IJPIS

USING STANDARDISED WEB SERVICES FOR INTEGRATING PROCESSES OF AUTONOMOUS ACTORS COMMUNICATING WITHIN A PUBLIC INFORMATION SYSTEM

- A CASE STUDY FROM GREECE

KOSTAS MARKELLOS
University of Patras, Computer Engineering and Informatics Department,
University Campus, 26500, Rio, Greece
kmarkel@cti.gr

MARIOS KATSIS
University of Patras, Computer Engineering and Informatics Department,
University Campus, 26500, Rio, Greece
makatsis@cti.gr

GEORGIA PANAGOPOULOU
National Statistical Service of Greece
Peiraos 46, 18510 Peiraias, Greece
panag@statistics.gr

SPIROS SIRMAKESSIS
Department of Applied Informatics in Administrations and Economy
TEI of Mesolonghi
New Buildings, 30200, Mesolonghi, Greece
syrma@cti.gr

ATHANASIOS TSAKALIDIS
University of Patras, Computer Engineering and Informatics Department,
University Campus, 26500, Rio, Greece
tsak@cti.gr

Abstract

This work presents the design and development of a prototype module for the National Statistical Service of Greece, which supports the daily data collection procedures within the Organisation, while also removing a severe burden on the responding enterprises. The basic idea behind this effort was to enable enterprises to use and integrate it with their commercial and accounting packages, in order to extract from the legacy systems all the data required for reporting, in an "invisible" and easy way. The major benefits arising from the use of the system include the overall improvement of the system and of the process, in addition to a reduction in the burden placed on the respondents, especially for providers with large amounts of declarations. As a result, an overall improvement of data quality is expected. We have implemented the prototype module as a Web Service API, since Web Services allow different applications from different sources to exchange data

without the necessity of implementing custom codes and applications for this purpose. Due to the fact that all underlying communication uses XML, the Web Service is not tied to any operating system or programming language. Additionally, Web services are being adopted in the marketplace as a mechanism for efficient process integration in the organizations within and across organizational boundaries.

Keywords: Public information system, web services, XML

1. Introduction

Web services are the new breed of Web applications. The Web services technology has already been established by the World Wide Web consortium[1]. Web Services [Ewald, 2003] are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. In other words, web services are interoperable building blocks for constructing applications. As an example, we can imagine a distributed digital library infrastructure built on web services providing functionality such as distributed search, authentication, inter-library loan requests, document translation and payment. These web services would be combined by a particular digital library application to offer an environment for reaching information resources tailored to its particular user community. While the web enables users to connect to applications, the web services architecture enables applications to connect to other applications. Web services are therefore a key technology in enabling business models to move from B2C (Business to Consumer) to B2B (Business to Business) [Gardner, 2001].

From a technological perspective the Web Services framework is the core for building well formed and efficient Web Services. It consists essentially of three basic components: The Web Service Description Language (WSDL), a language to allow formal functional characterization of the provided functionalities, the Simple object Access Protocol (SOAP), a protocol that defines the format of the information interchange, and the Universal Description, Discovery and integration (UDDI), which is a catalogue of Web Service Descriptions. The Web Services framework will be discussed further in the technology section.

In this paper, we encourage the use of standardised web services for integrating a variety of different processes in the communication between autonomous actors and public information systems. In other words we focus on the data exchange from enterprises and governmental institutes. More specifically, we present a prototype module implemented for the **N**ational **S**tatistical **S**ervice of **G**reece (NSSG) regarding the monthly **In**tra-Community **Tra**de **Stat**istics (INTRASTAT) data collection process, whose aim is to improve the entire data collection exercise, reducing the time interval between the collection time and publication of statistical results,. In addition it also aims to improve the overall data quality and most importantly, to remove the burden from the reporting enterprises. The module is designed in such a way to enable the enterprises to use and integrate it with their commercial and accounting packages, so that data may be directly extracted from their systems and transmitted to NSSG without actual user intervention. In this way, it is expected that the burden posed on reporting organisations will be significantly reduced. The NSSG Web Service embraces emerging standards for describing (WSDL), discovering (UDDI), and invoking (SOAP) Web services.

[1] http://www.w3c.org

The NSSG Web Service is built around a collection of features that include a framework for composing services for online submitting and managing INTRASTAT declarations, optimized querying services, and preserving privacy.

The remainder of this paper is organized as follows. Section 2 provides the motivation that led to the development of the module. It presents the basic results of the feasibility study that took place prior to the development, highlighting the drawbacks of the current system, as well as what the Web Service aims to do and which points could be improved. Section 3 presents the technology background (Web Service Framework and Functional Model) while in Section 4 we describe the architecture of the prototype module. Section 5 refers to the implementation of the case study and finally, Section 6 gives our concluding remarks.

2. Motivation

The monthly data collection of INTRASTAT declarations from the enterprises is one of the most important, arduous and extremely time-consuming operations performed in NSSG.

Intrastat is the name given to the method of collecting information and producing statistics regarding the export and import of goods between Member States of the European Union (EU). It is a system designed to be used by Agents, Traders and individual branches of larger companies who submit data (via the Internet) independently from their head offices. Intrastat's objective is the provision of information about trade flows between the EU Member States. This data provides a comprehensive source of information for the national institutions (Ministry of Economy, Ministry of Agriculture, Ministry of Finance, etc.), private companies and organizations, and individuals in order to plan production, produce market surveys, to forecast and analyze the development of the economy. Information, provided by Intrastat, is widely used by the EU institutions (European Commission, European Central Bank, etc.) and various international organizations.

The foundation of the system is the Combined Nomenclature Directory which is the common nomenclature of the European Community [European Commission]. The 8-digit sub-headings in the nomenclature are used in export declarations and in statistical declarations on internal trade. There are changes to the nomenclature every year and the nomenclature that will come into force at the beginning of the following year is published yearly in the Official Journal of the European Union, by the end of October at the latest.

Until only a few years ago, INTRASTAT data were submitted by the enterprises as an "additional" module of their VAT declarations (paper-based). The completed forms were collected by the Tax Authorities, on behalf of NSSG and were finally sent to the statistical institute for further processing (data entry, error correction, data analysis, production of final statistical data and dissemination of results). This whole process was extremely time consuming and error-prone, while it also placed an extreme burden on enterprises.

The continuously increasing workload, the important delays in the process and of course, the understanding of the burden placed upon the enterprises for accurate and timely data reporting has guided NSSG towards the development of a pilot web-based application for online data collection directly from the enterprises.

More specifically, NSSG offers the ability to enterprises to declare their products (imports, exports) via a web-based system for Intrastat statements' declaration. This electronic system has operated successfully since January 2003 and it serves 38.335 registered enterprises and 4.044 accountants / accountant offices, submitting an

average of 14.600 declarations monthly. The most important characteristics of the system are as follows:

- Use of SSL protocol for all tasks that concern control or treatment of personal data.
- Suppression of popup windows in all the levels of application.
- Increment of records' limit for manual input up to 1.000 records per statement.
- Increment of records' limit for file upload up to 50.000 records per statement.
- Projection of statements' content up to 1.000 records with printing facility.
- Simplification of customers' management for the accountants with global login.
- Incorporation of confirmation processes in order to avoid common users mistakes.

A company or its accountant has the ability to declare products, to modify declarations of products and to delete declarations of products.

The wide acceptance of this application by its users, their continuous demand for further improvements and the fact that most enterprises have access to the Internet and use ready-made accounting software packages, in combination with the latest advances in the field of IT and communications technology has further guided the system development. The aim has been to automate the data collection process as much as possible and make it practically "invisible" for the respondents, thus reducing the burden placed upon them.

The National Statistical service of Greece wanted to offer a new intelligent method for electronic data collection and decided to use web services (the Intrastat web service) for this purpose. This web service consists of several methods which have the same functionalities as the electronic system of Intrastat statements' declaration.

Under this framework, NSSG in collaboration with RA-CTI, which undertook the task of the S/W development, planned and put into force the design and development of a prototype web service module for integrating the most popular commercial and accounting packages used by enterprises (in order to retrieve already available data from the legacy systems of the enterprise). In addition it aimed to improve the electronic data collection method, moving from the "push" towards the "pull" model. The whole project included:

- A Feasibility study consisting of market research, in order to:
    - Track down the most commonly used accounting packages;
    - Visit and collaborate with the software houses;
    - Define the variables used for Intrastat data collection, according to the related EU regulation;
    - Decide on the format of data;
    - Agree on the validation checks needed for quality data etc.
- The actual development of the prototype module that enables the extraction of data from the enterprises' warehouses and their submission via web directly to the production systems of NSSG.

- The integration of the data collection system with the production system. Untill the present time the upload of data from the database of the collection system (SQL Server) to the database of the production system (Oracle ver. 8i) is made manually (extraction of a ASCII flat file from the collection data base which is then uploaded to the production database using a loading form).

In brief, the main aim of the whole project and the idea lying behind it was to improve different aspects of the quality involved in the process, namely:

- Reduction of the burden on the respondents, especially for providers with large amounts of declarations and

- Improvement of the quality of the final statistical product (statistical data), in terms of different quality parameters.

More precisely, there are a number of parameters, which are used to evaluate the overall quality of the produced statistical data. These parameters include the following:

- Timeliness of results

- Improved accuracy

- Elimination of errors in raw data

- Time and space independence

- Control of non-response

- Cost reduction

- Burden reduction (for the respondents)

- Re-use of previous data

- Security

It is widely known that the quality of statistical data depends heavily on different facets of the statistical data production. In particular, the data collection phase has a tremendous impact. Consequently, the minimisation of the errors occurring during the data collection phase (data entry errors), the elimination of logical errors through checks and controls embedded in the collection system, together with the minimisation of the time interval from collection to publication and the ability to use readily available data directly from the source (IT systems of the respondent) are expected to significantly improve the accuracy and quality of the data transmitted to Eurostat and to all NSSG data users.

The feasibility study implemented at an early stage of the project demonstrated the benefits of using web services as an entry point for data upload, as well as the necessity for companies to perform this procedure in the easiest and quickest manner. The users of the service were involved in the process from an early stage, since they

participated in the feasibility study by providing answers to the feasibility questionnaire transmitted to all potential users. The questionnaire (transmitted via e-mail) gauged their level of interest in using this new method for INTRASTAT declarations. The next step, involved the analysis of the replies in order to highlight their needs, identify existing problems and track down the most commonly used software packages. 141 companies replied to our e-mails, the majority of which were medium-to-big enterprises and thus "heavy" users of the service. In appendix B an analysis of the received responses is presented.

3. Technology Background

The prototype module has been developed as a Web Service API, since Web Services allow different applications from different sources to exchange data without time-consuming custom coding. Additionally, since data transfer uses XML (W3C, 2003), Web services are not tied to any operating system or programming language. The National Statistical Service of Greece Web Service API uses the SOAP and WSDL standards to enable enterprises to use and integrate it with any commercial and accounting packages.

A web service is a software application on a network that has an interface through which other programs can gain access. They are currently being used to assist large and small entities to obtain the most from their IT resources by enabling the integration of diverse software applications, from desktop programs to large enterprise-wide systems. This is useful, not only for day-to-day operations, but is also particularly helpful in integrating systems after a merger or acquisition.

The keys to making web services work are data, process, and communication standards. The communication protocol standard is the same as the Internet, TCP/IP. The data standard is XML, a set of syntax rules for adding meaning to data and for building other XML standards. The process standards are actually a set of evolving XML standards: SOAP (Simple Object Access Protocol) [Box et al., 2000], for packaging messages from one software application to another, WSDL (Web Services Description Language) [Data, 2003; Christensen et al., 2001] for describing the web services processes in such a way that a software application can understand and UDDI (Universal Description, Discovery and Integration) [Clement et al, 2004; UDDI Spec Technical Committee Draft, http://uddi.org/pubs/uddi_v3.htm] for describing how to find and use an available web service.

The actual function of web services can be described from both a provider's and user's perspective. From the provider's perspective, a web service is created by using the data, process and communication standards identified above to create a web interface to one or more software applications. The majority of the web services, described above, provide data from a database in response to specific request parameters. In essence, a web service responds to a "get data" command by reading the data from a database and sending it back to a software application on the Internet. To actually create such a web service, the provider uses WSDL to define the allowable read access "get data" commands that the database management software can understand. The web service also knows how to put the results in a SOAP envelope addressed to the requesting software application and how to send it via the Internet.

From the user's perspective, a software application must be able to issue the appropriate commands, put them in a SOAP envelope, and send them to the web service interface for further processing. This usually requires the downloading of the WSDL and then plugging it into a software application.

Figure 1 depicts the typical Web Services Functional Model. This model consists of three entities: the *provider* of the service, the *user* that consumes the service and the *registry*. The provider register its services to the UDDI registry using SOAP/WSDL and waits for service requests. A user process queries the registry for a Web Service, using SOAP messages and retrieves information again as SOAP messages.

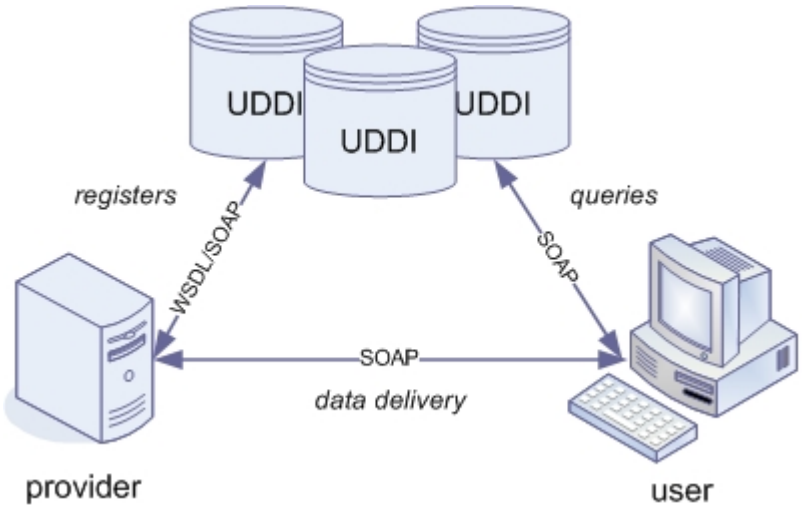

Figure 1. The Web Services Functional Model.

Implementing a Web Service for data interchange between two applications provides several advantages to the whole process. These include the following web services:

- provide interoperability between various software applications running on disparate platforms.
- use open standards and protocols. Protocols and data formats are text-based where possible, making it easy for developers to comprehend.
- can work, by utilizing HTTP, through many common firewall security measures without requiring changes to the firewall filtering rules.
- easily allow software and services from different companies and locations to be combined to provide an integrated service.
- allow the reuse of services and components within an infrastructure.
- can work through firewalls while other forms of Remote Procedure Calls may more often be blocked.

4. System Architecture

With regard to the specific web service built for NSSG, the basic web methods that constitute the Web Service are (Figure 2):

1. LoginUser (checks if the user has the rights to use the Web Service - Authentication Process. It also checks if the user is the accountant of a company or the company itself).

2. ReturnUserDetails (returns the details for an authorized user).

3. FindProduct (returns a full description of a Product).

4. ReturnCompanies (presents a list of the companies that an accountant is responsible for).

5. Retrieve (returns the Details for a selected Declaration, for later modification of the Declaration).

6. GetDeclarationByMonth (returns the declarations - from history - that have been submitted by the company for a specific month and a specific year).
7. GetDeclarationsAll (returns all the declarations - from history - that have been submitted by the company).

8. GetDeclarationDetail (finds and returns the details for a selected declaration).

9. GetDeclarationsByStatus (returns a list with the declarations of a Company order by status).

10. DeleteAllDeclaration (deletes all the rows of a selected declaration - resets the declaration).

11. DeleteARow (deletes a row, product, from a selected declaration).

12. InsertAProduct (inserts a row, product, into a specified declaration).

13. UpdateAProduct (updates the values of a selected row, product, from a declaration).

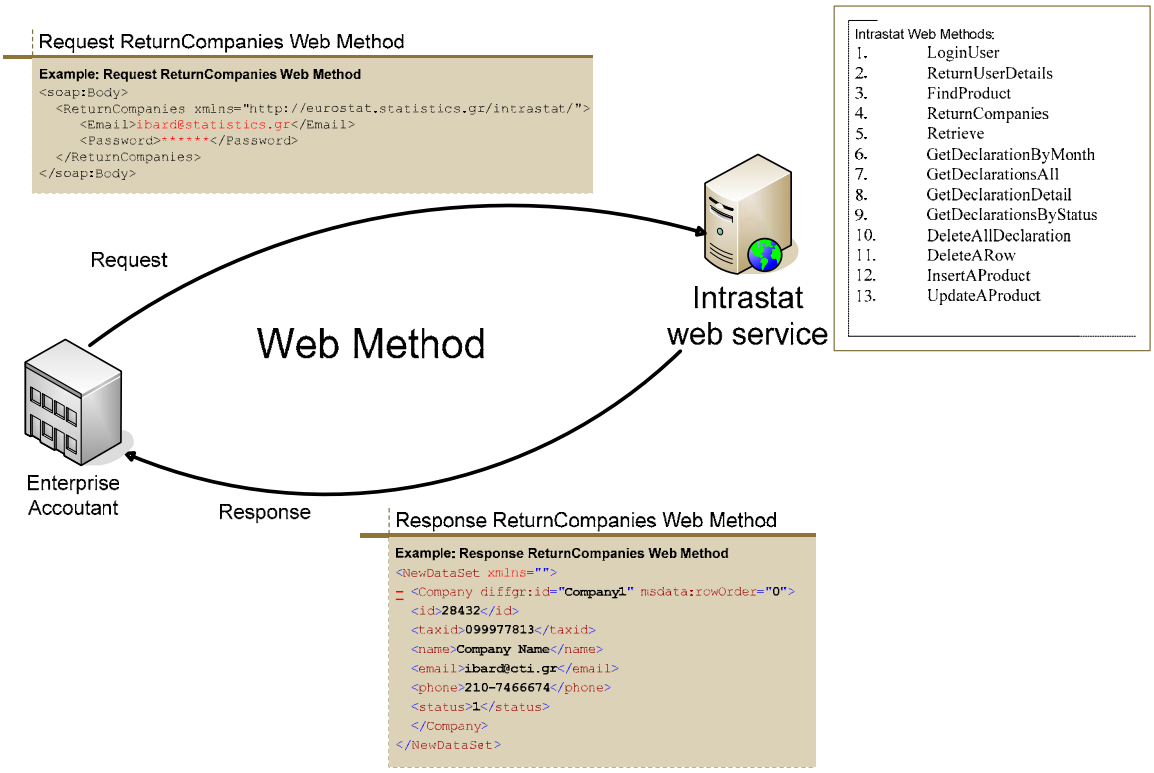

Figure 2. System architecture.

For example, the ReturnUserDetails web method does not accept any parameters, although it requires the authentication and authorization of the user. This method returns a dataset with a detailed description of an authorized user. If the user is a company, then the dataset is filled with details of the company. If the user is an accountant of more than one company, then the dataset is filled with the details of the accountant. The dataset consists of the following fields:

a. Name: The name of the user.

b. Address: The address of the user.

c. City: The city of the user.

d. Zip: The zip code of the user.

e. Phone: The phone number of the user.

f. Fax: The fax number of the user.

g. Email: The email address of the user.

h. TaxID: The VAT Number of the user.

i. Taxoffice: The code of the Tax Office of the user.

j. Password: The password of the user.

k. Responsible: The Responsible Person of the user.

l. Flag: The Type of the user (0 for company, 1 for accountant).

m. Status: The Status of the user (1 when access for the user is enabled, 0 when access for the user is disabled).

The code excerpt below presents an example response message for the ReturnUserDetails web method:

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://eurostat.statistics.gr/intrastat/">
-    <xs:schema    id="NewDataSet"    xmlns=""xmlns:xs=http://www.w3.org/2001/XMLSchema
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="el-GR">
- <xs:complexType>
- <xs:choice maxOccurs="unbounded">
- <xs:element name="User">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="name" type="xs:string" minOccurs="0" />
  <xs:element name="address" type="xs:string" minOccurs="0" />
  <xs:element name="city" type="xs:string" minOccurs="0" />
  <xs:element name="zip" type="xs:string" minOccurs="0" />
  <xs:element name="phone" type="xs:string" minOccurs="0" />
  <xs:element name="fax" type="xs:string" minOccurs="0" />
  <xs:element name="email" type="xs:string" minOccurs="0" />
  <xs:element name="taxid" type="xs:string" minOccurs="0" />
  <xs:element name="taxoffice" type="xs:string" minOccurs="0" />
  <xs:element name="responsible" type="xs:string" minOccurs="0" />
  <xs:element name="password" type="xs:string" minOccurs="0" />
  <xs:element name="flag" type="xs:unsignedByte" minOccurs="0" />
  <xs:element name="status" type="xs:unsignedByte" minOccurs="0" />
  <xs:element name="memo" type="xs:string" minOccurs="0" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:choice>
  </xs:complexType>
```

```
  </xs:element>
  </xs:schema>
-          <diffgr:diffgram          xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <User diffgr:id="User1" msdata:rowOrder="0">
  <name>Company Name</name>
  <address>Company Address</address>
  <city>ΚΗΦΙΣΙΑ-ΑΘΗΝΑ</city>
  <zip>14564</zip>
  <phone>210-8222222</phone>
  <fax>210-8222223</fax>
  <email>kmarkel@cti.gr</email>
  <taxid>094243464</taxid>
  <taxoffice>1159</taxoffice>
  <responsible> Accountant Name</responsible>
  <password>123456</password>
  <flag>0</flag>
  <status>1</status>
  </User>
  </NewDataSet>
  </diffgr:diffgram>
  </DataSet>
```

5. Case Study Implementation

Techniques have been implemented using the .NET v.1.1 framework technology and C# language [Banerjee et al., 2001] in particular. Generality is not lost as other Web Service supporting development platform may be utilized. The prototype is hosted on an MS Windows 2000 (Service Pack 4) Xeon at 3,2GHz using 2GB RAM. The quality database is designed and runs on an MS SQL Server 2000 sp4 instance. The particular development platform has been chosen mainly because of the authors' previous experience in the area.

The following usage scenario (Figure 3) describes how an enterprise or its accountant would make a declaration for a product, and how a developer would create a portion of a service.

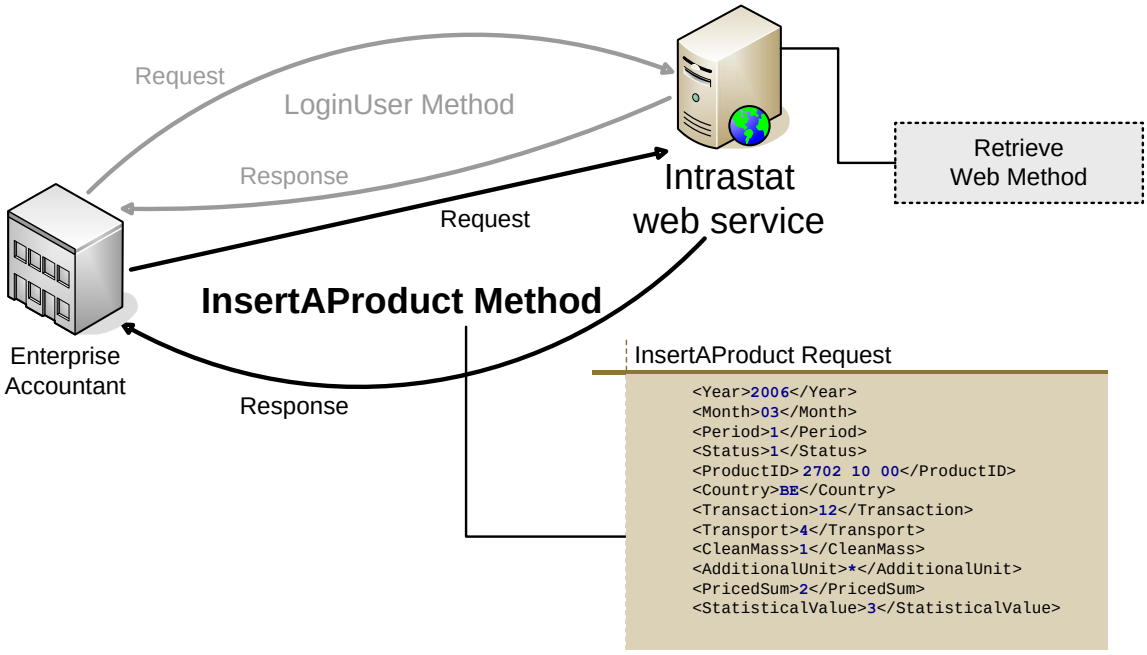

Figure 3. Overview of use case InsertAProduct Method using Intrastat web service.

Companies have to follow a number of steps in order to make a declaration that inserts a product using the Intrastat web service, namely:

1. The Company requests authentication using LoginUser web method.
2. The response authenticates the Company from the Intrastat web service (if the username and the password are correct).
3. The Company uses Retrieve web method in order to take some details (such as year, month, period, status) of the declaration that it is going to submit.
4. The Company requests the use of the InsertAProduct web method.
5. The response provides an answer “true” or “false”, as to whether or not the procedure of this declaration has been completed successfully.

Using a SOAP request message for the InsertAProduct web method the schema is as follows:

```
POST /intrastat/webservice/intraservice.asmx HTTP/1.1
Host: eurostat.statistics.gr
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://eurostat.statistics.gr/intrastat/InsertAProduct"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InsertAProduct xmlns="http://eurostat.statistics.gr/intrastat/">
      <Year>2006</Year>
      <Month>03</Month>
      <Period>1</Period>
      <Status>1</Status>
      <ProductID>2702 10 00</ProductID>
      <Country>BE</Country>
      <Transaction>12</Transaction>
      <Transport>4</Transport>
      <CleanMass>1</CleanMass>
      <AdditionalUnit>*</AdditionalUnit>
      <PricedSum>2</PricedSum>
      <StatisticalValue>3</StatisticalValue>
    </InsertAProduct>
  </soap:Body>
</soap:Envelope>
```

An example of SOAP response message for the InsertAProduct web method follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InsertAProductResponse xmlns="http://eurostat.statistics.gr/intrastat/">
      <InsertAProductResult>
        <xsd:schema>schema</xsd:schema>xml</InsertAProductResult>
    </InsertAProductResponse>
  </soap:Body>
```

```
</soap:Envelope>
```

The developer uses the identifier to retrieve a WSDL. The WSDL defines the message formats, data types, transport protocols and transport serialization formats that should be used between the requester agent and the provider agent. It also specifies one or more network locations at which a provider agent can be invoked and may also provide some information about the message exchange pattern that is expected. In essence, the service description represents an agreement governing the mechanics of interacting with that service. Then the developer creates an implementation of the service based upon the WSDL, tests it and deploys the implementation (makes a declaration that inserts a product) at the Intrastat web server.

6. Conclusions and Further Work

During the time this paper was prepared, the web service development was completed and the module was in the technical testing phase. As soon as this stage is over, the pilot operation phase will be launched, which will also involve the users (enterprises) participating in the feasibility study.

The feedback from the pilot phase will provide valuable input for the further improvement of the web service and will prepare the ground for the full scale operation.

The future steps include the identification of possible areas of expansion for the use of the service and the deployment of additional similar facilities for other statistical sectors, in order to support the NSSG policies for e-Government services to the citizens. Moreover, the use of standardised web services can be generalized in order to integrate the different processes that arise in the communication of autonomous actors (individuals or enterprises) with the various public information systems.

Appendix A

The Web Services framework

The Web Services framework consists of the following basic components and standards:

SOAP

Short for Simple Object Access Protocol, a lightweight XML-based messaging protocol used to encode the information in Web service requests and response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP.

WSDL

Short for Web Services Description Language, an XML-formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages. WSDL is an integral part of UDDI, an XML-based worldwide business registry. WSDL is the language that UDDI uses. WSDL was developed jointly by Microsoft and IBM.

UDDI

Short for Universal Description, Discovery and Integration. It is a Web-based distributed directory that enables businesses to list themselves on the Internet and discover each other, similar to a traditional phone book's yellow and white pages.

API

Abbreviation of Application Program Interface, a set of routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together.

XML

Short for Extensible Markup Language, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

Appendix B

The analysis of the responses are summarized in the following table:

Statistical Values of the Companies (Number of Companies:141)								
Year	**Flow**	**Value (via Tax Departments)**	**Value (via Web)**	**Total Value**	**Value of logistic Packages**	**% of Total Values**	**% of Values (via Web)**	**Number Company firms (via Web)**
2004	1(ARRIVALS)	8,124,396,169	16,112,843,894	24,237,240,063	2,933,902,199	12%	18%	8,699
2004	2(DISPATCHES)	2,743,357,513	3,732,913,201	6,476,270,714	251,423,939	4%	7%	1,967
2005	1(ARRIVALS)	6,402,628,028	15,776,795,358	22,179,423,386	2,979,046,317	13%	19%	9,592
2005	2(DISPATCHES)	2,477,939,191	4,191,257,266	6,669,196,457	383,870,654	6%	9%	2,543

Notes:

- Year: Year of the declarations.
- Flow: Flow the declarations.
- Value (via Tax Departments): The statistical values of companies' declarations, when they use the Tax Department for their declarations.

- Value (via Web): The statistical values of companies' declarations, when they use the Intrastat (via Web) for their declarations.
- Total Value: The total statistical values of companies' declarations.
- Value of logistic Packages: The statistical values of companies' declarations. These companies replied to our e-mails.
- % of Total Values: The percentage of the statistical values of companies' declarations (the companies replied to our e-mails) and the statistical values of all companies' declarations.
- % of Values (via Web): The percentage of the statistical values of companies' declarations (the companies replied to our e-mails) and the statistical values of companies' declarations (these companies make their declarations via Web).
- Number Company firms (via Web): The number of companies that make their declarations via Web.

In next figures, we can see the percentages associated with the statistical values of companies' declarations (the companies replied to our e-mails) and the statistical values of all companies' declarations. Additionally, itn is possible to observe the percentage of the statistical values of companies' declarations (the companies replied to our e-mails) and the statistical values of companies' declarations (these companies make their declarations via Web).

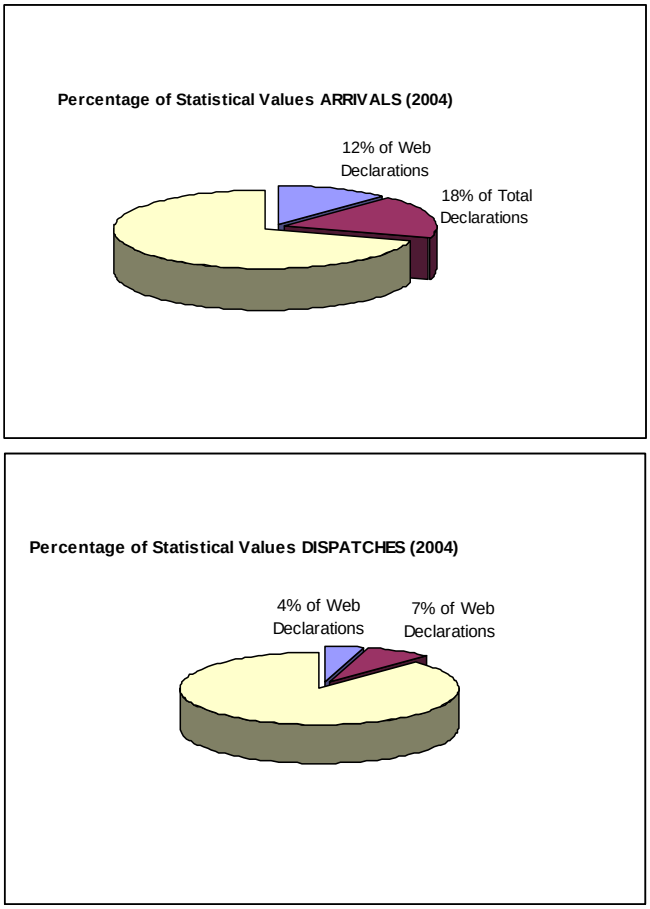

Figure B1. Percentage of Statistical Values Arrivals- Dispatches (2004).

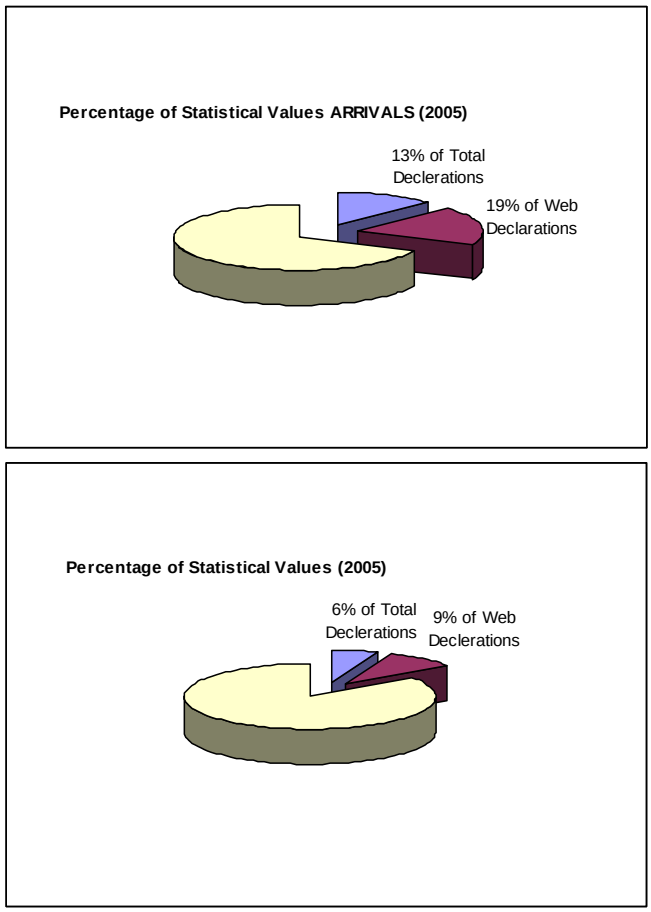

Figure B2. Percentage of Statistical Values Arrivals- Dispatches (2005).

It can be observed that a great percentage of the market place (18% for 2004 and 19% for 2005) use the Web for their declarations, which means that we should develop a new Service to enable them to make their declarations.

Additionally, after visiting the software houses and talking with developers, we decided upon the format of data, the validation checks needed for quality data that we should use in Web Service of Intrastat.

References

Banerjee, A., Corera, A., GreenVos, Z., Krowczyk, A., Maiani, B., Nagel, C., Peiris, C., and Thangarathinam, T. (2001). *C# Web Services: Building Web Services with .NET Remoting and ASP.NET*, Wrox Press, Birmingham.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S., and Winer, D. (2000, May). "Simple Object access Protocol (SOAP) 1.1", retrieved August 27, 2006 from http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

Christensen, E., Curbera, E., Meredith, G., and Weerawarana, S. (2001, March). "Web Services Description Language (WSDL) 1.1", retrieved August 27, 2006 from http://www.w3.org/TR/wsdl.

Data, R. (2003). "WSDL Tutorial", retrieved August 27, 2006 from http://www.w3schools.com/WSDL/default.asp.

European Commission, Official Site of the European Commission, http://ec.europa.eu/taxation_customs/customs/customs_duties/tariff_aspects/combined_nomenclature/index_en.htm, retrieved March 3, 2008.

Gardner, T. (2001, October). **"**An Introduction to Web Services", retrieved August 27, 2006 from http://www.ariadne.ac.uk/issue29/gardner/.

Clement, L., Hately, A., von Riegen, C., and Rogers, T. (2004). "UDDI Spec Technical Committee Draft", available at http://uddi.org/pubs/uddi_v3.htm

Panagopoulou G. (2004). "Data Quality Improvement", in *Proceedings of the CODACMOS EUROPEAN SEMINAR on The most effective ways to collect data and metadata: advanced strategies, tools and metadata systems*, Bratislava, 7-8 October 2004.

Riegen, C. (2002, July). "UDDI COMMITTEE.UDDI Version 2.03 Data Structure Reference", retrieved August 27, 2006 from http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm.

W3C (2003). "Extensible Markup Language (XML). World Wide Web Consortium", retrieved August 27, 2006 from http://www.w3.org/XML/