HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

Spatialytics.org
Open Source Geospatial Business Intelligence

Extending BIRT with geospatial data visualization capabilities by integrating the SOLAPLayers mapping component

Bachelor Thesis

Department of Computer Science
University of Applied Science Rapperswil

Spring Term 2011

Author:	Christoph Süess
Advisor:	Prof. Stefan Keller
Project Partner:	Spatialytics, Québec City, Canada
External Co-Examiner:	Claude Eisenhut
Internal Co-Examiner:	Prof. Hansjörg Huser

1 Abstract

Department	Computer science
Name of student	Christoph Süess
Academic year	Spring semester 2011
Title of the bachelor thesis	Extending BIRT with Geospatial Data Visualization capabilities by integrating the SOLAPLayers mapping component
Examiner	Prof. Stefan Keller
Topic	Reporting tools / OSBI / GeoBI
Project partner	Spatialytics
Institution	Institut für Software HSR

Summary

Today, Business Intelligence (BI) is an important part for every modern company. BI allows analyzing a large amount of data. The overview over the information simplifies to take decisions, which influence the strategies of the company. The different parts of BI are supported by a lot of proprietary and open source applications (OSBI).
Spatialytics from Québec, Canada is a small start-up company, which established in 2009. The main goal of Spatialytics is to provide the use of geo-spatial data for different OSBI-tools. Spatialytics has three software-tools which covering different aspects of the BI-process. One of these solutions is SOLAPLayers. SOLAPLayers is a reporting tool, which displays data from different data sources on an interactive, web based dashboard. The main feature is the possibility to retrieve geo-spatial data and display it on a map component.

This bachelor thesis forced to restructure the existing SOLAPLayers 2.0 version, to provide a more flexible, extendable and dynamical software component, which can be integrated into other considerable reporting tools. The new version allows providing new data-source drivers and output formats in an easy way to the framework. In a second step, a driver for relational databases has been added to the application. This driver was necessary to extend the range of potential users of SOLAPLayers, since not every company owns a data warehouse. The resulting software is not a final version. More data sources and other features will be added to SOLAPLayers before providing the software to the public.
The second goal of this project was to create a map component for the popular open source reporting tool BIRT. BIRT is on of the main projects of the Eclipse foundation. The founder and most active collaborator is the company Actuate. Based on different facts, the decision to integrate SOLAPLayers into BIRT was done. Finally, SOLAPLayers is used as data source and report item of the new BIRT-plugin.

This document contains the analysis and the project documentation of both parts of this bachelor thesis. Additional an excursion on the topic “state of the art of reporting tools” can be found in the appendix of this document. The whole thesis was produced by Christoph Süess at Spatialytics in Québec CA during a three month long internship and supervised by Prof. Stefan Keller at the Hochschule für Technik at Rapperswil CH.

2 Management Summary

In den 1990er Jahren wurde ein Verfahren mit dem Namen Business Intelligence populär. Hinter dem Begriff Business Intelligence - kurz BI - verbirgt sich der Prozess zur systematischen Analyse von Daten, welche meist in elektronischer Form vorhanden sind. Dieses Werkzeug wird von Unternehmen genutzt um sich eine Übersicht über diverse Zahlen zu verschaffen und die Erkenntnisse in die Strategien einfliessen zu lassen. Somit können die Unternehmensziele besser verfolgt und kontrolliert werden. Das BI-Verfahren kann grob in drei Phasen unterteilt werden. In der ersten Phase werden die Daten zusammengetragen. Diese Daten werden normalerweise in einem sogenannten Data-Warehouse abgelegt. In der zweiten Phase wird ein Zusammenhang zwischen den Informationen gebildet, so dass aus den Daten Schlüsse gezogen werden können. In der letzten Phase, werden die Fakten dem Unternehmen präsentiert und dafür gesorgt, dass die Erkenntnisse genutzt werden können.

Da es sich meistens um grosse Datenbestände handelt, ist es nicht verwunderlich, dass dieser Prozess durch diverse IT-Systeme unterstützt wird. So wurden im Verlaufe der Jahre Software-Komponenten für alle drei Phasen des BI-Verfahrens entwickelt. Inzwischen gibt es mehrere Unternehmen, die etablierte Applikationen für den BI-Prozess vermarkten. Die wohl namhaftesten Produkte sind diejenigen des weltweit tätigen Unternehmens SAP.

Figure 1 - Logo von populären OSBI

In den letzten Jahren wurde das Vertrauen in Open Source Softwarelösungen durch effiziente und erfolgreiche Produkte gestärkt. Viele Anwender erkannten die Stärken von Open Source und legten das weiterverbreitete Misstrauen gegenüber der freien Software nieder. Dieser Trend setze sich auch in der Businesswelt fort. Neben den proprietären BI-Tools entstand freie Software, welche mit OSBI (Open Source Business Intelligence) betitelt wurde. Manche Produkte unterstützen den gesamten BI-Prozess und andere konzentrieren sich auf bestimmte Teile, wie zum Beispiel dem Zusammentragen, der Zurverfügungstellung oder dem Auswerten der Informationen.

Figure 2 - Logo des Unternehmes Spatialytics

2009 entstand in Québec City in Kanada das Start-Up Unternehmen Spatialytics. Spatialytics ist nach wie vor ein sehr frisches Unternehmen, dass noch diverse Entscheidungen treffen muss. Die Firma ging aus einigen Projekten aus der Geoinformatik, welche an der Université Laval Fuss fassten, hervor. Die treibende Kraft hinter dem Projekt war Prof. Thierry Badard. Spatialytics kannte den BI-Markt und interessierte sich dafür, den Aspekt der räumlichen Information der Daten in einem Data-Warehouse für spezifischere Auswertungen zu

nutzen. Ein populäres Zitat aus der Geowelt besagt, dass achtzig Prozent aller gespeicherten Businessdaten einen räumlichen Bezug haben. Bis zum heutigen Tag wird dieser Aspekt jedoch von vielen BI- und OSBI-Systemen zu wenig genutzt. Der räumliche Gesichtspunkt ist aber wie der Zeitliche in der Wirtschaft sehr wichtig.
Spatialytics verfolgt im Moment drei Projekte. Mit GeoKettle bietet das Unternehmen ein ETL-Tool (Extract, Transform, Load) an, welches neben den herkömmlichen Datenformaten auch mit diversen geometrischen Datenformaten umgehen kann. Mondrian ist ein OLAP-Server, ein System das performante und komplexe Zugriffe auf ein Data-Warehouse ermöglicht. Spatialytics erweiterte dieses quelloffene Projekt mit dem räumlichen Aspekt, so dass die resultierenden Daten auch geometrische Informationen enthalten können. Daraus entstand das Produkt GeoMondrian.
Das dritte wichtige und im Zusammenhang mit der Bachelorarbeit interessanteste Projekt von Spatialytics, widmet sich dem Auswerten der Daten. Mit SOLAPLayers können Daten aus verschiedenen Datenquellen abgefragt werden und in einem interaktiven Report dargestellt werden. Natürlich liegt auch hier wieder das Hauptmerkmal auf der Präsentation der geografischen Daten. Das Herz von SOLAPLayers bildet die interaktive Karte, welche komplexe geografische Zusammenhänge abbilden kann. In Zukunft will Spatialytics diese geometrischen Features von SOLAPLayers auch für namhafte Open Source Reporting Tools anbieten. Dazu gehören Produkte wie Pentaho Reporting, Jasper Reports und BIRT.

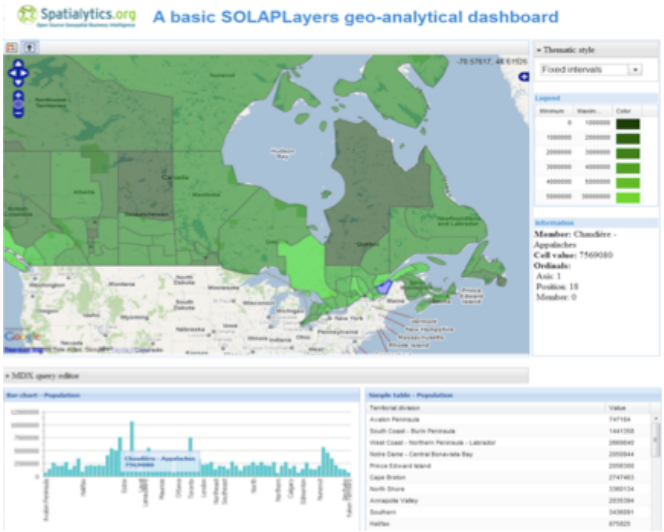


Figure 3 - SOLAPLayers Demo Dashboard

Diese Bachelorarbeit forcierte die Integration einer interaktiven Karte in Berichte, welche mit Hilfe von BIRT erstellt werden können. Es war nicht gegeben, aber naheliegend das man hierfür die existierende SOLAPLayers-Lösung nutzte. Das Ziel war ein Prototyp zu erstellen, welcher vor allem für Demos verwendet und gegebenenfalls später für den Ernstfall erweitert werden kann.

Die erste Version von SOLAPLayers war noch nicht bereit um in eine bestehende Software eingegliedert zu werden. Darum begann man an einer neuen Version zu arbeiten. Neben dem Hinzufügen von neuen Features, stand vor allem das Umstrukturieren von SOLAPLayers im Mittelpunkt. Die neue Version sollte flexibler sein, so dass die Applikation einerseits besser erweitert und andererseits einfach in Drittapplikationen eingegliedert werden kann. Die Umstrukturierung ging zu Gunsten dieser Bachelorarbeit. Das Resultat erlaubt nun neue Datenquellen und Ausgabeformate flexible hinzuzufügen. Der Aufwand beschränkt sich dabei auf das Entwickeln von neuen Treibern, die Integration ist mit sehr geringem Aufwand verbunden. Neben dieser Umstrukturierung wurde SOLAPLayers durch einen neuen Datenquellen-Treiber ergänzt. Ab sofort können auch relationale Datenbanken mit SOLAPLayers benutzt werden. Zuvor wurden nur SOLAP-Quellen unterstützt. Die Unterstützung von relationalen Datenbanken war aus mehreren Gründen überfällig. Entscheiden war, dass

nach wie vor nicht alle Unternehmen mit für die Analyse optimierten Data-Warehouse arbeiten. Öfters sind die Daten in einfacheren relationalen Datenbanken vorhanden. Auch wenn die Verwendung solcher Datenquellen, auf Grund von Performance-Einbussen, nicht empfehlenswert ist, sollte ein Reporting Tool deren Anbindung unterstützen. Nach dem Fertigstellen dieser Zwischenversion von SOLAPLayers wurde die Entwicklung an SOLAPLayers ausserhalb der Bachelorarbeit weitergeführt.

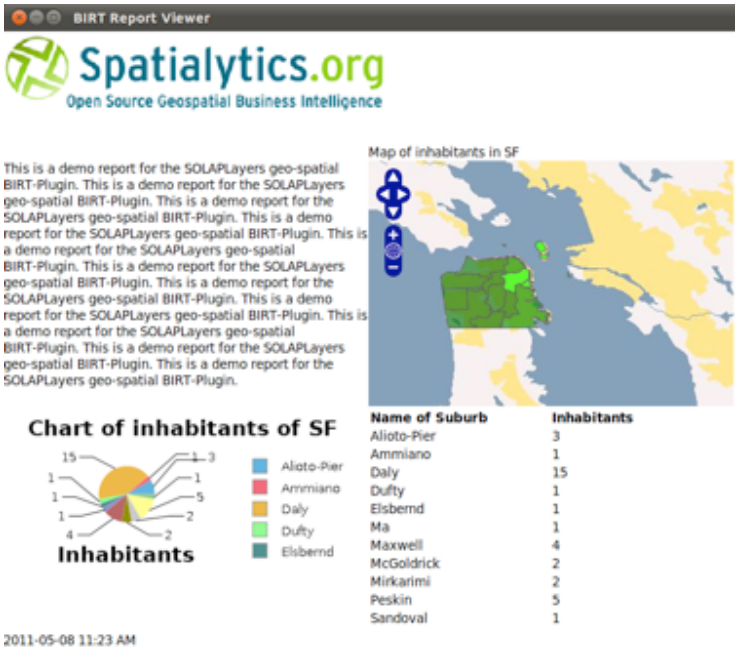


Figure 4 - BIRT Report mit SOLAPLayers-Karte

Die Bachelorarbeit widmete sich nun der BIRT Integration von SOLAPLayers. Die Möglichkeit der Integration wurde zuerst analysiert und diverse Szenarien besprochen. Die Schlussfolgerung zeigte, dass es Sinn macht SOLAPLayers als Datenquelle sowie zur Präsentation der Daten zu verwenden. Die Integration stellte sich als schwieriger als geplant heraus. Trotzdem konnte das geplante Szenario umgesetzt werden.

Neben der resultierenden Software wurde ein Bericht zum Thema „State of the Art of Reporting Tools“ erstellt, welcher die aktuelle Situation von Tools zur Erstellung von Berichten aufzeigt.

Die Umstände machten das Projekt nicht einfach. Da das Projekt in einem sehr jungen Unternehmen, welches sich schnell entwickelt, durchgeführt wurde, konnten viele Abhängigkeiten nicht vermieden werden. Des Weiteren existierte für SOLAPLayers keine Dokumentation und auch das sehr etablierte Tool BIRT ist leider sehr schlecht dokumentiert, was die Erweiterung des Reporting Tools betrifft. Die Kommunikation an sich war jedoch sehr gut, obwohl eine begrenzte Sprachbarriere vorhanden war. Schwierigkeiten brachten aber auch die verschiedenen Interessengruppen mit sich. Einerseits sollte meine Arbeit bei Spatialytics das Unternehmen weiterbringen und andererseits war das Bachelorprojekt gewissen Richtlinien unterstellt. Diese Koordination erforderte viel Durchhaltewillen und Aufwand.

3 Scope

Extending BIRT with Geospatial Data Visualization capabilities by integrating the SOLAPLayers mapping component

Bachelorarbeit von Christoph Süess im Ausland
Abteilung Informatik, Frühjahrssemester 2011

Ausgangslage

Reporting-Programme (Berichterstellungsprogramme) automatisieren die Analyse und Aufbereitung von Informationen. Damit lassen sich Berichte (Reports) in Form von Tabellen und Businessgrafiken (Charts) erstellen und in den gängigen Formaten (z.B. HTML, PDF, XLS und CSV) ausgeben. Die Berichte können nach der einmaligen Konfiguration immer wieder erzeugt werden.

“Business Intelligence Reporting Tools” (BIRT http://www.eclipse.org/birt/) ist ein solches Reporting-Programm. Es basiert auf Eclipse und ist Java Open Source (Eclipse Public License).

Um einen Report zu erstellen, muss zunächst eine Verbindung zu einer Datenquelle, z.B. zu einem Data Warehouse, hergestellt werden. Dann kann der Report mit Hilfe des "BIRT Report Designers" interaktiv definiert werden. Für flexible Anfragen gibt es auch eine Scriptingsprache. Das Resultat ist eine XML-Report-Design-Datei. Der eigentliche Report wird mit der "BIRT Report Engine" (Laufzeitumgebung) erzeugt. Diese Software lässt sich auch gut in Webapplikationen (Java/J2EE) integrieren.

Wird einer Information ein Raumbezug zugeordnet (z.B. eine Adresse oder Koordinate), so spricht man von Geodaten. Geodaten sind komplex, aufwändig in der Erfassung und Rechenintensiv in der räumlichen Auswertung (z.B. Umkreissuche). Sie werden typischerweise in bzw. Geo-Datenbanken (-Erweiterungen) gespeichert und mittels Geo-Informationssystemen (GIS) verwaltet.

Für die räumlich erweiterte Prozessierung von Informationen aus Data Warehouses (sog. „Spatial Online Analytical Processing“, Spatial OLAP, bzw. SOLAP) und v.a. deren Darstellung in einer Onlinekarte, gibt es die JavaScript-Bibliothek „SOLAPLayers“. SOLAPLayers basiert auf den Bibliotheken GeoExt/OpenLayers (vgl. http://www.spatialytics.org/?page_id=10). Im Bereich SOLAP hat sich die Univ. Laval, Québéc einen Namen gemacht.

Aufgabenstellung

Zurzeit werden von BIRT-Reports nur Businessdiagramme unterstützt. Was fehlt, ist die Ausgabe als Onlinekarte. Die Aufgabe der Arbeit besteht daher aus folgenden Teilen:

to restructure the SOLAPLayers application on the server side, to provide a more flexible application which allows to add new data sources and output formats in an easy way

to analyse how it will be possible to add a map component in the toolbox offered by BIRT and which features this component will have to provide (access to some spatial data sources, thematic mapping capabilities, etc.)
to analyse how it will be possible to add a map component in the toolbox offered by BIRT and which features this component will have to provide
to develop the code in order to effectively carry out this integration.
To write an excurse on the topic: „The state of the art of OSBI-tools“

Beteiligte

Diplomand:	Christoph Süess (csueess@hsr.ch)
Projektpartner:	Prof. Thierry Badard und Mitarbeiter des Centre for Research in Geomatics (CRG), Département des Sciences Géomatiques, Université Laval, Québec, Qc, G1V 0A6, Canada.
Betreuung HSR, Verantwortlicher Dozent:	Prof. Stefan Keller, IFS-HSR (sfkeller@hsr.ch)
Gegenleser:	Prof. Hansjörg Huser
Experte:	Claude Eisenhut, Eisenhut Informatik Burgdorf (ceisenhut@eisenhutinformatik.ch)

Projektabwicklung

Termine

Beginn der Arbeit:	28.03.2011
Abgabetermin:	17.06.2011 (12.00 Uhr).

Weitere Termine: Siehe https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html (intern)

Arbeitsaufwand

Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von 360 Stunden.

Hinweise für die Gliederung und Abwicklung des Projektes

Gliedern Sie Ihre Arbeit in 4 bis 5 Teilschritte. Schliessen Sie jeden Teilschritt mit einem Meilenstein ab. Definieren Sie für jeden Meilenstein, welche Resultate dann vorliegen müssen!

Folgende Teilschritte bzw. Meilensteine sollten in der Planung vorgesehen werden:

- Schritt 1: Projektauftrag inkl. Projektplan (mit Meilensteinen),
- Meilenstein 1: Review des Projektauftrages abgeschlossen. Projektauftrag von Auftraggeber und Dozent genehmigt
  Termin: ca. zwei Wochen nach Beginn der Arbeit
- Letzter Meilenstein: Systemtests abgeschlossen
  Termin: ca. eine Woche vor Abgabe

- Die Software ist in einem agilen, d.h. iterativen und inkrementellen Prozess zu entwickeln: Man plane möglichst früh einen ersten lauffähigen Prototypen mit den wichtigsten und kritischsten Kernfunktionen. In den folgenden Phasen kann dann schrittweise ausgebaut und getestet werden.
- Falls in der Arbeit neue oder unbekannte Technologien eingesetzt werden, sollte man parallel zum Erarbeiten des Projektauftrages mit dem Technologiestudium beginnen.
- Es sind Unit-Tests einzusetzen. Software und Dokumente werden auf einem Repository verwaltet (z.B. SVN, git). Ev. ist ein Build Server hilfreich (z.B. Cruise Control).
- Man halte sich im Übrigen an die Vorgaben aus dem Modul SE-Projekt.
- Projektadministration:
- Es ist ein Projekttagebuch zu führen aus dem ersichtlich wird, welche Arbeiten durchgeführt wurden (inkl. ungefährem Zeitaufwand). Diese Angaben sollten ggf. eine individuelle Beurteilung ermöglichen.
- Die Arbeiten sind laufend zu dokumentieren. Man lege die Projektdokumentation mit der aktuellen Planung und den Beschreibungen der Arbeitsresultate elektronisch in einem Projektordner ab. Dieser Projektordner sollte jederzeit einsehbar sein (z.B svn-Server oder File-Share).

Fortschrittsbesprechung:

- Regelmässig findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung statt (Arbeiten im Ausland: Wochenbericht).
- Teilnehmer sind Dozent und Studenten, bei Bedarf auch Vertreter der Auftraggeber
- Termin gem. Absprache (Arbeiten im Ausland z.B. Montags).
- Falls notwendig, können weitere Besprechungen / Diskussionen einberufen werden.
- Sie erstellen zu jeder Besprechung ein Kurzprotokoll, welches Sie spätestens 2-3 Tage nach der Sitzung per E-Mail an den Betreuer senden (Arbeiten im Ausland: Eintrag im Wiki o.ä.).

Inhalt der Dokumentation

- Die fertige Arbeit muss folgende Inhalte haben:
    1. Abstract, Management Summary, Aufgabenstellung
    2. Technischer Bericht
    3. Dokumente des Projektdokumentation
    4. Anhänge (Literaturverzeichnis, CD-Inhalt)
- Mind. die Dokumente der Punkte 1, 2 sowie Installation und Code müssen in Englisch sein.
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung für Broschüre, Poster) gemäss www.hsr.ch und gemäss Absprache mit dem Betreuer.

Form der Dokumentation

- Bericht (Struktur gemäss Beschreibung) gebunden (2 Exemplare) und in Ordner (1 Exemplar „kopierfähig“ in losen, gelochten Blättern).
- Alle Dokumente und Quellen der erstellten Software auf CD; CD's sauber angeschrieben (3 Ex.).

Bewertungsschema

Als Bewertungsschema gilt das in www.hsr.ch erwähnte, d.h. die folgenden Aspekte werden bewertet:

- Projektorganisation (Gewicht 1/6)
- Bericht (Gewicht 1/6): Inhalt, Gliederung, Sprache
- Inhalt (Gewicht ½): 1. Vorstudie, Anforderungsanalyse und Domainanalyse; 2. Entwurf (Systemarchitektur, Beschreibung des Entwurfs, Entwurf Benutzerschnittstelle); 3. Realisierung und Test
- Mündliche Prüfung (Gewicht 1/6)

Es gelten ansonsten die üblichen Regelungen zum Ablauf und zur Bewertung der BA-Arbeit des Studiengangs Informatik der HSR.

Rapperswil, 27. Oktober 2010, S. Keller

4 Table of contents

1 Abstract 2
2 Management Summary 3
3 Scope 6
4 Table of contents 10
5 Overview 12
6 SOLAPLayers 2.0 Extended 13
6.1 Overview 13
6.2 Technical report 13
6.2.1 Introduction 13
Usage of SOLAPLayers 13
SOLAPLayers versions 13
Guidelines 14
Course of action 14
6.2.2 State of the art of reporting tools 14
6.2.3 Results and evaluation 15
Achievement of objectives 15
Future Improvements 15
6.3 Project documentation 15
6.3.1 Requirements specification 15
6.3.2 Analysis - Result set to OLAP-cube transformation 15
Why using SQL for data analyzing? 15
Why parsing SQL-results to OLAP-cubes? 15
What is an OLAP-cube? 16
How does the transformation work? 16
6.3.3 Architecture 17
Architecture of SOLAPLayers 1.0 17
Architecture of SOLAPLayers 2.0 Extended 18
Activity diagram 20
Package diagram / layer overview 21
Class diagram 22
Technologies 23
Client application 24
6.3.4 Implementation 25
System test 25
Code quality 29
6.3.5 Future improvements 30
Chores 30
Extensions 30
6.3.6 Developers guide 31
Installation 31
Add a new connection type 32
Add a new output format 33
Register new output format 34
Add a new data source 34
Using MDX-queries 35
Using SQL-queries 36

Creating a dashboard 40
7 Geoextensions2Birt 41
7.1 Overview 41
7.1.1 Technical report 41
Introduction 41
State of the Art 41
7.1.2 Project documentation 41
Requirements specification 41
Analysis 42
Architecture 47
Implementation 48
System test 48
Code quality 51
Future Improvements 51
User Guide 52
Developers guide 59
8 Project management 60
9 Project review 60
9.1 Acknowledgment 60
9.2 Field report 60
9.3 Lessons learned 61
10 Further documentation 61
11 Appendices 62
APPENDIX A - Content of the CD 62
APPENDIX B - Glossary 62
APPENDIX C - References 64
APPENDIX D - Table of figures 65
APPENDIX E – Project management 67
APPENDIX F - License agreement 81
APPENDIX G - Agreement of the author 82
APPENDIX H - Instructions: Creating a dashboard using SOLAPLayers 82
APPENDIX I – Customized SCRUM-process for bachelor thesis 84
APPENDIX J – Interim report 86
APPENDIX K - Result presentations 88
APPENDIX L – State of the art of reporting tools 89

5 Overview

This document is structured in two different main parts. The first part documents the SOLAPLayers improvement, which was done at the beginning of the bachelor project. The second part describes the integration of SOLAPLayers 2.0 Extended into the BIRT-plugin. Both parts contain a technical report and project documentation. Information about the project progress and approach can be found at the end of this document. Additional documents are attached to this document in the appendices.
Note: SOLAPLayers 2.0 Extended is not an official name of a SOLAPLayers version. SOLAPLayers 2.0 Extended refers to the state of SOLAPLayers, which was reached at the end of the bachelor thesis-project.

6 SOLAPLayers 2.0 Extended

6.1 Overview

Main use of this chapter is to document SOLAPLayers 2.0 Extended. For better Understanding some parts of the SOLAPLayers base ideas will complete this documentation. The document is structured by two main topics, the technical report and the project documentation.

6.2 Technical report

6.2.1 Introduction

Usage of SOLAPLayers

SOLAPLayers is a product developed by the University of Laval. Spatialytics extends SOLAPLayers since 2009.
SOLAPLayers is a Java-reporting-framework, which is used to build interactive dashboards with mapping capabilities. SOLAPLayers retrieves data from different data sources and displays the information on dashboards. The core function of SOLAPLayers is to create interactive maps.
On the on hand, SOLAPLayers can be used as a web-based reporting tool. Dashboards can be defined using simple HTML-tags. On the other hand, SOLAPLayers can be integrated in existing reporting tools to extend them with mapping capabilities.

SOLAPLayers versions

SOLAPLayers 1.0

Figure 5 - SOLAPLayers logo

The initial version of SOLAPLayers provided the possibility of querying one single GeoMondrian-data sources. The whole project was structured in four classes. The connection string was configured in the web.xml-file. GeoJSON and OLAPJson existed as data transport format. The whole source code was high coupled and did not provide a lot of reusability. SOLAPLayers 1.0 used Dojo and OpenLayers for the representation of the map on the client side. The SOLAPLayers 1.0 source code is available for download at http://sourceforge.net/projects/spatialytics/. SOLAPlayers 1.0 is more a proof of concept than an actual version.

SOLAPLayers 2.0

SOLAPLayers 2.0 is a revision of the concept of SOLAPLayers 1.0. The client and the server part were reengineered. On the client side the Dojo- and OpenLayers-frameworks were replaced with ExtJS and GeoExt. SOLAPLayers 2.0 supports many more components for data representation (crosstab, bar chart, bar chart, etc.)

SOLAPLayers 2.0 Extended

SOLAPLayers 2.0 Extended is no official release of SOLAPLayers, moreover it references to the state, which was reached after the bachelor project. SOLAPLayers 2.0 Extended has two main goals:

- Build a flexible application, which simplifies to add new data sources and output builders. For reaching this goal, the whole source code will be reviewed and refactored.
- Add a SQL-data source to SOLAPLayers for querying relational databases.

GeoBIExt 1.0

GeoBiExt 1.0 will be the renamed product of the final SOLAPLayers 2.0 release. GeoBiExt 1.0 will be presented at the FOSS4G 2011 in Denver, Colorado.

Guidelines

SOLAPLayers was an existing project based on some technology- and architecture-desicions and principles they were defined before the bachelor project started. This bachelor thesis did not evaluate this decisions and continued on the given base. On the one hand, this was not a desire of Spatialytics, on the other hand, the evaluation would be too time consuming for this project. The following key-facts affected the course of the project:

- SOLAPLayers retrieves data from any data source and sends it to the client in the OLAPJson-format. OLAPJson is a undocumented format invented by Spatialytics. OLAPJson represents a OLAPCube in a JSON-format.
- The client-side of SOLAPLayers was never touched by the student during this project. The client-side parses the OLAPJson-file and represents the data using html, css and javascript. As long as the OLAPJson-file is valid, the possible failures on the GUI are problems of the client-side. A small overview of the client-side can be found in the sub-chapter “client application”.
- The OLAPCube transformation (OLAP4J to OLAPJson) was already a part of the 1.0 version During the project the source code was splited up in different parts to make the application more flexible. Some improvements on the source code and the algorithm were made as well. The evaluation of this transformation was not a part of this thesis and was never reviewed in detail.
- All the key technology decisions were done in advanced:
    - Java as the main programming language
    - OLAP4J-Library to handle three dimensional mdx-result-cubes
    - The 2D-spatial-java-library JTS Topology Suite, which is used to interpret geometry-formats
    - JSON as base data transport format
    - All the technologies used by the client side of SOLAPLayers (Javascript, HTML, CSS, OpenLayers and the EXT JS-framework)

Course of action

This project was developed in a customized SCRUM-process. For further information see “APPENDIX I – Customized SCRUM-Process for bachelor thesis”.

6.2.2 State of the art of reporting tools

For information about the state of the art of reporting tools, consider the document “State of the art of reporting tools” which is attached to this document. For the state of the art of SOLAPLAyers see the chapter “SOLAPLayers 2.0“.

6.2.3 Results and evaluation

Achievement of objectives

The goal of this project could be reached completely. Spatialytics was satisfied by the solution. The version was used for extending SOLAPLayers right after finishing SOLAPLayers 2.0 Extended.

Future Improvements

SOLAPLayers 2.0 Extended is not a major release of SOLAPLayers. It is not yet defined which features GeoBIExt 1.0 will provide. Some recommendations are available at the chapter "Future Improvements" of the project documentation.

6.3 Project documentation

6.3.1 Requirements specification

The requirement specification was done by Spatialytics in an agile way. The SCRUM-Task board can be found in the appendix (see "APPENDIX E – project management").

6.3.2 Analysis - Result set to OLAP-cube transformation

A big part of the analysis is already done by Spatialytics. The used technologies are given and the application will be developed based on the version 2.0 of SOLAPLayers. The following chapters document the analysis of the solutions of SOLAPLayers 2.0 Extended.

The following text shows the way in which SOLAPLayers 2.0 Extended transforms the result set of a SQL-query into an OLAP-cube representation.

Why using SQL for data analyzing?

Why accessing simple relational databases with SQL-queries instead of using MDX? OLAP-data sources are much faster for the analysis of a large amount of data, then accessing databases using SQL. Anyway, many enterprises still work with simple relational databases, even for analyzing issues. That is why SOLAPLayers has to support this kind of data sources as well.

Why parsing SQL-results to OLAP-cubes?

The data format is based on a data cube on all logical layers of SOLAPLayers. The common data format, which is processed by the output builders, is based on an OLAP-cube. The OLAPJson-format, which transports the result from SOLAPLayers to the calling application, represents a cube as well. This was given by the project guidelines.

What is an OLAP-cube?

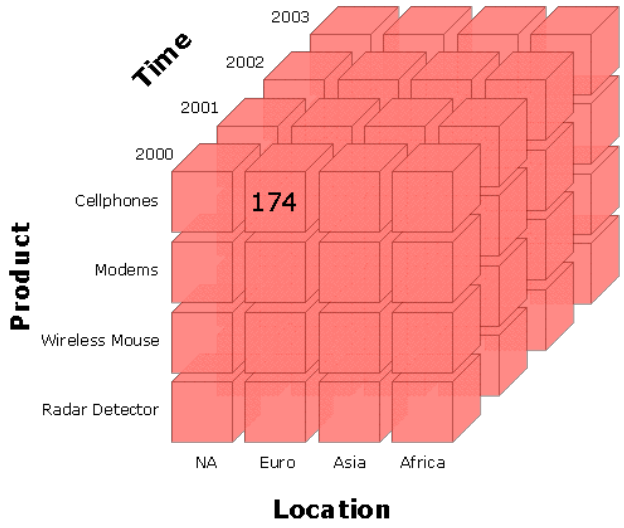


Figure 6 - Visualization of an OLAP-cube

OLAP-cubes result from MDX-queries and can be imagine as three dimensional result sets respectively as data cubes (more than three dimensions are possible). In the sample cube above "Product", "Time" and "Location" are dimensions and "174" is a specific measure. OLAP-cubes are often represented as cross-tables in reports or dashboards. Further information can be found on http://en.wikipedia.org/wiki/OLAP_cube.

How does the transformation work?

SQL is not able to receive an OLAP-Cube. That is the reason why it is necessary to parse the two-dimensional result set to an OLAP-cube representation. There are some steps necessary to reach a good result:

1. Requirements for the SQL-query
   a. **All the measures and dimensions have to be a part of the select-statement**
   b. **It is necessary to group the data by the dimensions**
   Basically, every row of the result set will contain two important pieces of information. The first part of the information contains the coordinates. The coordinates describe the position of the measure/s. The second part of the information is the measure, respectively a bunch of measures.
   The following example shows the data column, which describes the value in the cube above. Product, Location and Time are the dimensions, which are present in the group by-clause of the sql-query. "174" is the measure. With this information it is possible to locate all given measures in the cube. If a specific coordinate is not available in the result set, it means, that the value of this measure is zero.

Product	Location	Time	Value
Cellphones	Euro	2000	174
...	...	...	...

   c. **An aggregate-function has to be specified for the measures**
   Because we always use the group-by-clause it is necessary to define aggregate-functions for all the measures.

For example: The measure in the cube above could be the average price of a cellphone at a continent in a year. In the query we calculate the average price of all the countries of one continent using the avg-function.

Product	Location	Time	Country	Value
Cellphones	Euro	2000	Spain	171
Cellphones	Euro	2000	Germany	177
				174

2. **Additional meta data**
   To extract the information correctly additional meta data is necessary. The type has to be defined for each row of a result set:
   a. **Spatial dimension**
      Which row represents the name of the spatial dimension?
   b. **Geometry data**
      Which row represents the geospatial data (usually a polygon or a multi-polygon) of the spatial dimension?
   c. **Geometry point representation**
      Which row contains a point representation of the spatial dimensions in order to display data additionally as a simple marker on a map?
   d. **Dimensions**
      Which rows contain the other dimensions?
   e. **Measures**
      Which rows represent the measures?

   For guidelines and further information have a look at the chapter “Using SQL-Queries“ in the developers guide.

Limitations

The cube-transformation is still limited. Meta data are not a part of the resulting cube at the moment. Data can be related to just one specific spatial dimension.

6.3.3 Architecture

Architecture of SOLAPLayers 1.0

SOLAPLayers 2.0 has a very basic architecture. The whole process of retrieving and transforming data is done in one shot. SOLAPLayers does not provide any hooks to extend the software. It is an in four class structured procedural application, which is developed only for the reason of sending mdx-queries and receiving OLAPJson. SOLAPLayers was developed to access over a webservlet.

Architecture of SOLAPLayers 2.0 Extended

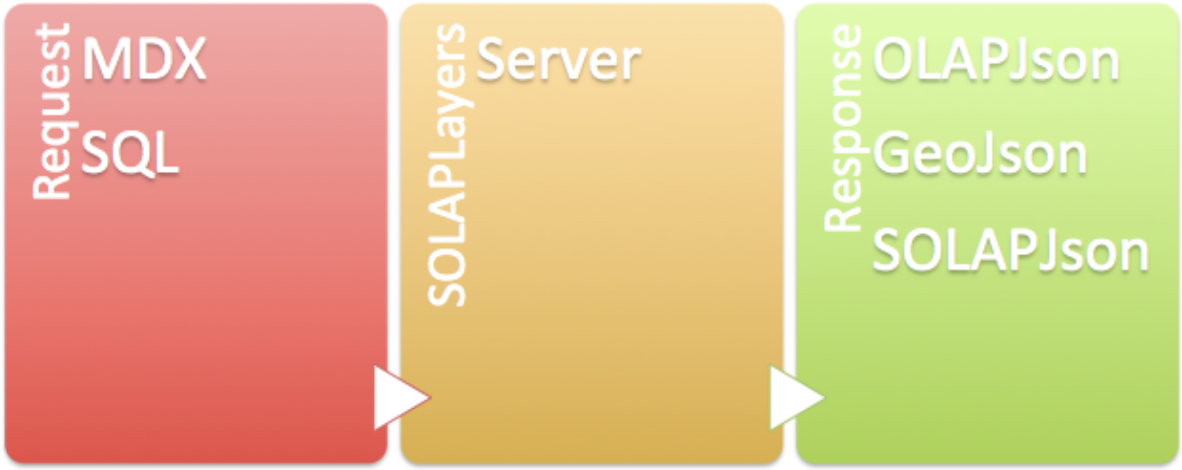


Figure 7 - Overview SOLAPLayers 2.0 Extended

The new architecture of SOLAPLayers uses the advantages of objectoriented programming. The different steps of the workflow were splitted into flexible parts. This enhances the extendability of SOLAPLayers. In use of configurations and parameters the prozess can be customized easely without touching the core source code. The new architecture brings the following advantages:

- The source code is logically structured in different classes and packages. This makes the source clear and understandable in general.
- Specific data sources can be configured using an xml-file. No further changes in the source code are necessary.
- New connection drivers (e.g. flatfiles, webservices, etc.) can be added easily. Interfaces and abstract classes help to develop and integrate the new connection type. No changes at the core are necessary. Defining the path to the new driver in the data source config is all what has to be done. SOLAPLayers uses reflection to load and construct the drivers.
- Scaffoldings for new output formats exist. After implementing an new output builder just very little changes on the core source code has to be done. In future versions this should not be necessary at all (see chapter “Future improvements”).
- SOLAPLayers 2.0 Extended provides access points for webapplications as well as for systems they using SOLAPLayers as a library.

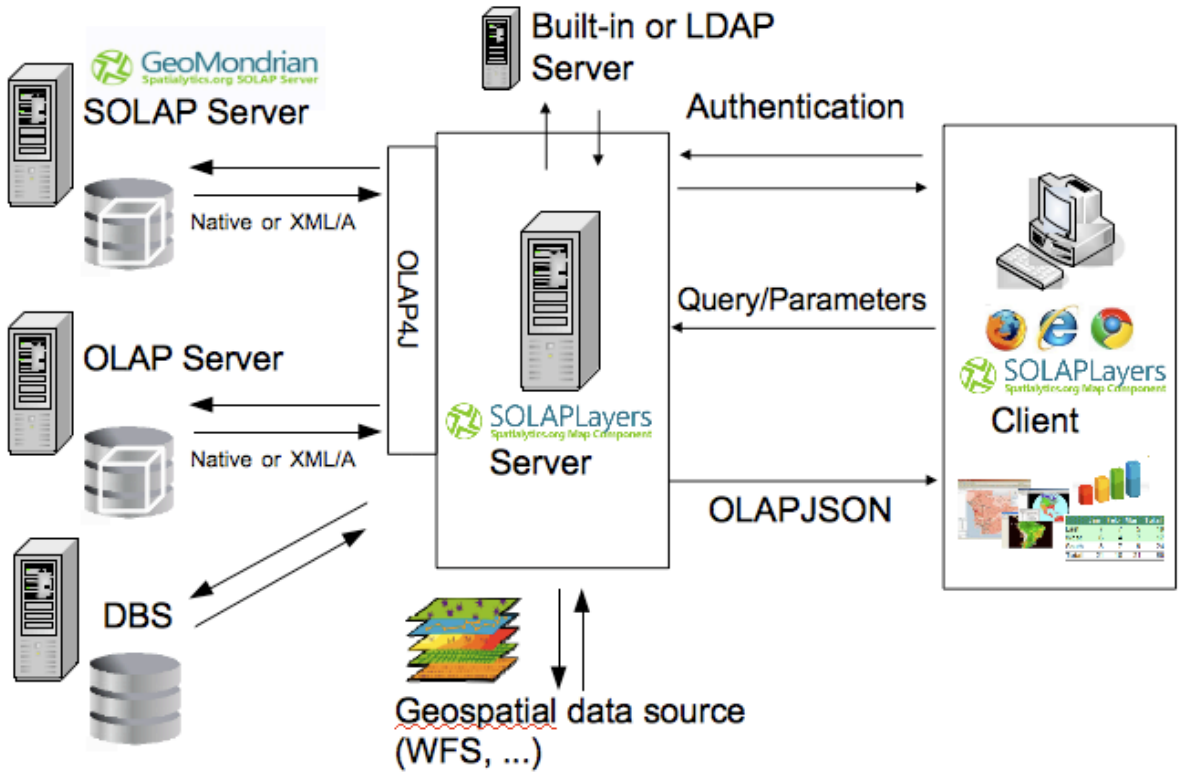


Figure 8 - Overview vision SOLAPLayers 2.0 Extended

Activity diagram

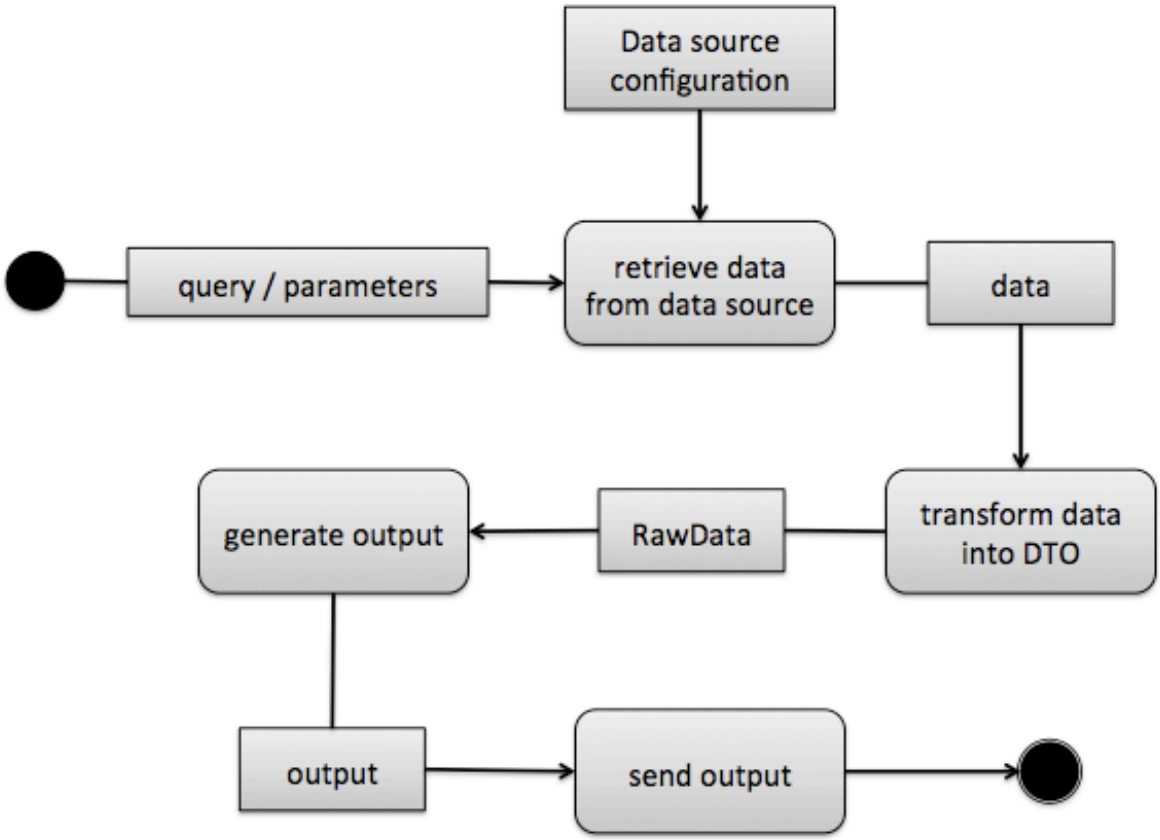


Figure 9 - Activity diagram

The activity diagram shows the workflow of SOLAPLayers. The following steps were performed:

1. The Sever retrieves the query and parameters. The source of this information can be an input of an user or the data of a system, which uses SOLAPlayers as a part of the application. In the same step, SOLAPLayers reads the data source configuration, which specifiy the connection for the data source (e. g. db-url, password, user, etc.). With the query, the additional paramters and the open connection the data from the data source can be received.
2. A comon data format is usefull to build outputs for each data source. That is why the received data structure gets transformed into a comon format. This comon format is mainly a OLAP-cube-representation plus some additionaly information (e.g. the executed query).
3. The comon data structure allows to create different output formats for every kind of data source. The output is not yet sended to the client, this makes it possible to reuse an output builder for another output builder. In this way really complex output builders can be developed.
4. In the last step, the output is sended back to the client or the calling application. Depending on the calling system, some information have to be added (e.g. HTTP-header informations).

All the activitys from above are flexible parts of SOLAPLayers. For new data sources or output formats this actions have to be implemented and contributed to the framework.

Package diagram / layer overview

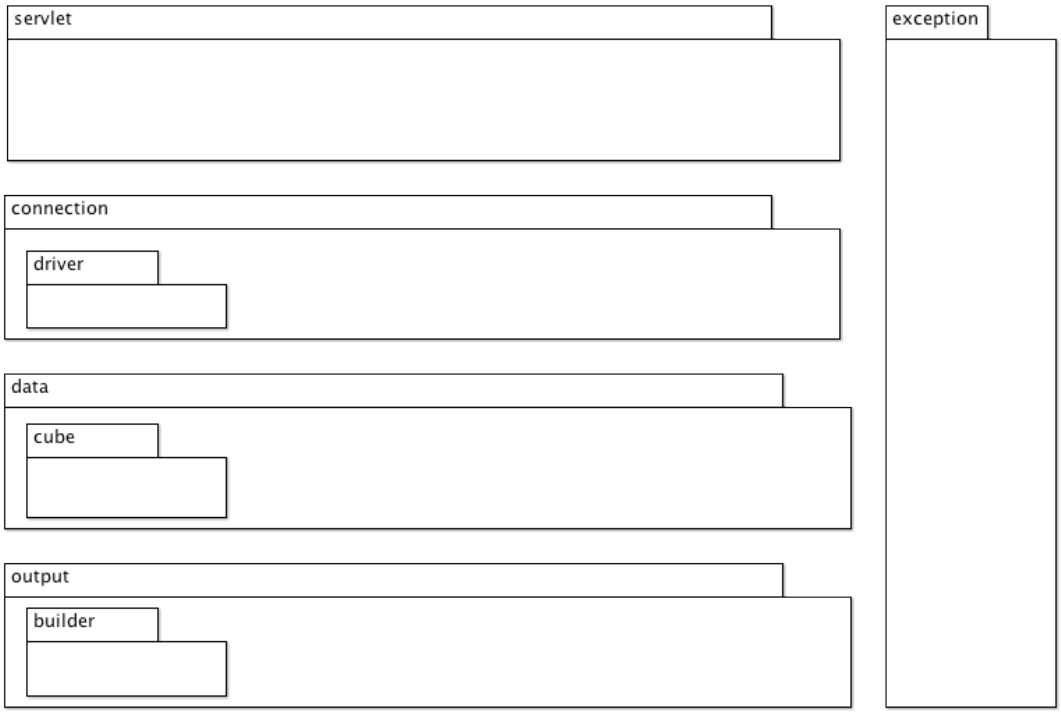


Figure 10 - Package diagram

- The servlet package contains all the classes, which are called by the dashboard or the third-party software.
- The connection package contains all components, which are responsible to create and use a connection to different data sources. The driver package, which is a part of the connection package, contains specific drivers for different data sources, which are structured in different classes. This helps to hold the overview over the different drivers.
- The data package contains classes, which work as data transport objects. Usually these classes are plain data classes without a lot of logic. They work as data interface between the data source and the output builder.
- The output package contains utility classes. They construct the output builder and help to create outputs. Concrete output builders for a specific format are located in the builder package inside of the output package
- The exception package holds some SOLAPLayers specific exceptions. It is used by all of the packages. These exceptions allow a more specific exception handling.

Class diagram

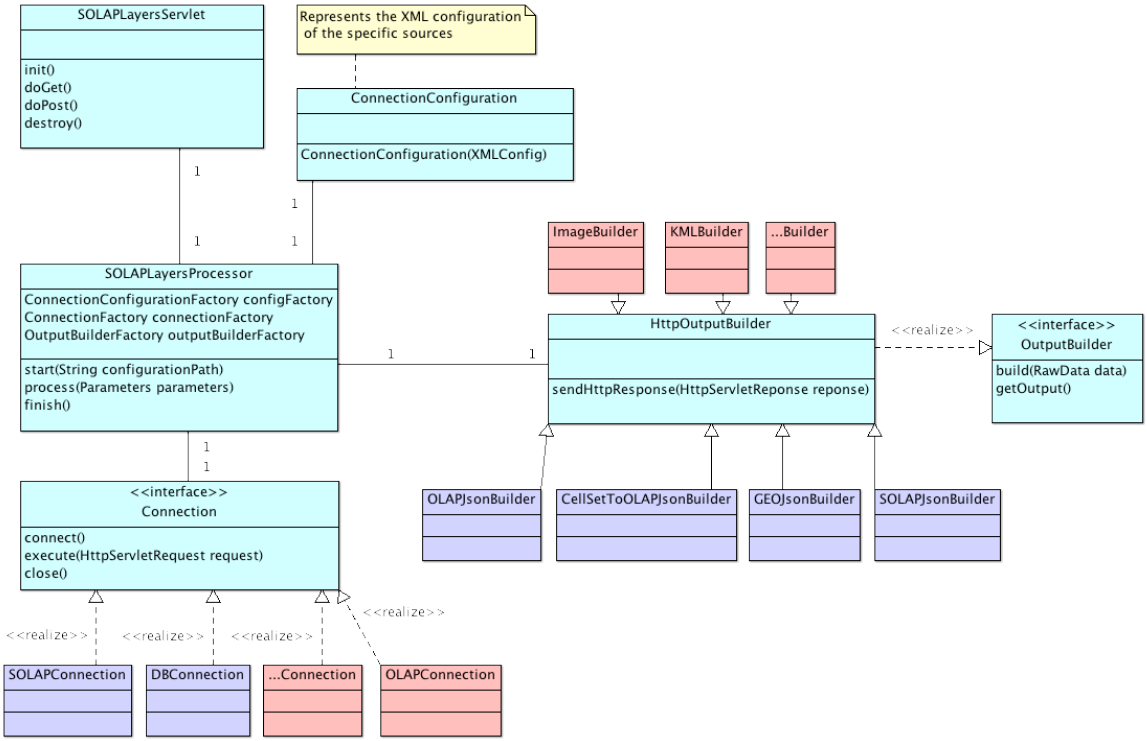


Figure 11 - Class diagram

The class diagram shows the most important classes in the SOLAPLayers-project. Some classes are not displayed for better understanding and overview. For more information have a look at the javadoc.
The light blue colored classes are components of the framework. They regulate the workflow and call the defined hook classes. The dark blue classes are flexible components, which are attached to the hooks of the framework. The red classes show not yet developed classes, they are just displayed to show the possibilities of SOLAPLayers.

SOLAPLayersProcessor

The main class of SOLAPLayers. The SOLAPLayersProcessor starts the data retrieving and executes the data transformations.

SOLAPLayersServlet

SOLAPLayersServlet is the entry point, if SOLAPLayers is used as a webapplication. SOLAPLayers reads the parameters out of the HTTP-request and forwards them to the SOLAPLayersProcessor.

Connection

A Connection gives access to a concrete data source. A ConnectionConfiguration is necessary for the construction. At the moment, two different concret connection-types are implemented (SOLAPConnection and DBConnection). More Connections will be developed for a further version of SOLAPLayers.

ConnectionConfiguration
The ConnectionConfiguration can be seen as a textual description of a Connection. The ConnectionConfiguration provides access to the source parameters defined in the sources.xml. A ConnectionConfiguration is necessary to create a Connection.

OutputBuilder
This interface contains the main operation for every OutputBuilder. It allows to build a output out of raw data and provides access to the resulting ouput.

HttpOutputBuilder
This OutputBuilder-interface adds a method to send a HTTP-response back to the client. At the moment all of the concrete OutputBuilders implement this interface. Concrete implementations are OLAPJsonOutputBuilder, SOLAPJsonOutputBuilder, GEOJsonOutputBuilder and the CellSetToOLAPJsonOutputBuilder.

Technologies
The technologies were already given, since this project was an enhancement of an existing software. This chapter illuminate the libraries and technologies, which were used additionally to the Java core libraries. The whole project is developed on Java SE 6.

JSON (JavaScript Object Notation)
The client-server communication uses the HTTP-protocol. The server response is formatted as JSON. JSON is a really compact data transport format with a small overhead. Since JSON is a JavaScript-technology the interpretation on the client side can be done very smooth. Because of the large amount of data is JSON far more capable for SOLAPLayers then other technologies like XML.
Further information: http://www.json.org/

GeoJSON
GeoJSON is a standardized JSON-format with the purpose to transport geo-spatial data. SOLAPLayers supports the generation of GeoJSON.
Further information: http://geojson.org/

Mapfish
Mapfish is a open-source framework to build rich-mapping applications. SOLAPLayers uses just a part of Mapfish. Mapfishs allows to create GeoJSON representations out of Java objects in an easy way. SOLAPLayers uses this library to create the GeoJSON outputs.
Further information: http://mapfish.org/

Log4J
The popular Log4J is used for logging purposes.
Further information: http://logging.apache.org/log4j/

Mondrian
Mondrian is an OLAP-server, which allows to analyse large quantities of data in realtime. SOLAPLayers uses Mondrian to define the XML-Schema to map the relational data into a OLAP-cube representation.
Further information: http://mondrian.pentaho.com/

OLAP4J
What JDBC is for relational databases is OLAP4J for multidimensional-queries. OLAP4J support to create MDX-queries and retrieve the resulting data as OLAP-Cube.
Further information: http://www.olap4j.org/

POSTGis
POSTGis is a PostegreSQL-extension, which allows to store geo-spatial data efficiently. The POSTGis driver is used to query the database out of the Java application.
Further information: http://postgis.refractions.net/

PostgreSQL
All the testdata is stored in PostgreSQL-databases. In general every database with JDBC support can be used, as long it is able to store geo-spatial data.
Further information: www.postgresql.org

JTS Topology Suite
The open-source JTS-library is used for representation of geo-spatial information as Java objects. The results from the database can be parsed directly into geometric-objects and can be used for further purposes.
Further information:
http://www.vividsolutions.com/products.asp?catg=spaapp&code=jts

Other technologies
All other libraries which can be found in the project directory are used by the above metioned technologies (dependencies) and will not be descriped in this documentation.

Client application
As mentioned in the guidelines chapter, the client-side of SOLAPLayers was not a part of this project. The client-side will be described shortly for a better understanding of SOLAPLayers.
SOLAPLayers client side is a HTML 5.0, JavaScript and CSS based application. It displays maps and other dashboard elements, like tables, in the browser. For the presentation of the map and the dashboard elements the EXT (http://www.sencha.com/products/extjs/) and GeoEXT (http://www.geoext.org/) library were used. The use of this library allows to create easyely really flexible and complex dashboards. The transformation of received OLAPJson-data is done by SOLAPLayers, respectively is not a part of the used libraries.
SOLAPLayers 1.0 used Dojo (www.dojotoolkit.org/) as JavaScript-framework. In the new SOLAPLayers 2.0 version, Dojo is not a part of the application anymore.

6.3.4 Implementation

System test

Goal

This test plan assures the correct behaviour of the SOLAPLayer 2.0 Extended server part. A demo dashboard is used for the test, to send and receive information. The object of test is the creation of output-formats using a mdx- or sql-query and specific parameters. This test does not verify the client part of SOLAPLayers, since there was no work done during the development period of the bachelor project.

Requirements and test data

The test data can be found in the Eclipse project in the directory "SpatialyticsMapReportingTool/testqueries/". The files containing the queries and the queries themselves are numbered. Information about the used parameters per query is available as well in the query-files.
The "Query and parameters"-column gives information about the used query and parameters. Technical information about the data sources can be found in the file "SpatialyticsMapReportingTool/webapp/WEB-INF/sources.xml". Preconditions are not available, since the application is stateless and every test conatins the whole A-to-B scenario.

Expectations

The result of every query is the presentation on the dashboard (includes a map, a table and a chart). For one given measure and no dimensions, the polygons on the map will be in different colors, which highlight the proportional relations of the measures. If more than one measure or at least one dimension exists, the polygons will appear in the same color, but including a bar-chart, which shows the different measures. This facts will not be mentioned as expectations in every test case.

Test plan

This test plan was accomplished on 28 March and 1 May 2011. The software was tested on a Linux Ubuntu 20.10 on the browser Google Chrome 11.

MDX-queries

For all the following tests just the parameters "Output Format", "Source Id" and "Query" have to be set, since the other information is included in the MDX-query. For all tests the following parameters are predefined:

- **Output Format:** olapjson *Mentioned if ohter source*
- **Source Id:** 1

The queries are defined in the file "1-mdx-requests.txt".

Number	Test case	Query and parameters	Result
1.1	MDX-Query (see query)	**Query:** 1A	Pass
1.2	MDX-Query (see query)	**Query:** 1B	Pass
1.3	MDX-Query (see query)	**Query:** 1C	Pass
1.4	MDX-Query (see query)	**Query:** 1D	Pass
1.5	MDX-Query (see query)	**Query:** 1E	Pass
1.6	MDX-Query (see query)	**Query:** 1F	Pass
1.7	MDX-Query (see query). Using deprecated output-builder.	**Query:** 1A	Pass
1.8	MDX-Query (see query). Using deprecated output-builder.	**Query:** 1B	Pass
1.9	MDX-Query (see query). Using deprecated output-builder.	**Query:** 1C	Pass
1.10	MDX-Query (see query). Using deprecated output-builder.	**Query:** 1D	Pass
1.11	MDX-Query (see query). Using deprecated output-builder.	**Query:** 1E	Pass
1.12	MDX-Query (see query). Using deprecated output-builder.	**Query:** 1F	Pass

SQL-queries

If a parameter is not listed in the "Query and parameters"-column, it has the default value or stays empty. For all tests the following parameters are predefined:

- **Output Format:** olapjson *The only format which can be displayed dynamically on the map*
- **Source Id:** 3 *Mentioned if ohter source*
- **Geometry Data:** the_geom *Mentioned if other value*
- **Geometry Point Representation:** the_geom_point *Mentioned if other value*

The queries are defined in the file "3-sql-request-db_test2.txt".

Number	Test case	Query and parameters	Additional Expectations	Result
2.1	Simple query with one measure and no dimensions.	**Spatial Dimension:** supname **Measures:** nopersons **Query:** 3A		Pass
2.2	Simple query with two measures and no dimensions.	**Spatial Dimension:** zip_code **Measures:** nopersons, totalsavings **Query:** 3B		Pass
2.3	Simple query with one measure and one dimension.	**Spatial Dimension:** supname **Dimensions:** sexe **Measures:** nopersons **Query:** 3C		Pass
2.4	Query with two measures and one dimension.	**Spatial Dimension:** supname **Dimensions:** sexe **Measures:** nopersons, totalsavings **Query:** 3D		Pass
2.5	Query with two measures and two dimensions	**Spatial Dimension:** zip_code **Dimensions:** sexe, civilstatus **Measures:** nopersons, totalsavings **Query:** 3E		Pass

2.6	Query with three measures and three dimensions	**Spatial Dimension:** supname **Dimensions:** sexe, civilstatus,agegroup **Measures:** nopersons, totalsavings, avgsavings **Query:** 3F		Pass
2.7	Query with one measure and two potential spatial dimensions (one of them will work as a normal dimension).	**Spatial Dimension:** supname **Geometry Data:** the_district_geom **Geometry Point Representation:** the_district_geom_point **Dimensions:** zip_code **Measures:** noperons **Query:** 3G	"the_district_geom" will be displayed on the map and the "supname" as the row value. "zip_code" will be displayed like a simple dimension.	Pass
2.8	Query with one measure and two potential spatial dimensions (one of them will work as a normal dimension).	**Spatial Dimension:** zip_code **Geometry Data:** the_zip_code_geom **Geometry Point Representation:** the_zip_code_geom_point **Dimensions:** supname **Measures:** nopersons **Query:** 3G	"the_zip_code_geom" will be displayed on the map and the "zip_code" as the row value. "supname" will be displayed like a simple dimension.	Pass
2.9	Simple query with one measure and no dimensions.	**Source Id:** 2 **Spatial Dimension:** supname **Measures:** noincidents **Query:** 2A	Works fine for another source as well.	Pass
2.10	Simple query with one measure and one dimension.	**Source Id:** 2 **Spatial Dimension:** supname **Dimensions:** dayofweek **Measures:** noincidents **Query:** 2C	Works fine for another source as well.	Pass

Code quality

The project was reviewed automatically by the Eclipse plugins Metrics (http://metrics.sourceforge.net/) and PMD (http://pmd.sourceforge.net/).

Code and dependencies analysis

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Overridden Methods (avg/max per type)	2	0.067	0.359	2	/SpatialyticsMapReportingTo	
Number of Attributes (avg/max per type)	71	2.367	2.287	8	/SpatialyticsMapReportingTo	
Number of Children (avg/max per type)	4	0.133	0.718	4	/SpatialyticsMapReportingTo	
Number of Classes (avg/max per packageFragment)	30	3.333	0.943	5	/SpatialyticsMapReportingTo	
Method Lines of Code (avg/max per method)	985	5.661	8.07	84	/SpatialyticsMapReportingTo	serializeCellSetAxis
Number of Methods (avg/max per type)	174	5.8	5.121	23	/SpatialyticsMapReportingTo	
Nested Block Depth (avg/max per method)		1.322	0.547	3	/SpatialyticsMapReportingTo	ConnectionConfigurationFactory
Depth of Inheritance Tree (avg/max per type)		1.533	0.763	3	/SpatialyticsMapReportingTo	
Number of Packages	9					
Afferent Coupling (avg/max per packageFragment)		5	5.538	15	/SpatialyticsMapReportingTo	
Number of Interfaces (avg/max per packageFragment)	2	0.222	0.416	1	/SpatialyticsMapReportingTo	
McCabe Cyclomatic Complexity (avg/max per method)		1.511	1.158	11	/SpatialyticsMapReportingTo	getJsonSchemaType
src		1.511	1.158	11	/SpatialyticsMapReportingTo	getJsonSchemaType
org.spatialytics.solaplayers.output.builder		2.067	1.999	11	/SpatialyticsMapReportingTo	getJsonSchemaType
CellSetToOLAPJsonBuilder.java		3	3.162	11	/SpatialyticsMapReportingTo	getJsonSchemaType
OLAPJsonBuilder.java		1.571	0.623	3	/SpatialyticsMapReportingTo	serializePosition
GEOJsonBuilder.java		1.667	0.471	2	/SpatialyticsMapReportingTo	build
SOLAPJsonBuilder.java		1.667	0.471	2	/SpatialyticsMapReportingTo	build
org.spatialytics.solaplayers.connection.driver.db		1.7	1.187	7	/SpatialyticsMapReportingTo	validateParameters
org.spatialytics.solaplayers.output		2.125	1.166	5	/SpatialyticsMapReportingTo	getInstance
org.spatialytics.solaplayers.connection.driver.solap		1.667	0.903	4	/SpatialyticsMapReportingTo	generateCells
org.spatialytics.solaplayers.connection		1.429	0.728	3	/SpatialyticsMapReportingTo	ConnectionConfigurationFactory
org.spatialytics.solaplayers.servlet		1.833	0.687	3	/SpatialyticsMapReportingTo	process
org.spatialytics.solaplayers.data		1.043	0.204	2	/SpatialyticsMapReportingTo	getList
org.spatialytics.solaplayers.data.cube		1	0	1	/SpatialyticsMapReportingTo	HierarchyDTO
org.spatialytics.solaplayers.exception		1	0	1	/SpatialyticsMapReportingTo	ParseException
Total Lines of Code	1902					
Instability (avg/max per packageFragment)		0.557	0.384	1	/SpatialyticsMapReportingTo	
Number of Parameters (avg/max per method)		1.075	1.175	7	/SpatialyticsMapReportingTo	GeoBIFeature
src		1.075	1.175	7	/SpatialyticsMapReportingTo	GeoBIFeature
org.spatialytics.solaplayers.data		0.913	1.666	7	/SpatialyticsMapReportingTo	GeoBIFeature
GeoBIFeature.java		0.889	2.183	7	/SpatialyticsMapReportingTo	GeoBIFeature
RawData.java		1.167	1.675	4	/SpatialyticsMapReportingTo	RawData
Parameters.java		0.75	0.661	2	/SpatialyticsMapReportingTo	put
org.spatialytics.solaplayers.connection.driver.solap		1.444	1.595	5	/SpatialyticsMapReportingTo	executeDrillReplace
org.spatialytics.solaplayers.connection.driver.db		1.167	1.003	4	/SpatialyticsMapReportingTo	getSubcolumn
org.spatialytics.solaplayers.data.cube		0.538	0.779	4	/SpatialyticsMapReportingTo	MemberDTO
org.spatialytics.solaplayers.output		1.625	0.992	4	/SpatialyticsMapReportingTo	sendHttpResponse
org.spatialytics.solaplayers.output.builder		1.233	0.803	4	/SpatialyticsMapReportingTo	serializePosition
org.spatialytics.solaplayers.exception		2	0	2	/SpatialyticsMapReportingTo	ParseException
org.spatialytics.solaplayers.servlet		1.167	0.687	2	/SpatialyticsMapReportingTo	doPost
org.spatialytics.solaplayers.connection		0.857	0.35	1	/SpatialyticsMapReportingTo	ConnectionConfigurationFactory
Lack of Cohesion of Methods (avg/max per type)		0.374	0.359	0.881	/SpatialyticsMapReportingTo	
Efferent Coupling (avg/max per packageFragment)		2.444	1.343	4	/SpatialyticsMapReportingTo	
Number of Static Methods (avg/max per type)	0	0	0	0	/SpatialyticsMapReportingTo	
Normalized Distance (avg/max per packageFragment)		0.371	0.382	1	/SpatialyticsMapReportingTo	
Abstractness (avg/max per packageFragment)		0.072	0.14	0.4	/SpatialyticsMapReportingTo	
Specialization Index (avg/max per type)		0.067	0.359	2	/SpatialyticsMapReportingTo	
Weighted methods per Class (avg/max per type)	263	8.767	8.421	34	/SpatialyticsMapReportingTo	
Number of Static Attributes (avg/max per type)	102	3.4	10.111	56	/SpatialyticsMapReportingTo	

Figure 12 - Metrics report of SOLAPLayers 2.0 Extended

The red highlighted problems occur because of the deprecated classes CellSetToOLAPJsonBuilder and GeoBIFeature.
The class CellSetToOLAPJsonBuilder is a class from SOLAPLayers 1.0 and remains in the project just for testing and developing reasons. All the functionality is given by the new class OLAPJson. No refactoring for the class CellSetToOLAPJsonBuilder was done, that is why the McCabe Cyclomatic Complexity is to high.
The class GeoBIFeature is deprecated and will be removed in a further version. All the information which this data class contains, is provided by the class CubeDTO as well. This refactoring is mentioned in the chapter „Futher Improvements“.

6.3.5 Future improvements

SOLAPLayers 2.0 Extended is not a final version. During the development process some ideas ,to improve the application, came up. Chores can be done without analyzing the solution deeply. The extension ideas have to be analyzed before the implementation.

Chores

- Add logging to the workflow, to allow to monitor the workflow and simplify the debugging.
- Differentiate the values null and zero when retrieving data from a data source and creating the output. It is important to know if the value does not exists or if the value is equals zero. This has to be made in the OLAPJson-specification.
- Accomplish the exception handling by passing the exception, respectively a related message, to the client. Add a sendException()-method to the abstract „OutputBuilder"-class. This method will be called when an exception arises during the process. So each „OutputBuilder" can send an appropriate exception.
- Adding all exception texts into a resource-file. This allows better administration of the texts and makes is easy to display exception-messages in different languages.

Extensions

Extend the data abstraction

There are two main requirements to the data abstraction:

1. The exchange-format „RawData" does contain redundancy. All the information the output builders need, are contained in the CubeDTO (a representation of an OLAP-cube). The list of „MfFeature" is unessential. On the one hand, The Feature-list simplyfies the creation-process of the output builder, on the other hand is it unnecessary for the connection to build this list. This list has to be removed for a clean implementation.
2. The CubeDTO is not complete. It contains all the essential data read by the connection. The metadata and filter Axis are not yet a part of the CubeDTO and should be add in an other iteration (attention: These changes affect the "OutputBuilder"- and "Connection"-classes. The affected classes have to be extended as well).

Add features to add an OutputBuilder without modifying core

At the moment every new OutputBuilder has to be registered by modifying the core-source code. This is a very poor implentation. This circumstances can be removed by adding a configuration-file or using reflection.

Add Connection manager

SOLAPLayers is stateless at the moment. A specific connection is opened and closed for each request. This needs a lot of ressources. A connection manager, which overviews the used connection, could remove this circumstances and cache open connections for further use before closing them.

Add Cache for geospatial-data

Reading geospatial-data and relations to geospatial-data need a lot of CPU power, respectively time. Different requests refer often to the same geometries. It would be useful to cache this data on the server- or client-side instead of recalculating them for each request.

Define XML-schema for sources.xml

For better validation and less exceptions it is recommended to create a XML-schema for the sources.xml. It allows to recognize problems in advanced, before running the programm.

Add query-builder

There are different guidelines for a SOLAPLayers SQL-query and the additional parameters (See "Using SQL-Queries" and "Analysis - Result set to OLAP-cube transformation"). A query-builder could simplify the construction of a valid sql-query. With an intelligent query-builder it is possible to validate the query in relation to the parameters and provide list boxes to set the query-parameters.

Standardize OLAPJson

OLAPJson is an undocumented data format, which contains an OLAP-cube representation. For better understanding and collaboration with other application, it would be useful to document and describe the OLAPJson format. In a further step a XML schema could be defined.

6.3.6 Developers guide

SOLAPLayers is a framework which can be used by developers. It is not yet enough understandable and mature to provide it to people with no technical background. It is recommended to read this instructions for using the SOLAPLayers-Framework successful. The instructions help to build new connection- and output-types and teach how to work with the SOLAPLayers server part.

Installation

The installation was tested on a Linux Ubuntu 20.10 system. The computer has to be connected to the Internet to run the application with the test databases.

Tools

For to set up and run SOLAPLayers 2.0 Extended some software has to be installed on the working machine. Installation and user guides for these applications can be found on their websites:

- **Eclipse Helios**. For further use with the BIRT-Plugin the "Eclipse IDE for Java and Report Developers" is recommended (http://www.eclipse.org/downloads/packages/eclipse-ide-java-and-report-developers/heliossr2).
- **Tomcat 6**. The program is tested on the version 6, newer tomcat versions should work as well (http://tomcat.apache.org/tomcat-6.0-doc/index.html).
- Additionally: **Subclipse** (http://subclipse.tigris.org/) or other SVN-Plugin for Eclipse. If the source code will be imported using SVN an appropriate Eclipse-plugin is useful.

Step by step manual

- Start Eclipse and create or open a workspace
- Get the source code
    - Import SOLAPLayers 2.0 Extended project code using the import-functionality of Eclipse "Existing Projects into Workspace". The zip-file is located on the project CD ("SOLAPLayers 2.0 Extended/solaplayers_2.0_extended.zip").
- To run SOLAPLayers 2.0 Extended properly, some adjustments have to be done:

  - Open the file “build.properties” and customize the “tomcat_webapps_path”. The path has to point on the webapps-directory of your tomcat installation.
  - Adjust the path to the sources.xml-file in the file “/webapp/WEB-INF/web.xml”. Point on the sources.xml-file, which is located in the workspace “webapp/WEB-INF/sources.xml” or create your own sources.xml at any directory (use an absolute path).
  - Notice: Check the sources-configuration in the sources.xml file. For using the sample OLAP-source, the “catalog”-parameter of the connectionstring-element has to be customized, so that it points on the “SOLAPLayers-2.0/WEB-INF/schemas/Canpop.xml“-file on the tomcat-server.
  - Copy the postgresql.jar into the “lib”-directory of the tomcat installation. The library can be found on the project CD widespread (“SOLAPLayers 2.0 Extended/Utils/postgresql.jar”).
- It is recommended to stop tomcat before deploying the application.
- To build the project drag and drop the build.xml-file into the “Ant”-view and double click on “deploy-war”. Execute first “clean”, if SOLAPLayers 2.0 Extended is already installed on the server.
- Start tomcat
- Access SOLAPLayers using a browser: http://localhost:8080/SOLAPLayers-2.0/ (not all browsers display SOLAPLayers 2.0 Extended properly. SOLAPLayers runs right on Ubuntu using Google Chrome).
- Login into SOLAPLayers (Username: demo, password: Spatialytics)
- Click on “Test your MDX queries and observe the JSON output from the server.”
- Send a correct SQL and MDX-query to check the SOLAPLayers installation.
  - Notice: If you use the sample queries located in the “testqueries”-directory check for wrong displayed special characters. It can happen that the operating system displays them wrong.

Add a new connection type

Everything that provides data, which can be transformed in an OLAPCube, is a potential data source. Before starting with developing it is useful to analyze the transforming.
As soon the idea is ready, a new class, which has to implement the “Connection”-interface, can be created. Three public methods have to be coded, to provide a working connection-implementation:

Connect() - Creating a connection to the data source

Depending on the kind of data source a connection maybe initialized before it is possible to read data. If no connection is required, this method has to be implemented empty, or can be used to read data out of the configuration.

Execute(Parameters parameters) - Retrieving and transforming data

This is the most important and complex part of the connection implementation. Depending on the complexity of this procedure, it is recommended to use additional classes to make the source code more understandable (E.g. Parser- or query-wrapper-class).
The first thing is to use the already established connection to retrieve the data. The query or/and the parameters which lead to the data are contained in the request-object, which is passed to this method.

After reading the data, the information has to be parsed into a “RawData”-object. At the moment a “RawData”-object contains a “CubeDTO”, which represents an OLAP-cube and a List of “MfFeatures”, which represents the geospatial-data in a simple way. The content of RawData may change with every version of SOLAPLayers.

Close() – Clean up and close connection

The last step of integrating a new connection type is to clean up and close the connection. As mentioned above does not every connection type need a permanent connection, since it is possible to access the data directly.
In the normal case, the connection to the data source has to be closed in this method (E.g. database).

Add a new output format

In comprehension to the connection type the developer has to do some changes in the core application (this may be improved by a next version, see “Add features to add an OutputBuilder without modifying core“). The following steps show how to develop a new output format.
After creating a new class, which extends the abstract class “OutputBuilder” the developer has to implement three methods:

Build(RawData data) – Build the output

This method receives the transformed data from a connection and has to retransform it into a specific output format. The method does not return the converted data to the caller and does not yet send any response to the client, so it can also be used by other output builders (e.g. Chart output builder delivers chart to a pdf output builder).

getOutput() – Access the generated output

This method allows accessing the generated output. The method returns an object, which can be parsed in the excepted type.

sendHttpResponse(HttpServletResponse response) – Send the response to the client

This method sets the Header of the http-response and may do some last modification, which are specific for the receiver. Finally, it sends the http-response. This is the end of the workflow.

Register new output format

The framework chooses the output-format by a parameter of the HTTP-request. The construction of all output-builders is done in the „OutputBuilderFactory“-class. Edit the getInstance(String outputFormat)-method to provide the new output-format to the framework.

```
public class OutputBuilderFactory {

    private static final String FORMAT_SOLAPJSON = "solapjson";
    private static final String FORMAT_OLDOLAPJSON = "oldolapjson";
    private static final String FORMAT_GEOJSON = "geojson";
    private static final String FORMAT_OLAPJSON = "olapjson";

    /**
     * Locates the correct OutputBuilder for the given output format.
     *
     * @param outputFormat String, which represent the output format.
     * @return Corresponding OutputBuilder
     */
    public OutputBuilder getInstance(String outputFormat) {
        if (outputFormat.equalsIgnoreCase(FORMAT_SOLAPJSON)) {
            return new SOLAPJsonBuilder();
        }
        if (outputFormat.equalsIgnoreCase(FORMAT_OLDOLAPJSON)) {
            return new CellSetToOLAPJsonBuilder();
        }
        if (outputFormat.equalsIgnoreCase(FORMAT_GEOJSON)) {
            return new GEOJsonBuilder();
        }
        if (outputFormat.equalsIgnoreCase(FORMAT_OLAPJSON)) {
            return new OLAPJsonBuilder();
        }
        // default: OLAPJson
        return new OLAPJsonBuilder();
    }
}
```

Figure 13 - OuputBuilderFactory class, which will be replaced in a future version

Add a new data source

Before adding a new data source two things have to be ensured. Firstly, a data source have to exist and the connection information has to be available. Secondly, an appropriate connection type has to be available.

Now the developer can simply add a new source at the end of the sources-files (or between two sources). Every new source needs a id and has to enclose a class-tag, which contains the full package path to the connection, respectively the driver, class. All other required parameters depend on the driver.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <sources>
3      <source id="1">
4          <class>org.spatialytics.solaplayers.connection.driver.solap.SOLAPConnection</class>
5          <driver>mondrian.olap4j.MondrianOlap4jDriver</driver>
6          <connectionstring>[redacted]                    MONDRIAN - MDX
7      </source>
8      <source id="2">
9          <class>org.spatialytics.solaplayers.connection.driver.db.DBConnection</class>
10         <connectionstring>jdbc:postgresql://dev.spatialytics.com/db_test</connectionstring>
11         <user>[redacted]</user>
12         <password>[redacted]</password>
13         <driver>org.postgresql.Driver</driver>          POSTGRE - SQL
14     </source>
15     <source id="3">
16         <class>org.spatialytics.solaplayers.connection.driver.db.DBConnection</class>
17         <connectionstring>jdbc:postgresql://dev.spatialytics.com/db_test2</connectionstring>
18         <user>[redacted]</user>
19         <password>[redacted]</password>
20         <driver>org.postgresql.Driver</driver>          POSTGRE - SQL
21     </source>
22 </sources>
```

Figure 14 - Sample of sources.xml with different data sources

DBConnection

The following parameters have to be set additionaly for a connection using the „DBConnection“-driver:

connectionstring	Path to the jdbc-ressource
user	Name of the database user
passwort	Passwort of the database user
driver	Driver package depending on the type of database (Example: Postgres, Oracle, DB2, MySql, etc.)

SOLAPConnection“

The following parameters have to be set additionaly for a connection using the „SOLAPConnection“-driver:

connectionstring	Path to the OLAP-ressource including name and password of DB-user
driver	Driver package depending on the type of database (Example: Postgres, Oracle, DB2, MySql, etc.)

Using MDX-queries

Driver: *org.spatialytics.solaplayers.connection.driver.solap.SOLAPConnection*

Using MDX-Queries is the simplest way to display BI- and geospatial data. Just three parameters have to be send to the server, all other parameters will be ignored:

- **Output Format:** For displaying the data on a SOLAPLayers-dashboard enter “olapjson” as output format. You can use any other existing output format, if you not want to display the data on a SOLAPLayers dashboard.
  An alternative to “olapjson” is the “oldolapjson”-output-format. It creates the exact same output as “olapjson”, but uses a different way of data transformation on the server side, which is deprecated. This possibility should be used just for testing and developing issues.
- **Source Id:** Enter the id of an existing source. The source id has to refer to a valid SOLAPConnection-source. The location of the source file is defined in the “WEB-INF/web.xml”-file located in the project directory. For defining a new source, read chapter “Add a new data source”.

- **Query:** Enter a mdx-query for retrieving data from the choosen data source. At the moment the only limitation is that the geographic data has to be selected on rows as a single field.

Example

```
SELECT
  CrossJoin
  (
    {[Measures].[Population]}
   ,CrossJoin
    (
      {
        [Age].[All Ages].[Ag√©s 65+]
       ,[Age].[All Ages].[Jeunes <25]
      }
     ,{
        [Sexe].[All Sexes].[Femme]
       ,[Sexe].[All Sexes].[Homme]
      }
    )
  ) ON COLUMNS
 ,{
    [Unite geographique].[All Unite geographiques].[Canada].[Atlantique].[Terre-Neuve-et-Labrador]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Atlantique].[Ile-du-Prince-Edouard]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Atlantique].[Nouvelle-Ecosse]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Atlantique].[Nouveau-Brunswick]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Centre].[Qu√©bec]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Centre].[Ontario]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Prairies Territoire du Nord-Ouest et Nunavut].[Manitoba]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Prairies Territoire du Nord-Ouest et Nunavut].[Saskatchewan]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Prairies Territoire du Nord-Ouest et Nunavut].[Alberta]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Prairies Territoire du Nord-Ouest et Nunavut].[Territoires du Nord-Ouest]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Prairies Territoire du Nord-Ouest et Nunavut].[Nunavut]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Pacifique Yukon].[Colombie-Britannique]
   ,[Unite geographique].[All Unite geographiques].[Canada].[Pacifique Yukon].[Territoire du Yukon]
  } ON ROWS
FROM [Recensements]
WHERE
  [Temps].[Rencensement 2001 (2001-2003)].[2001];
```

Figure 15 - Sample MDX query for SOLAPLayers

	Measures			
	Population			
	Age			
	+Agés 65+		+Jeunes <25	
	Sexe		Sexe	
Unite geographique	Femme	Homme	Femme	Homme
+Terre-Neuve-et-Labrador	971,405	1,002,824	944,965	970,248
+Ile-du-Prince-Edouard	286,428	295,436	283,504	284,762
+Nouvelle-Ecosse	1,733,856	1,754,315	1,730,663	1,734,700
+Nouveau-Brunswick	1,438,112	1,461,136	1,409,821	1,411,559
+Québec	9,876,769	9,731,161	9,598,395	9,771,142
+Ontario	5,100,121	4,926,097	5,630,262	5,753,942
+Manitoba	2,232,058	2,276,096	2,268,853	2,259,153
+Saskatchewan	1,757,862	1,783,384	1,710,732	1,706,682
+Alberta	1,914,691	1,875,273	2,087,206	2,120,970
+Territoires du Nord-Ouest	188,110	197,920	191,777	187,924
+Nunavut	295,360	284,214	272,659	292,291
+Colombie-Britannique	2,848,856	2,741,621	2,954,277	3,016,258
+Territoire du Yukon	105,068	83,569	90,468	87,952

Figure 16 - Result of sample MDX query displayed as cross table

Using SQL-queries

Driver: *org.spatialytics.solaplayers.connection.driver.db.DBConnection*

A classic SQL-query contains not as much information as a mdx-query. Since we use a OLAPJson-format as the data transfer format between the server- and the client side, we have to provide some additional information to convert the result set of a SQL-query to

an OLAPCube. We have to define the request in two steps for creating a sql-request: Creating a query and defining the parameters.

How to build a valid query

The used sql-query can be as complex as wished (using joins, where-conditions, unions, etc.). The only condition is, that the following considerations have to be done:

1. ***Which geographic information should be represented on the map?***
   Select a label (**Spatial Dimension**), a polygon-geometry (**Geometry Data**) and a point representation of the geometry(**Geometry Point Representation**). Group the query by **Spatial Dimension** and by each independent **Geometry**.
   Example: Country name, geometry and point representation geometry of country
2. ***Which values (Measures) should be displayed in the crosstable?***
   At least one measure has to be listed in the select part. The measure column has to contain numeric values. Use a corresponding aggregation function to group the measures.
   Example: GDP, Population, Average salary, etc.
3. ***From which perspective you want to look at the measures (Dimension)?***
   Every measure has to be listed in the select and the group by-clause.
   Example: Sex, age, nationality, profession, etc.
4. ***Do you want to read all the data?***
   Use where-conditions to restrain the retrieving data. Usually the where clause is used on dimensions, but it is possible to use them as well on measures and the geometry-data, if it is useful.
   Example: Receive just information about females, adults, Canadians, teachers, etc.

Requirements of the additional parameters

The following list shows the required and optional parameters, which are necessary beside the query. The careless use of the parameters can produce wrong reproduction of the information:

- **Output Format:** For displaying the data on a SOLAPLayers-dashboard enter “olapjson” as output format. You can use any other existing output format, if you do not want to display the data on a SOLAPLayers dashboard.
  Do not use the “oldolapjson”-format. This is a deprecated format which can only be used together with a SOLAP data source. Possible dataformats in the current version are “geojson”, “olapjson”, “oldolapjson” and “solapjson”.
- **Source Id:** Enter the id of an existing source. The source id has to refer to a valid DB-source. The location of the source file is defined in the “WEB-INF/web.xml”-file located in the project directory. For defining a new source, read chapter “Add a new data source”.
- **Spatial Dimension:** This required parameter has to match a single selected column in the sql-query. There should be a logical relation between the “Spatial Dimension” and the “Geometry Data”, since the “Spatial Dimension” is nothing else than an alias for the geometry.
- **Geometry Data:** This required parameter has to match a single selected column in the sql-query. It represents a polygon or a multi-polygon on the map. The SRID of the column has to be 4326, since the SOLAPLayer client part does not support any other SRID at the moment (*PostgreSQL/PostGIS*: the ST_Transform(geometry, srid)-method can provide transformations from other geo-data representations).
- **Geometry Point Representation:** This is a point representation of the “Geometry Data”, which is used to position diagrams on the map. It is required

and the name has to match a column of the sql-query. The same conventions as for the "Geometry Data" are necessary (*PostgreSQL/PostGIS*: the ST_PointOnSurface(geometry)-method can build a point representation out of the "Geometry Data", if the database table does not provide a point representation-column).

- **Dimensions:** This parameter is required if there is a dimension (beside the spatial dimension) in the sql-query. The parameter has to match the dimension/s in the sql-query. Each dimensions has to be listed (separated by comma). The order of the dimensions affects the presentation. The first dimension name is on the top of the table headings. Do not send any parameters, if no dimension is provided by the sql-query.

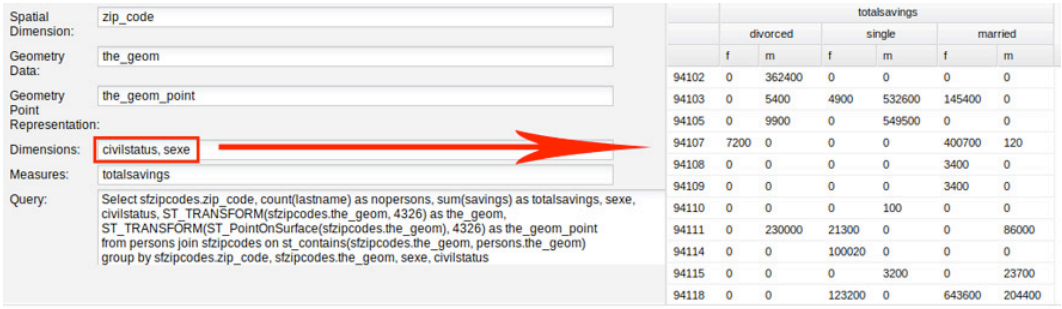


Figure 17 - Sample cross table in SOLAPLayers dashboard with dimensions civil status and sex

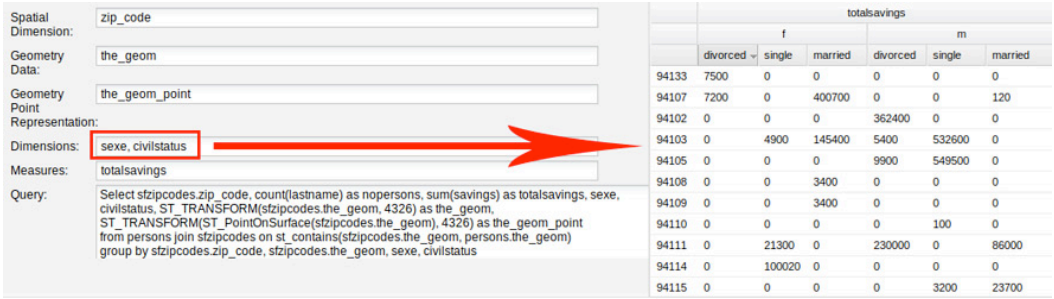


Figure 18 - Sample cross table in SOLAPLayers dashboard with dimensions sex and civil status

- **Measures:** At least one measure has to be set. If the sql-query provides more than one measure, one can be chosen or they can be list (separated by a comma). The first mentioned name will appear as the first column in the table representation.

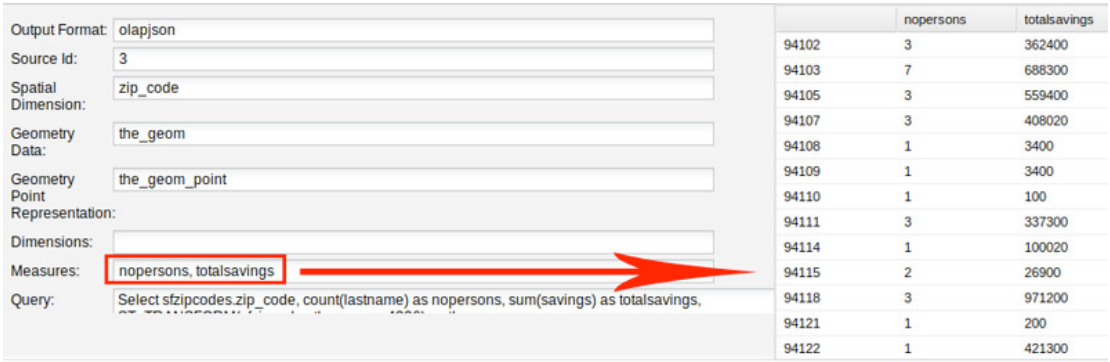
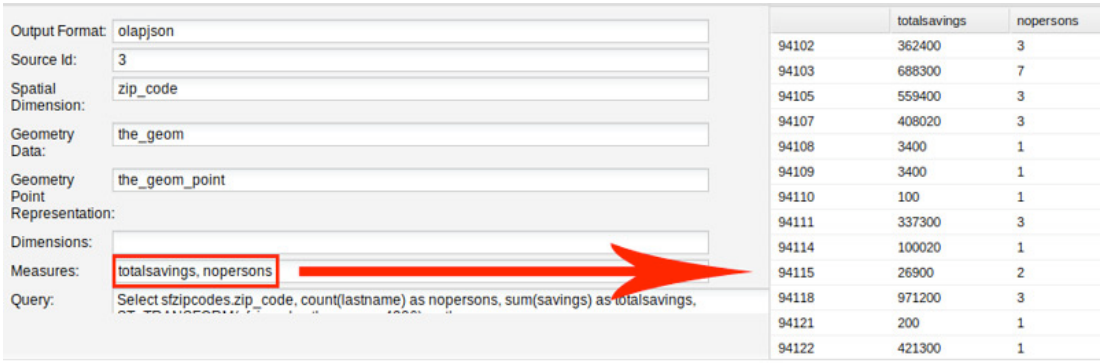


Figure 19 - Sample cross table in SOLAPLayers dashboard with the measures number of persons, total savings



	totalsavings	nopersons
94102	362400	3
94103	688300	7
94105	559400	3
94107	408020	3
94108	3400	1
94109	3400	1
94110	100	1
94111	337300	3
94114	100020	1
94115	26900	2
94118	971200	3
94121	200	1
94122	421300	1

Figure 20 - Sample cross table in SOLAPLayers dashboard with the measures total savings, number of persons

Example

```
/* Two Measures, No Dimension - Number of peoples and total savings per zip-code-area */
Select sfzipcodes.zip_code, count(lastname) as nopersons, sum(savings) as totalsavings,
ST_TRANSFORM(sfzipcodes.the_geom, 4326) as the_geom,
ST_TRANSFORM(ST_PointOnSurface(sfzipcodes.the_geom), 4326) as the_geom_point from persons
join sfzipcodes on st_contains(sfzipcodes.the_geom, persons.the_geom)
group by sfzipcodes.zip_code, sfzipcodes.the_geom
```

Figure 21 - Sample SQL query for SOLAPLayers

	zip_code double precision	nopersons bigint	totalsavings bigint	the_geom geometry	the_geom_point geometry
1	94102	3	362400	010600002(	0101000020E6100
2	94103	7	688300	010600002(	0101000020E6100
3	94105	3	559400	010600002(	0101000020E6100
4	94107	3	408020	010600002(	0101000020E6100
5	94108	1	3400	010600002(	0101000020E6100
6	94109	1	3400	010600002(	0101000020E6100
7	94110	1	100	010600002(	0101000020E6100
8	94111	3	337300	010600002(	0101000020E6100
9	94114	1	100020	010600002(	0101000020E6100
10	94115	2	26900	010600002(	0101000020E6100
11	94118	3	971200	010600002(	0101000020E6100
12	94121	1	200	010600002(	0101000020E6100
13	94122	1	421300	010600002(	0101000020E6100
14	94123	1	745000	010600002(	0101000020E6100
15	94132	2	124800	010600002(	0101000020E6100

Figure 22 - Result of sample SQL displayed as table

Creating a dashboard

An example how to create a dashboard can be find in the presentation “Dashboard and reporting tools: Integration of BI tools in the geospatial domain“ by Thierry Badard of Spatialytics. The mentioned notes can be found in the appendix.

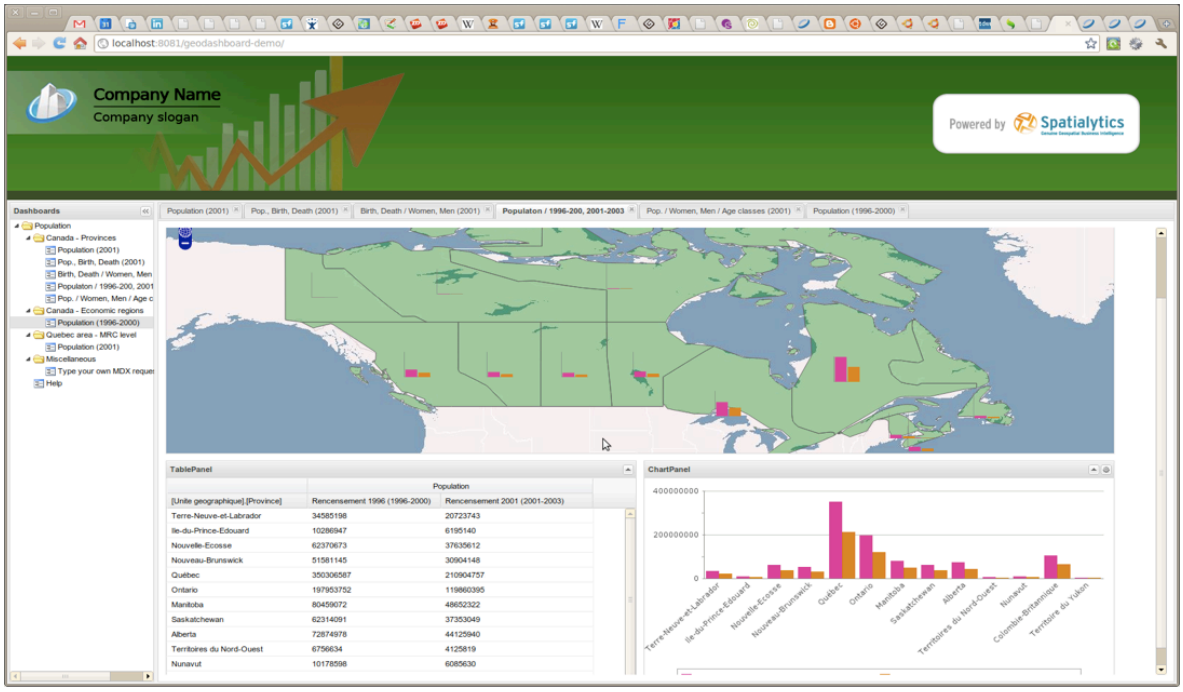


Figure 23 - SOLAPLayers sample dashboard

7 Geoextensions2Birt

7.1 Overview

This part of the documentation explains the integration of SOLAPLayers 2.0 Extended into a BIRT-plugin. The plugin has the name Geoextensions2Birt. Please have a look at the documentation above for SOLAPLayers related content. Beside the technical content, a user- and developer-guide can be found at the end of this chapter.

7.1.1 Technical report

Introduction

One of the main goal of Spatialytics is to integrate their products into well-known products. One of a couple of widespread reporting tools is BIRT. BIRT is a product of the American company Actuate and a part of the Eclipse Foundation. For further steps a prototype of a map report item for BIRT has to be developed. Decisions and results are documented in the following chapters.

State of the Art

BIRT itself does not contain any mapping features. The only solution to display maps in a BIRT report is to use one of the map-libraries, which Actuate provides.

Map It (Google Maps) and Flash Maps Showcase

Map It and Flash Maps Showcase are not a real BIRT plugins. The both extensions are BIRT report-libraries. The library contains a couple of predefined text-items, which can be dragged into a report. The items contain JavaScript-code, which displays a Google Map or the Flash representation. The different code-samples allow customizing the map and binding data. A Google Map API-key is necessary for Map It.

Advantages

- it is really flexible and easy to customize Map It and Flash Maps Showcase, since they contain just pre-scripted JavaScript- and some Flash-files,.

Disadvantages

- Can not be installed that easy as a plugin
- Does not allow to configure the map using extended editor functionality
- Displayed as a big construct of table and text fields in the report designer. The report designer does not represent the map in anyway.
- Hard to handle for non-developers

Conclusion

It is easy way to work with a map in Birt to use one of these extensions. The usability is very bad and the map is hard to customize for non-developers. All in all these plugins are more a workaround to display maps, than a real extension for BIRT.

7.1.2 Project documentation

Requirements specification

The requirement specification is done by Spatialytics in an agile way. The SCRUM-Task board can be found in the appendix (see “APPENDIX E – project management”).

Analysis

There are different possible solutions to receive the data from a data source. These different solutions were analysed and rated to find the best possible one for this particular plugin. The decision relayed on different aspects:

- **Usability**
  The goal is to make it easy for the user to define a new data source. Less user defined information as possible should be required.
  Weight: 1*:* Since this version of the plugin is a prototype and will be used for demonstration purpose, features will be more important than a good usability.
- **Number of connections**
  Reporting and performance have a strong relation. For a big amount of data, the report-generation can use hours to finish. That is why as less possible connections as possible should be created to receive the data. In the best case one connection per data source is necessary.
  Weight: 2*: Multiple connections have an influence on the usability, the performance and the complexity.
- **Data transformation**
  For performance reasons the received data should be displayed without to transform the data into another representation. This can cost a lot of CPU and time, if the data amount is large
  Weight: 1*: Data transformations cost performance, but are necessary for almost every data source, to fit the common data interface (ODA).
- **Reusability**
  BIRT is just one platform Spatialytics deals with. In the future plugins for other products should be deployed as well. As much as possible of the implemented code should be reusable for further projects.
  Weight: 3*: Spatialytics will provide this kind of plugins for further reporting tools. It saves a lot of time and effort, if the work can be reused for other projects.
- **Effort and time**
  Time is money. For this reasons the plugin should be implemented in a time saving way.
  Weight: 2*: The goal is to create a prototype as fast as possible, that is why "effort and time" is an important aspect.
- **Extensibility**
  The goal is to create a plugin, which can be extended to fit all possible upcoming requirements. If it is not possible to extend the plugin, a lot of additional work has to be done.
  Weight: 3*: A prototype is a demo-object. In the future a lot of requirements will come up. It should be possible to implement the new features into the prototype.

Some of the following solution ideas use SOLAPLayers as a component of the BIRT plugin. SOLAPLayers provides the possibility to query different data sources and an html-based interactive map component.
Two important guidelines have to be considered:

- All the guidelines, which were given for SOLAPLayers 2.0 Extended, are valid for the BIRT-plugin as well. Have a look at the "Guidelines"-chapter in the "Technical report"-chapter of the SOLAPLayers 2.0 Extended documentation.

- The BIRT data source connections are based on the ODA interface. All the data BIRT receives is in a two-dimensional format. Three- or multidimensional formats are not supported.

Variant 1 - SOLAPLayers with JSON-output and two connections

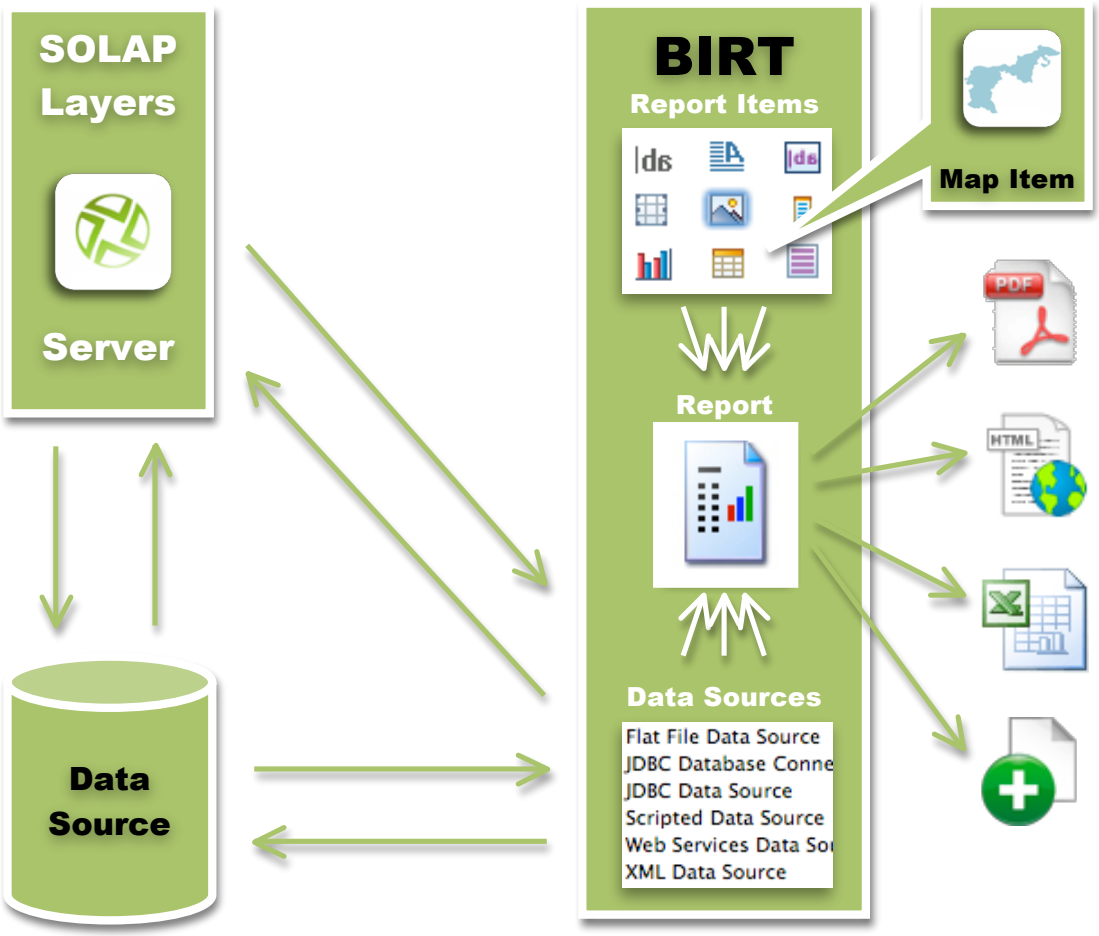


Figure 24 – Illustration: SOLAPLayers with JSON-output and two connections

Description

The BIRT plugin works with two different connections. The first connection reads data from the given source to display it in a table or another report item. This could be done in an easy way with the BIRT core features, no extension would be necessary. For to display the map item a second connection is necessary to receive the map-information.

Dis-/Advantages

Advantages	Disadvantages
+ Profit from SOLAPLayers features	- Two connections (and data source definitions) necessary - Map data source is not compatible with core report items - Hack for ODA data source necessary (result is not a two-dimensional result set)

Variant 2 - SOLAPLayers with Image-output and two connections

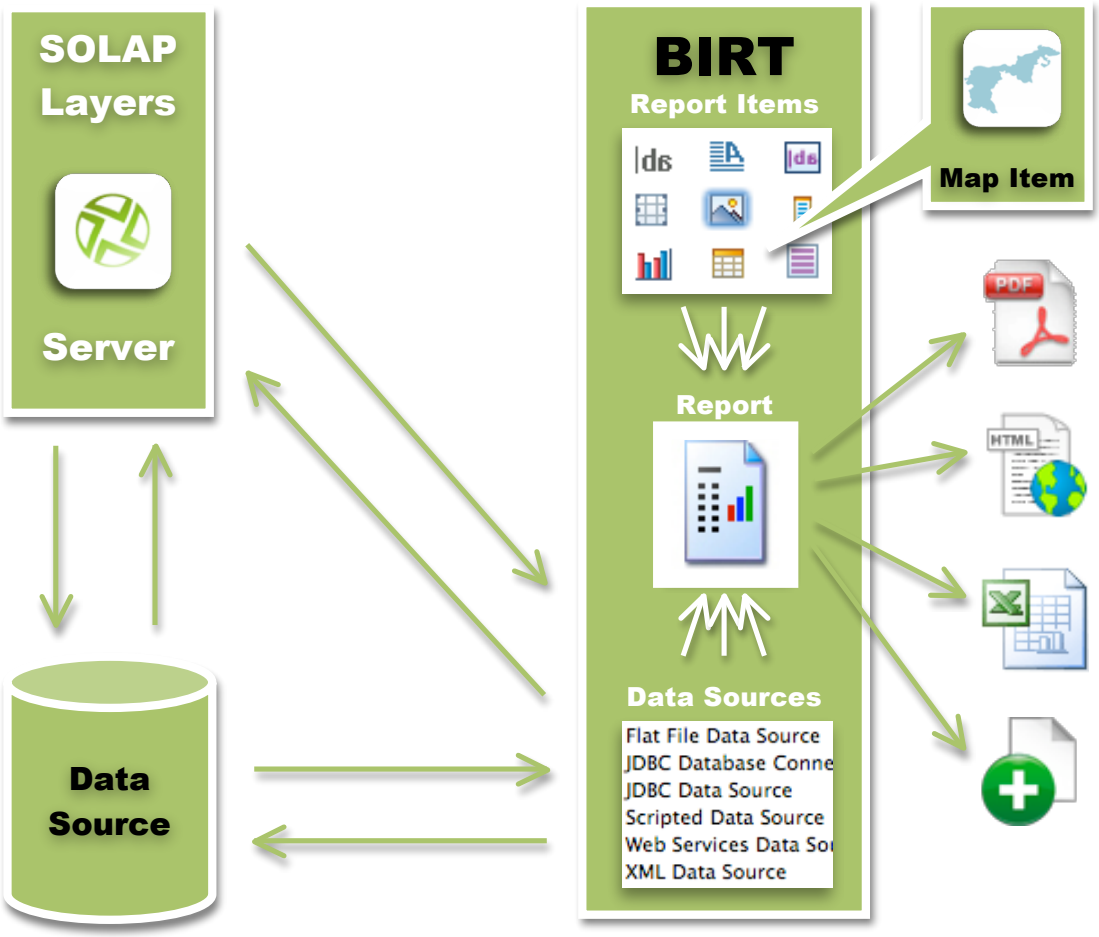


Figure 25 – Illustration: SOLAPLayers with Image-output and two connections

Description

As in variant one, the data for the normal data source can be received using the BIRT core functionality. For to display the map, SOLAPLayers to receive data and create an image-map (For a variant without SOLAPLayers see variant 4).

Dis-/Advantages

Advantages	Disadvantages
+ Profit from SOLAPLayers features	- Two connections (and data source definitions) necessary - Image is not interactive - Hack for ODA data source necessary (result is not a two-dimensional result set)

Variant 3 - SOLAPLayers and one connection

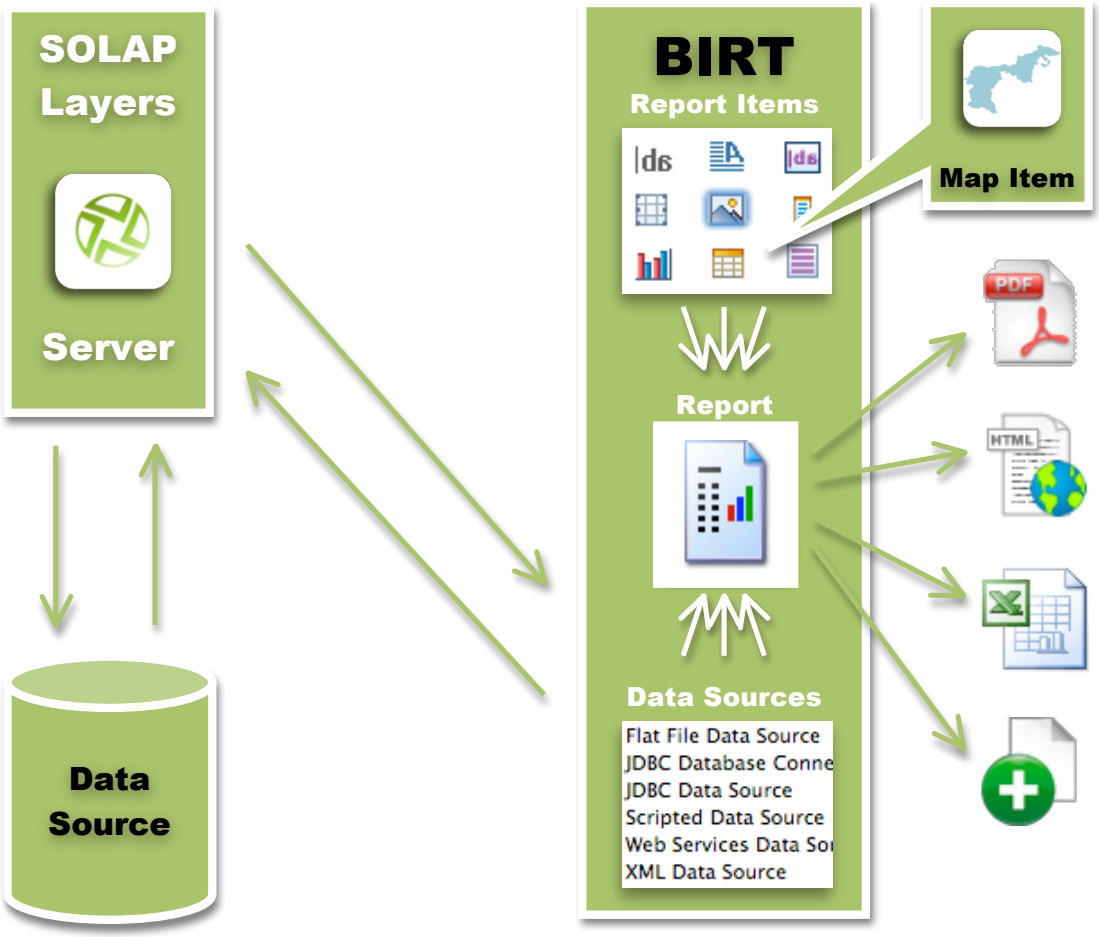


Figure 26 – Illustration: SOLAPLayers and one connection

Description

This solution just creates a connection to SOLAPLayers. No more connections are necessary. The received data contains the data as two-dimensional result set and an OLAPJson for to interpret the map representation. To display the received data from SOLAPLayers the data has to be transformed into a two-dimensional data set.

Dis-/Advantages

Advantages	Disadvantages
+ Profit from SOLAPLayers features + Just one connection + High flexibility	- Hack for ODA data source necessary (result is not a two-dimensional result set)

Variant 4 – One connection without SOLAPLayers

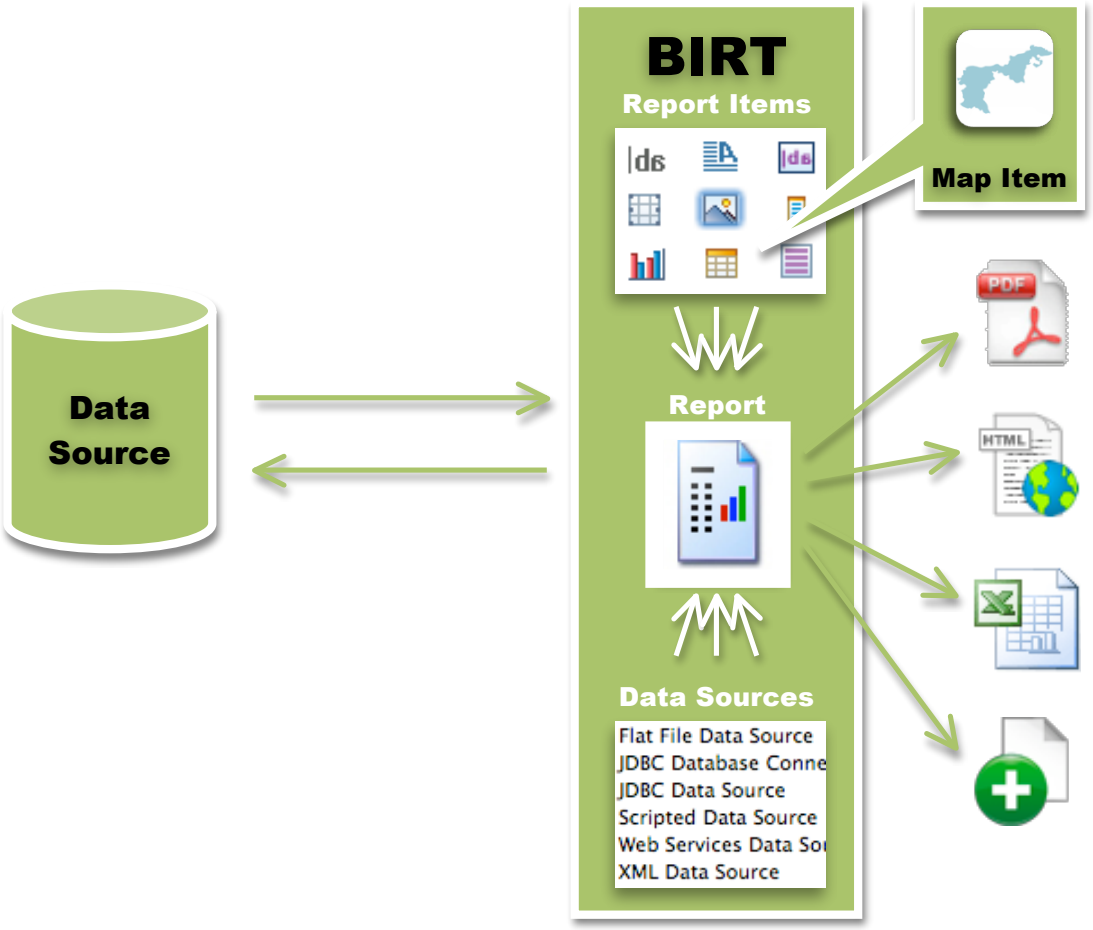


Figure 27 – One connection without SOLAPLayers

Description

This variant does not use SOLAPLayers, instead the data sources are connected using the BIRT drivers. The problem will be to store the data for the map representation into a two-dimensional result set. Maps can display complex relations, which are hard to store in a two-dimensional data set. The big advantage would be, that the ODA-definition could be used regularly.

Dis-/Advantages

Advantages	Disadvantages
+ Just one connection	- Bad reusability
+ Proper use of the ODA-interface	
+ All BIRT drivers can be used	

Conclusion

The first number shows the number of points, the second the final importance including the weight of the aspect.

	Variant 1	Variant 2	Variant 3	Variant 4
Usability (1*)	1 / 1	1 / 1	2 / 2	3 / 3
Number of connections (2*)	1 / 2	1 / 2	2 / 4	2 / 4
Data transformation (1*)	2 / 2	2 / 2	1 / 1	3 / 3
Reusability (3*)	2 / 6	2 / 6	3 / 9	1 / 3
Effort and time (2*)	3 / 6	2 / 4	3 / 6	1 / 2
Extensibility (3*)	3 / 9	3 / 9	2 / 6	1 / 3
Total	**26**	**24**	**28**	**20**

The variant 3 will be implemented. The ideas of the variant 1 and 4 will be dismissed. The variant 3 allows to implement an image-output as mentioned in variant 2 in the future.

Architecture

The architecture of the BIRT-plugin is quite simple. The project used different extension-points to integrate the plugin in the Eclipse environment.

Figure 28 - Extension points used by the BIRT-plugin

Some of the extensions points are directly provided by Eclipse. These are the extension points, which realize the ODA-interface and a common property page. All extension points, which refer directly to BIRT specific features, are provided by BIRT. The following hooks were used to integrate the map-plugin:

- **...birt.report.designer.ui.reportitemUI**
  - This implementation extends the report item in the BIRT report designer. The connected class implements the IReportItemImageProvider-interface, to display the map in the designer as an SWT-Image.
- **...birt.report.model.reportItemModel**
  - This hook defines the model behind the representation of the map item. The model defines the report item specific properties.
- **...birt.report.engine.reportitemPresentation**
  - The class, which is linked to this extension point, is responsible for the presentation of the map item in a rendered report. At the moment just HTML is generated, this could be extended in future versions.
- **...birt.report.designer.ui.elementAdapters**
  - This extension point connects to the map item as an extended report item
- **. ...datatools.connectivity.oda.dataSource**

  - This, from the Eclipse core provided extension point is responsible for the data source driver. The linked driver class implements the IDriver-interface.
- **...datatools.connectivity.oda.design.ui.dataSource**
  - The implementation of this extension point defines the wizards for the definition of a specific data source and data set.
- **...ui.propertyPages**
  - The implementation of this extension point provides a default data source property page.

Implementation

System test

Goal

This test plan assures the correct behaviour of the SOLAPLayers-BIRT-plugin. The BIRT-plugin works with SOLAPLayers 2.0 Extended, which is described in the first Chapter of this document. The use of SOLAPLayers in the Eclipse- and BIRT-enviroment will be tested. Some presentation-issues are based on the SOLAPLayers map, which was not touched during the bachelor thesis. Please notice the known problems (see chapter "Future improvements – unsolved problems"). This test plan does not test any BIRT core features, just the interatction of the plugin with BIRT.

Requirements and test data

The same test data as in the test of SOLAPLayers 2.0 Extended was used for this test. For further information look at the "System test" of SOLAPLayers 2.0 Extended.

Test plan

This test plan was accomplished on 12 May 2011. The software was tested on a Linux Ubuntu 20.10 with Eclipse IDE for Java and Report Designers in the version Helios Release 2.

Default A to B Scenario

The following test plan contains a default A to B scenario. At each step the precondition is the error-free completition of the previous step. For the initial position of the test a new "Report Project" and inside of this project a new report called "test.rptdesign" has to be created. Use the "Report Design" perspective in Eclipse to go through the test plan.

Number	Test case	Expectation	Result
1.1	Right click on the "Data Sources" icon in the "Data Explorer" view. Choose "New Data Source" in the context menu	"SOLAPLayers Data Source" is displayed as one of the data sources in the list.	Pass
1.2	Select the "SOLAPLLayers Data Source, enter the name "SOLAPLayers Data Source" and click on the "Next" button.	The property window is displayed.	Pass
1.3	Enter the 3A sql-string and the related parameters (see "SOLAPLayers 2.0 Extended system test" for further information"). Click on the "Finish" button.	"SOLAPLayers Data Source" is displayed on the "Data Sources"-list int the "Data Explorer".	Pass
1.4	Right click on the "Data Sets" icon in the "Data Explorer". Choose "New Data Set" in the context menu.	The entered "Data Source" is displayed in the "SOLAPLayers Data Source" list.	Pass
1.5	Choose the just entered data source, enter the name "SOLAPLayers Data Set" and press the "Next"-button.	The "Generated Columns" view is displayed and the three columns "OLAPJSON", "supname" and "nopersons" are listed.	Pass
1.6	Click on the "Finish"-button.	The "Output Columns" view is displayed. The columns are listed.	Pass
1.7	Click on the "Preview Results" link.	The three columns are displayed. The first column contains just the OLAPJSON in the first cell. All other cells of the first column are empty. The other columns contain the right data. Total eleven rows are displayed.	Pass

1.8	Click on the “OK” button	The window closes and a new data set called “SOLAPLayers Data Set” appears in the “Data Sets” list.	Pass
1.9	Drag the “SOLAPLayers Data Set” into the report. Delete the column “OLAPJSON”.	A table is displayed in the report designer.	Pass
1.10	Change to the “Palette” view and drag a map-item into the report. Resize the map to the size of the table.	The map appears in the right size on the report.	Pass
1.11	Select the map item and change the title in the “Property Editor – Map - Properties” view to “SOLAPLayers Map” and the zoom to “1”. Change to the “Binding” tab and choose the “SOLAPLayers Data Set” from the drop-down. Save the report. Click on the “View Report” item an choose “View Report as HTML”.	The report is displayed in a new window. The table shows all the eleven rows and the map is displayed in the right size. The areas on the map are highlighted in different greens. They show the high numbers in light colors.	Pass

Alternative scenarios

Number	Test case	Expectation	Result
2.1	Same scenario using a MDX-query	The generated column-names are composed out of the different measures.	Pass
2.2	Same scenario using a SQL-query with dimensions	The generated column-names are composed out of the different measures.	Pass

Code quality

The project was reviewed automatically by the Eclipse plugins Metrics (http://metrics.sourceforge.net/) and PMD (http://pmd.sourceforge.net/).

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Overridden Methods (avg/max per type)	3	0.176	0.706	3	/datasource/src/org/spatialytic	
Number of Attributes (avg/max per type)	25	1.471	1.719	6	/datasource/src/org/spatialytic	
Number of Children (avg/max per type)	0	0	0	0	/datasource/src/org/spatialytic	
Number of Classes (avg/max per packageFragment)	17	3.4	2.332	7	/datasource/src/org/spatialytic	
Method Lines of Code (avg/max per method)	478	2.779	3.698	23	/datasource/src/org/spatialytic	readFile
Number of Methods (avg/max per type)	171	10.059	10.16	38	/datasource/src/org/spatialytic	
Nested Block Depth (avg/max per method)		1.169	0.55	5	/datasource/src/org/spatialytic	createResultSet
Depth of Inheritance Tree (avg/max per type)		1.765	1.307	5	/datasource/src/org/spatialytic	
Number of Packages	5					
Afferent Coupling (avg/max per packageFragment)		2.6	4.224	11	/datasource/src/org/spatialytic	
Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/datasource/src/org/spatialytic	
McCabe Cyclomatic Complexity (avg/max per method)		1.273	0.755	6	/datasource/src/org/spatialytic	createResultSet
Total Lines of Code	1181					
Instability (avg/max per packageFragment)		0.725	0.374	1	/datasource/src/org/spatialytic	
Number of Parameters (avg/max per method)		0.959	0.911	5	/datasource/src/org/spatialytic	generateHtmlMap
Lack of Cohesion of Methods (avg/max per type)		0.222	0.283	0.778	/datasource/src/org/spatialytic	
Efferent Coupling (avg/max per packageFragment)		3.2	2.561	7	/datasource/src/org/spatialytic	
Number of Static Methods (avg/max per type)	1	0.059	0.235	1	/datasource/src/org/spatialytic	
Normalized Distance (avg/max per packageFragment)		0.275	0.374	1	/datasource/src/org/spatialytic	
Abstractness (avg/max per packageFragment)		0	0	0	/datasource/src/org/spatialytic	
Specialization Index (avg/max per type)		0.029	0.118	0.5	/datasource/src/org/spatialytic	
Weighted methods per Class (avg/max per type)	219	12.882	12.165	41	/datasource/src/org/spatialytic	
Number of Static Attributes (avg/max per type)	33	1.941	2.879	11	/datasource/src/org/spatialytic	

Figure 29 - Metrics report of the final Geoextensions2Birt-plugin

The numbers calculated by Metrics are all in a suitable range. Some code smells are given through the implemented interfaces. Some of the used interfaces have huge amount of methods, which have to be integrated.

Future Improvements

Unsolved problems

Presentation just as basic HTML

For unknown reasons the map is displayed just in the basic HTML view (in BIRT Eclipse plugin). The problem is hard to evaluate, because there is now specific documentation of the way html gets interpreted in the other HTML-based views (preview and web viewer). The problem relays probably on the compatibility of the JavaScript-libraries with the viewers, or is evoked by the pagination-algorithm.

Link to project internal JavaScript resources

Different JavaScript-libraries are necessary to display the map component in the HTML-view. The goal is to access these libraries in the project directory using relative paths. It could not be figured out, how the BIRT-browser addresses JavaScript-links. For this reason, all JavaScript have to be accessible through an URL-address. At the moment the libraries are located in “public_html”, which should be located in the wepapps-directory of the tomcat server (or on another type of server).

Multiple maps for one report

SOLAPLayers client side does not support multiple views in the current version. That is why the BIRT-plugin is not able to display more then one map as well. As soon SOLAPLayer has this feature, the BIRT-plugin has to adapt to SOLAPLayers.

Chores

Different views for MDX- and SQL-query
Create a different view for entering MDX and SQL-queries. This would improve the usability, since MDX does not need the same amount of parameters.

Separate data source- from data set-parameters
At the moment, all parameters for accessing a data source are entered in the data source property window. In the future just the connection information appears and the query as well as the parameters will be entered in the properties of the data set.

Error Handling
SOLAPLayers has a complete error handling implemented. The BIRT-plugin should adapt this and generate useful error message, which can be displayed in the front-end. This is necessary especially for validation and connection reasons.

Reuse SOLAPLayers libraries
SOLAPLayers is integrated in the BIRT-plugin as a library. This library contains other jars. The BIRT-plugin needs to access these libraries. There is no possibility to access these sub-libraries in Eclipse at the moment. This version of the plugin contains some libraries twice. They increase the file-size and could lead to problems with the compatibility (when using two different library versions).

Localization
At the moment just an English bundle exists and the localization is ignored. This should be fixed to provide the plugin in different languages.

Align the map in the center
The map is displayed without to align the important data in the middle of the map representation. The user has to find the highlighted polygons, line or points by them self. This could be solved programmatically.

Extensions

Support other output formats
SOLAPLayers 2.0 Extended supports no other output than the JSON-formats. As soon SOLAPLayers support other formats other report-formats can be provided as well. Especially the output of a map as an image is important, this would allow to create pdf, Excel, PowerPoint and other useful report-formats.

User Guide

Installation
The installation was tested on a Linux Ubuntu 20.10 system. The computer has to be connected to the Internet to use the test databases.

- Copy the “Geoextensions2Birt/datasource_1.0.0.jar” into the “plugins” directory of your Eclipse installation.
- Copy the “Geoextensions2Birt/Utils/public_html”-directory into the webapps directory of your tomcat and run tomcat. The “public_html”-directory has to be accessible using the URL http://127.0.0.1/public_html/ (For further information see “Unsolved problems - Link to project internal JavaScript resources)
- Copy the sources.xml to the root directory (“/”) of the computer.

- Start Eclipse using the console and the argument "-clean" to make sure that the plugin will be loaded.

Tutorial – Adding a map item to a report

1. After installing the BIRT plugin, open BIRT and change to the "Report Design" perspective. Use "File – New – Project…" to create a new "Report Project". Enter a name and go through the project wizard.

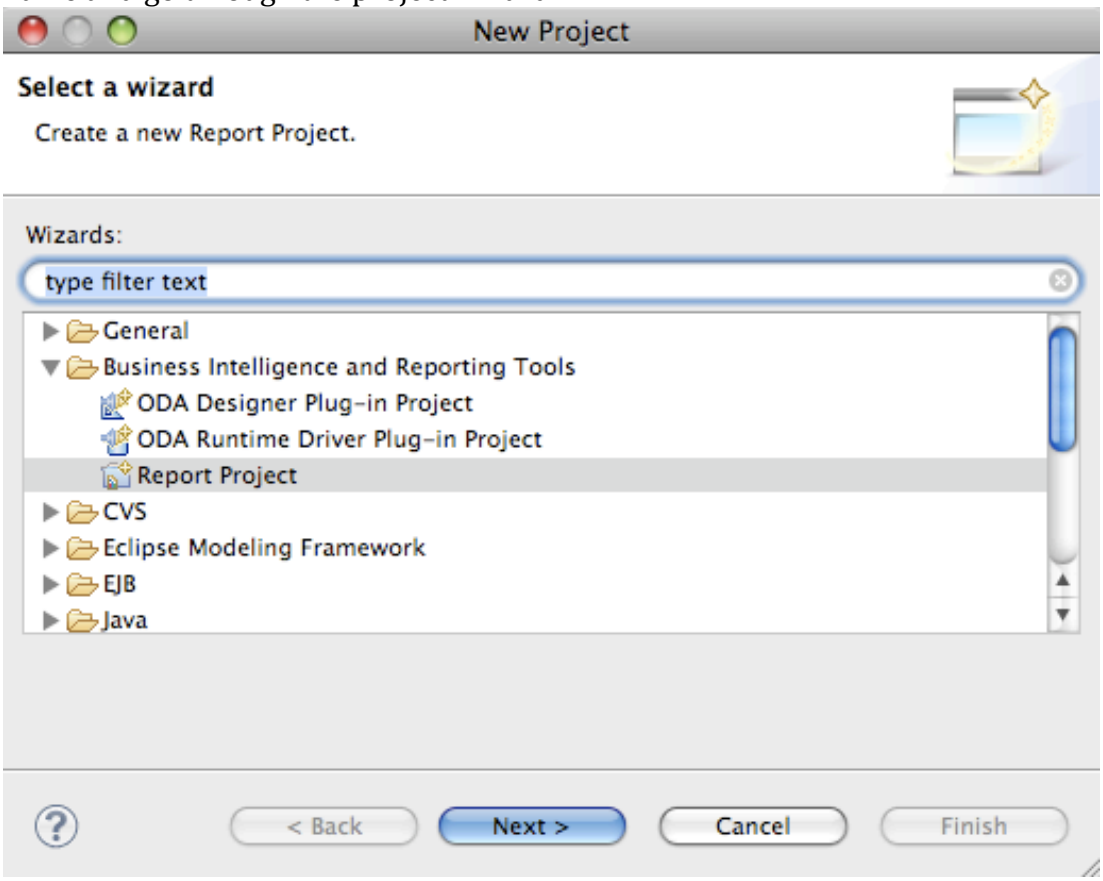


Figure 30 - Print screen: Create a Report Project

2. Execute a right-click on the new project in the "Navigator"-view and click on "New – Report". Enter a name and finish the wizard.

3. The next step is to define the data source. Open the new report and right-click on the "Data Sources"-icon in the "Data Explorer". Choose "New Data Source". A Window pops up.

4. Chose the “SOLAPLayers Data Source” and click on the “Next”-button.

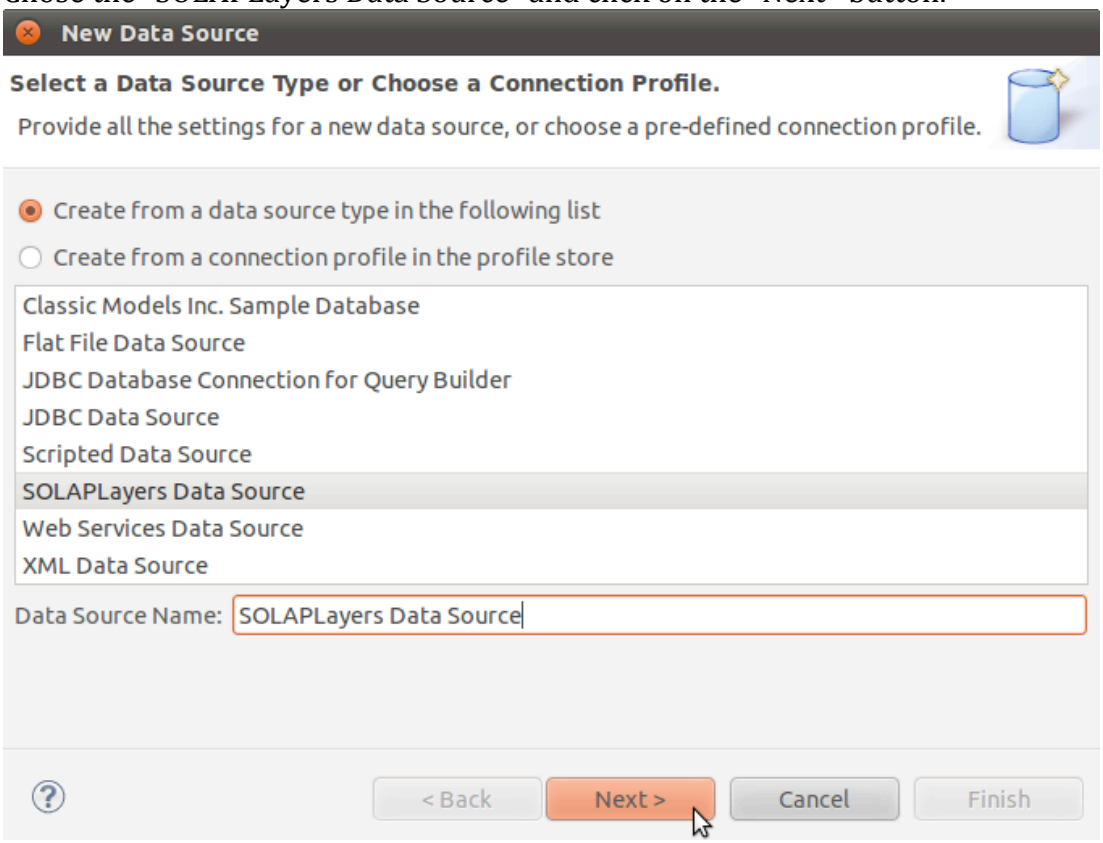


Figure 31 - Print screen: Select a data source type

5. Enter the SQL or MDX query and the correct parameters. For further information about the parameters and the format of the query have a look at the “SOLAPLayers 2.0 Extended – Developers guide” – chapter. Click on “OK”. You can see now the new data source in the “Data Explorer”.

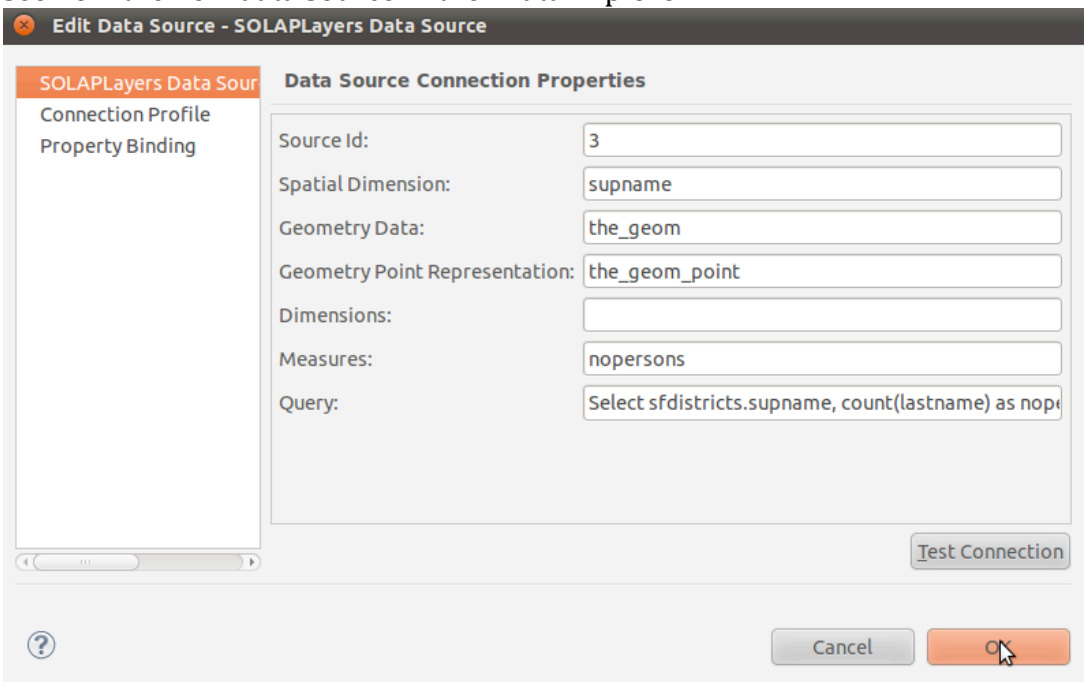


Figure 32 - Print screen: Fill out the data source parameter form

6. Perform a right-click on the “Data Sets”-icon, which is displayed as well in the “Data Explorer”. Choose “New Data Set”. A pop-up window appears.

7. Choose the SOLAPLayers data source and enter a name for the data set. Click on “Next” to continue the wizard.

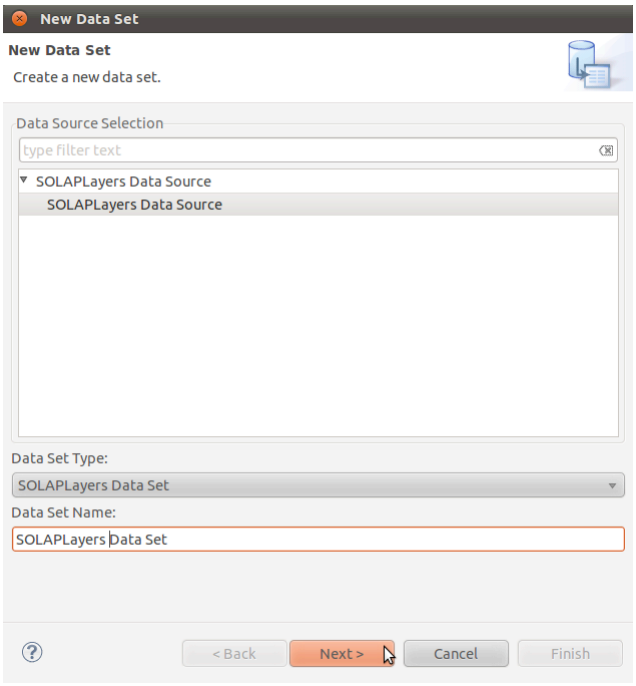


Figure 33 - Print screen: Choose the data source

8. For the two-dimensional representation of MDX-results and SQL-results, with dimensions as parameters, the columns are not obvious in the query. That is why you can see on this page of the wizard the generated column-names. Click on the "Finish" button.

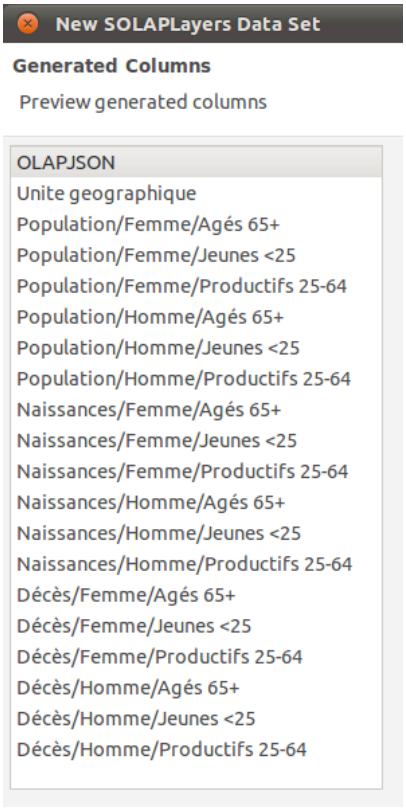


Figure 34 - Print screen: Generated column names

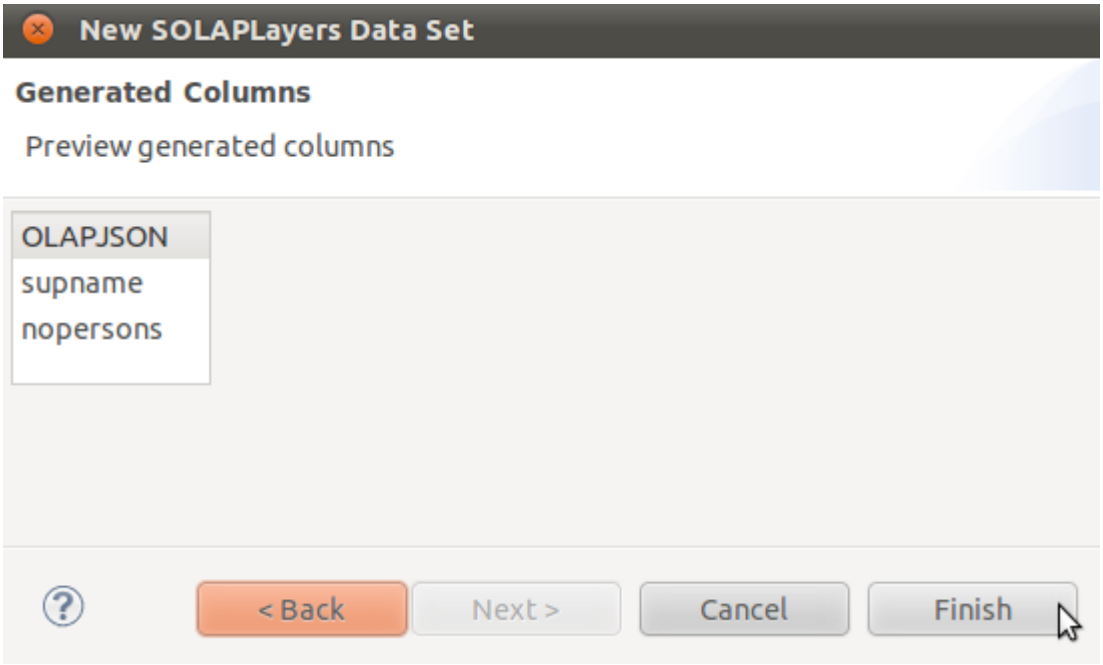


Figure 35 - Print screen: Preview column names

9. Change in the appearing window to the page "Preview Results" to have an overview of the data. Ignore the first coloumn, it will be hide in a further version

of Geoextensions2Birt. Click on “OK” to complete the data set creation.

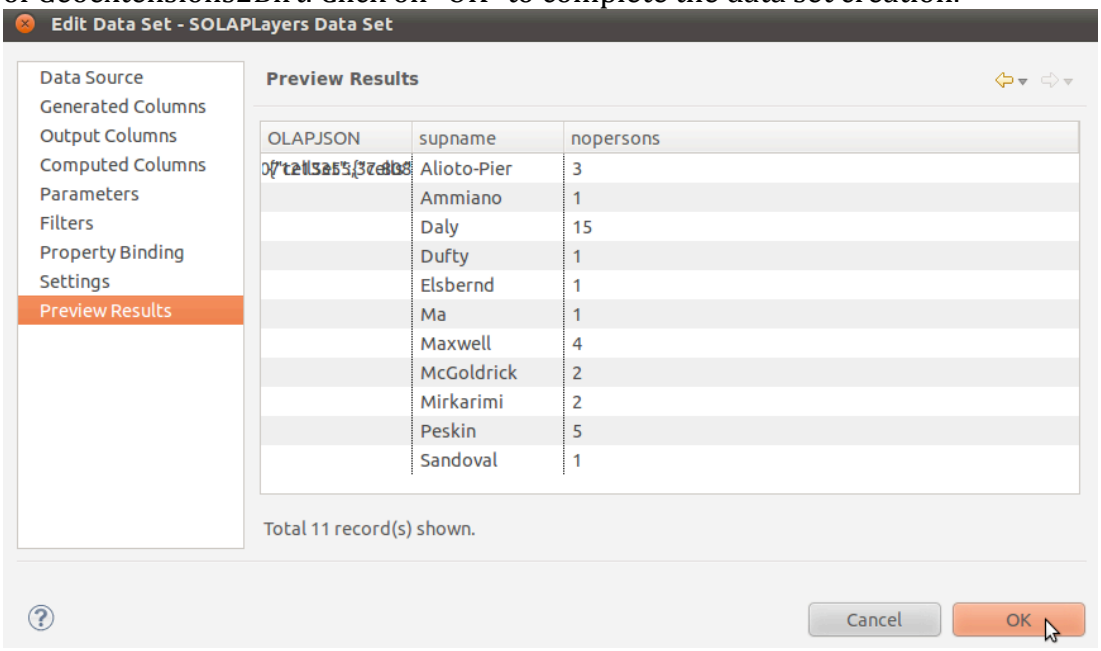


Figure 36 - Print screen: Preview result set

10. You can drag and drop the data set on the report. Please delete the first column “OLAPJSON”.

11. Change to the “Palette” view to drag and drop the map report item on the report.

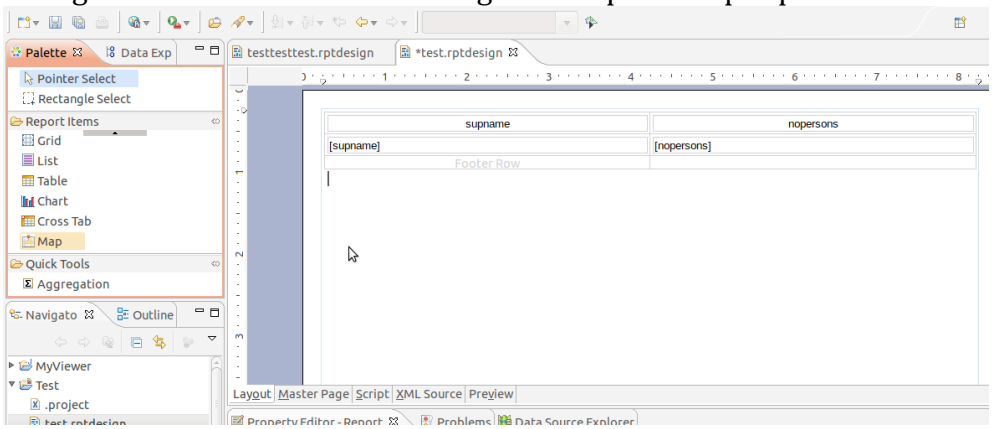


Figure 37 - Print screen: Drag and drop map report item

12. You can now resize the map item and enter the name and the zoom level using the “Property Editor – Map”-view.

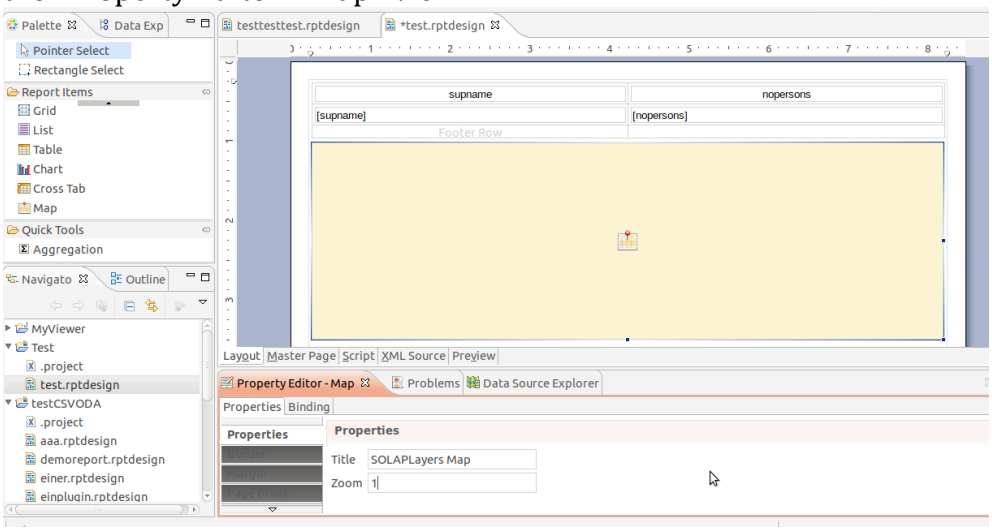


Figure 38 - Print screen: Define properties for map report item

13. To view the final report click on the “View Report”-icon and choose “View Report as HTML”. Notice that the preview and other formats are not working in this version.

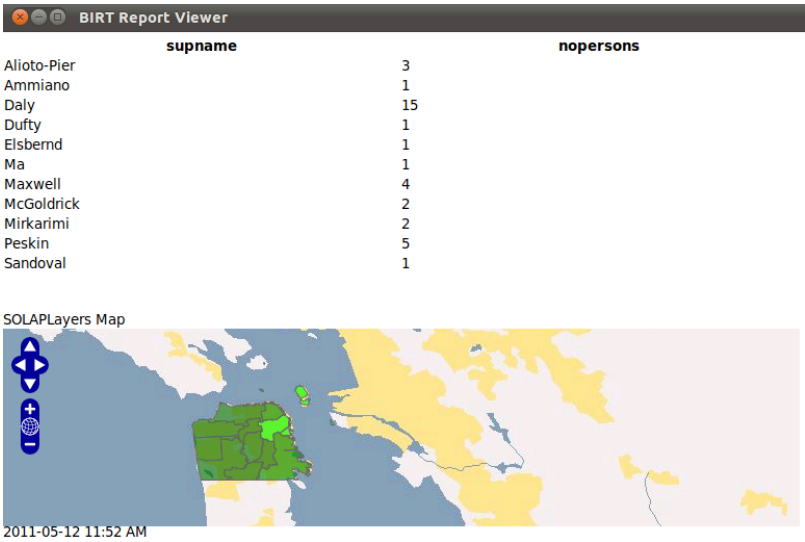


Figure 39 - Print screen: Report as HTML-output

Developers guide

Export Plugin

- Right-click on the project
- Click on “Export…”
- Choose “Plug-in Development – Deployable plug-ins and fragments”
- Define a directory and click on “Finish”

Plugin Installation

The installation was tested on a Linux Ubuntu 20.10 system. The computer has to be connected to the Internet to use the test databases.

- Import Geoextensions2birt-project code using the import-functionality of Eclipse “Existing Projects into Workspace”. The zip-file is located on the project CD (“Geoextensions2Birt/geoextensions2birt.zip”).
- Copy the “Geoextensions2Birt/Utils/public_html”-directory into the webapps directory of your tomcat and run tomcat. The “public_html”-directory has to be accessible using the URL http://127.0.0.1/public_html/ (For further information see “Unsolved problems - Link to project internal JavaScript resources)
- Specify the path to the source.xml (SOURCES_XML) in the “util/messages.properties”-file.
- Run project as “Eclipse Application”. A new Workspace opens and the plugin will be loaded.

8 Project management

The process of the two subprojects is documented in the appendix. ("APPENDIX E – Project management") For more information about the chosen project management method, have a look at "APPENDIX I – Customized SCRUM-process for bachelor thesis"."

9 Project review

9.1 Acknowledgment

I would like to thank to Prof. Stefan Keller for the big support before and during the bachelor thesis. This project could not proceed so unproblematic without the good assistance of him.
A big thanks goes to Spatialytics, namely Dr. Thierry Badard, Luc Vaillancourt and Jean Mathieu, for their effort to integrate me into there company, the good team work and the assistance. I wish you and Spatialytics just the best for the future challenges.

9.2 Field report

Im Anhang dieses Dokumentes befindet sich der Zwischenbericht der in der Halbzeit des Projekts erfasst wurde (siehe „APPENDIX J – Interim report"). Hier nochmals ein kurzer Rückblick über das gesamt Projekt:

Die Bachelorarbeit, die ich bei dem Unternehmen Spatialytics in Québec, Canada erarbeitet habe, brachte viele Herausforderungen mit sich. Mit meinem Vorhaben, die BA im Ausland zu schreiben, bin ich diese Challenge bewusst eingegangen. Auch im Nachhinein bin ich sehr glücklich, dass ich mich dieser Aufgabe gestellt habe. Ich konnte während diesen zweieinhalb Monaten von vielen verschiedenen Aspekten profitieren.

Die Arbeit bei Spatialytics war sehr angenehm. Das eingespielte Team hat mich sehr freundlich empfangen und mich schnell in die Firma eingegliedert. Spatialytics ist ein sehr motiviertes Start-Up Unternehmen, das mit viel Elan und Motivation die Aufgaben angeht. Der Fakt das Spatialytics erst in den Kinderschuhen steht, beeinflusste meine Arbeit stark. Es war leider nicht möglich ein total eigenständiges Projekt durchzuführen, was im Sinne einer Bachelorarbeit sehr zu schätzen wäre. Zu viele technische aber auch organisatorische Abhängigkeiten waren durch die Umstände vorhanden.

Eines dieser Probleme war, dass sich die Applikation SOLAPLayers zu Beginn der BA nicht auf einem Qualitätsniveau war, welches die HSR für eine BA akzeptieren würde. Dieser Fakt brachte mich bereits früh in einen gewissen Zwiespalt, da ich mit meiner Arbeit einerseits Spatialytics und andererseits die für die Bewertung verantwortlichen Personen zufrieden stellen wollte. Dies zu vollbringen erforderte viel Mehraufwand und eine gute Koordination. Da der zeitliche Rahmen es nicht zugelassen hat, dass ich mich zu fest in die bereits vorhandenen Umstände einmischte, musste ich mir Einschränkungen machen. So akzeptierte ich gewisse Entscheidungen, ohne den Sinn und Zweck zu hinterfragen.

Die organisatorische Herausforderung hatte zur Folge, dass der technische Aspekt etwas zu kurz kam. Leider konnte ich mich mit meiner BA nicht wie gewünscht profilieren und eine technische Arbeit mit dem gewünschten Anspruch entwickeln. Trotzdem, das erzielte Resultat ist in meinen Augen technisch sauber umgesetzt und verständlich und ausführlich dokumentiert. Das Hauptziel war die Zufriedenstellung meiner Betreuer. Spatialytics hat mehrmals ihre Dankbarkeit ausgesprochen und mich somit bestätigt. Ich freue mich, dass sie von meiner Arbeit profitieren konnten.

Die Zeit in Québec war definitiv eine prägende Zeit. Nicht nur durch die Arbeit, auch durch das Leben neben der BA konnte ich viel profitieren. So konnte ich zum Beispiel meine Englisch-Kenntnisse verbessern und meinen kulturellen Horizont erweitern. Nebenbei hatte ich eine super Zeit, mit vielen netten Begegnungen und einem super Umfeld. Die etwas schwierigeren Umstände waren eine super Vorbereitung auf das kommende Berufsleben, wo ich auch auf ähnliche Situationen treffen werde.

Herzlichen Dank an alle, die mich vor und während meiner Zeit in Québec unterstützt haben!

9.3 Lessons learned

Die errungenen Erkenntnisse aus dieser Arbeit sind in den wöchentlichen Sprint-Berichten dokumentiert. Diese können im Anhang E oder auf der Projekt CD im Dokument “project plan.xls” eingesehen werden. Weitere Informationen zum Verlauf der Arbeit können auch dem Schlussbericht („field report“) und dem Zwischenbericht („interim report“) entnommen werden.

10 Further documentation

Additionaly to this document a JavaDoc documentation for each project exists. The JavaDoc documents all classes briefly and important public methods more detailed. Overwritten methods are documented in the implemented interface or the extended superclass. Simple getter and setter methods as well as private-methods are not documented.

11 Appendices

APPENDIX A - Content of the CD

- **Bachelor Thesis**
  - Contains the final project documentation, the project plan, all presentations, which were held during the internship and the JavaDoc-documentation for both sub-projects. The project poster can be found in this directory too.
- **Geoextensions2Birt**
  - Contains the source code of the BIRT-plugin, the deployed plugin and some utilities, which are necessary to install and use Geoextensions2Birt.
- **SOLAPLayers 2.0 Extended**
  - Contains the source code of SOLAPLayers 2.0 Extended as Eclipse-project and as deployed java library, and utilities, which are necessary to run SOLAPLayers 2.0 Extended.
- **Utilities**
  - Contains files, which are useful for both sub-projects, beside all the documentation.

APPENDIX B - Glossary

This is a glossary about used terms and definitions. To understand the topic it is required to engage with the topics of BIs and spatial information and systems.

Term	Explanation
BI-Tools	Business Intelligence-Systems are tools, which allows analysing business data and creating reports. SAP is the most common commercial tool. There are a lot other competing commercial- and open source software products like Oracle BI, Pentaho BI Suite, Baan etc.
BIRT	BIRT is an Eclipse-based reporting tool. Easy to integrate in to other Java/Java EE application to create compelling reports. See http://www.eclipse.org/birt/phoenix/.
BIRT ROM	BIRT Report Object Model is used to create and save reports. The BIRT Object Model is described by an xml-file. See http://www.eclipse.org/birt/phoenix/ref/rom/index.html.
Data Mining	Data Mining describes the process of finding pattern in data sets. A similar name for Data Ming is „Knowledge discovery in Databases“.
Data-Warehouse	Contains data from different sources. The provided data get pushed into the warehouse through ETL. The created data-warehouse is used for data analysis and to make business decisions. In a data warehouse data gets never deleted and a lot of information is redundant. The redundancy increases the performance.
DBS	Data Base Systems save data efficiently, consistent and durable and provide it to user and systems.
Eclipse RCP	Eclipse Rich Client Platform is a platform for creating and deploying desktop rich client software. Eclipse is based on an OSGi-framework

	called Equinox. See http://wiki.eclipse.org/index.php/Rich_Client_Platform
ETL	ETL is the abbreviation for Extract, Transform, and Load: • Extract data out of different data sources. • Transform data to fit in the target database. • Load the transformed data in to the database.
GEF	The Graphical Editing Framework is a helpful framework to extend the Eclipse Workbench. It affords to extend Eclipse with SWT-based tree and Draw2d-based graphical editors. See http://www.eclipse.org/gef/.
GeoKettle	A Kettle extension of Spatialytics to extend the ETL-product for spatial data. See http://www.spatialytics.org/projects/geokettle/.
GeoMondrian	GeoMondrian is a project of Spatialytics to extend the standard Mondrian product. This SOLAP-Server provides integration of spatial objects into the OLAP data cube structure. SOLAP does not need an additional database for the geographic data. See http://www.spatialytics.org/projects/geomondrian/.
GIS	Geographic Information Systems capture, store, manage, analyze and present data, which is linked to location/s. The complex nature of the spatial data requires such information systems.
Jasper BI Suite	An Open source OSBI tool. Fully developed in Java. Since 2007 completes JasperETL the OSBI-solution. See http://www.jaspersoft.com.
Jedox	The worldwide leading Provider of OSBI-systems. Jedox is located in Freiburg, Germany. A well-known product is the Palo-suite. See http://www.jedox.com/de/home/uebersicht.html.
Kettle	Kettle Is the Pentaho Data Integration tool (ETL). See http://kettle.pentaho.com/.
MDX	Multidimensional Expressions is a query language for OLAP databases. MDX allows querying and modifying multidimensional data stored in OLAP cubes.
Mondrian	A statistical data-visualization system of Pentaho. It allows the visualization of almost any kind of data. It strength is to work with large data. Mondrian is an OLAP-Server. See http://mondrian.pentaho.com/.
ODA	ODA is the shortcut for Open Data Access. ODA describes a standard for a data representation as a two-dimensional result set. This standard is mainly used by BIRT and other Eclipse projects.
OLAP	OLAP is the abbreviation for Online Analytical Processing. OLAP is a method to send complex, multidimensional analytically queries to the database (MDX). The answer of the query is typically supplied as a matrix.
OLAP cube	OLAB cube is a method to present logical data. The data are arranged three- or multi-dimensional. The data can be chosen through one or more axes.
OLAP4j	OLAP4J is a JDBC acronym for accessing different OLAP server. It allows changing the used OLAP server rapidly and without changing

	any source code. It is an open source product. See http://www.olap4j.org/.
OLTP	Online-Transaction-Processing labels a paradigm for database systems. The systems uses database transactions and responses immediately to user requests.
OSBI	Open Source Business Intelligence products are the opponents to the more common commercial products like SAP, Oracle BI, Baan etc. Well-known OSBIs tools are by Talend, SpagoBI, Palo, Jasper BI Suite and Pentaho BI Suite.
Palo OLAP Server	Palo is a multidimensional real time OLAP-server (MOLPA). It is used to manage economically and statistically data sets. It can be integrated through different software environments (Java, PGP; C, .NET etc.). The Jedox company develops Palo OLAP Server. See http://www.jedox.com/de/produkte/palo-suite/palo-olap-server.html.
Pentaho BI Suite	A java-based OSBI tool of the Pentaho Company. Pentaho has ETL, reporting, OLAP/analysis and data-Mining functionality. Developed since 2004. See http://www.pentaho.com.
Snowflake	A schema used for OLAP and data warehouses. It normalizes the dimensions-table of the start schema and provides in this way faster repeated requests on large dimension tables and saving memory capacity.
SOLAP	SOLAP (Spatial On-Line Analytical Processing) extends OLAP. It allows to explorer additional spatial information provided by Geographic Information Systems (GIS).
SOLAPLayers	SOLAPLayers is a lightweight web cartographic component, which enables navigation in SOLAP data cubes. It aims to be integrated into existing dashboard frameworks in order to produce interactive geo-analytical dashboards. See http://www.spatialytics.org/projects/solaplayers/.
SpagoBI	In Java written OSBI. Covers the full range of analytic-tools. Developed by the OW2 Consortium. See www.spagobi.ow2.org.
Star	A schema used for OLAP and data warehouses. Can be described as a denormalized database, which holds duplicated data to provide a fast access for the prove of analysing and processing of large amount of data. A star has always a fact and different dimension tables The fact table contains the data and the dimensions limit and extend the specific view on the data.

APPENDIX C - References

Literature

Source

- Boris Gloger „SCRUM – Produkte zuverlässig und schnell entwickeln“, © 2008 Carl Hanser Verlag München Wien, ISBN 978-3-446-41495-2

- Jason Weathersby, Tom Bondur, Iana Chatalbasheva, Don French "Integrating and Extending BIRT", © 2008 Addison Wesley, ISBN 978-0-321-58030-6

Internet

Type	Source	Date
Image	http://www.infobarrel.com/A_Quick_Scrum_Tutorial	02.03.2011
Image	http://training.inet.com/OLAP/Images/kube0006.gif	01.05.2011
Image	Presentation of Thierry Badard "Dashboard and reporting tools: Integration of BI tools in the geospatial domain"	06.05.2011
Image	Presentation of Thierry Badard "Spatialytics", 2010	06.05.2011

APPENDIX D - Table of figures

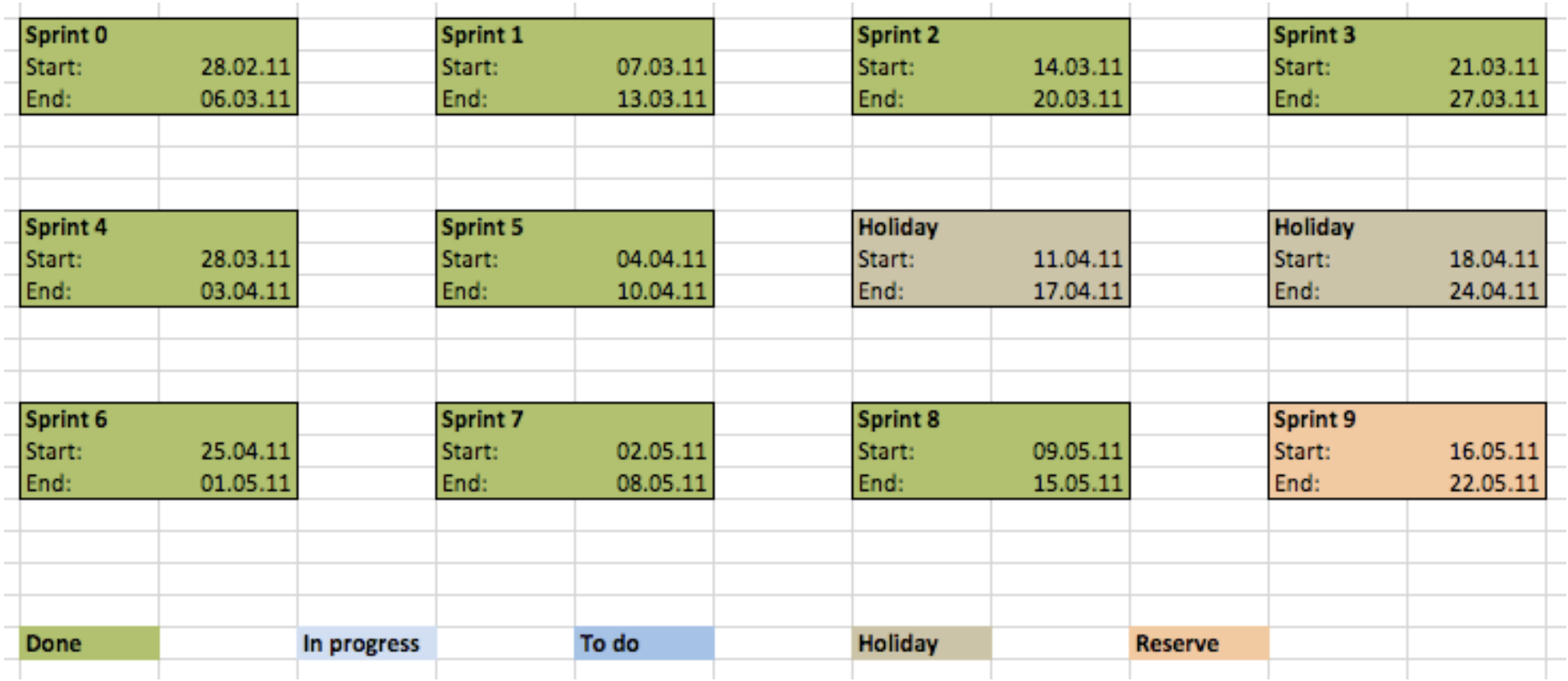
Figure 1 - Logo von populären OSBI 3
Figure 2 - Logo des Unternehmes Spatialytics 3
Figure 3 - SOLAPLayers Demo Dashboard 4
Figure 4 - BIRT Report mit SOLAPLayers-Karte 5
Figure 5 - SOLAPLayers logo 13
Figure 6 - Visualization of an OLAP-cube 16
Figure 7 - Overview SOLAPLayers 2.0 Extended 18
Figure 8 - Overview vision SOLAPLayers 2.0 Extended 19
Figure 9 - Activity diagram 20
Figure 10 - Package diagram 21
Figure 11 - Class diagram 22
Figure 12 - Metrics report of SOLAPLayers 2.0 Extended 29
Figure 13 - OuputBuilderFactory class, which will be replaced in a future version 34
Figure 14 - Sample of sources.xml with different data sources 35
Figure 16 - Result of sample MDX query displayed as cross table 36
Figure 15 - Sample MDX query for SOLAPLayers 36
Figure 17 - Sample cross table in SOLAPLayers dashboard with dimensions civil status and sex 38
Figure 18 - Sample cross table in SOLAPLayers dashboard with dimensions sex and civil status 38
Figure 19 - Sample cross table in SOLAPLayers dashboard with the measures number of persons, total savings 38
Figure 20 - Sample cross table in SOLAPLayers dashboard with the measures total savings, number of persons 39
Figure 21 - Sample SQL query for SOLAPLayers 39
Figure 22 - Result of sample SQL displayed as table 39
Figure 23 - SOLAPLayers sample dashboard 40
Figure 24 – Illustration: SOLAPLayers with JSON-output and two connections 43
Figure 25 – Illustration: SOLAPLayers with Image-output and two connections 44
Figure 26 – Illustration: SOLAPLayers and one connection 45
Figure 27 – One connection without SOLAPLayers 46
Figure 28 - Extension points used by the BIRT-plugin 47
Figure 29 - Metrics report of the final Geoextensions2Birt-plugin 51
Figure 30 - Print screen: Create a Report Project 53
Figure 31 - Print screen: Select a data source type 54
Figure 32 - Print screen: Fill out the data source parameter form 54

Figure 33 - Print screen: Choose the data source 55
Figure 34 - Print screen: Generated column names 56
Figure 35 - Print screen: Preview column names 56
Figure 36 - Print screen: Preview result set 57
Figure 37 - Print screen: Drag and drop map report item 57
Figure 38 - Print screen: Define properties for map report item 58
Figure 39 - Print screen: Report as HTML-output 58
Figure 40 - Introduction of creating a SOLAPLayers dashboard – Step 1 82
Figure 41 - Introduction of creating a SOLAPLayers dashboard – Step 2 83
Figure 42 - Introduction of creating a SOLAPLayers dashboard – Step 3 83
Figure 43 - SCRUM Process 84

APPENDIX E – Project management

The following illustrations show the project plan and the project progress. Detailed information can be found on the CD in the document "project plan.xls".

Scheduling



Task board

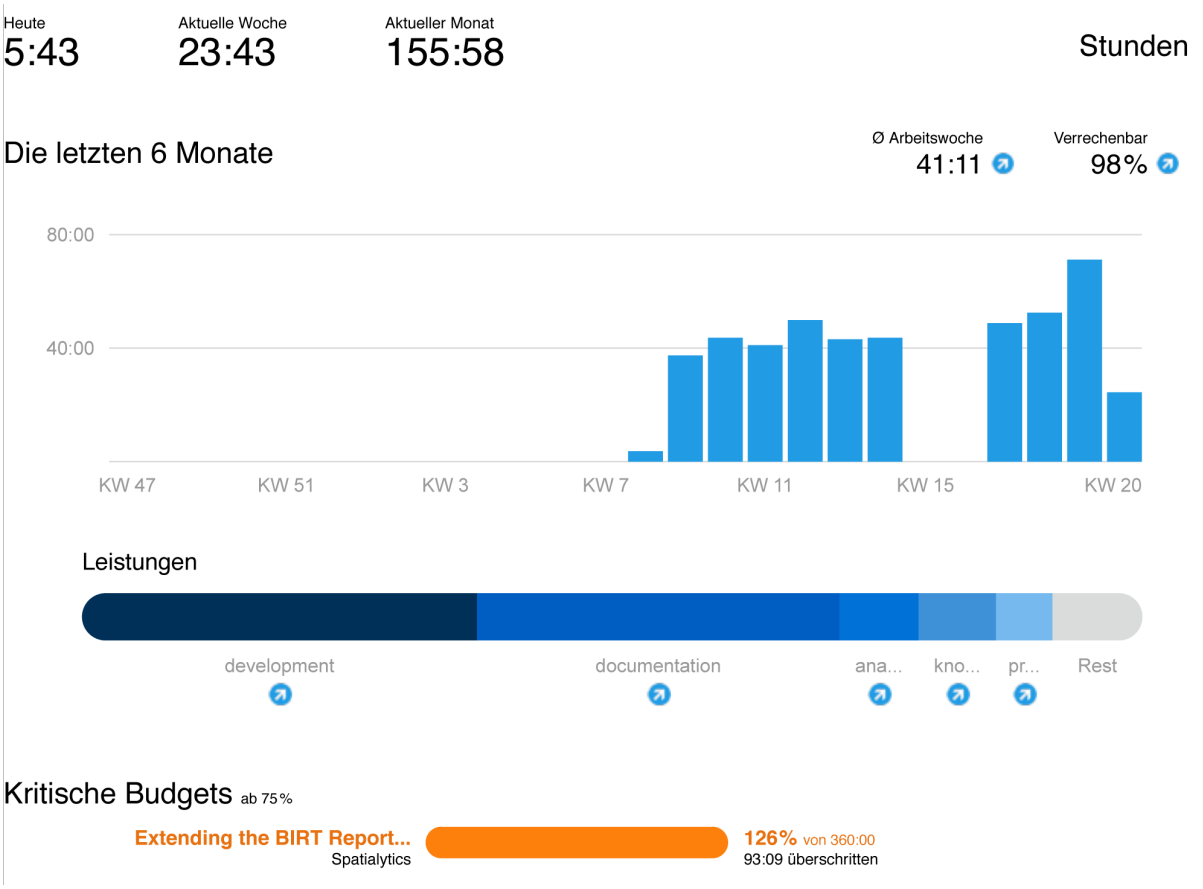
Task	Priority	Story Points	Task	Description	Category	State
1	30	4	Flexible architecture for SOLAPLayers data sources	Analyse and develop an architecture for SOLAPLayers, which allows a flexible choose of the data source. The data source can be an OLAP- or a SOLAP-server, a simple DB, flat file like XML, Excel, etc. The client sends depending on the data source a query and additional information to get the desired data. At the moment just SOLAP-data is provided.	Extending SOLAPLayer	Done
2	15	3	Flexible architecture for SOLAPLayers output formats	Analyse and develop architecture for SOLAPLayers, which allows a flexible output-format. At the moment just a couple of different JSON-formats are provided. At the future, formats like images, KLM, etc. should be provided.	Extending SOLAPLayer	Done
3	15	5	Possibility of DB as a SOLAPLayer data source	Include a strategy into SOLAPLayers to access data from a database through an SQL-query.	Extending SOLAPLayer	Done
5	20	4	Get confident with Birt, BI and spatial information	For the future work it is necessary to know how to work with Birt. As well it is a must to know the different technologies used by Spatialytics.	Knowledge accumulation	Done
6	20	2	Create a sample extension for Birt	Create a small sample extension for Birt, in order to get confident with the Birt environment and figure out the possibilities of extending Birt.	Knowledge accumulation	Done

7	20	1	Analyze existing map solutions for Birt	Have a look at the two existing map solutions for Birt and get the dis-/advantages. This helps to improve the quality of the extension, which has to be deployed.	Knowledge accumulation	Done
8	30	4	Extend the "SQL-result set to OLAPJson"-process	This task should add functionality to the transform-process to allow more complex SQL-queries. The goal is to create a cross table between a location and one or more dimension. Different measures are possible.	Extending SOLAPLayer	Done
9	10	2	Document the new SOLAPLayer architecture	Create a documentation of the new SOLAPLayers server architecture. The principal topic is how to extend the existing solution.	Documentation	Done
10	10	2	Document how to use SOLAPLayers for a dashboard	Create a simple documentation for using the dashboard. This may be a part of the architecture document.	Documentation	Done
11	25	3	Analyze BIRT and find a solution for integrating a map component	Find a way to integrate a map into BIRT. Search different solutions and figure out which is the one to go.	Extending BIRT	Done
12	10	3	Document analysis of Birt-plugin for future traceability	Show on which facts the final decision is based.	Extending BIRT	Done
13	30	3	Create data source plugin with dummy data	Create as fast as possible a solution to connect to an individual data source and to retrieve data in the correct way. To reach as fast as possible an A to B scenario, the data will be static for the moment. The received data has to be in the right format, to get displayed by a report item.	Extending BIRT	Done

14	30	3	Create a map report item plugin with displays static content	Extend the report item palette with a map-component, which shows a map with default, static data.	Extending BIRT	Done
15	25	3	Retrieve dynamic data using a default SQL-query and SOLAPLayers and display it with existing report items	The retrieved OLAPJson has to be converted into a result set representation. All the core report items handle data in a two-dimensional-format	Extending BIRT	Done
16	20	2	Retrieve dynamic data using a default SQL-query and SOLAPLayers and display it with the new map item	For representing the map the OLAPJson has to be forwarded to the report item.	Extending BIRT	Done
17	15	3	Create GUI and the logic behind for using customizable SQL-queries	Replace the hard coded SQL-query and allow the user to define his own query with the according parameters using a GUI.	Extending BIRT	Done
18	15	3	Create GUI for specify the map report item with customizable parameters	The displayed SOLAPLayers-map should be customizable using a GUI.	Extending BIRT	Done
19	10	6	Write a documentation about the state of the art of reporting tools	Write a documentation about the state of the art of reporting tools	Documentation	Done

Timetable

The following report shows the time overview on the 18.05.11. Some more working time will be needed for corrections. For detailed information about the used time have a look at http://sueess.mite.yo.lk (the user- and password-data were send to the supervisor) or at the file "Bachelor Thesis/time-entries.xls" on the project CD.



Sprint 0 (28.02. - 06.03.11)

	Tasks	Status	Story points
5	Create a sample extension for Birt	done	2
6	Analyze existing map solutions for Birt	done	1
7	Extend the "SQL-result set to OLAPJson"-process	done	4

	Status	
	to do	0
	in progress	0
	done	7
	number of total story points	**7**

Log book

Date	*Description*
01.03.10	Removing the task one "Flexible architecture for SOLAPLayers data sources" from the sprint task board. It does not exist a proper release of the base product.

Impediment backlog

Problem

> It was not possible to finish task one "Flexible Architecture for SOLAPLayers data sources". There is not yet a version of SOLAPLayers available, that could get extended properly. There will be a right version soon. A first draft of the analysis could be finished.

Retrospective

> The initial sprint has shown, that the main organizational challenge is to realize the project with all the dependencies to the surrounding environment. (organizational and technologic dependencies).

> Another point is the documentation of Birt. There not allot opensource extensions for Birt available at the moment. The official tutorial for expending Birt had failures, which avoided to run the solution on Mac OSX.

> Finally I saw the advantages and problems of working on the bachelor thesis at the stretch, without having lectures. On the one hand there is a big advantage, the motivation, the knowledge, the flow is not disturbed by the breaks. On the other hand the breaks let space to get work done, which is dependent on people there not directly related to the project.

Sprint review

I learned a lot the during the first week. I am now able to understand the relations between Spatialytics and the his working environment. I learned about BI-systems and especially about Geo-BI.

In a second step I worked into the OSBI-tool BIRT. I learned how to create reports and use different data sources. Furthermore, I was looking at the API and the possibility to extend the product. I also looked around for existing map-solutions and compared this implementations.

In the part I could not finish, I analysed the existing SOLAPLayers Server part to extend it. I created a draft solution for the new architecture.

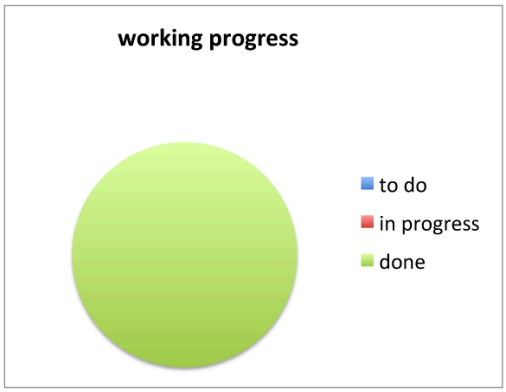
Documents

> glossary (initial)
> scrum
> project plan
> birt presentation
> knowledge summaries (Birt, Eclipse, Birt map extensions)

Sprint 1 (07.03. - 13.03.11)

	Tasks	Status	Story points
1	Flexible architecture for SOLAPLayers data sources	done	4
2	Flexible architecture for SOLAPLayers output formats	done	3

	Status	
	to do	0
	in progress	0
	done	7
	number of total story points	**7**



Log book

Date	*Description*
07.03.10	Not able to start with the implementation of SOLAPLayers. Working version still not available.
08.03.10	Running SOLAPLayers version available. Start Sprint 1.
10.03.10	Finished tasks of Sprint 1. Continue with refactoring of SOLAPLayers
11.03.10	Problem with geo data-format in pg-database. Waiting for solution (Thierry and Jean forcing the problem)

Impediment backlog

Problem

> The installation of PostGIS on the OS X affords a lot of problems. Undocumented errors were raised and just old versions of Postgre und PostGIS available as Fink-installer. Because of the problems with Birt on OS X, the coming development will happen on a Linux (Ubuntu) computer.

Retrospective

> The first official sprint (no preparation) was really interesting and efficient. There was a lot to do and not many problems, which speeds up the project-progress.

> The existing source code of SOLAPLayers-server was really bad structured. It needs a lot of time to understand the logic behind the code (and it will need some more time for special parts). The methods and classes were to big and the code has to many comments.

Sprint review

The initial version of SOLAPLayers (server side), was an application with four classes: A lot of business logic bundled in a few classes. There was one straight flow, without any hooks, which could be used for extend the application.

During the sprint different solution got isolated in new smaller classes. A bunch of new interfaces define the extending point. Now it is possible to add data sources and new output formats easily.

To add a new data source-type, a class has to implement the Connection-interface and overwrite all methods. In the new configuration xml, called source.xml, has to be a reference to the new data source-type and a list of all parameters for the connection. The data source id (defined in sources.xml) has to be a part of the request.

The output format has to be add to the request as in the older versions. The big improvement of the new version, is the flexibility between output format and data source. There is no more hardcoded relation between this two components. Every data source can provide data in any output format.

The result will be presented to the responsible persons, after finishing the second Sprint. Changes and improvements are possible.

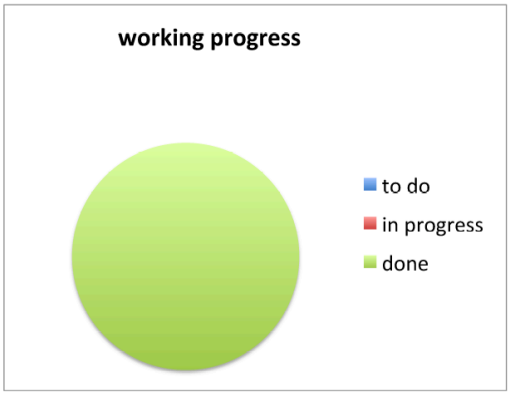
Documents

analysis and documentation (initial)

Sprint 2 (14.03. - 20.03.11)

	Tasks	Status	Story points
3	Possibility of DB as a SOLAPLayer data source	done	5

	Status	
	to do	0
	in progress	0
	done	5
	number of total story points	**5**



Log book

Date	*Description*
15.03.11	Day off. Meeting with Paul Ramsey (PostGIS) in Montreal.
16.03.11	Solved database problem

Impediment backlog

Problem

> The database problem was solved by Thierry and Jean Mathieu. The data format of the spatial-data had no SRID. Since the SOLAPLayer-client-part does just interpret a defined SRID correct, I had to apply a correct SRID to the database and execute a transformation in the sql query.

Retrospective

> As a geo-greenhorn I was a challenged to work with spatial databases. The simplicity of POSTGis and the good support of my collaborators helped me, to find a fast entry into the world of geo-spatial-data.

> As new as geo-spatial-data was the topic of OLAP-Cubes and MDX-queries. Even I had contact with this themes in the last weeks, I still had to get deeper into it. The bigger problem was to understand the OLAPJson, which is completely undocumented.

> A big challenge was (and still is) to write professional source code. The SQL-result set to cube transformation is a job, which needs a lot of not-selfdescribed methods. It is really easy to end with a messy, unstructured code.

Sprint review

A new data source-driver was added to the SOLAPLayer-server-part. It is now possible to use a relational database. SOLAPLayer creates a connection using JDBC to a given database and transforms the result set into a OLAP JSON format. This OLAPJSON-format represents a OLAP-Cube, which includes spatial information.

The result was reviewed by Thierry B.

During the second sprint a lot of refactoring work was done as well.

Sprint 3 (21.03. - 27.03.11)

	Tasks	Status	Story points
8	Document the new SOLAPLayer architecture	done	2
9	Document how to use SOLAPLayers for a dashboard	done	2
10	Analyze BIRT and find a solution for integrating a map	done	3

	Status	
	to do	0
	in progress	0
	done	7
	number of total story points	**7**

Log book

Date	*Description*
28.03.10	Finish SOLAPLayer improvements

Impediment backlog

Problem

> The time did not suffice to finish all the tasks. The documentation part has to be delayed to the next week. During the next week I will add a short sprint on documentation and testing, before starting with the BIRT-plugin. Probably, I will not finish the whole documentation but at least build a first version.

Retrospective

> With the end of the third sprint is as well the end of one component of the project (at least for the next weeks). During the last weeks I could improve the really procedural source code of SOLAPLayer-server to a full object-oriented software, with all the known advantages: higher comprehensibility, much easier to extend, lower coupling - high cohesion, etc.

> The big amount of work made it sometimes hard to decide where to start. Another challenge is to find a good point of time to stop improving the source code (there is almost always a possibility to do some more improvements).

Sprint review

During the last sprint I could improve the software. I did again a lot of refactoring. Now the sql-queries can be displayed on the sample dashboard as good as a mdx-query. I added some javadoc for explaining the functionality of the different classes.

Sprint 4 (28.03. - 03.04.11)

	Tasks	Status	Story points
9	Document how to use SOLAPLayers for a dashboard	done	2
10	Analyze BIRT and find a solution for integrating a map	done	3
11	Document analysis of Birt-plugin for future traceability	done	3
13	Create a map report item plugin with displays static co	done	3

	Status	
	to do	0
	in progress	0
	done	11
	number of total story points	**11**

Log book

Date	*Description*
29.03.11	Presented SOLAPLayers to Thierry and Jean. Received a positive feedback.
31.03.11	Presented the map2BIRT-analysis to the team and discussed the solution

Impediment backlog

Problem

> As expected did the time not suffice to complete the documentation. I could document a lot, but not finish it by now. My core work will be implementing, as soon the goal is reached. I will reserve some time for documentation each week.

Retrospective

> A really shocking moment was when I had a lock at the implementation of the BIRT-report-items. The source code is really bad and crowded with code smells (while-true-loops, unused code, totally over documented, irreproducible solutions, ...). I did not expect this from a Eclipse core project.

> As already mentioned in the first sprint is the documentation of BIRT really poor, at least out of the view of somebody who wants to extend BIRT. Almost hilarious is the book about extending Birt, which is written by Actuate-employees. The book is nothing more than a circumscribed API and the examples not even run on Linux and mac (after some modification it is possible to run the plugin on all supported OS).

Sprint review

The last sprint was really important. Now I really see the goal of the whole project without any fog. The analysis-part of the sprint helped me and the whole team as well to find a good way to the solution. We all are optimistic, that I will reach the goal until the end of my internship.

The BIRT-plugin-project is set up and the first plugin already in the beginning of the implementation-phase. The goal for the coming sprint is to reach a good point, for the two weeks break, so the last three weeks after the break can be used for improving and especially documenting the solution.

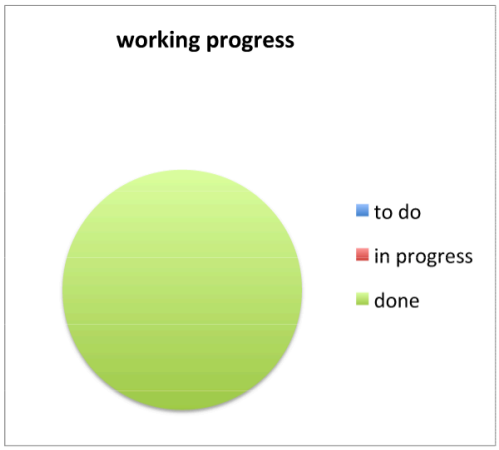
Documents

> SOLAPLayers - Architecture and Developer guide
> Birt extension analysis (initial)
> Presentation - SOLAPLayers2Birt
> Presentation - New SOLAPLayers version

Sprint 5 (04.04. - 10.04.11)

	Tasks	Status	Story points
9	Document how to use SOLAPLayers for a dashboard	done	2
10	Analyze BIRT and find a solution for integrating a map	done	3
14	Retrieve dynamic data using a default SQL-query and SOLAPLayers and display it with existing report items	done	3
15	Retrieve dynamic data using a default SQL-query and SOLAPLayers and display it with the new map item	done	2
16	Create GUI and the logic behind for using customizable SQL-queries	done	3

	Status	
	to do	0
	in progress	0
	done	13
	number of total story points	**13**



Log book

Date	*Description*
04.04.10	Problems with BIRT Preview in Eclipse

Impediment backlog

Problem

> BIRT does not display the SOLAPLayers-Map in the preview view of Eclipse. I did some research, but I did not find the reason. However, the Item gets displayed in the normal HTML-view. I stoped searching a solution for the moment, to not waste time at a complex problem.

> Data binding in BIRT using no table is nowhere documented. I went further to not waste time. Will come back to this problem at the end of the internship.

> Several other problems with BIRT could be solved after time consuming researches and tests.

Retrospective

> What I really learned, is that I should never spend more than one day for a "small" problem. Some little things can be really time consuming. I decided to drop them on the side and resume them if I should have time left at the end of the internship. Instead I force bigger challenge, to reach a better end product.

> Be critically. Not everything which is produced with a well-known-label on it, has to be a good product! All that glitters is not gold!

Sprint review

In this sprint I really forced to reach a A-to-B-scenario: Retrieving data, transform data and display the data with the core report item and my own map report item. I had a lot of problems with working with the BIRT-API. Finally, I reached my goal, even if there are some flaws.

I have now a good basis to improve the plugin during the last weeks and create a useful documentation.

Documents

> Zwischenbericht

Sprint 6 (25.04. - 01.05.11)

	Tasks	Status	Story points
9	Document how to use SOLAPLayers for a dashboard	done	2
10	Analyze BIRT and find a solution for integrating a map	done	3
17	Create GUI for specify the map report item with customizable parameters	done	3

	Status	
	to do	0
	in progress	0
	done	8
	number of total story points	**8**

Log book

Date	*Description*

Impediment backlog

Problem

Retrospective

The work with Birt still brought up a lot of problems. But I didn't spend to much time to solve them and went further to more important parts of my project.

Sprint review

The plugin is now able to send user defined queries to the data source. Because of the many problems, which occurred during the work with Birth, the plugin has not yet really a good usability. For the moment I won't continue with the work on the plugin, so I can concentrate on the documentation. Some estimations will be done at the end of the project.

At the weekend I started to finish the SOLAPLayers documentation. I completed the Javadoc and the textual documentation. The documentation is on a good way and should be finished by the end of the next sprint.

Sprint 7 (02.05. - 08.05.11)

	Tasks	Status	Story points
9	Document how to use SOLAPLayers for a dashboard	done	2
10	Analyze BIRT and find a solution for integrating a map	done	3
19	0	0	0

	Status	
	to do	0
	in progress	0
	done	5
	number of total story points	**5**

Log book

Date	*Description*

Impediment backlog

Problem

Retrospective

Documentation can be such a nice work. After processing quite slow with developing the last weeks, it is nice to see how much is reachable in a week. Still, documenting in English is much more time costly as in German and it can be hard to describe things in the right way.

Sprint review

A big part of the documentation could be finished. The whole SOLAPLayers-documentation is ready. Additionally some important drafts, like the abstract, the management summary and the poster were done.
Some research for the excursion "State of the art of reporting tools" were done.

Sprint 8 (09.05. - 15.05.11)

	Tasks	Status	Story points
12	Create data source plugin with dummy data	done	3
19	0	0	0

	Status	
	to do	0
	in progress	0
	done	3
	number of total story points	**3**

Log book

Date	*Description*

Impediment backlog

Problem

There were no bigger problems this week

Retrospective

Sprint review

The last sprint was time consuming. A lot of documentation could be done. The state of the art of reporting tools document and the BIRT-plugin documentation are almost finish. Now just formal work has to be done in the last remaining days.

APPENDIX F - License agreement

Agreement

[This is a translation of the original German agreement, which is related to the Swiss law.]

1. Matter of the agreement

This agreement defines the rights for the use and the further development of the results of the bachelor thesis „Extending BIRT with Geospatial Data Visualization capabilities by integrating the SOLAPLayers mapping component“ by Christoph Süess. The thesis was developed in cooperation with the company Spatialytics.

2. Intellectual property rights

The student keeps all his intellectual property rights.

3. Usage

The resulting source code is the exclusive property of Spatialytics. The student and HSR have no rights to use and extend the resulting source code. All documents beside the source code can be used and extended by Spatialytics, by HSR and by the student.
The student declares to have read this agreement and agrees to these regulations.

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit „Extending BIRT with Geospatial Data Visualization capabilities by integrating the SOLAPLayers mapping component“ von Christoph Süess unter der Betreuung von Prof. Stefan Keller geregelt. Die Bachelorarbeit wurde in Zusammenarbeit mit dem Unternehmen Spatialytics erarbeitet.

2. Urheberrecht

Die Urheberrechte stehen dem Student zu.

3. Verwendung

Sämtliche Rechte des resultierenden Quellcodes stehen exklusiv Spatialytics zu. Der Student und die HSR haben kein Recht, den resultierenden Source Code zu verwenden. Die resultierenden Dokumente – abgesehen vom Source Code – können von Spatialytics, der HSR und dem Studenten verwendet und erweitert werden.
Der Student erklärt hiermit von dieser Vereinbarung Kenntnis genommen zu haben und stimmt den Bedingungen zu.

Québec City, 29.04.11

.....
Christoph Süess, Student HSR

Québec City, 29.04.11 .....
Dr. *Thierry Badard*, Spatialytics

Rapperswil, ...
Prof. Stefan Keller, Supervisor HSR

Rapperswil, ...
Prof. Hansjörg Huser, Head of department Computer Science HSR

APPENDIX G - Agreement of the author

Ich erklären hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum: Name, Unterschrift:
Christoph Süess

APPENDIX H - Instructions: Creating a dashboard using SOLAPLayers

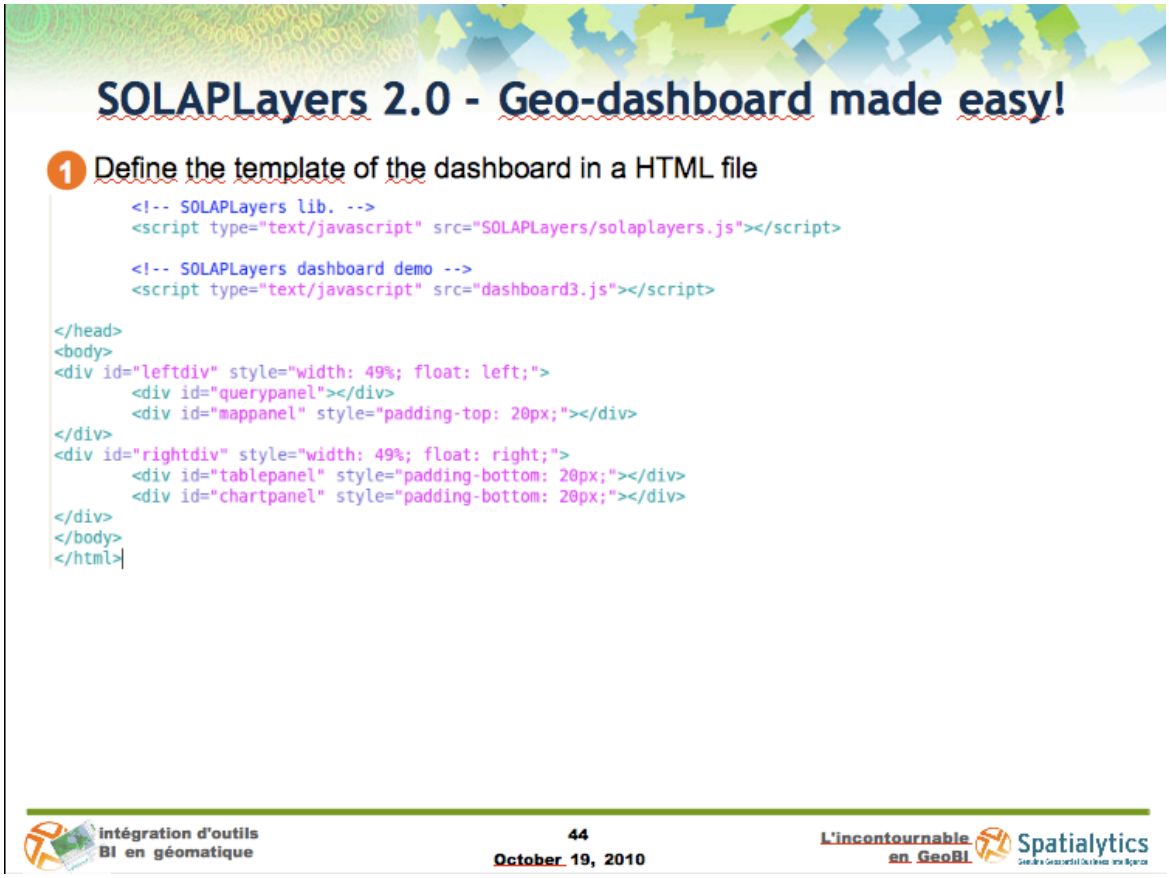


Figure 40 - Introduction of creating a SOLAPLayers dashboard – Step 1

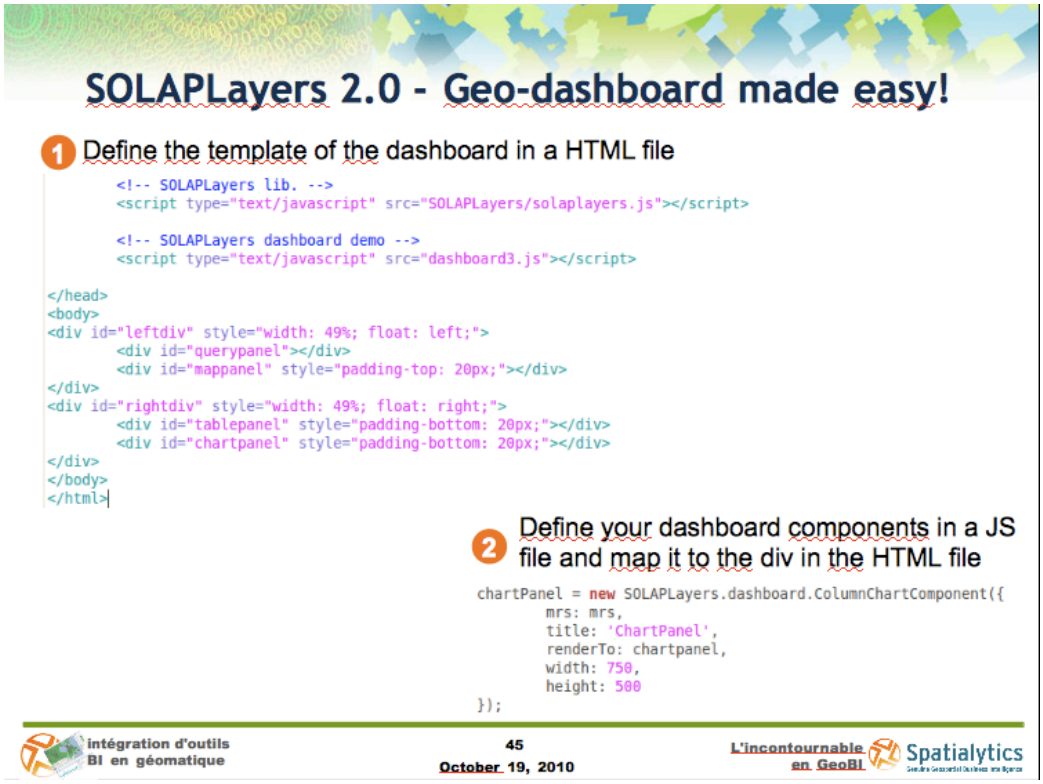


Figure 41 - Introduction of creating a SOLAPLayers dashboard – Step 2

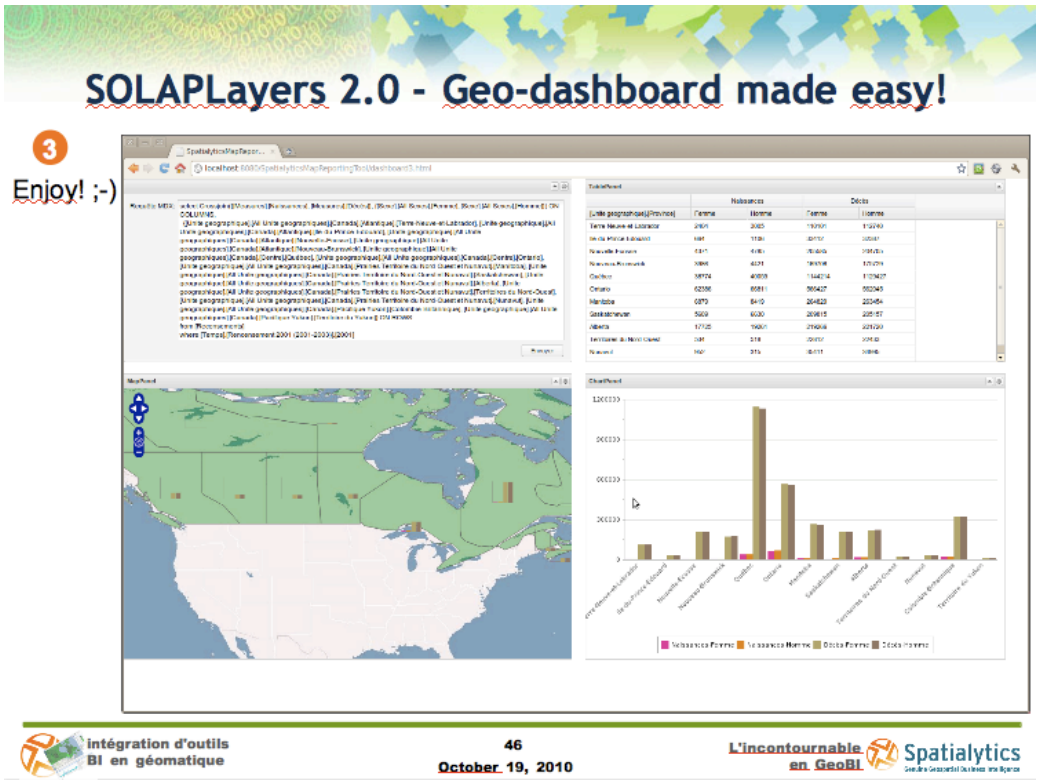


Figure 42 - Introduction of creating a SOLAPLayers dashboard – Step 3

APPENDIX I – Customized SCRUM-process for bachelor thesis

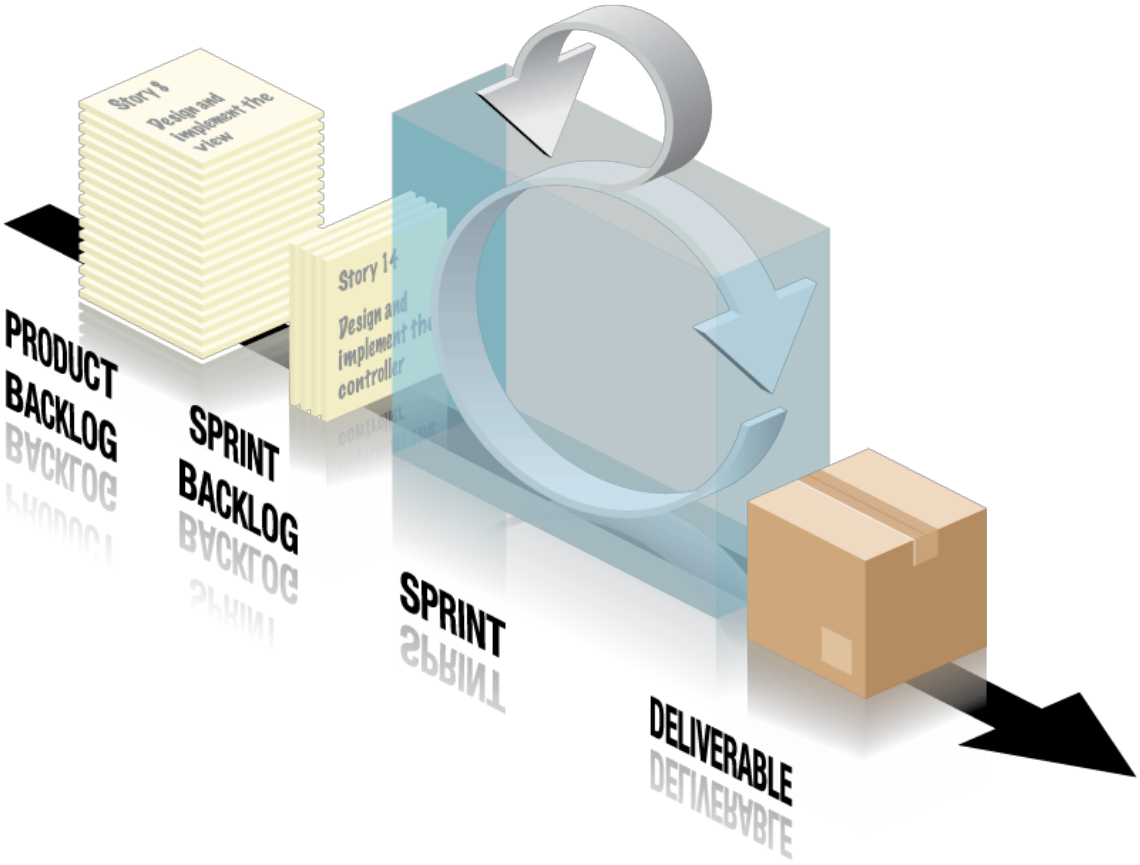


Figure 43 - SCRUM Process

Introduction

SCRUM as a project management instrument is used for the project „Extending the BIRT Reporting Tool with Geospatial Data Visualization". The goal is to work with an agile project management instrument that fosters a fast and high quality way of development. In the case of this bachelor thesis SCRUM has to be customized for different reasons:

- The project has just one main participant
- It is not possible to realize the project isolated from the rest of the team, because of existing technologically dependencies
- The main participant assumes different roles
- The project environment does not work with a specific software development process

This document describes the customized SCRUM-process for this specific project.

Basic principles

In comparison to non-agile development processes, SCRUM does not ignore the fact, that it is not possible to completely plan a project in advanced. SCRUM knows the reality: The vision, the goals and the project process are products of the whole process.

SCRUM base on these important guiding principles:

- At the end of each sprint exists a full tested, running version of the application

- No documentation for the sake of just doing it.
- "Working more hours" does not necessarily mean "producing more output"

Roles

SCRUM defines the following roles:

- **Product Owner**
  - Defines the Vision. The Scrummaster helps to find the right direction. The Product Owner defines the goal during the sprint. He plans and reviews the done work formal at the end of every sprint.
- **Team**
  - Creates together the Product Backlog, a list of by business value prioritized features. The team defines the needed time for each functionality. During a sprint, the team is completely organized by themself. If a problem occurs, the team talks to the Scrum Master.
- **Scrummaster**
  - Helps the team if it is stuck. He writes no code by himself. He has the connection to the management.
- **Management**
  - Talks primary to the Scrummaster concerning financial and other matters.
- **Customer**
- **User**

Because of the size of the project team the roles cannot be assigned and defined in the same way as usually: The team (Christoph Süess) will adopt the duties and responsibilities of the product owner and the Scrummaster in co-operation with the management (Thierry Badard).

Sprint

A sprint will have the duration of one week, starts at Monday and ends usually at Friday (weekend as reserve). The sprint ends with a report to the project advisor (Prof. Keller) on Monday. A sprint contains an amount of items out of the task board, which gets expanded continually.
A sprint should not be changed during the week. In some cases it is not possible to retain on the items (organizational reasons, dependencies to other projects). In this case the team changes the sprint and notes the reason.

Project documentation

Besides the knowledge, design and analysis documents, every sprint is represented by a document containing the following information:

- **Sprint task board**
  - The chosen items from the project task board
- **State of progress**
  - The actual state of progress
- **Retrospective**
  - What did the team learn during the sprints? Improvements for future sprints?
- **Sprint review**
  - What did the team achieve?
- **Immediate Backlog**

  - Where are the problems? Why does de development not go as fast as planed.
- **Logbook**
  - Date and reason for the adjustments of the sprint

APPENDIX J – Interim report

Seit dem 23. Februar 2011 befinde ich mich in Québec City, einer sympathischen Stadt an der Ostküste Kanadas. Ich habe hier eine Wohnung gefunden und mir ein kleines, aber feines soziales Umfeld geschaffen. Meine Entscheidung, die BA im Ausland zu schreiben, bereue ich trotz allem Aufwand und einigen Hindernissen (siehe Herausforderungen und Probleme) in keinem Fall. Bereits schaue ich mit Wehmut ans Ende meines Aufenthalts.

Ich schreibe hier meine Bachelor-Arbeit mit dem Titel „Extending BIRT with Geospatial Data Visualization capabilities by integrating the SOLAPLayers mapping component“ im drei Mann Unternehmen Spatialytics. Der Kopf der Firma bilden Luc Vaillancourt (Management) und Thierry Badard (Professor an der Universität Laval, Entwicklung). Jean Mathieu ergänzt das Team. Alle drei haben Geologie studiert und sind auf verschiedene Wege in die Informatik gekommen. Als waschechter Informatiker ergänze ich das Team super und kann mit meinem Wissen dem Unternehmen weiterhelfen.

1. Woche – Umfeld, Tools und Technologien kennenlernen

Die erste Woche im neuen Unternehmen: Es gab viel zu lernen. Vor allem Wissen rund um OSBI, Reporting Tools und Birt, aber auch das klassische kennenlernen des Unternehmens hielten mich während den Tagen auf Trab. Nebenbei plante ich mein Vorgehen und versuchte die ersten Ziele zu formulieren.

2. – 4. Woche – SOLAPLayers – OSBI meets Geo-Dashboard

In den drei Wochen kümmerte ich mich voll und ganz um SOLAPLayers. Meine Ausgangslage war ein kleines Projekt. Das Tool war funktionstüchtig, aber nur in wenigen Hinsichten objektorientiert und für den Quellcode hätten Sie mir wahrscheinlich meine SE2-Note entzogen. Was ist SOLAPLayers? Welches ist die Vision hinter SOLAPLayers?

„SOLAPLayers ist primär ein Framework um geografische Daten von SOLAP-Datenbanken in einem interaktiven Dashboard darzustellen. Sekundär adressiert SOLAP aber auch andere Datenquellen und präsentiert auch nicht geografische Daten. Somit ist SOLAPLayers ein Tool um komplette Dashboard-Anwendungen zu erstellen. SOLAPLayers ist eine Webapplikation, die serverseitig in Java umgesetzt ist und im Clientbereich mit JavaScript (EXT JS-Framework), HTML und CSS arbeitet. SOLAPLayers befindet sich noch in den Kinderschuhen und steht unter einer open-source Lizenz (Mehr Infos: www.solaplayers.org)“

Mein Aufgabenbereich befand sich ausschliesslich im Server-Bereich. Grundsätzlich habe ich zwei grosse Änderungen, sprich Erweiterungen forciert:

1. Umgestalten der Software Architektur, so dass neuen Datenquellen und Output-Formate einfach und im objektorientierten Sinne hinzugefügt werden können. Dazu gehörte auch ein Refactoring des Quellcodes (so dass der Quellcode SE2 würdig ist).

2. Hinzufügen eines Treibers für relationale Datenbanken. Dieser SOLAPLayers-Treiber erlaubt es anhand eines SQL-Queries und weiteren Parametern einen MDX-Query zu simulieren und die Resultate wie gewohnt anzuzeigen. Diese Erweiterung war nötig, da nach wie vor relationale Datenbanken das Zepter halten (auch wenn sie im Bereich der Analyse sehr viele Nachteile mit sich bringen).

Da ich anfänglich von der mir zur Verfügung gestellten Basis etwas geschockt war, habe ich sehr oft in Eigenregie gehandelt. Thierry war dann aber jeweils sehr zufrieden mit dem Resultat.

5. – 6. Woche

Nun kam das Reporting Tool BIRT ins Spiel, welches mir seit zwei Wochen das Leben schwer macht. BIRT ist ein Eclipse-Core-Project und wird vor allem von dem Unternehmen Actuate entwickelt. BIRT ist zur Anwendung und zum Integrieren in andere Java-Applikationen bestimmt ein brauchbares Werkzeug. Wer BIRT jedoch erweitern will, fühlt sich öfters wie ein Nomade auf der Suche nach Wasser irgendwo tief in der Sahara ☺.
Nach vielen überwindeten Hürden habe ich es geschafft ein simples „A to B"-Scenario umzusetzen. Ab Heute ist es nun also Möglich, SOLAPLayers als BIRT-Plug-In zu nutzen, auch wenn die Bedienung noch weit von Benutzerfreundlich entfernt ist.
Thierry sieht die Probleme mit BIRT und äussert sich sehr positiv zu meinem aktuellen Stand.

Weiteres Vorgehen

Die nächsten zwei Wochen suche ich kein Wasser, sondern geniesse ich die neue Welt in vollen Zügen. Nach den zwei Wochen Break, wende ich mich erneut dem BIRT-Plug-In zu. Da ich eine sehr fixe Deadline habe, steht in den letzten drei Wochen vor allem die Dokumentation an erster Stelle. Ziel ist es Spatialytics ein gut dokumentiertes, funktionsfähiges BIRT-Plug-In zu hinterlassen, mit dem Spatialytics, Sie und auch ich sehr zufrieden sind.

Herausforderungen und Probleme

Die Themen des Semesterprojekts und meiner Studienarbeit habe ich jeweils selber definiert und zusammen mit meinem Team sehr gute Resultate erzielt. Die Herausforderung lag jeweils ganz klar im technischen Bereich, sprich Performance, Concurrency usw. waren Probleme die wir forcierten.

Meine BA bringt ganz andere Probleme mit sich. Diese Hindernisse sind nicht weniger lehrreich, aber leider verhindern sie teilweise, dass ich mich den interessanten, technischen Probleme widmen kann:

1. Spatialytics ist ein Start-Up, die stabilen Säulen fehlen noch, sprich alles ist im Aufbau. Auch wenn das sehr spannend ist, ist es grundsätzlich ein schlechtes Umfeld für eine BA. Es bestehen viele Abhängigkeiten, die sich immer wieder ändern.
2. Auch wenn Spatialytics inzwischen ein grosses IT-Knowhow hat, ist der Quellcode nicht so professionell, wie er an der HSR erwartet wird. Darauf aufzubauen ist schwierig. Darum erfordert meine BA viel Zeit um solchen Code in

Ordnung zu bringen (Refactorings). Gewisse Vorgaben akzeptiere ich aber aus zeitlichen Gründen, ohne den technischen Sinn zu hinterfragen. Schliesslich will ich mit meiner BA neben Spatialytics auch Sie, als meine Betreuer und Bewerter, zufrieden stellen.

3. In zwischen klappt das Dokumentieren und Diskutieren in Englisch ziemlich gut, trotzdem die Fremdsprache kostet Zeit und Energie.
4. Alleine an einem spezifischen Projekt zu arbeiten bringt genau so viele Vor- wie Nachteile. Einerseits entfällt ein Grossteil an Kommunikation, andererseits fehlt auch eine Person um technische und organisatorische Probleme zu diskutieren und sich eine Zweitmeinung einzuholen.
5. Das einzige Problem, das mich wirklich auf Trab hält und auch mal auf meine grundsätzlich hohe Motivation schlägt heisst BIRT. Das Erweitern von BIRT ist sehr schlecht dokumentiert, auch wenn BIRT grundsätzlich eine normale RCP ist, gibt es viele BIRT-spezifische Dinge, die nicht dokumentiert sind. Die zwei existierenden offiziellen Tutorials sind nur auf Windows lauffähig, da Sie ein Basis-Konzept von SWT ignorieren. Selbst das eine existierende Buch ist nichts mehr als ein Abbild des ohnehin schon schlechten JavaDocs. Wer in der Community und der Dokumentation keine Antworten findet, hat nur eine Möglichkeit: Direkt den Quellcode analysieren. Bei BIRT ist man dort bei der Quelle des Grauens angekommen. Unglaublich wie schlecht der Code ist, eine super Vorlage für ein SE2-Projekt. Es kann sehr frustrierend sein, wenn man stundenlang anstelle von produktiver Arbeit sich dem formulieren von Google-Suchstrings und dem durchforsten von Foren widmen muss.

Fazit

Obwohl die Probleme viel Zeit in Anspruch nahmen, bin ich nach wie vor überzeugt, dass ich mich auf einem sehr guten Weg befinde. Das Feedback von Thierry war bisher immer super und von der Betreuer-Seite wurden bis anhin noch keine Bedenken oder ähnliches geäussert. Auch wenn es etwas schade ist, dass ich auf Grund der Probleme nicht gleichviel Aufwand in das bewältigen von technischen Hürden stecken kann (wie beim Semesterprojekt und der SA) ist diese Art von BA sicher eine super Vorbereitung für die Arbeit nach dem Studium, wo die Bedingungen auch unterschiedlich sind.

Ich habe in den letzten sechs Wochen sehr viel gelernt und freue mich die BA in den letzten drei Wochen fertigzustellen. Ich hoffe, dass ich auch Sie mit meinem Resultat begeistern kann.

APPENDIX K - Result presentations

During the project multiple presentation were hold. The notes can be found on the CD:

- BIRT - Open-source Business Intelligence and Reporting Tool
- SOLAPLayers - Changes and improvements of the new version
- Map extension for BIRT
- SOLAPLayers to BIRT – Final results presentation

APPENDIX L – State of the art of reporting tools

1 Introduction 90
2 Purpose of reporting tools 90
3 Functionality of a basic report tools 91
4 Dashboard 91
5 Open source reporting tools 92
Pentaho 92
About 92
Reporting tools 92
JasperSoft 92
About 92
Reporting tools 93
Actuate 93
About 93
Reporting Tools 93
Comparison Matrix 94
Design Paradigm - Pixel positioning 103
Report compilation / Report format 103
Eclipse plug-in available 103
Data Sources Types 104
Graphical Query Designer 104
A look to the future 104
6 Conclusion 105
7 Sources 105
8 Table of figures 106

1 Introduction

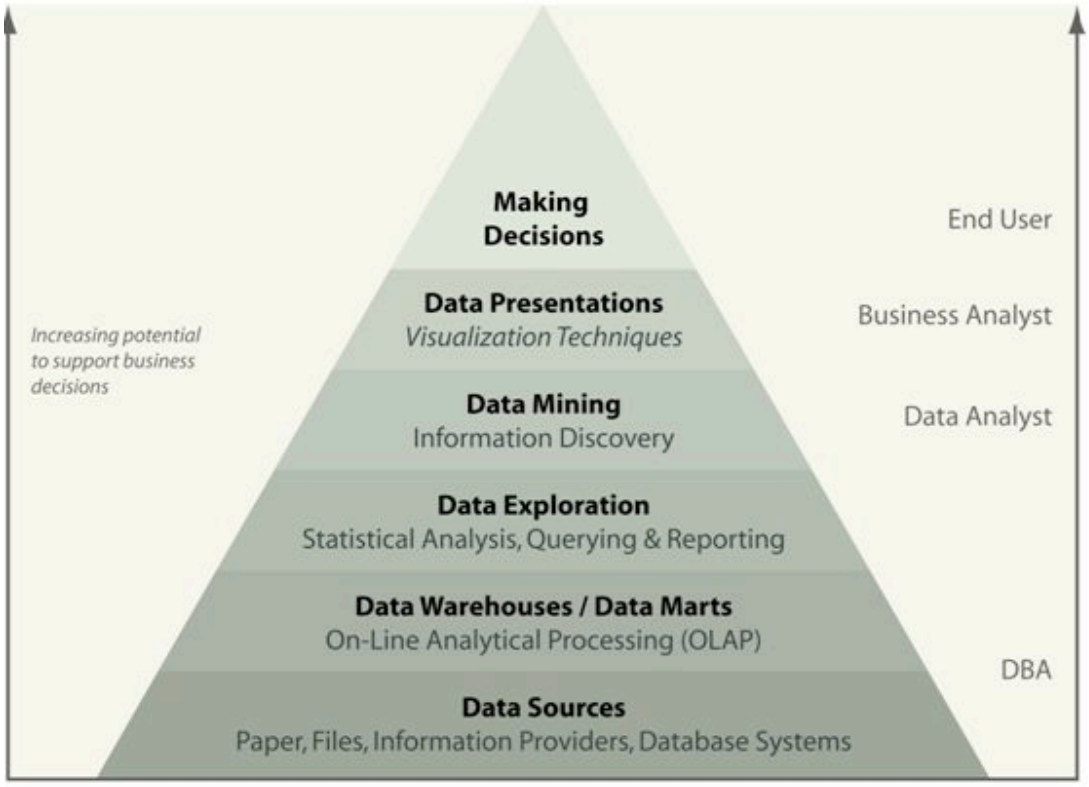


Figure 44 - Ilustration of the decision making process

In the nineties the term Business Intelligence came up. Business Intelligence is a business-process for decision-making. BI helps to navigate the running business into the right direction. To make correct decision, the decision-maker needs an overview of all information, which could influence his decision. BI is a process, which manages the collation and the transformation of raw data into meaningful information. Now a day, BI is supported by a lot of powerful application, which can handle the huge amount and the complex relations of the information.
This excursion will inform the reader about a special part of BI, the reporting part. Reporting tools are applications they access data and represent the information in a human readable format. The goal is to create reports, which helps the decision-maker to overview and find the important information. This article describes the purpose and the evolution of reporting tools as well as it shows some concrete open source software-tools and there advantages and disadvantages.

2 Purpose of reporting tools

Not all companies use professional reporting tools. Especially some smaller firms work without reporting tools. Microsoft Excel is not a real reporting tool, but it is used by a lot of companies for calculation and reporting reasons. The majority of big companies uses BI-suites to cover the whole business-process.
The leading company in business software is SAP. SAP has tools for different parts of the BI-process. One of there reporting tools is SAP Business Explorer. Other well-known proprietary reporting tools are Oracle Reports, Bissantz DeltaMaster, Cognos 8 BI, SAP Crystal Reports and Cubeware Cockpit V6pro. Just the Oracle and the SQP-tools are pure reporting tools, the other products are complete BI-suites, which include a reporting component.

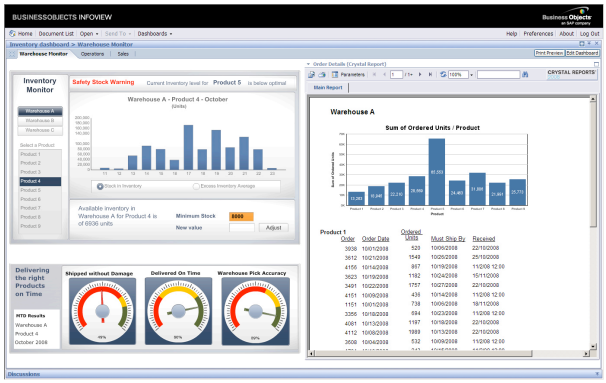


Figure 45 - Crystal Reports dashboard

With the evolution of the proprietary software, some open source tools entered the market as well. The proprietary providers of BI-tools are still leading, but the open source part at market is growing fast. A lot of open source project have proven, that open source does not just stand for a social thought, moreover it has a lot of advantages for the customers and for the companies, which contribute the source code.

3 Functionality of a basic report tools

All common reporting tools function in the same basically way:

- The first step is to define one or more data sources. This can be a relational database, an OLAP-Server, flat files, web services or any other supported data source.
- In a second step the user creates a report-template. A lot of reporting tools provide a designer, which allows creating a report in an easy way (WYSIWYG). The report creator defines which data, from which data source is displayed in which way. He can choose report items like tables, cross tables, charts and many other representations. This report is often represented by an xml-file on the file system.
- The resulting report is now interpreted and filled with the information using the given data sources. Common outputs are pdf-, xls-, html- or doc-files.
- For more advanced use, the reports can be deployed on servers. This allows providing the reports to different peoples or even as a web service.

4 Dashboard

Reports represent a static view on data. After the engine created the reports, they are saved in a common format like pdf, xls, ppt, doc, etc. These reports can be created new at any time, to get actual information.

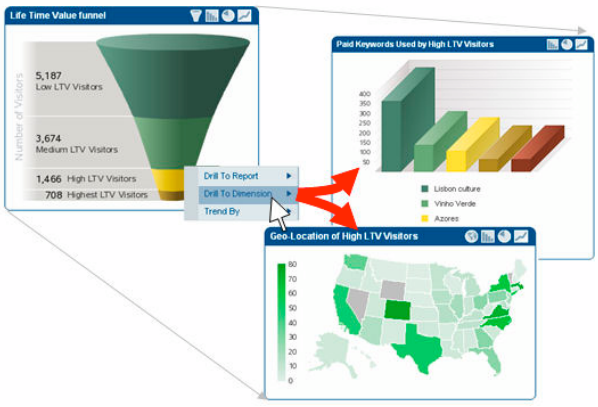


Figure 46 - Drill down report

Dashboards go a step further. Dashboards allow navigating through data, getting more details and seeing related content. Dashboard are often implemented as websites, this allows to set access rights for different users (e.g. different views for director, department heads and team leaders). Some reports provide some dashboard functionality that is why in some cases it is not possible to clearly separate dashboards from reports.

The following table shows the advantages of reports and dashboards in general:

Report	Dashboard
• Easy to print and send by E-Mail • Can be used offline after creation • Easy to archive	• Allows to navigate through the information (drill down / roll up) • Better visualisation of relations between different data • Better understanding and overview through interactive presentation

5 Open source reporting tools

Figure 47 - Well-known OSBI-tools

The most famous open source reporting tool providers are JasperSoft, Pentaho and Actuate, all of them use Java. For all this products exists a commercial version. In the following chapters the companies and there reporting tools are explained. In a further chapter the three products are compared and some of the important facts are described more detailed.

Pentaho

About

Pentaho is a company from Orlando, Florida with around 200 employees. The company offers a complete BI-suite. Important and well-known tools of Pentaho are Kettle, Mondrian and Weka. With more than six million downloads of there products, around thousand enterprise customers and a community of 20'000 people is Pentaho the head of the open source BI-movement.

Reporting tools

The Pentaho Reporting Community Edition provides three different tools.

- The **Pentaho Report Designer** is used to create reports. It is a standalone application, which allows dragging and dropping report items into a report-document. Properties allow customizing these items. OpenFormula/Excel-formula expressions make it possible to format data dynamically. Pentaho Report Designer is easy to use for developers as well as for business people.
- The **Pentaho Reporting Engine** is a part of the Pentaho Report Designer. This engine creates the reports within the designer.
- The **Pentaho Reporting SDK** is used to integrate Pentaho in server or offline applications. It allows using the Reporting Engine-features as a java library.

Additionally a report server called **Pentaho BI Server** exists.

JasperSoft

About

Like Pentaho, JasperSoft provides a complete Bi-suite as well. JasperSoft is located in San Francisco and has around 100 employees. From the twelve million downloads around twelve thousand people use the commercial licence. One of the popular products of JasperSoft is the OLAP server Mondrian.

Reporting tools

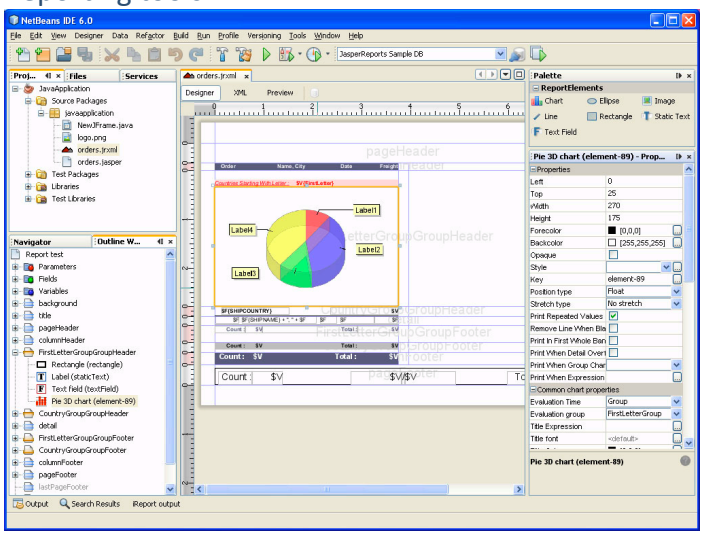


Figure 48 - IReport from JasperSoft

IReport is the name of the JasperSoft report designer. With IReport reports for the **JasperReports** engine are created. **JasperServer** allows uploading reports and providing them as web sites, web services and scheduled distributions.

Actuate

About

The company Actuate has his headquarter in San Francisco, California. Actuate is the main contributor of the Eclipse core project BIRT. Actuate does not provide any other BI-products. Around 600 people work for Actuate.

Reporting Tools

BIRT is an Eclipse based reporting tool. It can be integrated into J2EE applications or can be used as a simple standalone-reporting tool. BIRT allows specifying almost every detail of a report. BIRT is consisted of two components. The Eclipse integrated **BIRT REPORT Designer** allows to create xml-based reports using a comfortable WYSIWYG-editor. The **BIRT Report Engine** processes the xml-based reports and creates reports in different output formats. The **BIRT IServer** is a proprietary product, which delivers the information to an unlimited number of persons.

Comparison Matrix

The following table was published by Innovent Solutions (http://www.innoventsolutions.com/birt-jasper-pentaho-comparison-matrix.html) and updated on 03.05.2011 by the author of this document. The marked rows are described more in detail at the end of the document.

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
Website	www.birt-exchange.com www.eclipse.org/birt	www.jasperforge.com	reporting.pentaho.com
License	Eclipse Public License	GNU Lesser General Public License	GNU Lesser General Public License
REPORT DESIGNER	BIRT Report Designer 2.6.2	Jasper IReport 4.0.2	Pentaho Report Designer 3.8 stable
Designer Platforms	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X
Eclipse Plug-in Available	Y	N JasperAssistant is available commercial, non-free, third-party plug-in	N Note that the Pentaho Design Studio, an Eclipse plug-in, is not a report designer.
NetBeans Plug-in Available	N	Y	N
Standalone Java Client Available	Y	Y	Y
Design Paradigm	Web Page Design paradigm: frames, tables, lists	banded reports, pixel positioning, table	banded reports, pixel positioning
Report Compilation	Not required	Required	Not required

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
Report Format	XML (.RPTDESIGN files are pure XML)	.JRXML report design files are compiled into .JASPER report files, which are Java Byte Code. You then deploy the .JASPER file.	XML (.PRPT report file is a ZIP that contains the XML file plus other resources)
Report Designer Components			
Common Report Designer Components: **- Report Editor Palette** **- Data Explorer Property** **- Editor Outline view** **- Preview Expression** **- Builder Report** **- Problems Chart Builder** **- Script Editor**	Y	Y	Y
Sub-reports	Y	Y	Y
Side-by-side report components	Y	Y	Y via sub-reports
Tables	Y	Y	N
Cross-tabs	Y	Y	“Experimental” in Pentaho 3.5
Sorting Horizontal Panning	Y	Always scroll down, even if crosstab expands side-to-side	N
Newspaper / multi-column layout	N	Y	N

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
Hyperlinks within a report	Y	Y	Y
Actionable charts	Y drill-down, hyperlinks, mouse-overs	Partial – hyperlinks	N
Cascading Style Sheets (CSS controlled format)	Y	Y	N
Conditional Formatting	Y	Partial	Y
Data Sources			
Multiple data sources and queries per report	Yes, plus support for joining them	Only via sub-reports (one data source per sub-report)	Only via sub-reports (one data source per sub-report) or in charts
Support for joining multiple data sources in the Designer	Y	N	N
Report can further re-sort, filter, or group data returned from a query.	Y	Partial – Jasper can further manipulate data sets before building cross-tabs, but not for regular tables.	N
Data Sources Types	- Database JDBC	- Database JDBC	- Database JDBC
	- XML File	- XML File	- XML files
	- Web Services	- CSV File	- Table
	- Flat files: CSV,SSV,PSV,TSV	- Microsoft Excel	- OLAP MDX
	- Custom data source	- JavaBeans	- Pentaho Metadata
	- OLAP MDX	- Hibernate HSQL	- Pentaho Data Integration

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
	- Scripted Data Source: POJO, EJB, Hibernate, XML Stream	- Spring Hibernate EJBQL	- OLAP MDX
		- XMLA Server	- Scripted Data Source: POJO,EJB,Hibernate
		- Mondrian OLAP	
		- Text File	
		- NetBeans JDBC	
		- POJOs	
		- Custom	
		- Remote XML files	
Query Designer	Y	Y	Y
Graphical Query Designer	Prototype only	Y (SQL Leonardo)	Y (SQL-Leonardo)
Scripting	- JavaScript	- JavaScript	- JavaScript
	- Java Event Handlers	- Groovy	- Bean Script Framework (BSF)
		- Java	- Bean-Script Host (BSH)
			- Single Value Query
Output Formats			
Paginated HTML	Y	Y	Y
Unpaginated HTML	Y	N	N
PDF	Y	Y	Y
Excel (XLS)	Y	Y	Y
XML	Y	Y	Y
Plain Text	Y	Y	Y
RTF	Y	Y	Y
Powerpoint (PPT or PPTX)	Y	Y	N

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
CSV	Y	Y	Y
Postscript	Y	Y	Y
OpenOffice report types (document & spreadsheet)	N	Y	N
Microsoft Office 2007 report types (DOCX, XLSX)	N	Y	N
Flash (SWF)	N	Y viewed within Jasper Server's flash report viewer	N
Custom Formats	Y	Y	Y
Geometric shapes & lines	Y	Y	Y
Barcodes	Y	Y	Y
Charts			
Chart Wizard	Y	Partial two-step wizard, rest is a big dialog box	N
Chart Interactivity	Y mouse-over, tool tips, hyperlinks, etc.	Y hyperlinks only	N
Chart themes	Y	Y	N
Precise control over format of all control elements	Y	N	N

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
Charts types that all have: 2D, 3D, Pie, Multi-pie, Bar, Stacked Bar, Bar XY, Line, Line XY, Area, Area XY, Stacked Area, Bar Line, Bubble, Scatter, Plot, Multi-Axis	Y	Y	Y
Study Charts	Y	N	N
Ring Chart	N	N	Y
Tube chart	Y	N	N
Cone chart	Y	N	N
Pyramid	Y	N	N
Time Series	Y	Y	N
Meter / Gauge	Y	Y	N
Waterfall	N	N	Y
Step	N	N	Y
Step Area	N	N	Y
Difference	Y	N	Y
Radar/ Spider	Y	Y	Y
Thermometer	N	Y	N
Candlestick / Stock Chart (High/Low)	Y	Y	N
Gantt	Y	Y	N
Survey Scale	N	N	Y
Bar Sparkline	N	N	Y
Line Sparkline	N	N	Y
Pie Sparkline	N	N	Y

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
Report Parameterization			
Static Parameters select parameter values from a hard-coded list of values	Y	Y	Y
Dynamic Parameters users select parameters from a list of values that came from a database	Y	Y Cascading input controls are report independent	Y
Cascading parameters linked data driven prompts e.g. **- Country** **|- State** **|- City**	Y	Y	Y
Calendar date-picker for parameters of type date.	Y	Y	Y
Can specify default values	Y	Y	Y
Drop-down list boxes	Y	Y	Y
Radio buttons	Y	Y	Y
Check boxes	Y	Y	Y
Combo boxes	Y	Y	Y
Aggregates / Summarize Data			
Common Aggregations	- Average - Count	- Average - Count	- Average - Count

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
	- Distinct Count	- Distinct Count	- Count by Page
	- First	- Sum	- Group Count
	- Is-Bottom-N	- First	- Sum
	- Is-Botton-N-Percent	- Lowest (Minimum)	- Minimum
	- Is-Top-N	- Highest (Maximum)	- Maximum
	- Is-Top-N-Percent	- Standard Deviation	- Sum Quotient
	- Last	- Variance	- Sum Quotient Percent
	- Max	- System	- Calculation
	- Median		- Count for Page
	- Min		- Sum for Page
	- Mode		- Sum (Running)
	- Moving Ave		- Count (Running)
	- Percentile		- Group Count (Running)
	- Percent-Rank		- Count Distinct (Running)
	- Percent-Sum		- Average (Running)
	- Quartile		- Minimum (Running)
	- Rank		- Maximum (Running)
	- Running Count		- Percent of Total (Running)
	- Running Sum		
	- Standard Deviation		
	- Sum		
	- Variance		
	- Weighted Average		
User Defined Functions / Expressions	Y Java, JavaScript	Y Java, JavaScript, or Groovy	Y OpenFormula (Excel-like), Java
User Define Aggregates	Y	Y	Y

	BIRT 2.6.2	JasperSoft Community 4.0.2	Pentaho Reporting Community Edition 3.8
Designer provides "code hooks" (aka callbacks, event handlers. …) for each report element.	Y	Partial – only the report itself has callbacks – not at the per-element level.	Not in Designer.
Support for Reuse Within Designer			
Templates (custom report starting points)	Y	Y	Y
User-defined Libraries (reusable report pieces)	Y	N	N
Styles colors, fonts, borders, margin…	Y	Y	Y
CSS	Y	Y	N

Design Paradigm - Pixel positioning

There are two ways to position the report items, like tables, charts, etc. into a report. Pentaho and JasperSoft allow positioning their elements using pixels. In this way the positioning can be done really exactly. This may have some advantages. The big disadvantages of this variant are that the reports are shown the same way on all the different screens and that all elements have to be positioned, since there is no automatism. The both reporting tools show a report in the same way on a wide-screen as they show it on a normal screen. This is not very comfortable.
BIRT uses another way to position his elements. The report can be split into different cells using a grid layout. The effective size of report items will be calculated dynamically for each screen.

Report compilation / Report format

Pentaho and BIRT are working the same way. The designer creates an XML-file, which is used by the report engine to retrieve the data and create the expected output. Using a XML based file has some important advantages. XML is a human-readable format. A designer is not necessary to create reports, as long the syntax is documented well. Another advantage is the collaboration with source control tools. The difference between two versions of a report is easy to figure out.
JasperSoft compiles the reports in the designer and creates an unreadable data format. This may speed up the interpretation-process, but all the mentioned advantages of readable formats are lost.

Eclipse plug-in available

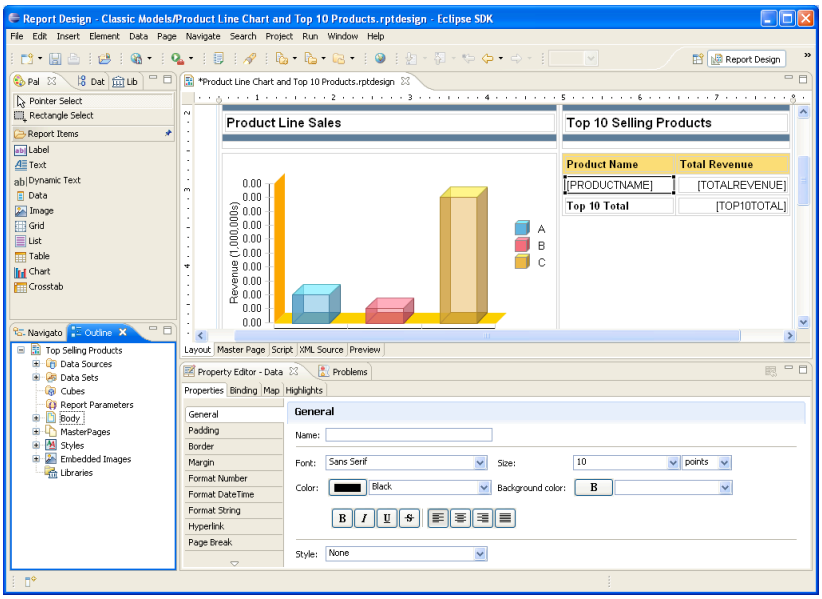


Figure 49 - BIRT-plugin for Eclipse

Why do we care about an Eclipse plug-in? Does it matter if I create my report in Eclipse or in a standalone application? Yes, it does. Big companies use the reporting tools rarely for to create some reports on a standalone computer. They want to provide reports to persons in different executive positions. For this matter J2EE applications were written. Since Eclipse is the most widespread IDE, it makes sense to be able to work on the report and the application in the same environment.

Data Sources Types

Figure 50 - Visualization of a data cube

A lot of data sources have something important in common. There results of this data sources ends in a two-dimensional result set. All the report items are able to display a table or a cell out of the table. MDX queries for example do not deliver a two-dimensional result set, the result is three- or even multidimensional. BIRT for example supports OLAP just in a non-native way. It allows to create data cubes out of tables, but it is not possible to send a mdx query to a OLAP-server and display the result directly as a cross table.

Graphical Query Designer

A common problem with reporting tool is there complexity. Creating complex reports using different data sources and report items in an easy way is obviously an antithesis. This results in the fact that reports often have to be created by developers and not by business-people, which are familiar with the thematic. Another fact is, that the communication between to different involved parties often raises problems. Features like a graphical query designer try to reduce these dependencies.

A look to the future

The business-world grows fast. New trends, like social medias, change the methodologies and strategies of companies. New technologies allow the company to capture new and more information, which could influence their strategy. Reporting tools have to adapt to this alternations.

Figure 51 - Logo SAP

On the one hand, it is the technical aspect, which has to be improved. The gigantic amount of information makes it difficult to create reports and dashboards in an appropriate time. In our fast-changing world is no time to wait for hours to get an overview over the facts. That is why known distributors of reporting tools work on a solution, to make the process faster then ever before. For millions of data sets, it is normal to wait for some hours. Using the new systems the process can be done in real-time.

One of these solutions was presented at the end of 2010 by SAP. SAP HANA (SAP High-Performance Analytic Appliance) is high performance system. HANA uses a technology called In-Memory Computing. The two main character of In-Memory are, that the databases are stored in the main memory and with a new database concept the amount of touched data for a request could be reduced. This combination of soft- and hardware innovation improves the usage of reporting tools immense. Hilti, a construction-company from Liechtenstein, is one of the first well-known companies, which use SAP HANA for there BI-process.

Figure 52 - Spatialytics connects BI with geo-spatial components

Besides of the use of new technologies to speed-up the process, the type of interesting information changes. A good example for the growing of the amount of data shows the bonus programs they came up in the early nineties. With new information of customers, the marketing-strategy can be optimized for each customer individual.
An interesting aspect of information, which wins more and more of the attention, is the geo-spatial aspect. With the evolution of smartphones, location based apps came up fast. A new trend is the location based real-life games, like foursquare. It is for sure, that this trend will reach the business-world as well and the geo-spatial information will influent the strategies and decisions of the companies.
The modern reporting tools contain no or just simple components to display geo-spatial information. But a pin on a map is not the best and only way to show geo-spatial information. There are much more opportunities to display complex forms and charts directly on maps and relate them to other data representations. Spatialytics is a company, which works on this parts and tries to bring this feature to open source reporting tools

Figure 53 - BIRT Mobile Viewer IPad app

A third aspect, which will be improved in the future, is the way of the presentation of the data. Already now, some contributors are developing apps for displaying reports on mobile devices. Especially tablet-pcs allow to discuss decisions everywhere, at any time in a confortable way.

6 Conclusion

The importance of reporting tools is huge and it is still growing. The influence of open source products is bigger then ever before. All the presented open source products have similar features. The real different can be found in the details. In the future new features will amplify the usage for business-companies and the contributors of reporting tool can benefit from these developments.

7 Sources

Type	Source	Date
Image	http://a3.mzstatic.com/us/r1000/008/Purple/ea/c2/40/mzi.nagcebvp.175x175-75.jpg	15.05.2011
Data	http://de.sap.info/hana_newdb_westmere_inmemory/44054	15.05.2011
Data	http://en.wikipedia.org/wiki/Report	15.05.2011
Data	http://fico-forum.de/Painter.php	
Data	http://help.sap.com/saphelp_ish471/helpdata/DE/84/b0b7397a672963e10000000a114084/content.htm	13.05.2011
Data	http://it-republik.de/jaxenter/artikel/Pentaho-BIRT-und-JasperReports-im-Vergleich-1737.html	14.05.2011
Data	http://jasperforge.org/plugins/espnews/?group_id=102	13.05.2011
Data	http://jasperforge.org/plugins/espnews/?group_id=112	14.05.2011
Data	http://jasperforge.org/plugins/espnews/?group_id=83	14.05.2011
Image	http://multivu.prnewswire.com/mnr/sap/37408/images/37408-hi-sales_report.png	13.05.2011
Image	http://plugins.netbeans.org/data/images/1197052333893_ireport.jpg	14.04.2011

Data	http://www.cmscritic.com/pentaho-bi-suite-3-8-puts-more-power-in-the-hands-of-end-users/	14.05.2011
Data	http://www.computerwelt.at/detailArticle.asp?a=133566&n=2	14.05.2011
Data	http://www.computerwoche.de/software/bi-ecm/1934051/index2.html	13.05.2011
Data	http://www.computerwoche.de/software/bi-ecm/2368387/	13.05.2011
Image	http://www.conbusinessintelligence.com/wp-content/uploads/2009/11/database_and_business_intelligence.JPG	15.05.2011
Image	http://www.fernstudium-fernschulen.de/assets/images/sap.jpg	15.05.2011
Data	http://www.heise.de/developer/meldung/Sapphire-Now-SAP-forciert-In-Memory-Computing-1003014.html	13.05.2011
Data	http://www.innoventsolutions.com/open-source-reporting-comparison.html	14.05.2011
Data	http://www.innoventsolutions.com/open-source-reporting-comparison.html	13.05.2011
Image	http://www.panorama-novaview.de/images/cube_medium_2.jpg	15.05.2011
Image	http://www.pentaho.com/images/rich_chart_editor.jpg	15.05.2011
Data	http://www.refocusingtechnology.com/2009/03/03/4-adv-dashboard/	15.05.2011
Data	http://www.sap.com/germany/plattform/in-memory-computing/index.epx	15.05.2011
Data	http://www.sap.com/germany/solutions/sap-crystal-solutions/index.epx	13.05.2011
Image	http://www.stonetemple.com/images/UNICA_Drill_Down.jpg	14.05.2011
Data	http://www.tecchannel.de/server/sql/1750691/business_intelligence_teil_8_bi_software_tools/index9.html	13.05.2011
Data	http://www.youtube.com/watch?v=RecSOmN34Sc	13.05.2011
Data	https://foursquare.com	15.05.2011
Image / Data	OSBI market analysis of Luc Vaillancourt of Spatialytics, 2011, Québec City, Canada	15.05.2011

8 Table of figures

Figure 44 - Ilustration of the decision making process 90
Figure 45 - Crystal Reports dashboard 91
Figure 46 - Drill down report 91
Figure 47 - Well-known OSBI-tools 92
Figure 48 - IReport from JasperSoft 93
Figure 49 - BIRT-plugin for Eclipse 103
Figure 50 - Visualization of a data cube 104
Figure 51 - Logo SAP 104
Figure 52 - Spatialytics connects BI with geo-spatial components 104
Figure 53 - BIRT Mobile Viewer IPad app 105