Providing Scheduling Applications as Secure Internet Services

Goulas George, Barkayannis Vassilios, Housos Efthymios
Computer Systems Laboratory, Department of Electrical & Computer Engineering,
University of Patras, GR-26500, Rio, Patras, Greece
goulas@ee.upatras.gr, barkayannis@ee.upatras.gr, housos@ee.upatras.gr

Abstract
This paper describes SchedSP, a proposed architecture for providing scheduling applications as services over the Internet in a secure manner. This architecture is based on secure XML messaging over the HTTP protocol and utilizes a distributed Grid-based organization of computational resources for the execution of the scheduling solutions. Along with the prototype of the system, three case studies are presented. One of the case studies provides a combined bus-driver scheduling algorithm for use over the Internet, another exposes a nurse shifts scheduling system for remote usage, and the third case study allows for an Internet based automatic creation of high school timetables.

Keywords Application Service Provision, Grid, XML Security

1 Introduction

In this paper, a platform for providing secure access to scheduling applications over the Internet, named SchedSP, is presented. The motive of this work is the existence of several scheduling algorithm implementations as products of scientific and research work. These implementations concern mainly the areas of crew assignment and scheduling as well as timetabling of academic institutions. Scheduling algorithms have as purpose to best allocate specific resources to tasks in order to create an acceptable work schedule, subject to several linear and non-linear constraints. A common characteristic of scheduling problems is their NP-hard complexity and thus they have special needs in computational power and memory capacities. This could lead to large execution times and need for powerful computational systems in order to generate the optimal solution.
Since such powerful computational systems are hard to find and quite expensive, the Internet, as a pool of many computers interconnected, could provide an adequate environment for the execution of such scheduling applications. The increase in availability of sufficient and always-on Internet connectivity creates an opportunity to offer software and computing resources in a leasing manner through the Internet, using the Application Service Provider (ASP) model. An ASP is an organization that provides software as a service over the Internet, either based on the pay per use model or the fixed periodic fee approach. Furthermore, the ASP is responsible for the software upgrade and maintenance, relieving the end user from such problems [SCN Education B.V.(2000)].
The SchedSP system was created following the ASP model for providing scheduling applications as secure services over the Internet. The SchedSP framework utilizes the PLEIADES system [Koulopoulos(2002A), Koulopoulos(2002B)], a distributed Grid-based organization of computational resources, in order to satisfy the special needs of scheduling applications in computing environments. For communicating purposes, SchedSP uses XML messages over the HTTP protocol. In order to secure the transactions between the end user and the SchedSP system, issues such as confidentiality and integrity of the data, authentication and authorization must be taken into account. Therefore, and due to the fact that data are transmitted in XML formats, the new XML security standards [Eugene(2001), Naedele(2003)] are taken into consideration and investigated.

2 Scheduling Applications

Scheduling applications are applications with special needs in computational power. The usual scheduling application scenario includes input files with resources that need to be matched with other resources in an optimal manner and obeying a set of rules and constraints. Scheduling resources can be in most cases divided to human resources and other resources, such as machines or classrooms, in the case of school timetabling. The main characteristic of such applications is the complexity of the solution process, which can be of exponential order of the problem size, thus leading to a great amount of execution time.
In order to solve a scheduling problem, it is first transformed to a mathematical model and then various optimization techniques are used to find the optimal solution. These techniques can be provided by various libraries and software tools, which need to be installed on the computational system that executes the scheduling algorithm. Such systems must provide considerable computational and memory resources so that to execute the scheduling applications and generate the optimal solution.

3 Applications Service Providers

An Application Service Provider (ASP) is an entity that provides and manages software solutions as services to customers across the Internet or other types of wide area networks. The ASP not only offers the software solutions but also the computational resources and software environment to execute them, adopting a thin client model for the user interface running on the customers workstations, allowing the user to execute computational intensive applications remotely. In addition, the end user is relieved from the maintenance problems and costs since the software maintenance and upgrades are carried out exclusively by the ASP. The ASP model allows the organizations that follow it to choose flexible pricing models in return to the services they provide, such as per use or per month pricing, or even provide demo applications for a limited time horizon.

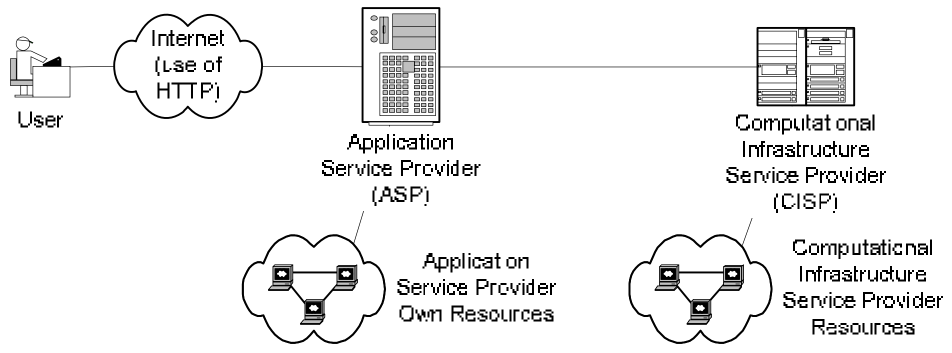
User
Internet (use of HTTP)
Application Service Provider (ASP)
Computational Infrastructure Service Provider (CISP)
Application Service Provider Own Resources
Computational Infrastructure Service Provider Resources

Figure 1: The CISP Concept

One key quality measure that an ASP should focus on is the computational power. The computational needs can be distinguished into two levels. Computational needs that have soft real-time requirements and needs that could be satisfied by an offline computational service. The latter case includes computations like simulations or human resource scheduling applications that usually require great amounts of execution time. An environment that offers distributed resources, like a Grid [Foster(1998)] or Condor Resource Manager [Condor Project Homepage, Livny(1997)], can be suitable for this type of computational needs. Therefore, an ASP could utilize such a computational service from a third-party provider. This delivery of computational power in the form of a Grid or Cluster as a service over the Internet is often called Computational Infrastructure Service Provider (CISP) [Koulopoulos(2002A)]. A CISP is shown in Figure 1.

A CISP system under the name of PLEIADES [Koulopoulos(2002A), Koulopoulos(2002B)] has been developed recently. The PLEIADES system utilizes a pool of computers managed by the Condor Resource Manager for the computational infrastructure. As shown in Figure 2, the services offered by PLEIADES are services for submitting jobs for execution, monitor services, space for file storage and management and a service for cross-compile which allows compilation of programs in selected platforms offered by PLEIADES. These services are accessible either by a web-based interface or by XML Web Services over HTTP.

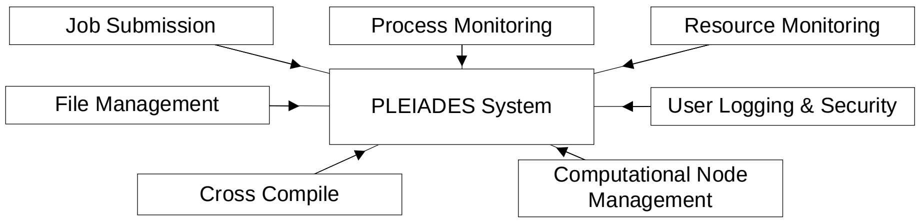

Figure 2: The PLEIADES Services

4 SchedSP

The SchedSP framework design aims at providing scheduling applications as services over the Internet [Goulas(2002)]. The special input/output and modeling nature of scheduling applications together with their demanding computational needs were the basic considerations in the design of the core services of SchedSP. For the user interface, a thin-client approach is able to satisfy the user requirements for the creation of the necessary interface modules and is the selected operation mode. This practically means that there are minimal user equipment expectations meaning that a machine capable to present rich web content containing either Java applets or ActiveX components is the only SchedSP requirement. The thin client interface components communicate with SchedSP using the XML messaging over the HTTP protocol. Figure 3 presents the communication layers, where SchedSP layer is responsible for authentication, authorization as well as other similar issues of managerial nature and the SchedSP Service Provisioning layer is responsible for providing the actual services.

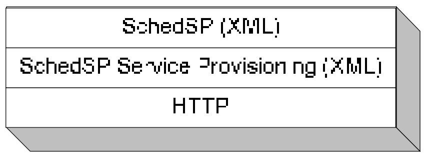

Figure 3: SchedSP Communication layers

SchedSP provides several services, which fall into two main categories: File storage, execution and monitoring of scheduling applications. The file storage support the need for storing files needed by the scheduling and other applications offered by the SchedSP framework, the execution and monitoring of scheduling applications utilize the PLEIADES system for Grid-based execution of these computationally demanding applications. An architecture overview of the SchedSP framework is presented in Figure 4, while the present prototype architecture is presented in Figure 5. In the following section, an attempt will be made to present the steps taken for a typical use scenario.

A user, using the SchedSP interface, connects to SchedSP using the HTTP protocol over the Internet. The user requests are channeled through a web server to the Gateway component, which is responsible for web server security issues and for forwarding the secure requests to the Executive component. The Executive component implements the SchedSP layer of the communication stack, and is responsible for the user authentication and authorization. The Executive component also distinguishes the various requests in accordance with the service component required for its service and forwards the necessary portion of the request to the appropriate component. The appropriate component also called a Service Provisioning Component (SPC) serves the user request and the corresponding response returns to the user interface following the same path backwards, through the Executive and the Gateway component.

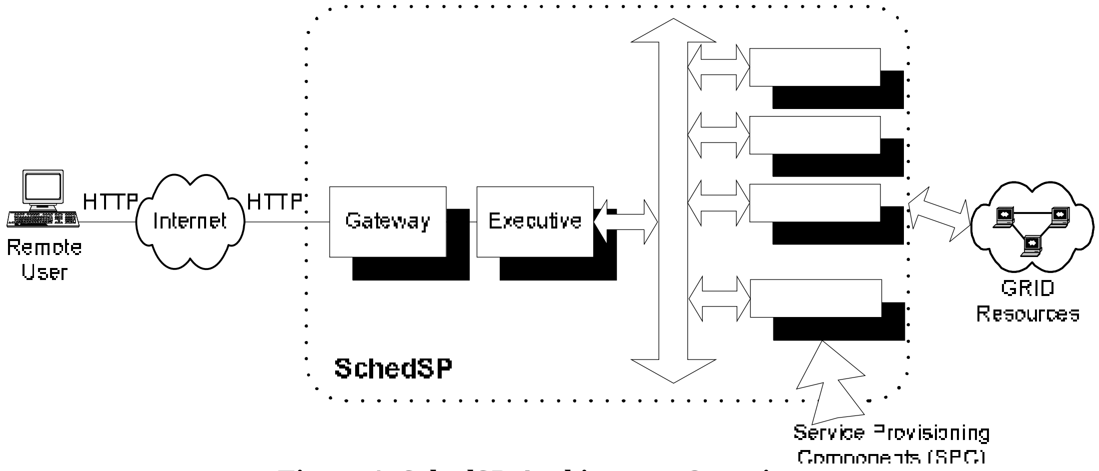

Figure 4: SchedSP Architecture Overview

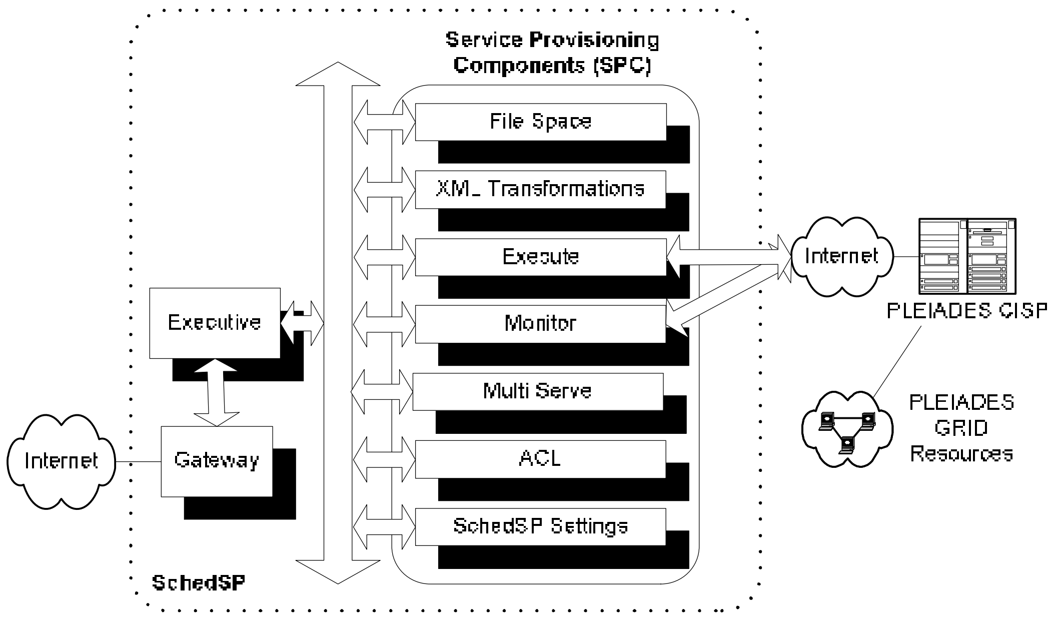

Figure 5: SchedSP Prototype Architecture

Figure 5 presents the present set of SPC components of the SchedSP prototype. The File space SPC serves the file operations. The XML Transformations SPC performs several XML transformations including XSLT and XPath-based modifications to XML files. The Execute and Monitor components support the execution and monitoring of scheduling applications. The ACL (Access Control List) SPC provides uniform role-based access control to resources across the SchedSP components while the Settings component provides other uniform functionality. Finally, the MultiServe SPC provides a useful script for a sequence of service requests in order to save the round-trip delays during complex operations.

5 SchedSP Security

SchedSP, as a service provisioning system, inherits most of the security risks that the Application Service Providing (ASP) and Web Services models present. Therefore, one of the most important security concerns involves the security vulnerabilities of the Internet protocols set (TCP/IP, HTTP, etc.) and the security of the servers that host web applications, in this case the SchedSP platform. Particularly, in the case of the security of web servers, solutions involve the use of traditional techniques such as firewalls and Intrusion Detection Systems (IDS), which in most cases are quite effective [Anderson(2001)].
Apart from the above security risks that every web application has to confront, SchedSP presents other security concerns that have to do with the nature of the applications that the platform serves. In addition, the distributed nature of the SchedSP system and its interaction with multiple computational systems arise new security concerns [Moffett(1995)]. Especially, attention must be given to the security of the data that is transmitted between the user and the SchedSP system. In most cases, this data comprises sensitive data, either personal or confidential data that an enterprise uses. Therefore, it is of great importance to secure the messages that the communicating parties exchange. In the next paragraphs message security is examined and solutions are presented.
One of the most important security requirements is confidentiality of the transmitted message, which ensures that the data cannot become available to a third malicious party. Encrypting either whole the transmitted message or selected portions of data from the message provides confidentiality. The Secure Sockets Layer protocol (SSL) ensures encryption of the data by ciphering the transmitted message. The main drawback of SSL is the fact that it does not provide persistent encryption, meaning that data is encrypted only during the communication between the two parties, while on the server the message is stored unencrypted [Hartman(2003), O'Neill(2003)]. Moreover, SSL presents a significant overhead in ciphering/deciphering the messages. In order to provide persistent encryption, the XML Encryption standard is used [W3C XML Encryption WG]. The main advantage of XML Encryption is that portions of an XML document can be selectively encrypted. This contributes to reducing significantly overhead because only sensitive portions of the message need to be encrypted and thus avoiding the encryption of the whole set of data [Dournaee(2002)].

XML Encryption structure

```
<EncryptedData Id? Type?>
        <EncryptionMethod/>?
        <ds:KeyInfo>
                <EncryptedKey>?
                <AgreementMethod>?
                <ds:KeyName>?
                <ds:RetrievalMethod>?
                <ds:*>?
        </ds:KeyInfo>?
        <CipherData>
                <CipherValue>?
                <CipherReference URI?>?
        </CipherData>
        <EncryptionProperties>?
</EncryptedData>
```

XML Signature structure

```
<Signature>
        <SignedInfo>
        (CanonicalizationMethod)
                (SignatureMethod)
                (<Reference (URI=)? >
                        (Transforms)?
                        (DigestMethod)
                        (DigestValue)
                </Reference>)+
        </SignedInfo>
   (SignatureValue)
  (KeyInfo)?
  (Object)*
</Signature>
```

Figure 6: XML Encryption and XML Signature Structures

Another security requirement of the SchedSP system is the integrity of the transmitted messages. Integrity is the requirement of detecting if a third party has tampered the transferred data and can be ensured by using the XML Signature standard [IETF/W3C XML Signature WG]. XML Signature defines how data contained in XML formatted messages can be digitally signed by using the existing algorithms for signatures and message digests and

then represent the resulting signature in XML. Tampering with the data causes the generation of different message digests that do not verify the original digest, therefore unveiling the alteration [Dournaee(2002)]. In addition, the use of digital signatures can be used for satisfying the authentication requirement of the SchedSP system, since each digitally signed message contains information about its sender that cannot be disputed [Eastlake(2003)]. The XML Encryption and XML Signature structures are shown in Figure 6.
For authorization purposes of the SchedSP platform the ACL component is used. Authorization ensures the fact that each user with access to SchedSP and especially to the filespace it provides, has well defined access privileges and these are well enforced. The ACL component defines access control both in user level and in role level and enforces these policies to each user.

6 Case Studies

6.1 Bus Scheduling

The daily bus and driver scheduling for a typical Greek bus company is a difficult combinatorial problem that has to be solved every day. A quick heuristic scheduling procedure named Quick Shifts (QS) has been developed for the solution of the combined bus-driver problem, along with a column generation procedure named Column Generation and Quick Shifts combination (CGQS) [Valouxis(2002)]. Based on these algorithms, a Windows 32-bit console application has been developed with input and output files in plain-text format and a set of useful information and status report presented on the user screen.
The above algorithm is utilized as a core part of a solution provided as service over the Internet for the creation of daily bus and driver schedules [Mavrommatis(2003)]. This solution involves an efficient user interface implemented as an ActiveX component and a wrapper executable for the controlled execution of the scheduling algorithm with translation of input and output files into XML format. By the use of the wrapper the scheduling algorithm uses XML for its input and output files, since SchedSP provides significant advantage for such files stored in the filespace. The role of the graphical user interface is to gather all input from the user, create the input files for the scheduling algorithm, instruct SchedSP to execute the scheduling algorithm through the wrapper and then present the results back to the users. The Execute SPC has been configured to utilize the wrapper on the appropriate computing platform, as well as transfer several useful files for the execution of the algorithm.

6.2 Nurse Scheduling

The second solution based on the SchedSP framework involves the creation of working schedules for nursing personnel. A hybrid algorithm, using an approximate Integer Linear Programming (ILP) model with Tabu search strategies, has been developed for the creation of nurse scheduling solutions of local medical institutions [Valouxis(2000)]. The application developed based on this algorithm has many characteristics similar to the bus scheduling application. It is a console application with input and output files in plain-text format, while useful status messages are displayed on the screen.
An integrated solution has been developed in order to provide nursing personnel scheduling over the Internet as a service [Ntanos(2002)]. This solution involves a graphical user interface, implemented as an ActiveX component, and a wrapper program that executes the nursing personnel scheduling algorithm while it translates the input and output files into XML formatted documents. The user interface, apart from gathering the input information and displaying the results, allows certain steps of rescheduling. The Execute SPC has been configured to utilize the wrapper on the appropriate computing platform, as well as transfer several useful files for the execution of the algorithm. The wrappers for the case studies are essentially different programs, although they share common outline functionality and a small portion of code.

6.3 High School Scheduling

The third solution based on SchedSP provides the creation of teacher schedules for a Greek high school as an Internet service. In typical Greek high schools, the creation of timetables is a very time-consuming task, which is usually done manually. The discrete and combinatorial nature of the high school scheduling problem makes its solution quite difficult and the problem NP complete. An integer programming approach has been adopted for the solution of the problem, leading to successful solutions of very high quality and in accordance to the problem definition [Birbas(1997)]. The prototype of this approach is an executable running on a HP-UX machine and is based on the ILOG CPLEX library. The input of this type of problem is complex and as a result is split into numerous files because of the many relations between the various entities described. The output is a single file with the generated schedule. All these files follow a custom plain-text format.

For the provision of this algorithm as an Internet service, a solution is currently developed. The main component of this solution is a graphical user interface, implemented as a Java applet. A main difference is the use of the XML Transformations SPC instead of the wrapper used in the previous case studies. The Java applet gathers the user input, creates and updates the corresponding XML files in the SchedSP file space, and requests the execution of the scheduling algorithm. Prior to execution, the applet requests the transformation of the XML input into the numerous plain-text files needed by the algorithm. The Execute SPC is configured to transfer these files at the execution site, and utilize only machines based on the HP-UX operating system. Following the completion of the execution, the applet having knowledge of the custom format of the output presents it to the user while it allows for certain re-scheduling steps.

6.4 SchedSP Case Studies Overall Experience

The development of the above case studies presented only little overhead development effort over a hypothetical desktop solution to provide scheduling solutions. In both situations, the existence of a ready-to-use scheduling algorithm implementation is supposed. Setting aside the starting learning curve, the overall development process did not present significant concerns since the functionality of the various SchedSP services resembles to similar operating system services. This resemblance of services between local operating system services and SchedSP services was indeed between the primary design goals. The use of widely accepted and utilized protocols, namely XML and HTTP, turn the development of the client side of the interface into an easy task since it is based on widely used software libraries. For the first two case studies – the bus-driver scheduling and the nursing personnel – there was no client API designed for SchedSP for usage because SchedSP was also at a final stage of development that time. Even under this situation the main concern was not of the service provisioning concept and the usage of remote resources but the design of the efficient user interface.

7 Conclusions

In this paper, the possibility to provide scheduling applications as secure services over the Internet is investigated. The various details in architecture and security of a system designed to allow such a provision of scheduling solutions are analyzed, followed by several case studies. The SchedSP framework presents an easy to use environment in order to efficiently create complete scheduling solutions as services over the Internet, as shown in the case studies section. A significant advantage is the common set of functionality provided by the SchedSP framework that would have to be implemented in all case studies if they had been created independently.

In the immediate plans is the further enhancement of the security level of the various SchedSP interactions with other online systems. Moreover, since the adoption of the SOAP protocol by the SchedSP system is within the future considerations, a more wide use of the XML security standards can be carried out. By using the SOAP protocol the SchedSP will be considered an XML Web Service, allowing the security model of SchedSP to be implemented as an independent online service, playing the role of a third trusted party between the user and the SchedSP.

References

1 Anderson, R. (2001). *Security Engineering*, Wiley

2 Birbas, T., Daskalaki, S., and Housos, E. (1997). Timetabling for Greek High Schools, *Journal of the Operational Research Society,* 48(12), pp. 1191-1200

3 Condor Project Homepage: http://www.condorproject.org/

4 Dournaee, B. (2002). *XML Security*, McGraw-Hill/Osborne

5 Eastlake, D. III and Niles, K. (2003). *Secure XML: The New Syntax for Signatures and Encryption*, Addison-Wesley

6 Eugene, X. P. (2001). XML based Security for E-Commerce Applications, *Eighth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '01)*

7 Foster, I. and Kesselman, C. (1998). *The Grid: Blueprint for a New Computing Infrastructure*, Morgan–Kaufmann

8 Goulas, G. and Housos, E. (2002). SchedSP: Providing GRID-enabled Real-World Scheduling Solutions as Application Services, *Proceedings of the Euroweb 2002 conference, W3C*

9 Hartman, B., Flinn, D., Beznosov, K., and Kawamoto, S. (2003). *Mastering Web Services Security,* Wiley Publishing Inc.

10 IETF/W3C XML Signature Working Group, http://www.w3.org/Signature/

11 Koulopoulos, D., Papoutsis, K., Goulas, G., and Housos, E. (2002A). PLEIADES: An Internet-based parallel distributed system, *Software: Practice and Experience,* Wiley, 32(11), pp. 1035-1049

12 Koulopoulos, D., Goulas, G., Papoutsis, K., and Housos, E. (2002B). A Parallel / Distributed Platform for University Computational Infrastructure Service Provisioning, *Proceedings of the CSIT Workshop 2002, CSIT, Patras, Greece*

13 Livny, M., Basney, J., Raman, R., and Tannenbaum, T. (1997). Mechanisms for High Throughput Computing, *SPEEDUP Journal*, 11(3)

14 Mavrommatis, M. (2003). *Design and Implementation of Applications for an xSP Service Provisioning System,* Diploma Thesis, Department of Electrical & Computer Engineering, University of Patras

15 Moffett, J. (1995). Distributed Systems Security, *Encyclopaedia of Microcomputers*, Vol 15. New York: Marcel Dekker Inc

16 Naedele, M. (2003). Standards for XML and Web Services Security, *IEEE Computer,* 36(4), pp. 96-98

17 Ntanos, E. (2002). *Design and Implementation of Applications for an xSP Service Provisioning System,* Diploma Thesis, Department of Electrical & Computer Engineering, University of Patras

18 O'Neill, M. et al. (2003). *Web Services Security*, McGraw-Hill/Osborne

19 SCN Education B.V. (Eds). (2000). ASP – Application Service Providing, The Ultimate Guide to Hiring than Buying Applications, Vieweg

20 Valouxis, C. and Housos, E. (2000). Hybrid Optimization Techniques for the Workshift and Rest Assignment of Nursing Personnel, *Artificial Intelligence in Medicine*, 20, pp. 155-175

21 Valouxis, C. and Housos, E. (2002). Combined Bus and Driver Scheduling, *Computers & Operations Research*, 29, pp. 243-259

22 W3C XML Encryption Working Group, http://www.w3.org/Encryption/2001/