PRODUCT OVERVIEW

March 2015
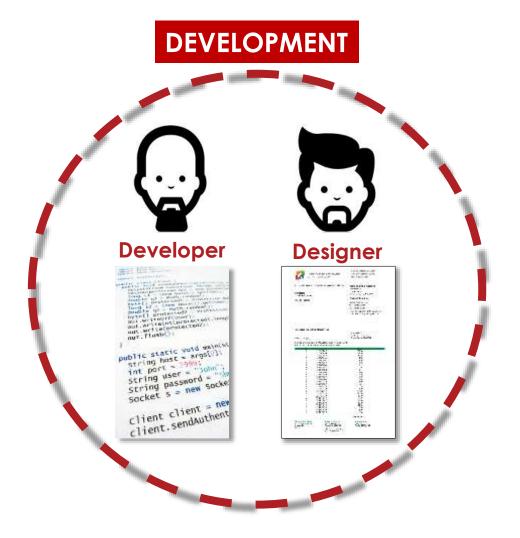
AFP support

Overlay support

Very high volume printing (e.g. billing batch in a limited time window)

Legacy reports

Mass personalization

Process connector

Support for Structured input

Application integration

Complex layouts

Impact analysis

Type safety

Dynamic Layout

Hyperion SQR, Stream Serve

SAP Reporting

GRW

Crystal, Jasper, Birt

Cognos, Business Objects

Ease of use   Ad hoc capabilities   Slice and Dice capabilties   Data mining

Graphical query generator

Desktop data sources (e.g. Excel files, csv files)

Premade designs

WYSIWYG editor and preview

Excel, HTML and RTF output

Drill down (sub reports)

Pivot tables

Drill down (OLAP)

Slicing

Dicing

Roll up

Drill across

Drill through

DEVELOPMENT

PRODUCTION

Developer

Designer
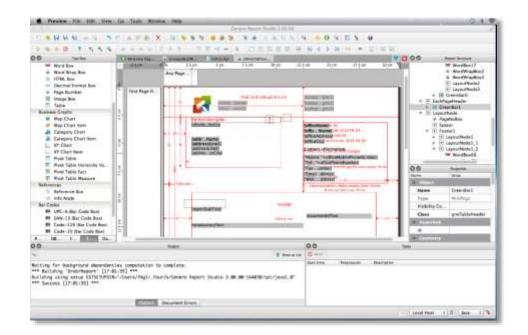
End-user

- ➢ **Genero Studio (GST)**

  Design report data & graphical layout

- ➢ **Genero Report Designer (GRD)**

  Standalone designer for end-users & developers

- ➢ **Genero Runtime Engine (GRE)**
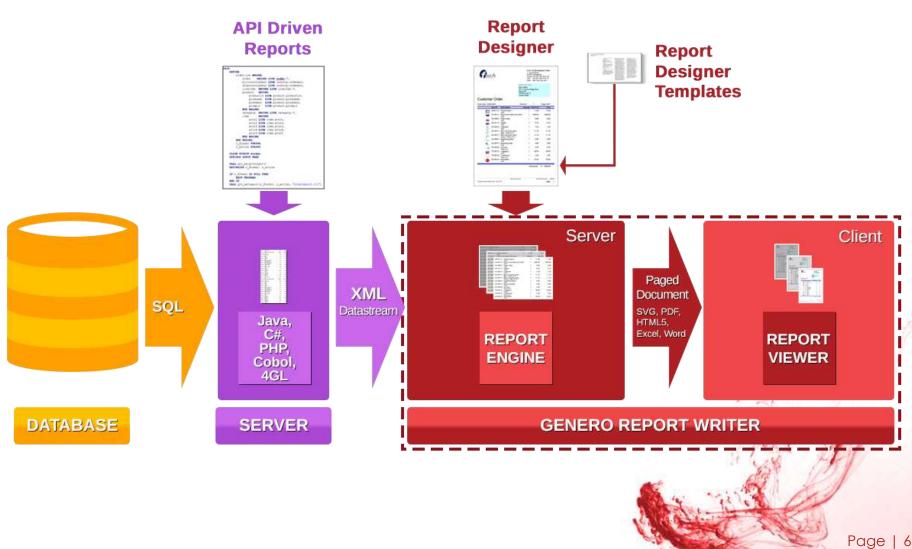
  Execute report via runtime system (DVM)

- ➢ **Enterprise reports**
- ➢ **Streaming architecture**
- ➢ **Scalable**
- ➢ **Dynamic layouting**
- ➢ **API driven**
- ➢ **Template driven**
- ➢ **Maintainable**

API Driven Reports

Report Designer

Report Designer Templates

DATABASE

SQL

Java, C#, PHP, Cobol, 4GL

SERVER

XML Datastream

Server

REPORT ENGINE

Paged Document

SVG, PDF, HTML5, Excel, Word

Client

REPORT VIEWER

GENERO REPORT WRITER

Java, C#,
PHP, Cobol,
4GL
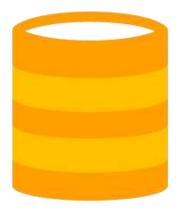
DATABASE

LANGUAGE

REPORT ENGINE

REPORT VIEWER

➤ **Streaming architecture**

- Immediate results
  - No temp tables
  - Real-time data
- Fast throughput
  - Simultaneous execution of multiple selects
  - Fast merging of multiple data sources
- Very large reports
  - Reduced resource consumption
    - Uses less disk space
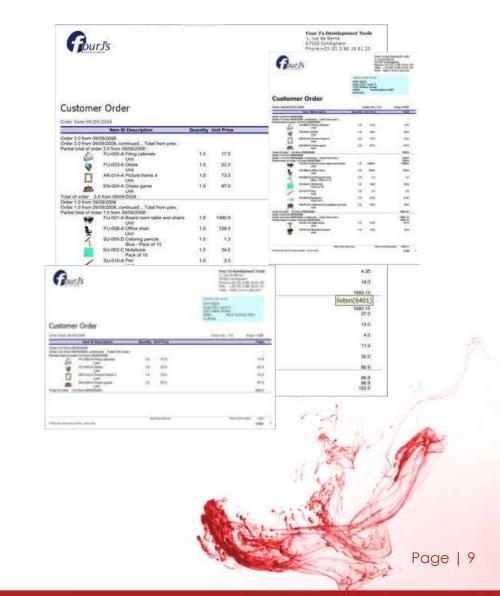   - Less memory
  - Optimised report management

➢ **Dynamic layouting**

- One report fits all
- Adapts to different:
  - Fonts
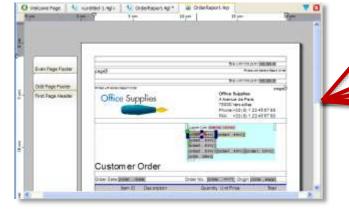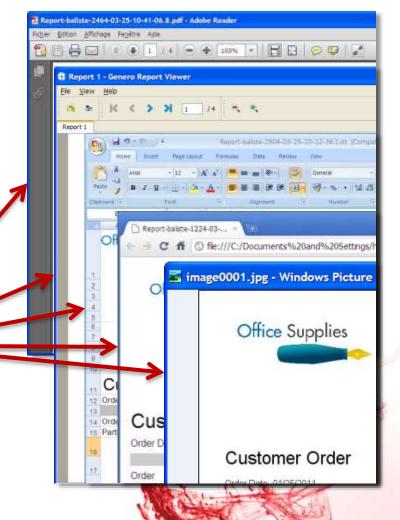 - Field values
  - International languages
  - Paper sizes and orientation

➢ **Dynamic layouting**

- One report fits all
- Adapts to different:
  - Fonts
  - Field values
  - International languages
  - Paper sizes and orientation

➢ **Dynamic layouting**

- One label fits all
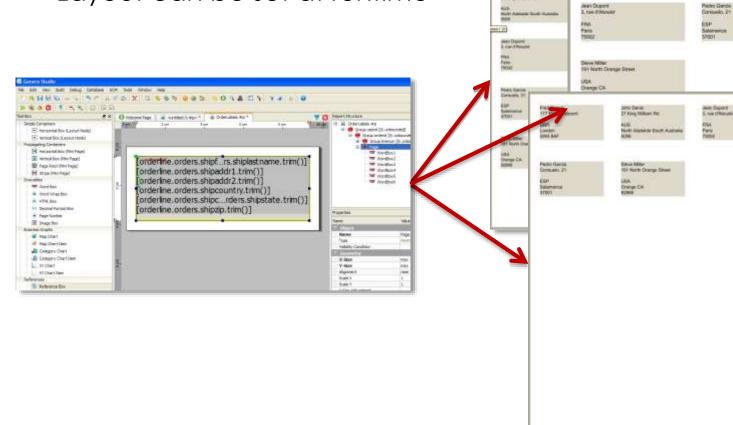- Layout can be set at runtime

- ➤ **API driven**
  - ➤ Java
    - • Supports J2SEE & J2EE,
    - • Supports JAXB,
      (Schema generation and marshalling)
  - ➤ .Net
    - • Supports Xsd.exe/XmlSerializer,
      (schema generation and marshalling)
  - ➤ PHP
    - • BAM data source generation.

➢ **API driven**

   ➢ 4GL
- Compatibility mode,
- Supports rdd/xsd,
  (via fglcomp – build rdd)
- Marshalling via REPORT object.

   ➢ Cobol
- Compatibility mode,
- BAM data source generation.

➢ **Respects existing report code**
  - Same code, new output formats.
➢ **New output formats**
  - PDF,
  - Browser,
  - MS-Excel® and MS-Word®.
➢ **Old & new reports co-exist, sharing same output formats**
  - Simplify & reduce existing code,
  - Eases migration, works immediately with existing reports,
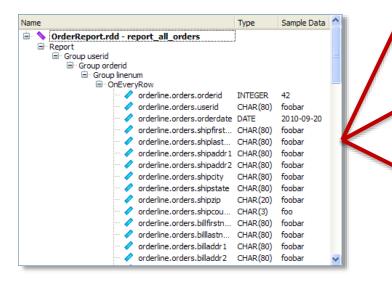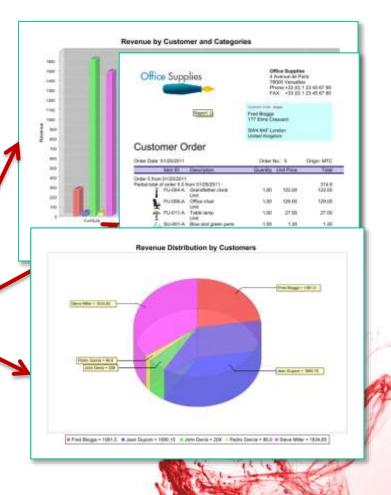 - Migrate priority reports first, embellish the rest later.

- ➢ **Multiple, simultaneous input data sources**

- ➢ **Multiple output formzats from single data source**
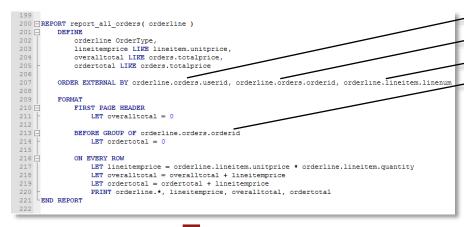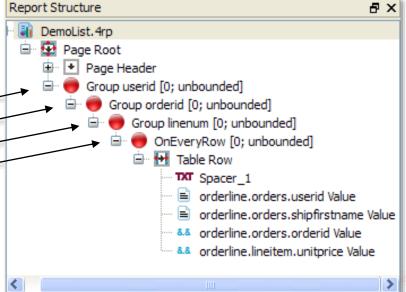 - Ex: data rendered as table or chart

Data sources – 4GL

➢ **Schema from 4GL code: REPORT**

- RDD schema produced by the compiler (fglcomp)
- PRINT, PRINTX variable
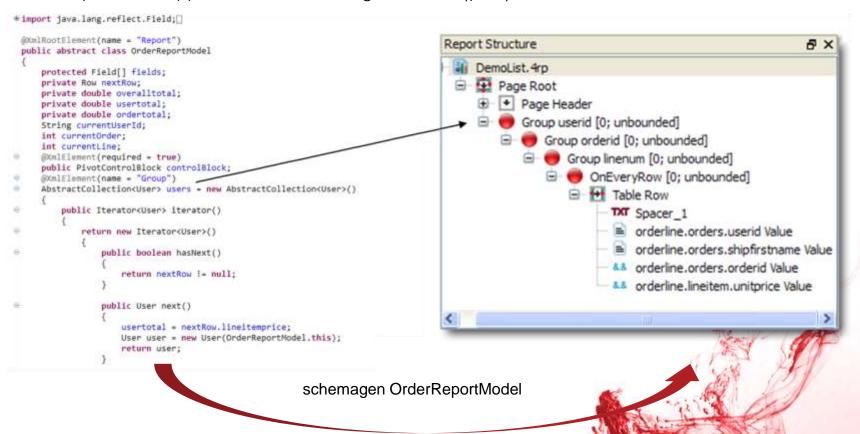- Iterators and conditional handled (FOR, FOREACH, IF…)
- ORDER BY

```
199
200 REPORT report_all_orders( orderline )
201     DEFINE
202         orderline OrderType,
203         lineitemprice LIKE lineitem.unitprice,
204         overalltotal LIKE orders.totalprice,
205         ordertotal LIKE orders.totalprice
206
207     ORDER EXTERNAL BY orderline.orders.userid, orderline.orders.orderid, orderline.lineitem.linenum
208
209     FORMAT
210         FIRST PAGE HEADER
211             LET overalltotal = 0
212
213         BEFORE GROUP OF orderline.orders.orderid
214             LET ordertotal = 0
215
216         ON EVERY ROW
217             LET lineitemprice = orderline.lineitem.unitprice * orderline.lineitem.quantity
218             LET overalltotal = overalltotal + lineitemprice
219             LET ordertotal = ordertotal + lineitemprice
220             PRINT orderline.*, lineitemprice, overalltotal, ordertotal
221 END REPORT
222
```

Report Structure

- DemoList.4rp
  - Page Root
    - Page Header
    - Group userid [0; unbounded]
      - Group orderid [0; unbounded]
        - Group linenum [0; unbounded]
          - OnEveryRow [0; unbounded]
            - Table Row
              - TXT Spacer_1
              - orderline.orders.userid Value
              - orderline.orders.shipfirstname Value
              - && orderline.orders.orderid Value
              - && orderline.lineitem.unitprice Value

fglcomp –build-rdd OrderReport.4gl

➢ **Schema from Java « POJO » object:**

- XML-Schema produced by the schema compiler (schemagen)
- Collections are recursively iterated
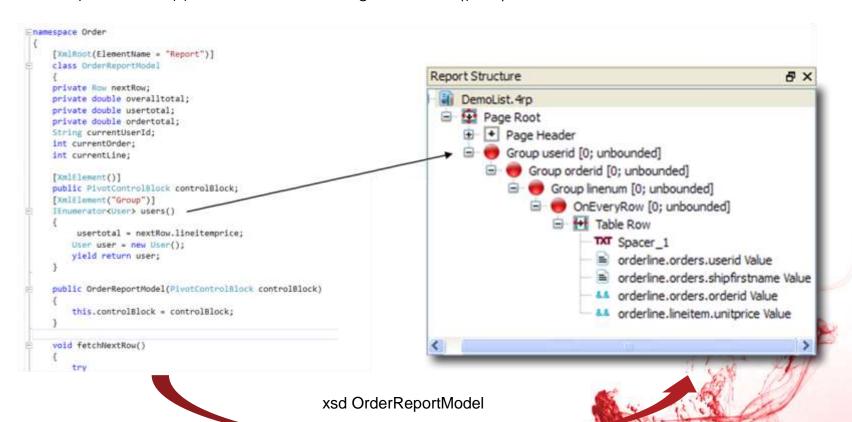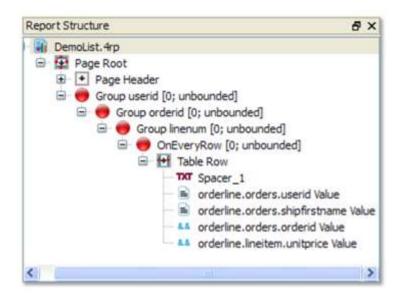- Lazy retrieval supported via Iterator using coroutines (yield)

```java
+import java.lang.reflect.Field;

@XmlRootElement(name = "Report")
public abstract class OrderReportModel
{
    protected Field[] fields;
    private Row nextRow;
    private double overalltotal;
    private double usertotal;
    private double ordertotal;
    String currentUserId;
    int currentOrder;
    int currentLine;
    @XmlElement(required = true)
    public PivotControlBlock controlBlock;
    @XmlElement(name = "Group")
    AbstractCollection<User> users = new AbstractCollection<User>()
    {
        public Iterator<User> iterator()
        {
            return new Iterator<User>()
            {
                public boolean hasNext()
                {
                    return nextRow != null;
                }

                public User next()
                {
                    usertotal = nextRow.lineitemprice;
                    User user = new User(OrderReportModel.this);
                    return user;
                }
            }
```

Report Structure

- DemoList.4rp
  - Page Root
    - Page Header
    - Group userid [0; unbounded]
      - Group orderid [0; unbounded]
        - Group linenum [0; unbounded]
          - OnEveryRow [0; unbounded]
            - Table Row
              - **TXT** Spacer_1
              - orderline.orders.userid Value
              - orderline.orders.shipfirstname Value
              - && orderline.orders.orderid Value
              - && orderline.lineitem.unitprice Value

schemagen OrderReportModel

➤ **Schema from C# object:**

- XML-Schema produced by the schema compiler (Xsd.exe)
- Collections are recursively iterated
- Lazy retrieval supported via Iterator using coroutines (yield)

xsd OrderReportModel

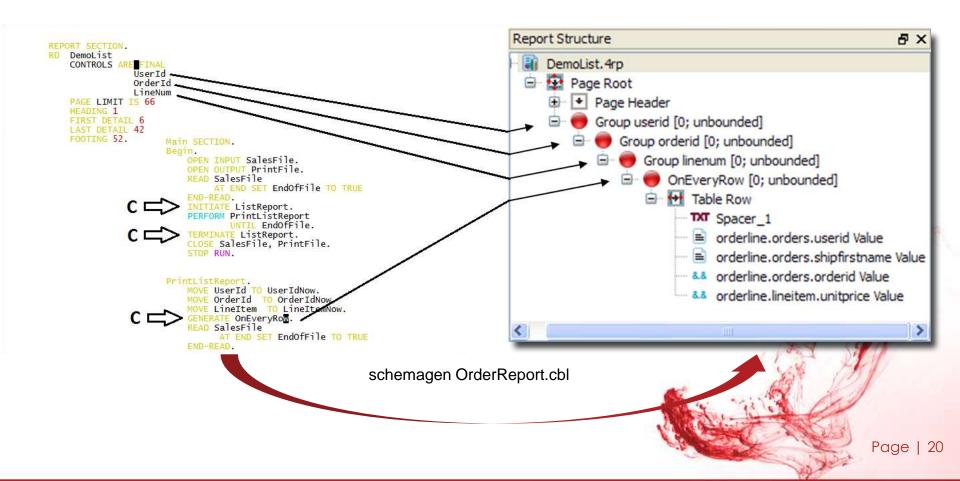➢ **Schema from BAM diagram:**
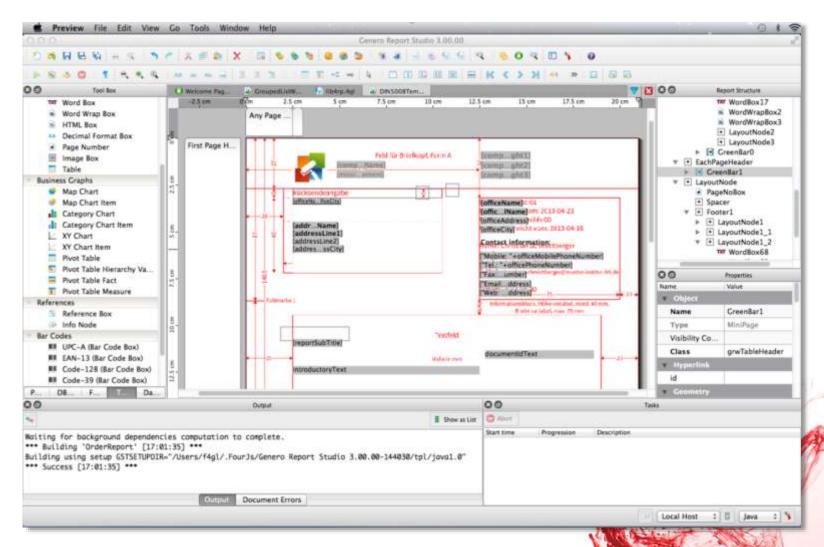
- XML-Schema produced by the schema generator

➢ **Schema from COBOL code: REPORT SECTION used**
  • XSD schema produced by schemagen compiler
  • INITIATE, GENERATE and TERMINATE instructions remapped to C-Functions
  • Groups triggers based on CONTROLS

schemagen OrderReport.cbl
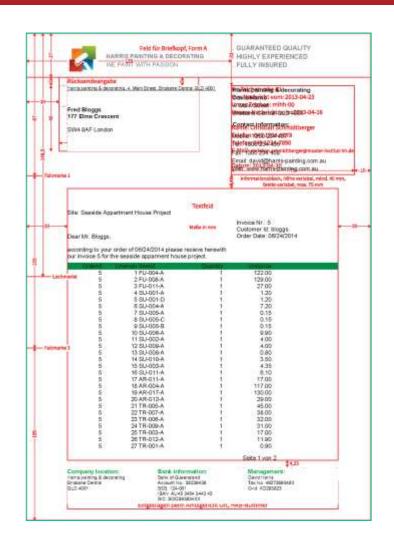
- ➤ **Template driven**
  - ➤ Create report design from templates
    - • Design (.4rp) generated from:
      - • a template and
      - • a data schema.
    - • Support for JAXB (schema generation and marshalling).
  - ➤ Used at design time
    - • Generated design can be saved for further mods.
  - ➤ Command line interface to create the design (.4RP)
  - ➤ Used at runtime
    - • Internally generated design used at runtime.
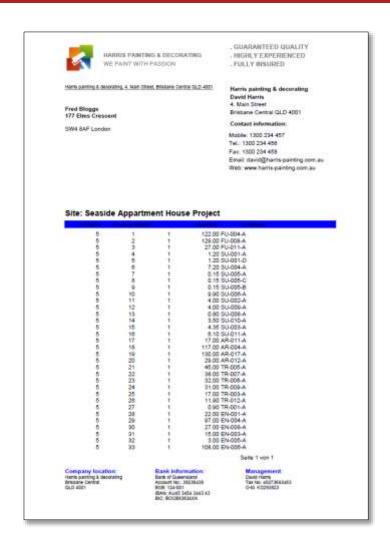
Report templates
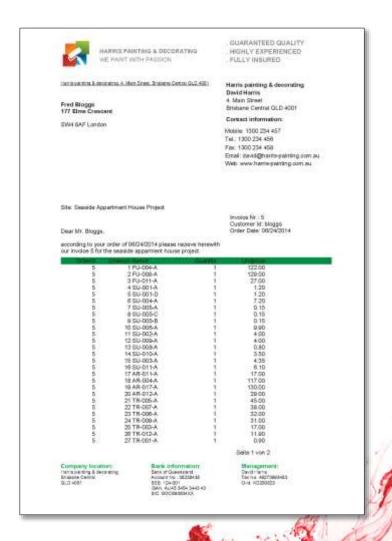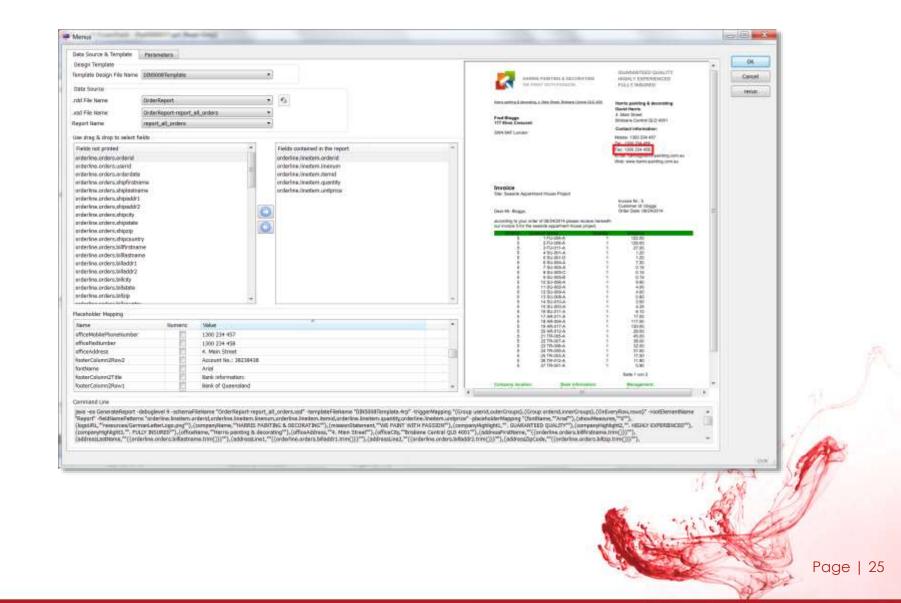
Report templates

➢ **Schema based template editing**
- Predefined schemas for specific report classes,
- Type-safe editing of templates,
- Easy conversion of reports into templates.

➢ **High flexibility**
- **Template field** toolbox object for creation of document objects based on fields,
- Placeholder substitution,
- Support for all types of data representation (e.g. from primitives, tables, pivot tables, charts, etc.),
- Support for groups and typical layouts based on groups,
- Structure of actual data and structure of data in template can differ.

➢ **Maintainability**

- Manage large numbers of reports
- Separation of report logic from presentation
  - Isolate database from external providers
  - Redecorate existing reports
- Version control
- Schema error checking
  - Type safe expressions
  - Schema modifications
  - Design, rather than runtime time discovery
- Template driven

- **Pixel exact rendering**
  - Exact output as GRV,
  - Text is declared as strings keeping the document small.
- **Creation of WOFF fonts on-the-fly**
  - Supports all fonts seen by GRE (Type1 and TrueType),
  - Fonts are cached and can be shared between documents,
  - Good performance also for Asian fonts.
- **Streaming (progressive) viewing and printing**
  - Support of random navigation in unfinished reports.
- **Support for very large documents**
  - Works well on low bandwidth and/or high latency connections.
- **Reports can be bookmarked and shared via URLs**

- Calendar
- CategorychartByAreasAndCategories
- CategorychartRevenueByCategoriesAndCustomers
- CategorychartRevenueByCustomersAndCategories
- DynamicPivotTable
- GenericList
- GroupedTableDemo
- ListDemo
- MapchartRevenueByCategories
- MapchartRevenueByCustomers
- MapchartRevenueByCustomersAndCategories

- MasterReport
- OrderLabels
- OrderList
- OrderReport
- OrderReportASCII
- OrderReportASCII4
- OrderStock
- StaticPivotTable
- TableDemo
- XYChart
- Chinese Report
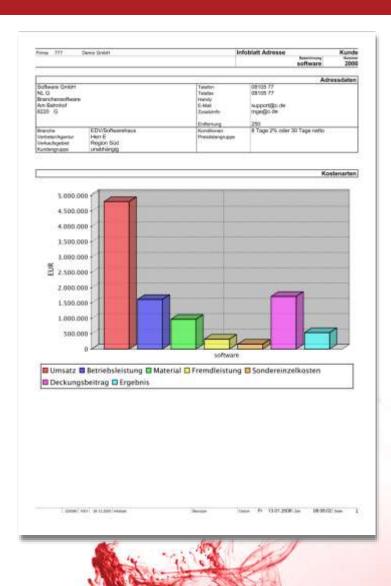
➢ **Report Engine**

- Graphic rendering of text reports
- Quick and easy data layout
- Advanced layout
    - Position & size data elements
    - Avoid data truncation
    - Translation & internationalization
    - Design single report for multiple paper sizes

➢ **Report Engine**

- XML streamed output

- Low memory consumption

- "Print-as-you-go"
  - Rapid 1st, 2nd, 3rd… nth page delivery

- Print very large reports
  - No need for temp files
  - Merge data/blobs from
  - multiple sources

- Relative or absolute positioning of elements
  - Headers/footers on page breaks
  - Horizontal/vertical box propagation

➢ **Hand coded data sources**

- Allows reusing existing objects (e.g. JPA/Hibernate classes)
- Enables custom data sources (e.g. web service).
- Requires schema-from-code utility and marshaller to be workable (Available for 4GL, Java and .Net).
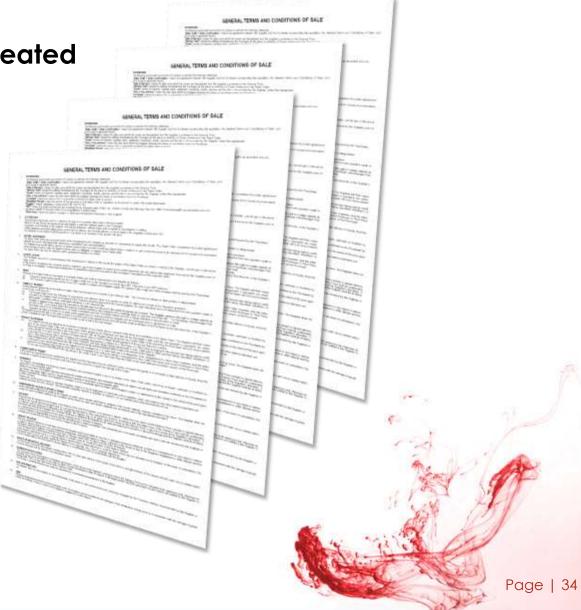
➢ **Generated data sources (via. BAM)**

- Abstracts language specific details
- Works for languages without schema compiler (e.g. PHP, C, COBOL)
- Enables the creation of identical data sources in different languages.

➢ **Efficient printing of repeated verso pages**

➢ **Output formats:**

Excel

Word

PDF

HTML

SVG

➢ **Excel output**

- Output any REPORT to Excel

- Design Excel reports from Report Designer

- Graphical elements rendered as images

➤ **Excel output**

- Output any REPORT to Excel
- Design Excel reports from Report Designer
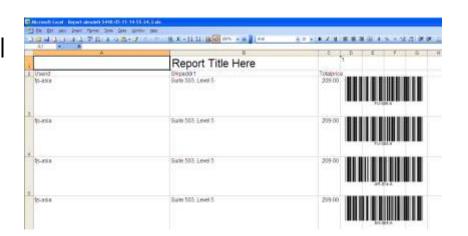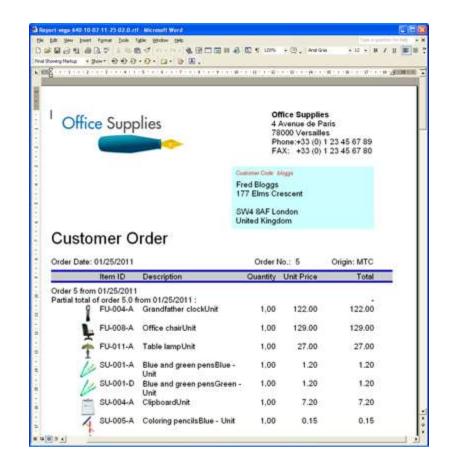- Graphical elements rendered as images

➢ **Excel output**

- Output any REPORT to Excel
- Design Excel reports from Report Designer
- Graphical elements rendered as images

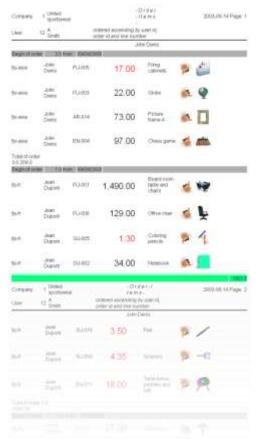➢ **Page 'n of M'**
- 'M' page count for all pages or sub-ranges
- Display total number of pages 'M' on each page
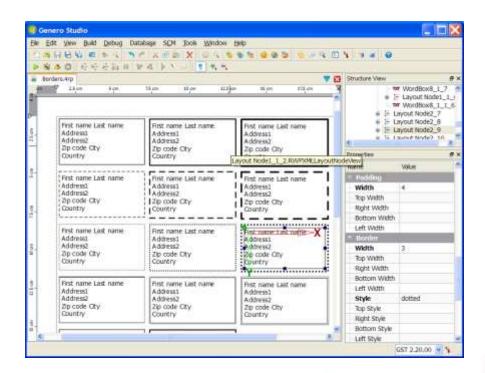
➢ **Borders and padding**

- Border style & thickness
- Padding:
  - Defines outer space
- Marging:
  - Defines inner space
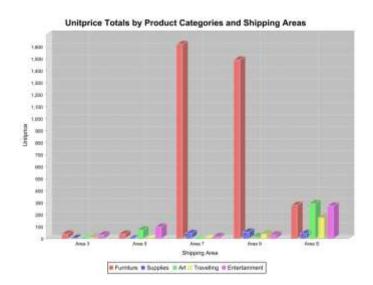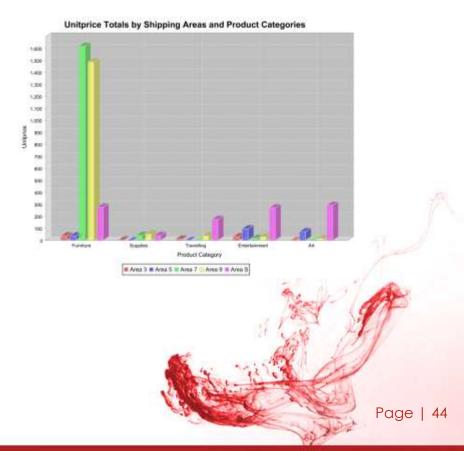
- **XML data sources**
    - XML as data
    - XML schema(or DTD) as report data definition
- **Report Designer**
    - Dataview shows XML schema
    - Report design based on XML schema
- **Third party product integration will use XML data sources**

- ➢ **Charts with multiple dimensions,**
- ➢ **Table with fixed column role types (measure & dimension),**
- ➢ **Table rows known as 'facts'.**

➢ **Table object manipulates columns, lines and cells for easier formatting and alignment**
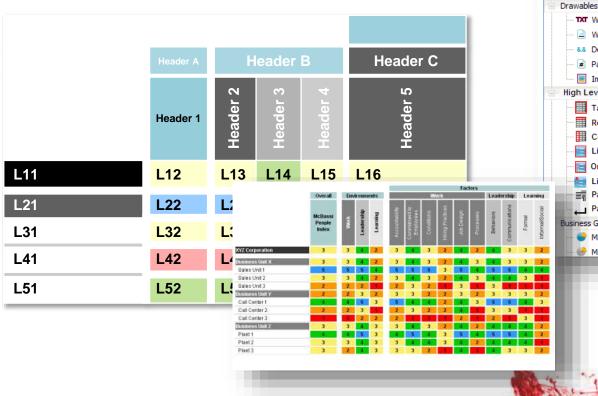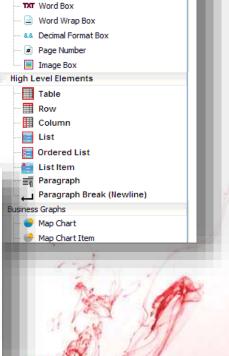
Hello World: width="3cm"

http://www.4js.com

Hello World: size not specified
(default error correction (3))

➢ **Sub-reports**

    Produce REPORT data using sub-reports

    Design a single graphical representation

➢ **Templates**

    Store part of the graphical representation in a Template library

    Re-use elements of the template library (aggregation)

    Parameterized template elements

    User defined regions in template elements

➢ **Sub-reports**

REPORT data using sub-reports
Design a single graphical
representation

Three column sub-report in a 'detail' section. When more data is added, the sub-report will stretch the section downwards adding rows but keeping three columns across.

Requisition Spending by Section - Current

Department: ADP 30300 Mgmt.

Section: 30300000 MGMT.

Requisition Type	Budget	Total $
Police Detail for Road	$0.00	$1,500.00
Cell Phones	$20,000.00	$3,758.15

Summary for 'ReqDept' 30300000 (2 detail records)

Sum $5,258.15

Section: 30304000 SAFETY

Requisition Type	Budget	Total $
Safety data sheets	$0.00	$998.00

Summary for 'ReqDept' = 30304000 (1 detail record)

Sum $998.00

Department: ADP 33300 Grounds

Section: 33304000 Grounds

Requisition Type	Budget	Total $
Heavy Equipment Repair	$95,000.00	$2,057.00
Paint Booth VOC Tracking Soft	$1,100.00	$550.00
State Inspection Stickers	$1,250.00	$850.00
Stock	$10,750.00	$604.76
Tires	$45,000.00	$2,281.28

Summary for 'ReqDept' = 33304000 (5 detail records)

Sum $6,343.04

Thank you

QUESTIONS ?