RIGA TECHNICAL UNIVERSITY

Janis Kampars

ON DEMAND DATA INTEGRATION SOLUTIONS FOR REMOTE DATA SOURCES

Summary of Doctoral Thesis

Riga 2011

RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science and Information Technology
Institute of Information Technology

Janis KAMPARS
Student of Doctoral study program “Management Information Technology”

ON DEMAND DATA INTEGRATION SOLUTIONS FOR REMOTE DATA SOURCES

Summary of Doctoral Thesis

Scientific supervisor
Dr. sc. ing., professor
J. GRABIS

Riga 2011

UDK 004.78(043.2)
Ka 354 o

Kampars J. On Demand data integration solutions for remote data sources. Summary of Doctoral Thesis.-R.:RTU, 2011.-32 p.

Printed according to the decision of the board of RTU Institute of Information Technology, protocol No 11-07 of 6th July, 2011.

This work has been supported by the European Social Fund within the project "Support for the implementation of doctoral studies at Riga Technical University".

ISBN 978-9934-10-238-7

DOCTORAL THESIS
SUBMITTED FOR THE DOCTORAL DEGREE
IN ENGINEERING SCIENCE AT RIGA TECHNICAL UNIVERSITY

The defense of the thesis submitted for doctoral degree in engineering science will take place at an open session on December 14th, 2011 at 14:30 in Meza Street 1/3, auditorium 202.

OFFICIAL REVIEWERS:

Professor, Dr.habil.sc.ing. Leonids Novickis
Riga Technical University, Latvia

Professor, Dr.habil.sc.ing. Peteris Rivza
Latvia University of Agriculture, Latvia

Professor, Dr.sc.ing. Enn Õunapuu
Tallinn University of Technology, Estonia

AKNOWLEDGEMENT

I confirm that I have developed the doctoral thesis submitted for the doctoral degree in engineering science at Riga Technical University. This work has not been submitted to any other university for the doctoral degree.

Janis Kampars ……………………………(signature)
Date: ………………………

The dissertation is written in Latvian, contains and introduction, 5 chapters, a conclusion, bibliography, 26 tables, 75 figures, 5 appendixes, a total of 165 pages. The bibliography contains 174 references.

TABLE OF CONTENTS

1. GENERAL DESCRIPTION OF THE THESIS

The Doctoral thesis is devoted to investigation of on demand data integration from remote, heterogeneous data sources. This chapter of the thesis summary outlines research motivation, defines thesis's objective and tasks as well as research methods used. The main results, scientific novelty and practical importance and approbation are also presented.

1.1. Research motivation

Companies make various decisions on the regular basis. Decision making requires data located in different data sources. Data integration is the process of gathering and combining data from different sources and acquisition of a single clear data view [4]. This work concentrates on business intelligence data integration [9], which focuses on gathering and transforming data for analytical purposes. Data availability is essential to decision-making [8, 12, 20, 25].

In a traditional data integration scenario data is gathered from local enterprise systems. In this case typical data sources are enterprise controlled database management systems, flat files and spreadsheets. Data gathering and consolidation is performed using ETL (Extract, Transform, Load) systems.

Relatively low cost of data storage solutions and high availability of the Internet have allowed various organizations to gather large amounts of data and to provide others with access to it [26]. This leads to a wide range of potential external data sources. Traditionally, external data is first imported into a centralized data base, however this approach is hard to implement [27].

There is an alternative – on demand data integration using Data as a Service (DaaS) [5, 23, 24] based data integration solution. In this case, data sources are remote web services, which provide data retrieval and processing functionality. In this work, methods provided by remote web services, are referred as data retrieval operations as they return data. A DaaS based data integration solution has distributed architecture and constitutes a distributed system.

Development of DaaS based data integration systems is complicated because:

- traditional data integration and application integration solutions are not suitable for DaaS based data integration [3, 11];
- there is a large variety of protocols and standards [14, 15, 16, 21] used in web services;
- the data transferred is in textual, semi-structured format, however binary data can also be used [6, 16, 28];
- web service interfaces are dynamically changing [24];
- load balancing and error recovery must be implemented;
- web service quality has to be taken into account [2, 18, 22], however the available information can be incomplete [10];
- automated data source discovery based on functional and nonfunctional requirements is complicated [17, 18, 22, 27];
- web service and data legal aspects are ambiguous [23].

1.2. Objective and tasks

The objective of the thesis is to develop on demand data integration solutions for data retrieval from remote, heterogeneous data sources and transformation of data retrieved in the necessary form. To achieve this objective, the following tasks are defined:

1. identification of problems that complicate data integration from remote, heterogeneous data sources;
2. synthesis of general data integration system model for heterogeneous, remote sources;
3. definition of architecture requirements for data integration from heterogeneous, remote sources;
4. development of corresponding architecture;
5. architecture implementation;
6. evaluation of defined solution effectiveness;
7. review of possible architecture applications.

The **object** of the research is data integration solutions for remote, heterogeneous data sources, the **subject** is development and application of remote source on demand data integration architecture.

1.3. Research methods

To define requirements for remote, heterogeneous data integration system, literature and technical solution analysis is performed. High level abstraction of architecture model is described using UML (*Unified Modeling Language*) [1]. Using synthesis the model is built from different elements including the data retrieval process logic, load balancing, quality of service control and error recovery. Data processing is based on XML and related specifications. The prototype of the data integration system is developed using .NET framework. To compare the prototype with other data integration solutions, design of experiments [7] is used. Some of the experimental results are approximated with regression models.

1.4. The main result and scientific novelty

The main result of the thesis is development of the remote source on demand data integration system model and architecture. Its most important scientific contributions are:

1. Design of a new remote source data integration method, which is based on web service method level abstraction and separation of data integration process and data source access logic.
2. Development of an adaptive web service selection and load balancing algorithm, which is based on functional and nonfunctional requirements.
3. Development of an algorithm providing correct and timely execution of individual data integration tasks.
4. Effectiveness evaluation of solutions developed.

The developed algorithms provide individual data integration task parallelization and correct execution order, late binding and load balancing, error recovery. The architecture provides complete data integration process separation from web service access logic, which promotes reusability and facilitates maintenance.

The effectiveness of the developed data integration task parallelization algorithm is proved, the importance of the load balancing is confirmed, and the cases, when remote data sources are more effective than local ones, are identified.

1.5. Application

To prove feasibility of the architecture, system prototype was developed. Results of the work were used in solving express mail delivery planning problem in "Latvijas Pasts" (research was done as part of a project "Integration of global positioning technologies in enterprise information systems: Application in solving Latvian transportation planning problems"). The system prototype was modified for the use in "BalticTaxi" to solve a passenger transportation planning problem. The prototype was also used to solve an object location problem.

The system developed in this work can be applied in data gathering from remote, heterogeneous data sources (web services), transforming and aggregating it.

1.6. Approbation

The results of this research are published in 8 publications:

1. Bonders M., Grabis J., Kampars J. Web Service Selection: Beyond Quality of Service// 9th Conference on Databases and Information Systems (DB&IS). - Latvia, Riga: University of Latvia, 2010. - p.125-137.
2. Grabis J., Bonders M., Kampars J. Combining Functional and Nonfunctional Attributes for Cost Driven Web Service Selection Frontiers in Artificial Intelligence and Applications// Databases and Information Systems: Selected Papers from the Ninth International Baltic Conference, Db&is 2010. - 2011. - p.227-239.
3. Grabis J., Kampars J., Bonders M. A Methodology for Integration of Spatial Data in Enterprise Applications// International Business Information Management Conference (7th IBIMA). - Brescia, Italy: IBIMA, 2006. - p.169-175.

4. Kampars J. Globālās pozicionēšanas datu integrēšana uzņēmuma informācijas sistēmā// a/s DATI Exigen Group informācijas tehnoloģijas speciālistu un Latvijas universitāšu datorzinātņu studentu XI konference. - Riga: DATI Exigen Group, 2006. - p.51-56.
5. Kampars J. New Generation Enterprise Geographic Information Systems// 9th International Conference "Environment. Technology. Resources" - Rezekne: Rezekne Higher Education Institution, 2009. - p.235-240.
6. Kampars J., Grabis J. Development of adapter for connecting GPS/GIS and ERP systems// RTU zinātniskie raksti, Datorzinātne. - 2006. - p.51-56.
7. Kampars J., Grabis J. Enterprise Application Integration problems, approaches and standards// RTU zinātniskie raksti, Datorzinātne. - 2007. - p.77-83.
8. Kampars J., Grabis J. Spatial Data Integration Approach with Application in Facility Location// 16th International Conference on Information and Software Technologies. - Kaunas, Lithuania: Kaunas University of Technology, 2010. - p.117-125.

The results obtained were presented in 6 conferences:

1. RTU 47th International Scientific Conference, October 13, 2006, Riga, Latvia.
2. International Business Information Management Conference (7th IBIMA), 2006, December 14-16, Brescia, Italy.
3. RTU 48th International Scientific Conference, October 12, 2006, Riga, Latvia.
4. 9th International Conference "Environment. Technology. Resources", 2009, June 25-27, Rezekne, Latvia.
5. RTU 50th International Scientific Conference, October 16, 2009, Riga, Latvia
6. 16th International Conference on Information and Software Technologies, 2010, April 21-23, Kaunas, Lithuania.

The results of the dissertation were used in a research project “Integration of global positioning technologies in enterprise information systems: Application in solving Latvian transportation planning problems”.

1.7. Thesis structure

The dissertation contains an introduction, 5 chapters, conclusion, 5 appendixes and bibliography.

Introduction motivates the topicality of the research problem, formulates the goal of this work and defines tasks. Research methods, scientific novelty, application and approbation are also described.

The first chapter opposites traditional and DaaS based data integration and identifies the main differences. The main problems, which complicate the use of remote web services as data sources, are reviewed. Related researches in the area of remote source data integration are reviewed.

The second chapter defines data integration system model. The model is platform and technology independent.

The third chapter is devoted to defining requirements for technical architecture and developing appropriate architecture. The components of the architecture, their interaction and operating principles are described in detail.

The fourth chapter compares DaaS based approach with local data storage. Prototype is developed to estimate the effectiveness of previously defined architecture and compared with commercial ETL system and sequential data retrieval solution. The effect of load balancing on data retrieval time is also evaluated.

The fifth chapter describes two application examples. The architecture is used to solve passenger transportation planning problem and object location problem.

Conclusion summarizes obtained results and defines further research directions.

The dissertation contains 5 appendixes:

1. appendix – database model;
2. appendix – user interface for passenger transportation application example;
3. appendix – data integration process logic for passenger transportation planning application example;
4. appendix – user interface for facility location application example;
5. appendix - data integration process logic for facility location application example.

2. SHORT SUMMARY OF THESIS CHAPTERS

This chapter contains a short summary of the thesis content while focusing on results and conclusions. The main problem addressed in this work is data integration from heterogeneous, remote sources (Figure 2.1.)

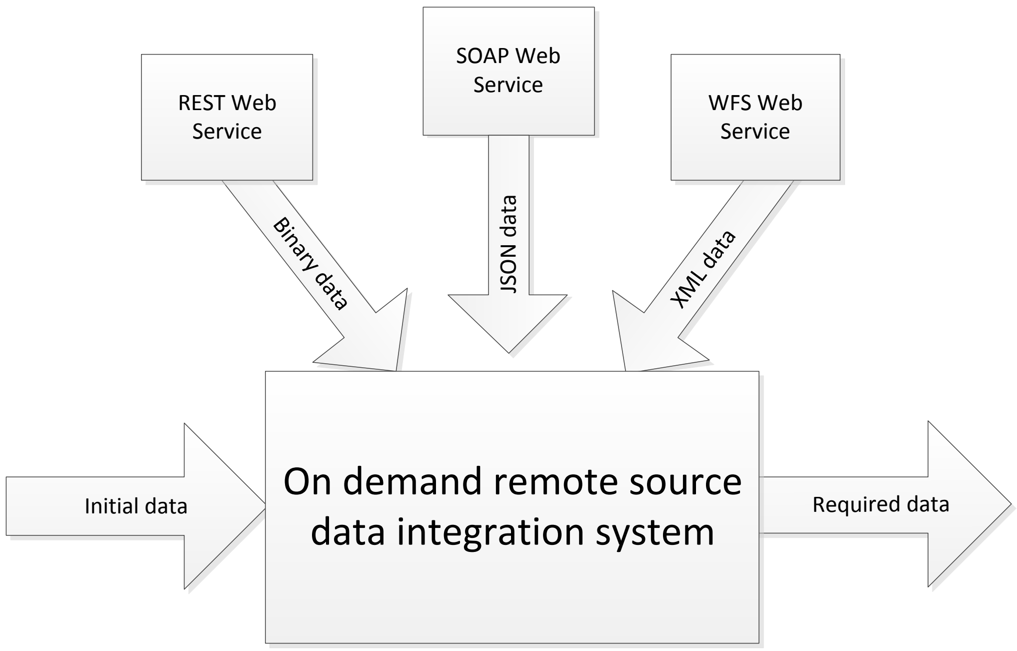


Figure 2.1. Remote source data integration problem

There is a limited amount of the initial data and information about the required data for decision making, which is gathered on demand and transformed by a corresponding system. The heterogeneous nature of data sources, different data formats, models and provided functionality must be taken into consideration. Although data sources are technically heterogeneous, they can be functionally equal and provide the same information.

2.1. State of the art and main challenges

The objective of the first chapter of the thesis is to identify the main problems complicating development of DaaS based data integration solutions. Initially, the main differences between the DaaS based approach and local data storage are identified on the basis of the literature analysis (Table 2.1).

Data integration from DaaS based sources is more complex than usage of local data sources. The main problems are data source heterogeneity and interface variability [24], the lack of suitable data integration solutions and appropriate methodological support.

Table 2.1

DaaS based data storage and locally stored data

Local data	**DaaS approach**
Data sources can be adapted for specific needs	Data sources can't be adapted for specific needs
Relatively static data sources	Dynamically changing data sources [24]
Mainly structured data	Mainly semi-structured data
Data is transferred over local network, Internet connection is insignificant	Data is transferred over Internet, significance of Internet connection speed and stability
Low risk of error during data integration process	High risk of error during data integration process
Terms of use are well known	Terms of use are ambiguous [23]
Potential data sources and their data models are well known	Potential data sources are unknown, searching for data sources is problematic [17, 18, 22, 27] and available metadata can be incomplete [10]
Unlikely existence of multiple data source alternatives	Many possible data source alternatives, quality of services [2, 18, 22] must be considered
Load balancing is implemented in the data source, no restrictions for load balancing algorithms	Load balancing is implemented in data integration solution, only centralized load balancing can be used
ETL systems deal with data source heterogeneity, there are also various adapters available	Traditional data integration solutions are not suitable [11, 27]

There are only a few works in the area of distributed data integration, which concentrate on using remote web services as data sources. Some of them are based on assumption, that there is an option to adapt web services for data integration purposes, which won't be possible in most cases. There is too little focus on individual data integration task correct and timely execution, load balancing, minimization of data integration time, consideration of nonfunctional requirements and promotion of reusability.

2.2. Data integration system model

In order to address the identified on demand data integration problems, a generic data integration system model was developed in Section 2 of the thesis. It generally describes retrieval of required data from heterogeneous, remote sources. The general form of the model is:

IM={**DA**, **AO**, **AL**, **ID**, **OD**, **P**, **Q**, **LB**}, where

DA – data sources, **AO** – abstract data retrieval operations, **AL** – data source abstraction layers, **ID** – initial data , **OD** – necessary data, **P** – data integration process, **Q** – quality of web services and data provided, **LB** – late binding and load balancing.

The goal of the data integration system model is to solve the data integration task while minimizing the time needed for data retrieval and facilitating maintenance of the data integration solution. The key principle used in the model is separation of data integration logic from data source access logic. To achieve that, an abstraction approach is used – abstract data retrieval operations (**ao**) and their corresponding input (**oim**), output and error message (**oem**) data models are defined. Web service (**da**) methods (**m**) are mapped to abstract data retrieval operations using the abstraction layers (**al**).

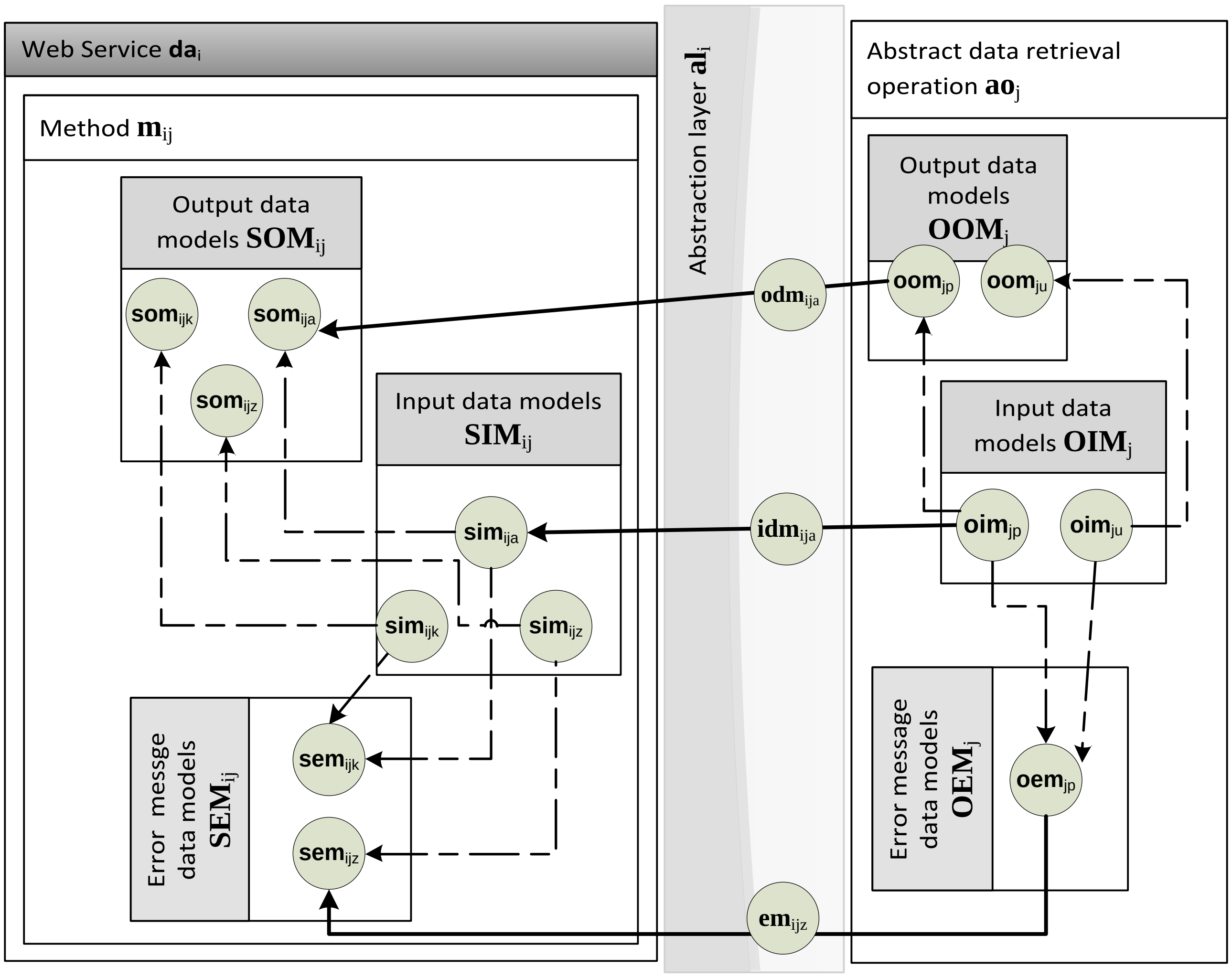
The abstraction layer in action is shown in Figure 2.2. The example shows, how the abstraction layer maps specific web service method input (**sim**), output (**som**) and error (**sem**) message data models to corresponding abstract data retrieval operation data models.

Data usually is retrieved from multiple sources and interdependencies among individual data integration tasks are defined by a data integration process **P**. The process starts with the set of initially available data **ID** and contains all data integration tasks, which should be performed to obtain the necessary data **OD**. The data integration tasks are executed in a specified sequence. At the beginning of the data integration process, the initial data is copied into a temporary dataset. During the process, the temporary data set is iteratively updated until all data has been gathered and the necessary data set is obtained. At the end of the process temporary data set contains all of the necessary data.

Five different task types are defined, that are used to realize data integration process logic:

- execution of abstract data retrieval operations;
- transformations;
- loops;

- conditional expression evaluations;
- data aggregations.



Web Service $\mathbf{da}_i$
Method $\mathbf{m}_{ij}$
Output data models $\mathbf{SOM}_{ij}$
$\mathbf{som}_{ijk}$
$\mathbf{som}_{ija}$
$\mathbf{som}_{ijz}$
Input data models $\mathbf{SIM}_{ij}$
$\mathbf{sim}_{ija}$
$\mathbf{sim}_{ijk}$
$\mathbf{sim}_{ijz}$
Error messge data models $\mathbf{SEM}_{ij}$
$\mathbf{sem}_{ijk}$
$\mathbf{sem}_{ijz}$
Abstraction layer $\mathbf{al}_i$
$\mathbf{odm}_{ija}$
$\mathbf{idm}_{ija}$
$\mathbf{em}_{ijz}$
Abstract data retrieval operation $\mathbf{ao}_j$
Output data models $\mathbf{OOM}_j$
$\mathbf{oom}_{jp}$
$\mathbf{oom}_{ju}$
Input data models $\mathbf{OIM}_j$
$\mathbf{oim}_{jp}$
$\mathbf{oim}_{ju}$
Error message data models $\mathbf{OEM}_j$
$\mathbf{oem}_{jp}$

Figure 2.2. The operational principle of the abstraction layer

To execute an abstract data retrieval operation, a suitable web service method is found and executed using the corresponding abstraction layer. Not all web services that are mapped to a certain abstract data retrieval operation support all of its input data models, so while searching for alternative data sources input data model must be considered. Nonfunctional requirements consisting of minimal acceptable measures of web service quality, weights of quality measures and the maximum number of web services used for a single abstract data retrieval operation load balancing must also be taken into account. The web service quality measurements are updated upon invoking the web service.

The model is a high level abstraction and is platform, programming language and technology independent. It provides effective remote source data integration and defines the main parts of the data integration system and their structure.

2.3. Architecture of the data integration system

Chapter 3 of the thesis elaborates the architecture of the data integration system. The architecture defines remote source on demand data integration system components and their relations. It is used as a basis for implementation of the data integration system.

According to the trends in the area of distributed systems and enterprise application integration and in order to address the most common heterogeneous data integration problems identified in the literature analysis, the following requirements for the data integration architecture are defined:

- data source heterogeneity encapsulation into data integration solution;
- support for different data formats and access protocols;
- definition of the data integration process as an executable business process;
- load balancing and error recovery;
- promotion of reusability and late binding;
- consideration of functional and nonfunctional requirements.

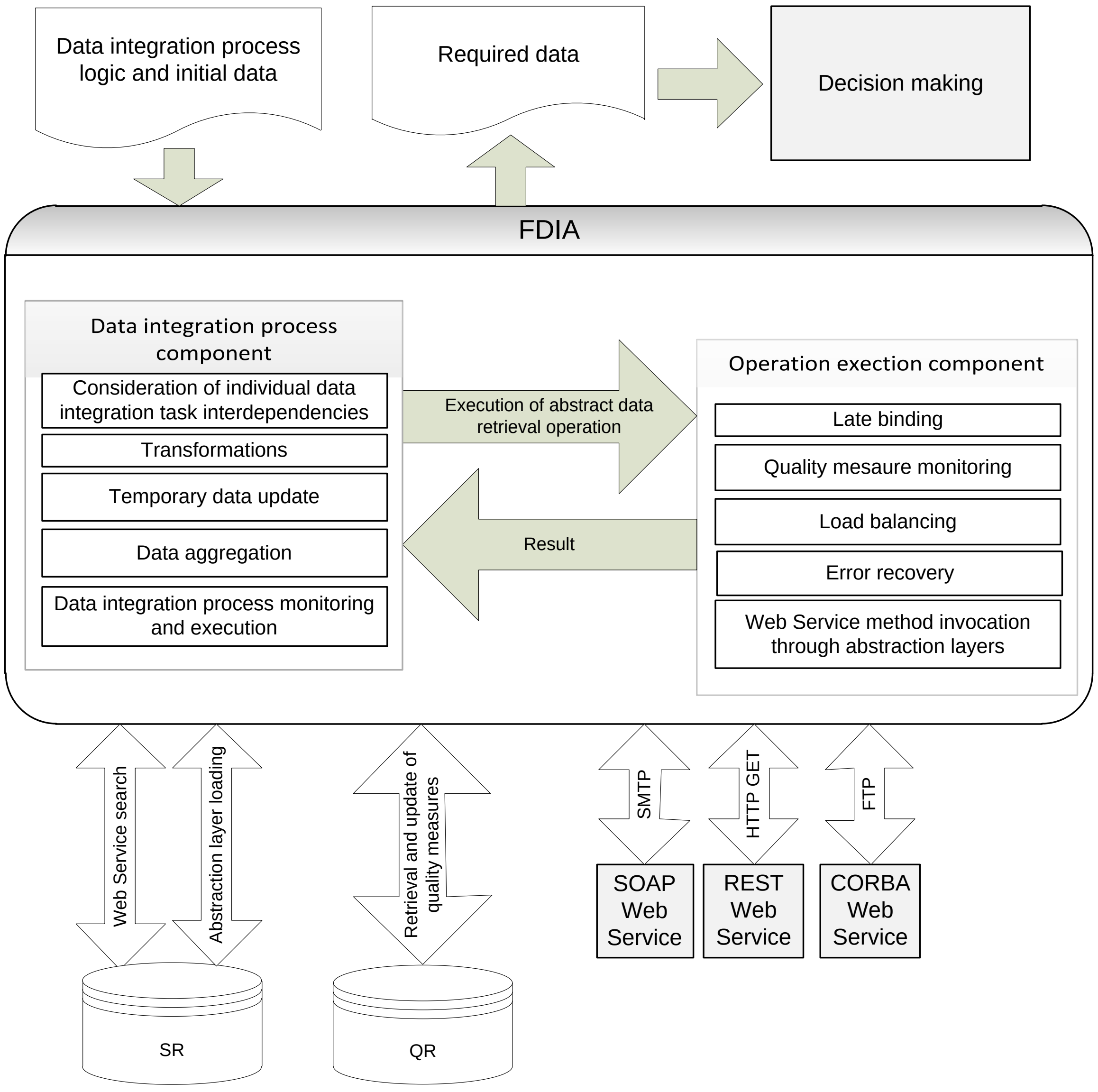
The architecture (Figure 2.3) consists of Service Register (SR), Quality Data Repository (QR) and Functional Data Integration Adapter (FDIA).

SR stores information about abstract data retrieval operations, their data models and abstraction layers, which provide web service mapping to abstract data retrieval operations. Data models are defined as XML schemas.

Quality of web services and their data is stored in QR. Quality measures stored in QR are monitored and updated during the data integration process. QR allows effective load balancing and web service selection based on nonfunctional requirements. The main part of the architecture is FDIA, which receives XML based initial data and data integration process logic and returns the required data.

The data integration process logic can represent complex individual data integration task interdependencies and it contains:

- minimal allowable quality measures for used web services;
- weights of quality measures;
- a set of tasks forming the data integration process;
- task interdependencies and correct execution order;
- maximum number of web services used for load balancing within single abstract data retrieval operation.



Data integration process logic and initial data
Required data
Decision making
FDIA
Data integration process component
Consideration of individual data integration task interdependencies
Transformations
Temporary data update
Data aggregation
Data integration process monitoring and execution
Execution of abstract data retrieval operation
Result
Operation exection component
Late binding
Quality mesaure monitoring
Load balancing
Error recovery
Web Service method invocation through abstraction layers
Web Service search
Abstraction layer loading
Retrieval and update of quality measures
SMTP
HTTP GET
FTP
SOAP Web Service
REST Web Service
CORBA Web Service
SR
QR

Figure 2.3. On demand remote source data integration architecture

At the beginning of the data integration process, initial data is copied into the temporary data document. The temporary document contains input data for abstract data retrieval operations and is iteratively updated until it contains all of the necessary data.

FDIA is divided into Data Integration Process (DIPC) and Operation Execution (OEC) components. The structure of DIPC is shown in Figure 2.4. The class diagrams of other components are also developed and they are given in the third chapter of the thesis. DIPC is responsible for monitoring of integration process execution, input data transformation according to specific abstract operation input data models, realization of loops, evaluation of conditional expressions and data aggregation. The data integration process is defined using blocks and their configuration parameters. The whole data integration process is formed by data integration task interexchange between blocks.

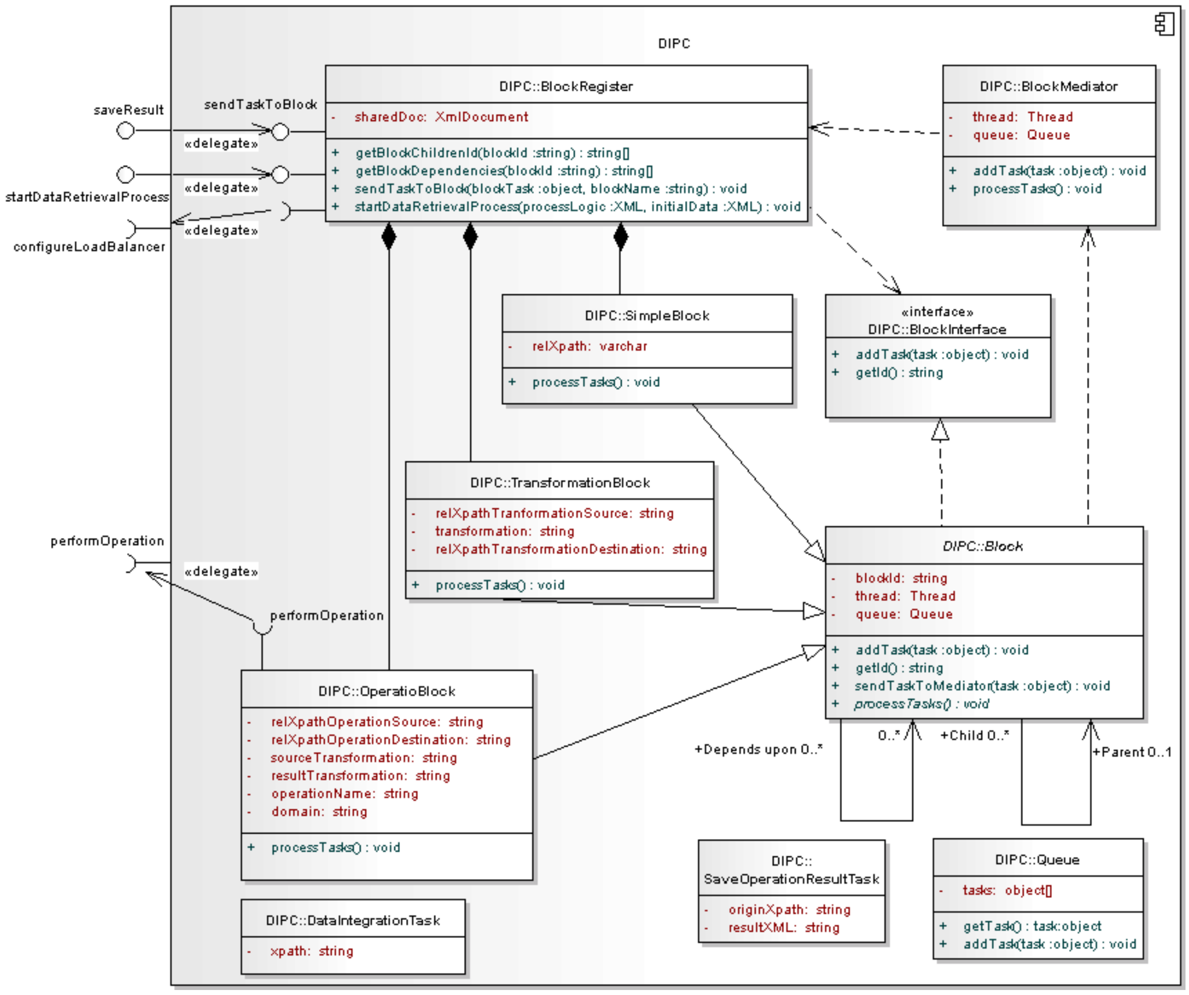

DIPC
DIPC::BlockRegister
- sharedDoc: XmlDocument
+ getBlockChildrenId(blockId :string) : string[]
+ getBlockDependencies(blockId :string) : string[]
+ sendTaskToBlock(blockTask :object, blockName :string) : void
+ startDataRetrievalProcess(processLogic :XML, initialData :XML) : void
DIPC::BlockMediator
- thread: Thread
- queue: Queue
+ addTask(task :object) : void
+ processTasks() : void
saveResult
sendTaskToBlock
«delegate»
startDataRetrievalProcess
configureLoadBalancer
DIPC::SimpleBlock
- relXpath: varchar
+ processTasks() : void
«interface»
DIPC::BlockInterface
+ addTask(task :object) : void
+ getId() : string
DIPC::TransformationBlock
- relXpathTranformationSource: string
- transformation: string
- relXpathTransformationDestination: string
+ processTasks() : void
performOperation
DIPC::Block
- blockId: string
- thread: Thread
- queue: Queue
+ addTask(task :object) : void
+ getId() : string
+ sendTaskToMediator(task :object) : void
+ processTasks() : void
DIPC::OperatioBlock
- relXpathOperationSource: string
- relXpathOperationDestination: string
- sourceTransformation: string
- resultTransformation: string
- operationName: string
- domain: string
+ processTasks() : void
+Depends upon 0..*
0..*
+Child 0..*
+Parent 0..1
DIPC::SaveOperationResultTask
- originXpath: string
- resultXML: string
DIPC::Queue
- tasks: object[]
+ getTask() : task:object
+ addTask(task :object) : void
DIPC::DataIntegrationTask
- xpath: string

Figure 2.4. DIPC structure

To solve data integration task interdependencies a block mediator (BM) class is defined. Data integration tasks created by blocks are sent to BM, which verifies if the task can be further sent to its recipients for execution. Block receives a data integration task only when there are no blocking task dependencies. The possible data integration task states are defined as idle, created, sent, processing, finished and error.

The architecture contains three different block types:

- Simple Block (SB) - used to create loops and evaluate conditional expressions.
- Transformation Block (TB) - document transformation services.
- Operation Block (OB) - abstract data retrieval operation execution.

When an abstract data retrieval operation execution is initiated in OB, OEC is called asynchronously to search for suitable web services in SR and to execute a specific web service method through the abstraction layer. This process also contains error recovery, load balancing and update of web service related quality data.

In some cases it is reasonable to virtualize data source in the abstraction layer (Figure 2.5). The data source can be considered virtual, because from architectural point of view there is no difference from usage of physical web services, however data processing logic is implemented in the abstraction layer and requests to data sources are not performed.

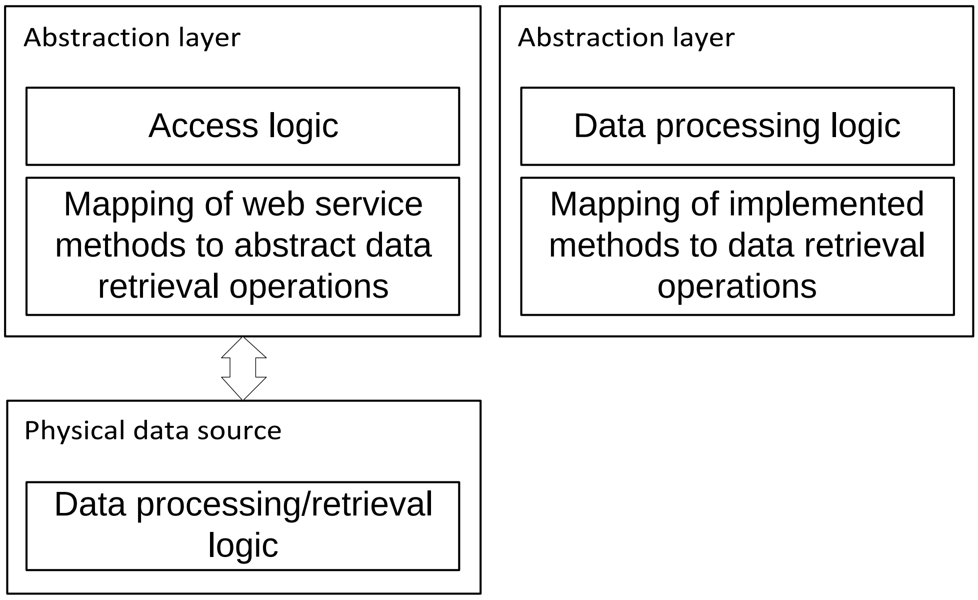

Abstraction layer
Access logic
Mapping of web service methods to abstract data retrieval operations
Abstraction layer
Data processing logic
Mapping of implemented methods to data retrieval operations
Physical data source
Data processing/retrieval logic

Figure 2.5. Virtual data source

FDIA functionality is made available through application programmable interface, thereby allowing automation of data integration process and integration with other systems.

The architecture minimizes the time needed for data integration, facilitates maintenance, modification of the data integration process and addition of new data sources.

2.4. Evaluation of solutions included in the architecture

The fourth chapter of the thesis experimentally evaluates DaaS approach and the data integration solutions developed. Local data storing solutions are compared with a remote web service, the developed data integration task parallelization technique is evaluated and the significance of load balancing is estimated.

To compare local data storage with usage of DaaS based data sources, two local data sources and one remote web service are compared. For local data sources the total amount of time needed for data integration (T_K) consists of initial data loading time (T_I) and data retrieval time (T_S). For the remote data retrieval solution T_K=T_S. Experimental results are approximated with regression models.

The calculation of population in the area defined by a central coordinate and a radius is chosen as a data retrieval scenario. Two local data storage solution (A and B) with different performance are compared to an external data retrieval solution (C). The remote data source provides the requested data using WFS (*Web Feature Service*), local data sources store data in Microsoft SQL Server 2008 database. To accelerate the data integration process, local data sources store only the minimum of the required data. T_I and T_S dependence on data loading radius $R_I \in$ {200km, 575km, 950km, 1325km, 1700km} and data retrieval radius $R_S \in$ {10km, 60km, 110km, 155km, 200km} is evaluated.

The relation between T_I and R_I is shown in Figure 2.6. T_I is affected by the amount of loaded data (R_I) and the performance of the used server. Although when loading small amount of data the performance is not critical, data volume increase causes significant T_I differences in local data sources.

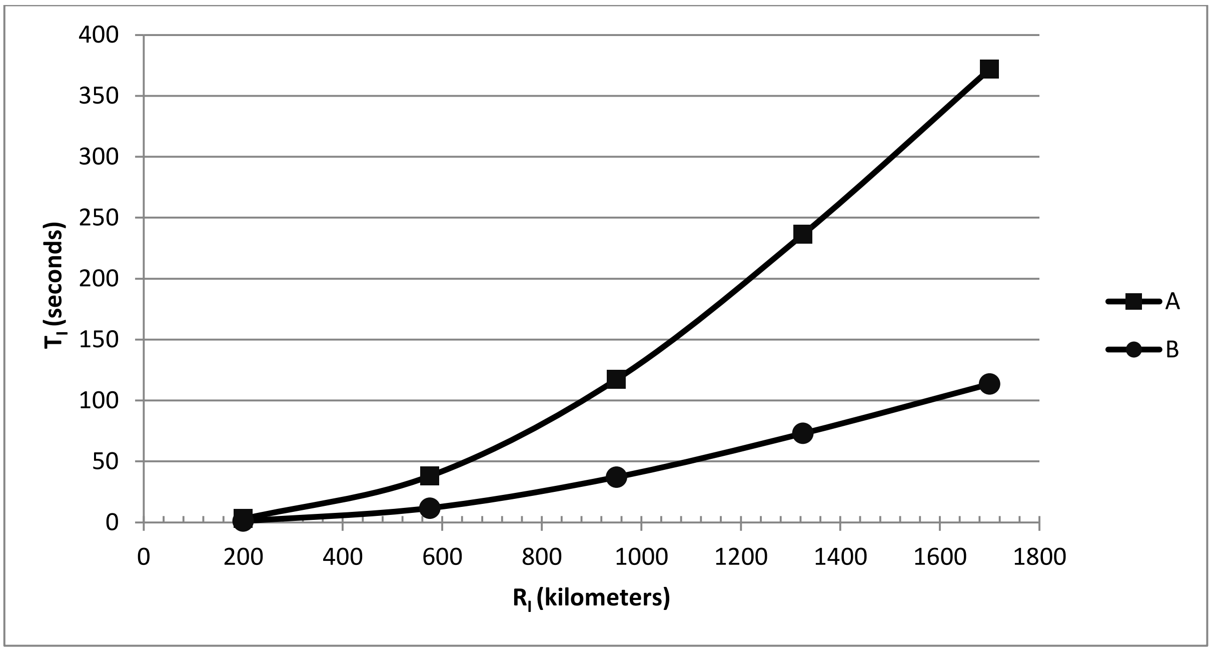


Figure 2.6. T_I dependence on R_I

While developing the regression model approximating effect of R_S and R_I on T_S, the observed experimental error was greater than the effect of R_S on T_S. The only considerable factor is R_I, which reflects the amount of processed data. In the case of a remote solution increase of R_S significantly increases T_K.

In most cases local solutions return data in shorter periods of time, however the need for appropriate infrastructure, load of initial data and regular data updates have to be taken into account. The amount of work caused by those actions can lead to decision in favor of a remote data source even if the potential performance of the local data source would be greater.

Despite the fact that the remote data source wasn't optimized for the concrete data retrieval scenario, it was more effective in cases when the amount of processed data was relatively larger compared to data returned. Comparison of observed T_S values for remote solution C and local solutions A and B (R_I=1700 kilometers) is shown in Figure 2.7.

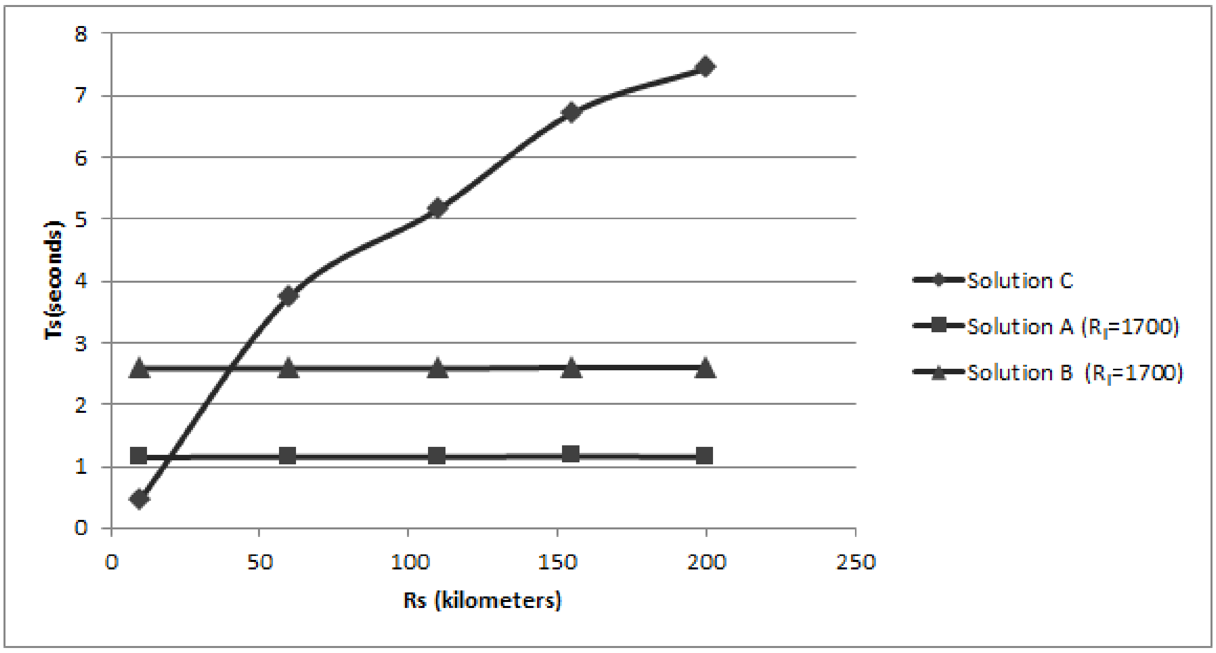


Figure 2.7. T_S for solutions A, B, C

To evaluate the influence of data integration task parallelism on T_K and effectiveness of the defined parallelism algorithm, architecture prototype is compared to commercial ETL system (Microsoft SQL Server 2008 Integration Services) and sequential data retrieval solution. Maximum number of simultaneous requests performed on data source L_V is varied during experiments ($L_V=\infty$ and $L_V=1$) in the prototype and ETL.

Calculation of the mathematical expression (2.1) is chosen as an example, web services performing mathematical operations are used to simulate remote data sources. To determine T_K, calculation of the expression is repeated several times $N_A \in \{1,3,5\}$.

$$Y = \frac{\sqrt{x_1^2+x_2^3+x_3^4-(x_4*x_5*x_6)^2}}{x_7+x_8+x_9} \tag{2.1}$$

The web services are implemented using the REST architectural style. The order of actions needed to solve the expression (2.1) in the case of parallel and sequential data retrieval scenario is shown in Figure 2.8. The effect of N_A on T_K is graphically shown in Figure 2.9.

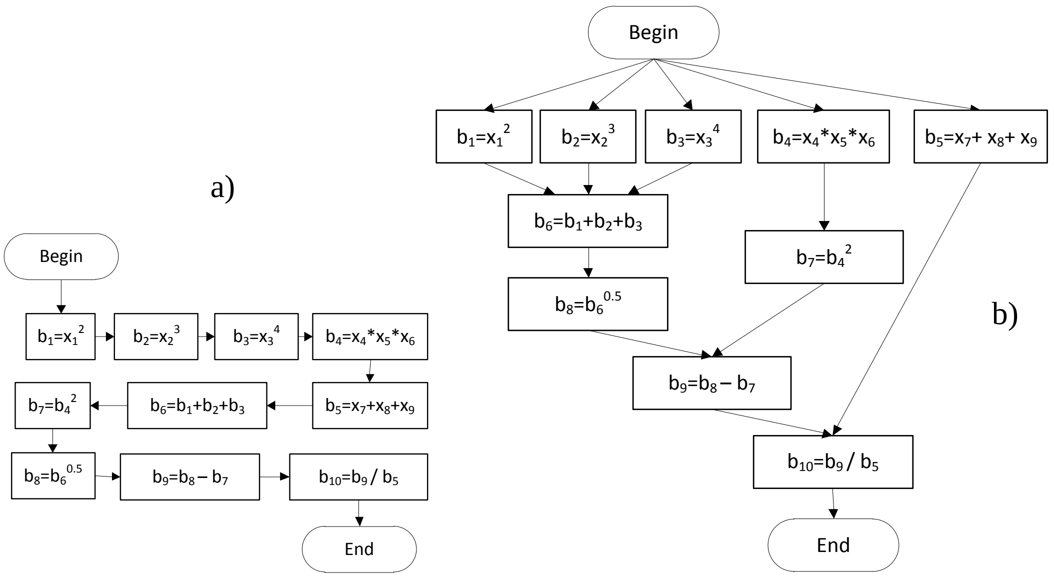


Figure 2.8. Mathematical operation performance order a) sequential b) parallel

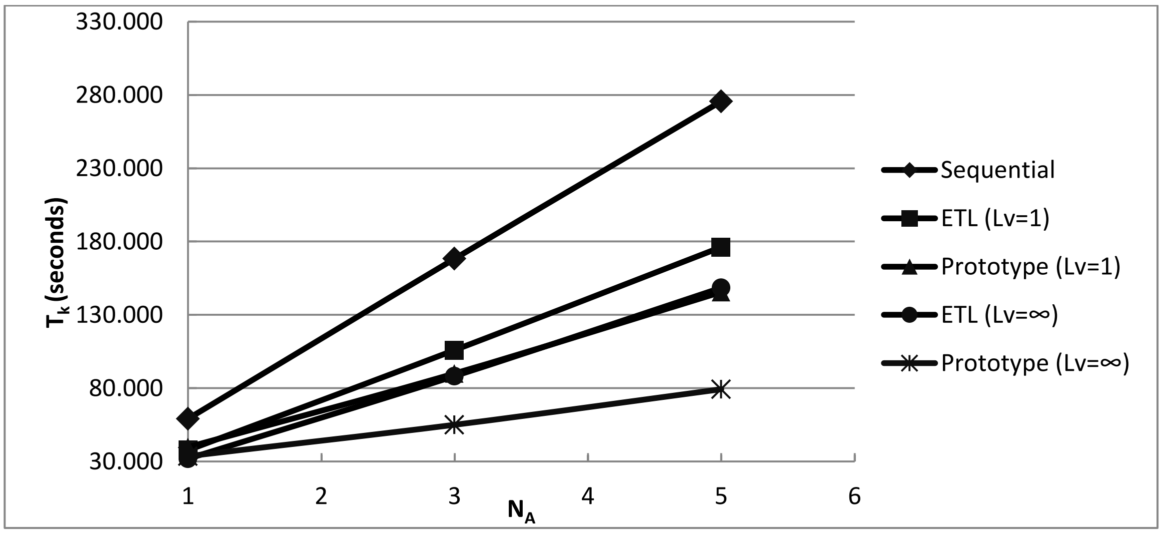


Figure 2.9. Effect of N_A on T_K

Increase of N_A shows advantage of the developed data integration task parallelization algorithm over the one used in ETL. Removing restrictions for simultaneous data integration tasks allows to significantly reduce T_K in case of prototype usage, however using ETL system the benefit is less obvious. The experimental results show that data integration task parallelization has a significant effect on T_K and no usage of parallelism (sequential data retrieval) is the most inefficient.

To evaluate the significance of load balancing, the effect of simultaneous requests for a single web service $N_V \in \{5,6,7,8,9\}$ and the number of alternative web services $N_{WS} \in \{1,2,3,4,5\}$ on T_K is estimated.

The chosen example is to test if a number 1’000’000 is a Fibonacci number. REST web service performing a Fibonacci test is deployed on five identical computers.

Experimental results have shown that while the maximum number of simultaneous requests for a single data source stays constant, there is practically no change in the value of T_K. The maximum number of simultaneous requests for a single web service is calculated as a rounded up value of N_V/N_{WS}. The effect of N_V/N_{WS} on T_K is shown in Figure 2.10.

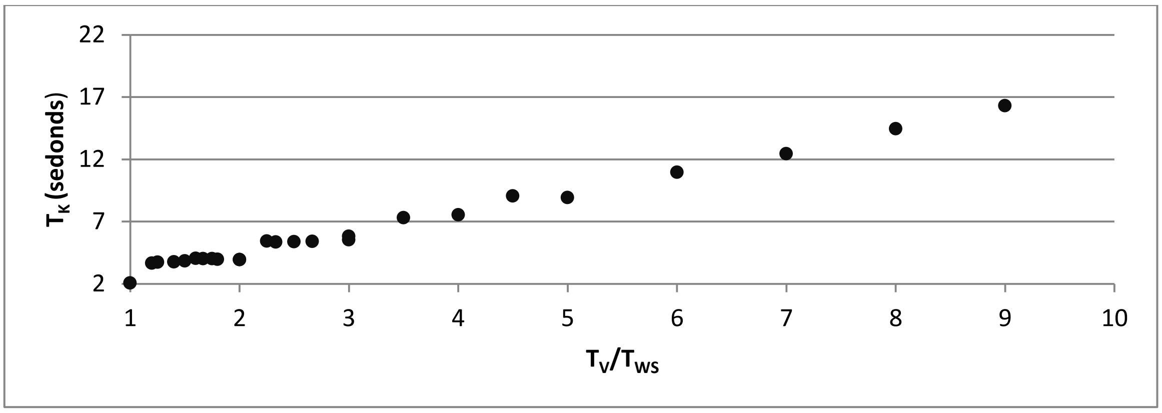


Figure 2.10. Effect of T_V/T_{WS} on T_K

The results confirm that large amount of alternative data sources allows significant reduction of T_K. When using load balancing, T_K is not linearly related to the amount of data to be retrieved.

2.5. Application of the architecture

The fifth chapter of the thesis contains two application examples of the developed architecture. In the first application example, a prototype is used to solve a passenger transportation planning problem at the “BalticTaxi” company. The data integration problem is defined by a cab ordering process, in which an operator receives a client call, chooses the most appropriate cab and informs the client about the time of cab arrival, cost and planned arrival time at a client destination. The initial data consists of the client location and destination, which can be defined as an address, street intersection or point of interest.

The choice of the most suitable cab and calculation of the radial distance are implemented as virtual data sources. To provide routing and geocoding functionality, five remote web services are used – Bing, Yahoo, Google, MapQuest and CloudMade. A local system is used to get information about available cabs. Five abstract data retrieval operations and their corresponding data models are defined. Data source abstraction layers mapping data sources to those abstract data retrieval operations are developed. Input data model of an abstract data retrieval operation routing is shown in Figure 2.11.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="getRoute">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="from">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="latitude" type="xs:decimal" />
              <xs:element name="longitude" type="xs:decimal" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="to">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="latitude" type="xs:decimal" />
              <xs:element name="longitude" type="xs:decimal" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element></xs:schema>
```

Figure 2.11. Input data model for an abstract data retrieval operation routing

When executing an abstract data retrieval operation, a request can be performed to four alternative data sources, previously transforming the input data according to the data model used in the specific data source. The data integration process logic is shown in Figure 2.12. A user interface is developed to generate the initial data XML document, visualize results and monitor the data integration process (Figure 2.13).

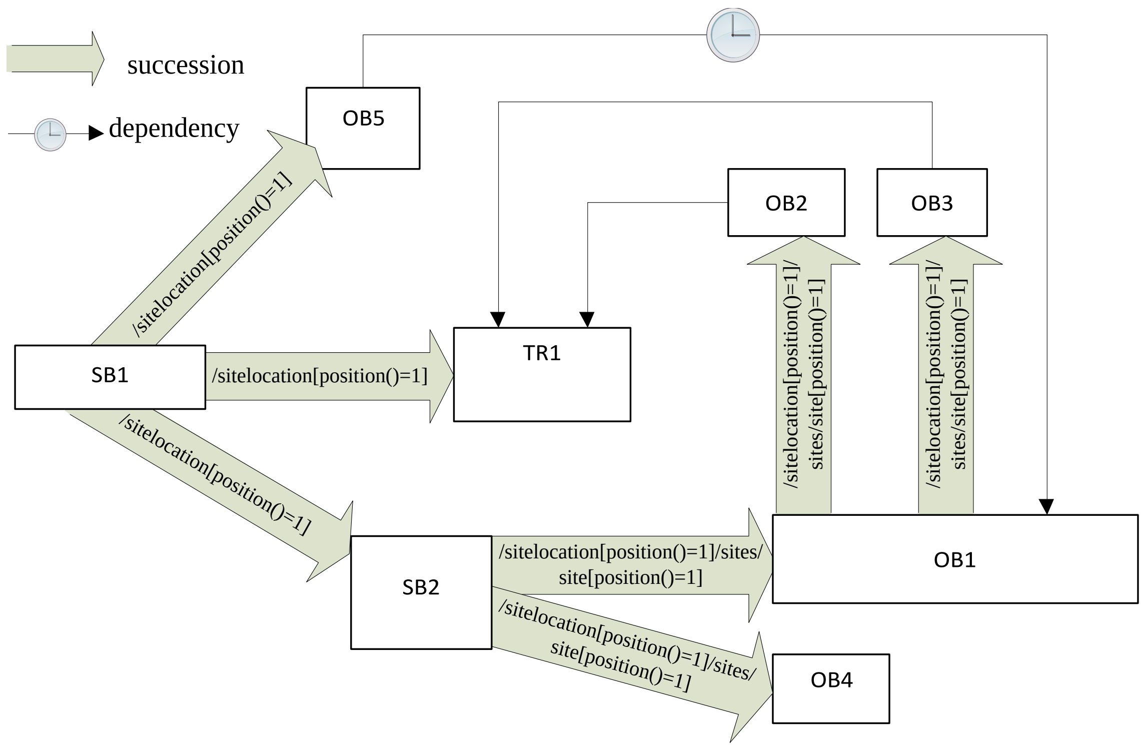
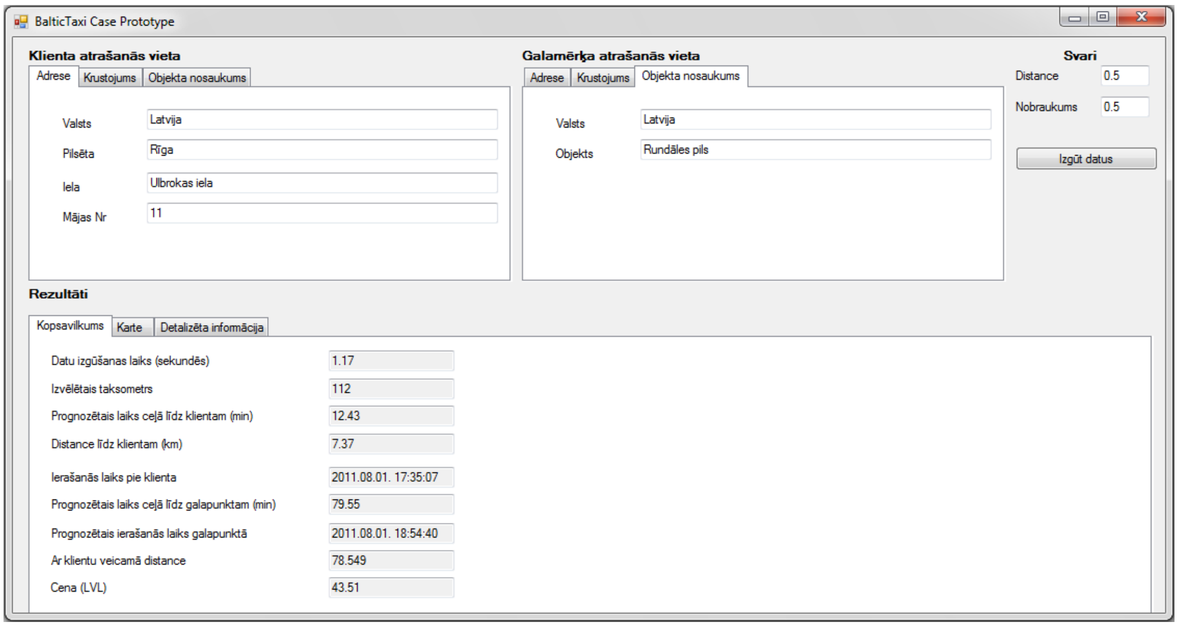

succession
dependency
OB5
OB2
OB3
TR1
SB1
SB2
OB1
OB4
/sitelocation[position()=1]
/sitelocation[position()=1]
/sitelocation[position()=1]
/sitelocation[position()=1]/sites/site[position()=1]
/sitelocation[position()=1]/sites/site[position()=1]
/sitelocation[position()=1]/sites/site[position()=1]
/sitelocation[position()=1]/sites/site[position()=1]

Figure 2.12. „BalticTaxi” data integration process logic

One of the advantages of the developed data integration system is usage of free spatial data processing web services, which allows reducing costs. None of the web services used is able to provide all of the necessary functionality (geocoding of address, street intersection, point of interest and routing), however the defined architecture allows to effectively combine functionality provided by all of the web services in a composite system.



Klienta atrašanās vieta

Valsts	Latvija
Pilsēta	Rīga
Iela	Ulbrokas iela
Mājas Nr	11

Galamērķa atrašanās vieta

Valsts	Latvija
Objekts	Rundāles pils

Svari

Distance	0.5
Nobraukums	0.5

Rezultāti

Datu izgūšanas laiks (sekundēs)	1.17
Izvēlētais taksometrs	112
Prognozētais laiks ceļā līdz klientam (min)	12.43
Distance līdz klientam (km)	7.37
Ierašanās laiks pie klienta	2011.08.01. 17:35:07
Prognozētais laiks ceļā līdz galapunktam (min)	79.55
Prognozētais ierašanās laiks galapunktā	2011.08.01. 18:54:40
Ar klientu veicamā distance	78.549
Cena (LVL)	43.51

Figure 2.13. att. „BalticTaxi" user interface

The advantages for using multiple alternative data sources are:

- reduction of the time needed for data integration;
- lower probability of a data integration process blocking error;
- one web service supplements the data sets available in other web services.

The benefits were also assessed by representatives of the "BalticTaxi" and positive feedback was received. At the moment a new information system is developed for the company and further adaption of the developed data integration system is considered.

The second application example is based on object location problems reviewed in literature [19]. The decision-making problem in this example is finding optimal locations for fast food restaurants with regards to the number of potential customers, competitors and real estate cost. The initial data consists of potential site addresses, radius for customer and competitor data retrieval, the number of sites to be opened and minimal distance between two open fast-food facilities. In order to solve the described facility location problem, a multi-objective mathematical programming model [13] is constructed and commercially available optimization software is used. The necessary data consists of the population, number of competitors, real estate cost and potential site radial distance matrix.

Geocoding is performed using Google, Bing, Yahoo and CloudMade web services. Population data is retrieved from Sedac WFS web service, Zillow provides real estate data, and MapQuest returns competitor data. A virtual data source is developed to calculate the potential site radial distance matrix. Five abstract data retrieval operations are defined. The input data model of the geocoding abstract data retrieval operation is shown in Figure 2.14.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="geocode">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="location">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="street" />
              <xs:element name="house" />
              <xs:element name="city" />
              <xs:element name="zipcode" type="xs:unsignedShort" minOccurs="0"/>
              <xs:element name="country" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 2.14. Abstract data retrieval operation geocoding input data model

While performing geocoding input data is transformed and a request is made to one of the four alternative data sources. The defined data integration process logic is represented in Figure 2.15. A user interface was developed to allow data integration process monitoring (Figure 2.16).

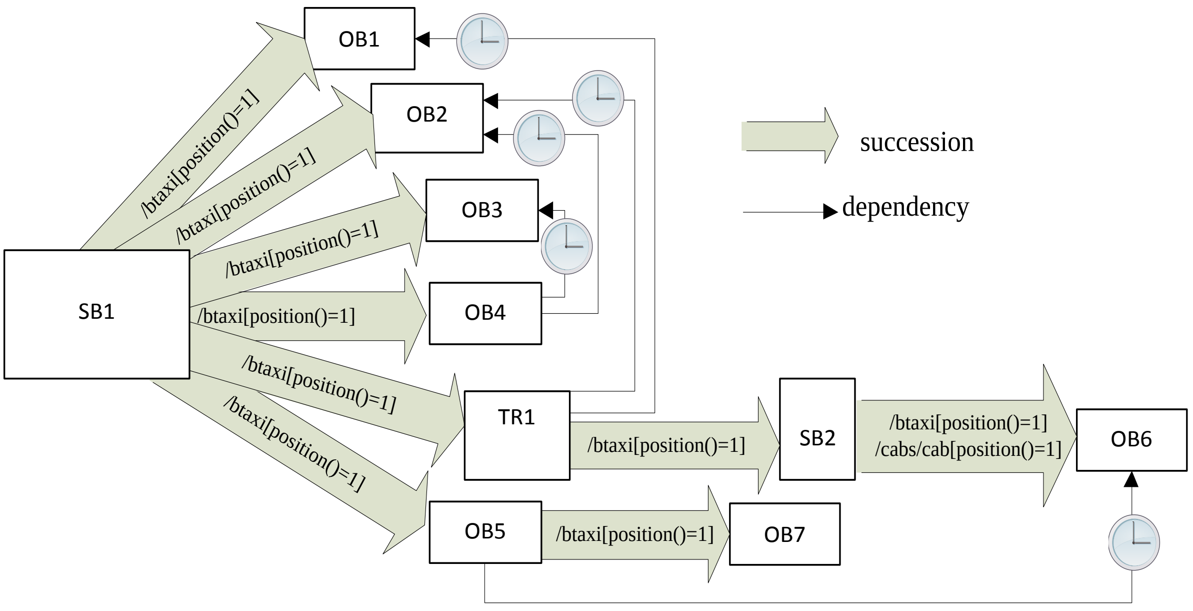
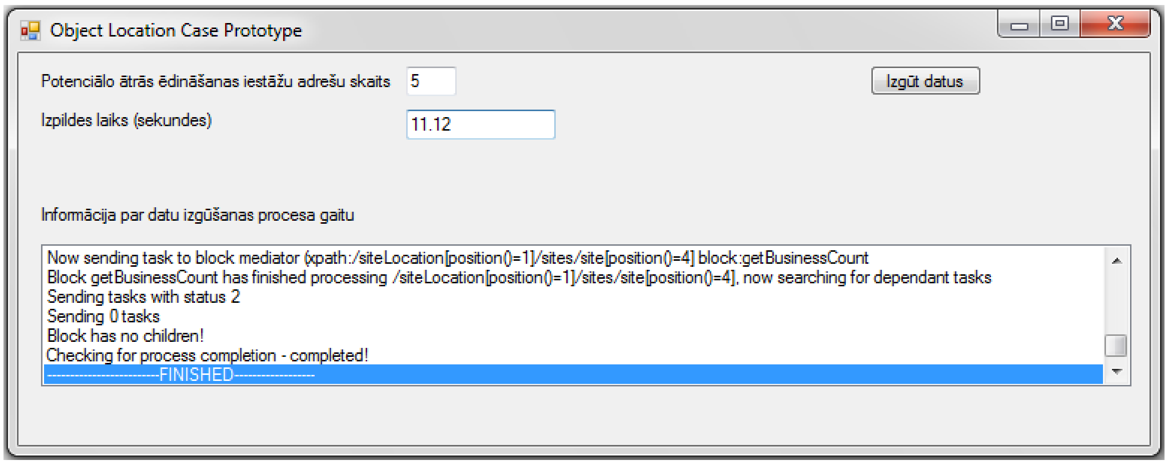


Figure 2.15. Facility location data integration process logic

During experiments the effect of the number of potential sites N∈{50,100,200} on the data integration time and modeling time is estimated (Figure 2.17).



Potenciālo ātrās ēdināšanas iestāžu adrešu skaits 5

Izpildes laiks (sekundes) 11.12

Informācija par datu izgūšanas procesa gaitu

```
Now sending task to block mediator (xpath:/siteLocation[position()=1]/sites/site[position()=4] block:getBusinessCount
Block getBusinessCount has finished processing /siteLocation[position()=1]/sites/site[position()=4], now searching for dependant tasks
Sending tasks with status 2
Sending 0 tasks
Block has no children!
Checking for process completion - completed!
----------------------FINISHED----------------
```

Figure 2.16. Facility location user interface

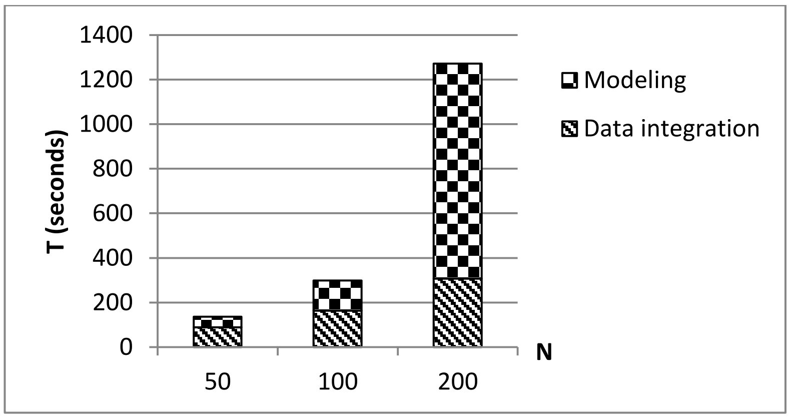


Figure 2.17. Data integration time compared to modeling time

The proportion of data integration time decreases with larger N values, because object location problem is a NP-hard problem while data integration time increases linearly. Only the time needed for calculation of the distance matrix increases nonlinearly, however it constitutes only a small part of the total time needed for data integration. During the data integration process individual web service response times were saved in QR, allowing to calculate the possible data integration time in case of a sequential data integration task execution. T_K variation in case of sequential and parallel data integration is visualized in Figure 2.18.

The results have shown, that parallelization of data integration operations and load balancing has allowed to dramatically reduce the time needed for data integration. To reduce T_K even further, more alternative web services providing competitor and population data must be added as those operations were the most time-consuming.

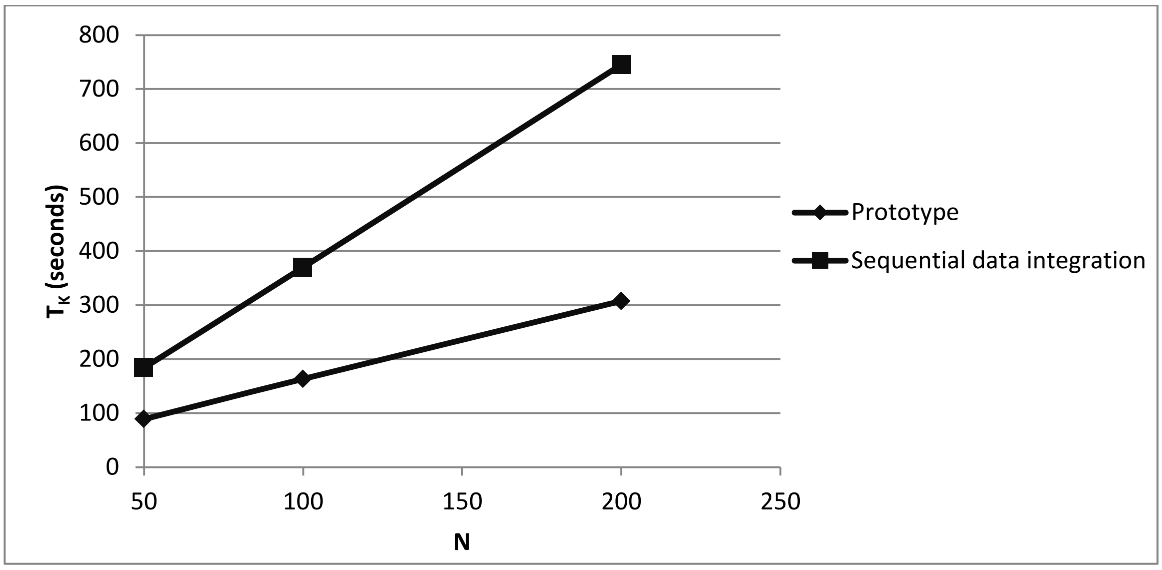


Figure 2.18. T_K for sequential and parallel data integration in case of site location

3. CONCLUSION

The main theoretical result of the dissertation is development of the on demand data integration architecture including the overall architecture, elaboration of its components and development of data integration algorithms.

The prototype of the architecture has been used to solve a theoretical facility location problem and a practical passenger transportation planning problem at “BalticTaxi”. Representatives of the “BalticTaxi” have given positive feedback, because the data retrieval time fluctuates within a few seconds, hardware requirements are very low and there is no need to buy or rent expensive geographic information systems.

The developed architecture solves the business intelligence data integration problem by using the DaaS approach. The elaborated generic model of the on demand data integration system can be implemented in different ways. The developed architecture and its implementation prototype is only one of them. Load balancing and parallelization of data integration tasks has allowed for considerable reduction of data integration time. Data integration solutions that are based on the developed architecture are easy to maintain and modify because the data source access logic is completely separated from the data integration process logic.

The theoretical results of this work are:

1. Identification of the main heterogeneous remote source integration problems, which are data source discovery, incomplete metadata, a wide range of used data formats and protocols, semi-structured data format, safety, necessity to consider nonfunctional requirements, the need for load balancing and ambiguous licensing policy. The related works are analyzed and it is concluded that there is not enough focus on correct and timely execution of individual data integration tasks, load balancing, minimization of data integration time, consideration of nonfunctional requirements and promotion of reusability.
2. Definition of remote data source integration system model which is based on abstraction. Abstract data operations are defined and web service methods are mapped to them using abstraction layers. The data integration process contains references to abstract data retrieval operations.
3. Definition of the main architecture components, their operation principles and interactions. The architecture consists of the Service Register (data source abstraction layers, abstract data retrieval operations and their corresponding data models), Quality Data Repository (web service related quality measures) and functional data integration adapter (FDIA). FDIA is divided into the Data Integration Process component (implements data integration process logic) and Operation Execution component (searches for web service methods that are mapped to abstract data retrieval operation, monitors quality data, performs load balancing and error recovery).
4. Development of an algorithm for parallelization and interdependency detection of individual data integration tasks, that is based on predefined data integration task states and XPath addressing.
5. Definition of possible block types (simple, operation and transformation) and block relations (dependency and succession).
6. Definition of data integration task states – idle, created, sent, processing, finished, error.
7. Development of an adaptive nonfunctional and functional requirements based web service selection and load balancing algorithm. At the first step of the algorithm functionally suitable web services are found. Web services whose individual quality measures are lower than allowable are not further examined. For the remaining web services the overall fitness indicator is calculated and a

specific number of the best web services are chosen. Data is retrieved from the web service with the smallest predicted response time.

The practical results of the work are:

1. A prototype of the architecture is developed using the .NET framework. Data integration task parallelism is achieved by multithreading.
2. Data retrieval from a DaaS based data source is practically compared with data retrieval from local source, disadvantages and advantages of both approaches are summarized. To use local data sources, the necessary infrastructure has to be created and initial data loading and renewal performed, however in most cases local data sources provide smaller response time. DaaS based data sources can’t be optimized for specific data integration scenarios and the retrieved data is transferred over the Internet. The main advantage is that there is no need for creating infrastructure, initial data loading and data renewal. It has been identified that data retrieval from DaaS based data sources is faster in cases when the amount of data to be searched is relatively larger than the amount of data transferred over internet.
3. The need for data integration task parallelization and the efficiency of the corresponding algorithm is confirmed.
4. Several publicly available web services have been tested and their corresponding quality of service data has been gathered. Considerable fluctuations of the response time have been observed, and the need for error recovery has been confirmed.
5. The developed architecture is used in two application examples – a theoretical facility location problem and passenger transportation planning problem at “BalticTaxi”. Positive feedback is received from the representatives of the company. Further adaptation of the developed data integration system and its inclusion in the enterprise information system is considered.

The directions for future research are:

1. Addition of new blocks and block configuration parameters.
2. Development of an algorithm allowing data integration process resumption in case of a manual suspension or error.

3. Visualization of data integration process for monitoring purposes and development of a user interface allowing definition of a data integration process in a more convenient way.
4. Implementation of more complex load balancing algorithms.
5. Assessment of architecture component decentralization possibilities.
6. Development of new application examples.

BIBLIOGRAPHY

1. Arlow J., Neustadt I. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design - USA: Addison-Wesley Professional, 2005. - p.624.
2. Batini C., Cappiello C., Francalanci C., Maurino A. Methodologies for data quality assessment and improvement// ACM Computing Surveys. - 2009. - Vol.41 - No.3 - p.1-52.
3. Bhide M., Agarwal M. K., Bar-Or A., Padmanabhan S., Mittapalli S. K., Venkatachaliah G. XPEDIA: XML processing for data integration// Proceedings of the VLDB Endowment. - 2009. - Vol.2 - No.2 - p.1330-1341.
4. Casters M., Bouman R., Dongen J. Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration - Canada: Wiley, 2010. - p.720.
5. Dan A., Johnson R., Arsanjani A. Information as a Service: Modeling and Realization// International Workshop on Systems Development in SOA Environments (SDSOA). - Minneapolis, Minnesota, USA: IEEE, 2007. - p.2-2.
6. Donglai Z., Coddington P., Wendelborn A. Binary Data Transfer Performance over High-Latency Networks Using Web Service Attachments// IEEE International Conference on e-Science and Grid Computing. - Bangalore, India: IEEE, 2007. - p.261-269.
7. Eriksson L., Johansson E., Kettaneh-Wold N., Wikström C., Wold S. Design of Experiments, Principles and Applications - Sweden: Umetrics Academy, 2008. - p.329.
8. Frada Burstein C. W. H. Handbook on Decision Support Systems 1: Basic Themes - Germany: Springer, 2008. - p.854.

9. Giordano A. D. Data Integration Blueprint and Modeling: Techniques for a Scalable and Sustainable Architecture - USA: IBM Press, 2011. - p.416.

10. Grosu D., Chronopoulos A. T. A Truthful Mechanism for Fair Load Balancing in Distributed Systems// Network Computing and Applications (NCA). - Cambridge, MA, USA: IEEE, 2003. - p.289-296.

11. Guohua Y., Jingting W. The Design and Implementation of XML Semi-structured Data Extraction and Loading into the Data Warehouse// International Forum on Information Technology and Applications (IFITA). - Guangzhou, China: IEEE, 2010 -p.30-33.

12. Hillier F. S. Introduction to Operations Research - USA: McGraw Hill Higher Education, 2000. - p.1220.

13. Kampars J., Grabis J. Spatial Data Integration Approach with Application in Facility Location// 16th International Conference on Information and Software Technologies. - Kaunas, Lithuania: Kaunas University of Technology, 2010. - p.117-125.

14. Kopecký J., Gomadam K., Vitvar T. hRESTS: An HTML Microformat for Describing RESTful Web Services// IEEE/WIC/ACM International Conference. - Sydney, Australia: IEEE, 2008. - p.619-625.

15. Lathem J., Gomadam K., Sheth A. P. SA-REST and (S)mashups : Adding Semantics to RESTful Services// International Conference on Semantic Computing (ICSC). - Irvine, CA, USA: IEEE, 2007. - p.469-476.

16. Lu W., Chiu K., Gannon D. Building a Generic SOAP Framework over Binary XML// 15th IEEE International Symposium on High Performance Distributed Computing. - Paris, France: IEEE, 2006. - p.195-204.

17. Maximilien E. M., Singh M. P. A framework and ontology for dynamic web, services selection// IEEE Internet Computing. - 2004. - Vol.8 - No.5 - p.84-93.

18. Mou Y.-j., Cao J., Zhang S.-s., Zhang J.-h. Interactive Web service choice-making based on extended QoS model// The Fifth International Conference on Computer and Information Technology (CIT). - Shanghai, China: IEEE, 2005. - p.1130-1134.

19. Owen S. H., Daskin M. S. Strategic facility location: A review// European Journal of Operational Research. - 1998. - Vol.111 - No.3 - p.423-447.

20. Paršutins S., Borisovs A. Data Mining Driven Decision Support// Polish Journal of Environmental Studies. - 2009. - Vol.18 - No.4 - p.8-11.

21. Pautasso C., Zimmermann O., Leymann F. RESTful web services vs. "Big" web services: Making the right architectural decision// 17th International Conference on World Wide Web 2008. - Beijing, China: ACM, 2008. - p.805-814.

22. Ran S. A model for web services discovery with QoS// ACM SIGecom Exchanges. - 2003. - Vol.4 - No.1 - p.1-10.

23. Truong H. L., Dustdar S. On analyzing and specifying concerns for data as a service// 2009 IEEE Asia-Pacific Services Computing Conference (APSCC). - Biopolis, Singapore: IEEE, 2009. - p.87-94.

24. Truong H. L., Dustdar S. On evaluating and publishing data concerns for data as a service// IEEE Asia-Pacific Services Computing Conference. - Hangzhou, China: IEEE, 2010. - p.363-370.

25. Turban E., Sharda R., Delen D. Decision Support and Business Intelligence Systems (9th Edition) - USA: Prentice Hall, 2010. - p.780.

26. Vercellis C. Business Intelligence: Data Mining and Optimization for Decision Making - United Kingdom: Wiley, 2009. - p.436.

27. Wang J., Yu A., Zhang X., Qu L. A dynamic data integration model based on SOA// Second ISECS International Colloquium on Computing, Communication, Control, and Management (CCCM). - Sanya, China: IEEE, 2009 -p.196-199

28. Werner C., Buschmann C. Compressing SOAP messages by using differential encoding// IEEE International Conference on Web Services. - San Diego, CA, USA: IEEE, 2004. - p.540-547.