Debreceni Egyetem

Informatikai Kar

Webalkalmazások tervezése projektszemléletben

Témavezető:

Dr. Juhász István
egyetemi adjunktus

Készítette:

Krakomperger Róbert
programtervező informatikus (MSc.)

Debrecen
2009

Tartalomjegyzék

	Bevezetés	2.
I.	**Miért fontos a projektszemlélet?**	4.
I.1.	**A The Chaos Report**	4.
I.2.	**A Chaos Report általam kulcsfontosságúnak vélt megállapításai**	6.
I.3.	**A projekt nem IT szemszögből**	9.
I.4.	**Konklúzió**	12.
I.5.	**Hipotéziseim a projektmenedzsment hibáinak okaira**	13.
II.	**A tervezés során használt szoftverek**	16.
III.	**Az első lépés előtt**	18.
IV.	**A tervezés menete**	20.
IV.1	**Projektalapítás**	20.
IV.2	**Projektterv**	20.
IV.3	**Specifikáció**	22.
IV.4.	**Szoftvertervezés**	24.
V.	**Továbbfejlesztési lehetőségek**	30.
VI.	**Implementációs kérdések**	32.
	Összefoglalás	36.
	Irodalomjegyzék	38.
	Függelék	39.

Bevezetés

Öt éves egyetemi pályafutásom alatt az informatika legkülönfélébb területeivel ismerkedhettem meg. Betekintést nyertem a szoftverfejleszés alapjaiba, a C programozási nyelven keresztül az eljárásorientált, majd a Java nyelven keresztül az objektumorientál paradigma sajátosságaival ismerkedtem meg. Az alapvető eszközök által nyújtotta lehetőségek elérhetővé tették számomra a Java Enterprise világát, mely által napjaink egyik legdinamikusabban fejlődő vívmánya, a web kapui nyíltak meg előttem úgy, hogy azt nemcsak használni, hanem alakítani is képessé váltam. Attól a pillanattól kezdve, hogy megírtam első szervletemet, a Java EE technológiákat sorra véve többnél több újdonságot igyekeztem megismerni. Így kerültem kapcsolatba a JSF, Struts, Spring és Hibernate keretrendszerekkel, az Enterprise Java Beans technológiával, különböző szervlet konténerekkel, portál motorokkal, adatbáziskezelő rendszerekkel és még rengeteg más olyan eszközzel, melyek bármilyen módon is épülnek az előzőekre, kiegészítik azokat, vagy esetleg éppen az alapjait képezik, úgy mint – a teljesség igénye nélkül – a Liferay és BEA WebLogic portál, JNDI, JDBC, Reflection, Java Beans és Apache termékek garmadája. Igyekeztem ismereteimet tudatosan, az alapoktól kezdve az ezekre épülő magasabb technológiák felé orientálódva felépíteni. Így jutottam el a webszolgáltatásokon át a SOA alapjaihoz, majd az Enterprise Service Busokig.

Természetesen, mindezen tudás megszerzése rengeteg munkát és mindenekelőtt időt igényelt, többször kudarcot, máskor sikerélményt, olykor tanácstalanságot és elveszettséget okozva. Amint egyre szélesebb teret láttam be az informatika ezen világában, annál összetettebbé és ezáltal áttekinthetetlenebbé váltak a szoftverek, melyek készítésébe belevágtam. Ez egyértelmű iránymutató volt afelé, hogy ráébredjek, a technológiai tudás korántsem minden, sőt, a tervezés, a letisztult modell szinte mindenek felett áll. Úgy gondolom, hogy a standardok követése, magas fokú szakértelem és technológiai ismeretek, dokumentáció és széleskörű tesztelés nélkül nem lesz jó szoftver. Viszont körültekintő tervezés és projektmenedzsment nélkül nem hogy jó, de inkább

semmilyen szoftver sem lesz. Mindez rávitt arra, hogy dolgozatomat két vezérfonal mentén építsem fel: szoftvertervezés és projektmenedzsment. Munkámmal tehát igyekszem bemutatni egy összetett alkalmazás megtervezésének menetét, figyelembe véve a projektmenedzsment támogatását.

Tanulmányaim során korábban már kapcsolatba kerültem napjaink egyik legelterjedtebb modellezési eszközével, az UML nyelvvel. Dolgozatomban az UML nyújtotta lehetőségeket fogom felhasználni, mint tervezőeszközt.

Természetesen nemcsak egyetemi keretek között, hanem az üzleti életben is igyekeztem kamatoztatni tudásomat. Dolgoztam webfejlesztőként, adatbázis programozóként, alkalmazásfejlesztőként, s újabban architektként is. Talán utóbbi tárta elém a legtöbb olyan problémát, mely e dolgozat témáját is kialakította. Tervezőként nemcsak a jó modell felépítését és kialakítását, hanem az alkalmazás fejlesztésének menetét, metodológiáját is szem előtt kell tartanom, többek között az emberi erőforrásokat és az egyes modulok kifejlesztéséhez szükséges időt is, tehát a projektmenedzsmentet. Ez a feladat igen sok újdonságot és kihívást jelentett – s jelent mind a mai napig –, s sok olyan problémára felhívta a figyelmemet, melyeket e dolgozatban szeretnék az olvasó elé tárni.

Megjegyezném, hogy mindkét fő témakör, melyek vezérfonalként szolgáltak munkám során, igen összetett, rengeteg tudást és tapasztalatot vár el. Ebből kifolyólag, hiába dolgozom évek óta a szakmában, terveim még mindig erősen igénylik a fejlesztés általi visszajelzéseket egy-egy tervezési hibát illetően. Persze itt kell megjegyeznem, hogy célom e dolgozattal nem az, hogy egy tökéletes – mely talán lehetetlen is – modellt mutassak be, hanem az, hogy rávilágítsak az informatika világában igen gyakran elhanyagolt kulcsfontosságú alapvetésekre.

I. Miért fontos a projektszemlélet?

A szoftverfejlesztés mint iparág, mióta létezik, több krízisen is átesett, melyek igen erősen befolyásolják az informatikai befektetések sikerességét. Az ily módon előálló problémák megszüntetése vagy hatásainak enyhítése a mai napig sarkalatos pontja az IT szférában működő organizációk tevékenységének. Újabbnál újabb fejlesztési metodológiák, paradigmák, technológiák születnek, melyek többé-kevésbé a minél hatékonyabb, hibatűrőbb és sikeresebb szoftverfejlesztést helyezik előtérbe. Ennek ellenére kevés eredménye mutatkozik az ezirányú próbálkozásoknak.

Utóbbi állításomat saját és több éve a szakmában dolgozó releváns személyek tapasztalataira tudom alapozni.

Továbbá itt ejtenék néhány szót a The Standish Groupról (http://www.standishgroup.com/) és az általuk készített [1] The Chaos Report (1994) dokumentum néhány fontos megállapításáról.

A The Standish Group főbb üzleti tevékenységei:

- Információ Technológiai befektetések vizsgálata, kockázat- és költségelemzése, ár-érték arányainak elemzése
- A fenti tevékenységek által gyűjtött adatok alapján az egyes IT befektetésekkel foglalkozó vállalatok támogatása

Fentiek alapján a csoport igen nagy rálátással bír az informatikai projektek sikerének vagy bukásának körülményeire.

I.1 A The Chaos Report

A kimutatás fókuszában sikertelen IT projektek, azok legfőbb okai, a sikertelenség elkerülésének kulcsfontosságú feltételei állnak.

Metodológia: Az eredmények több vezető menedzserrel történt interjúkon és átfogó kutatásokon alapulnak. A kutatások a jelentősebb ágazatokban (pl: bank, biztosítás, egészségügy, termelés

stb.) tevékenykedő kis, közepes és nagy vállalatok eredményeit egyaránt vizsgálták, ezzel biztosítva egy objektív kép kialakulását.

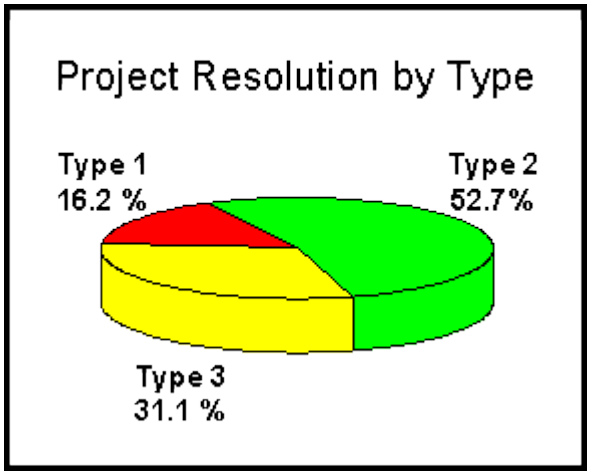
A vizsgált projekteket 3 főbb osztályba sorolták:

Type 1: a projekt időben, az előre meghatározott költségkeretben fejeződött be és a specifikációban foglaltaknak teljes mértékben megfelelő eredményt produkálta

Type 2: a projekt befejeződött, a produktum működőképes, viszont magasabb költségek és idő árán készült el, továbbá az eredetileg specifikált követelményeknek csak egy részhalmazát teljesíti

Type 3: a projekt a produktum elkészülte előtt le lett állítva.

A kutatás eredménye:



A fenti ábrán látható eredmény a különböző méretű cégekre lebontva is közel azonos. A költségkeret túllépésének egyik fő oka a projektek - akár többszöri – újraindítása. A teljes költség átlagos mértéke az eredetileg meghatározott büdzsé 189%-a. A projektre fordított összidő pedig átlagosan az eredetileg meghatározott időkeret 222%-a. A specifikációban meghatározott funkcionalitásnak átlagosan mindössze 61%-a lett megvalósítva.

Úgy gondolom, hogy a fenti eredményeket látva bárkinek az első gondolata ez lenne: SIRALMAS!

I.2. A Chaos Report általam kulcsfontosságúnak vélt megállapításai:

1. Párhuzam a hídépítés és szoftverfejlesztés között: egyrészt általában a hidak időben, a meghatározott költségkereteken belül épülnek föl, aztán nem omlanak össze. Másrészt a szoftverek sosem készülnek el időben és/vagy a meghatározott költségkereten belül. Ráadásul mindig összeomlanak (1986, Alfred Spector, a Transarc Corporation elnöke). Természetesen nem elhanyagolható különbség, hogy egy híd tervei elkészültük után befagyasztásra kerülnek, mely a szoftverfejlesztésben egyáltalán nem megengedhető, valamint a hídépítés mögött 3000 év tapasztalat áll. Viszont egy híd összeomásának körülményei igen alapos feltárás során feljegyzésre kerülnek. Nem így a szoftverfejlesztésben. Az IT bukások, sikertelenségek körülményeit nem veszik figyelembe, elfedik, elmossák vagy egyszerűen észre sem veszik.
2. IT vezetők véleményei alapján a sikeres projekt elengedhetetlen feltételei
   - leendő felhasználók bevonása
   - vezető menedzserek támogatása
   - jól meghatározott, közel fix (időben nagyjából állandó) követelmények
3. Idézet IT vezetőktől:
   1. “Being that it's a mindset, it's very difficult to get all of the management ... to agree on a set of rules”
   2. “We just had a reorganization today. So now that's going to sap all the resources. And explaining to senior management that, 'Well, it's really taking us the time we said it was going to take. But because you've reorganized the company, I'm going to take another six months on this other project, because I'm doing something else for you. That's the biggest issue I have.”
   3. “Changes, changes, changes; they're the real killers.”
   4. “Brain-dead users, just plain brain-dead users”
   5. “The project was two years late and three years in development. We had thirty people

on the project. We delivered an application the user didn't need. They had stopped selling the product over a year before.”

4. A Standish Group kiemelt 2-2 híresebb projektet a Type 1 (sikeres) és a Type 3 (sikertelen) csoportból, s összeállított egy-egy esettanulmányt a projektek körülményeiről. Ezek alapján a sikeres projekt legfontosabb kritériuma a leendő felhasználó bevonása, míg a legkevésbé fontos a kemény munka és a mély technológiai szakértelemmel bíró csapat lett.

   A konkrét esettanulmányok:

   1. California DMV:
      - A projekt: sofőrjeik jogosítványát és regisztrációját támogató alkalmazás újraírása a piac új technológiáinak felhasználásával (az alkalmazás korszerűsítése lett volna a cél)
      - A projekt jellemzői: a projekt nem hozott anyagi hasznot, nem volt megfelelő menedzsment, gyenge követelményfeltárás és tervezés, nem tisztázott célok
      - Az eredmény: annak ellenére, hogy egy viszonylag egyszerű fejlesztés lett volna, a projekt megbukott
   2. American Airlines:
      - A projekt: autókölcsönzéseket és hotel foglalásokat regisztráló alkalmazás fejlesztése
      - A projekt jellemzői: az IT vezetők nemcsak külső támogatóként, hanem a projekt aktív résztvevőiként vettek részt a fejlesztésben. Emelett félkész követelmények, a leendő felhasználók ignorálása valamint a követelmények és a specifikáció állandó változtatása jellemezte a projektet.
      - Az eredmény: Bukás.
   3. Hyatt hotels:
      - A projekt: szobafoglalásokat regisztráló és karbantartó alkalmazás fejlesztése
      - A projekt jellemzői: felhasználók bevonása, a felső vezetők támogatása, letisztult

követelmények, megfelelő tervezés, több, jobban granulált mérföldkövek

- Az eredmény: a specifikációban foglaltakat teljesítő alkalmazás készült el idő előtt, költségkereten belül, extra szolgáltatásokkal

4. Banco Itamarati:
   - A projekt: 2000-re Brazilia Top 5 bankjai közé tartozni. Ennek eléréséhez kulcsfontosságú elemként felhasználva az elérhető IT vívmányokat. Kliens-szerver architektúra kiépítése a bank üzleti folyamatainak támogatására
   - A projekt jellemzői: letisztult követelmények és célok, erős felhasználói részvétel, az ügyfelek magasfokú bevonása, stratégia alkalmazása, mely az eredményeket mérhető formában prezentálja
   - Az eredmény: 51% profitnövekedés, Brazilia bankjainak sorában 47. helyről a 15. helyre sikerült felkúszniuk.

A Standish Group munkáját nyomonkövetve az előzőekben vázolt arányok napjainkig javuló tendenciát mutatnak, viszont az újabb statisztikák (*függelék 1., 2., 3.)* sem kielégítőek. Az IT projektek sikerességi rátájának növekedése több tényező együtthatásának következménye lehet. Többek között köszönhető új fejlesztési metodológiák megjelenésének, élen az agilis fejlesztéssel. Bővebb információ: [5]

Az objektivitást megőrizvén meg kell említenem, hogy a Standish Group által prezentált Chaos Reportok megjelenésük óta (1994) közvita tárgyát képezik. Többen megkérdőjelezik ([3]) mások kiállnak hitelessége mellett ([2]). A Standish Group módszereiről és vizsgálataik mintáját képező cégekről és szerveződésekről további információ olvasható itt: [4].

I.3. A projekt nem IT szemszögből

Amennyiben általánosan, nem elsősorban az informatika oldaláról – mindinkább globálisan, szakterületekre nem összpontosítva – tekintünk egy projektet, a projekt jellemzői az alábbiaknak megfelelően alakulnak.

A projekt három kulcsfontosságú eleme (projekt háromszög):

- Idő
- Költség
- Minőség

Általában a sikeres projekt előfeltételei nagyvonalakban:

1. A projekt világos, letisztult célkitűzésekkel kell bírjon. Az egyes célok elérhetőségének valamilyen formában bizonyítottnak kell lenni (pl. megvalósíthatósági tanulmány). A projekt határainak kialakítása, kompromisszumok felállítása, a projekt értelmes keretek közé helyezése kritikus fontossággal bír.
2. A projekt tagjait a lehető legnagyobb odafigyeléssel kell megválasztani figyelembe véve a leendő tag szaktudását, a pozíciót, melyre egy adott tagot megválasztunk, a tagok között a projekt kezdete előtt fennálló szociális kapcsolatokat, a tagok vérmérsékletét és szociális viselkedését.
3. A projekthez közel álló külső személyek támogatása és bizalma (külső vezetők, beszállítók, az eredmény haszonélvezői, befektetők stb.) elengedhetetlen.
4. A projekt végrehajtásához szükséges erőforrások rendelkezésre állása, megfelelő szervezése.
5. Állandó, irányított és tervezett kommunikáció mind a projekt résztvevői között, mind a külső *stakeholder*ek irányában.
6. A projekt lépéseinek tervezése, időben és költségkeretben való elhelyezése, az erőforrások megfelelő elosztása.

7. Folyamatos kontroll, a tevékenységek állandó felülvizsgálata, monitorozása, a projektben való előrehaladás regisztrálása, az eredmények megfelelő kommunikálása az érintettek felé, minden hasznos információ megfelelően struktúrált rögzítése, az egyes momentumok dokumentálása.
8. Kockázat- és változáskezelési mechanizmusok alkalmazása, tervezése. Külső és belső körülmények változásának hatékony és gyors kezelése. A fellépő problémák, azok körülményei és a problémára adott megoldás dokumentálása.
9. A projekt megfelelő lezárása: az összegyűlt dokumentumok kiértékelése későbbi felhasználás céljából.

Utóbbi megállapításaimat összevetve a Chaos Reportban foglaltakkal könnyen felfedezhetők az alábbi párhuzamok:

Chaos Reportban vázolt problémák	**Alapvető projekt tényezők**
1. ...egy híd összeomásának körülményei igen alapos feltárás során feljegyzésre kerülnek. Nem így a szoftverfejlesztésben. Az IT bukások, sikertelenségek körülményeit nem veszik figyelembe, elfedik, elmossák vagy egyszerűen észre sem veszik.	7-8-9. megállapítás: Folyamatos dokumentáció és monitorozás s az ezek alapján a lezárás során előálló értékelés felhasználásával a korábbi tapasztalatokra építve a legtöbb ilyen kudarc elkerülhető.
2. Leendő felhasználók bevonása	5. megállapítás: Jól szervezett külső kommunikációval biztosítható.
2. vezető menedzserek támogatása	3. megállapítás: Külső szereplők támogatása és bizalma.

2. jól meghatározott, közel fix (időben nagyjából állandó) követelmények	1-7-8. megállapítás: Jól meghatározott célok jól meghatározott követelményeket eredményeznek, továbbá megfelelő változáskezeléssel jól ellensúlyozhatók az időben kissé változó követelmények által okozott problémák
3.1. “Being that it's a mindset, it's very difficult to get all of the management ... to agree on a set of rules”	2-3-9. megállapítás: A külső (felső vezetők) és belső (projekt szereplői) résztvevők megfelelő támogatása és szervezettsége megoldás lehet. Valamint korábbi, dokumentált tapasztalatok kellő meggyőző erővel bírhatnak az egyes szereplők felé.
3.2 “We just had a reorganization today. So now that's going to sap all the resources. And explaining to senior management that, 'Well, it's really taking us the time we said it was going to take. But because you've reorganized the company, I'm going to take another six months on this other project, because I'm doing something else for you. That's the biggest issue I have.”	2-3. megállapítás: A szereplők összeállítása, a felső vezetés kellő támogatása elengedhetetlen. 6-7-8. megállapítás: Jól tervezett és szervezett erőforrások a külső támogatás és a megfelelő változáskezelés mellett a probléma megoldható. Ha mégsem, kellő dokumentáltság és kommunikáció esetén könnyedén feltárható az előállt helyzet, melyet a vezetőségnek tudomásul kell venni.
3.3 “Changes, changes, changes; they're the real killers.”	5-8. megállapítás: Tipikus változáskezelést és jó kommunikációt igénylő probléma

3.4 “Brain-dead users, just plain brain-dead users”	5. megállapítás: talán a legnehezebben kezelhető probléma. Csak türelemmel és a lehető legszofisztikáltabb kommunikációs módszerekkel oldható fel (vagy fel kell hagyni az IT szférában való tevékenykedéssel).
3.5 “The project was two years late and three years in development. We had thirty people on the project. We delivered an application the user didn't need. They had stopped selling the product over a year before.”.	1-5. megállapítás Letisztult, jól formált célok és a külső kommunikáció hiányában ilyen problémák biztosan előállnak.

A 4.1-es és 4.2-es projektek tipikus ellenpéldái egy sikeres projektnek, szinte az összes alapvetésre hoznak valamilyen ellentétes megoldást. A 4.3-as és 4.4-es projektek pedig az előzőeknek tipikus ellenpéldái. A Chaos Report alapján pedig egyértelműen kijelenthető, hogy az utóbbiakból szignifikánsan kevesebb van mint az előbbiekből.

I.4. Konklúzió:

A fenti szembeállítással azt szerettem volna megmutatni, hogy az IT projektek már a sikeres projekt alapvető feltételeit sem teljesítik. Így hogyan is lehetne elvárni, hogy a fejlesztések sikerrel fejeződjenek be, amikor egy IT projekt végrehajtásához még rengeteg, további, a sikert garantáló feltételnek kell megfelelni, mint például továbbfejlesztési lehetőségek biztosítása, sztandardok betartása, az egyes technológiák rendeltetésszerű használata, bugmentesség, szofisztikált tesztelési metodológiák alkalmazása mind felhasználói, mind szoftverfejlesztői aspektusban, könnyű módosíthatóság, lazán kapcsolt alrendszerek használata és még sorolhatnám.

Természetesen, mint ahogy az a Standish Group kimutatásaiból is látható, jó szoftverfejlesztési metodológiák használata nagyban javíthatja egy IT projekt sikerességi rátáját, hiszen az IT fejlesztésekben szükséges feltételeket igyekeznek kikényszeríteni, emelett biztosítják az általános

értelemben vett projekt sikerességéhez szükséges feltételek betartását.
Újra rápillantva a Standish Group eredményeire és a szabad szemmel megfigyelhető tendenciára, mindez sajnos továbbra is kevés.
Jim Johnson szerint a Standish Group a projektek bukását helyezi előtérbe, s igyekszik figyelmen kívül hagyni a projektmenedzsment által okozott bukásokat, ezzel igyekezvén rávilágítani a csak technológiai hiányosságokra. Ebből arra tudok következtetni, hogy az 1994 és 2004 között végbement javuló tendencia valóban a technológiai fejlődés eredménye, vagyis az újonnan megjelenő rendszerfejlesztési módszereknek, technikai vívmányoknak, új programozási szemléletmódoknak köszönhető. Az pedig, hogy a statisztikák még mindig nem mutatnak kielégítő eredményeket, igen nagy valószínűséggel igen nagy százalékban az emberi tényezőnek köszönhető, vagyis figyelembe véve, hogy milyen jellemzőkön buknak el az IT projektek, az alapvető projektszemléletbeli és projektmenedzsmentbeli hiányosságok állhatnak a háttérben.

I.5. Hipotéziseim a projektmenedzsment hibáinak okaira:

- Véleményem szerint a probléma már az oktatásnál kezdődik. Általános és középiskolákban a projektszemlélet kialakítását célzó módszerek szinte meg sem jelennek. A felsőoktatásban pedig nem elég hangsúlyos. Sokszor még az alapvető technológiai ismeretek elsajátítása sem megfelelően támogatott, továbbá projekt jellegű feladatok igen ritkán kerülnek kiosztásra. Ha mégis, akkor pedig a projekt alapvető támpillérei, a projekt háromszög oldalai szinte semmilyen hangsúlyt nem kapnak (az egyedüli, viszont annál szigorúbban figyelembe vett tényező az idő, és az is csak azért, mert az egyetemi/főiskolai oktatás félévekre van bontva, melyek között nem jellemző az átjárás). A minőség és a ráfordított költség (bármi is legyen az ebben az aspektusban) csak a legritkább esetekben jelenik meg.
  A sikeres projekt alapvető feltételei szintén nem teljesülnek. Nincs tervezett és szervezett folyamatos kommunikáció. A legtöbb esetben a csoportmunkában dolgozó hallgató(k)

közötti kommunikáció elenyésző, az oktató és a hallgatók közötti, a projektet érintő kommunikáció pedig kétszer nyilvánul meg: amikor a feladat kiosztásra kerül, majd amikor a hallgatók prezentálják az eredményeiket. A többi alapvető feltételt nem is említem, mert azokon a területeken még siralmasabb a helyzet. Külföldön szerzett szerény tapasztalataim valamivel jobb eredményt mutatnak, de még messze van az is az elvárható minimumtól.

Mindezek következtében a leendő szoftverfejlesztőkben nem alakul ki a saját maguk menedzseléséhez szükséges szemléletmód és tapasztalat, ami gátat szab azon fontos tulajdonságnak, melynek segítségével a fejlesztő jól közelítő idő és munkamennyiség becslést képes nyújtani, ami pedig rendkívül szignifikáns segítség lenne a menedzsment számára, ezáltal elősegítené a projekt sikerét.

- Internship/munkavállalás: Kis cégeknél a legtöbb esetben ugyanaz a helyzet, mint az oktatásban. Nincsenek kialakult, jól meghatározott sztandardok, a “projektek” végrehajtása ad-hoc módon, dokumentálatlanul, időbeli csúszással és hatalmas minőségbeli hiányosságokkal végződik. A multinacionális cégeknél már sokkal jobb a helyzet, legalábbis az alapvető projektszemléletet tekintve. Ilyen helyeken elsősorban technológiai oldalon vannak szignifikáns hiányosságok és ad-hoc, átgondolatlan megoldások. Ezekre megoldást nyúthatnak a jól kidolgozott sztandard metodológiák.
- Kevesen hajlandóak vagy egyáltalán képesek betölteni azt a rést, mely elválasztja a szoftverfejlesztőt a projektmenedzsertől. A legtöbb esetben egy személy vagy projektmenedzser, ezáltal hiányában van a fejlesztésekhez szükséges háttérismereteknek, vagy pedig szoftverfejlesztőből avanzsál projektmenedzserré, így viszont hiányában van egy projekt menedzseléséhez szükséges háttérismereteknek.
- Bármilyen okból hiányosak a projektvezetéshez szükséges ismeretek, vagy ha a szükséges tudás létezik is, hiányzik a tapasztalat, mely által az elmélet átültethető a gyakorlatba. Esetleg emberi mulasztás okán nincs is meg a szándék arra, hogy az elmélet valóban gyakorlattá váljon.

- Az emberi tényezők figyelmen kívül hagyása: nincs meg az a minimális empatikus, szociális és pszichológiai ismeret, mely által a projekt résztvevőit kellően inspirálhatja, irányíthatja a projekt vezetője, sőt, a legtöbb esetben e tényezők fontossága nincs eléggé kihangsúlyozva, vagy egyáltalán figyelembe véve.
- A szereplők hatáskörének összemosása vagy nem megfelelő kialakítása. Ezáltal a projekt önszabályozó mechanizmusai nem, vagy csak csekély mértékben érvényesülhetnek.

A fent ismertetett tények, eredmények vittek arra, hogy a következőkben bemutatott alkalmazás tervét projektszemlélet mentén vázoljam, s a terv magába foglaljon a projektmenedzsmenthez elengedhetetlen dokumentumokat, ezzel is mintegy irányt mutatva az olvasó számára.

Természetesen kompromisszumokat kell lefektetnem arra vonatkozóan, hogy milyen jellemzőket veszek figyelembe, hiszen engem nem kötnek olyan tényezők, mint megbízó által szabott költség- és időkeret, nincs külső nyomás, valamint a tervezésben egyedül veszek részt. E tények – melyek egy az üzleti életben végrehajtandó éles projektet nagyon szigorúan meghatároznak – ellenére igyekszem a valóságot a lehető legnagyobb mértékben megközelíteni.

II. A tervezés során használt szoftverek

A tervezés során kizárólag nyílt forráskódú vagy szabad szoftvereket használtam:

- OpenOffice: e dokumentumot valamint az összes szöveges dokumentumot, mely a terv és a specifikáció elkészítéséhez szükséges OpenOffice-ban készítettem. Alternatívák:
  - MS Office: használhatóság és megjelenés szempontjából elfogadhatóbbnak bizonyulhat az OpenOffice-szal szemben, viszont ár/érték arányában meglehetősen alulmúlja azt. Valós környezetben viszont sajnos az MS Office a leggyakoribb a megbízói oldalon, így az átadott vagy közösen használt - megbízói és fejlesztői oldalon egyaránt szerkesztésre kerülő – dokumentumok esetén problémát jelent, mivel a két szoftver nem mondható kifejezetten kompatibilisnek.
  - GoogleDocs: megfelelő alternatíva lehet, ha mindkét fél (megbízó – fejlesztő) hajlandó a használatára, valamint ha szolgáltatásai elegendőnek bizonyulnak. Problémás lehet, ha komoly üzleti titoknak számító információk kerülnek a dokumentumokba, figyelembe véve a GoogleDocs körül néha előforduló hibákat (pl. nem megosztott dokumentumok hirtelen publikussá válnak)
  - Wiki: használata valamivel nehézkesebb a többi szoftvernél, wysiwig kiegészítéssel viszont a kevésbé haladó felhasználók is viszonylag könnyedén használhatják. Megoldás lehet, ha mindkét fél elfogadja.
- ArgoUML: néhány fontosabb és elterjedtebb UML diagram készítésére alkalmas. Használata kissé nehézkes, az előállt dokumentumok igen puritánnak mondhatók. Összehasonlítva a többi ingyenes UML szerkesztővel, ez bizonyult a legjobbnak, ezért ezt választottam.

  Alternatívák:
  - Visual Paradigm: eddigi tapasztalataim alapján a legszélesebb körű szolgáltatásokat

nyújtja. Használhatóság és dizájn szempontjából szintén kimagasló. A probléma az, hogy az egyetlen ingyenesen elérhető változatot (*Community Edition*) minden hónapban meg kell újítani, és ami a leghátborzongatóbb, hogy adott típusú diagramból egy projekten belül maximum egy db készíthető. Továbbá nagyon erőforrásigényes.

- Umbrello: KDE (Linux) alapú alkalmazás. Használhatóság alacsony, dizájn nem túl kedvező, és rendkívül bugos.
- Eclipse MDK: egyáltalán nem hozta az elvárt szintet
- Netbeans: elfogadható dizájn, elfogadható használhatóság, korlátozott szabadság a diagramtípusokban. Viszont osztálydiagramok *reverse-engineering* útján történő generálására egész jó.
- DbVisualizer: Nem tartozik szorosan a fentebb említett eszközök közé, viszont mindenképpen meg szeretném említeni. Java alapú alkalmazás, mely bármely JDBC meghajtóval rendelkező adatbáziskezelő rendszerhez tud kapcsolódni, s egy kiválasztott adatbázisból ER diagramot generál. Meglepően jól rendezi el az entitásokat. Nincsenek egymást keresztező kapcsolatokat reprezentáló vonalak, s a generált diagram többféle elrendezésbe szervezhető.

- GanttProject: a projekt során szükséges erőforrások, végrehajtandó feladatok szervezését támogatja, minimális eszközöket nyújt a projekt nyomkövetésére. Gantt és PERT nézetben használható. Ami igazán megfogott benne, az az egyszerűsége, és a széleskörű exportálási lehetőségek (felső vezetés felé történő riportálást jól támogatja ezáltal). Alternatívák:
  - MS Project: sokkal összetettebb, sokkal szélesebb körű szolgáltatásokat nyújt, használhatósága jóval magasabb, mint a GanttProject esetén. Viszont pénzbe kerül.
- FreeMind: brainstorming támogatására abszolút nyerő alkalmazás. Könnyen kezelhető, gyors. Konzekvens és jól használható billentyűkombinációk segítik a gördülékeny munkát.

III. Az első lépés előtt

Azon célból, hogy az elgondolt szoftvertermék tervét minél autentikusabb módon mutathassam be projektszemléletben, létrehoztam az alábbi két fiktív céget (a valósággal való bármilyen egyezés csak a véletlen műve lehet):

Intelligent Services Corporation: Egy fiatal, korábban marketingterületen dolgozó magyar cég átalakulásának eredményeként létrejött szolgáltató szervezet. Marketinges tapasztalataik alapján megbízhatónak mondható képük alakult ki a piaci igények és a piaci kínálat között mutatkozó hiányosságokról a szolgáltatási szférában.
Ötleteik megvalósításához kellő rugalmasságot nyújtó információ technológiai megoldásokat kívánnak használni, követve az aktuális trendeket. Korábban több olyan projekttel bíztak meg IT tanácsadó cégeket, melyek eredményeként létrejött egy alapvető képük a számukra szükséges technikai megoldások körvonalairól, s kapcsolatba kerültek az AIDi-Soft szoftverfejelsztő bázissal.

AIDi-Soft: Egy innovatív, viszonylag nagy hazai és külföldi tapasztalattal rendelkező, megbízható, többnyire magyar érdekeltségű szoftverfejleztő szerveződés. Fő profiljuk az ügyfélközpontú, intelligens hozzállás ötvözése magas technológiai szakértelemmel.

A projekt körülményeiről:

A *MessageForce* projekt a fenti két vállalat együttműködéséből született meg. Az Intelligent Services Corporation megbízásából az AIDi-Soft a *MessageForce* projekt menedzselését teljesíti. Biztosítja a szükséges erőforrásokat, melyet a projekt végrehajtása után az Intelligent Services Corporation megtérít. Mivel a megbízónak nincs 100%-osan kialakult képe a technikai megvalósításról és csak az alapvető üzleti elvárásokat ismerteti, a megbízott a szoftverterven és implementáción felül a részletes specifikációt is szolgáltatja, melyet az Intelligent Services Corporation jóvá kell hagyjon.
A leendő felhasználóbázist megtestesítő felhasználógárdát a megbízó nyújtja a specifikáció elkészítéséhez és az elkészült szoftver felhasználói tesztjéhez.
Tehát a fentiek értelmében a projekt lényegében az AIDi-Soft tulajdonában van, így a projektterv valójában ezen vállalat szemszögéből kerül létrehozásra, s az ő tevékenységüket írja le, kezdve a követelmények tisztításával, a specifikáció elkészítésén át a tervezésig és megvalósításig.
Dolgozatom csak a tervezést, mint konkrét tevékenységet foglalja magába, a fejlesztés, tesztelés és járulékos teendők csak a projektterv részét képezik, így azok végrehajtása nem történik meg.

IV. A tervezés menete

IV.1. Projektalapítás

Első lépésben a projekt megalapítását végeztem el. A projektalapító dokumentum magába foglalja a projekt tulajdonosait, résztvevőit (jogi személyekként), tisztázza a projektcélokat, megállapítja a költségeket, s tartalmazza a projekttervet. A projektterv elkészítésére gantt chartot használtam. Ahogy az a megvalósításból is látszik, igyekeztem az Agilis fejlesztés néhány alapvetését felhasználni, viszont mivel nincs sok tapasztalatom e területen, a tervet nem tejles mértékben agilis módon hoztam létre. Elképzeléseim között szerepel az Agilis fejlesztésekben elterjedt Velocity és Burndown chart alkalmazása a későbbiekben a Gantt rovására.
A projekttervnek szólnia kellene változás- és kockázatkezelési metódusokról, viszont tekintve a projekt méretét mind időben (néhány hónap, mely alatt számottevő külső és belső változás nem várható) és költségben (alacsony költségvetés, s a tárgyalások során a két fél kölcsönösen kedvező megállapodást kötött, mely minimalizálja az esetlegesen felmerülő veszteségeket), ezt a részt elhagytam.

IV.2. Projektterv

Megvalósítására Gantt diagramot használtam (kapcsolódó dokumentummok: *függelék 5., 6., 7.a, 7.b, 7.c)*. A terv magába foglalja az erőforrások reigsztrálását, s elosztását az egyes feladatok között. Mivel a dolgozat középpontjában a tervezés áll, a specifikációt, a különböző dokumentációs és a fejlesztéshez szükséges járulékos tevékenységeket (forráskód dokumentálása, szerveradminisztráció, jogosultságok biztosítása) valamint az ezeket végző munkatársakat mint erőforrásokat a diagramon nem tüntetem fel.
A projekt kezdetétől (2009.02.02) a gantt-charton látható első feljesztési ciklus kezdetéig (2009.04.10) a megbízó által biztosított felhasználó gárda segítségével a specifikáció

összeállításra kerül, a megbízott előkészíti a tervezéshez és fejlesztéshez szükséges környezetet.

A *task*ok közötti összefüggéseket és a mérföldköveket a Ganttproject szolgáltatásainak hiányosságai miatt nem tüntettem fel.

Ahogy az a diagramon látható, a projekt három viszonylag rövid fejlesztési ciklusból áll, egyenként egy-egy nagyobb mérföldkövet reprezentálva. A ciklusokban található gyűjtőtaskok (Tervezés, Fejlesztés, Tesztelés) egy-egy kisebb mérföldkövet testesítenek meg:

- Első fejlesztési ciklus: Az alkalmazás számára egyfajta platformot biztosító keretrendszer megtervezése, implementálása és tesztelése
- Második fejlesztési ciklus: Az ügyfelek és adminisztrátorok számára elérhető felületeket biztosító alkalmazások tervezése, implementálása és tesztelése, valamint az ezek számára keretet biztosító modul API létrehozása, mely a keretrendszer transzparens elérését biztosítja.
- Harmadik fejlesztési ciklus: Az e-mailek, mint üzenetek küldésére és fogadására alkalmas modul tervezése, implementálása és tesztelése

A diagramon látható időbeosztás közelíti a valós időket, figyelembe véve, hogy a tervezést a diagram szerint két architekt végezte, míg a valóságban egyedül dolgoztam a terveken.

A gantt diagram hiányosságaira világít rá, hogy az egyes főbb tevékenységek ciklikusságának hangsúlyozására nem nyújt megfelelő eszközöket.

A Ganttproject a projekt haladásának nyomonkövetésére egyetlen szolgáltatást biztosít: az egyes feladatok elkészültsége százalékértékekkel megadható. A költségek követésére sajnos szintén nem nyújt eszközöket. Ellenben ha a dolgozók fix munkaidőben dolgoznak, munkájuk mennyisége időben mérhető, így a költségek nagy részét mégis nyomonkövethetjük.

IV.3. Specifikáció

A projekt megalapozását a specifikáció elkészítése követte. Köszönhetően annak, hogy a specifikációt nem a megbízó szolgáltatja, a végrehajtónak lehetősége van arra, hogy a dokumentumot már jól struktúrált, a tervezést és fejlesztést nagyban támogató formában prezentálja (a valóságban sajnos ez gyakran nem teljesül és nagyrészt használhatatlan specifikációkat eredményez). Ezért a specifikáció több, egymásra épülő részből tevődik össze. Ezek sorrendben:

1. Missziós összefoglaló (*függelék 8.*): tömör megfogalmazása a követelményeknek. Kiindulópont az egyes szolgáltatások felfedezéséhez a követelmények alapján. A dokumentum 3 részből áll:
   - Név: az eredmény neve. Igen fontos összetevő, hiszen kézzelfoghatóvá teszi a képlékeny, elméletben is éppen csak létező entitást, vagyis a leendő szoftvert.
   - Cél (purpose): a szoftver rövid áttekintése, az elérni kívánt eredmény összefogalaló megfogalmazása. Nagyvonalakban felvázolja a leendő alkalmazást.
   - Üzleti elvárások (Requirements): a projekt alapításakor megfogalmazott követelmények tovább bontása, valamivel szakmaibb és részletesebb módon. Az itt vázolt követelményrendszer keretet képez a leendő szolgáltatásoknak.
   - Ami a hatáskörön kívül esik (exclusions): segítik pontosítani a rendszer határait (system boundary). Megadja azokat a lehetőségeket, melyeket a rendszer nem teljesít.
2. UseCase diagram (*függelék 9.*)*:* Mivel a felhasználóval való kommunikációt nagyban támogatja egyszerűsége és közérthetősége okán, a diagramot a specifikáció részeként használtam fel. A dokumentum segít átlátni az egyes felhasználók által igénybe vehető szolgáltatásokat. A diagram előállításához nagy segítséget nyújthat egy a szolgáltatásokat követelmények alapján felsoroló dokumentum. Bevett szokás, hogy a követelményeket tovább granuláljuk szolgáltatásokra, így egy faszerkezetbe rendezve azokat. A fa gyökerét a missziós cél adja, annak gyerek csomópontjait a követelmények, s a levélelemek pedig a

szolgáltatások, ahol egy-egy szolgáltatás valamely követelmény gyereke. Ezáltal nemcsak könnyebben fedezzük fel a megvalósítandó folyamatokat, hanem egy jól használható, a validációt nagyban támogató eszközt kapunk. A fejlesztés befejeztével a levélelemektől kiindulva könnyedén megállapítható, hogy mely szolgáltatások lettek implementálva, ezáltal megkapva, hogy mely követelményeknek tettünk eleget. Az egyes levélelemek pedig definiálják a UseCase diagramon feltüntetett használati eseteket. További lehetőségként adódik, hogy a kiépített fát felbonthatjuk részfákra, ahol a gyökerek az egyes követelmények lesznek. Az ilyen részfák nagyobb mérföldköveket reprezentálhatnak, melyek meghatározhatják, hogy milyen ciklusokat hozunk létre az alkalmazás fejlesztéséhez. Egy-egy részfa implementációjának elkészültekor az előállt alkalmazás akár felahasználói tesztelés alá is kerülhet, ezzel is idomulva az agilis fejlesztés alapvetéseihez.

3. Fogalomszótár (*függelék 10.*): A *funkcionális specifikáció* kiegészítéseként az alkalmazás működésének megértését segíti. A szoftver által végrehajtott üzleti folyamatok által érintett entitások, üzleti fogalmak meghatározásait tartalmazza. A dokumentum továbbá támogatja a közös munkát oly módon, hogy igyekszik egy közös nyelvet biztosítani a különböző szakterületek képviselői között.
4. Üzleti folyamatok (*függelék 11.*): Célja teljesen ugyanaz, mint a fogalomszótárnak, viszont középpontjában nem az entitások és fogalmak állnak, hanem az üzleti folyamatok.
5. Funkcionális specifikáció (*függelék 12.*): mind a felhasználók, mind a fejlesztők számára közérthető, (webalkalmazás lévén) képernyőorientált részletes összefoglalása a rendszer által nyújtott szolgáltatásoknak. Azáltal, hogy használja a *Fogalomszótár* és *Üzleti folyamatok* dokumentumokban definált fogalmakat és folyamatokat, könnyen áttekinthetővé válik, s mintegy interfészként az előbbi dokumentumok fölött, átfogó képet alkot a rendszer funkcionalitásáról.

IV.4. Szoftvertervezés

A tervezés során az agilis körökben is rendkívül elterjedt UML modellező nyelv leggyakrabban használt diagramjait alkalmaztam, úgy mint deployment, component, activity és class diagramokat. Igyekeztem a modellezés során az egyszerűségre törekedni, s az egyes dokumentumokat nem túlterhelni, ezzel is támogtva a könnyű olvashatóságot, áttekinthetőséget.

A tervezés lépései:

1. A szükséges osztályok, komponensek felfedezéséhez és egy kiinduló kép kialakításához a főbb üzleti folyamatok alapján elkészítettem néhány activity diagramot a fontosabb folyamatokról, melyeket a rendszernek meg kell valósítania. Ezek a *függelék 20., 21.* és *22.* pontja alatt találhatók.
2. Architektúra (*függelék 13.*): a rendszer főbb komponenseinek egymáshoz való viszonyát reprezentálja a függelékben látható deployment diagram. Természetesen ahhoz, hogy az egyes a modellen látható alrendszerek mibenléte és hatásköre tisztább megvilágításba kerüljön, ezen alrendszerek további granulálására van szükség. Így az architekturális tervezést több lépésben hajtottam végre, kibontva az egyes alrendszereket. Az ezáltal előállt diagramokat a *3. Komponenesek* pontban tárgyalom.
   A deployment diagram elemei:
   - Keretrendszer: az egész rendszer törzsét képezi. Fogadja és továbbítja a megfelelő protokollon az üzeneteket (*Üzenetkezelés*), karbantartja a rendszert elérő felhasználókat (*Felhasználókezelés*), regisztrálja és kezeli az egyes szolgáltatásokat megvalósító modulokat (*Modulkezelés*), valamint a felsorolt komponensek által kiváltott eseményeket rögzíti.
   - Szolgáltatások API: biztosítja az egyes alrendszerek közötti átjárást, sőt lehetőséget nyújt arra, hogy egy-egy alrendszeren belüli komponensek egymás szolgáltatásait transzparens módon érhessék el.

- Modul API: a rendszerben fellelhető modulok számára nyújt transzparens felületet a keretrendszer szolgáltatásainak eléréséhez. Egyfajta kapcsolattartóként is elképzelhető, mintegy entitásként, mely biztosítja a problémamentes kommunikációt a modulok és a keretrendszer között. Ezen fő tevékenységét a Szolgáltatások API által nyújtotta lehetőségek kibővítésével valósítja meg elsősorban.
- AdminInterface, ClientInterface, BaseMail: konkrét modulok, tevékenységük a *Fogalomtár*ban ismerhető meg (*függelék 10.*)

3. Komponensek:
   1. Keretrendszer komponensek (*függelék 14.*): A keretrendszer alrendszereinek komponenseit mutatja be. Az egyes komponensek működése:
      - Bejövő üzenetek: a beérkező üzenetek kezelése. Az üzenetek fogadása számára különböző sorokat (*Témakör*) tart fenn. Minden soron valahány modul regisztrálhat, mint feldolgozó modul. A sorokon megfigyelők (*Megfigyelők*) ülnek, melyek üzenet érkezésekor tájékoztatást küldenek a megfelelő, regisztrált moduloknak.
      - Kimenő üzenetek: a modulok által előállított üzenetek kiküldését végzi. Egy modul, miután egy elkészült üzenetet kiküldésre szán, elhelyezi azt a kimenő sorban. Ezen a kollekción szintén egy megfigyelő ül, mely feléleszti a kiküldést végző alrendszert, mely a sorban lévő üzeneteket továbbítja a megfelelő protokollon.
      - Üzenet adapter: a kimenő és beérkező üzeneteket protokollspecifikus konverterekkel a rendszer által transzparensen kezelhető formára hozza.
      - Eseménykezelés: az *Eseményregisztráció* komponens segítségével a rendszerben kiváltódott eseményeket ellátja a szükséges járulékos információkkal, s perzisztens módon tárolja azokat. Az *Eseményhalászat* komponens pedig a tárolt eseményeket paraméterezhető módon szelektálja s megjeleníthető formára alakítva továbbítja egy-egy hívó komponens számára.

- Modulkezelés: biztosítja, hogy futási időben bármikor egy-egy új modul regisztrálja magát a rendszerben, vagy egy meglévő kilépjen onnan (*Modulregisztráció*). A *Szolgáltatásmenedzser* biztosítja a modulok számára a keretrendszer szolgáltatásainak elérését (pl. üzenet fogadása, kiküldése, üzleti jellegű események regisztrálása stb.).
- Perzisztenciakezelés: a keretrendszer komponenseinek működéséhez szükséges perzisztens réteg biztosítása
- Felhasználókezelés: ahhoz, hogy a modulok transzparens módon kezelhessék az őket elérő felhasználókat, a keretrendszer felületet biztosít azok létrehozásához, módosításához (*Manipuláció*), authentikálásához (*Authentikáció*)
- Naplózás: A rendszer működésével kapcsolatos nem üzleti jellegű események naplózását végzi.

2. Egy modul komponensei általában (*függelék 15.*): Leírja általános formában egy modul komponenseit. A *Modul API* implementálásával minden modul biztosítja a saját működéséhez szükséges perzisztenciaréteget (*Adatkezelés*), statisztika megvalósításokat (*Statisztikák*), a rendszer felhasználójaként nem megjelenő (pl. különböző üzenetcímzettek, akik valójában a regisztrált *ügyfelek* kliensei) egyedek kezelését lehetővé tevő eszközöket (*Klienskezelés*), a modul és a keretrendszer komponensei közötti kommunikációt (*Szolgáltatásmenedzsment*), azokat az eszközöket, melyek a modul üzleti szolgáltatásait definiálják (*Modul üzleti szolgáltatások*), s a modul működésével kapcsolatos nem üzleti jellegű események naplózását. A *view* réteg biztosítja a webes megjelenítést, melyet a *Kontroll* kapcsol össze az alkalmazás réteggel, vagyis az üzleti folyamatokat megvalósító komponensekkel.

4. Osztályok: Az egyes komponensek struktúrájának kialakításához osztálydiagramokat használtam. A teljes kép kialakításához szükséges, ezáltal fontos komponensek jelentősebb interfészeit tüntetem fel a modelleken:

1. Üzenetkezelés (*függelék 16.*): A diagramon az alkalkmazás alapját képező üzenetkezelés struktúrája (*org.messageforce.messaging* csomag) és annak szűk környezete látható. Az *org.messageforce.history.Traceable* interfész támogatja az üzenetkezeléssel kapcsolatos üzleti szempontból fontos *esemény*ek regisztrálását. Az *org.messageforce.service* és az *org.messageforce.service.impl* csomagok az üzenetkezelő osztályok transzparens elérését teszi lehetővé az egész alkalmazás számára. Ezek a csomagok szinte az összes diagramon helyet kapnak, hiszen a rendszerben fellelhető minden eszköz ezek segítségével éri el a rendszer további elemeit.

   Ahogy az a diagramon látható, az *IncomingMessageHandler* és az *OutgoingMessageHandler* interfészek *Service*-ként publikálva vannak, tehát a teljes üzenetküldő és -fogadó mechanizmus ezen két belépési ponton érhető el. Az *IncomingMessageHandler* példányok kezelik a beérkező üzeneteket. Az *org.messageforce.protocol* eszközei által egy bármilyen protokollon keresztül érkező üzenetet a rendszer által kezelhető *Message* objektummá alakítja, így a modulok protokollfüggetlen módon képesek azokat feldolgozni. Természetesen a keretrendszert minden még nem ismert protokoll esetén ki kell egészíteni az adott protokollon közlekedő konkrét üzenet fogadására alkalmas eszközzel, például a *BaseMail* esetén egy mailszerverrel kommunikálni képes alkalmazással. Az *IncomingMessageHandler* miután előállította az új beérkező üzenetet, elhelyezi azt a *Topic*ban, mely eseményről az *Observer* minta mentén tervezett *TopicObserver* értesíti a megfelelő *MessageRepresentative* segítségével a kívánt modulokat.

   Szintén ezen csomagban találhatók az üzenetek kiküldését megvalósító eszközök. Egy-egy modul az *OutgoingMessageHandler Service* segítségével kimenő üzeneteket képes elhelyezni az *OutgoingMessageQueue* sorban. Ezen objektumon a *QueueListener* ül, mely új kiküldendő üzenet esetén felébreszti a *MessageSenderWorker* szálakat, melyek a sorban várakozó üzeneteket dolgozzák fel,

vagy küldik ki. A használni kívánt protokoll meghatározására és az üzenet tartalmának megadására a *Header* és *Body* kompozíciójából (az argoUML nem ismeri a kompozíció fogalmát interfészek között) álló *Message* megfelelő attribútumokat tartalmaz.

2. Protokolltámogatás (*függelék 17.*): az üzenetkezelést támogató protokollfüggő eszközrendszer APIja. A *MessageConverter* a *Builder* tervezési minta mentén felépített beérkező üzeneteket konvertáló objektum interfésze. Az *IncomingMessageHandler* az üzenet protokolljától függően meghívja a megfelelő *MessageConverter* objektumot, mely a *MessageBuilder* segítségével előállítja a beérkező üzenetet reprezentáló *Message* objektumot.
   A kimenő üzenetek inicializálása a *MessageFactory* segítségével történik, mely felkészíti kiküldésre az üzenetet a kívánt protokollnak megfelelő attribútumok beállításával. A *MessageSender* példányok szintén a kiküldést segítik oly módon, hogy adott protokollnak megfelelő módon továbbítják a *MessageSenderWorker* szálak által kiküldeni kívánt üzeneteket.
3. Modulkezelés (*függelék 18.*): a keretrendszer általi modulkezelést teszi lehetővé. A *ModulRegistry* biztosítja az ismert modulok tárolását és transzparens elérését. Minden egyes, a rendszerben ismert modult egy-egy *ModuleRepresentative* példány reprezentál, mintegy kapocsként a keretrendszer és a modul között. A *ModulHandler* segítségével az egyes moduloknak lehetőségük van regisztrálni vagy kilépni a rendszerből.
4. A modul-api legfontosabb elemei (*függelék 19.*): Szintén fontos szerepet kap az *org.messageforce.service* csomag, mint a modulok szolgáltatásait menedzselő eszközrendszer. Az API az előbbin kívül két fontos résszel bír. Az egyik az üzenetek fogadását és kiküldését támogatja (*org.messageforce.messaging.modules*) míg a másik a keretrendszerrel való dedikált kapcsolat létrehozását és bontását (*org.messageforce.modules*). Előbbiben a *MessagePublisher* példányok az üzenetek

keretrendszer felé publikálását, míg a *MessageProcessor* a beérkező üzenetek *Topic*ból való kiemelését és előfeldolgozását végzi.

5. Modul regisztráció (*függelék 23.*): Végül, az egyetlen nem felfedett folyamat, a modul regisztrációjának szekvencia diagramját készítettem el.

A fent felsorolt lépésekben egy alkalmazás tervének létrehozását igyekeztem reprezentálni. Az előállt modell viszont nem nevezhető teljesnek és validnak, hiszen ahhoz, hogy egy jól működő alkalmazás készüljön el egy terv alapján, jóval mélyebben fel kell fedni a működést és struktúrát, valamint a modell mindenképpen igényelné pilot alkalmazások elkészítést mintegy validálva a terveket. A pilot visszahatva a modellre tovább szofisztikálná azt, s felfedné annak hiányosságait, s az esetleges inkonzisztenciát.

Továbbá nem készítettem el a konkrét modulok modelljét, egyrészt azért, mert fő célom a működés alapjait biztosító keretrendszer megtervezése volt, másrészt pedig egy olyan részletekbe menő modell előállítása, mely az ily egyszerűséggel bíró alkalmazások tervét is magába foglalja nyilvánvalóan túlmutat ezen dokumentum keretein.

Kihangsúlyozandó, hogy a *ClientInterface*, az *AdminInterface* és a *BaseMail* valójában minimális applikációs réteggel rendelkező, elsősorban a megjelenítést megvalósító rendkívül kisméretű alkalmazások, hiszen szinte mindegyik a keretrendszer alapvető funkcióit használja, azokat nem kiegészítve komoly üzleti logikával. Ebből kifolyólag tervük túl implementációközeli lenne, ezért sem próbáltam azokat kifejteni.

V. Továbbfejlesztési lehetőségek

Mivel a terv egy nagyon magas szinten foglalkozik a megvalósítandó szolgáltatásokkal, igen sok bővítési lehetőséget garantál. Ha vetünk egy pillantást a deployment diagramra, látható, hogy eleve több, lazán kapcsolt, különálló alrendszerből áll. Ezek száma könnyedén növelhető, akár a *ServiceAPI* használatával, akár annak kihagyásával, kiegészítésével vagy esetleg a lecserélésével. Továbbá minden egyes alrendszer komponensei kiegészíthető újakkal, már terv a terv szintjén is. Az is nyilvánvaló, hogy a keretrendszer még tovább granulálható, felbontva olyan, szintén kiegészíthető részegységekre, melyek együttes egészként nagyon sokrétű szolgáltatásokkal bíró rendszert alkothatnak.

Megjegyzendő továbbá – mint ahogy azt a szemfüles olvasó már talán fel is ismerte – a rendszer valójában semmi olyan elképzelést nem vonultat föl, mely a mai szakmai gyakorlatban ne lenne megvalósítva közel olyan módon, ahogy azt a model mutatja. Ezeket sorra véve (mivel Java technológiákban mozgok otthonosan, elsősorban a JavaEE eszközeit hoznám fel alternatívaként):

- Keretrendszer: a keretrendszer maga egy nagyon egyszerű *ESB (Enterprise Service Bus)* megvalósítása lehet. Egy ESB körenyezetet biztosít különböző modulok számára, melyek SOAP mentén érkező kéréseket dolgoznak föl. Az ESB fogadja a kivülről érkező SOAP üzeneteket, s konfigurációjának megfelelően továbbítja azt egy vagy több modul számára valamilyen *queuing* rendszer segítségével. Gyakori, hogy a sor perzisztálását is lehetővé teszi, ezzel biztosítva a rendszer hibatűrését: időnként a sor s ezáltal a tartalma mentésre kerül, s ha bármilyen okból kifolyólag a rendszer leállásra kényszerül, a sorok tartalma nem vész el, vagy legalábbis visszaállítható egy konzisztens állapotra. A *MessageForce* jelen állapotban ilyen szolgáltatással nem rendelkezik. A JavaEE világában ilyen eszközrendszert nyújt a JBossESB vagy a Mule.
- Üzenetkezelés: egyértelmű alternatíva a JMS. A topic ötletét eleve a JMS szintén topicnak nevezett eszköze alapján terveztem. A JMS üzenetek különféle tárolását és feldolgozását támogató Java Enterprise eszköz.

- Modul: tipikus EJB megfelelő. Minden egyes modul körülbelül megtestesít egy *Message-driven bean*t. Ha megfigyeljük a *MessageForce* rendszer működését, tökéletesen ráillik az alábbi modellre: a rendszer moduljai, mint EJB-k, pontosabban *Message Driven bean*ek regisztrálnak a keretrendszerben, vagyis egy ESB-ben (itt találhatunk viszont egy különbséget, mégpedig azt, hogy az ESB-k használatához eleve konfigurálni szükséges az egyes modulokat, tehát fordítási időben dől el, hogy milyen modulok milyen *topic*okon várják a beérkező üzeneteket) valamilyen, az ESB által deklarált soron (vagy *topic*on), ahol a *topic*ot JMS *topic*ok valósítják meg, s az ESB egyes *bean*jei webszolgáltatásként publikálásra kerülnek, s ettől a pillanattól kezdve nyilvánvaló, hogy a webszolgáltatásnak akár felhasználói interfésze is lehet, tehát a *MessageForce* minden funkciója biztosított már meglévő, s remélhetőleg jól megtervezett, kipróbált s valószínűsíthetően hatékonyan implementált eszközökkel.
- Üzenet (*Message*): tipikus XML. Sztandard módon, a rendszer által transzparensen kezelhető formában különböző protokollok mentén közlekedő üzenet csomagolására alkalmas eszköz. Minden olyan követelmény, melyet a *MessageForce* alkalmazás megfogalmaz, az XML teljesíti. Mint azt már az keretrendszer kapcsán említettem, az ESB-k elsősorban SOAP köré lettek építve, mivel az XML széles körben elterjedt sztandard eszköz, tehát a *MessageForce* keretrendszere akár átalakítható is lenne oly módon, hogy a keretrendszertől teljes mértékben függetlenül állnak elő SOAP kéréseket megtestesítő XML állományok, s a rendszert csak és kizárólag XML feldolgozásra kellene alkalmassá tenni. Persze jelen formájában a keretrendszer bővíthető különböző konverziós eszközzel, viszont ezáltal sokkal nagyobb felelősség is hárul rá, hiszen a legkülönfélébb protokollokat is ismernie kell.
- *MessageForce:* egy SOA mentén felépített alkalmazás. Kifelé a lehető legkülönfélébb protokollok mentén látható, szolgáltatásokkal tetszőlegesen kibővíthető rendszer. Azáltal pedig, hogy bármilyen protokoll támogatására alkalmas, kiegészíthető akár SOAP interfészekkel is, biztosítva a manapság favorizált kommunikációs módot. Továbbá

bármely modul implementálható webszolgáltatásként, vagy kiegészíthető webszolgáltatásnak megfelelő interfészekkel.

VI. Implementációs kérdések

Annak ellenére, hogy az előzőekben felvonultattam a *MessageForce* alkalmazásba tervezett minden eszközre egy-egy alternatívát, létjogosultságát látom egy olyan rendszer létrehozásának, mely a fenti lehetőségeket mégsem használja ki. Miért gondolom ezt így? Mert elképzelhetőnek tartom, hogy a *MessageForce*-szal szemben jelen pillanatban vízionált nagyon sokrétű szolgáltatásnak csak egy részhalmazára van szükség. Tehát sokkal specifikusabb, szűkebb funkcionalitással rendelkező alkalmazást szeretnénk eredményül kapni. Ezt a létrehozott modell egyes komponenseinek a kidobásával, vagy kevésbé sok irányba támogató implementálásával könnyedén elérhetjük, míg a JMS, vagy egy JBossESB esetleg az Enterprise Java Beans rendszerekkel ezt nem tehetjük meg, vagy legalábbis sokkal nagyobb idő ráfordításával, s végrehajtásához sokkal mélyebb és tágabb szaktudásra is szükség van. Ennek ellenére természetesen dönthetünk úgy, hogy a fenti nehézsúlyú “bajnok”-okat alkalmazzuk, viszont egyes esetekben (kis projekteknél túlnyomórészt) ez a megfontolás úgy tűnhet, mintha kombájnnal próbálnánk learatni a hátsó kiskertben hobbiból 2 négyzetméteren termesztett néhány szál kukoricát. Ami végülis megoldható, de sokkal több erőforrást igényel.
Éppen ezért az implementációs lehetőségeket két irányban közelíteném meg (természetesen Java eszközökön keresztül):

- A könnyűsúlyú: a *MessageForce* alkalmazást a terveknek megfelelően saját eszközökkel valósítjuk meg. Nyerjük ezzel azt, hogy könnyűsúlyú szervert kell alkalmaznunk, sőt elegendő egy Tomcat, vagy Jetty, mint szervlet konténer. Nem kell fölöslegesen *entity container*-eket futtatnunk, s fenntartani mindazt a hatalmas eszközrendszert, mely együtt

jár például egy JBossESB alkalmazásával. A probléma viszont, hogy valamivel többet kell dolgozni, hiszen már meglévő eszközöket kell saját elképzeléseink szerint implementálni. Ez magában hordozza azt a lehetőséget, hogy rengeteg hibát vétve egy rosszul működő rendszert hozunk létre.

Egy lehetséges felépítés: mikor terveztem az alkalmazást, félig-meddig szem előtt tartottam a következő képet az alkalmazásról. Tomcat 6.0 szervlet konténerben futtatott webalkalmazás. A Tomcatet ismerve arra jutottam, hogy a keretrendszer valójában egybe lenne forrasztva a konténerrel, ezáltal a Tomcatben futtatott összes webalkalmazás eléri a keretrendszer szolgáltatásait. Nyilván a *ServiceAPI*, a *UserManagement* szintén a Tomcat szintjén lenne futtatva. Az összes modul pedig, beleértve a *ClientInterface*-t és az *AdminInterface*-t, önálló webalkalmazásként futna.

A perzisztenciaréteg támogatására személy szerint *Hibernate 3*-at használnék, viszont semmi akadálya nincs annak, hogy valamilyen könnyűsúlyú (akár saját fejlesztésű) *JDBC framework*-öt, esetleg csak egyszerűen JDBC-t használjunk. Adatbáziskezelő rendszerként szívesen használnék Postgresqlt, de nincs akadálya egyéb vendornak sem (pl. MySQL, Oracle vagy Apache Derby). Postgresql és Oracle mellett szól, hogy sok lehetőséget biztosít az üzleti folyamatok adatbázishoz közeli megvalósításához (PL/pgSQL vagy PL/SQL, esetleg C, Java). Az Oracle ellen szól, hogy nincs ingyen. A *ServiceAPI* implementálásához igen hasznos platformot biztosíthat a Java SPI (*Service Provider Interface*).

A nézet oldalon szintén rengeteg választási lehetőségünk van. Személy szerint Struts2-t használnék elsősorban, viszont létjogosultsága van a JSF-nek és mindenféle kiterjesztésének (pl. MyFaces, ADF), vagy Springnek, GWTnek, de minden további nélkül dolgozhatunk alapvető Servlet és JSP technológiákkal. Mindez megtámogatható Ajax4JSF JavaScript kiterjesztéssel JSF esetén, vagy Jquery esetleg Dojo, vagy plain JavaScript megoldásokkal, alkalmazkodva ily módon is a mai trendekhez. Elképzeléseimben egy olyan kép is él, melyben a *ClientInterface* önmagában felületet

biztosít a különböző modulokhoz. Ennek a megvalósításához kézenfekvő megoldásnak tűnik valamilyen portál alkalmazása (például BEA Weblogic Portal, Jetspeed2 vagy Liferay), viszont megvalósítható sztandard HTML eszközökön keresztül is (iframe, újabban object, bár utóbbi a böngészők által még nem kifejezetten támogatott, legalábbis olyan aspektusban, amilyenre szükség lenne jelen esetben). Utóbbi lehetőséget választva probléma lehet a bejelentkezett felhasználó átmigrálása egyik alkalmazásból a másikba (hiszen a felhasználó bejelentkezett a *ClientInterface*-be, ami egy önálló webalkalmazás, aztán szeretne kapcsolódni egy modulhoz, ami szintén egy önálló webalkalmazás). Ez a probléma könnyedén feloldható a Tomcat SingleSignOn authentikációs lehetőségével, vagy megtámogathatjuk a keretrendszerből valamilyen session szinkronizációs eszközzel.

Itt hívnám fel a figyelmet egy a terven nem reprezentált problémára: authorizáció. Miszerint mely szolgáltatás, milyen szerepkörrel rendelkező felhasználó által érhető el. Ha elképzeléseimet tekintjük, a modell mégsem nevezhető hiányosnak ebből a szempontból, hiszen ha például Struts2-ben gondolkodunk, a szolgáltatások belépési pontjai (*struts action*-ök) névterekre bonthatók. Ha szinkronizáljuk a *struts* névtereket és a szerepköröket (adott belépési pontok adott, felhasználó szerepköréhez kötött névterekben érhetők el), a *struts2 framework* alapvető szolgáltatásai által tudjuk megakadályozni az egyes szolgáltatásokhoz való illetéktelen hozzáférést. S mindez egyszerű konfigurációs úton (*struts.xml*) elérhető. Amennyiben ez mégsem megfelelő megoldás, minden további nélkül kiegészíthető a keretrendszer egy authorizációs modullal, mely a szolgáltatásokat kellően bezárja. Esetleg használhatjuk a *JAAS* nyújtotta lehetőségeket.

- A nehézsúlyú: az összes elérhető JavaEE technológiát felvonultatjuk. Maradva a már korábban említett JBossESB környezetben, SOAP mentén kommunikált XML dokumentumokat mozgatva és feldolgozva implementáljuk a szolgáltatásokat. Az üzenetek fogadására *webszolgáltatás end-point*okat definiálunk JMS sorokon vagy *topic*okon, melyeken *MessageDriven EJB bean*ek várakoznak. Authentikációt és

Authorizációt végzünk *JAAS* eszközökkel, tranzakciókezelést biztosítunk a *JTA* nyújtotta lehetőségeket kihasználva. A minimum, hogy Hibernate-et használunk perzisztenciakezelőként, például egy Oracle adatbázisra építve. Megjelenítésre WSRP és hagyományos JSR-168 portleteket alkalmazunk, egy Liferay vagy egy BEA-Weblogic portál keretein belül. A portleteket továbbá megtámogathatjuk Springgel, melyet kombinálhatunk JSF (vagy valamely kiterjesztése) esetleg Struts2 keretrendszerekkel. Ezutóbbi, nehézsúlyú megoldással egy robusztus, bővíthető, hibatűrő, teljes mértékben lazán kapcsolt eszközökből álló alkalmazás architektúrát hozhatunk létre, mely hibatűrése mellett innovatív, a trendeket követő, és mindenképpen eladható eredmény. Valószínűleg sokba is kerül, továbbá igen erőforrásigényes, viszont a mai elvárásokat figyelembe véve ezen *trade-off*-ok bőven beleférhetnek a szabott keretekbe.

Összefoglalás

Dolgozatom tárgyát képező tervezési folyamat végrehajtása mind jelen, s korábbi munkáim alapján rendkívül összetett, odafigyelést és kitartást igénylő feladat. A problémák már eleve a használható munkaeszközök kiválasztásánál felmerülnek. Mivel fontosnak tartom az open source világ eredményeit a lehető legtöbb esetben alkalmazni, mind hitvallás és filozófia, mind az elérhető eszközök sokszínűsége okán, sokszor szembesülök is annak hiányosságaival. Az elérhető UML modellező szoftverek sokszor igen korlátos szolgáltatásokkal rendelkeznek, mindez bugokkal párosulva olykor rendkívüli hátráltató tényezőt jelent. Ugyanez jellemző a projektmenedzsmentet támogató eszközökre is. Természetesen ez nem jelenti azt, hogy temetném az open source világ vívmányait, mindössze hibáira hívnám fel a figyelmet.

További, szintén gyakorlati probléma, hogy a jelenlegi hardveres ellátás kissé gátat szab az UML modellezés néhány alapvetése követésének. Mivel törekednünk kell a modellek egyszerűségére és áttekinthetőségére, a bonyolultság az egyes modellek struktúrálásával érhető el, tehát egyik modell egy-egy eleme egy-egy másik modellen kerül kifejtésre. Ebből kifolyólag nem nehéz arra a következtetésre jutni, hogy egy idő után nehézkessé válig a struktúra követése egy egyszerű monitoron, annak fizikai korlátai miatt.

A fenti gyakorlati problémák mellett tisztán látható, hogy mivel két olyan tág és összetett témakört igyekeztem e dolgozatban összefoglalni, mely a mai napig nem eléggé kiforrott, folyamatosan fejlődik, s a szakma legnagyobb képviselői számára is kihívást jelenthet, a dolgozatban kialakított kép korán sem teljes, s hibátlan, s részletesnek végképp nem mondható. Viszont tekintettel arra, hogy nem rendelkezem sokrétű és mély tapasztalattal a szoftverfejlesztés és a projektmenedzsment területén, valamint e diplomamunka hatásköre is korlátot szab, ezt a tényt elfogadhatónak találom.Továbbá igen sok terület, melyet csak egy rövidítés megemlítése erejéig, vagy néhány mondatban, esetleg valamivel bővebben érintettem, önállóan is kitehet egy egész szakdolgozatot, diplomamunkát, vagy akár egy PhD dolgozatot.

Mindezt figyelembe véve kijelenthetem, hogy hatalmas tudásbázisra és tapasztalatra van szükség ahhoz, hogy bármely két fenti területen sikereket érjünk el. Szándékaim szerint a továbbiakban követni kívánom az általam leírt metodológiát folyamatosan javítva esetleg egyes elemeit lecserélve és/vagy kiegészítve a szakma neves képviselői által megosztott tudással és ismeretekkel. Úgy gondolom, hogy ezt a filozófiát követve előbb vagy utóbb egy jól használható, konzisztens folyamata alakítható ki egy jó szoftver kifejlesztésének a projekt megalapításától az eredmény átadásán és működtetésén át a szoftver végérvényes törléséig. Természetesen e folyamat kialakítása egy igen hosszú út bejárását kívánja meg, de úgy gondolom, hogy az első lépéseket megtettem, s úgy vélem, jófelé haladok.

Irodalomjegyzék

[1] The Standish Group – The Chaos Report (1994):
http://standishgroup.com/sample_research/chaos_1994_1.php

[2] Dave Nicolette – Chaos report under scrutiny (blogbejegyzés - 2006):
http://dnicolet1.tripod.com/agile/index.blog?entry_id=1537094

[3] Deborah Hartmann – Standish CHAOS Report Methods Questioned (2006):
http://www.infoq.com/news/Standish-Chaos-Report-Questioned

[4] Deborah Hartmann – Interview: Jim Johnson of the Standish Group (2006):
http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS

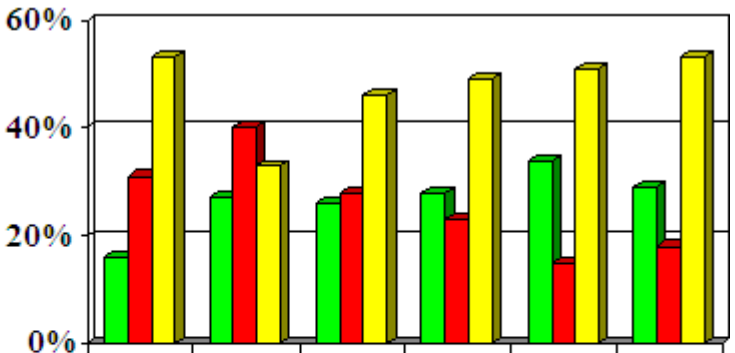
[5] Jim Johnson – My Life is Failure (Standish Group International, 2006)

[6] Agile Modeling
http://www.agilemodeling.com/

[7] Erich Gamma – Design Patterns (Addison Wesley, 2007)

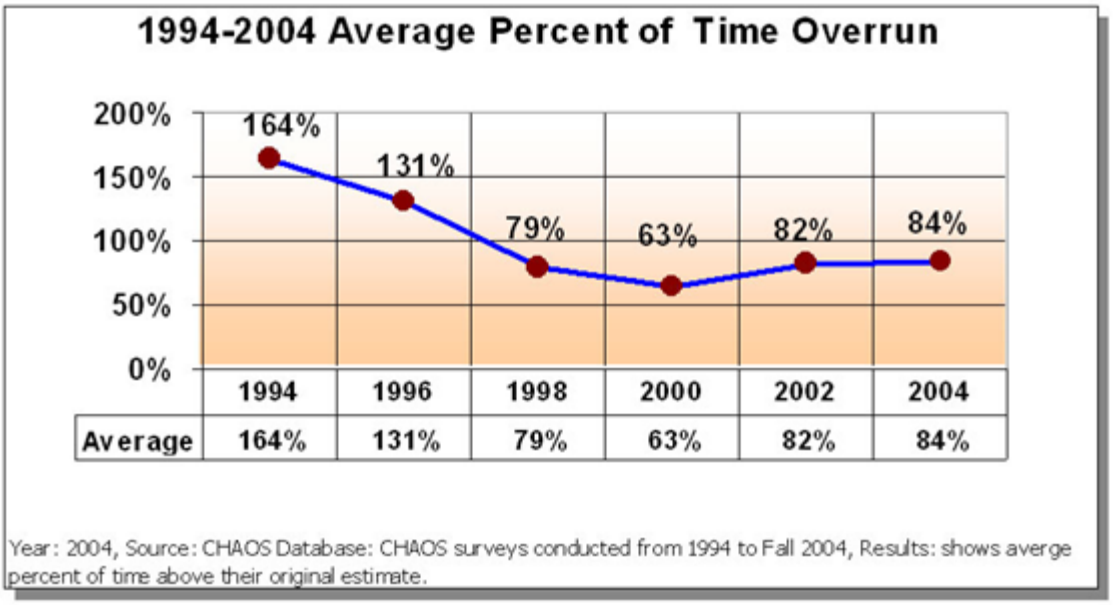
Függelék

1. IT projektek sikerességi rátája 1994 és 2004 között

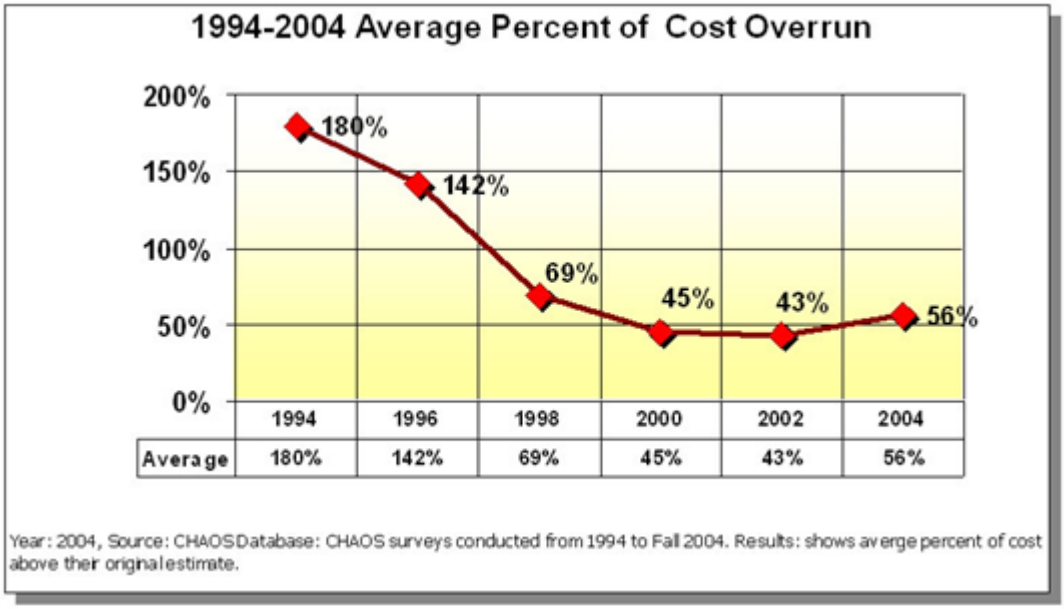


	1994	1996	1998	2000	2002	2004
Succeeded	16%	27%	26%	28%	34%	29%
Failed	31%	40%	28%	23%	15%	18%
Challenged	53%	33%	46%	49%	51%	53%

2. IT projektek időtúllépési rátája 1994 és 2004 között



3. IT projektek költségtúllépési rátája 1994 és 2004 között



4. Projektalapító dokumentum

Megbízó: Intelligent Services Corporation

Megbízott: AIDi-Soft

Rövid leírás: Pilot szoftver készítése melynek kiterjesztésével egy olyan szoftverrendszer állítható elő, mely magasfokú skálázhatóság és bővíthetőség mellett biztosít felületet a legkülönfélébb, felhasználói szemszögből vonzó szolgáltatások nyújtására.

Célkitűzések:

- Korszerű, felhasználóbarát, letisztult megjelenés, mely könnyen elsajátítható kezelést biztosít a kevésbé kimagasló informatikai ismeretekkel rendelkező ügyfelek számára is
- Az alkalmazás integrálható legyen az Intelligent Services Corporation által fenntartott szoftverrendszerekbe
- Az alkalmazás lehetőséget biztosítson a szolgáltatások gyors átalakítására a piaci igények változásának megfelelően

A projekt kezdete: 2009. 02.02.

A projekt vége: 2009.07.03.

Projektszervezet:

- Projekt tulajdonosa: Symore (AIDi-Soft)
- Külső menedzser: Xaero (Intelligent Services Corporation)
- Projektmenedzser: Symore (AIDi-Soft)
- Fejlesztő csapat:
    - Team-lead: Sarge
    - Architekt: Uriel, Biker
    - Arculattervező: Phobos
    - Grafikus: Anarki
    - Webfejlesztő: Crash
    - Fejlesztők:

    - Klesk
    - Slash
    - Grunt
    - Ranger
- Tesztelő csapat: Megbízó által biztosított felhasználók

Projektterv: Gantt-chart (*függelék x.*)

5. Végrehajtandó feladatok listájából, s a hozzárendelt erőforrások (részlet)

Tasks List

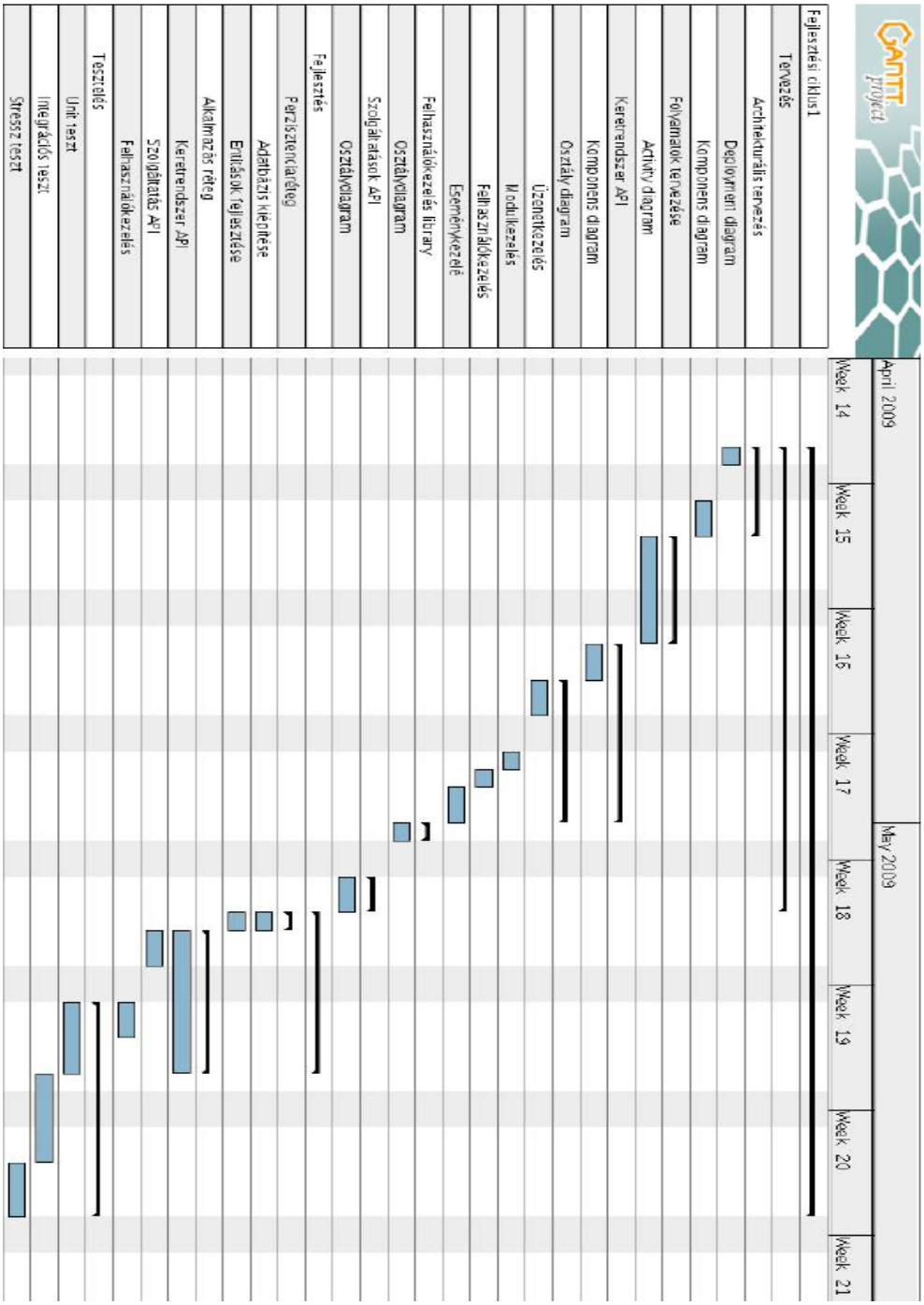
Name	Start	End	Milesto	%	Resources	Notes
Fejlesztési ciklus1	4/10/09	5/23/09	false	0		
Tervezés	4/10/09	5/6/09	false	0		
Architekturális tervezés	4/10/09	4/15/09	false	0		
Deployment diagram	4/10/09	4/11/09	false	0	Uriel Biker	
Komponens diagram	4/13/09	4/15/09	false	0	Uriel Biker	
Folyamatok tervezése	4/15/09	4/21/09	false	0		
Activity diagram	4/15/09	4/21/09	false	0	Uriel Biker	
Keretrendszer API	4/21/09	5/1/09	false	0		
Komponens diagram	4/21/09	4/23/09	false	0	Uriel Biker	
Osztály diagram	4/23/09	5/1/09	false	0		
Üzenetkezelés	4/23/09	4/25/09	false	0	Uriel Biker	
Modulkezelés	4/27/09	4/28/09	false	0	Uriel Biker	
Felhasználókezelés	4/28/09	4/29/09	false	0	Uriel Biker	
Eseménykezelé	4/29/09	5/1/09	false	0	Uriel Biker	
Felhasználókezelés	5/1/09	5/2/09	false	0		
library						
Osztálydiagram	5/1/09	5/2/09	false	0	Uriel Biker	
Szolgáltatások API	5/4/09	5/6/09	false	0		
Osztálydiagram	5/4/09	5/6/09	false	0	Uriel Biker	
Fejlesztés	5/6/09	5/15/09	false	0		
Perzisztenciaréteg	5/6/09	5/7/09	false	0		
Adatbázis kiépítése	5/6/09	5/7/09	false	0	Klesk Crash	
Entitások fejlesztése	5/6/09	5/7/09	false	0	Klesk Crash	
Alkalmazás réteg	5/7/09	5/15/09	false	0		
Keretrendszer API	5/7/09	5/15/09	false	0	Klesk Crash	
Szolgáltatás API	5/7/09	5/9/09	false	0	Slash Grunt	
Felhasználókezelés	5/11/09	5/13/09	false	0	Grunt	
Tesztelés	5/11/09	5/23/09	false	0		
Unit teszt	5/11/09	5/15/09	false	0	Slash	
Integrációs teszt	5/15/09	5/20/09	false	0	Slash Grunt	
Stressz teszt	5/20/09	5/23/09	false	0	Klesk Crash	
Fejlesztési ciklus2	5/20/09	6/10/09	false	0		
Tervezés	5/20/09	5/29/09	false	0		
Modul API	5/20/09	5/23/09	false	0		
Komponensdiagram	5/20/09	5/21/09	false	0	Uriel Biker	
Osztálydiagram	5/21/09	5/23/09	false	0	Uriel Biker	
AdminInterface	5/25/09	5/26/09	false	0		
Osztálydiagram	5/25/09	5/26/09	false	0	Uriel Biker	
ClientInterface	5/26/09	5/27/09	false	0		
Osztálydiagram	5/26/09	5/27/09	false	0	Uriel Biker	
Felületterv	5/20/09	5/29/09	false	0	Phobos	
Fejlesztés	5/27/09	6/6/09	false	0		
Perzisztenciaréteg	5/27/09	5/28/09	false	0		
AdminInterface	5/27/09	5/28/09	false	0	Crash	

6. Felhasznált erőforrások és szerepkörük

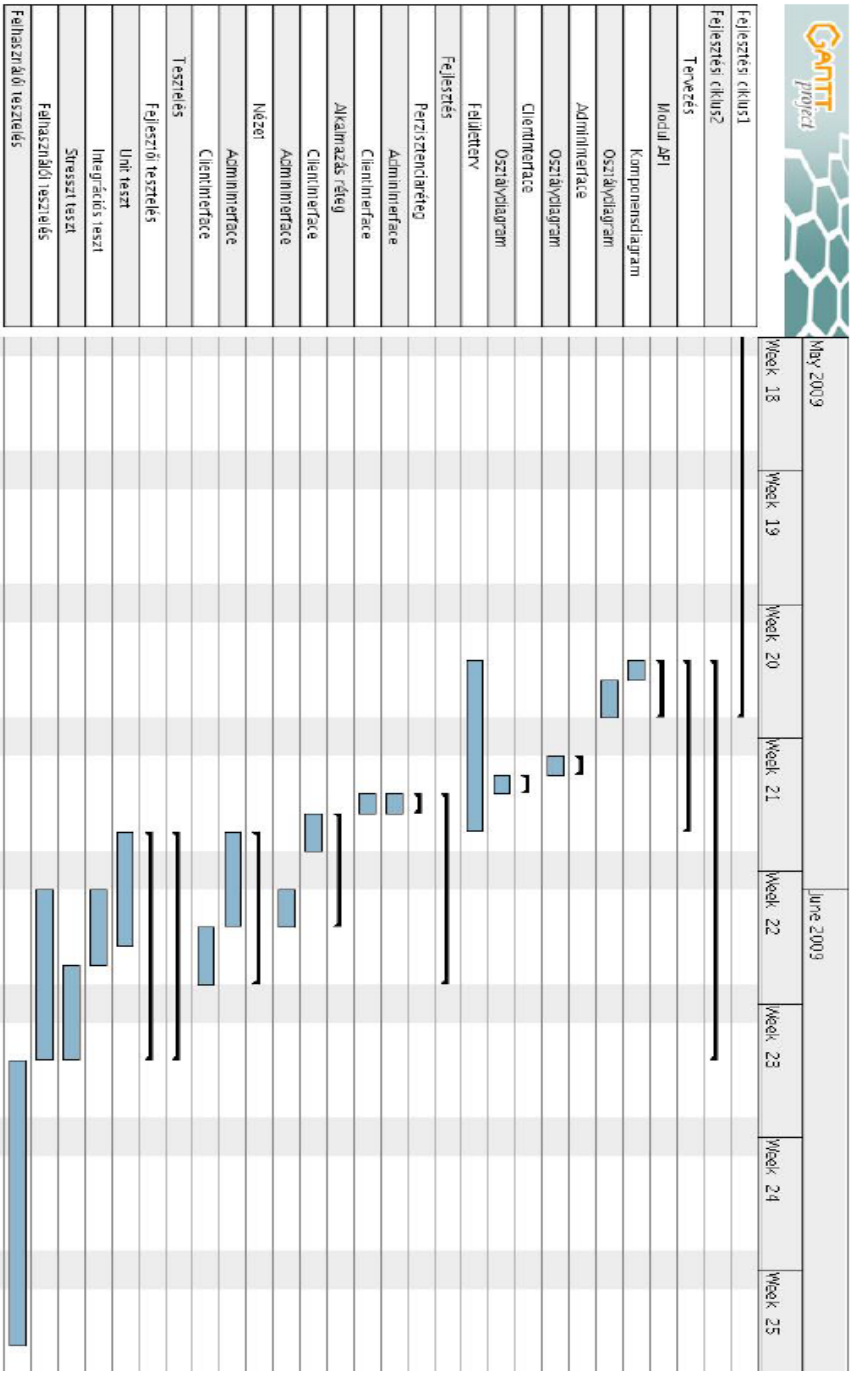
Resources List

Name	Default role	Mail	Phone
Name:Uriel	analysis		
Name:Biker	analysis		
Name:Phobos	graphic designer		
Name:Anarki	web designer		
Name:Crash	developer		
Name:Klesk	developer		
Name:Slash	developer		
Name:Grunt	developer		
Name:Ranger	developer		
Name:Külső tesztelők	tester		

7.a Gantt diagram (első fejlesztési ciklus)



7.b Gantt diagram (második fejlesztési ciklus)



7.c Gantt diagram (harmadik fejlesztési ciklus)

8. Missziós összefoglaló

Név: MessageForce

Cél: olyan webes interfésszel rendelkező több felhasználós, üzenetközpontú, moduláris, skálázható és bővíthető szoftverrendszer implementálása, mely különböző protokollok mentén közlekedő üzenetek, üzenetcsomagok fogadására, feldolgozására és továbbítására alkalmas

Üzleti elvárások:

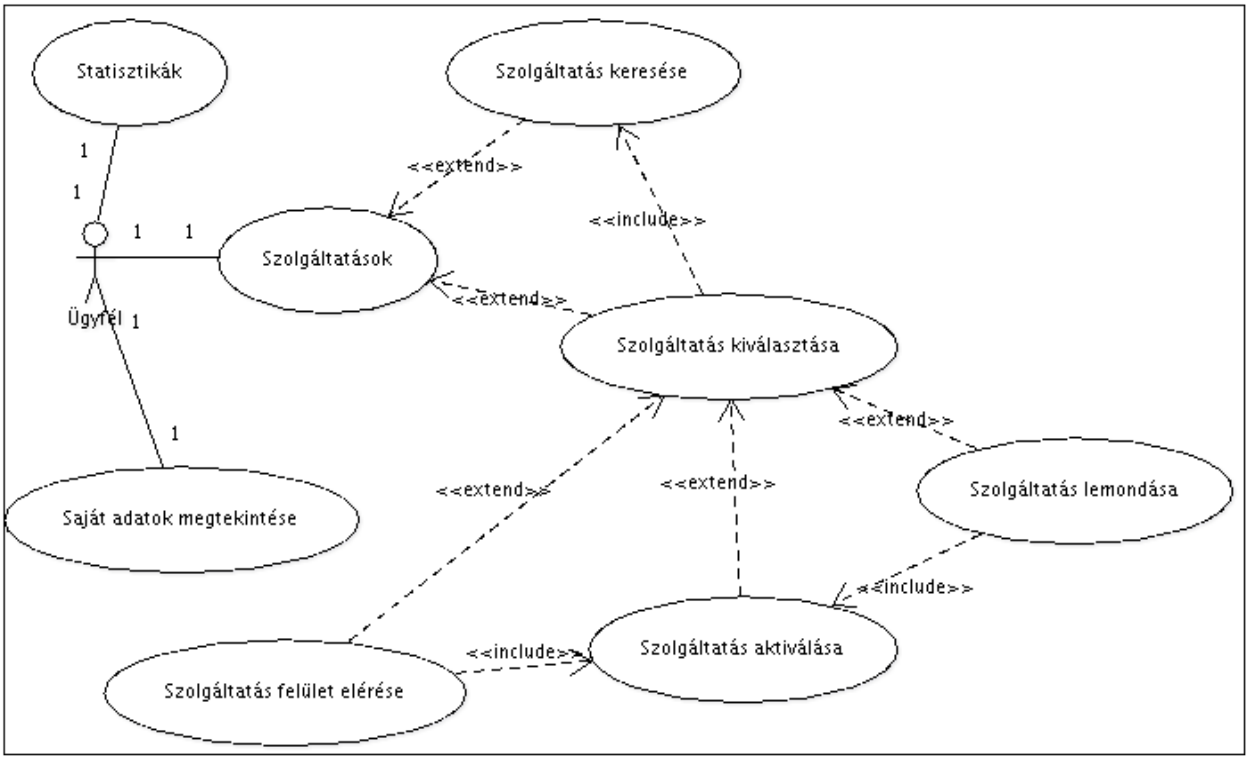
- Korszerű, felhasználóbarát webes interfész
- Különböző szerepkörrel rendelkező adminisztrátorok és ügyfelek támogatása (felhasználómenedzsment)
- Üzenetek fogadása és továbbítása
- Paraméterezhető statisztikák biztosítása az üzenetek forgalmáról
- Események (felhasználói tevékenység, üzenetforgalom, rendszer működése) regisztrálása, az események közötti keresés biztosítása az adminisztráció számára
- Folyamatos funkcióbővítési lehetőség biztosítása
- Egy darab, az alapvető működést (felhasználókezelés, üzenet fogadás és továbbítás) kiegészítő feldolgozó modul készítése, mely lehetővé teszi a felhasználók számára egyszerű üzenetek (e-mail) fogadását, előállítását és továbbítását
- Adatbiztonság

Ami a hatáskörön kívül esik:

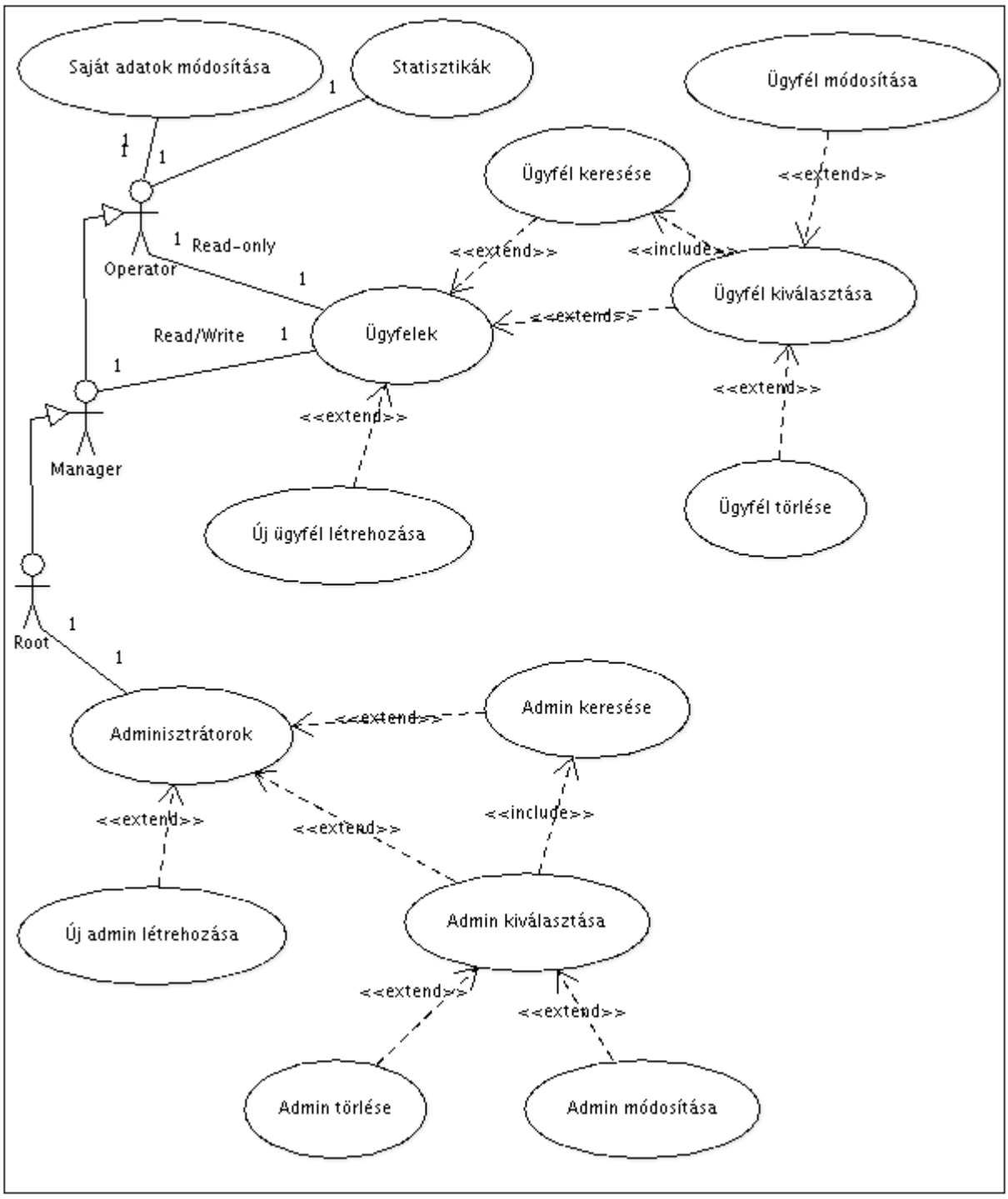
- Adatok archiválása
- E-mailen kívüli további protokollok támogatása
- Összetett üzenetfeldolgozó modulok készítése az egyszerű fogadást, előállítást és továbbítást végző modulon kívül.
- Felhasználói hibából eredő adatvesztés és/vagy károsodás elleni védelem

9. UseCase

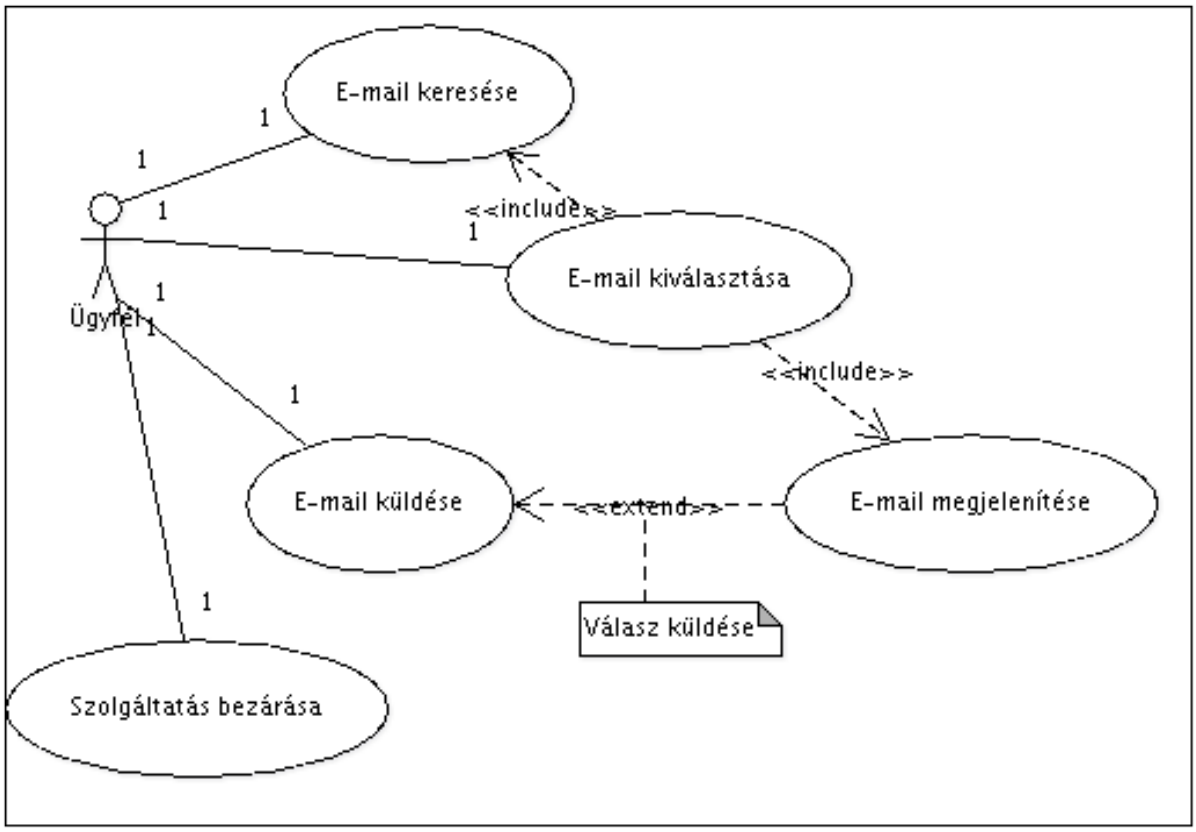
9.a Ügyfél által elérhető felület funkciói:



9.b Adminisztrátorok által elérhető felület funkciói:



9.c A BaseMail modul szolgáltatásai:



10. Fogalomszótár:

Adminisztrátor felhasználó: A rendszerben adminisztratív feladatokat ellátó felhasználó. Egy adminisztrátor egy időben pontosan egy szerepkörrel rendelkezhet. A rendszerben az alábbi szerepkörök léteznek:

- Operator: az ügyfelekkel és üzenetekkel kapcsolatos információkat tekintheti meg. Csak olvasási jogokkal bír.
- Manager: minden operátori joggal bír, valamint az ügyfelek karbantartását végzi
- Root: minden menedzseri jogosultsággal bír, továbbá az adminisztrátor felhasználók karbantartását végzi. A rendszerben telepítés után egy alapértelmezett Root felhasználó létezik.

BaseMail: egy olyan, az *ügyfel*ek által elérhető szolgáltatást megtestesítő *modul*, mely beérkező e-maileket (mint *üzenet*eket) képes feldolgozni (tárolni és megjeleníteni), valamint új e-mail *üzenet*eket képes előállítani, s kiküldeni a *keretrendszer* szolgáltatásain keresztül.

ClientInterface: egy olyan *modul*, mely globális webes interfészt nyújt az *ügyfel*ek számára a rendszer összes szolgáltatásának az eléréséhez. Lehetőséget biztosít az egyes szolgáltatásokkal kapcsolatos statisztikák paraméterezésére, futtatására és megjelenítésére, az *ügyfél* a *modul* segítségével karbantarthatja saját adatait, a rendszerben elérhető szolgáltatásokat megrendelheti vagy lemondhatja.

AdminInterface: egy olyan *modul*, mely globális webes interfészt nyújt az *adminisztrátor*ok számára feladataik ellátásához. Lehetőséget biztosít az egyes szolgáltatásokkal és ügyfelekkel kapcsolatos statisztikák paraméterezésére, futtatására és megjelenítésére, az *adminisztrátor* a *modul* segítségével karbantarthatja saját és az *ügyfel*ek adatait, a rendszerben kiváltódott eseményeket megjelenítheti.

E-mail: *üzenet*

Esemény: olyan entitás, mely a különböző felhasználói tevékenységek és az üzenetforgalom körülményeit rögzíti

Felhasználó: azon személyek, akik a rendszerben regisztrálásra kerültek. A rendszer *adminisztrátor* és *ügyfél* felhasználókat különböztet meg.

Keretrendszer: a rendszer alapját képező alrendszer. Az *üzenetek* fogadását és továbbítását végzi, különböző szolgáltatásokat nyújt a *modul*ok számára. Biztosítja az alapvető felhasználókezelést, transzparens felületet nyult a modulok számára az események kezelésére.

Modul: olyan alrendszer, mely a *keretrendszer* által támogatott protokoll mentén közlekedő üzenetek feldolgozására alkalmas. Egy-egy, a felhasználók számára elérhető *szolgáltatás*t testesít meg.

Normál üzenet: olyan üzenet, melyet egy ügyfél valamely, a rendelkezésére álló modul segítségével hoz létre.

Rendszerüzenet: a rendszer által előállított automatikus üzenet.

Szolgáltatás: *modul*

Ügyfél: olyan felhasználó, aki *regisztráció* után a rendszer által nyújtott szolgáltatásokat elérheti és igénybe veheti.

Üzenet: a rendszer által kezelt protokollok mentén közlekedő entitás

11. Üzleti folyamatok

Belépés: a felhasználó megadja felhasználói nevét és jelszavát.

E-mail fogadása: *üzenet fogadása*

E-mail küldése: *üzenet küldése*

Felhasználó törlése: Logikai törlés. Logikailag törölt felhasználó nem regisztrált felhasználónak számít. Nem használhatja a rendszer szolgáltatásait. Újbóli regisztráció esetén korábbi adatai nem kerülnek visszaállításra, teljesen új felhasználót eredményez. Kiváltott esemény: *ügyfél törlése, adminisztrátor törlése*

Regisztráció: a rendszerben új *felhasználó* regisztrálására csak *adminisztrátor* felhasználó jogosult. A regisztráció így megegyezik az *Új ügyfél létrehozásá*val.

Új felhasználó létrehozása: a rendszerben korábban nem létező felhasználó a létrehozást követően létezik, s a rendszer használatára jogosulttá válik. A létrehozást követően a felhasználó e-mailben értesítést kap a sikeres regisztrációról, s megkapja belépési adatait. Jelszavát a rendszer generálja, melyet az első belépést követően kötelezően meg kell változtatnia. A felhasználó szükséges adatait egy *adminisztrátor* felhasználó adja meg. Kiváltott esemény: *új ügyfél létrehozása, új adminisztrátor létrehozása*

Ügyfél aktiválása: Letiltott ügyfél újra aktívvá válik, s a rendszer használatára újra jogosulttá válik. Kiváltott esemény: *ügyfél aktiválása*

Ügyfél letiltása: Letiltott ügyfél regisztrált felhasználóként létezik a rendszerben, viszont a rendszer szolgáltatásainak igénybevételére nem jogosult mindaddig, míg újra aktiválásra nem kerül. Kiváltott esemény: *ügyfél letiltása*

Üzenet fogadása: beérkező üzenet regisztrálása, továbbítása a célzott feldolgozó modul felé. Kiváltott esemény: *beérkező üzenet*

Üzenet küldése: a rendszer által létrehozott *rendszerüzenet* vagy egy ügyfél által létrehozott *normál üzenet* továbbítása. Kiváltott esemény: *kimenő üzenet*

12. Funkcionális specifikáció:

A *MessageForce* egy olyan webes interfésszel rendelkező több felhasználós, üzenetközpontú, moduláris, skálázható és bővíthető szoftverrendszer, mely különböző protokollok mentén közlekedő üzenetek, üzenetcsomagok fogadására, feldolgozására és továbbítására alkalmas.
Az üzenetek feldolgozását különböző modulok végzik, melyek tulajdonképpen a rendszer által biztosított üzenetalapú szolgáltatásokat testesítik meg.

A rendszer alapvető üzleti tevékenysége üzenetek fogadása, kezelése és továbbítása. Rövid távú célja, hogy e-mail üzeneteket legyen képes feldolgozni (*BaseMail modul)*, viszont lehetőséget kell biztosítania további protokollok adaptálására.
A rendszernek alapvetően képesnek kell lennie a feldolgozott üzenetek biztonságos tárolására, biztosítania kell azok visszakereshetőségét paraméterezhető módon. A kimenő és beérkező üzeneteket megkülönböztetett formában tudnia kell megjeleníteni. Fontos szempont az üzenetek nyomonkövethetősége: mikor, milyen feladótól, milyen protokollon milyen modulhoz, mely ügyfélhez érkezett egy beérkező üzenet, továbbá mikor, mely ügyféltől, milyen modul által, milyen protokollon, hová lett küldve egy kimenő üzenet. A nyomkövetésre épülő statisztikai lekérdezéseket biztosítania kell.
A rendszer további, elsődleges célja az ügyfelek kiszolgálása. Transzparens, weben elérhető felületet (*ClientInterface modul*) kell biztosítani a rendszer által nyújtott szolgáltatások használatára. Az ügyfél könnyedén tarthassa karban saját adatait, igénybe vett szolgáltatásait. Az igénybevett szolgáltatásokkal kapcsolatos információk, statisztikai adatok megjelenítésére lehetőséget kell nyújtani. Garantálni kell az ügyfél adatainak biztonságát.
A rendszer további elsődleges célja az ügyfélkezelés. Az adminisztrátorok számára transzparens, weben elérhető felületet (*AdminInterface modul*) kell biztosítani az ügyfelek menedzselésére: regisztrálására, karbantartására. Az ügyfelek által végrehajtott tevékenységek nyomonkövetésére,

statisztikai adatok megjelenítésére az ügyfelekkel, az általuk igénybevett szolgáltatásokkal és az üzenetforgalommal kapcsolatban.

A rendszer által biztosított funkciók:

Az ügyfél és az adminisztrátor felülete különböző url-en érhető el. A felhasználót fogadó oldal a belépést biztosító képernyő. Hibás felhasználónév vagy jelszó megadása esetén a rendszer tájékoztat a sikertelen belépésről.

Adminisztrátor felület:

Üdvözlő képernyő:
Belépés után a felhasználó az üdvözlő képernyőre jut. A képernyőn megjelennek az adminisztrátor által elérhető funkciókat biztosító funkciógombok:

- Operátor funkciók:
  - Ügyfelek: ügyfélkezelés képernyő
  - Statisztikák: statisztikák képernyő
  - Saját adatok: Adminisztrátor adatai képernyő
  - Kilépés: a belépett felhasználót kilépteti. Kilépés után a belépés képernyő jelenik meg.
- Manager:
  - Operátor funkciók
- Root:
  - Manager funkciók
  - Adminisztrátorok: adminisztrátorkezelés képernyő

Ügyfélkezelés képernyő:
A képernyőn megjelenik az összes, a rendszerben regisztrált ügyfél úgy, mintha egy paraméterek nélkül lefutattot keresés zajlott volna le. A képernyőn az alábbi funkciók érhetők el:

- Operátor funkciók:
  - Ügyfél keresése: a képernyőn a keresést segítő beviteli mezők jelennek meg:
    - Felhasználói név: tartalmazás alapú szűkítést biztosít. Ha a beírt szöveget valamely ügyfél felhasználói neve tartalmazza, a keresés gombra kattintva megjelenik az adott ügyfél az eredménylistában. A kis és nagybetűket a keresés nem veszi figyelembe. A kitöltetlen mezőt rendszer nem veszi figyelembe.
    - Név: működése megegyezik a felhasználói névnél vázoltakkal, viszont a keresés az ügyfél teljes nevére illeszt.
    - Szolgáltatások: a rendszerben aktuálisan elérhető szolgáltatások neve alapján szűkít. Több szolgáltatás is kiválasztható. Az eredménylistában azon ügyfelek fognak megjelenni, akik a kiválasztott szolgáltatások valamelyikét megrendelték. Ha nincs kiválasztott szolgáltatás, a rendszer a mezőt nem veszi figyelembe.
  - Ügyfél megtekintése: a keresés által eredményezett lista valamely sorát kiválasztva megjelenik az ügyféladatok képernyő
- Manager funkciók:
  - Operátor funkciók
  - Új ügyfél regisztrálása: a funkciót kiválasztva megjelenik az ügyféladatok képernyő
- Root funkciók:
  - Manager funkciók

Ügyféladatok képernyő:

Egy ügyfél adatai jelennek meg a képernyőn.

- Operátor funkciók:
  - Visszatérés az előző oldalra
- Manager funkciók
  - Operátor funkciók
  - A képernyőn megjelenő adatokat módosítása
  - Adatok mentése
  - Ügyfél törlése
- Root funkciók:
  - Manager funkciók

Statisztikák képernyő:

- Operátor funkciók:
  - Ügyfelek statisztika
  - Kimenő üzenetek statisztika
  - Bejövő üzenetek statisztika
- Manager funkciók:
  - Operátor funkciók
- Root funkciók:
  - Manager funkciók

Ügyfelek statisztika képernyő:

A statisztika paramétereinek beállítását segítő beviteli mezők jelennek meg a képernyőn:

- Szolgáltatás: lista, melyből kiválasztható 0, 1 vagy több szolgáltatás.
- Kezdő dátum: naptárból kiválasztható dátumérték állítható be

- Záró dátum: naptárból kiválasztható dátumérték állítható be
- Bontás: lista, melyből az alábbi értékek választhatók ki:
  - Nap
  - Hónap
  - Év

A Futtat funkciót aktiválva, a statisztika lekérdezésre kerül. A ki nem töltött beviteli mezők nem lesznek figyelembe véve. Az eredmény listaszerűen megjelenik a képernyőn:

- A kiválasztott bontásnak megfelelő dátumformátumban megjelenő dátumérték
- A bontásnak megfelelő időegységben a kezdő és záródátum között létező ügyfelek száma
- A következő *n* db oszlop: ha volt kiválasztva szolgáltatás, az *i*-edik oszlop az *i*-edikként kiválasztott szolgáltatásra regisztrált ügyfelek számát tartalmazza

Kimenő üzenetek statisztika képernyő:

A statisztika paramétereinek beállítását segítő beviteli mezők jelennek meg a képernyőn:

- Szolgáltatás: lista, melyből kiválasztható 0, 1 vagy több szolgáltatás.
- Kezdő dátum: naptárból kiválasztható dátumérték állítható be
- Záró dátum: naptárból kiválasztható dátumérték állítható be
- Bontás: lista, melyből az alábbi értékek választhatók ki:
  - Nap
  - Hónap
  - Év

A Futtat funkciót aktiválva, a statisztika lekérdezésre kerül. A ki nem töltött beviteli mezők nem lesznek figyelembe véve. Az eredmény listaszerűen megjelenik a képernyőn:

- A kiválasztott bontásnak megfelelő dátumformátumban megjelenő dátumérték
- A bontásnak megfelelő időegységben a kezdő és záródátum között kiküldött üzenetek száma

- A következő *n* db oszlop: ha volt kiválasztva szolgáltatás, az *i*-edik oszlop az *i*-edikként kiválasztott szolgáltatás által küldött üzenetek számát tartalmazza

Bejövő üzenetek statisztika képernyő:

A statisztika paramétereinek beállítását segítő beviteli mezők jelennek meg a képernyőn:

- Szolgáltatás: lista, melyből kiválasztható 0, 1 vagy több szolgáltatás.
- Kezdő dátum: naptárból kiválasztható dátumérték állítható be
- Záró dátum: naptárból kiválasztható dátumérték állítható be
- Bontás: lista, melyből az alábbi értékek választhatók ki:
  - Nap
  - Hónap
  - Év

A Futtat funkciót aktiválva, a statisztika lekérdezésre kerül. A ki nem töltött beviteli mezők nem lesznek figyelembe véve. Az eredmény listaszerűen megjelenik a képernyőn:

- A kiválasztott bontásnak megfelelő dátumformátumban megjelenő dátumérték
- A bontásnak megfelelő időegységben a kezdő és záródátum között beérkező üzenetek száma
- A következő *n* db oszlop: ha volt kiválasztva szolgáltatás, az *i*-edik oszlop az *i*-edikként kiválasztott szolgáltatás által fogadott és feldolgozott üzenetek számát tartalmazza

Adminisztrátorkezelés képernyő

A képernyőn megjelenik az összes, a rendszerben regisztrált adminisztrátor úgy, mintha egy paraméterek nélkül lefutattot keresés zajlott volna le. A képernyőn az alábbi funkciók érhetők el:

- Root funkciók:
  - Adminisztrátor keresése: a képernyőn a keresést segítő beviteli mezők jelennek meg:

- Felhasználói név: tartalmazás alapú szűkítést biztosít. Ha a beírt szöveget valamely adminisztrátor felhasználói neve tartalmazza, a keresés gombra kattintva megjelenik az adott adminisztrátor az eredménylistában. A kis és nagybetűket a keresés nem veszi figyelembe. A kitöltetlen mezőt a rendszer nem veszi figyelembe.
- Név: működése megegyezik a felhasználói névnél vázoltakkal, viszont a keresés az adminisztrátor teljes nevére illeszt.
- Szerepkör: a rendszerben létező adminisztrátor szerepkörök neve alapján szűkít. Több szerepkör is kiválasztható. Az eredménylistában azon adminisztrátorok fognak megjelenni, akik a kiválasztott szerepkör valamelyikével rendelkeznek. Ha nincs kiválasztott szerepkör, a rendszer a mezőt nem veszi figyelembe.

- Adminisztrátor megtekintése: a keresés által eredményezett lista valamely sorát kiválasztva megjelenik az Adminisztrátor adatai képernyő

Adminisztrátor adatai képernyő:

A felhasználó a megjelenő adatokat módosíthatja és mentheti.

Ügyfél felület:

Üdvözlő képernyő:

Belépés után a felhasználó az üdvözlő képernyőre jut. A képernyőn megjelennek az ügyfél által elérhető funkciókat biztosító funkciógombok:

- Szolgáltatások: megjelenik a Szolgáltatások képernyő
- Statisztikák: megjelenik a Statisztikák képernyő
- Saját adatok: megjelenik az Ügyfél adatai képernyő
- Kilépés: a belépett felhasználót kilépteti. Kilépés után a belépés képernyő jelenik meg.

Szolgáltatások képernyő:

A képernyőn megjelenik az összes, a rendszerben elérhető szolgáltatás úgy, mintha egy paraméterek nélkül lefutattot keresés zajlott volna le. A képernyőn az alábbi funkciók érhetők el:

- Szolgáltatás keresése: a képernyőn a keresést segítő beviteli mező jelenik meg:
  - Szolgáltatás neve: tartalmazás alapú szűkítést biztosít. Ha a beírt szöveget valamely szolgáltatás neve tartalmazza, a keresés gombra kattintva megjelenik az adott szolgáltatás az eredménylistában. A kis és nagybetűket a keresés nem veszi figyelembe. A kitöltetlen mezőt rendszer nem veszi figyelembe.
  - Megrendelt szolgáltatások: három elemű lista, az alábbi értékek választhatók ki:
    - Üres elem: A mezőt a keresés nem veszi figyelembe.
    - Megrendelt: a keresés csak a megrendelt szolgáltatásokat veszi figyelembe
    - Elérhető: a keresés csak az elérhető, az ügyfél által még nem megrendelt szolgáltatásokat veszi figyelembe

A Keresés funkciót aktiválva megjelenik a feltételeknek megfelelő szolgáltatásokat tartalmazó eredménylista. A szolgáltatások valamelyikét kiválasztva megjelenik a Szolgáltatás részletei képernyő.

Szolgáltatás részletei képernyő:

A képernyőn az ügyfél megtekintheti a szolgáltatás leírását. A képernyőn elérhető funkciók:

- Szolgáltatás aktiválása: az adott szolgáltatás a felhasználó számára aktívvá válik. Csak akkor érhető el a lehetőség, ha az ügyfél az adott szolgáltatást még nem aktiválta.
- Szolgáltatás lemondása: az adott szolgáltatás kikapcsolása az adott felhasználó számára. Csak akkor érhető el a lehetőség, ha az ügyfél az adott szolgáltatást korábban már aktiválta.
- Szolgáltatás kiválasztása: Csak akkor érhető el a lehetőség, ha az ügyfél az adott szolgáltatást korábban már aktiválta. A funkcióval az adott szolgáltatás felülete érhető el.

Ügyféladatok képernyő:

A bejelentkezett ügyfél adatai jelennek meg a képernyőn. A felhasználó módosíthatja, majd mentheti adatait.

Statisztikák képernyő:

- Kimenő üzenetek statisztika
- Bejövő üzenetek statisztika

Kimenő üzenetek statisztika képernyő:

A statisztika paramétereinek beállítását segítő beviteli mezők jelennek meg a képernyőn:

- Szolgáltatás: lista, melyből kiválasztható 0, 1 vagy több az ügyfél számára aktív szolgáltatás.
- Kezdő dátum: naptárból kiválasztható dátumérték állítható be
- Záró dátum: naptárból kiválasztható dátumérték állítható be
- Bontás: lista, melyből az alábbi értékek választhatók ki:

- Nap
- Hónap
- Év

A Futtat funkciót aktiválva, a statisztika lekérdezésre kerül. A ki nem töltött beviteli mezők nem lesznek figyelembe véve. Az eredmény listaszerűen megjelenik a képernyőn:

- A kiválasztott bontásnak megfelelő dátumformátumban megjelenő dátumérték
- A bontásnak megfelelő időegységben a kezdő és záródátum között kiküldött üzenetek száma
- A következő *n* db oszlop: ha volt kiválasztva szolgáltatás, az *i*-edik oszlop az *i*-edikként kiválasztott szolgáltatás által küldött üzenetek számát tartalmazza

Bejövő üzenetek statisztika képernyő:

A statisztika paramétereinek beállítását segítő beviteli mezők jelennek meg a képernyőn:

- Szolgáltatás: lista, melyből kiválasztható 0, 1 vagy több az ügyfél számára aktív szolgáltatás.
- Kezdő dátum: naptárból kiválasztható dátumérték állítható be
- Záró dátum: naptárból kiválasztható dátumérték állítható be
- Bontás: lista, melyből az alábbi értékek választhatók ki:
  - Nap
  - Hónap
  - Év

A Futtat funkciót aktiválva, a statisztika lekérdezésre kerül. A ki nem töltött beviteli mezők nem lesznek figyelembe véve. Az eredmény listaszerűen megjelenik a képernyőn:

- A kiválasztott bontásnak megfelelő dátumformátumban megjelenő dátumérték
- A bontásnak megfelelő időegységben a kezdő és záródátum között beérkező üzenetek száma

- A következő *n* db oszlop: ha volt kiválasztva szolgáltatás, az *i*-edik oszlop az *i*-edikként kiválasztott szolgáltatás által fogadott és feldolgozott üzenetek számát tartalmazza

BaseMail szolgáltatás felülete:

A felület az Ügyfél felületen a Szolgáltatás részletei képernyőn a Szolgáltatás kiválasztása funkció aktiválásával érhető el.

A felületre való belépést követően az Üdvözlő képernyő jelenik meg.

Üdvözlő képernyő:

A képernyő bemutatja a BaseMail szolgáltatást. A képernyőn az alábbi funkciók érhetők el:

- E-mail keresése: E-mail keresése képernyő jelenik meg
- E-mail küldése: E-mail küldése képernyő jelenik meg
- Kilépés: a BaseMail szolgáltatás bezárásra kerül, a felhasználó visszatér az Ügyfél felületre.

E-mail keresése képernyő:

Az ügyfél számára lehetőséget biztosít a küldött és fogadott e-mailek közötti keresésre. A képernyőn a keresést segítő beviteli mező jelenik meg:

- Címzett: tartalmazás alapú szűkítést biztosít. Ha a beírt szöveget valamely e-mail címzettjének e-mail címe tartalmazza, a keresés gombra kattintva megjelenik az adott e-mail az eredménylistában. A kis és nagybetűket a keresés nem veszi figyelembe. A kitöltetlen mezőt rendszer nem veszi figyelembe.
- Címzett neve: mint a Címzett esetén, viszont az e-mail küldőjének nevére illeszt.
- Küldő: mint a Címzett esetén, viszont az e-mail küldőjének e-mail címére illeszt.
- Küldő neve: mint a Címzett esetén, viszont az e-mail küldőjének nevére illeszt.
- Téma: mint a Címzett esetén, viszont az e-mail Téma mezőjére illeszt
- Tartalmaz: mint a Címzett esetén, viszont az e-mail Tartalom mezőjére illeszt

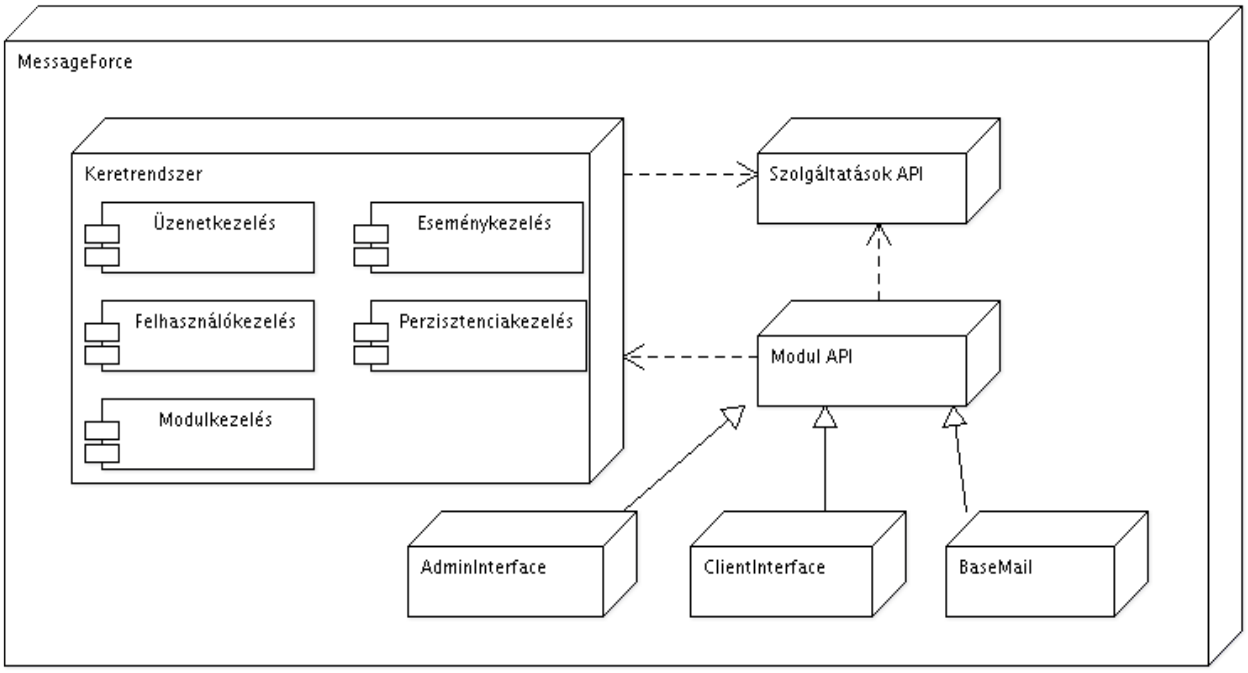
- Küldött/Fogadott: lista, mely az alábbi értékeket tartalmazza:
  - Üres elem: A mezőt a keresés nem veszi figyelembe.
  - Küldött: a keresés csak a küldött e-maileket veszi figyelembe
  - Fogadott: a keresés csak beérkező e-maileket veszi figyelembe

A Keresés funkciót aktiválva megjelenik a feltételeknek megfelelő e-maileket tartalmazó eredménylista. Az e-mailek valamelyikét kiválasztva megtekinthető annak minden adata. Itt a felhasználó válaszolhat a kiválasztott e-mailre. Ezt a lehetőséget aktiválva megjelenik az E-mail küldése képernyő, ahol a válaszhoz szükséges adatok kitöltésre kerülnek (címzett, küldő, téma).

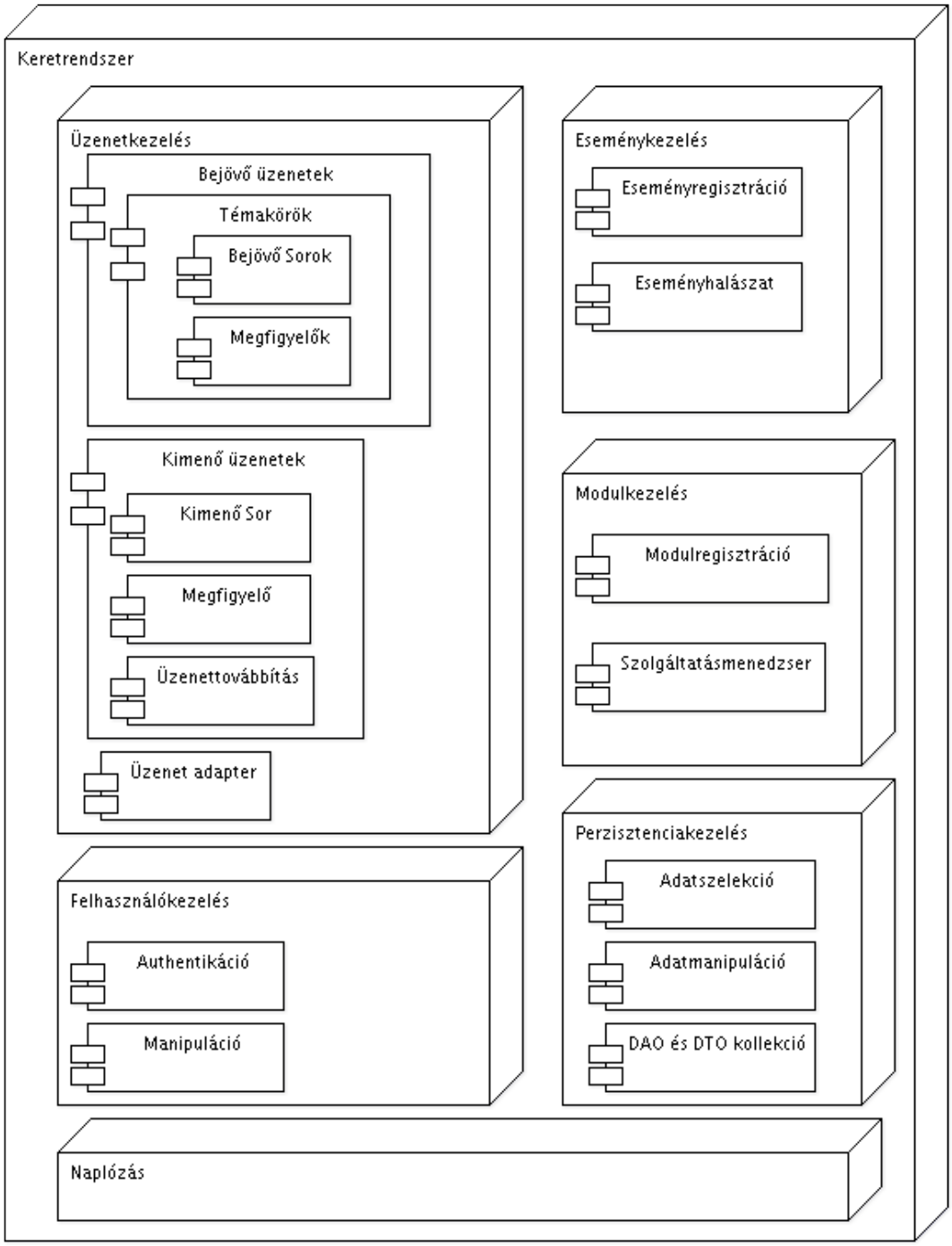
E-mail küldése képernyő:

Az oldalon az ügyfélnek lehetősége van e-mail küldésére. A képernyőn megjelennek a küldést segítő beviteli mezők. A Küldés funkciót aktiválva az e-mail kiküldésre kerül.

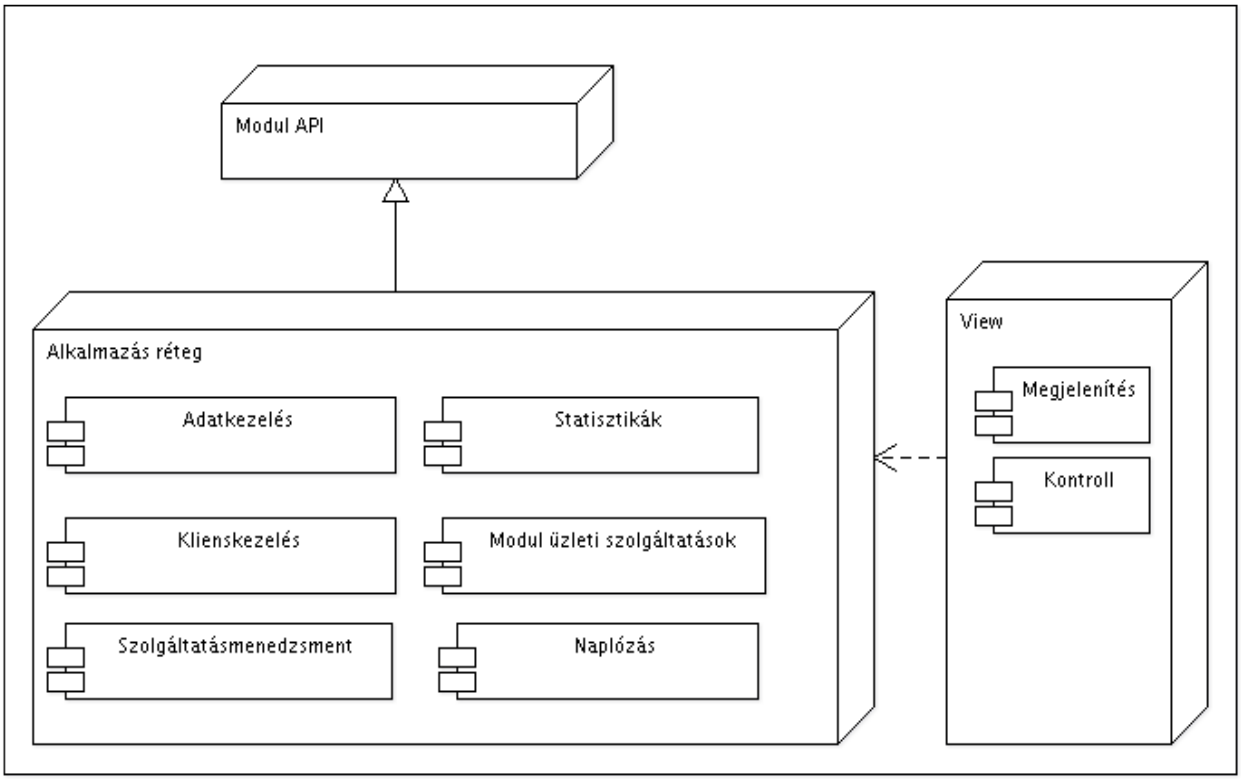
13. Deployment diagram (a rendszer főbb alrendszerei):



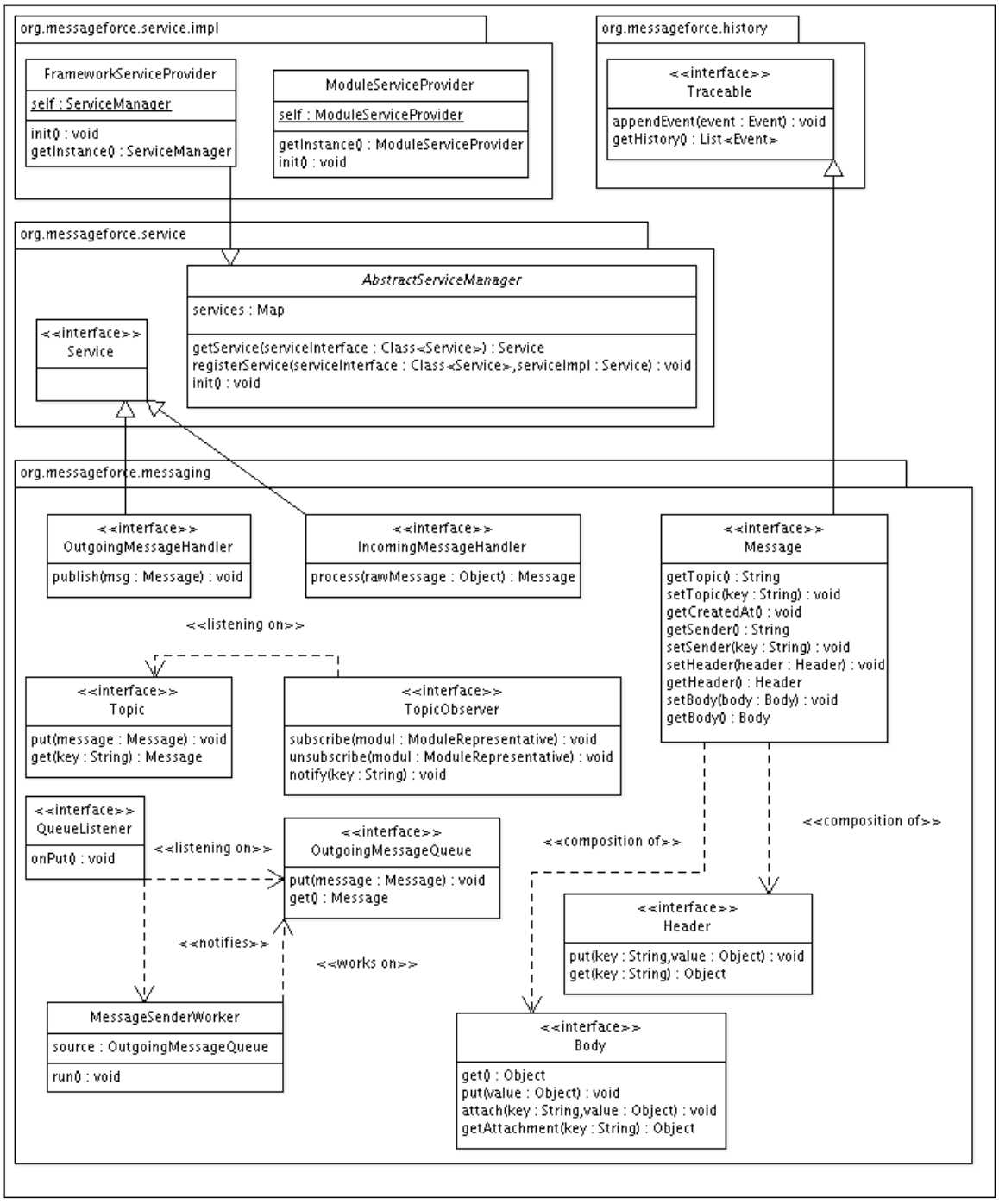
14. Komponens diagram (a keretrendszer főbb komponensei):



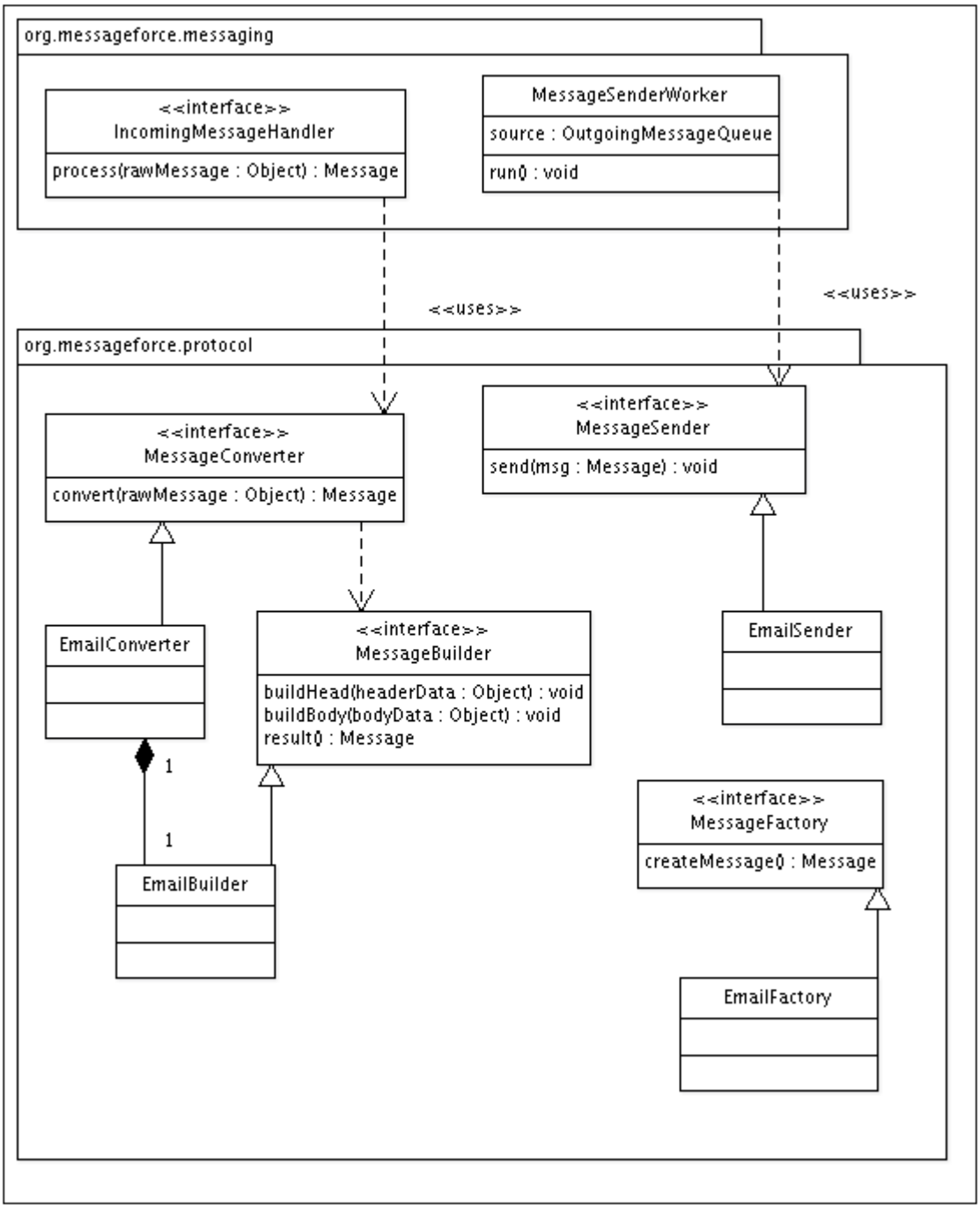
15. Egy modul komponensei általában:



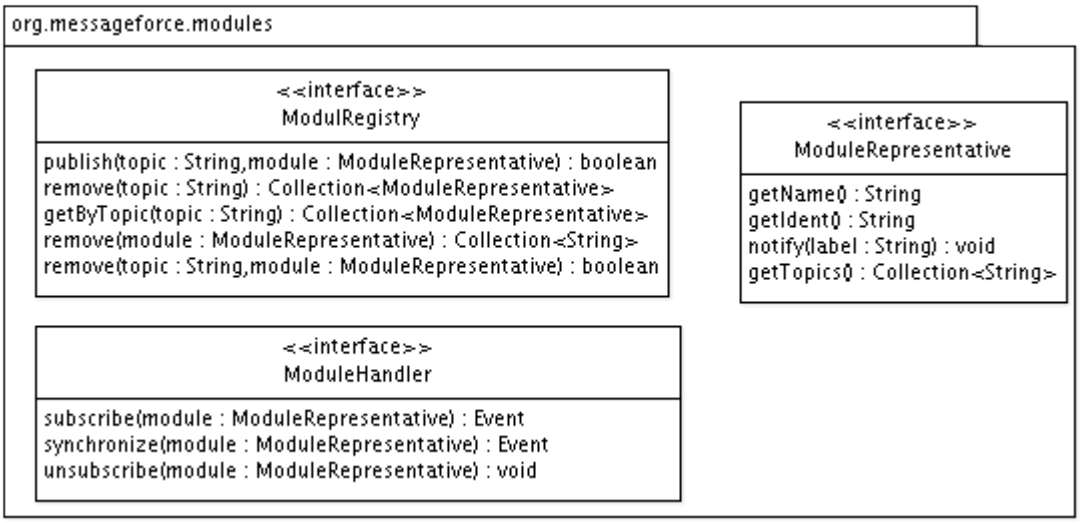
16. Üzenetkezelő komponens osztálydiagramja



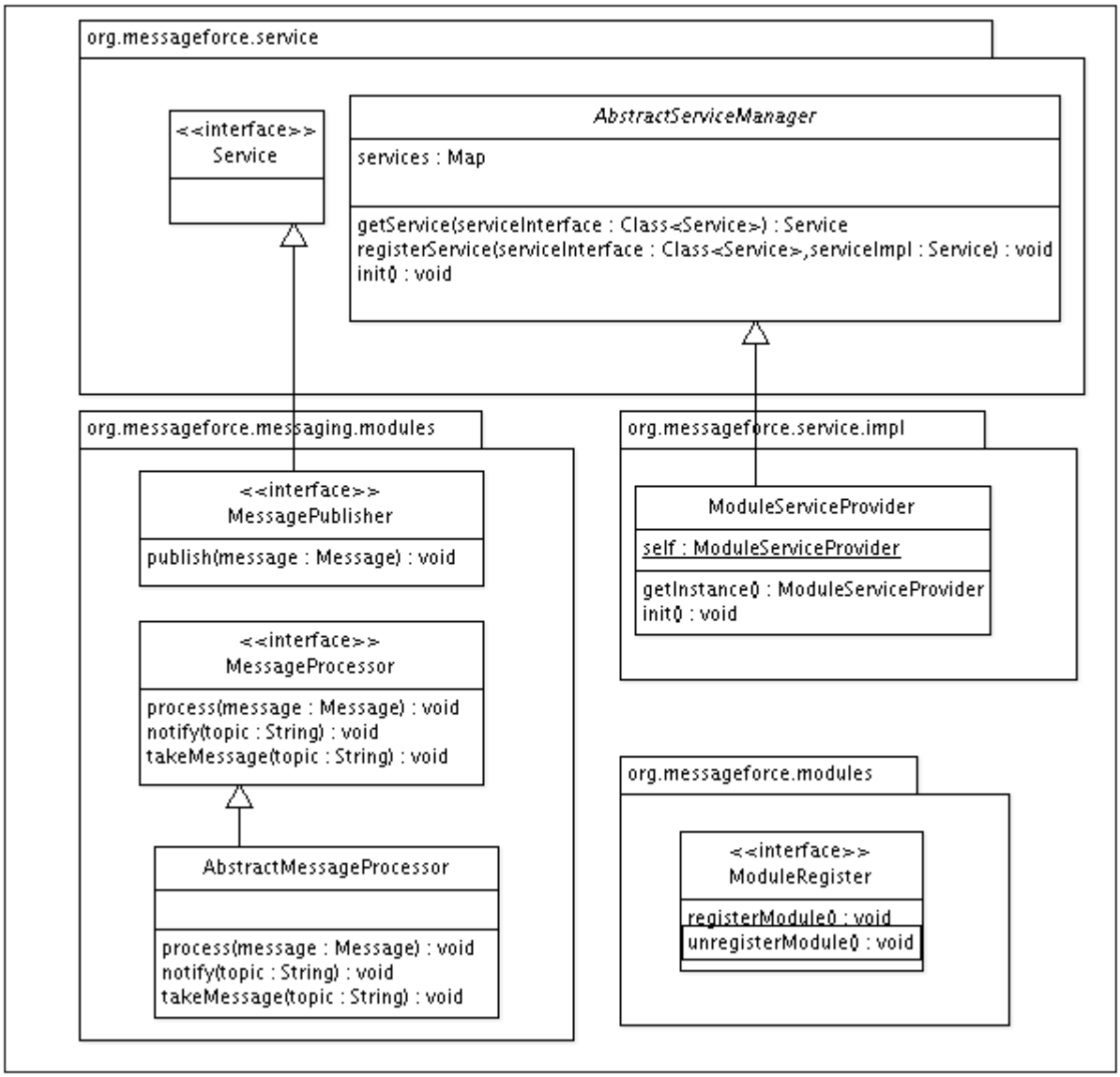
17. Protokolltámogatás osztálydiagramja:



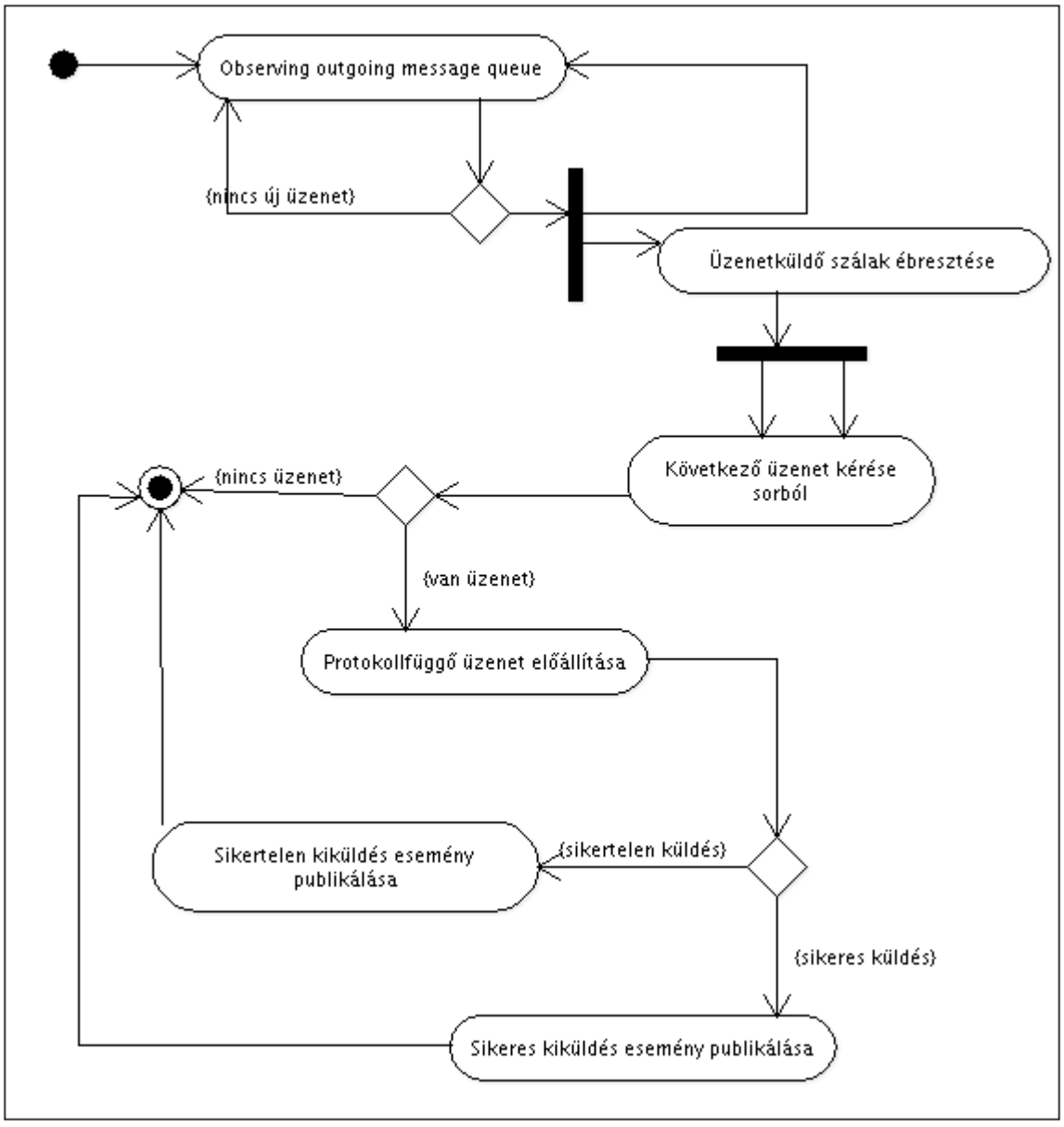
18. Modulkezelés osztálydiagramja:



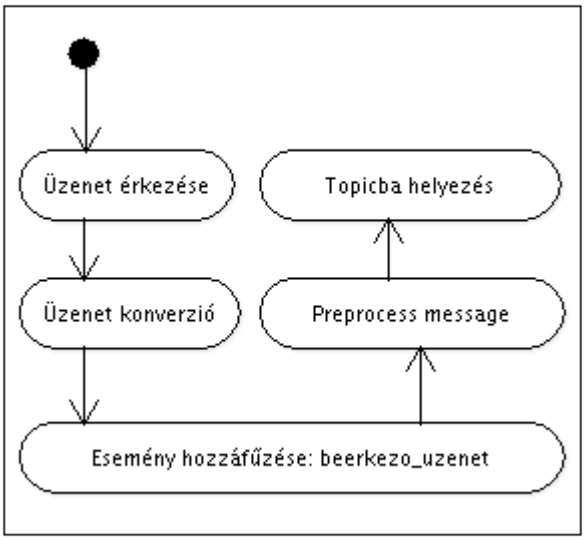
19. A modul-api legfontosabb elemei:



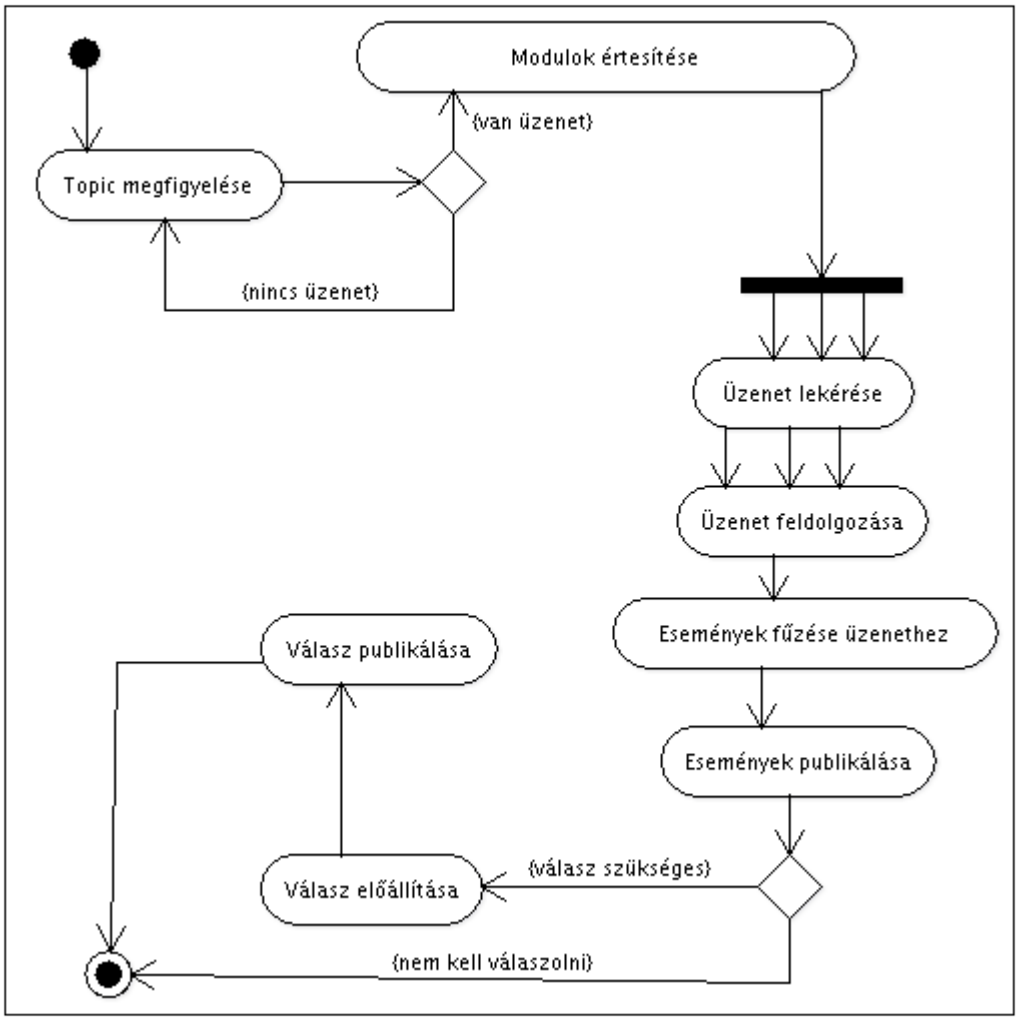
20. Üzenet küldése



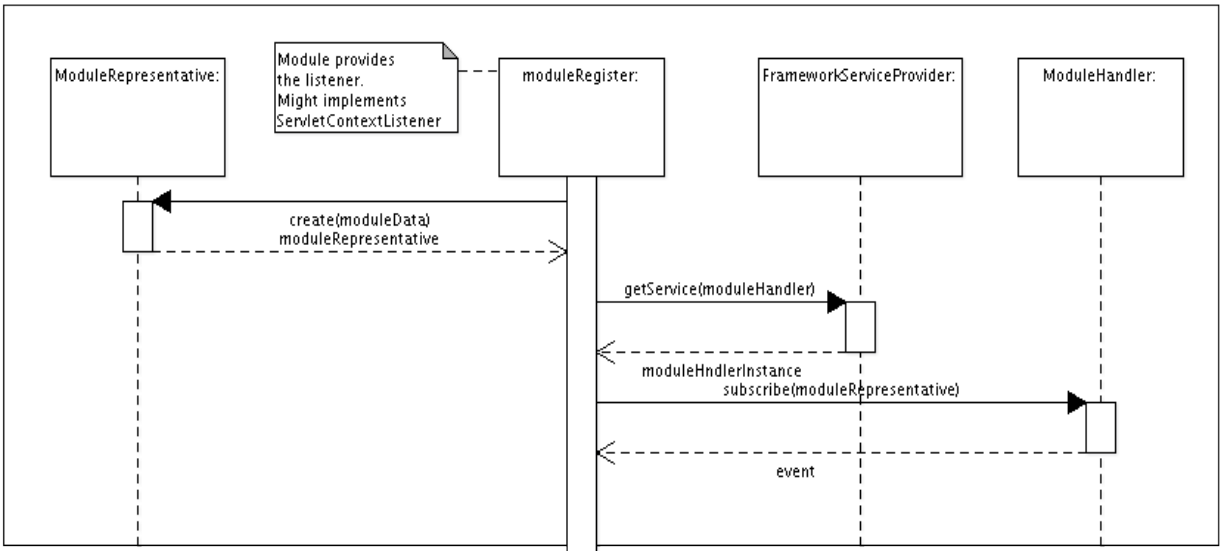
21. Beérkező üzenet megfelelő *Topic*ba helyezése



22. Üzenet feldolgozása



23. Modul regisztrációja a keretrendszerben



Köszönetnyílvánítás

Ezúton szeretnék köszönetet mondani mindazoknak, akiknek munkája, segítsége, tanítása nélkül e dolgozat nem jöhetett volna létre.

Köszönettel tartozom:

Dr. Juhász István Tanár Úrnak a sok ismeretért, tudásért, s nem utolsó sorban a szemléletmódért, mely nélkül nem lennék ott, ahol tartok;

Szüleimnek állandó támogatásukért, s megértésükért;

Barátaimnak, mert nemcsak tanulásból és munkából áll az élet