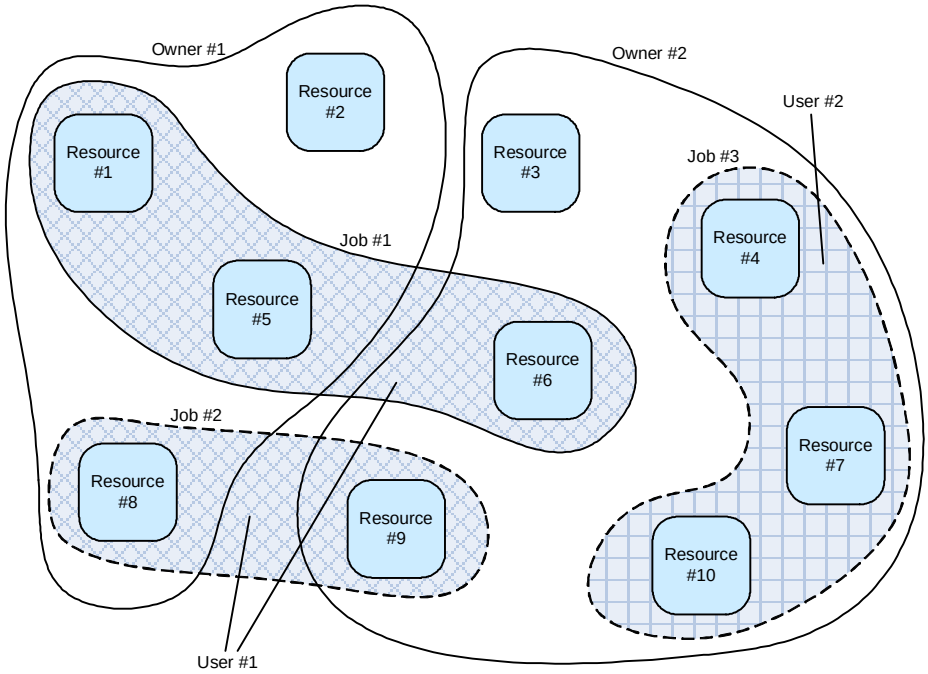
ERŐFORRÁSFELHASZNÁLÁS-NYILVÁNTARTÓ ÉS -ELSZÁMOLÓ RENDSZER GRID KÖRNYEZETBEN

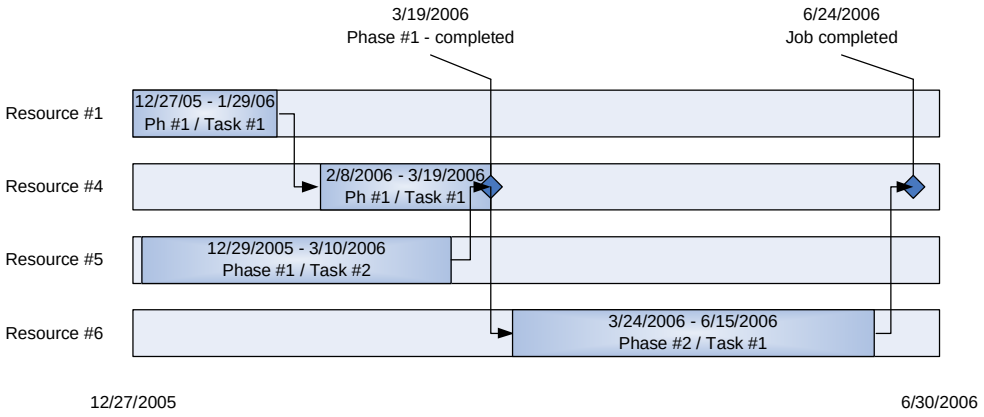
Somogyi Csongor, somogyi@iit.bme.hu
BME Irányítástechnika és Informatika Tanszék
Szmrecsányi Márton, szmrecsanyi@ik.bme.hu
BME Informatikai Központ
László Zoltán, laszlo@iit.bme.hu
BME Irányítástechnika és Informatika Tanszék

1. BEVEZETŐ

Egy elosztott erőforrásfelhasználás (efh) -nyilvántartó és -elszámoló rendszer létrehozása a Grid üzleti alkalmazásának elengedhetetlen feltétele. A Gridben az erőforrás-felhasználás térben és időben elosztott módon folyik. Az efh-nyilvántartó rendszereknek követniük kell ezt a fajta elosztottságot, képesnek kell lennie a begyűjtött és rendszerezett adatokat a Grid működését követve kezelni.

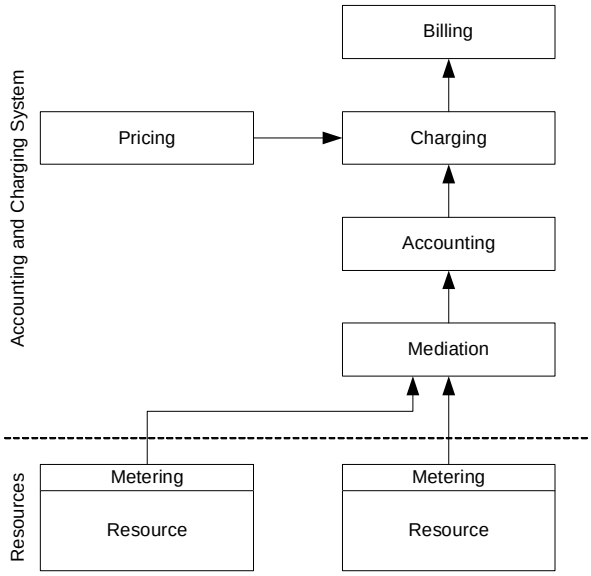


1. ábra – Elosztottság térben



2. ábra – Elosztottság időben

Az efh-nyilvántartás és -elszámolás folyamatának vázát egy svájci kutatócsoport dolgozta ki [2, 3]:



Ezek szerint a folyamatnak 6 fő eleme van:

- mérés (metering)
- mediálás, adataggregálás (mediation)
- nyilvántartásba vétel (accounting)
- árazás (pricing)
- elszámolás (charging)
- számlázás (invocing, billing)

Az elosztott rendszerekhez készített efh-elszámoló rendszernek ugyanúgy tartalmaznia kell a fenti komponenseket.

3. ábra – Az efh-nyilvántartás és elszámolás folymata

A fenti koncepció szerint indultunk neki eredetileg a SzuperGrid [4] majd a KlaszterGrid [5] projektekben a felesztési munkálatainknak. Célul tűztük ki:

- egy a koncepciónak megfelelő, elosztott rendszer létrehozását
- a Grid biztonságtechnikájának megfelelő módszerek alkalmazását
- olyan webes kezelőfelület létrehozását, amely általánosan alkalmazott számlázási forgatókönyvek szerint képes egy erőforrás-tulajdonos erőforrásainak számlázására

Ebben a cikkünkben bemutatjuk az általunk megalkotott rendszert. Ismertetjük az alkalmazott modellt (2. fejezet), a rendszerterv lényeges elemeit és a megvalósítást (3. fejezet, végül összegezzük eredményeinket (4. fejezet).

2. MODELL

2.1. Statikus modell

A 2003-as Networkshop cikkünkben [6, 7] már bemutattuk a rendszer alapját képező entitás-relációs és adatfolyam modelleket. Ezek kisebb-nagyobb változtatásokon mentek keresztül azóta, de az alapelvek nem változtak. Hely hiányában itt csak hivatkozunk a fenti cikkünkre.
Egy lényeges dolgot emelnénk ki. Az adatok alapjában véve rendszerezhetők erőforrások és mérési paraméterek alapján. Így az egyes funkciók (accounting, charging, stb.) is eloszthatók eszerint, amit mint általános kocepciót tekintve terveztük meg a rendszer egyes adatrögzítő és -szolgáltató elemeit.

2.2. Dinamikus modell

A dinamikus modell a rendszeren belüli folyamatokat hivatott leírni. Ismertettjük, hogy milyen folyamatokat terveztünk megvalósítani ill. valósítottunk meg az efh-elszámoló rendszerben.

2.2.1. Adminisztráció

A szolgáltatók, felhasználók, erőforrások és árazási szabályok karbantartására külön szolgáltatásokat definiáltunk. Az egyes entitások jellegéből adódnak a rajtuk végezhető műveletek. Minden entitás létrehozható és törölhető. A felhasználók és szolgáltatók információja módosítható. Az árazási szabályok szintén módosíthatóak, viszont bonyolultabb szabályok alapján, hiszen itt vannak kötelezően és opcionálisan megadható mezők.

2.2.2. Adatgyűjtés

A felhasználók lefoglalják az erőforrásokat egy job részére, amit az erőforrásütemező regisztrál (allocation service, begin és end command). Ezek a foglalások mutatják, hogy milyen időszakban, melyik felhasználó használta az adott erőforrást. Ez később a számlázás alapja is.
A foglalás közben automatikusan kerülnek a számlázó rendszerbe a fogyasztási adatok vagy szintén az erőforrásütemező segítségével egy adott időpillanatban megállapítható a fogyasztás mértéke.

2.2.3. Elszámolás (Charging)

A charging végrehajtásához szükség van egy erőforrás, egy job azonosító és egy időintervallum megadására. A folyamat azzal kezdődik, hogy az adott job mely időintervallumokban foglalta le az adott erőforrást. Ha ezen intervallumokban az erőforrás fogyasztás változott, akkor le kell kérdezni a megfelelő árazási szabályt. Az erőforrások fogyasztását többféle paraméter is jellemezheti, ezért az árazási szabály lekérdezésekor meg kell adni az erőforrás paraméterének nevét és egységét is. Ezeket egy konfigurációs állományban célszerű tárolni. Mivel a charging kérésben nem adható meg paraméter, ezért a konfigurációban található összes paraméterre le kell kérdezni az árakat, és minden egyes árhoz létre kell hozni egy charge rekordot.
Ha az adott intervallumban már léteznek még nem érvénytelenített (cancelled) charge rekordok, akkor azokat meg kell vizsgálni. Ha egy rekord beleesik az intervallumba és a hozzá tartozó árazási szabály nem változott, akkor a rekord érintetlen maradhat. Ha azonban változott, akkor a rekordot érvényteleníteni kell. Ez csak abban az esetben lehetséges, ha a rekord nem része egy számlának. Az érvénytelenítés után újra fel kell venni a rekordot az új árazási szabály alapján. Ha egy létező rekord kilóg az intervallumból, akkor a rekordot ketté kell bontani. Ez szintén csak akkor lehetséges, ha a rekordot nem tartalmazza számla, ellenkező esetben a folyamatot meg kell szakítani és hibával kell visszatérni.

Felmerülhet a kérdés, hogy a charge rekordok érvénytelenítése helyett mért nem töröljük egyszerűen a rekordokat. A válasz egyszerű. Egyrészt a számlák és a charge rekordok közötti szoros összefüggés nem teszi lehetővé, ugyanis érvénytelenyített számlákról is tudnunk kell, hogy milyen hozzá tartozó rekordok érvénytelenek. Másrészt a rendszer elosztottságából adódóan a műveleteket úgy igyekeztünk megtervezni, hogy azok félig-meddig állapotmentesek (semi-transparent) legyenek. Ha véletlenül egy művelet megszakad, akkor az adatok sose váljanak inkonzisztenssé, legfeljebb a műveletet újra végre kelljen hajtani, de ekkor már csak azok a változások menjenek végbe amik még szükségesek.

2.2.4. Számlázás (invoicing)

A számlázás többféleképpen történhet. Először is meg kell különböztetni a felhasználó szerinti és job szerinti számlázást.
Lehetőség van egy job-ot számlázni, valamint egy felhasználó egy időszakra vonatkozó erőforrás használatát. Mindkét lehetőségnél meg kell adni, hogy mely erőforrásokat szeretnénk számlázni. A kapott adatok alapján a szerver lekérdezi a charge rekordokat, majd felveszi őket a számlába.
Minden esetben csak érvényes charge rekordok kerülnek leszámlázásra.

2.2.5. Érvénytelenítés (Cancelling)

A számlázás és charging folyamatokhoz szorosan hozzátartozik a számlák érvénytelenítése. Az érvénytelenítés során nem csak a számla érvénytelenítődik, hanem a számla minden sora, azaz az összes charge rekord is. Ennek oka az elosztottság, hiszen mivel a charge rekord szorosan kapcsolódik egy számlához, ezért nem feltétlenül lenne

szükség az érvénytelenítés nyilvántartására minden egyes charge rekordnál. Viszont a charge rekordok lekérdezésekor szükség van annak státuszára is, és ha nem lenne tárolva, akkor le kéne kérdezni a számlázó szervert, hogy a rekordhoz tartozó számla érvénytelenített-e vagy sem. Ez viszont csökkenti a hatékonyságot, és logikailag sem helyes a charging szervernek a számlázó szervertől szolgáltatás kérnie.

Az adatgyűjtés, charging és számlázás egymásra épülő folyamatok. A számlázáshoz szükség van a charge rekordokra, melyek a charging során keletkeztek. A charginghoz szükség van a fogyasztási adatokra, melyek az automatikus adatgyűjtés során keletkeznek. Ez a felhasználó számára is ilyen formában jelenik meg.

3. RENDSZERTERV, MEGVALÓSÍTÁS

3.1.1. Adatbázis réteg és business modell

Adattárolásra MySQL [8] adatbázist használtuk. Az adatbázis eléréséhez a „hibernating" technológiát alkalmaztuk [9]. A „hibernating" használatából fakadóan a szakterületi modell és az adatbázis réteg szorosan összefonódik a kezelés szempontjából. Hiszen így nincs szükség külön az adatbázis mezők és az objektum változók közti összerendelés megvalósítására, mivel az xml [10] állományokban megadható. Természetesen ez az összefonódás nem jelenti azt, hogy a két réteg nem választható el egymástól, ugyanakkor MySQL helyett alkalmazhatnánk bármilyen más JDBC [11] kapcsolattal rendelkező adatbázis rendszert.

3.1.2. Végrehajtó réteg (gacs-schema és gacs)

Ez a réteg lehetőséget biztosít a számlázó rendszer által nyújtott szolgáltatások eléréséhez. A szolgáltatóknak elküldött kérések folyamatokat indítanak, és egy formailag a protokollban meghatározott választ küldenek vissza. Az egyes szolgáltatások különböző parancsokat képesek fogadni.

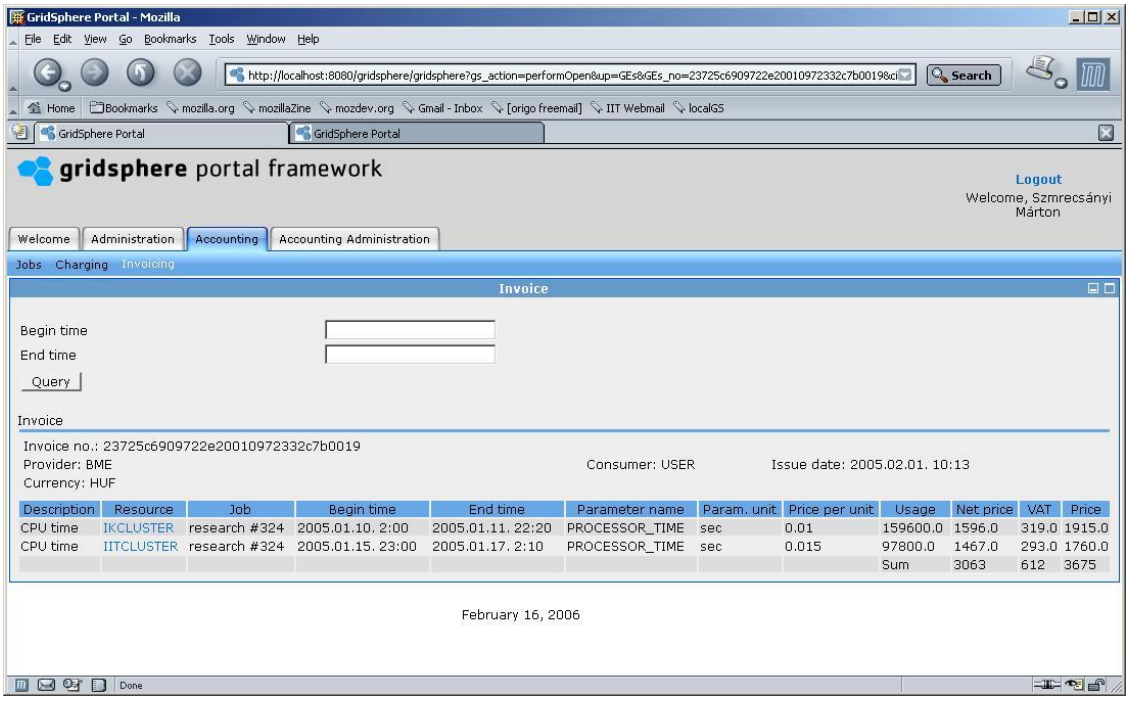
A szolgáltatások és a hozzájuk tartozó parancsok:

- Adminisztratív szolgáltatások:
  - Felhasználók: felvétel, felhasználói adatok módosítása, törlés és lekérdezés
  - Szolgáltatók: felvétel, információk módosítása, törlés és lekérdezés
  - Erőforrások: felvétel, törlés és lekérdezés
  - Árazási szabályok (pricing): felvétel, módosítás, törlés, lekérdezés; árazás
- Számlázási szolgáltatások:
  - Fogyasztás (accounting): felvétel, lekérdezés
  - Elszámolás (charging): lekérdezés, érvénytelenítés, hozzárendelés számlához; elszámolás
  - Számlázás (invoicing): számlázás, érvénytelenítés, lekérdezés

A végrehajtó réteget Servlet technológiára épülve alkottuk meg.
A kommunikációs közeg standardizálásához az egyes modulok HTTP-n [14] keresztül GET ill. ún. „form” POST kérésekkel XML dokumentumok küldésével cserélnek információt.
Mivel a http-n keresztül átküldött xml dokumentumok feldolgozása körülményes feladat, ezért a JAXB-t (Java API for XML Binding) [15] hívtuk segítségül. Így a protokollt leíró xml sémából generáltuk az osztályhierarchiát, melynek segítségével már a küldött xml-ben lévő elemeket objektumokként kezelhettük.
Minden szolgáltatáshoz különböző „parancsokat” definiáltunk, melyekre a felhasználó külön kaphat engedélyt (részletesebben a biztonsági részben).
A kommunikációs protokollt és az egyes szolgáltatások konfigurációját egy xml séma írja le (gacs-schema specification v1.0) [12].

3.1.3. Megjelenítés (gacs-ui)

A megjelenítési réteg külön szerveren működhet, mivel minden szolgáltatás http-n keresztül elérhető. A megjelenítő kliensnek egy url-re van szüksége, mégpedig a serviceURL-re, amelyen egy szolgáltatásleíró állomány található. Ebből kiindulva az árazási és számlázási folyamatokhoz szükséges szolgáltatások elérhetőek.



4. ábra – Felhasználói kezelőfelület

Mivel a rendszer megjelenítése a felhasználó felé is fontos feladat, ezért nagy hangsúlyt fektettünk annak kialakítására. Szükség volt egy olyan keretrendszerre, amely beleillik a grides környezetbe. Mivel a grides világ többnyire a GridSphere portál keretrendszert [13] használja, ezért kézenfekvő volt a mi rendszerünket is ebbe a környezetbe illeszteni.

A megjelenítést két részre osztottuk. Egyrészt szükség van egy adminisztrációs felületre, mely kezeli a szolgáltatókat, felhasználókat, erőforrásokat és az árazási szabályokat, valamint a számlázási folyamat 3 fázisát megjelenítő részre. A felületen megjelenített funkciók tükrözik a protokollban leírt lehetőségeket.

3.1.4. Biztonság

A biztonsági rendszer alapjául a GSI (Grid Security Infrastructure) [16] proxy tanusítványok technológiáját választottuk, ami az Interneten széles körben elterjedt publikus-kulcs infrastruktúrára épül [17]. A módszer lényege, hogy a felhasználó aláír egy speciális ún. proxy tanusítványt, majd ezzel a tanúsítvánnyal azonosítja magát. Ennek az az előnye, hogy ez a proxy tanúsítvány további proxy tanúsítványokat hoz létre, amivel a felhasználó jogosultságai delegálhatóak annélkül, hogy az eredeti tanúsítványt alkalmaznunk kellene. Ez azért fontos, mert a felhasználói tanúsítvány általában jelszóval védett. Továbbá fontos, mert egy harmadik személy is el tud járni a felhasználó személyében. A felhasználó az első proxy tanúsítványban beállíthat olyan attribútumokat, amivel szabályozni tudja a további proxy tanúsítványok létrejöttét.
A számlázó rendszer a GSI Grid Toolkit 3-as és afeletti verziójában alkalmazott [17, 18] ill. az RFC 3820 [19] szerint szabványosított proxy tanúsítványokat kezeli és fogadja el.
Amellett, hogy a szabványokat implementáltuk, lehetővé tettük, hogy a jogosultságdelegáció ténylegesen működjön, vagyis egy szolgáltatás elő tudjon állíttatni egy proxy tanúsítványt, amivel újabb szolgáltatásokat tud elérni a felhasználó nevében.

4. EREDMÉNYEK

Az elmúlt évek folyamatos kutató-fejlesztő munkájának eredményeképpen létrehoztunk egy elosztott efh-nyilvántartó és -elszámoló rendszert, ami a 3. ábra szerinti funkcionalitással rendelkezik és architektúrája a Grid elosztottságát tükrözi.
Készítettünk egy protokoll specifikációt (gacs schema specification 1.0) és egy Java Servlet alapú referenciaimplementációt (gacs). Erre a rendszerre építve készítettünk egy GridSphere portál keretrendszerre épülő felhasználói kezelőfelületet (gacs-ui). Biztonsági protokollokba beépítettük a GSI-t, valamint megoldottuk a delegáció kérdését is, biztosítva ezzel a szolgáltatások közötti kommunikációt.
Az így elkészült rendszeren lehetőségünk nyílik további fejlesztéseket eszközölni:

- kifinomultabb árazási szabályok bevezetése (felhasználó, fogyasztásfüggő, napszakfüggő, stb.)
- fejlettebb, a szolgáltatók igényeit magasabb színvonalon kielégítő kezelő-felület (pl. havi bontások, kimutatások)
- átszámlázás – ami lehetővé teszi szolgáltatók közötti erőforrás megosztásra, kölcsönzésre

KÖSZÖNETNYILVÁNÍTÁS

Jelen cikkben bemutatott munka az IKTA 00064/2003 valamint az NKFP OM-00262/2004 projekt támogatásával készült.

HIVATKOZÁSOK

1. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke.: A Resource Management Architecture for Metacomputing Systems. Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing (1998)
2. B. Stiller, J. Gerke, P. Reichl, P. Flury and Hasan: Charging Distributed Services of a Computational Grid Architecture. IEEE International Symposium on Cluster Computing (CCGrid 2001), Workshop on Internet QoS for the Global Computing (IQ 2001), (May 2001) 596–601
3. B. Stiller, J. Gerke, P. Reichl and P. Flury: A Generic and Modular Internet Charging System for the Cumulus Pricing Scheme. Journal of Network and Systems Management, **3(9)** (September 2001) 293–325
4. IKTA-00075/2001, Magyar Szuperszámítógép Grid. Pályázat információs és kommunikációs technológiák és alkalmazások fejlesztésének támogatására, Oktatási Minisztérium, Budapest (November 2002)
5. IKTA-00064/2003, A magyar SzuperGrid és KlaszterGrid rendszerek felhasználó-orientált egységesítése. Pályázat információs és kommunikációs technológiák és alkalmazások fejlesztésének támogatására, Oktatási Minisztérium, Budapest (2003)
6. Somogyi Cs., Dr. László Z., Szeberényi I.: Erőforrás-felhasználás nyilvántartó és elszámoló rendszer Condor Grid ütemező környezetben, Networkshop 2003 Konferencia, Pécs
7. C. Somogyi, Z. László, I. Szeberényi: A Resource Accounting and Charging System in Condor Environment. Euro-Par 2003, 26 - 29 August 2003, Klagenfurt, Austria
8. MySQL Database Management System, `http://www.mysql.org`
9. Hibernate Relational Persistence, `http://www.hibernate.org`
10. T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler: Extensible Markup Language (XML) 1.0, (Second Edition). W3C Technical Report 20001006, `http://www.w3.org/TR/2000/REC-xml-20001006` (2000)
11. JDBC: Java Database Connectivity 3.0, `http://java.sun.com/products/jdbc/`
12. Somogyi, C., Szmrecsányi, M.: Grid Accounting and Charging System (GACS) Schema Specification 1.0, `http://www.iit.bme.hu/gacs/schema/1.0`, BME IIT, Budapest, 2004-2006
13. GridSphere Portal Framework 2.0, `http://www.gridsphere.org`
14. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, June 1999.
15. JAXB: Java API for XML Binding 1.0: `http://java.sun.com/webservices/jaxb/`
16. GSI: Grid Security Infrastructure, `http://www.globus.org/security/`
17. Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.
18. Java CoG kit, jGlobus API, a renference implementation for proxy certificates, `http://wiki.cogkit.org/index.php/Main_Page`
19. Tuecke, S., Welch, V., Engert, D., Pearlman, L., and M. Thompson, "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile", RFC 3820, June 2004.