléma, hogy egyes eszközök nem képesek elégségesen megtámogatni a csoportos munkavégzést, mely a nagy és komplex feladatoknál elengedhetetlen. Szerintem törekedni kell továbbá ezen CASE eszközök minél nagyobb fokú integráltságának, illetve együttműködésének biztosítására. Hisz hiába van egy olyan eszközünk, ami az adott részfolyamatot tökéletesen segíti, de az adott rendszer feltöltése a szükséges adatokkal több időt elvesz, mintha egy olyan rendszert használnánk, amely esetleg nem olyan hatékony céleszköz, de a környező rendszerekkel való kommunikációja révén a feladatot mégis gyorsabban képes ellátni, és a fejlesztési folyamatban az értékek áramlását, kevésbé gátolja.

## A szövegbányászatról általánosságában

Az emberek, már a kezdeti, ősi civilizációkban is, a szóbeli mellett, jellemzően írásbeli szövegek segítségével tárolták, és adták át egymásnak az ismereteiket. Napjainkban a rögzített tudásanyagainknak egyszerű szöveges dokumentumokban található. Ezt a feltevést támasztják alá, többek között a Merill Lynch elemzései is, melyek becslése szerint az üzleti információk körülbelül 85%-a található strukturálatlan, illetve gyengén strukturált szövegekben. Az általunk kezelt szövegek növekvő arányban digitálisan tárolt dokumentumok.

Így talán nem meglepő, hogy a szövegek, és főképp a bennük lévő információk, kezelésének hatékonyabbá tétele, napjaink egyre fontosabbá váló informatikai tevékenysége. Az egyik tudományág mely ezzel foglalkozik, a szövegbányászat.

A szövegbányászatot definiálhatjuk úgy, mint szöveges adatokon végzett feldolgozási és elemzési tevékenység, mely célja a dokumentumban rejtett információk feltárása, azonosítása, és elemzése. A szövegbányászat interdiszciplináris szakterület, mely olyan informatikai eszközök mellett, mint a gépi tanulás és a hatékony algoritmusok, a matematika és a nyelvészet eszközeit is felhasználja.

A szövegbányászat két nagy alaptípusa a keresés és a rendszerezés. A keresésnél kiválasztjuk azokat a dokumentumokat, ahol egy adott keresőkifejezés előfordul, míg a rendszerezésnél valamilyen kategóriákba, vagy előre nem definiált csoportokba soroljuk azokat. A szövegbányászat főbb feladattípusai is ilyen jellegű feladatokat, vagy ezek kombinációját hajtják végre céljaik elérése során.

Ezek a főbb feladattípusok a kereséstámogatás és információ-visszakeresés, az Információkinyerés, az osztályozás, a csoportosítás, az összegzéskészítés, a kivonatolás, a válaszkereső rendszerek, a szövegelemzés, és a napjainkban egyre inkább tért nyerő webes tartalomkeresés. Ezekről külön alfejezetekben részletesebben is írok, de csak a dolgozat szempontjából leginkább hangsúlyos információkinyerést fogom mélyebben részletezni.

## Szövegbányászati előfeldolgozás

Számos szövegbányászati feladat megoldható, már létező, adatbányászati eszközökkel, és algoritmusokkal. Ehhez a szöveges adatokat úgy kell transzformálni, olyan alakra kell hozni, hogy ezek a bejáratott eljárások, lehetőleg hatékonyan, képesek legyenek működni rajta. Ezt a transzformációs lépést szokták előfeldolgozásnak is nevezni. A folyamat végeredménye a dokumentumot reprezentáló modell. Az előfeldolgozás egységesítési, formalizációs és normalizációs feladatokat is tartalmaz.

## Az előfeldolgozás közben lényeges alapvető dokumentumjellemzők

### dokumentumot hordozó médium:

Ha digitális, gondoskodni kell az elérhetőségről, ha pedig analóg, akkor előbb digitalizálni kell. ha képi, akkor a képen látható szövegeket, ki kell nyerni, lehetőleg automatikus karakterfelismeréses vagy akár manuális begépeléses módszerrel.

### A dokumentum elérési helye:

A hagyományos világban lehet egy könyvtár egy polca, míg a digitális világban jellemzően lehet egy URL (Uniform Resource Locator) mely egy egységes erőforrás-azonosító és az interneten keresztüli adatelérésre szolgál, vagy lehet egy háttértárolón egy fájl elérési útvonala. Szövegbányászatnál mindenképpen fontos, hogy rendelkezzünk minimum olvasási jogokkal az adott dokumentumon.

### Mérete

A nyers digitális állományok legelterjedtebb általános méretmegadási módja az adattárolón elfoglalt terület alapján, bájt alapon történik. A szövegfájlok mérete megadható még benne szereplő karakterek, szavak vagy az oldalaik számával. A szöveges korpuszok méretét a benne szereplő dokumentumok összesített mérete adja.

Az előfeldolgozási fázis után a méretet jellemzően szavaik, ritkábban karaktereik számával adhatjuk meg. A modellbeli reprezentációjuk méretét pedig olyan a felírásukra használt elemek számával jellemzik, mint a tokenek vagy az egyedi szavak száma, esetleg a modell indextömbjének mérete.

### statisztikai jellemzői

A későbbi feldolgozás során szükségünk lehet egyes statisztikai adatokra is a dokumentumunkról. A legfontosabb ilyen jellemző lehet a szavak eloszlása, de esetenként fontos lehet még a karaktereinek eloszlása, a szóhosszoknak az eloszlása vagy átlaga. A magán és mássalhangzók száma és eloszlása, vagy a mélyebb nyelvi elemzéssel kideríthető zöngés és zöngétlen magán- illetve mássalhangzók száma, melyekből például következtetni lehet az adott dokumentum nyelvére.

### Metaadatai

A dokumentum adathordozón való tárolásakor keletkeznek metaadatok, mint a keletkezési ideje, az utolsó módosítás ideje, a fájlnév, illetve a hozzáférést módosító attribútumok.

Szövegbányászati szempontból egyéb jellemzők is fontosa, ilyen például a dokumentum tartalmára vagy a keletkezési körülményeire vonatkozó metaadatok, mint a nyelv, a keletkezési idő/hely, a dokumentum témája vagy más szempont szerinti besorolása, a stílusa, a nyelvezete vagy zsánere. Ezek osztályozáskor vagy csoportosításkor lehetnek segítségünkre. Ezek közül az adatok közül nem áll rendelkezésünkre minden, de néhányat, ha hiányzik, szövegbányászati eszközökkel meg tudunk határozni. A nyelv vagy a téma is lehet ilyen.

### Formátuma

A dokumentum formátumát többnyire a készítésére használt eszköz típusa bekorlátozza. egyes eszközök egyszerű szövegként (plain text, txt) mentenek, mígy másik eszközök formátumai különböző formázási és szerkezetre vonatkozóadatok, de akár a dokumentum régebbi változatai is lehetnek. Az ilyen tárolási formátumok, a teljesség igénye nélkül, például a doc/docx, a pdf, rtf, de akár a HTML és az XML is ide sorolható.

A dokumentum készítői által kiemelt szövegrészek lehetnek külön formázásúak, mint például a félkövéren szedett, a dőlt, vagy az aláhúzott szövegrészek, és lehetnek szerkezetileg meghatározó helyen, mint a különböző szintű címekben, vagy például a webes elérésű dokumentumok az elérési útvonalában is szerepelhetnek. A dokumentumban kiemelt kifejezések a dokumentum szempontjából relevánsabbak lehetnek, ezért érdemes lehet, ezeket az iformációkat a modell reprezentációba is átvinni.

### Karakterkódolása

Az általunk használt adatok különböző karakterkódolásokban fordulhatnak elő, hiszen a számítógép, a szöveges adatokat is kettes számrendszerű számokként tárolja, ezért szükséges annak a meghatározása, hogy egy-egy szám milyen karaktert kódol.

Az egyik első széles körben használt kódtábla az ASCII (American Standard Code for Information Interchange), amit az Amerikai Egyesült államokban alkották meg, először hét biten ábrázolt, ami 128 karakter ábrázolását teszi lehetővé, melyek némelyike vezérlő karakter, mint az új sor, vagy a tabulátor, illetve vannak nem látható vezérlő karakterek is melyek a korabeli nyomtatókat vezérelték. Az ASCII 8. bitjét eleinte paritásbitnek szánták, de később általánossá vált, hogy a karakterkészlet kibővtésére használták, az így nyert további 128 lehetséges állapottal az ISO 8859-1 Latin-1-es kódolás a nyugat európai, míg az ISO 8859-2 inkább a kelet európai nyelvek speciális ékezetes karaktereit írja le. Napjainkban egyre inkább az Unicode veszi át a vezető karakterkódolási szabvány szerepét, mely arra hivatott, hogy legyen képes az ismert nyelvek összes karakterét kódolni. Legelterjedtebb megvalósítása az UTF-8, mely változó hosszon kódolja a karaktereket, ezért tömör kódolást biztosít, mindeközben kompatibilis a 7 bites ASCII-vel. A karakterkódolások különbözősége mellett olyan dolgok is nehezíti a feladatunkat, mint hogy a a sor végének jelölése egyes platformokon eltérhet.

Az ismert kódolások részben, vagy teljesen inkompatibilisek, ezért a konvertálásnál kiemelt kérdés a karakterkódolás, mivel ennek a helytelen használata, a beolvasott szöveg torzulásához vezethet, vagy az akár teljesen használhatatlanná is válhat.

A karakterkódolási információt esetenként maga a fájl tartalmazza, ha nem ismerjük, akkor az előfeldolgozás helyessége nem garantálható.

## Szövegbányászati modellalkotás

### Az ismertebb dokumentum reprezentációs modellek bemutatása

A legelterjedtebb modellek jellemzően valamilyen numerikus objektumok. Ez számos előnnyel jár, mint a kisebb tárolási méret. Ugyanis ha a szavakat karakterenkénet letároljuk, és jellemzően egy-két bájtos egy karakter, akkor nagyobb helyet vennénk igénybe, mint a numerikus tárolásnál, ahol szavanként egy darab 2 vagy akár 4 bájtos számmal számolunk. Mindemellett, a számok használatának van egy másik jelentős előnye, méghozzá az, hogy matematikai műveleteket, és transzformációkat hajthatunk végre az ilyen modelleken. Ráadásul a modellben a matematikai eszközökkel való munka elősegíti a dokumentumok hatékony kezelését.

Azt, hogy ténylegesen milyen modellt és adatábrázolást használunk befolyásolja a megoldandó feladat típusa. Keresés jellegű feladatoknál egy megfelelő szóelőfordulás táblázat is nagy szolgálatot tehet, míg a rendezés jellegű feladatoknál összetettebb dokumentum összehasonlító módszerekre van szükség.

A modellalkotásnál használt három nagy matematikai elméleti megközelítés a halmazelméleti, az algebrai, illetve a valószínűségelmélet alapú. A halmazelmélet alapú modellek jó szolgálatot tehetnek az egyes keresőrendszerekben, hisz kereséskor minden dokumentumra fennáll, hogy része az eredményhalmaznak, vagy sem. Az algebrai modellben a dokumentumokat olyan algebrai objektumokként reprezentáljuk mint a vektor, vagy a mátrix. Ezeket algebrai műveletekkel össze is hasonlíthatjuk, ezért ezek már használhatók, rendszerezési feladatok megoldásánál is. A legelterjedtebb megvalósítása a vektortér modell és annak változatai. A valószínűségi modellben maguk a dokumentumok valószínűségi események által reprezentáltak, míg a kapcsolataik feltételes valószínűségi becslések eredménye. Ezek mellett léteznek komplex függvénytani eszközöket alkalmazó spektrális szövegbányászati modellek is.

### A vektortér modell

A vektortér modellben hatékonyan meg lehet határozni a dokumentumok távolságát, illetve hasonlóságát. A szövegbányászatnál gyakran élünk különböző intuitív heurisztikákkal, melyek meghatározzák, az ezeket felhasználó eszközök felhasználhatósági körét és korlátait. A vektortér modellnél azt jelentjük ki intuitív módon, hogy azokat a dokumentumokat tekintjük hasonlónak, melyek szókészlete átfedi egymást, és ennek a hasonlóságnak a mértéke arányos az átfedés mértékével. A modell egy sokdimenziós vektortérben, vektorokkal reprezentálja a dokumentumokat. A vektortérben az egyes dimenzióit a dokumentumgyűjtemény egyedi szavai adják. Tehát egy-egy dokumentum a szavaiból álló vektor, abban a vektortérben, ahol az egyes szavak a téret kifeszítő vektorok. A dokumentumgyűjteményt a szó-dokumentum mátrixszal reprezentáljuk. Az egyedi szavak összessége a szótár, vagy más néven lexikon.

Általában a teljes szótár csak egy kis része fordul elő egy-egy dokumentumban, ezért a szó-dokumentum mátrix ritka. Mindemellett az egyedi szavak száma meglehetősen nagy, akár több százezres, vagy akár milliós nagyságrendet is elérheti. Ezért a mátrix méretét nyelvtechnológiai, illetve matematikai eszközökkel érdemes lehet csökkenteni. A vektortérmodell másik nehézsége az alaphipotézisből származik, miszerint a szavak megléte, illetve száma adja a dokumentumok hasonlóságát. De nem tér ki, és nem is tárolja a szavak sorrendjére se a szövegen belüli pozícióira vonatkozó információkat. Ezért hívják szózsákmodellnek is. Ez a modell a feladatok nagy részében jól használható, egyszerű és hatékon megoldást nyújt. A szósorrend elvesztéséből való hátrányát sokszor más modellekkel való együttes alkalmazásával küszöbölik ki.

### A szó-dokumentum mátrix jellemző súlyozási sémái

A legegyszerűbb módszer, ami csak a szó dokumentumbeli esetleges meglétét jelöli, a bináris reprezentáció. Ahol a mátrix egy adott dokumentumot reprezentáló oszlopvektorában, egy adott szóhoz tartozó sorban nullát írunk, ha az adott szó nem szerepel a dokumentumban és egyet, ha igen, és nem számít az, hogy hányszor, ez az információ elvész. Egy másik egyszerű módszer, ha a súly maga az előfordulások száma. Itt viszont felvetődhet a kérdés, miszerint tényleg lináris-e a függés a szó adott dokumentumra vonatkozó relevanciája, és az adott szó előfordulásainak száma között. Tehát egy jóval gyakrabban előforduló szó tényleg ugyanilyen mértékben fontosabb is. Az esetek nagy többségében a válasz egyértelműen nem, sokkal nagyobb megkülönböztető értéke van, ha az adott szó előfordul a dokumentumban, mint annak, hogy az tízszer, vagy százszor fordul elő. Ezt kiküszöbölendő kezdték el használni a logaritmikus súlyozást. Ezen egyszerű súlyozási sémák nagy hátránya, hogy nem veszik figyelembe az egyes dokumentumok hosszát. Pedig feltételezhetően egy rövidebb dokumentumban egy szó bizonyos számú előfordulása fontosabb, mint az ugyanekkora számú előfordulása egy jóval hosszabb szövegben. Így a bonyolultabb súlyozási függvények figyelembe veszik a szó dokumentumbeli előfordulási gyakoriságát is. Ezt a gyakoriságot az angol irodalomben term frequency kifejezéssel illetik, és a szakirodalmakban gyakran TF-nek rövidítik. Ezen elven alapuló jellemző megvalósítások a gyakoriság alapú, illetve a normalizált logaritmikus súlyozások. Ez a megközelítés már közelebb áll a valósághoz, de nem veszi figyelembe azt, hogy vannak olyan szavak, melyek minden dokumentumban egységesen sűrűn fordulnak elő és nincsenek semmilyen összefüggésben az adott dokumentumok tartalmával. Az angol terminológiát átvéve ezeket stopszavaknak hívjuk. Ezért a szó fontosságát befolyásolhatja a teljes gyűjteményben való előfeldolgozásainak száma is, amit szoktunk gyűjteménytámogatottságnak is nevezni, vagy angolul collection frequency az elterjedt megnevezése. Fontos lehet továbbá az előfordulások eloszlása a korpuszon, ugyanis a koncentráltan kevés dokumentumban, de azokon belül nagy számban szereplő szavak valószínűleg fontosabbak is az adott dokumentumokra nézve, illetve nagyobb a dokumentumok közti megkülönböztető szerepük is. Ha nk-val jelőljük a tk szót tartalmazó dokumentumok számát, és N pedig a dokumentumok száma a korpuszban. Akkor a dokumentum gyakoriság a df = nk/N képlettel számolhatjuk. Ez a szám megadja, az adott szó ritkaságát, azt hogy a szó megléte, és annak száma mennyire különbözteti meg a dokumentumokat egymástól az adott korpuszban. Az idf (inverse document frequency) súlyozási sémában ennek inverzével számolunk. Maga a dokumentumgyakoriság inverzét számoló egyik elterjedt súlyozási függvény:

idf(tk) = log(N/nk)

A leggyakrabban használt tk-idf (term frequency and inverse document frequency) súlyozást úgy kapjuk, hogy a vektortérmodell szavakat reprezentáló tengelyeit az adott szavak idf által megadott relevanciájával arányosan súlyozzuk.

Tehát a szó-dokumentum mátrix adott dik súlya kiszámítható a

dik= fki \* idf(tk) képlettel

E mellett számos más képlete, és módosulata létezik, amikre most nem térek ki.

A tf-idf súlyozás értéke magas lesz a nagy megkülönböztető képességű, adott dokumentumra gyakori, de a korpuszra ritka szavaknál. Alacsonyabb lesz a korpuszban gyakoribb, vagy az adott dokumentumban ritkább szavakban, és elhanyagolhatóan alacsony, akár zérus, az olyan szavakban melyek az egész korpuszban gyakran fordulnak elő.

### Módok a szöveg felbontására, illetve a szótár felépítésére

Mielőtt felírhatnánk a választott modellbe a dokumentumainkat, számos azt megelőző előfeldolgozási lépésre lehet szükségünk. Az első ilyen lépés a reprezentációs egységekre való bontás. Ez leggyakrabban egyszerű szavakra való bontás, de egyes esetekben, mint például a kivonatolás a dokumentumok bekezdésekre történő, illetve mondat szintű felbontására is szükség van. Ezután a vektortér modell használata előtt meg kell határozni a szótár szavait is. Az összes egyedi szó szövegekből való kinyerése után, célszerű lehet ezek egy kis fontosságú, de esetenként sok szót tartalmazó, részét elhagyni, ezzel jelentősen csökkentve, az indexelt szavak számát és a vektortér méretét.

### A strukturális szegmentálás

Dokumentumaink szövege számos hierarchiaszintbe sorolható. Ilyen strukturális egységek lehetnek például a kötet, rész fejezet, szakasz, pont, illetve a bekezdés a mondatok és a szavak. Ezek és az ezekből összeállítható dokumentumtérkép automatikus meghatározása egyes fájlformátumokból kinyerhető információk alapján viszonylag egyszerű feladat is lehet, míg például az egyszerű szövegekből nehézkes, és esetenként koránt sem egyértelmű feladat is lehet.

### Mondatokra bontás

A szövegeink mondatokra való bontása automatizálható, de nem triviális feladat. Kézenfekvő lenne egyszerűen a mondatvégi írásjeleket nézni, és ezek szerint elszeparálni a mondatokat. Ilyenkor viszont álmondathatárokkal is találkozhatunk, hisz például a pont is előfordulhat mondat közepén, rövidítéseknél, sorszámoknál, dátumokban, vagy akár IP-címekben is. A feladat megoldására általában szabály alapú döntési algoritmust alkalmaznak, mely az ilyen téves mondathatárokra utaló, vagy ezeket cáfoló jelekre és tulajdonságokra tartalmaz szabályokat. Ezeket a szabályokat próbáljuk illeszteni a szövegeinkre. A tulajdonságok lehetnek felszíniek, mint például a jelet követő szó nagy kezdőbetűs e vagy sem, esetleg a jel egy szó közepén van e, stb. Az algoritmus az illesztés mellett kereshet gyakori álmondathatárokat tartalmazó listákban is, mint például a rövidítéslisták. Az ilyen listák nyelvenként eltérőek lehetnek, és időnként a lista karbantartására, frissítésére is szükség lehet. A szabályok tartalmazhatják, hogy a vélt határtól milyen pozícióra elhelyezkedő és milyen jellegű szavakra, vagy pontosabban tokenekre vonatkozik, és szokás megadni hozzájuk egy súly értéket is, hogy amennyiben egy végere több szabvány is illeszkedik a nagyobb súllyal rendelkező juthasson érvényre. Legyen a szabályrendszerűnk bármilyen komplex, és kifinomult is, előfordul, hogy az algoritmus nem képes jól dönteni, mert a mondat, vagy mondatok értelmezésével lehet csak jól meghatározni, a mondatok valódi határát.

### Tokenizálás

A token egy bizonyos karaktersorozat konkrét előfordulása, míg típusnak nevezzük az azonos tokenek osztályát. Így a típusok összessége alapján állítható elő a szótár. Ez a tokenizációs lépés minden lényeges szövegbányászati feladatnál, mind a keresés, mind pedig a rendszerezés jellegűeknél, egy fontos előfeldolgozási lépés. Hisz míg például a keresésénél ezekre illesztünk, információkinyerésnél, ezek potenciális információt hordozó entitások lehetnek, addig a dokumentumok rendszerezésénél, ezek olyan attribútumok lehetnek, segítségével mérhetjük az egyes dokumentumok összetartozását, hasonlóságát. Ezek a feldolgozásunk legkisebb részelemei. A gyakorlatban sokszor egyszerű szavak, de mint a fenti definícióból látható lehetnek más szövegegységek, és karaktersorozatok, ritkán és szélsőséges esetekben egész dokumentumok is. A tokenek alatt leggyakrabban mégis elszeparált szavakat értünk. Az ilyen tokenek előállítása elsőre egyszerű is lehet, hisz elég csak az írásjeleket elhagyva, a szóközöknél elválasztani a szöveget. De számos további kérdés is felmerül, mint, hogy hogyan kezelhetjük a kötőjellel elválasztott vagy aposztrófot tartalmazó szavakat, egyben vagy külön tokenként, vagy az egyéb írásjeleket tartalmazó szavakat, mint a kukac és pont karaktert tartalmazó e-mail címeket. Vagy hogyan kezeljük azokat a jeleket, amilyen az informatikusok többségének egyértelműen külön fogalmat reprezentáló C++, itt sem hagyható el a két plusz jel, hisz úgy már egy merőben más fogalmat kapnánk. Kérdés továbbá, hogy miként kezeljük a tulajdon neveket, különböző együttesek vagy csoportok több tagból álló neveit vagy a konkrét eseményt jelölő dátumokat. Természetesen az előbbi kérdésekre több helyes válasz is elképzelhető a feladatunktól függően, illetve a nyelvi és a felhasználási környezet is nagyban befolyásolhatja, hogy konkrétan milyen módon tokenizálunk.

Egy alternatív megoldás is létezik, amit karakter n-gramm módszernek nevezünk. Ekkor a szöveget n hosszúságú karaktersorozatokra bontjuk. Például: a „karakter 4-gramm” szöveget 4-grammokra bontva a következő eredményt kapjuk: kara, arak, rakt, akte, kter, ter , er 4, r 4-, 4-g, 4-gr, -gra, gram, ramm. Ez az eljárás jól használható nyelvek meghatározásakor, vagy más egyszerű osztályozási problémánál, esetleg olyan jellemzően távol-keleti nyelveknél szótárépítésre, ahol nincsenek a különböző szavak szóközökkel elválasztva egymástól. De nem célszerű a használata az európai eredetű nyelveknél olyan feladatoknál, ahol a szótár elemeinek értelmes egységeknek kell lenniük.

A tokenek típusai lesznek a nyelvi elemzés alapvető szövegreprezentáló egységei, és különböző szűrő eljárások után a típusok építik fel magát a szótárat is.

### Stoppszószűrés

Egyes szavak, illetve tokenek nagy számban fordulnak elő a dokumentumgyűjtemény dokumentumaiban és nincsenek szoros kapcsolatba azok témájával, ilyenek a névelők, a határozószavak, és a névutók. Ezek a stopszavak. A stopszószűrésre, vagy azok feldolgozás közbeni kiküszöbölésére láttunk egy matematikai jellegű módszert, a td-idf súlyozás szerepében. Tehát a stopszavakat a tokentípusok előállítása után, különböző dokumentumgyakorisági adataik alapján határozzák meg, majd a gyakorlatban esetenként manuálisan ellenőriznek, és ezek egy részét, a feladattól függően elhagyhatják. Napjainkban például keresésénél, esetleg maximum néhány tízes vagy akár tíz alatti szót hagynak el. Míg az elhagyások száma egyes osztályozási és csoportosítási feladatokban ez több százas vagy akár ezres nagyságú is lehet.

Ennek az az oka, hogy a keresésénél, az elhagyott stopszavak szerepelhetnek ismert kifejezésekben, különböző irodalmi vagy más művészeti művek címeiben, és elhagyásukkal ezekre való keresés is nehézkessé válna. Az indexállományok tömörítése, illetve a megnövekedett egyre olcsóbb tárkapacitás együttesen elérték, hogy az idők folyamán egyre kevesebb és kevesebb stopszót kelljen elhagynunk. Mindeközben a rendezési, és más pontos egyezést nem igénylő feladatoknál, ezek a szavak, konkrét jelentés nélkül, csak megnehezítenék a feldolgozást, nélkülük nagy mértékben lecsökkenhet a reprezentációs modell mérete. Gyakran szabják az adott korpuszra, az elhagyható szavak listáját, így biztosítva a lehető leghatékonyabb és mégis biztonságos feldolgozást.

### Lemmatizálás és Szótövezés

A stopszószűrés mellett léteznek nyelvészeti eszközök is a szótárak szűrésükre és méretének redukálására. Az alapötlet az, hogy a legtöbb nyelvben vannak a szavaknak különböző módosult alakjai. Ha ezeket a különböző alakokat egy közös alakként tárolnánk, egyes feladatokban elhanyagolható szóalak információ elvesztése mellett, nagy mértékben redukálhatnánk a modell méretét. A csökkentés mértékét befolyásolhatja az adott nyelv morfológiájának gazdagsága. Például ez a mérték az Angol nyelvnél megközelítőleg 40-70 százalék között lehet, addig a Magyarnál elérheti akár a 90%-ot is. A lemmatizálás, és a szótövezés is ilyen közös, úgymond kanonikus alakra hozó módszerek. A szó módosult alakjait a toldalékok, mint a képző, a jel, és a rag, adják. Ezek a toldalékok a szó előtt prefix-ként, és a szó után is állhatnak, körül is vehetik a szótövet, vagy a magyartól idegen módon, be is ékelődhetnek a szótőbe. A közös alakra hozásnál, ezeket a toldalékokat igyekszünk valamilyen módon leválasztani az adott szóról.

A lemmatizálás közben a szó normalizált szótári alakját azaz a lemmáját határozzuk meg. Nyelvészeti motivációjú eljárás közben, mindig értelmes szótári alakot hozunk létre. A lemmatizálás nem egyértelmű, ezért az eredmény több szó is lehet. Például a falunk szó szótári alakja a falu, de a fal szó is.

A szótövezés egyfajta szócsonkolást eredményez. Nem az a célja, hogy értelmes szótári szó jöjjön létre, sokkal inkább az, hogy a kialakított új szó a valódi szövegben is megtalálható legyen. A szótövező eredménye függ az adott algoritmustól.

Hasonlítsuk össze néhány példán a két megközelítést. A munkát és a munkám szavak lemmája a munka, szótöve algoritmustól függően a munka, vagy a munká. A ló lemmája a lovak, lovát és maga a ló szavaknak, de ezek szótöve a választott szótövező függvényében lehet lahet a lo, vagy a ló is.

### Szövegbányászati alapfogalmak

Ontológia

Natural Language Process

### A szövegbányászat és az adatbányászat összehasonlítása

### Szövegbányászati előfeldolgozás

Számos szövegbányászati feladat megoldható, már létező, adatbányászati eszközökkel, és algoritmusokkal. Ehhez a szöveges adatokat úgy kell transzformálni, olyan alakra kell hozni, hogy ezek a bejáratott eljárások, lehetőleg hatékonyan, képesek legyenek működni rajta. Ezt a transzformációs lépést szokták előfeldolgozásnak is nevezni.

### Szövegbányászat főbb feladattípusai

Kereséstámogatás, információ-visszakeresés

Információkinyerés

Osztályozás

Csoportosítás

Összegzéskészítés, kivonatolás

Szövegelemzés

Webes tartalomkeresés

Válaszkereső rendszerek

## A legismertebb ingyenes szövegbányászati csomagok és rövid bemutatásuk.

StanfordNLP

OpenNLP

GATE

<https://gate.ac.uk/>

UIMA

R nyelv

MagyarLánc

## Alternatív szövegelemzési megfontolások

„Abbot”; clean codeing;

## Az irodalomkutatás alapján leszűrt tanulságok

### Milyen lehet egy jó és hasznos követelményelemzést támogató eszköz.

### Az irodalomkutatás alapján milyen egy jó, és modern CASE eszköz

Véleményem szerint a jövő CASE eszközeinek a csoportos munka támogatására kell törekedniük, illetve arra, hogy a szoftverfejlesztés minél nagyobb területét lefedjék, és ezt úgy tegyék, hogy egymással a lehető legnagyobb összhangban működnek. Persze emellett a továbbiakban is törekedniük kell a munka, mimnél teljesebb, minél hatékonyabb, a lehető legkevesebb emberi beavatkozást igénylő, segítésére.

### A megismert szövegbányászati feladattípusok közül melyek, és milyen formában lehetnek relevánsak egy CASE eszközben

# A rendszerrel szemben támasztott követelmények

## Vázlatos felhasználói követelmények, illetve igények.

(VÁZLATOS)

A megvalósítandó rendszer egy követelmény specifikációs és követelmény analízist támogató intelligens, elosztott eszköz.

Az eszköz kell, hogy rendelkezzen webes felhasználói felülettel, mely lehetőleg minél több platformon helyesen jelenik meg.

Könnyen kezelhető és intelligens eszközöket biztosít a követelmény feltárás, elemzés és modellezés megkönnyítésére.

Módot kell adnia a különböző követelmény- és modell elemek egymáshoz való kapcsolására, és a meghatározott kapcsolatok segítségével történő navigálásra,

Esetleg jó lenne, ha lehetőséget biztosítana az összekapcsolt elemek egy azon nézetben való megjelenítésére.

Jó lenne, ha a rendszerben, különböző szótárak kezelésével lehetőséget biztosítana a rendszerben jelen lévő entitások azonosítására, esetleges szövegekben való kijelölésére, írás közbeni szöveg-kiegészítésére.

Jó lenne, ha a rendszerrel végzett munkát, minél előbb, akár a munkavégzés pillanatában, valós időben látnák a rendszer további felhasználói.

A rendszer módot biztosít a rendszer felhasználóinak különböző felhasználói szerepkörökhöz való rendelésére.

Jó lenne, ha a rendszer, az alap felhasználói szerepkörök finomhangolására is módot adna, az arra jogosultak számára.

A rendszer felhasználói interfészének törekednie kell egyes modern felülettervezési ajánlások betartására.

Jó lenne, ha a felhasználó által kijelölt elemek környékén megjelenő környezetfüggő menü átlátható módon tartalmazná a legfontosabb teendőket az adott elemen, ezzel elősegítve a kényelmes és hatékony kezelést. Az itt nem megjelenő menüelemek, egy a modern Office programokból megismert szalagmenüben kaphatnának helyet.

A rendszernek módot kell biztosítania a további bővítésének megkönnyítésére.

Jó lenne, ha egy követelmény megváltozása kijelölné felülvizsgálatra a kapcsolódó követelmény, és modell elemeket.

Jó lenne, ha le lehetne kérdezni, hogy mely követelmények lettek kifejtve az adott szinten, és melyek azok, amelyek még „érintetlenek”.

Az eltérő követelménytípusok különböző színű „kártyákon” jelenhetnének meg.

## A rendszerben megjelenő felhasználói szerepkörök

### A rendszer felhasználóinak használati eset nézete



### A rendszerben megjelenő, előre definiált felhasználói szerepkörök jellemzése

A felhasználó szerepköre megszabja, hogy az adott felhasználó milyen módon és milyen jogokkal férhet a rendszerhez. A különböző szerepkörök képviselői hasonló felhasználói felülettel rendelkeznek, de a számukra nem engedélyezett funkciók, nem jelennek meg előttük.

A **General Reader** (általános olvasó) van a hierarchia tetején és a legáltalánosabb jogokkal rendelkezik a rendszeren belül.

A **Software Architect**, a **Business Analyst**, illetve a **Convention Manager** az olvasó szerepkört specializálják, az ő funkcionalitását öröklik, illetve saját felelősségi körökkel bővítik. Tehát a General Reader lehetősségein felül további jogokkal, és lehetősségekkel rendelkeznek.

A **Convention Manager** a rendszerben megjelenő sablonokat, konvenciókat, és megkötéseket olvashatja.

A **Business Analyst** a felhasználói követelmények feltárására, és szerkesztésére specializálódott szerepkör.

A **Software Architect** a rendszerkövetelményeket, és rendszermodelleket állítja elő a magas szintű követelmények alapján.

A Convention Manager konkretizálása a **Company Contact**, aki a cégre vonatkozó általános konvenciókat szerkeszti, illetve a **Project Manager**, aki project szinten finomítja ezt. A **Project Manager** a Business Analyst leszármazottja is, így a magas szintű követelmények szerkesztésére is módja nyílik.

## Funkcionális követelmények

### A rendszer főbb felhasználói funkcióinak áttekintő használati eset nézete



A rendszer fő funkciói, az UML használati esetet modelljének OMG általi ajánlásainak megfelelően, kerettel el van határolva, a rendszer külső felhasználóitól.

A General Reader általános olvasási jogokkal rendelkezik, illetve megjegyzéseket is hozzáfűzhet a rendszer elemeihez.

A Software Architect kezeli a rendszerkövetelményeket, míg a Business Analyst a felhasználói követelményeket szerkeszti.

Míg a Convention Manager olvashatja a céges, illetve a project szintű megszorításokat, addig a Company Contact, illetve a Project Manager szerkeztheti is a felelősségi körének megfelelő konvenciókat.

A szerepkörök egy másik ága az Administrator, aki a felhasználókat, és azok szerepköreit és jogosultságait kezeli.

## Nem funkcionális követelmények

# A rendszer kialakítása során felhasználható technológiák és jellemzésük

## Szerver oldali technológiák

### ASP.NET

### Java

EE vagy Spring MVC esetleg valami egyszerűbb REST FW?

## A kliens oldali megjelenítés kezelése

### HTML5

SVG vagy Canvas

### CSS

## A kliens oldali dinamika kezelése

### JavaScript, és a TypeScript

## Kommunikációs technológiák

REST – a WinApi, illetve egy Javas megoldás;

SignalR

## Szövegbányászati csomagok

# Követelményanalízis

## A rendszer statikus modelljének elemzése

### Szerkezet áttekintő nézet

### Szerkezeti nézet

## A rendszer dinamikájának és folyamatainak elemzése

## Képernyő vázlatok

# Rendszerterv

## Képernyő tervek

## Csomagáttekintő nézet

## Részletes komponens leírások

## Részletes dinamikus nézetek

## Telepítési nézet és rendszerkörnyezet

# Az implementáció részletei

# Tesztelés

## Statikus tesztelés

## Dinamikus tesztelés

### Egység teszt

### Komponens teszt

### Integrációs teszt

### Rendszer teszt

### Átadási teszt

# Továbbfejlesztési lehetőségek

# A szakdolgozat tartalmi összefoglalója

# Irodalomjegyzék

Ian Somerwille: Szoftver rendszerek fejlesztése

Vég Csaba: Alkalmazásfejlesztés a Unified Modelling Language szabványos jelöléseivel

# Mellékletek