

A Domain-Specific Visual Language for Report Writing

Using Microsoft DSL Tools

- Ruskin Dantra, John Grundy and John Hosking

Overview

- Introduction
 - Domain
 - Problem
- Background
 - Motivation
- Approach and design
- Evaluation
- Future work

Domain Introduction

- In conjunction with Prism
 - MIS for the printing and graphics industry
- Prism exposes a proprietary reporting language via which end-users can query their database and display the results in a visual form
- Language is called RWL for the purpose of this paper
- Procedural, “sort-of” object-oriented, interpreted programming language

Background and Motivation

- **RWL is complicated**
 - Implicit semantics
 - Complicated enterprise database
 - **No design time validation**
 - **No dedicated IDE**
 - Lack of context sensitive help
 - **Ease change management**
 - Time to market
- ```

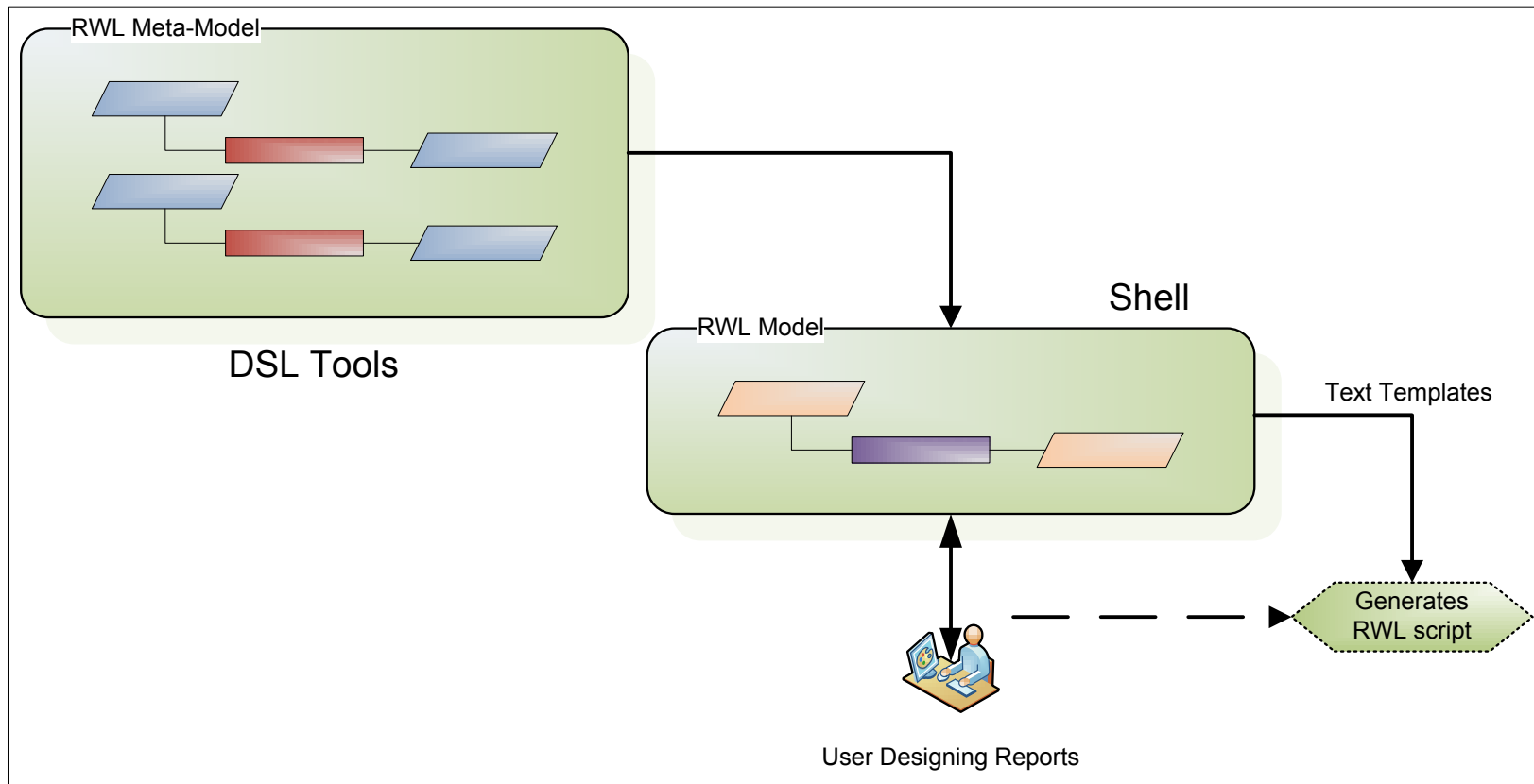
Code CASE STUDY 1
Type Standard
Access
- Scan RM
Print RM_CUST + RM_NAME;
Print "All Jobs For " + RM_NAME;
- Scan QM
Choose (QM_CUST_CODE, MATCH, RM_CUST)
Choose (QM_QUOTE_JOB, MATCH, QMM_JOB)
Print QM_JOB_NUM + QM_TITLE;
End
End
Print StandarReportFooter;

```

## Why a VL?

- Visual aid, cues and context sensitive help
  - Visual DSL
- Only expose “absolutely necessary” information
- Minimize user errors by allowing them to “design” reports rather than write them

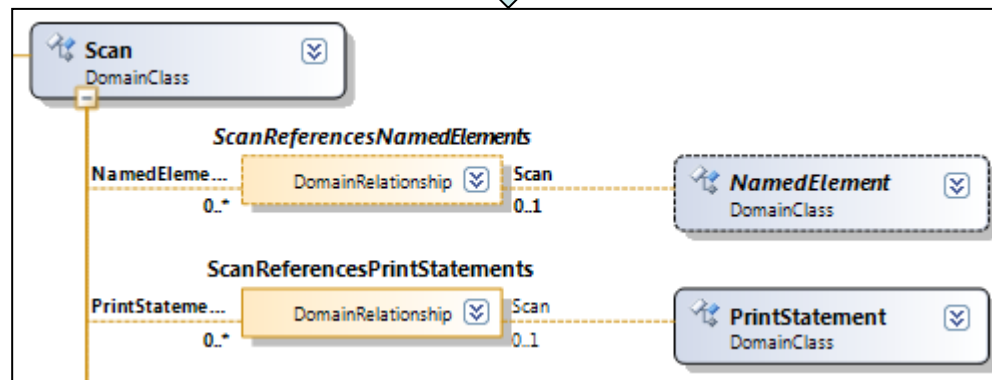
# Approach overview



# Our Approach

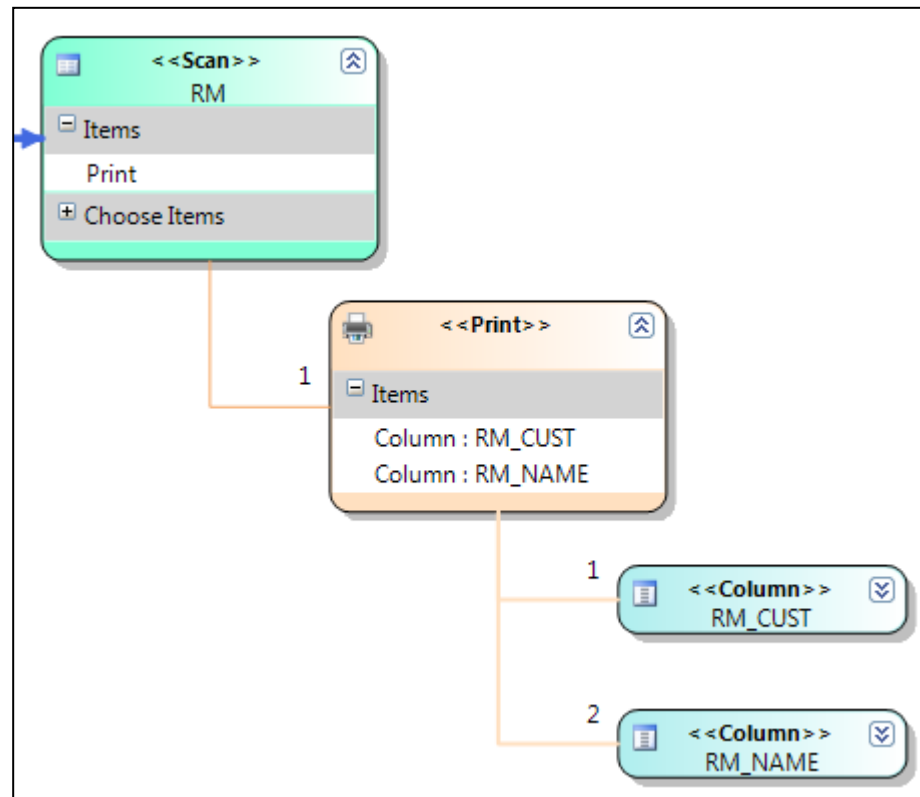
1. Reverse engineer meta-model from RWL specification

```
Scan RM
 Print RM_CUST + RM_NAME;
End
```



## Our Approach (Cont'd...)

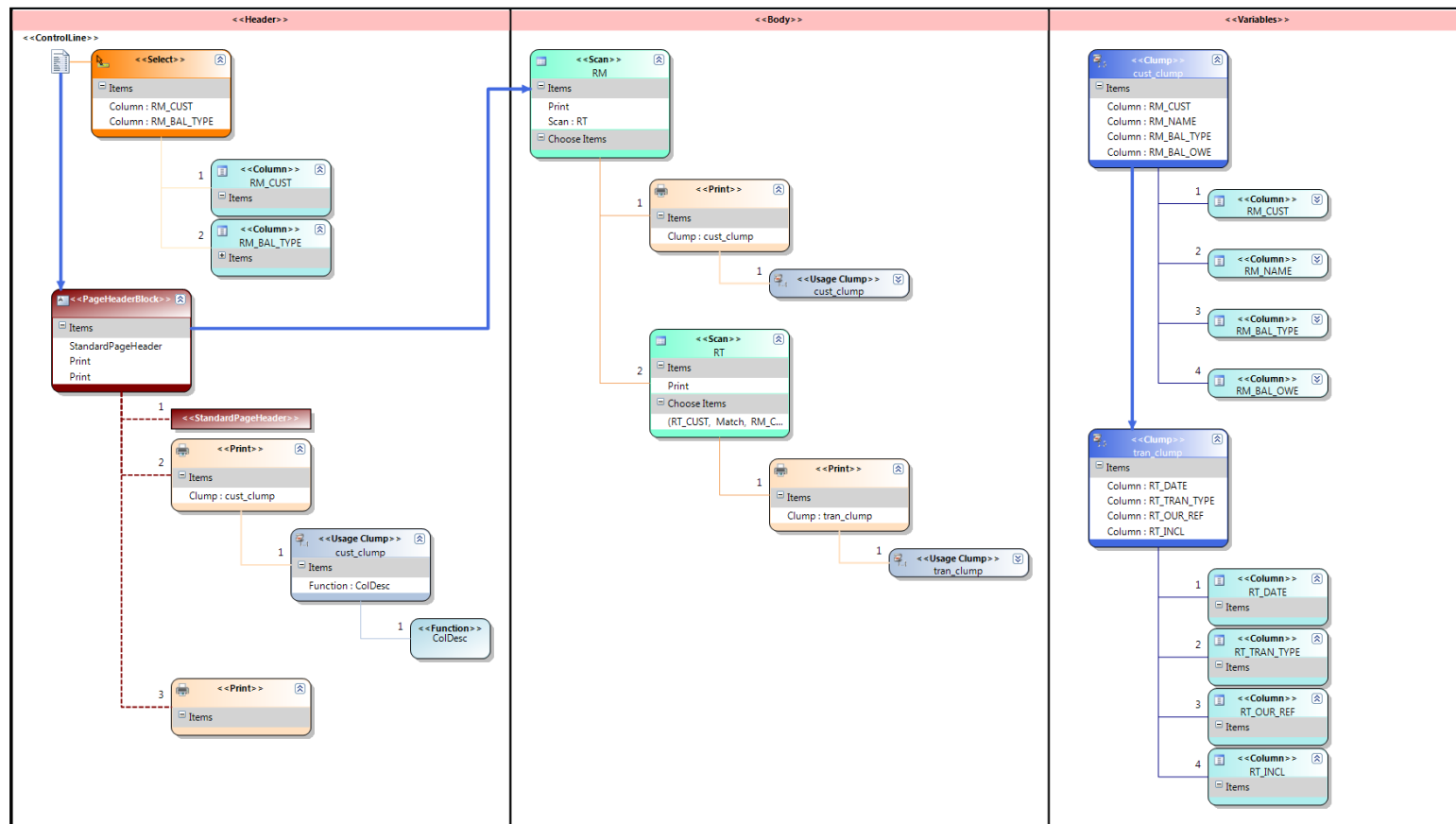
2. Design a VL using surface level notation





# Our Approach (Cont'd...)

## 3. Allow end-users to create entire RWL models

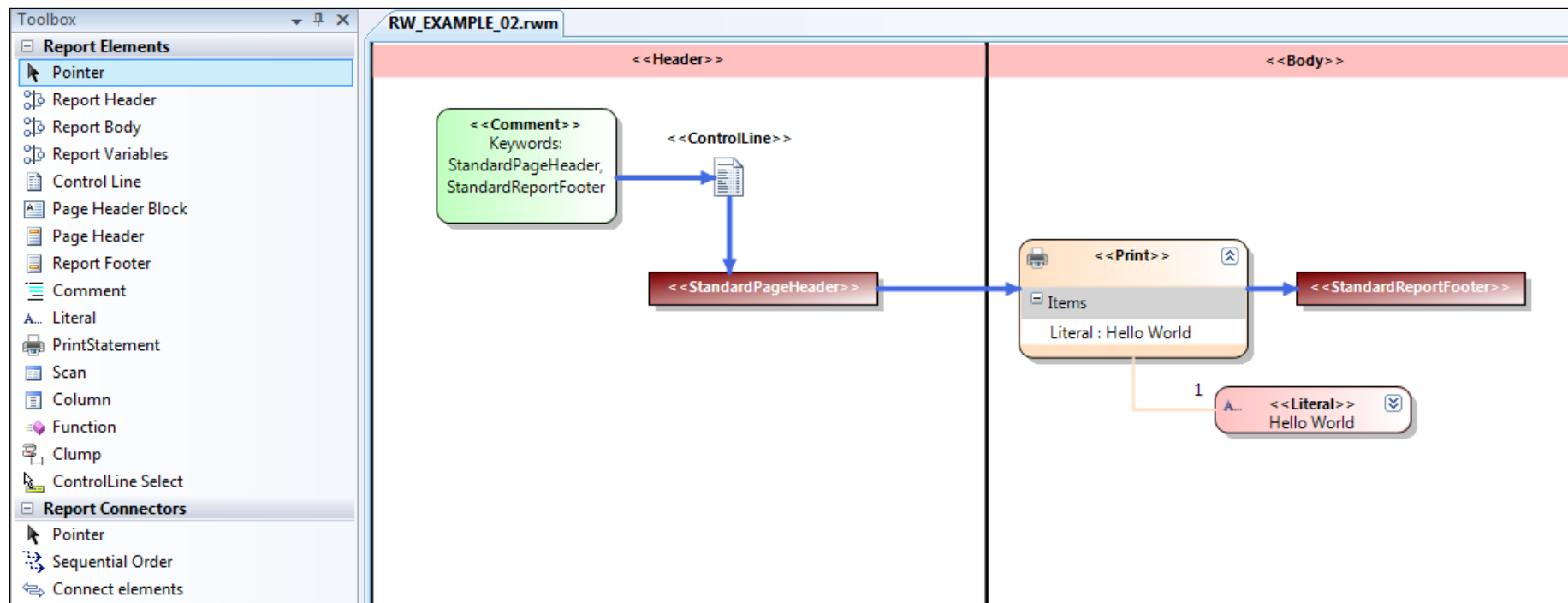


# Our Approach (Cont'd...)

## 4. Automated RWL script generation from RWL model

```
1 //-----
2 // <auto-generated>
3 // This code was generated by a tool.
4 //
5 // Changes to this file may cause incorrect behavior and will be lost if
6 // the code is regenerated.
7 //
8 // Generated on 16/09/2009 11:06:29 p.m.
9 // ToolVersion 1.0.0
10 // </auto-generated>
11 //-----
12
13 Code RW_EXAMPLE_07
14 Type Standard
15 Access STSR
16
17 // variables
18 Clump tran_clump = RT_DATE + RT_TRAN_TYPE + RT_OUR_REF + RT_INCL;
19 Clump cust_clump = RM_CUST + RM_NAME + RM_BAL_TYPE + RM_BAL_OWE;
20
21 Select RM_CUST + RM_BAL_TYPE;
22
23 PageHeader
24 Print StandardPageHeader;
25 Print cust_clump.ColDesc;
26 Print;
27 End
28
29 Scan RM
30 Print cust_clump;
31 Scan RT
32 Choose (RT_CUST, Match, RM_CUST)
33
34 Print tran_clump;
35 End
36 End
37
```

# Simple Example



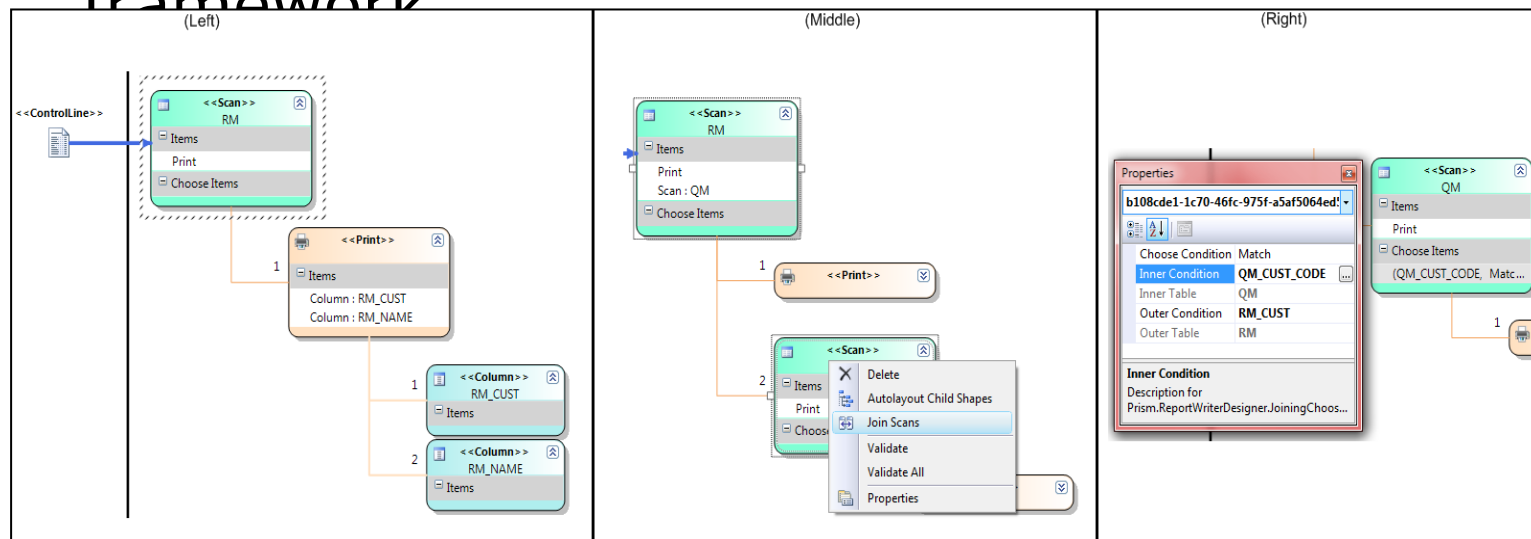
Error List

4 Errors 1 Warning 0 Messages

|   | Description                                             | File              | Line | Column |
|---|---------------------------------------------------------|-------------------|------|--------|
| 2 | ReportHeader ReportHeader1 has no ControlLine.          | RW_EXAMPLE_02.rwr | 0    | 1      |
| 3 | Comment(Comment1) is unreachable.                       | RW_EXAMPLE_02.rwr | 0    | 1      |
| 4 | StandardPageHeader(StandardPageHeader1) is unreachable. | RW_EXAMPLE_02.rwr | 0    | 1      |
| 5 | PrintStatement(PrintStatement1) is unreachable.         | RW_EXAMPLE_02.rwr | 0    | 1      |

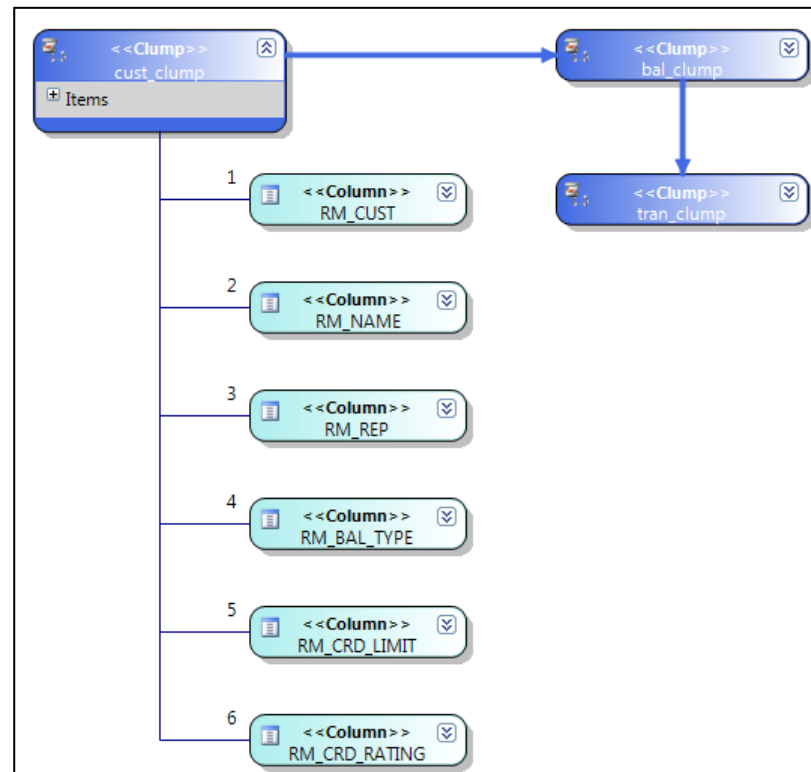
# Evaluation – Design

- Evaluated using the Cognitive Dimensions framework



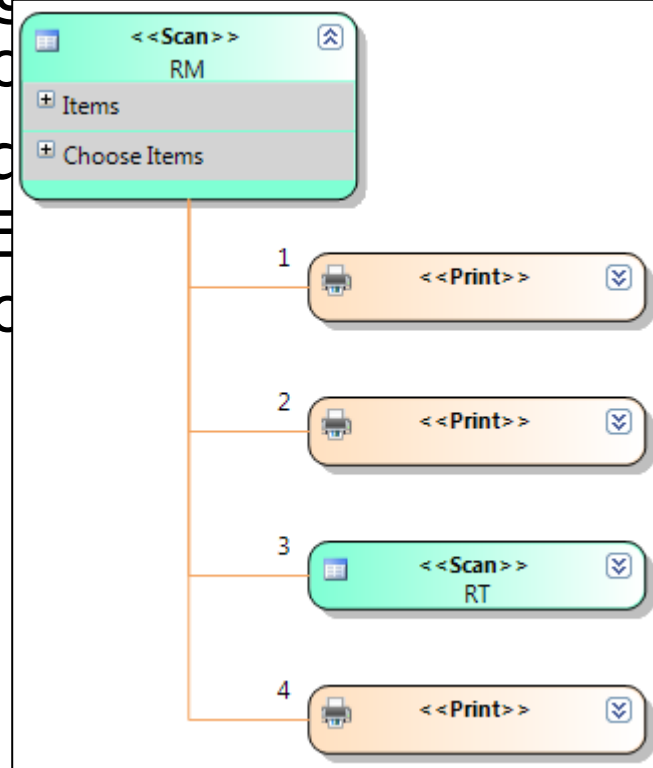
## Evaluation – Design (Cont'd...)

- Auto layout and simple refactoring capabilities reduce viscosity



## Evaluation – Design (Cont'd...)

- Reducing textual entry from users reduces error probability
- Hidden components can be used by doing simple relationships using elements



## Evaluation – Survey

- 14 participants
- Six developers and eight non-technical end-users
- Both groups given two task: an easy task and a comparatively harder task

## Evaluation – Survey (Cont'd...)

- End-user survey
  - Novice and intermediate users found tool useful
  - Experienced users felt a bit constrained
  - Scalability was raised as an issue
    - At what point does a visual report become harder to understand than a textual report?
  - Making small changes require comparatively more steps
    - E.g. Adding a simple *print* statement



## Evaluation – Survey (Cont'd...)

- Developer survey
  - Tasks were easy and procedural with automated code generation
  - Scalability was raised as an issue
    - Will the meta-model become so large than maintenance will be difficult?

## Future Work

- Improve VL
- Improve auto-layout algorithms
- Versioning
- Wizards and code snippets