Interoperable Distributed Data Warehouse Components

Mohammed Awad[1] and Issam Jebreen[2]

[1] Software Engineering Department, University of Palestine, Gaza-Alzahra City, Gaza Strip, Palestine

[2] Software Engineering Department, Applied Science University, Amman City, Jordan

Abstract

Extraction, Transformation and Loading (ETL) are the major functionalities in data warehouse (DW) solutions. Lack of component distribution and interoperability is a gap that leads to many problems in the ETL domain, because these ETL components are tightly-coupled in the current ETL framework. Furthermore, complexity of components extensibility is another gap in the ETL area, because of the same tight-coupling reason. The missing extensibility feature causes impediments to add new components to the current ETL framework; to meet special business needs.

This paper discusses how to distribute the Extraction, Transformation and Loading components so as to achieve distribution and interoperability of these ETL components. In addition, it shows how the ETL framework can be extended easier. To achieve that, Service Oriented Architecture (SOA) is adopted to address the mentioned missing features of distribution and interoperability by restructuring the current ETL framework. Moreover, a Classified-Fragmentation component to enhance the report generation speed is added to the new framework as a proof of the extensibility concept. Therefore, this paper came out with a conceptual framework for interoperable distributed ETL components. This framework is defined to be a common ETL framework, which is valid for any ETL implementation that chooses this framework as a base. Moreover, the theoretical framework is validated by experts from industrial companies.

Keywords: *Data Warehouse, Distributed Components, ETL, SOA.*

1. Introduction

Data warehouses are complex systems employed to integrate the organization's data from several distributed and heterogeneous sources. The heterogeneous sources are located in different locations far from each other's [2]. Each location has its own specific infrastructure for the systems running in that location, for example, it has its own operating system and deployment infrastructure such as .NET, J2EE, or IBM mainframe. Each of those infrastructures needs a special ETL tool to be compatible with. Furthermore, each data source needs a complete ETL tool to be installed in the same location of the source, while sometimes only the Extract function is needed to extract data from this source. This results in a problem of an increase in the ETL licenses needed for a DW project and an increase in the complexity for the ETL user due to the redundant.

Sometimes, due to the complexity, long learning curve of the available ETL tools, and difficulty to achieve some extensibility in terms of additional functionalities; some organizations prefer to turn to in-house development to perform ETL tasks[2], which increases the cost and effort of the data warehouse project. Furthermore, tightly coupled components of software require teams of architects and designers to untangle the complex implications of change in the support of new business requirements or system enhancements[3]. As a result of that, tightly coupled components cause problems in terms of cost, maintenance, enhancement, and reusability of the ETL components. This leads to the necessity that the architecture of the software framework has to consider the component coupling factor and how much loosely should the components be coupled.

The current ETL framework lacks of components flexibility and loose coupling[4-9]. That results in complications to add new components to the ETL tools to support special business needs. For instance, although data warehouses provide an appropriate infrastructure for efficient querying, reporting, mining, and other advanced analysis techniques, the complexity of data warehouse environments specially the ETL framework is rising every day, and data volumes are growing at a significant pace, which makes report generation relatively slow due to the massive amount of data[2]. Data warehouse repositories based on one central server often suffer from either storage or computing bottlenecks, especially when complex aggregates need to be stored permanently or computed on demand. [10]refers to a high cost solution to the massive data problem, which is cluster-type systems with large numbers of worker nodes connected through high-speed LAN. In addition to the high expenses, integrating existing resources from several distant sites using this solution

needs a system that efficiently organizes these resources in a transparent manner, which sometimes is a challenge to implement. Some implementations of the ETL frameworks like "Pentaho Open Source Business Intelligence" include a fragmentation feature like Pentaho "Partitioning"; however, this feature does not classify data based on certain conditions to fulfill specific business needs. Furthermore, this fragmentation aims mainly to enable the fact and the dimension tables in the data warehouse to be separated among a cluster of servers[11]. That belongs to a physical (hardware) solution of the performance problem, and that solution is outside the scope of this research. Therefore, an extensible ETL framework with loosely coupled components can resolve this complication, because, a specific component can easily be added as an extension to the framework to resolve the performance problem.

Administration of ETL tools in many data source locations for the same project to extract data from many different sources needs extra administration, communication and maintenance effort[12], which is a problem resulted from the reality that the administrators are often different persons from one location to another and they could use different ETL tools and do different configurations to those tools. Furthermore, in the current ETL framework, there are impediments to include ETL as a part of a complete portal that manages the whole DW project[12], because of lack of standards in the current ETL framework to communicate with other components of the portal.

Based on that, there are gaps and missing features in the current ETL framework summarized by lack of: distribution, interoperability, loose coupling, extensibility, and reusability of the ETL components. These gaps lead to problems of the current ETL framework, which are: complexity of extending the ETL tools to suit special business needs, ETL administration complexity, an increase of effort needed to implement a DW project, impediments regarding ETL compatibility with different administrator environments, an increase of the cost to implement a DW project because of the increase of the number of ETL licenses needed and the extra effort needed to develop and use the ETL, and finally, the redundancy problem of including all the ETL features in every ETL administrator location due to the tightly coupled architecture of the available ETL framework.Therefore, a conceptual framework for ETL that includes the features of component distribution and interoperability addresses the problems highlighted in this section.

Based on the explored problems of ETL framework, an enhancement to the current ETL framework by including the features of component distribution and interoperability will address the extensibility problem.SOA is adopted in this research to address the mentioned missing distribution and interoperability features by restructuring the current ETL framework.

In this paper, as a proof of the framework extensibility, a Classified-Fragmentation component is added as an extension to the enhanced ETL framework to solve the relatively low speed report-generation of data warehouse projects. This component is important to some companies because data volumes are growing at a significant pace, which makes report generation relatively slow due to the massive amount of data. Some implementations of the ETL framework like Pentaho Open Source Business Intelligence include similar fragmentation components. However, the fragmentation feature of those implementations is tightly-coupled in the ETL tool and it is not based on distribution and interoperability standards. Furthermore, those types of fragmentations target mainly to enable the fact and the dimension tables in the data warehouse to be separated among a cluster of servers. That belongs to the physical (hardware) solution of the performance problem, but this paper concentrates on adding a software-based and loosely-coupled Classified-Fragmentation component to the ETL framework, as an extension to it; to prove the availability of the extensibility of the framework and to meet some special fragmentation needs of an organization.

2. Current ETL Framework

The traditional ETL framework has common tightly-coupled functionalities. Those functionalities, concepts behind them, and relationships between them can be concluded in one framework diagram as shown in Figure 1. In the data layer, the data stores that are involved in the overall process are depicted. On the left side, the original data providers (typically, relational databases and files) are shown. The data from these sources is extracted (as shown in the upper left part of Figure 1) by Extraction routines. Then, this data is propagated to the Data Staging Area (DSA) where it is transformed and cleaned before being loaded to the data warehouse. The data warehouse repositories are depicted in the right part of Figure 1 and comprise the target data stores. Eventually, the data loading to the central warehouse is performed through the loading routines depicted on the upper right part of Figure 1[4-7, 9, 13].

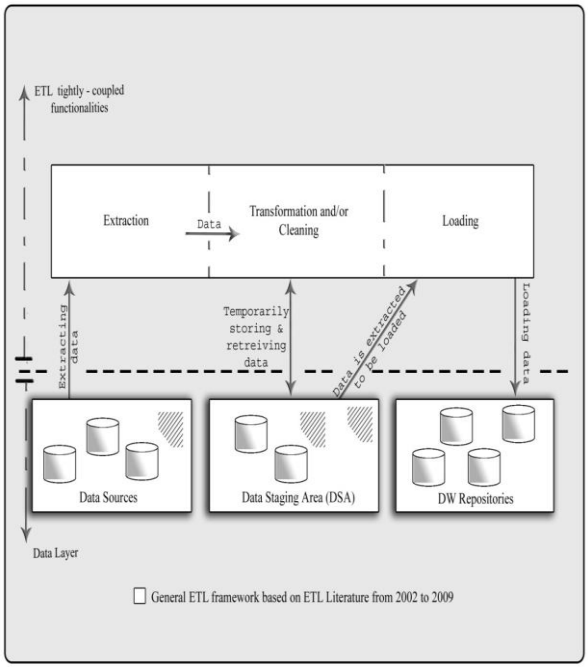

Fig. 1 Traditional ETL Framework

3. ETL Framework with Interoperable Distributed Components

This section briefs the enhanced ETL framework. Figure 2 illustrates this framework, which is based on SOA.In the data layer of Figure 2, the data stores are exactly similar to those available in Fig 1 of the traditional framework.

The business layer of Figure 2 that is built based on SOA framework; includes four main parts which are:

Service Orchestration Point (also called Directory Service or Service Registry): It describes the services available in its domain which are Extraction, Transformation, and Loading. Those three services are called Service Providers and register themselves in the Orchestration Point.

Service Providers: each of them is a component that performs a service in response to a customer request. The framework has three Service Providers which are Extraction, Transformation, and Loading services.

Service Consumers: each of them is a component that consumes the result of a service supplied by a provider. The main Service Consumer in the framework is the client which represents ETL administrators. In addition, the three Service Providers can be Service Consumers to other services. For example, the Transformation service can request some functions to be done by the Extraction service in case that the Extraction and the Transformation are executed in one patch.

Service Interface: it defines the programmatic access of the three services, and establishes the identity of the service and the rules of the service invocation.

The relationship between a Service Provider and Consumer is dynamic and established at runtime by a binding mechanism done by the Orchestration Point. This dynamic binding minimizes the dependencies between the Service Consumer and Service Provider. Indeed, that supports the loose-coupling feature of the framework.

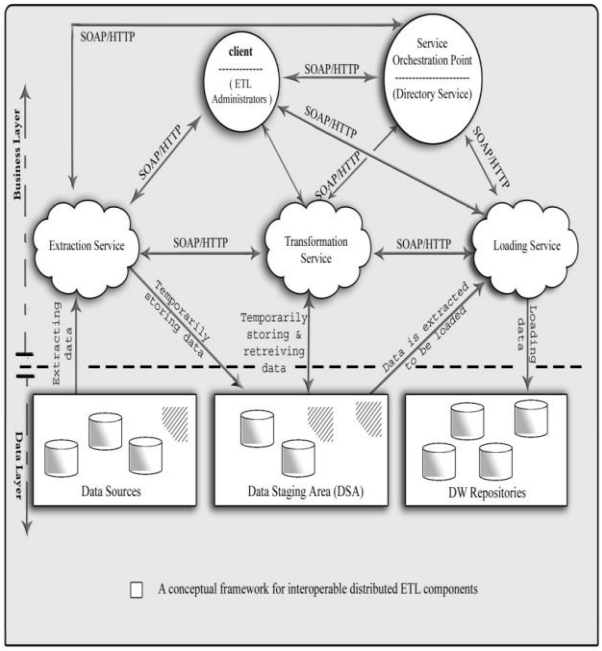

Fig. 2 A Framework for Interoperable Distributed ETL Components

In other words, Extraction, Transformation, and Loading functionalities are loosely-coupled by distributing them into interoperable web services. The Orchestration Point contains information about each service such as its interface. A client can discover services by examining the Orchestration Point. After looking up the required ETL service, the client continues remote communication directly with any distributed ETL service. Indeed, the client is the Service Consumer and the ETL service is the Service Provider. Further, an ETL service can also be a consumer to another ETL service. It discovers the availability of that service using the Orchestration Point.

A simple flow diagram is shown in Figure 3 and described below; to clearly show the flow of actions when a client demands an execution of an ETL functionality.

When a client demands to consume (execute) a certain ETL service, the Orchestration Point starts with a "receive" activity in which it receives the client request. Then, proceeds with invoking the suitable ETL service(s) and finishes by replying back to the client.

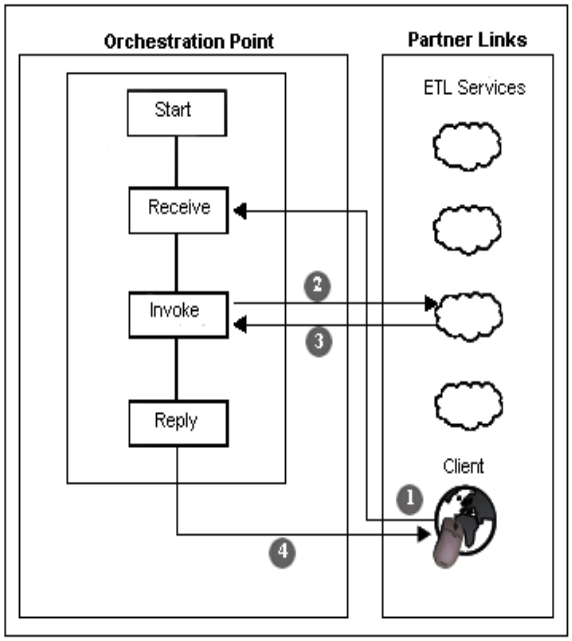

Fig. 3 Flow diagram for steps to consume an ETL service by a client [1]

An Orchestration Point Process typically interacts with one or more ETL web services (the Orchestration Point Process is also a web service). These ETL web services are called partner services or external service.

4. Adding a Classified-Fragmentation Component as an Extension to the restructured Framework

A Classification-Fragmentation component is added as an additional component to the enhanced ETL framework to speed up of the report generation, and to show the simplicity and flexibility in adding any new component as an extension to the restructured ETL framework, without affecting other components. That comes as a result to the distribution and interoperability features of the framework, which leads to loosely-coupled ETL components. In addition, it is proved that, it is easy to reuse, expand, extend, or add any new component to any loosely-coupled software framework [14, 15]. As shown in Figure 4, after adding the Classified-Fragmentation component, the framework is the same as Figure 2 except the new loosely-coupled Classified-Fragmentation component and its relations with other components and with the Orchestration Point.

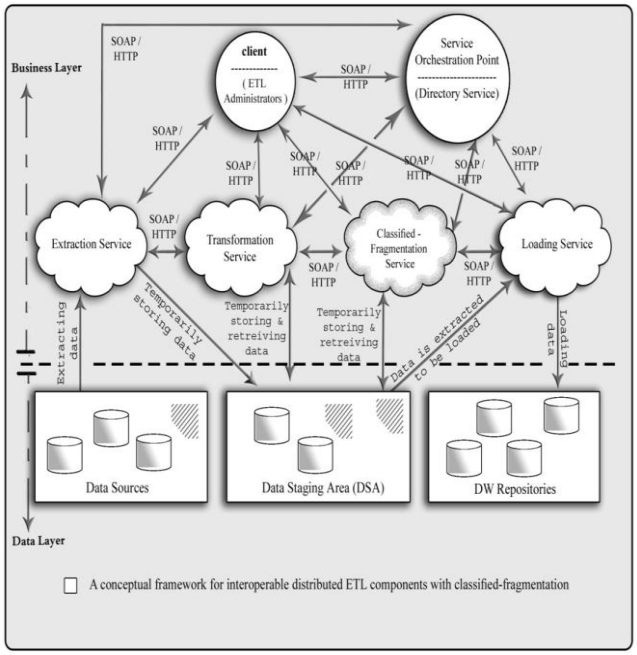

Fig. 4 A framework for interoperable distributed ETL components with classified - fragmentation

After doing the prototype of the Classified-Fragmentation component based on the framework of Figure 4, it is shown that by following the SOA concept, any other component can be added to the framework without any complications.

5. Prototype based on the new framework

This chapter has explored the analysis, design, development, deployment and testing of the SOA-based ETL prototype, considering the specifications of the theoretical framework for interoperable distributed ETL components. In addition to the basic programming functionalities that are developed to build the basic functionalities of the prototype, the development of the prototype included:

5.1 Business Process Execution Language (BPEL) Creation

As an implementation to the orchestration point, BPEL (Business Process Execution Language) is chosen to implement the orchestration point of the prototype.
Few years ago, BPEL has rapidly been emerged as a standard for combining a set of services into a number of discrete and long running enterprise processes[16, 17]. Most of industrial organizations are either using BPEL or planning to use it over their other middleware framework. NetBeans IDE is used to design and implement the required BPEL functionalities. In conclusion, a PBEL is designed and developed using NetBeans IDE and deployed

to a separate runtime environment for execution. This runtime is the OpenESB runtime that is integrated with the GlassFish application server.

Figure 6.5 shows the design of the BPEL orchestration point based on the new theoretical framework. On the left side, a client web service is depicted that is eligible for consumption by any ETL administrator application, while on the right side, four partner links for the four ETL web services are depicted. In the middle, the core business logic of the BPEL web service is depicted.

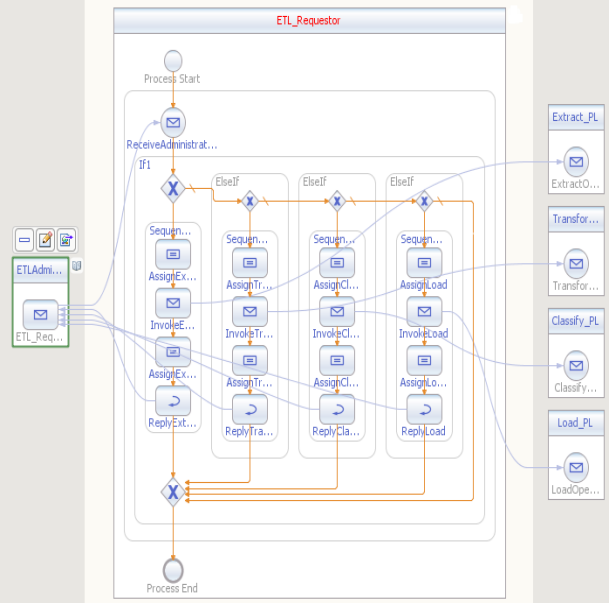

Fig. 5 Design of the BPEL orchestration point

In Figure 5, a client (ETL administrators) Web service can discover services by examining the BPEL. After looking up the required ETL service, the client continues communication remotely and directly with any distributed ETL service (Partner Service), in that case the client is the service consumer and the ETL service is the service provider. An ETL service can also be a consumer to another ETL service; in that case it discovers the availability of that service using BPEL service as well.

In more exploration, when a client demand to consume (execute) a certain ETL service, the BPEL starts with a receive activity in which it receives the client request. Then, proceeds with invoking the suitable ETL service(s) and finishes by replying back to the client. A BPEL process typically interacts with one or more ETL web Services. These ETL web services are called also partner services or external service.

5.2 Assembling Prototype Components in One Composite Application

Following the theoretical framework, all the components of the prototype are combined in one deployable composite application. In addition to the theoretical framework, SOA architecture recommends building loosely coupled applications and treating each one of them as independent "service units". Well-designed composite applications implement this architectural approach by providing an easy way to build business applications[16, 18]. They also provide integration of existing applications with other existing, as well as new applications. This SOA concept of linking together business processes is the hub of composite applications.

There are many tools available today that are used as editors or IDEs for the creation of composite applications. Out of these, NetBeans [19]and OpenESB runtime [20]compose proper IDEs for creating and editing the required composite application for this prototype. Figure 6 shows the design of the composite application for the prototype parts. In brief, it combines the BPEL and other components through the SOAP protocol, and opens a port for each partner link as shown in the same figure. The final composite application resulted from the design of Figure 6 is deployable in GlassFish application server.

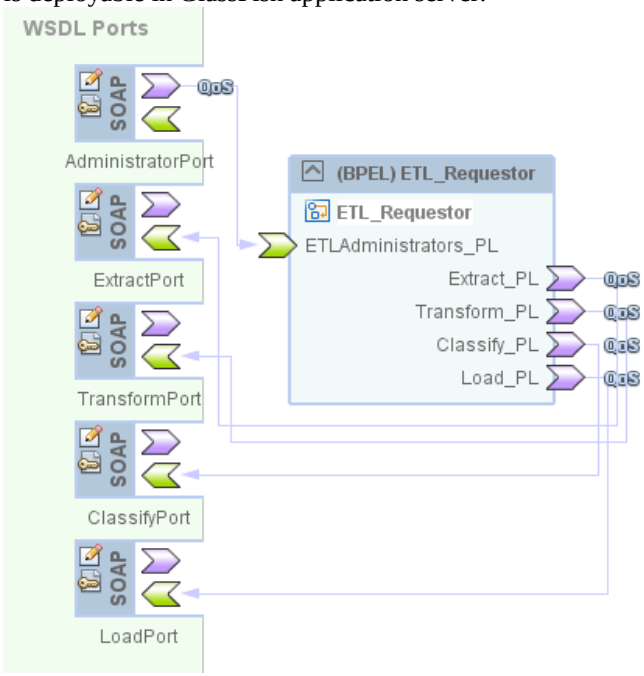

Fig. 6 Design of the Composite Application

5.3 Testing

The testing parts are done for the purpose of validating and verifying that the prototype meets the specifications of the framework to prove that the prototype works as expected, and can be implemented with the same required characteristics. The needed types of testing are: unit

testing, Web service tests, compatibility testing, classified-fragmentation speed and scalability testing, and end to end testing.

5.4 Unit Testing

Unit testing is a method by which individual units of source code are tested to determine if they are fit for use [21]. A "unit" in the prototype is the distributed ETL component that is wrapped in a Web service. GlassFish Tester [20] is used to test each Web service individually. The four main Web services passed the test. As a sample for testing results, a screenshot of the SOAP request and response of the Extraction component are shown in Figure 7.

Table 1: Time deference between Fragmented and Un-Fragmented data for report generation.

	No. of Records	No. of Users	Fragmented data (ms)	Un-fragmented data (ms)
1	100,000	1	188	297
2	100,000	2	192	207
3	100,000	3	255	364
4	200,000	1	359	515
5	200,000	2	369	520
6	200,000	3	369	643
7	300,000	1	516	688
8	300,000	2	531	719
9	300,000	3	625	815

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:ExtractOperation xmlns:ns2="http://prototype.phd.awad/">
            <extractParameter>true</extractParameter>
        </ns2:ExtractOperation>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:ExtractOperationResponse xmlns:ns2="http://prototype.phd.awad/">
            <return>Extract Service Invoked ! <p align='center'><b><font size='4'>
        </ns2:ExtractOperationResponse>
    </S:Body>
</S:Envelope>
```

Fig. 7 GlassFish Tester result for the Extract Web service

- *Classified-Fragmentation Speed and Scalability Testing*

To test the speed of report generation before and after using the Classified-Fragmentation component, and to test that the prototype is scalable for more than one concurrent ETL administrators, clinical data is used to generate a statistical report. The data warehouse repository of that clinic consists of a number of tables which are: (PATIENTS, PHYSICIANS, DEPARTMENTS, TESTOPERATIONS, DISEASES, MEDICINES, GENDER and CITY). These tables store data about patients, physicians, test operations, and other related data. The data is classified according to its type (video, text, or image), and then, the original tables are fragmented in which each table consists one type of data e.g. video, text, or image. This fragmentation process decreases the number of records in each table, which leads to a faster report generation.

A report generation speed test is done using Apache JMeter tool[22], which is a java desktop application designed to test software functions and performance, especially for web applications. Testing is done to calculate the time consumed to generate the report of Figure 8. This testing is done 9 times using fragmented data and 9 times using un-fragmented data, while different numbers of concurrent users are considered.

Disease/City	Alor Star	Baling	Changlun	Dong Dang	Dusun Kapas	Gua Tinggi	Jitra	Langkawi	Merbok	Tawar	Total
influenza	886	834	864	876	801	819	820	897	810	803	8410
heart disease	795	793	840	850	860	815	840	840	795	845	8273
headache	768	840	823	818	825	891	815	833	872	860	8345
hypertension	891	802	894	886	825	853	803	831	803	905	8493
disease1	1162	1053	1246	1124	1102	1162	1170	1145	1195	1126	11485
disease2	820	752	752	799	771	767	830	720	813	729	7753
disease3	736	750	790	805	792	812	785	782	775	747	7774
disease4	755	678	743	848	745	727	781	796	797	772	7642
disease5	731	764	708	852	781	813	778	751	811	724	7713
disease6	783	764	769	775	761	759	778	711	804	789	7693
Total	**8327**	**8030**	**8429**	**8633**	**8263**	**8418**	**8400**	**8306**	**8475**	**8300**	**83581**

Fig. 8 A statistical report generated from a clinical DW repository

Figure 8 report is generated using an amount of records from 100,000 to 300,000, and from one to three concurrent users. The time consumed for each of fragmented and un-fragmented data is shown in table 1 in milliseconds (ms).

As shown in table 1, it is clear that the time required to generate the report shown in Figure 8 in case of fragmented data is less than the time needed for un-fragmented data.

- *Compatibility Testing*

The compatibility testing is done to test the compatibility of the prototype with different application and Web servers, as well as with different browsers. Once the

prototype is developed using Java, a set of J2EE (Java 2 Enterprise Edition) application and web servers are used to test whether the prototype is compatible with them or not. Those servers are: GlassFish application server, Apache Tomcat Web server and JBoss application server. The prototype is deployed and run successfully on all of them. Furthermore, Microsoft Internet Explorer, Mozilla Firefox and Google Chrome browsers are used to test the browser compatibility, and the results show that it works exactly same with the three browsers.

- *End To End Testing*

End To End testing is used to validate the prototype starting from sending the request by the ETL administrator, passing through BPEL orchestration point and invoking the proper Web Service, and finally, finishing by executing the proper ETL functionality.

To do the End To End testing, GlassFish Tester is used to create a test case with an XML file as an input, then another XML file is auto generated as an output. The input file acts as an ETL administrator who needs to execute one of the four ETL functionalities, then the BPEL forwards the client request to the appropriate Web service according to the parameter included in "<etl:operationParameter></etl:operationParameter>" tag of the input XML file. For the current test case, "1" means "Extract", "2" means "Transform", "3" means "Classify", "4" means "Load". Figures 9 and 10 respectively are screenshots of the input and output XML files used in a test case done with parameter "1", i.e. the input XML file tag is: "<etl:operationParameter>1</etl:operationParameter>".

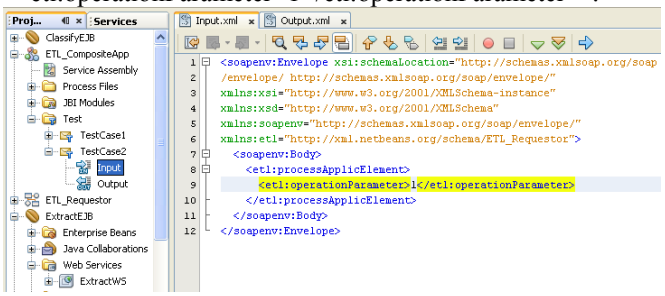

Fig. 9 Input.xml file (End To End test case input file)

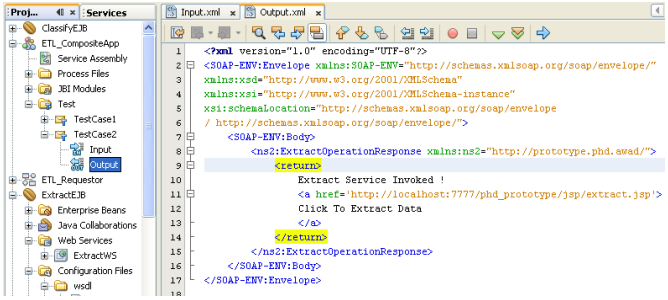

Fig. 10 Output.xml file (auto generated End To End test case output file)

In addition to the "GlassFish Tester" testing, the effect of executing each of the ETL prototype components on the data available in the DW repository tables is verified. For example, the transformation component transforms the date of birth data available in P_BIRTHDAY field of Figure 11 to age data available in P_AGE field of Figure 12, and instead of having the date of birth of the patient; it calculates his age. Figure 11 shows the source data before executing the transformation component, while Figure 12 shows the transformed data after executing the transformation component.

DISEASE_ID	MEDICINE_ID	Gender_ID	P_BIRTHDAY	CITY_ID
4	8	1	1941-03-27	10
9	8	1	1977-10-04	6
5	7	1	1942-09-03	9
7	4	0	1943-08-15	6
8	2	1	1989-09-17	3
10	1	1	1989-09-17	3
10	4	1	1989-09-17	3
8	6	1	1950-06-04	9
4	6	1	1950-06-04	9
7	1	0	1945-11-01	9
7	10	0	1945-11-01	9
7	5	0	1945-11-01	9
10	3	1	1997-07-05	3
4	9	1	1997-07-05	3
4	10	1	1997-07-05	3
7	6	0	1959-05-11	7
10	5	0	1959-05-11	7
4	9	0	1959-05-11	7
7	9	1	2007-04-04	3
10	8	0	1996-10-25	3
7	7	0	1996-10-25	3
9	5	1	1975-01-06	7

Fig. 11 Sample Data **before** Executing the Transformation Component

DISEASE_ID	MEDICINE_ID	Gender_ID	P_AGE	CITY_ID
4	8	1	70	10
9	8	1	34	6
5	7	1	69	9
7	4	0	68	6
8	2	1	22	3
10	1	1	22	3
10	4	1	22	3
8	6	1	61	9
4	6	1	61	9
7	1	0	66	9
7	10	0	66	9
7	5	0	66	9
10	3	1	14	3
4	9	1	14	3
4	10	1	14	3
7	6	0	52	7
10	5	0	52	7
4	9	0	52	7
7	9	1	4	3
10	8	0	15	3
7	7	0	15	3
9	5	1	37	7

Fig. 12 Sample Data **after** Executing the Transformation Component

6. Evaluation

In this section, the evaluation process of the SOA-based ETL prototype is briefed and discussed. The evaluation was done to measure experts' satisfaction of the prototype to substantiate that the features and specifications of the theoretical framework are implemented in this prototype. In addition, the evaluation measures experts' opinions regarding the prototype: usefulness, ease of use and ease of learning. Furthermore, the evaluation aims to find out if the problems identified in this research were successfully solved in this prototype, and the solution meets the theoretical framework specifications.

The evaluation is done using USE questionnaire [23]. USE stands for Usefulness, Satisfaction, and Ease of Use. Together with Ease of Learning, the four dimensions are used to design the questionnaire. The USE method is used to design a short questionnaire that is used to measure the most important dimensions of the prototype usability by industry experts, and to measure those dimensions across domains.

Forty Eight (48) experts were chosen to answer the questionnaire prepared for this evaluation process. Since the evaluation is used to measure much specified dimensions of the prototype, these evaluators were selected from the industrial companies that are specialized in data warehousing and/or adopt web services and/or SOA in their business solutions.

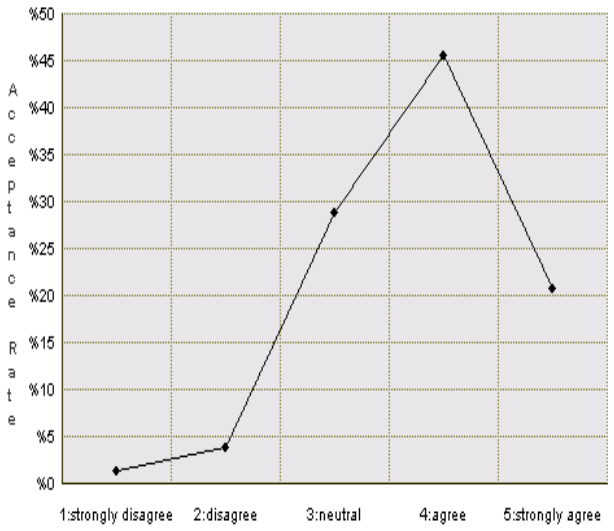

Fig. 13 Mean of Means

Figure 13 shows the mean of the four sections of the questionnaire (mean of the means of the four sections " Usefulness, Satisfaction, Ease of Use, Ease of Learning"). Overall 66.21% of the experts are satisfied with it. They rated 20.69% as "strongly agree" and 45.52% as "agree", while overall 28.8% of the experts have "neutral" opinions. Nevertheless, 5.09% of the experts are unsatisfied with the prototype.

7. Conclusions and Future Works

A goal of this research was to define and validate a conceptual framework for distributable and interoperable ETL components. The research succeeded in filling existing gaps of the current ETL framework regarding distribution and interoperability of ETL components. This research has identified the gaps and the problems of the current ETL framework. It solved the problems and filled the gaps by providing the new ETL framework solution. After that, a prototype validated the theoretical framework. Eventually, the work is evaluated and the evaluation results have supported the solution.

For the prototype, although many technologies can be used to implement an ETL tool based on the theoretical framework, all of the technologies used in implementing the prototype of this research are open source technologies. Open source technologies are chosen to make the prototype available freely for industry experts if they need to use it as a base for their future ETL tools that are based on the new ETL framework. Open source technologies used to implement the prototype are: Java, J2EE, MySQl, Apache Tomcat web server and GlassFish application server. In addition to that, they are freely available. Open source technologies are more flexible because the source code is available in case that there is a need to extend or enhance any part of the code of those technologies.

This research has distributed the components of the ETL framework. Sometimes, an ETL tool based on this framework and has distributed components; needs to share the same session variables to suit some implementation needs, specially, if an ETL tool is needed to act as a portal. To share the session variables among all the distributed components is quite a complicated issue and defers from one implementation to another of the new ETL framework. A potential future work is to standardize the session management of the distributed ETL components in a model that can act as a standard model for all implementations of the new ETL framework.

References

[1] D. Salter and F. Jennings, *Building SOA-Based Composite Applications Using NetBeans IDE 6*: PACKT Publishing, 2008.
[2] Massachusetts, *Introduction to the Data Warehouse*: Massachusetts, 2008.
[3] M. Barai, Binildas, and V. Caselli, *Service Oriented Architecture with Java*: Packt Publishing, 2008.

[4] Z. Zhang and S. Wang, "A Framework Model Study for Ontology-driven ETL Processes," *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference* 2008.
[5] V. Tziovara, P. Vassiliadis, and A. Simitsis, "Deciding the Physical Implementation of ETL Workflows," in *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, 2007.
[6] R. Kimball and J. Caserta, *The Data Warehouse ETL Toolkit*, second edition ed.: Wiley Publishing, Inc., 2004.
[7] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual modeling for ETL processes," in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*2002, pp. 14-21.
[8] A. Simitsis, P. Vassiliadis, M. Terrovitis, and S. Skiadopoulos, "Graph-based modeling of ETL activities with multi-level transformations and updates," *Lecture notes in computer science*, vol. 3589, p. 43, 2005.
[9] D. Skoutas, A. Simitsis, and T. Sellis, "Ontology-driven conceptual design of ETL processes using graph transformations," *Journal on Data Semantics XIII*, p. 120, 2009.
[10] P. Wehrle, M. Miquel, and A. Tchounikine, "A Grid Services-Oriented Architecture for Efficient Operation of Distributed Data Warehouses on Globus," *21st International Conference on Advanced Networking and Applications*, 2007.
[11] Pentaho, "Pentaho Business Intellegence," 2009.
[12] L. Wu, G. Barash, and C. Bartolini, "A Service-oriented Architecture for Business Intelligence," in *IEEE International Conference on Service-Oriented Computing and Applications(SOCA'07)*, 2007.
[13] P. Vassiliadis, A. Simitsis, M. Terrovitis, and S. Skiadopoulos, "Blueprints and measures for ETL workflows," *Lecture notes in computer science*, vol. 3716, p. 385, 2005.
[14] A. Brown, S. Johnston, and K. Kelly, "Using service-oriented architecture and component-based development to build web service applications," *interactions*, vol. 1, p. 2, 2003.
[15] S. Mulik, S. Ajgaonkar, and K. Sharma, "Where Do You Want to Go in Your SOA Adoption Journey?," *I E E E Comp u t e r S o c i e t y*, 2008.
[16] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*: Prentice Hall, 2005.
[17] B. Simon, B. Goldschmidt, and K. Kondorosi, "A Human Readable Platform Independent Domain Specific Language for BPEL," *Networked Digital Technologies*, pp. 537-544.
[18] E. Newcomer and G. Lomow, *Understanding SOA with Web Services (Independent Technology Guides)*: Addison-Wesley Professional, 2004.
[19] NetBeans, "NetBeans IDE," 2010.
[20] glassfish, "glassfish," 2010.
[21] W. T. Tsai, R. Paul, Y. Wang, C. Fan, and D. Wang, "Extending WSDL to Facilitate Web Services Testing," *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02)*, 2002.
[22] Apache, "Apache JMeter," 2010.
[23] A. M. Lund, "Measuring Usability with the USE Questionnaire," in *STC Usability SIG Newsletter*, 2001.

First Author He has B.Sc. in Computer Engineering, M.Sc. in Information Technology, and Ph.D. in Software Engineering and Data Warehousing. He won many medals worldwide. In addition to his research experience, he has 9 years industrial experience. Currently, he is the IT unit director and the dean of admission and registration at university of Palestine.

Second Author He has B.Sc. in CIS, M.Sc. in Information Technology, and Ph.D. in Software Engineering. In addition to his research experience, he has 4 years industrial experience. Currently, he is a lecturer at applied science university at department of software engineering.