Master Thesis

Implementing data analysis and visualization into a SOA based architecture for AddTrack Enterprise 4G

Student:
Björn Laurell

Supervisor:
Aida Causevic

Industrial Supervisor:
Amel Muftic

Examiner:
Cristina Seceleanu

Addiva Consulting AB
Bombardier Transportation

School of Innovation, Design and Engineering
Mälardalen University

January 11, 2012

Abstract

Service-Oriented Architecture (SOA) is an architectural approach designed to provide scalable, resuable, and extensible software by dividing the software into self-contained services. Online Analytical Processing (OLAP) is an approach for data management that supports resolving ad hoc and complex multi-dimensional analytical queries. In this thesis, a case study for adapting these two technologies to an existing off-board train diagnostics system, called AddTrack Enterprise, is presented. The thesis also present a proposal for how to visualize the data contained in this system, such that it accommodates the OLAP approach. The thesis outlines the study of the subject matter and of the implementation of software for AddTrack based on these approaches.

Sammanfattning

Tjänsteorienterad Arkitektur (SOA) är ett arkitekturellt tillvägagångssätt tänkt att tillhandahålla mjukvara med bra skalbarhet, återanvändbarhet och utökningsförmåga genom att dela upp mjukvaran i självständiga tjänster. Online Analytical Processing (OLAP) är ett tillvägagångsätt för data hantering som möjliggör att snabbt besvara ad hoc och multi-dimensionella analytiska frågor. I den här avhandlingen presenteras en studie i hur dessa teknologier kan anpassas för användande i ett off-board tåg diagnostiksystem, kallat AddTrack Enterprise. Avhandlingen presenterar också ett förslag på hur data från det här systemet kan visualiseras på ett sätt så att OLAP tillvägagångsättet befrämjas. Avhandlingen täcker en studie i ämnet och implementeringen av mjukvara för AddTrack baserat på dessa tillvägagångsätt.

Contents

List of Figures

List of Tables

0.1 Terminology and abbreviations

3G
Third generation

4G
Fourth Generation

AMO
Analysis Management Objects, API for managing SSAS instances.

API
Application Programmer's Interface

BIDS
Business Intelligence Developer Studio, a Microsoft Visual Studio add-on which adds Business Intelligence related functionality and enables design of OLAP cubes.

CLR
Common Language Runtime, the .NET platform's virtual machine component

FTP
File Transfer Protocol

GSM
Global System for Mobile Communications, a standard for digital cellphone communication

HDDV
Hierarchical Dynamic Dimensional Visualization, a visualization technique for navigating OLAP data

HTTP
Hyper-Text Transfer Protocol

HTTPS
Secure Socket Layer over Hyper-Text Transfer Protocol

OLAP
Online Analytical Processing

POCO
Plain Old CLR Object

SOA
Service Oriented Architecture

SOAP
Simple Object Access Protocol

SSAS
SQL Server Analysis Services

SSIS
SQL Server Integration Services

TCP/IP
Transmission Control Protocol over Internet Protocol, an end-to-end protocol for reliable data transmission

TDS
Train Diagnostics System

WCF
Windows Communication Foundation, A framework for computer communication

WPF
Windows Presentation Foundation, a framework for constructing graphical user interfaces

X.509
A standard for Public Key Infrastructures, including certificates

XML
eXtensible Markup Language

XMLA
XML for Analysis, a SOAP-based language for accessing and controlling analytical systems

Chapter 1

Introduction

1.1 Background

1.1.1 Addiva

Addiva Consulting AB (in the rest of the paper referred to as Addiva) is a Swedish technology consultant company with about 70 employees. They offer services within system development, automation, production engineering, diagnostics, telematics, energy, and production of automation equipment. In addition Addiva also provides services of business consulting. Addiva has offices and facilities in Västerås, Stockholm, and Ludvika.

1.1.2 Bombardier Inc.

Bombardier Inc. is a world leading manufacturer of aerospace and railway transportation that has 65400 employees divided over 69 production and engineering sites in 23 different countries all over the world. Bombardier Inc. manufacture everything from trains and railway bogies to amphibious aircrafts and Lear jets.

1.1.3 AddTrack Enterprise

AddTrack Enterprise is an off-board diagnostics tool for trains originally developed and owned by Bombardier Transportation and currently maintained and developed by Addiva. AddTrack uses and processes data originating from various on-board equipments on the trains, like computers, sensors and data recorders. This data can then be retrieved and transformed into useful information which in turn can be used to monitor and troubleshoot issues with the trains.

Generated data from the trains is sent as data files (in one of 3 possible formats: OTI, TDF and XML, see Section 5.1), wirelessly by GSM to a station on land. Then the files are uploaded, through FTP, to a designated location (the Landing Zone) on the AddTrack Server . The data in these files are extracted, transformed, and loaded into databases. End-users access the data through a WPF based client application that retrieves the data from the AddTrack server.

1.2 Thesis purpose and goal

The purpose of this thesis is to together with Addiva develop an implementation proposal for some key services in the new AddTrack architecture, investigate the possibilities of integrating Analysis Services into AddTrack to improve data management and information processing, and finally to implement a reference version of this service.

The system should fulfill the following requirements.

- Performance - The system should at least have the same, preferably better, end-user performance.
- Modularity - The system components should be loosely coupled and capable of supporting easily adding new functionality and customizing existing functionality for the specific needs of individual customers, without having any major impacts on adjacent subsystems.
- Scalability - It should be easy to scale up the amount of data the system manages without having a negative impact the system's performance. The system should also provide means to easily add new customers/projects to AddTrack without affecting existing customers.
- Security - A service-oriented architecture poses different security challenges compared to a more traditional client-server model, which makes the existing security model inapplicable/insufficient. The new version of the AddTrack system should offer protection against unauthorized access and data confidentiality to at least the same degree as the old system.

1.3 Limitations

The following requirements were put on the implementation by Addiva:

- Source code must be written in C# using .NET Framework version 4.0.
- Inter-service communication must be done using WCF.
- The interaction with the databases must be done using appropriate Microsoft SQL Server Analysis Services APIs, alternatively where that is not applicable with ADO.NET Entity Framework.

These limitations were imposed to allow for a produced solution to leverage their existing software infrastructure.

1.4 Thesis outline

Chapter 2 provides a theoretical background to the topics that have been covered and a short discussion on some related work. SOA, OLAP and visualization techniques are covered in Sections 2.1, 2.2, and 2.3, respectively. In Chapter 3 the problems this thesis is addressing are presented. In Chapter 4 we analyze more in-depth the challenges presented in our problem formulation. Chapter 5 presents our models for the new AddTrack 4 system. The implementation of models, introduced in Chapter 5, is described in Chapter 6, where we present how the models described in Chapter 5 have been implemented. Chapter 7 covers some of the tests and verifications that have been performed. We present our results in Chapter 8. Unresolved issues, possible future work are presented in Chapter 9. Finally we conclude the report in Chapter 10.

1.5 Conventions

In the rest of this thesis, when items are listed, a bulleted list is used whenever the ordering of the listed items is arbitrary. In the cases where a specific ordering must be followed the items are listed in an enumerated list instead. The usage of the terms, AddTrack 3 and AddTrack 4, refers to the existing AddTrack product (AddTrack Enterprise 3G) as a whole and the planned new product (AddTrack Enterprise 4G) as a whole, respectively. Whenever we refer to the components of the AddTrack system we will use either their full name or the official abbreviation of their full name.

Chapter 2

Related work and theoretical background

2.1 Service Oriented Architecture

Service-Oriented Architecture (SOA) is an architectural approach for software design that uses loosely coupled, self-contained software entities called services. SOA provides software that is scalable and flexible, suitable for environments with rapidly changing requirements [MPP03].

A single service consists of the following components: a service contract, a service provider, and a service consumer. The service contract consists of one or more interfaces which define how and with which data a service can be invoked. The service provider is a concrete software implementation of a service contract. A service consumer is any software that uses a given service. A service consumer is in some terminologies also called a requestor. The service contract forms a binding contract between provider and consumer to which both must adhere.

Ang et al. define service-oriented architecture as: "an approach for building distributed systems that deliver functionality to either end-user applications or other services"[EAA+04]. The SOA approach for software development has its roots in the Component-Based Systems (CBS) paradigm , which in turn has evolved from the Object-Oriented (OO) paradigm. SOA services are coarse grained software entities which are self-contained in terms of the tasks they do. Each service is intended to provide one specific task. The coarseness of the task depends on the specifics of the business logic, but should generally correspond to a single entity in the business domain.

2.2 Online Analytical Processing

Online Analytical Processing (OLAP) has played an important role as a support tool for enabling enterprises to have effective decision making for years. Many enterprises gather data from their business-, operations-, and industrial-processes into large data warehouses.

The operations used to manipulate a data cube differ from the operations used to manipulate relational and other types of databases. Operations include drill-down, roll-up, pivoting, slicing, dicing, and filtering [SAM98]. Drill-down and roll-up operations changes the perspective of the cube to be less or more general respectively. For example, if one would have a view of the cube that shows sales quantities by city and by month a drill-down operation might change the view to show sales quantities by individual store per day, and a roll-up operation might change it to show sales quantities per state and quarter. Slicing operations extracts data along a particular dimension of the cube (a slice). Dicing operations are the process of extracting a sub-cube by using multiple, intersecting slices. Filtering operations performs selection on the cube using one or more constants or ranges of values. Pivoting performs rotation of the various axes of the cube so that the cube can be examined from different angles.

The system stores the metadata required for OLAP capabilities in a multidimensional format, which can either be based on a relational scheme or in a (often) proprietary multidimensional array format optimized for OLAP. Exactly which system is used depends on the implementation of the OLAP engine used. There exist several commercial vendors which offer OLAP solutions.

The data is presented in a form called a cube, which consists of numeric facts called measures (the basis for the data presented) and dimensions (used to categorize the measures into chunks from which interesting information can be extracted). The dimension data can be organized in a star schema, with each single dimension table directly joined to a fact table or in a snowflake schema with several dimensions having indirect relationship to fact tables over one or more intermediate dimensions [CD97]. Conceptually, there is no limit to the number of dimensions and measures a cube can have or present, although that does not rule out limitations in specific OLAP implementations. A measure can, for example, be the number of sold products, gross income, and total revenues (for a chain of retail stores). The specifics depend on the domain of the data the company keeps.

A dimension can, for example, be geographical information, domain specific data like individual stores, employees, a product catalog, or time data. The main purpose of the dimension data is to categorize and classify measure data. OLAP systems are often classified by how it stores the data and metadata. The three common approaches are Multidimensional OLAP (MOLAP), Relational OLAP (ROLAP), and Hybrid OLAP (HOLAP). Other approaches based on these also exist.

In [Gor03] and [MK98] have been shown results of studies conducted to determine which method is suitable for various situations, pointing out advantages

and drawbacks for each of them. Some of these are restated in the following text.

MOLAP stores metadata, aggregated data, and fact data in specialized multidimensional structures. Aggregated data is pre-computed in a processing step when data is loaded into the cube. This has the advantage of fast query performance (if done properly, no approach can turn a poor design into a good design) and potentially savings in disk space usage, since the specialized formats allows for various compression techniques to be used. Drawbacks include data latency due to having to preprocess the data to incorporate pre-calculated aggregations, which can take significant time for large cubes. MOLAP databases can also suffer from data explosions for large dimensions in certain circumstances.

ROLAP only stores the schema metadata and satisfies queries directly from the underlying relational data warehouse. This has the advantage of the data always reflecting the most recent state and this method is very scalable for dimensions with many data members. Disadvantages include poor query performance, because of the fact that the data has to be retrieved from the underlying database and the result has to be computed from the data every time. As a consequence of this the OLAP database has to maintain a constant connection to the relational data source at all times to function properly. A MOLAP cube however would only need to be connected to the underlying database during reprocessing.

HOLAP is an approach that tries to combine the best of both worlds. Depending on the specific implementation, HOLAP based solutions allow the user to decide which data should be stored using ROLAP and which should be stored using MOLAP. In addition to this some vendors offer their own proprietary solutions for bridging the gap between ROLAP and MOLAP. One example is Microsoft's Proactive Caching technology which uses a notification based approach to initiate reprocessing whenever changes to the underlying data is detected.

2.3 Visualization techniques

Although being able to extract useful information from data sources containing large amounts of data, which is in itself very powerful, the usefulness of OLAP can be limited unless powerful visualization techniques are used to present the data in ways which clearly make the interesting patterns and information visible. Due to the multidimensional nature of OLAP traditional tabular visualization techniques are usually insufficient to view the data extracted from a cube.

Marghescu compares various multidimensional visualization techniques in the context of financial benchmarking [Mar08]. The studied techniques includes: scatter plot matrices, permutation matrices, survey plots, multiple line graphs, tree maps, parallel coordinates, Sammon's mapping, self-organizing maps, and Principal Component Analysis.

Scatter Plot Matrices, Parallel Coordinates, Principal Components Analysis, Sammon's Mapping, and the Self-Organizing Map belong to a group of visu-

alization techniques called geometrically transformed displays. Common to all of these techniques is that they use some form of geometric transformation to project a multidimensional data set onto a two-dimensional display surface. The exact transformations performed vary between each method. In a Scatter Plot Matrix all possible pairings of dimensions are displayed in its own Scatter Plot which is a two-dimensional display of the values (distributed along the dimension pairing). Plots for each possible pairing are then grouped into a matrix which forms the whole display.

In Parallel Coordinates each dimension is represented by a parallel axis. Each data item is represented by a polyline that intersects each axis at a point proportional to the value of that data item along the dimension that the axis represents. This method is suitable for detecting outliers.

In Principal Component Analysis the dimensions are correlated with a principal axis such that each axis is correlated to that or those dimensions that displays the highest variance of the data set, given the constraints imposed by already existing Principal Component axes. This is then visualized as a scatter plot or similar, using the principal axes instead of the dimensions directly. This method can discover outliers, clusters and relationships between data.

Sammon's Mapping uses a non-linear projection of higher dimensional data to a lower dimensionality while trying to preserve the higher dimensions structure of inter-point distances. The result of which is then plotted as a scatter like graph. Due to the non-linearity of the transformation comparing data points on the graph with each other provides no meaningful information in regards how items relates to each other. This technique suits itself for detecting classes of similar items in the data.

Self-Organizing Maps are a technique that uses neural networks to create a mapping from a higher dimensional input space to a lower dimensional representation of that space, preserving the topology of the input. There are many ways to visually represent the output of a self-organizing map including Scatter Plots, U-Matrices, and Feature Planes.

Multiple Line Graphs, Permutation Matrices, and Survey Plots are variations of common two-dimensional visual displays. A single line graph can display a measure along one dimension. By using multiple line graphs several dimensions can be displayed by letting each individual line graph to be the measure at an intersection between several dimensions. By generating one such graph for each dimension additional dimensions can be displayed in a single plot. However due to the way the number of individual lines grows with the number of dimensions and the number of members along each dimension, displaying very large dimensions and/or many dimensions at the same time is often infeasible.

Permutation Matrices are similar to Multiple Line graphs in that it tries to display multidimensional data by showing a two-dimensional display for each measure along a particular dimension and by adding more graphs for each additional dimension. In this method individual measures are visualized as vertical bars in a bar graph along a horizontal line. Each horizontal line represents a collection along a dimension. Survey plots are a variation of Permutation Matri-

ces in which the bars are horizontal and centered with no space between them. This method clearly shows outliers, values that are a lot smaller or a lot larger then surrounding values.

The Tree Map technique belongs to a category of techniques called stacked displays. The Tree Map in a hierarchical approach of visualizing multidimensional data by using hierarchies of nested rectangles to represent items in the data. Different dimensions can be mapped to different properties of the rectangles such as size, color, position, and associated label of the rectangle. This provides a view of the data that is compact. This method can distinguish classes, describe the properties of classes and discover outliers in the data.

Maniatis proposes the Cube Presentation Model in his paper and show how it can be mapped onto the Table Lens visualization technique [MVSV03]. The technique works by using a Degree of Interest function on a set of data within a multidimensional cube and mapping that data onto a Table Lens, which is a two-dimensional table subset which is used analogous to a magnifying glass to look at a data point at a higher granularity level. Before that the multidimensional nature of the data is flattened by cross joining data along several dimensions to produce a two-dimensional tabular representation of the data. This is then visualized as a compacted two-dimensional table at low granularity with the user interactively choosing areas of interest which is "zoomed" to a higher granularity level with more detail. This method is suitable for exploring large data sets at a detailed level without losing the larger picture the data presents.

Datta and Techapichetvanich presents a visual model, Hierarchical Dynamic Dimensional Visualization (HDDV), which specifically addresses the issue of how to enable navigation through the hierarchical structure that OLAP data often is organized, through the use of a graphical interface [TD05] . In this method each hierarchy in a dimension is represented by a horizontal bars (bar sticks in their terminology), which is subdivided into smaller rectangles where each rectangle represents a member at that level in the dimension.

Chapter 3

Problem formulation

3.1 The current AddTrack

The current version of AddTrack Enterprise (3G) is based on a simple client-server architecture consisting of a client application, AddTrack Enterprise client, and two principal server components: AddTrack Import Manager (AIM) and Data Access Server (DAS). The data backend is logically divided into projects where each project belongs to one or more customers. Each project has one database for each type of data in that project. The data importer and data access service is shared between all projects.

When a file is uploaded the AIM reads each file, parses, verifies, and transforms the data to be suitable for storage in the target database. Which database the data from the file ends up in depends on the type of data the file represents and the source train set from which the received file originates from.

In case an end-user wants to access the data he/she, connects to the AddTrack server (DAS) using the AddTrack Enterprise Client, which handles data retrieval, via HTTPS. DAS retrieves the requested data by using stored procedures[1] in each database, and then sends all that raw data to the client. The AddTrack Enterprise client then performs all the necessary calculations and transformations on the data to make it suitable for presentation as information to the user.

When a user first starts AddTrack, a welcoming screen is shown and the user is asked to select which project and what type of data he/she wants to work with from a list of projects he/she has access to. After a project is selected, a filter panel is displayed, where the user can set restrictions on what data should be pulled from the server. The provided options includes filtering by train sets, event types, processes, subsystems, associated error codes, signal priorities, and

[1] A stored procedure is a subroutine in a relational database system that can be called by application programs to perform certain tasks.

the time they occurred. Selecting at least one time based filter is mandatory.

Users also need to specify one of three result types to retrieve data by: simple, detailed, and summary. The simple view gives only the most basic information about each event, like its type, the subsystem, train, car, and specific system process that has generated the event. A short textual description describing the event is also sent. The detailed view contains everything the simple view does and in addition to that it also contains the values of certain preselected on-board sensors and meters. These values are the values of selected system variables at the time when the event was triggered. In the case where some of these selected values contain GPS data, the user can also select one or more events to have the locations of the trains (at the time the events occurred) displayed through Google Earth. The user can, in both the detailed view and simple view, for each event request a snapshot of the values of all measured onboard systems.

Given the chosen settings all the data matching the filter is then pulled from the DAS to the client as a single data table. This is initially displayed to the user in its raw data table form. From here the user can visually inspect the data and/or perform one or more of several available analysis functions. These functions include viewing the data as several forms of graphs including bar, line, and stacked area graphs, as well as the proprietary Deviation Detection module which determines how much a given frequency of event occurrences deviates from the expected frequency, as calculated from historical data. There is also a function which determines, for a chosen event type, which other events occurred close in time to that event within the loaded data set.

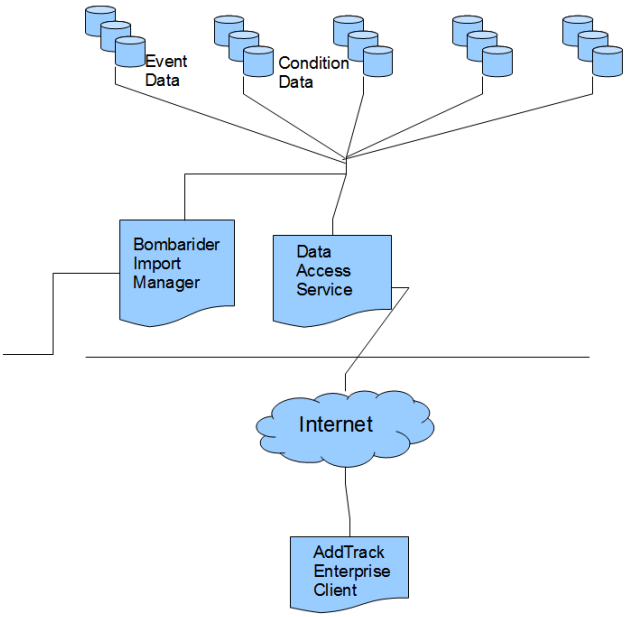

Figure 3.1: AddTrack 3G architecture

3.2 The drawbacks of the current approach

The approach of a client-server based architecture, with a single server application has shown to lack scalability and performance, as well as providing poor isolation between different customers.

In the current model the majority of the business logic is performed on the client-side. This approach limits the length of the time periods that can be analyzed. This limitation is a result of the massive amounts of data generated in those time periods combined with limitations to the client resources available. For some projects a time period window can be limited to weeks or even days, while the intention is to provide analysis for longer periods (months to years).

Another problem is maintainability. In AddTrack 3, a substantial part of the business logic resides in stored procedures in the databases, with part of the code tailored to meet specific needs of different customers, and the rest being common to all customers. Since there is no mechanism to share common code between the projects with the chosen solution, bug fixes and changes that affect the common parts must be done manually on each database,

Additionally since the server application is shared between all customers, project specific issues that negatively affect the stability of the system or cause an outage, might affect all customers regardless if they are part of the problem or not. Based on this Addiva has taken the initiative to develop a new version of AddTrack, AddTrack Enterprise 4G, based around a SOA. They have also expressed interest in using the OLAP capabilities offered in Microsoft SQL Server Analysis Services (SSAS) in the new AddTrack.

3.3 The issues that should be addressed and scope of implementation

In this thesis we discuss several issues including determining if SSAS can be used to solve the problem of handling rapidly growing data loads, investigating how the architectural changes affects security and propose a suitable model that accomodates these changes, investigating how SSAS can be integrated into AddTrack, what data suits itself for SSAS and for what data alternate solutions are needed, and finally investigate methods for how this data can be visualized in a good way.

Within the scope of this thesis project a data service for serving event data should be developed and implemented, a middleware service for abstracting the logical division between services and for controlling access to the services should be designed and implemented, and a proposal for a suitable method to visualize the data delivered through these services should be presented. The Event data service should be suitable to be used as a reference for implementation of additional data services for supporting additional file formats and data types. The data service should use SSAS for its data, in cases where possible. The

hypothesis is that the OLAP capabilities of SSAS will enable faster queries and a more consistent query performance that is independent of the total data load.

Chapter 4

Analysis

4.1 Security analysis

The change towards a SOA poses a different set of security concerns than what exists in a traditional client-server architecture. In SOA, a server is no longer a single coherent piece of software that performs everything, but an eco-system of loosely coupled services that are orchestrated together to perform the required tasks. They are exposed to each other through service endpoints, where an endpoint consists of an address and an interface served at that address. This means that the total potential exposed surface area that can be attacked is by default larger in the SOA approach. Steps have to be taken to secure each endpoint to only allow the intended users/services to access each endpoint.

There are two levels of software trust. Internal trust, which is the trust between the diffrent components that a service or application consists of, and external trust, which is trust for some external entity. There exists many models for how internal trust can be handled, one way is by requiring digital signing of software modules. The study of CBS trust and security models are outside the scope of this thesis and we refer to relevant litterature.

The loosely coupled nature of SOA has as a consequence that some of the internal trust bounadaries becomes external boundaries. By default each service has no inherent mechanism to check which external entity is invoking it unless that information is provided explicitly [BAB+07]. To resolve this, in cases where access to operations should be controlled and restricted, one can require callers to present credentials to the service which can then be authenticated and used to perform authorization of the user on the requested operation. The additional requirement of performing some form of authentication and authorization at each service endpoint poses a logistical problem on how to manage authoritative records of credentials and also where the responsibility of performing these tasks should be made.

Security related issues that are carried over from the traditional server-client

model are the issues of data confidentiality and integrity. When it comes to data confidentiality there exist two main approaches. The first one is to encrypt at the underlying transport level and the other is to individually encrypt each message.

Transport level security is highly dependent on the limitations underlying transport protocol. Protocols like HTTPS are technically mature, well understood, and widely available. Transport level security however is only on point-to-point basis which means that integrity and confidentiality cannot necessarily be guaranteed over several hops. This solution is suitable when communication is single hop or one has direct control over all hops between entities in the system. A solution using transport level security is tightly coupled to the choice of transport mechanism [MFT+08].

With message level security each application message is encrypted using some encryption mechanism (specifics varies between implementations), usually asymmetric encryption keys are involved in some way. This approach makes security independent from the underlying transport mechanism, enables having heterogeneous security architecture and provides end-to-end security. Message level security however does not support the use of message streaming. Instead each message is required to be individually encrypted. This, together with the fact that the method also requires implementations to understand the available security concepts present in the message format, has as a consequence that message-level based security implementations generally have lower performance then transport-level based security, since asymmetric key encryption is expensive [MFT+08].

4.2 SOA architecture

AddTrack 4G architecture is divided into three logical layers, the client layer, a front layer and a backend layer. The backend layer consists of several independent services logically divided by type and project. These services main responsibilities are to perform business logic and data management tasks. The following types of backend services are planned: Event Data Analysis-, Condition Data-, Version data-, and Data Recorder data-services. Each project will have one or more of these services depending of the needs of that particular customer. The backend data services are also responsible for abstracting customer specific differences and customizations. Scalability will be achieved by having a service for each project and data type combination. Each service can be scaled out to exist as several services/instances located at different machines.

The front layer consists of the AddTrack Service Manager (ASM) Service and the AddTrack Importer Service. The primary purpose of the Importer service is to parse and verify incoming files and send them on to the proper backend service for insertion into the data store. ASM has the task of authorizing the user for each operation called from the client and, if authorized, redirect each call to the appropriate service in the back-end layer. An ACL service keeps the authoritative records of user credentials. It is responsible for performing user

authentication.

This decoupling of functionality and responsibilities allows each service to grow independently of the others. Scalability is achieved by giving each subsystem its own dedicated pool of hardware resources and can (if needed be) be scaled-out to span over several physical machines by deploying copies of the services and then load-balance between them.

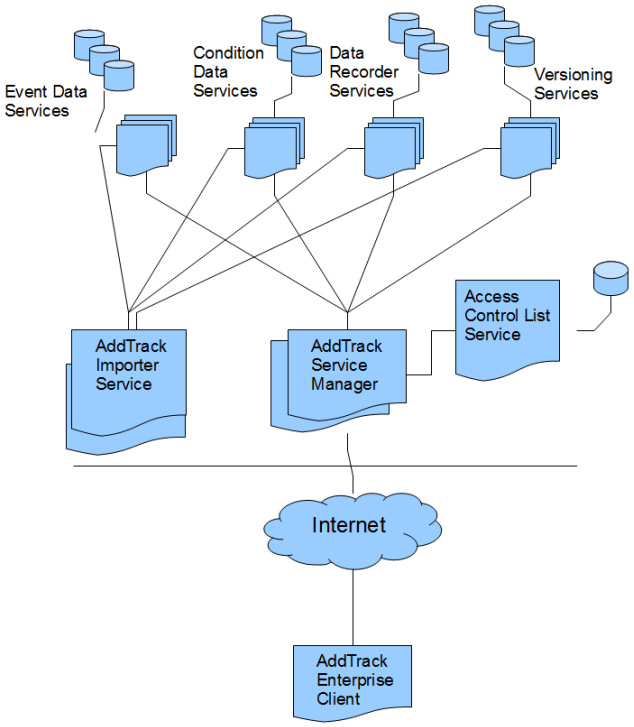

Figure 4.1: AddTrack 4G architecture

4.3 Evaluation of Analysis Services

SQL Server Analysis Services (SSAS) is a system within Microsoft SQL Server, which is a database management system. SSAS is a part of Microsoft's line of Business Intelligence products integrated into SQL Server. It provides OLAP and data mining capabilities for SQL Server data. It takes a neutral stance in the MOLAP vs. ROLAP debate, providing implementations for both methods. Accessing OLAP data in an application can be done through several different methods. XMLA which is a low level API based on XML can be used from any system that support sending and receiving HTTP and XML, ADOMD.NET which is an extension of ADO.NET which works on all managed code running on CLR platforms, and AMO which is an object based API which allows connecting

to and manipulating all objects contained in a running instance of Analysis Services.

The process of making data browsable through an OLAP cube in SSAS follows at a minimum the following steps:

1. Binding the SSAS project to an underlying data source. This can be a data warehouse, data mart or any type of relational SQL database.
2. Creating a view of the underlying data source that the cube can use. Common things done in this step is to denormalize the data to a star- or shallow snowflake-schema to simplify the cube and dimension design.
3. Design appropriate dimensions according to the created view.
4. Create a cube.
5. Create measures for each fact table. Measures based on the same fact table are grouped together in measure groups.
6. Deploy the solution to an instance of Analysis Services and process the cube.

In terms of OLAP the functionality offered by SSAS includes the ability to define measures based on built-in aggregation functions like sum, average over time, min, max, count and distinct count, as well as the ability to specify how aggregations should be made through the use of calculations and custom rollups. To be able to navigate all data SSAS allows one to specify user hierarchies based on the attributes in each dimension.

In addition to the OLAP functionality SSAS also offers advanced data mining capabilities which can be used on its own or be combined with the OLAP cubes as data sources. Data mining functionality includes Decision Trees, Neural Networks, Regression algorithms, sequence clustering, association rules learning and time series.

To determine what functionality (if any) could be suitable for the types of problems we faced with AddTrack we have performed some initial exploratory prototyping were we have created OLAP cubes and data mining models and structures based on copies of the existing data model. The databases used as samples have been based on real event and condition data for the projects RC2T44 and DM2.

The goals of this exploration have been to:

- Determine what elements in our data was interesting to perform OLAP analysis on.
- Determine if there was any interesting measures to perform data mining operations on and if so, using which algorithms.

- Construct a model proposal for what data analysis functionality to integrate into AddTrack.

During this exploratory prototyping we have quickly discovered that our data does not carry an abundance of interesting numerical facts (that have been in use in the current version). To make the highly normalized form of the source data, and to make it more manageable for use in our analysis model, many of the potential dimension tables were de-normalized in our data view.

Further, we have been unable to find features in the event data that produced interesting data mining results. If this lack of interesting results depended of our choice of how we had picked our data samples or if it is something inherent to the data model, was left unanswered. The decision was to not use the data mining features in SSAS for the new AddTrack 4 system.

For OLAP the frequency of event occurrences of different event types, and the total count of events was early identified as interesting candidate measures.

4.4 Performance

One assumption that the designers of the previous AddTrack generations have taken, has been the assumption that the average and peak data loads for all projects/customers would be similar to what is now observed in some of the more average projects. This assumption has been shown not to hold. One problematic project has been DM2. Their average data load is about two orders of magnitude greater than the assumed average load.

For AddTrack 4G the aim is to reduce the data loads pulled to the client and the stress that puts on the system by adopting a more interactive approach where the user is initially presented with highly aggregated data in the form of predesigned reports and dashboard style summaries combined with an interactive drilldown approach, directly inspired by OLAP, to allow the user to make more intelligent choices in what data to retrieve from the AddTrack servers. By incrementally only fetching the data that the user has specified an explicit interest in the potential amount data set to the clients in the average case is greatly reduced while still providing the user with an informative overview on both the fleet level and on individual trains.

Combine this with the scaling offered by the new architecture this allows for better control over the impact the flow of data in the system has on overall system performance. Assuming of course that the user base can adapt their behaviors in how they use AddTrack to fit with the intent of the new approach. This however remains to be seen once the new AddTrack becomes available to end-users.

The initially estimated usage scenarios that are expected are something in line with 200 users per project (not necessarily concurrent), with a peak load of about 200-300 MB of data per request per user.

The main performance measure we would like to optimize is minimizing the perceived delay when making queries for the end-users. A secondary measure is minimizing data latency (the time it takes between a change to the data occurs to when that change is visible to users). These two measures sometimes conflict. Our default approach of handling this is to primarily optimize for the primary measure and only optimize for the secondary measure on explicit customer request on a project by project basis.

Chapter 5

Modeling

5.1 Data analysis model

Data in AddTrack originates from the flat data files sent from each train. These files can be in one of three different file formats:

- OTI (MITRAC TDS Off-board Tool Interface) [Koc09]
- TDF (Train Diagnostics system File format)
- XML

OTI and TDF are proprietary file formats that use binary encoding of data. XML is an open, text-based format.

For the purpose of this thesis the event data files of the OTI format have been chosen as the reference to model due to the facr that event data is the most commonly analyzed data by AddTrack users and that the majority of AddTrack projects use OTI as the format for their data files.

The core of the data model is the event data and environment sample tables. Each entry in the event data table represents a single unique occurrence of an event in one of the on-board systems. Each received event is accompanied by a set of samplings of various program variables which represents various sensors and meters on-board the train. These are stored in the environment sample table in a packed binary format. Each sample entry represents a full view of all components measured for a specific event type at a given instant in time. To each event belongs at least one such sample, the environment as it was at the instant of time when the event triggered. Events can have additional samplings taken at times before and/or after the time of the triggering. These are, for each unique event occurrence, uniquely identified by the order in which the samplings have been taken, with the sample taken at the time of the event triggering given time index 0. The time between samplings can vary between

event types. Exactly what data is sampled is specific to each project, and can even vary between trains within the same project.

To unpack and interpret what the sample data means the data model also has a list of signal definitions which is tied to the specific environment an event occurred in, which tells how a specific sample value should be extracted and transformed to be meaningful, as well as what the value represents. The event data table also contains references to other tables in a snowflake schema that represents information about where an event occurred.

As a basis for our Analysis Services data model, we have initially used the data model currently used by AddTrack 3 as used by our sample projects (used during exploratory prototyping, see Section 4.3). This model has been altered to expose a denormalized view in which descriptive text string data has been directly associated with the corresponding data rows instead of stored separately. The data tables that denote car and unit type for train sets has also been merged into their corresponding car and unit tables.

During implementation several flaws has been discovered in the existing model which required making alterations to the model. The final data model is shown in Figure 5.1.

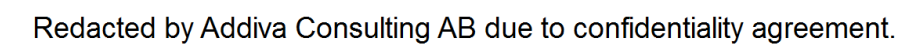

Figure 5.1: AddTrack 4G Data model

From this model we have isolated several tables that were interesting to use as dimensions to organize our data along. These dimensions are as follows:

- Trains (Consisting of a hierarchy of both unit and cars).
- Subsystems (which is a logical view of the organization of the onboard systems).
- Location (which is the physical location of an onboard system).
- Process (the software process that generated the event).
- Event Description (a description of the event type).
- Priority (the seriousness of the event).
- Error Code (an associated code denoting a specific type of error).

- Version (The version of the file format definition that was valid when the event occured).
- Event Time (the time the event occurred).
- File (through which dump file the event was inserted through).

In addition to these dimensions additional dimensions have been constructed to help in tying together the event data with its corresponding environment sample data when performing drillthroughs down to individual data rows.

When constructing the OLAP cube the primary fact table of interest has been the event data table. When drillthrough functionality has been added the need for additional measure groups to use as intermediaries has been discovered, after the requirements for what data to display when doing drillthroughs has been more accurately defined. Also the need to be able to see the sampling values for event time index 0 for each event in their unpacked form in the drillthroughs was desired, as well as to drillthrough on each event to retrieve the full set of environment data associated with each event.

5.2 Security

5.2.1 Common architectural security patterns

Direct authentication

In the direct authentication pattern each service is equipped with its own local authentication and authorization logic based on its own local identity store. This is suitable when an application only consists of a single service or when it is the only service in the system requiring security (the other services being unsecured services allowing anonymous calls).

Advantages:

- Easy to implement.
- No single point of failure.

Disadvantages:

- Does not handle scaling well.
- The logistics of synchronizing identities between several separate identity stores becomes a hindrance.

Gateway - Identity Delegation

In this pattern there is a single service at the front of the system through which all service consumer requests have to pass. Authentication is done by the gateway service. When a service consumer makes a request the gateway service makes the requests on one or more backend resources on behalf of the consumer by impersonating the consumer. When impersonating, the impersonating service assumes the identity of the original caller, authorization is done at the backend resource using that identity. In this pattern, the existence of a centralized service responsible for maintaining an authoritative identity store for identities used in the system, is common.

Advantages:

- Information about backend resources are hidden from the clients, resulting in a less exposed surface area for an attack.
- Impersonation allows for fine grained access control at backend resources.
- Impersonation allows for detailed security audit information at backend resources.
- Centralized identity store simplifies identity management

Disadvantages:

- Impersonation requires information about user identities to be available to backend resources.
- Impersonation requires complex authorization logic to be present at backend resources.
- User credentials has to be exchanged with each call/session.
- Gateway service can potentially become a bottleneck.

Gateway - Trusted Subsystem

This pattern is very similar to the previous pattern, with some key differences. Most notable is that the gateway service does not impersonate the service consumer when making requests on backend resources. Instead it uses its own credentials for authentication and authorization. This isolates the backend resources from the specifics of who the original caller is and thus eliminates the need for them to keep track of identities, instead only requires keeping track of the single identity of the trusted subsystem (gateway). This pattern assumes that there is an established full trust relationship between the gateway subsystem and all the backend services/resources it needs to access.

Advantages:

- Information about backend resources are hidden from the clients, resulting in a less exposed surface area for an attack.
- Impossible for regular users to connect directly to backend resources even if information required for that would leak out.
- Centralized authorization management.
- No need for having complex authentication and authorization logic at backend resources.
- Centralized identity store simplifies identity management

Disadvantages:

- No user identity information available at backends means no advanced security auditing possible.
- If the gateway service is compromised attackers can potentially gain unrestricted access to the entire system, which means a single point of failure.
- Gateway service can potentially become a bottleneck.

Brokered Authentication

This pattern is useful in situations where a client needs to access several different services with a single sign on, the service and the client do not trust each other to handle credentials securely and/or the identity store and the service does not trust each other directly. For brokered authentication the system contains a special authentication broker service which has direct access to the identity store.

When a client wants to make a request on a service it first sends its credentials to the authentication broker. The broker authenticates the user against the identity store and issues a security token to the client. The client then makes the request on the desired service attaching the security token to the request. The service validates the token and sends back its response of the token passes validation. The token can be reused by the client for more requests during the specified lifetime of the token. This approach and related approaches commonly is coupled with having central service registries to allow for clients to dynamically discover services.

Advantages:

- Centralized authentication management.
- Does not impose a specific topology on the architecture.
- User credentials is only exchanged between client and authentication broker and only once.

Disadvantages:

- Authentication broker service adds to overall system complexity.
- No common system for authorization policy enforcement.

Federated Brokered Authentication

Federated authentication brokering is commonly used in business-to-business scenarios and other scenarios where services is required to communicate across two or more security domains possibly with no common identity management infrastructure. This approach depends on the existence of established trust between the participating business entities.

Advantages:

- Centralized authentication management.
- Can cope with multiple disjoint security domains.
- Does not impose a specific topology on the architecture.
- User credentials is only exchanged between client and authentication broker and only once.

Disadvantages:

- Authentication broker service adds to overall system complexity.
- No common system for authorization policy enforcement.
- Management overhead that only pays off if system requires usage over multiple security domains.

5.2.2 Choosing an appropriate model for AddTrack

For the AddTrack system the following requirements were given:

- The chosen model should be easy to maintain.
- The client application must never require direct access to a backend resource.
- The services cannot make assumptions about which network or security domain each service resides in.
- The system has to be able to cope with scaling out and/or adding new services.

- A logged in user should automatically be logged out from the whole system after 15 minutes of inactivity.
- A user account (set of credentials) is not allowed to be logged in to the system simultaneously multiple times.
- Credentials and data must be held confidential.

Given this list of requirements and the proposed architecture, the choice of security model can be limited as described in the following text.

The Direct Authentication at each service approach is unsuitable as given the large number of services (that the system will consist of), since it is unfeasible for each service to maintain its own authentication and authorization. Also choosing this model would violate the requirement that the system should not allow users to connect directly to the backend data services.

The federated authentication broker model is also unsuitable for this system as it is unnecessarily complex for our purposes. The system will not have any of the problems of crossing multiple disjoint security domains that the model is designed to overcome. This model and the brokered authentication model also does nothing to enforce the requirement of not letting users have direct access to backend resources.

This leaves us with the class of models commonly known as Message Interceptor Gateways. This approach can be combined with two basic methods for propagating credentials in the system, one where the gateway service impersonates the caller when in turn calling the backend resource, and the other where the gateway is trusted by the backend resource to authenticate and authorize the caller and then calling the backend resource with its own identity.

Both models provide an intermediary between the calling client and the backend resource, which fits our requirements. Since all calls gets routed through the gateway this allows us to have a single sign-on solution and to handle the requirement of automatically logging out the user if there is more than 15 minutes of inactivity. The impersonation approach enables the possibility of having detailed security auditing capabilities on every individual service in the system, although that comes at the price of having the same level of authentication and authorization logic at each secured service. In comparison, the trusted subsystem approach centralizes all the complex authorization logic to the service designated as the trusted subsystem. This allows us to greatly simplify the authorization logic at the backend resources to only have to authorize the trusted subsystem, but at the cost of losing the ability to perform detailed security auditing on those resources [MFT+08].

Since there is no stated requirement of the system requiring detailed security auditing of individual services and a stated requirement of keeping it simple. Our choice of model to adapt a message interceptor gateway based model to secure the AddTrack 4G System, using the trusted subsystem approach to secure backend services and resources. This choice allows us to minimize the need for complex authorization logic for the majority of our services and instead have

a simpler implementation that only checks if the identity of the calling service is the identity of the trusted subsystem or not. The trusted subsystem will of course have to have more complex authorization logic, since it will be responsible for fine grained authorization of users before calling the appropriate backend services.

Since the backend services only allows access to the trusted subsystem identity, there are less identities that are possible to get compromised, this reduces the surface area that is possible to attack. However, this also puts extra focus on the need to protect the credentials for the identity of the trusted subsystem and the trusted subsystem itself, since if this service gets compromised it could potentially leave the entire system exposed.

5.3 Evaluating visualization techniques

For AddTrack 4, one of the goals was to improve both how data was displayed to users and how users navigate the data, to an approach that provided users with a better overview of the data without losing the ability to examine it close up in great detail.

In AddTrack 3 visualization can be divided into two basic approaches. A tabular (textual) view and a graph view (can one of a multitude of basic 2D and 3D graphs). In addition to this environment sample data, which represents a time series over a given variable, is visualized using a set of separate line graphs (one per sequence).

The way AddTrack 3 displays data is non-interactive. The flow is such that before any data is displayed one must first fully select all criteria by which to filter ones query and select how it should be visualized. If one would like to take a closer look at only some more specific area, or one would like to change the granularity of the data one has to go back and redo the query from scratch. Also in AddTrack 3 summaries is limited to summaries over data organized by time, cars and event type, with time granularity as the only variable.

OLAP in this context offers both more flexibility in terms of data dimensions which can be summarized by, as well as consistent performance which is not dependent of the total amount of data in the database when performing these sorts of ad hoc queries.

This and the higher dimensional nature of OLAP queries makes visualizing this using traditional flat tabular and 2D graphing techniques impractical, since one either has to project the data to a flat format or restrict oneself to only ever viewing 2D slices of the data at any given time.

A visualizer for OLAP has to take into consideration the hierarchical structuring possible within dimensions and enable the users to navigate up and down the levels of these hierarchies.

In terms of how different methods compare for different tasks Margehscu writes that for detecting outliers in data Multiple Line Graphs, Permutation Matrix, Survey Plot, Scatter Plot Matrix, Parallel Coordinates and Principal component analysis are effective methods. For detecting clusters Self Organizing Maps are most effective. Survey Plot, Tree map, and Sammon's Mapping are effective in revealing classes in the data and of those Tree map is the most effective in deriving class descriptions. For making comparisons between entities in the data Permutation Matrix, Survey Plot, and Parallel Coordinates can be used effectively [Mar08].

For AddTrack 4 the visualization systems can be divided broadly into two areas: Fixed dashboard style summaries of pre-aggregated coarse-grained data, and visualizing ad hoc style OLAP queries and data drillthroughs.

For data drillthrough, the data is mostly textual in nature. Since the grain of the data is individual event occurrences and the important information contained in at this level does not suit itself to graphical displays without losing some detail, the most suitable way of presenting this data would still be in a tabular format. However to avoid having to send a large amount of data to the client, encouraging the users to narrow the selection of data even further through drill-down operations on the data through the other summarized views before requesting the details would have to be enforced.

For the predetermined summaries ,the data mostly represents variations in event frequency over a fixed interval of time for a given dimension of study. It can, for example, be variation in event occurrences per train in the fleet over the last 30 days, the total count of events per subsystem over a selected car for the present day, the total counts of events per train for the entire fleet for the last 7 days, or other similar measures.

The primary usage of this would be for finding deviations from the usually expected frequencies of events based either by past behavior on the same train/ car/subsystem or by other trains/car/subsystems of the same type within the fleet. This would be for determining which regions of the data to perform a closer inspection on. For this visualization techniques that are good at presenting outliers in a clear manner would be suitable. It would seem multiple line graphs and/or survey plots could be used effectively for this since they are, as previously stated, good at detecting outliers in data.

An example how this could be used in the context of AddTrack 4 is show in Figure 5.2. Each subgraph represents a train set and each horizontal line represents an event type. The length of the line represents the number of occurrences of that event type over the displayed period.

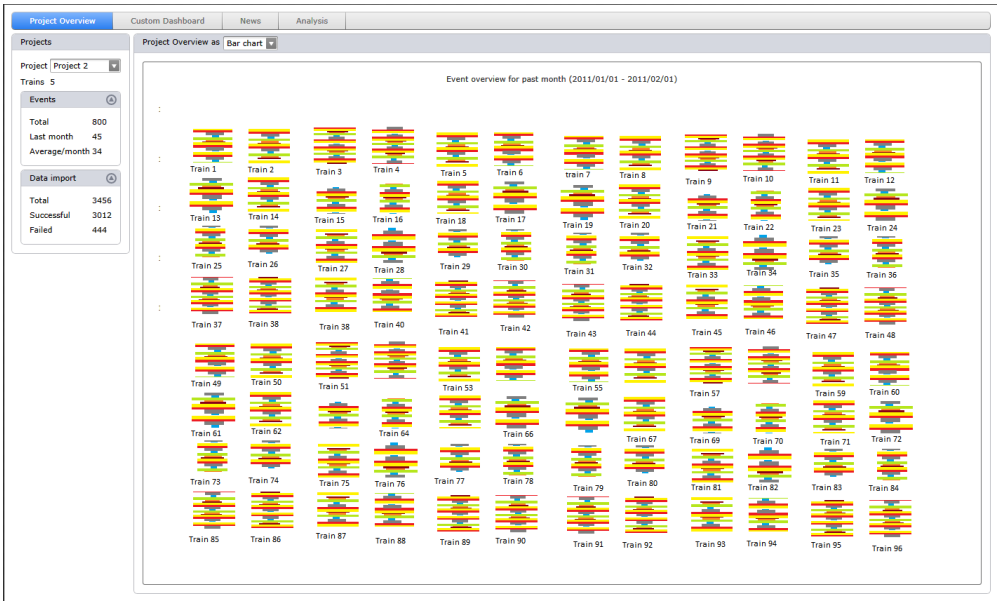

Figure 5.2: AddTrack 4 mock-up of survey plot usage

By clicking on one of the individual survey plots one would drilldown on that train and display summarized data at a lower granularity. An example of this is shown in Figure 5.3. In this example the left side graph represents the total event count for some event types, and the right side graph represents the degree of deviation from the historical average frequency.

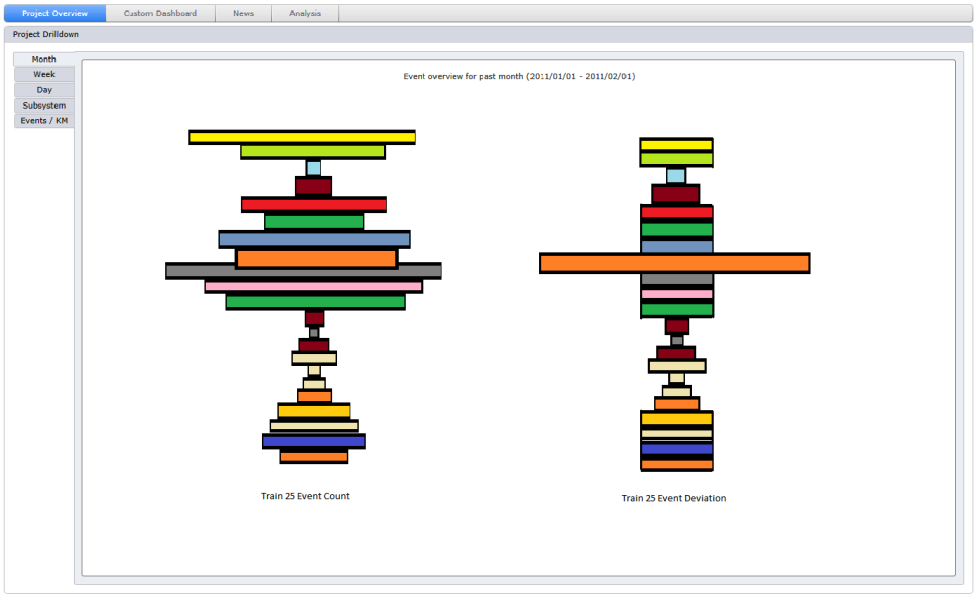

Figure 5.3: AddTrack 4 mock-up of survey plot drilldown

Visualizing the result of ad hoc queries is more problematic. The underlying OLAP engine and the AddTrack query framework supports OLAP queries that can return n-dimensional data slices as result sets. The arbitrary nature of the dimensionality of the results and the two-dimensional nature of the display area has to be resolved. This mismatch can partially be resolved by forcing restrictions on how the users can make their queries and by projecting higher dimensional result sets down to two dimensions. This would however mean the power and expressiveness of OLAP is not used to its fullest. Also since what properties of the data which needs to be most clearly made visible in these ad hoc queries is not as clear cut as in the predetermined dashboard information displays, determining which methods are most suitable becomes a lot harder.

Marghescu claims that no single visualization technique can make clear all possible properties of the data, and that using multiple different ways of visualizing the data should be used [Mar08]. Based on this it can be determined that, for AddTrack 4, several methods should be chosen and these methods should preferably complement each other. For this selection multiple line graphs, Tree maps, and survey plots, and combining this with a tabular display, like a Pivot grid or the method presented in [TD05], would give a solution that covers most eventualities and allows for effective, interactive navigation of the cubes.

An example mock-up of how this could look like is depicted in Figure 5.4. Selection of data is done through the tree view and by selecting hierarchy members on each bar stick. When the number of events has been filtered enough one can drillthrough on the OLAP data to retrieve the underlying event data in the selected cell, which will be displayed in its own table. On the right side of Figure 5.4, a multiple line graph of the environment data for the event instance selected on the data grid is shown.

Figure 5.4: AddTrack 4 mock-up of ad hoc HDDV view

Chapter 6

Implementation

For this thesis the scope of the implementation was limited to a non-project specific variant of the Event Data backend service, the parts of the Gateway service which exposes the functionality provided by the Event Data service and the user authentication logic in the ACL service, and designing the data models for the various data stores that the services requires to function.

6.1 Data backend

The data backend layer consists of, for each project, a SQL database which is used as the core data store and an Analysis Services MOLAP database which stores the precomputed aggregations and an indexed copy of the fact data. The data model used by the SQL database is the model described in Section 5.1 and Figure 5.1. This model was then further transformed in SSAS to the denormalized model seen in Figure 6.1, to further simplify design.

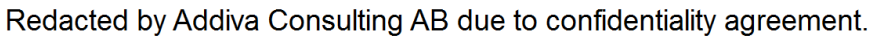

Figure 6.1: AddTrack 4G denormalized data view

To keep the SQL and SSAS database synchronized a combining Microsoft's Proactive Caching functionality and a SSIS package containing a handwritten partitioning and processing script was devised. The responsibilities between the

two parts have been divided as described in the following text.

The SSAS database initiates incremental reprocessing whenever a change is detected, which is the insertion of new rows in the underlying tables the SSAS measure groups is based on as well as the corresponding fact-related dimensions. The other dimensions are processed in a similar manner except the Trains, Event Time and Default Environment Signal dimensions which behaves like Type-2 slowly changing dimensions. For these dimensions a full process is done whenever change is detected. This allows for having a low latency on data updates while still leveraging the precomputed and optimized nature of MOLAP storage.

When it comes to the fact data for the event data measure group, all attributes and relations except one does not change once the record has been inserted into the database. The one relationship that breaks this is the foreign key relating to Event Time representing the time an event stopped being active (its End Time). For events which have been active when they were dumped to file, the End time attribute is missing since the events have not ended at those times. The events which have been active during a particular dump is sent again in the next dump after they are no longer active. At that point the correct end time is added to the events entry in the database.

This combined with the way measure group partitions is incrementally processed means Proactive Caching is in itself not sufficient for the needs for AddTrack. When a partition is incrementally processed a new temporary partition is created which spans the new data. This partition is processed, indexed and merged with the existing partition. This method of processing can over time result in fragmentation in the partition data, which have a negative impact of query performance.

To prevent this, a SSIS package have been created which periodically realigns the partitioning scheme and fully reprocesses the partitions. The interval in which the partitions are reprocessed varies with how wide each partition's span is. For example a partition spanning the time period representing the current day is processed more often than an old partition spanning the period that is the previous year. The script itself uses AMO objects to generate the required XMLA scripts to achieve the desired result.

Due to the nature of how the data is partitioned by time and the fact that events is not necessarily uploaded in order or during a fixed time from when they actually happened, an event that does not belong to the latest partition, but to an older partition, could be uploaded at any time. Since the partitions that still can receive additional events or End Time updates is not predictable all partitions must be reprocessed occasionally to ensure the data in the cube is consistent with the source data.

6.2 Services

6.2.1 Event Data Service

The Event Data Service was implemented using WCF for communication with other services and ADOMD.NET to retrieve data from the Analysis Services database. The service provider implementation itself is composed of two separate major components. A data import module, which performs transformation and loading of data from the import stage. The specifics of the implementation of this module are outside of the scope of this thesis.

The other module handles retrieval of the stored data. Some of the preparation of the data for analysis is performed in the Analysis Services engine. The parts that have been unable to be implemented in SSAS have been done by this module of the Event Data service. This includes unpacking environment sample data, transformation to a technology independent data format for transmission through the SOA architecture and business logic functions like filtering event data by temporal proximity to certain other events.

For some data that rarely changes the Event Data service caches the data using an AppFabric Cache Cluster. AppFabric Caching Service offers an out-of-process, in-memory distributed cache. This allows sharing of cached data between service instances, even across application domains and over machine boundaries. That means the service can be scaled-out without having redundant caches for each host process.

6.2.2 AddTrack Service Manager

The tasks performed by this service is to perform user authorization and forwarding incoming calls to a suitable service at the backend of AddTrack that can provide the appropriate response. For user authentication and authorization the extensible architecture of WCF has been leveraged. To allow smooth integration of the functionality provided by our ACL service into the authorization logic used by the ASM. We have provided our own implementations of key components of the authentication and authorization components in the WCF framework.

For the authentication of users the ASM delegates the responsibility completely to the ACL service. After a user has been authenticated the ASM uses an WCF authorization policy to populate a rich claim set with the system permissions assigned to that user, as provided to it by the ACL service. This claim set is then kept for the duration of the active session and used to authorize the user on a per operation basis. Operations is categorized according to whether if they have to be applied for a specific project or not and which type of backend service has to be bound to get the proper response.

During the initial design this coarse grained authorization was sufficient to meet the requirements set on AddTrack 4. However, halfway through the implemen-

tation a potential change to the requirements has been brought to attention. The proposed change has been related to adding the ability to restrict access on the data level to specific subsets of the data based on customer requirements. Although, at that time, this has not been a fully stated requirement, several possible solutions to adapt the design, have been discussed. This was narrowed down to two potential solutions, with both solutions having their own set of issues that prevented them from being integrated into the current solution smoothly. Since this has only been an anticipated future requirement the decision of which solution to use, was pushed to the future and the implementation of such an addition was deemed to be outside the scope of this thesis work.

Due to the nature of the semi-static nature of SSAS while running in MOLAP mode, once a process cycle is finished, the data served by the database does not change until the next processing cycle. For the duration of the window between processing cycles, data request calls on the read-only methods exposed by the Event Data contract can be considered referentially transparent. For the ASM to determine when the window changes the Event Service exposes a method which gives the timestamp of when the last processing that potentially affected any element in the cube was completed. This value is requested before each call to the cache and compared to the value of the last known change to determine if additional changes has been made.

Once a request has been authorized, the ASM looks up if the requested data is already present in the AppFabric Cache. The ASM caches all responses for which the referential transparency assumptions hold for as long as they hold. When the start of a new processing window is detected all cached data for a given project is invalidated, since the referential transparency assumption does not hold between processing windows. This use of caching gives increased performance in multi-user scenarios where several users makes the same set of queries within the same processing window.

When the ASM cannot satisfy a request directly from the cache, the ASM looks up the endpoint address(es) for the specific requested service from a database which lists each available service, its endpoint address, and the type of service exposed at that endpoint. Based on the specified project the ASM then makes a routing decision to forward the request. The type of the services under consideration is implicitly known from which method on the ASM's contract. When the reply is received it is added to the cache before forwarding it to the client for future requests.

6.3 The AddTrack development and testing environment

To secure each service according to the chosen security model (described in Section 5.2.2), all of the different services have been granted a X.509 certificate. Each backend service have been given an authorization manager WCF plug-in which only allows calls to its methods if the identity presented by the calling

services matches a known list of specified certificates, in addition to the service being able to properly authenticate the certificate as a trusted certificate.

For hosting the services AppFabric Hosting service running on top Internet Information Services on a virtualized server running Windows Server 2008 R2 has been used. A separate virtual server running the same operating system has been used for the backend databases. A single host AppFabric Cache Cluster has also been setup on the same virtual machine as the hosted services, using the database server as a cluster manager.

The testing environment setup does not fully leverage the scale-out support implemented into AddTrack 4, since it runs on only two servers. This has been done for cost reasons and that scaling out on the same virtual server offers no additional benefit as long as the virtual server still can be scaled up.

Chapter 7

Verification

7.1 Testing

To facilitate testing of the server architecture for functionality, a test client has been prototyped. This client has changed several times during both the development and testing cycles. For the first set of tests all the software has run on the same machine. For the further testing the server side setup has been implemented as described in Section 6.3. The virtual servers have been configured with 4 GB of RAM each, two 1.60 GHz Intel Xeon cores on the application server, and four 1.60 GHz Intel Xeon cores on the database server. The client has run on a laptop with an Intel i5 2.67GHz processor, with 4 GB RAM, running Windows 7 professional.

Since AddTrack 4 as a product is still in its early development stages, it has not been tested in a full scale production environment yet.

The data sets used for testing have been based on a selection of data duplicated from the live environment. The data was taken from projects DM2 and CHE, restricted to chosen days in June, August, and September 2011, as well as initial testing has been performed on a data set based on the project RC2T44 spanning all dates from January 2010 to June 2011.

7.1.1 AddTrack Performance

The main performance counter that has been measured has been total roundtrip time for each operation that the interface exposes. The time required to bind the result to the display component has been measured separately. These two measures combined provides the total time between user action and the result being displayed without the interference of the measuring instrumentation. This is in line with the main performance goal of minimizing the perceived delay from input to response as the user experiences it.

The display components used for the tests have been a WPF Data Grid (bound to a set of data row objects designed to give the appearance of a pivot grid like layout), seven different WPF data grids each data-bound to a list of POCO objects, a text box into which formatted sample data has been written in plain, a WPF tree view into which the available dimensional hierarchies has been written, and a WPF data grid bound to the raw event data.

The tested operations have been:

- Logging in into the system.
- Retrieving all the dimensional hierarchies that have been available to use for making OLAP queries.
- Retrieving all the data members for each of the dimensions used in filtering (trains, event type, process, error code, subsystem, location, priority, and version).
- Summary query requesting an event count summary calculated over the entire database for the dimensions event start time and trains (per train set level).
- A drillthrough action on a cell with less than ten (10) events.
- A drillthrough action on a cell with over 20000 events.
- A drilldown on the current year.
- A sample data request on the first event retrieved in each drillthrough.

The measured time have been compared to equivalent functions in AddTrack 3, or to the function closest matching if no exact equivalent existed.

The initial round of testing have showed a consistent slowdown on requests in AddTrack 4 performance at about twice the response time for all but the smallest of response sets. After investigation this have been tracked down to the choice of security mode and transport on the last hop from Gateway service to the client. At the time security mode has been set to use message-based security with HTTP as the transport protocol used. After changing the security mode to use transport-based security with attached message-based credentials using HTTPS as the transport protocol on the hop between gateway and client, and also changing the link from gateway to backend to use TCP/IP as the transport protocol, a significant decrease of response time have been observed. These preliminary results have indicated that the system operated within tolerable limits. Some uncertainty in the results have existed since the test results for the AddTrack 4 system have showed a higher degree of variance than the results for AddTrack 3.

Once the system was running in the testing environment another performance bottleneck have been discovered, which prompted a redesign of the data model. This have been discovered in the results for the case of measuring response times for data drillthrough using the detailed view, where the response time started

to grow unexpectedly as the number of listed error codes in the system have grown due to a known data loading defect. The design has been changed to always assume there are exactly four error codes associated with each event. In the case where certain error codes are missing in the source data, these are filled with default values instead. This has change allowed us to reduce some of the data post processing done in the Event Data service.

For the final performance tests the measured response times for data drillthrough has been very close to the measured equivalent AddTrack 3 functions, the sample data retrieval time has been slightly slower than the AddTrack 3 but still deemed to be small enough to be subjectively undistinguishable. The summary functionality in AddTrack 4 is significantly different from AddTrack 3 that the measured values are not meaningful to compare directly, the result of some measured values is however shown for completeness.

7.1.2 AddTrack Security

The security requirements for AddTrack 4 is described in Section 5.2.2. For the purpose of testing the model that has been chosen, these requirements can be summarized to the following:

- User credentials must be held confidential.
- Data must be held confidential.
- Users are only allowed to directly access the designated trusted subsystem.
- Only subsystems designated as trusted are allowed access to other services in the system.
- Users are only allowed to view data for the projects for which the business agreement states they should be able to view.

Since the AddTrack 4 product has been in a very early development stage during the course of this thesis work, a lot of planned components and subsystems have not existed at the time of writing this thesis. This means that we have been unable to test this system as a whole. Because of that it is impossible to predict how secure the final AddTrack product will be. This means all the security testing described in this thesis should be performed again at a later point in time, when the product is closer to be complete.

For the subsystems that have existed at the time of writing this thesis a testing regimen based on performing security assessment combined with a security review has been devised.

All potential points of vulnerability have been identified. All the (implemented) services: ASM, Event Data Service(s), and ACL service have been inspected to determine the extent of their exposed contact surfaces. The databases used by each service have been inspected in a similar manner. This has been done to identify possible attack vectors.

After this the list of possible attack vectors have been inspected for redundancies with a simple reasoning in mind. If two services in the system are based on exactly the same code and, apart from their endpoint address, and are configured in exactly the same way, then logically they are assumed to have the same weaknesses.

The aspects that we have studied have included endpoint configurations, user credentials, service credentials, service source code, and generated logs. Limited probing on the endpoints of the services in the backend layer have been done to ensure that directly accessing back-end services is not possible (which it should not be). User validation has also been tested with regards to both real and bogus user credentials. These tests has been performed using a specially developed test client designed to allow specification of all settings nessecary to connect to each tested endpoint.

Chapter 8

Results

8.1 Results

The results from the final round of performance testing is shown in Table 8.1.

	AddTrack 3 (average)	**AddTrack 4 (average)**
Simple data	0.8 s	1.3 s
Detailed data	2.2 s	2.4 s
Summary	4.18 s	3.49 s
Filter options [1]	3 s	4.75 s
Query options	N/A	4.2 s

Table 8.1: Average times from final set of performance tests

Some of the differences in the measured performance can be explained by AddTrack 4 including columns that AddTrack 3 does not transmit, for example all four possible error code columns and the file definition version. When accounting for these differences the data retrieval times are very close. When it comes to the filter option data AddTrack 4 includes full support to filter specific items by a particular definition version, which AddTrack 3 does not support. As a consequence of this the total amount of data sent by AddTrack 3 is in this case a lot less.

[1] Measured as the total time for all operations required to get all the data

The results from service endpoint probing is shown in Table 8.2.

Probe test	**Expected Outcome**	**Result**
ASM access, valid credentials	access granted	access granted
ASM access, invalid credentials	access denied	authentication error [2]
ASM access, multiple logins	authentication error	authentication error
Event direct access, invalid cert.	authentication error	authentication error
Event direct access, same name	access denied	access denied
Event direct access, valid cert.	access granted	access granted

Table 8.2: Results from endpoint probing with different settings.

[2] In the Table 8.2 the term "authentication error" means the service has thown a SecurityTokenValidation exception.

Chapter 9

Discussion

Within the scope of this thesis the following has been accomplished:

- A study of related research in the fields of OLAP and SOA.
- A model security handling in AddTrack 4 has been designed.
- An implementation of a data service for OTI event data has been done.
- A partial implementation of the system's ACL service has been provided.
- A partial implementation of the ASM has been provided.
- A short study in visualization techniques that could be used for AddTrack 4 has been conducted.

We have started our work with expectations that using SSAS would greatly improve query performance. However, after having conducted detailed study and series of tests, we have been able to conclude that our findings do not comply with our expectation. We do however have to take into account the uncertainty added by the fact that the data sent by the different versions of AddTrack, we have tested on, is not exactly identical. Since the uncertainty in the result is larger than the measured differences, the only conclusion we can draw is that the results are inconclusive. Futhermore, early indications show that there might be some potential benefits for some cases of summarized data. This point would warrant further study.

The real measure of interest is how well the system performs under load in a multiuser scenario. As the system itself has not been completed enough, at the time when tests have been performed, no multiuser testing using actual usage patterns has been performed.

The security tests have not uncovered any vulnerabilities. Having in mind the inability for these tests to be exhaustive over the entire AddTrack 4 system, the findings do not allow us to generalize that weaknesses in the system do not exist.

It is therefore strongly recommended that more testing should be performed in the future, covering both new subsystems and already tested subsystems, once AddTrack 4 is feature complete.

We have found out that for data visualization a combination of different approaches works the best. Therefore, our suggestion is to use multiple line graphs, tree maps, and survey plots for visualizing specific slices with possible drilldown by clicking on various elements in the graphs, and a HDDV view or similar for navigating the data in ad hoc queries.

One problem we have encountered during development has been the clunkyness of the ADOMD.NET API. Even though it is based on ADO.NET, it still lacks the advanced higher level programming interfaces and metaphors, like LINQ and the Entity-Relation model (Entity Framework), that the relational ADO.NET has. This is an area where the market, even through OLAP technology and ADOMD has been around for a long time, needs more time to provide adequate solutions.

9.1 Scaling out

When it comes to scaling out AddTrack 4 services several actions can be taken. The most basic one would be to move out the resources of the most heavily used projects to their own dedicated machines.

In addition, the data analysis backend can be scaled out by deploying several instances of SSAS, designate one of them as a staging server which performs processing, and then use the XMLA Sync command to synchronize the contents of each additional database. Combining this by deploying multiple copies of the same backend data service and point each of them to its own SSAS database instance allows balancing out the load between them. Minor changes to the ASM would enable proper load balancing.

Multiple copies of the ASM can also be deployed to accommodate additional concurrent users of the system. This however would require the addition of a thin load balancing layer infront of the service.

In essence the AddTrack 4 architecture allows for a great deal of scaling in the case when scaling up the available hardware becomes infeasible. However it is not expected that all of these options would be required all the time in a full scale production environment.

9.2 Unresolved issues

One issue that has not been resolved within the course of this thesis, due to the timing constraints, has been to implement an equivalent Entity Framework based solution for the key data retrieval functions. If implemented, this would

enable a more directly comparable baseline for our performance test and we would be able to more accurately determine the possible benefits of Analysis Services. Since AddTrack 3 is not one-to-one with our AddTrack 4 solution, in terms of the exact functionality, there is a high degree of uncertainty in our test results. Such a solution could also have been used as a fallback in the case, that the SSAS based solution would have been found to not measure up to the performance requirements.

Another issue that has been discovered late in our work, and thus has not been addressed, has been the storage size of the sample stream attribute. We have found that the storage size of the environment sample stream attribute on the environment sample dimension has been growing more rapidly than expected. We have determined that factoring out that attribute from the fact table to its own dedicated dimension table could save up to about 25% of the storage space, but would require changes to be done to the data importer component in the service, that could potentially have a negative performance impact on the data importer.

An alternative solution, which would require even more changes to the data model, has also been proposed. This solution could eliminate the need for including environment sample data in the analysis model entirely, but at the cost of query performance for queries that do require that data and a significant time investment redesigning the analysis data model and rewriting the existing implementation. This proposal was submitted as a possible future change to the AddTrack development team.

The third issue that has been left unresolved was a change in requirements for the security model. According to the new requirement, the system has to be able to have fine grained access control to data in the databases based on specific data members in the dimensions. Two possible solutions has been discussed. The first solution would involve adding role-based security to the Analysis Service database and let the backend service change its process identity for each call. The second solution would be based on filtering the incoming query requests. For both solutions we have identified their drawbacks including a significant time investment to implement them in a correct way. However since this requirement has not been a hard stated requirement, but only a notification on a possible future requirement it has been decided to push a final decision on this issue to a future date.

At the time of this thesis, AddTrack 4 has still been in an early development phase, we have not had access to actual usage statistics and this has prevented us from performing usage-based optimizations on our aggregation design. We have recommend continuous logging of OLAP queries on a project-by-project basis once AddTrack 4 becomes available to its intended users and periodically perform usage-based aggregation optimization. This should ensure maximum database performance and that the aggregation design mirrors the most recent trends in usage patterns.

9.3 Further recommendations

For performance improvements in transferring a large amount of data we have recommended further studies on whether there are third-party solutions that have efficient data serializers/deserializers capable of providing better data-transmission performance than the default WCF DataContractSerializer (transferring the data in XML-based formats carries significant overhead).

Another possiblity for future work is to consolidate the data from the various backend data sources for the different types of data into a data warehouse for that project. The data warehouse would then contain summarized data at a higher granularity than the existing database which would contain the detailed operational data. Data would be pulled from the operational data stores at fixed well-defined intervals. SSAS would be set to run on top of this warehouse. This would allow us to better control SSAS processing, since it would be easier to synchronize the steps, as well as by setting a longer interval between updates would enable better use of data caching techniques to improve average response times and provide better scaling. In cases where highly detailed data would be required, this should be satisfied directly from the operational stores. Redesigning the system in this way would allow us to construct better measures, more suitable for aggregation.

9.4 Future work

As future work we would like to address some of the issues discussed in Section 9.2, especially the problem with the environment sample data and the problems they cause in our SSAS databases. We would also like to investigate whether there are better performing solutions for data transfer between our services.

Chapter 10

Conclusions

Based on the work performed in this thesis we can conclude that OLAP provides a powerful mechanism for performing coarse-grained analysis and provides to users a good overview over the whole train fleet, which can scale with high data loads. It does however not solve all the problems that AddTrack 4 faces. When it comes to the binary environment data it proves to be a non-optimal solution and shows signs of data explosion. We can say that OLAP is not an end-all, fix-all solution for the problems AddTrack 4 is meant to solve.

For system scalability, the adopted SOA approach does offer a very flexible solution that provides the ability to scale out subsystems (that shows signs of suffering from usage overload) without impacting on other subsystems. The gateway/trusted subsystem model reduces the security infrastructure complexity and centralizes user authorization management, and still provides adequate security.

We have also discussed visualization techniques and proposed a visualization solution which is a compromise of different techniques, since we do not yet have full understanding of how AddTrack 4 users would prefer to state their data queries. This means we would have to provide a solution that allows for free-form, ad hoc queries and observe user behaviour in order to better understand what queries on the data that is meaningful to users of the AddTrack system.

We have also provided Addiva with an operational implementation of an Event data service, a partial implementation of the ASM, and a partial service for user access management, which can be used as reference for further development.

Bibliography

[BAB+07] Axel Buecker, Paul Ashley, Martin Borrett, Ming Lu, Sridhar Muppidi, and Neil Readshaw. *Understanding SOA Security Design and Implementation*. IBM International Technical Support Organization, second edition edition, 2007.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *SIGMOD Rec.*, 26:65–74, March 1997.

[EAA+04] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pål Krogdahl, Min Luo, and Tony Newling. *Patterns: Service- Oriented Architecture and Web Services*. IBM International Technical Support Organization, first edition edition, 2004.

[Gor03] Narasimhaiah Gorla. Features to consider in a data warehousing system. *Commun. ACM*, 46:111–115, November 2003.

[Koc09] H. Koch. *MITRAC TDS Offboard Tool Interface Event data*. Bombardier PPC/TEDT, version 2.2.0.0 edition, 2009. Engineering specification.

[Mar08] Dorina Marghescu. *Evaluating Multidimensional Visualization Techniques in Data Mining Tasks*. PhD thesis, Åbo Akademi University, 2008. TUCS Dissertations 107.

[MFT+08] J.D. Meier, Carlos Farre, Jason Taylor, Prashant Bansode, Steve Gregersen, Madhu Sundararajan, and Rob Boucher. *Improving Web Services Security*. Microsoft Corporation, 2008.

[MK98] Seigo Muto and Masaru Kitsuregawa. Improving main memory utilization for array-based datacube computation. In *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP*, DOLAP '98, pages 28–33, New York, NY, USA, 1998. ACM.

[MPP03] D. Georgakopoulos M. P. Papazoglou. Service-oriented computing. *Communications of the ACM*, Volume 46(Issue 10), October 2003.

[MVSV03] Andreas S. Maniatis, Panos Vassiliadis, Spiros Skiadopoulos, and Yannis Vassiliou. Advanced visualization for olap. In *Proceedings of the 6th ACM international workshop on Data warehousing and OLAP*, DOLAP '03, pages 9–16, New York, NY, USA, 2003. ACM.

[SAM98] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of olap data cubes. In *In Proc. Int. Conf. of Extending Database Technology (EDBT'98*, pages 168–182. Springer-Verlag, 1998.

[TD05] Kesaraporn Techapichetvanich and Amitava Datta. Interactive visualization for olap. In *Computational Science and Its Applications Ũ ICCSA 2005*, volume 3482 of *Lecture Notes in Computer Science*, pages 293–304. Springer Berlin / Heidelberg, 2005.