# Automated Data Mapping Specification via Schema Heuristics and User Interaction

Sebastian Bossung[1], Hermann Stoeckle[2], John Grundy[2], Robert Amor[2] and John Hosking[2]

[1]Software Systems Group,
Technical University of Hamburg, Germany

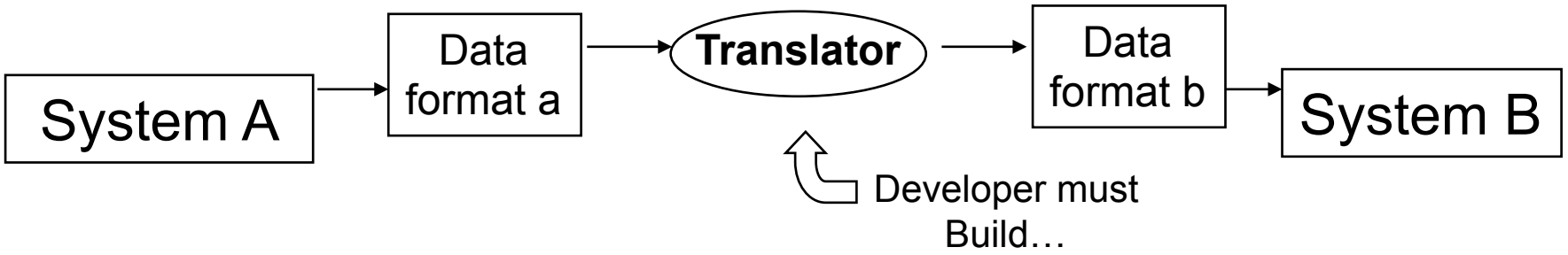
[2]Department of Computer Science,
University of Auckland, New Zealand

# Outline

- Motivation
- Mapping Tool Requirements
- Our approach: mapping agents & user control
- Mapping Agents – heuristics & examples
- VisAXSM – mapping visualization & examples
- Evaluation
- Conclusions & Future work

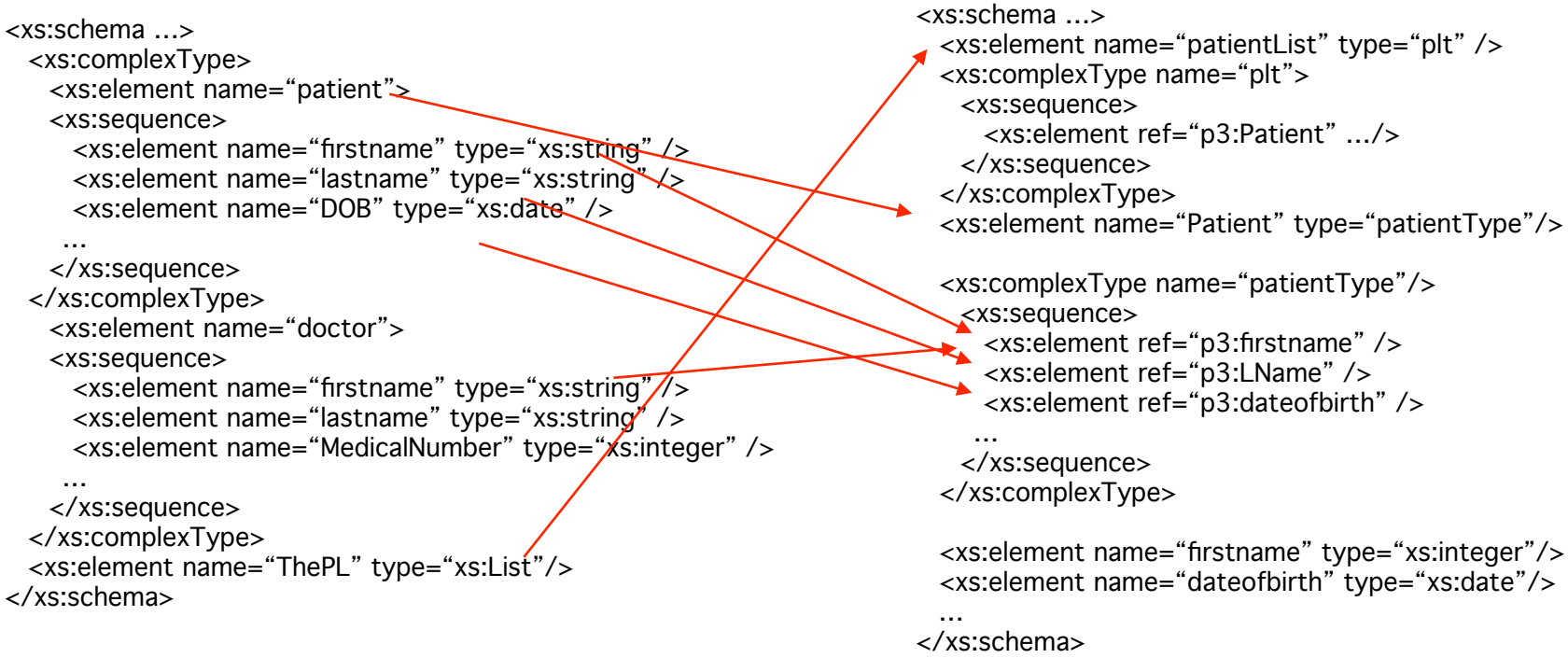
# Motivation

- Data mapping is a major problem for system integrators




- Development of data translator very tedious, time consuming and error prone
- But our experience is that many mappings could be "automatically" determined
- Idea: a new tool for translator generation
  - uses "mapping agents" to determine "obvious" correspondences between XML schema elements…

# An Example

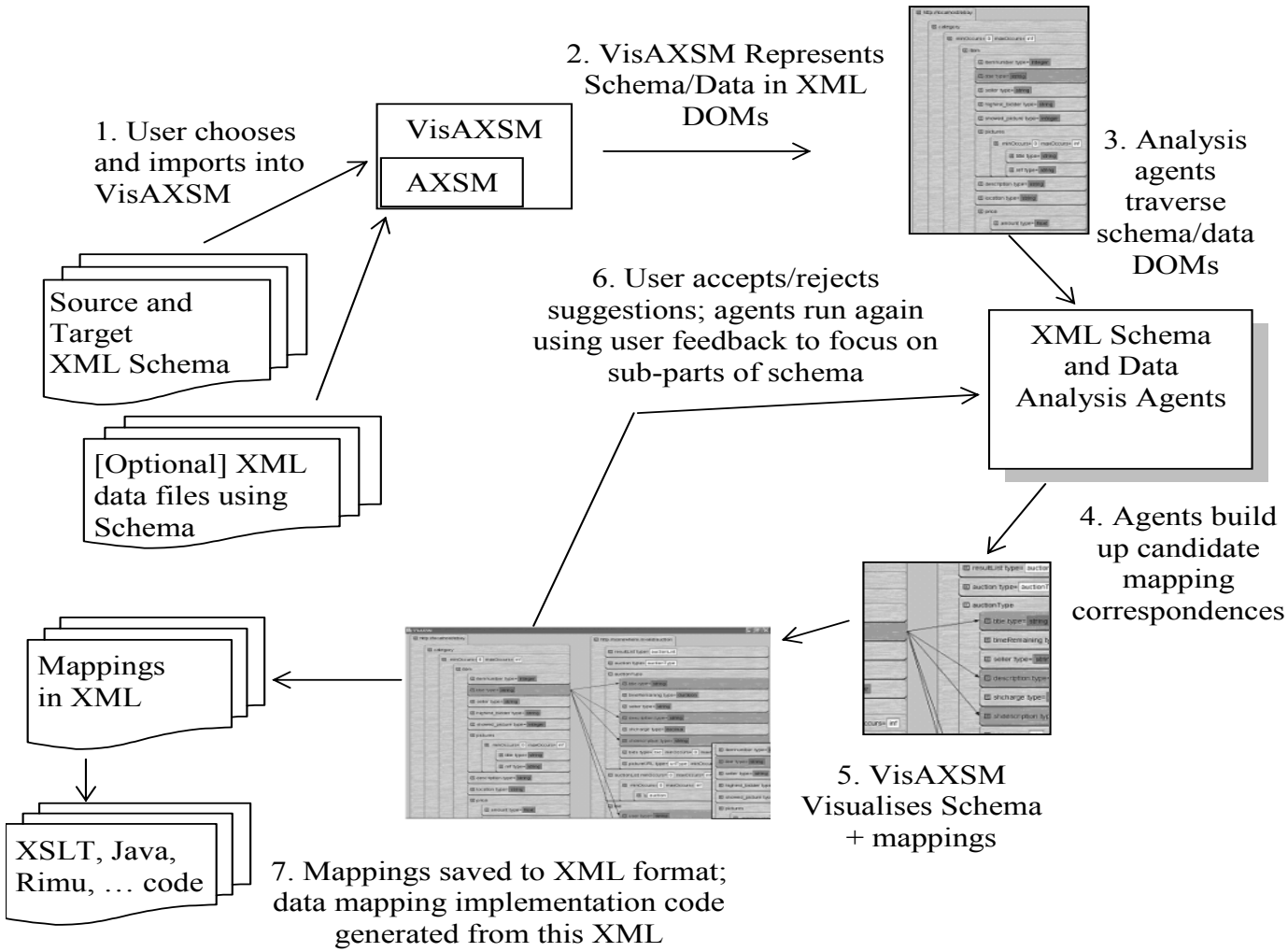



- "firstname" maps to "firstname" ("Same Name" match)
- "patient" maps to "patientType" ("Partial Name" match)
- "lastname" maps to "LName" ("Similar Name" match)
- "DOB" maps to "dateofbirth" ("Synonym" match)
- "firstname" maps to ?? ("Same Name" match, multiple context)
- "ThePL" maps to "PatientList" (user-defined mapping)

# Requirements for "VisAXSM"

- Able to load two XML Schema definitions
  - the data formats to map between
- Automatically traverse the two schemas and suggest correspondences (mappings) between schema elements
- UI must allow the user to focus on schema sub-parts
- Able to suggest correspondences
  - users can accept or reject suggestions
  - the system uses these responses to control further traversal and suggestion generation
- Extensible via new plug-in:
  - "mapping agents" (suggesters) and
  - "renderers" (schema/mapping element visualizations)
- Able to generate a translator from the set of accepted schema mapping correspondences
  - e.g. in XSLT, Java etc

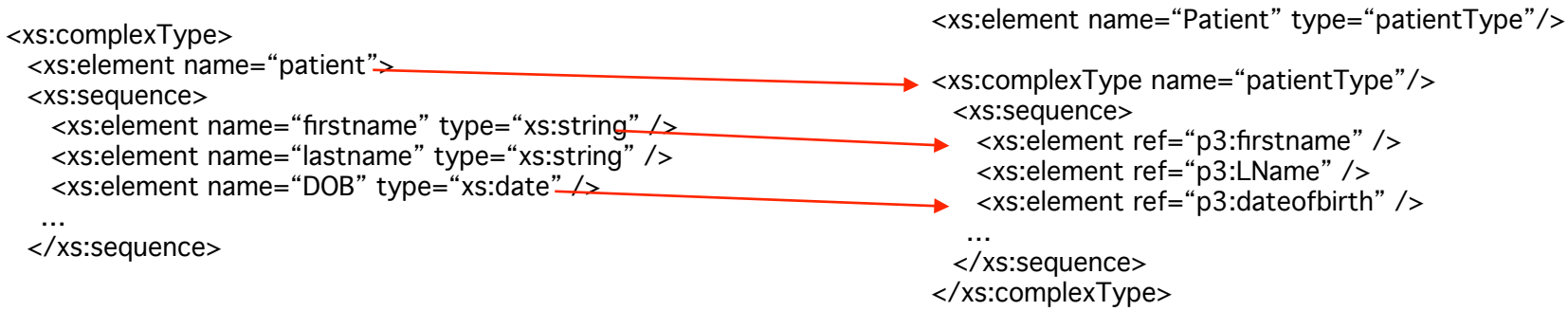
# Our Approach



# Mapping Agents

- Name matchers
  - Exact Name Matcher
    Elements have the same names in the source and the target schema (also does case-insensitive matches)
  - Partial Name Matcher
    One element name is the substring of a element name in the other schema
  - Levenshtein Name Matcher
    Computes the "Levenshtein distance" of two element names, i.e. number of edit operations to convert from one name to another
  - Synonym/Acronym Matcher
    Uses domain knowledge base to identify synonyms in the source/target schema element names

# Example #1




Same Name matcher:

- Easy to do, but will find lots of false +ve unless closely constrained

Partial Name matcher:

- Looks for sub-strings; can have less false +ves (usually…)

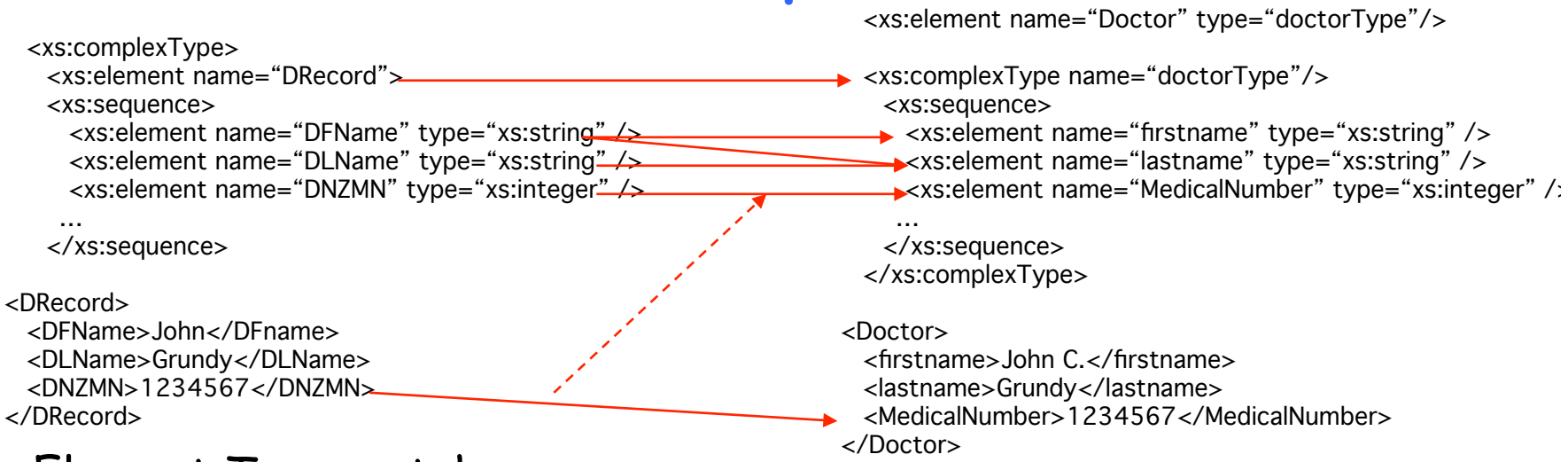
Synonym matcher:

- Needs knowledge base – may be generic or domain-specific

# Other Mapping Agents

- Type matchers
  - Element Type Matcher
    Compares the data type of elements
  - Record Type Matcher
    Compares record element sub-types
- Data matchers
  - Exact Data Value Matcher
    Looks for data value matches in source/target schema XML data files
  - Partial Data Value Matcher
    Looks for partial data value matches in XML files

# Examples




Element Type matcher:

- Easy to do, but must constrain part of schema heavily unless "unusual types" being compared

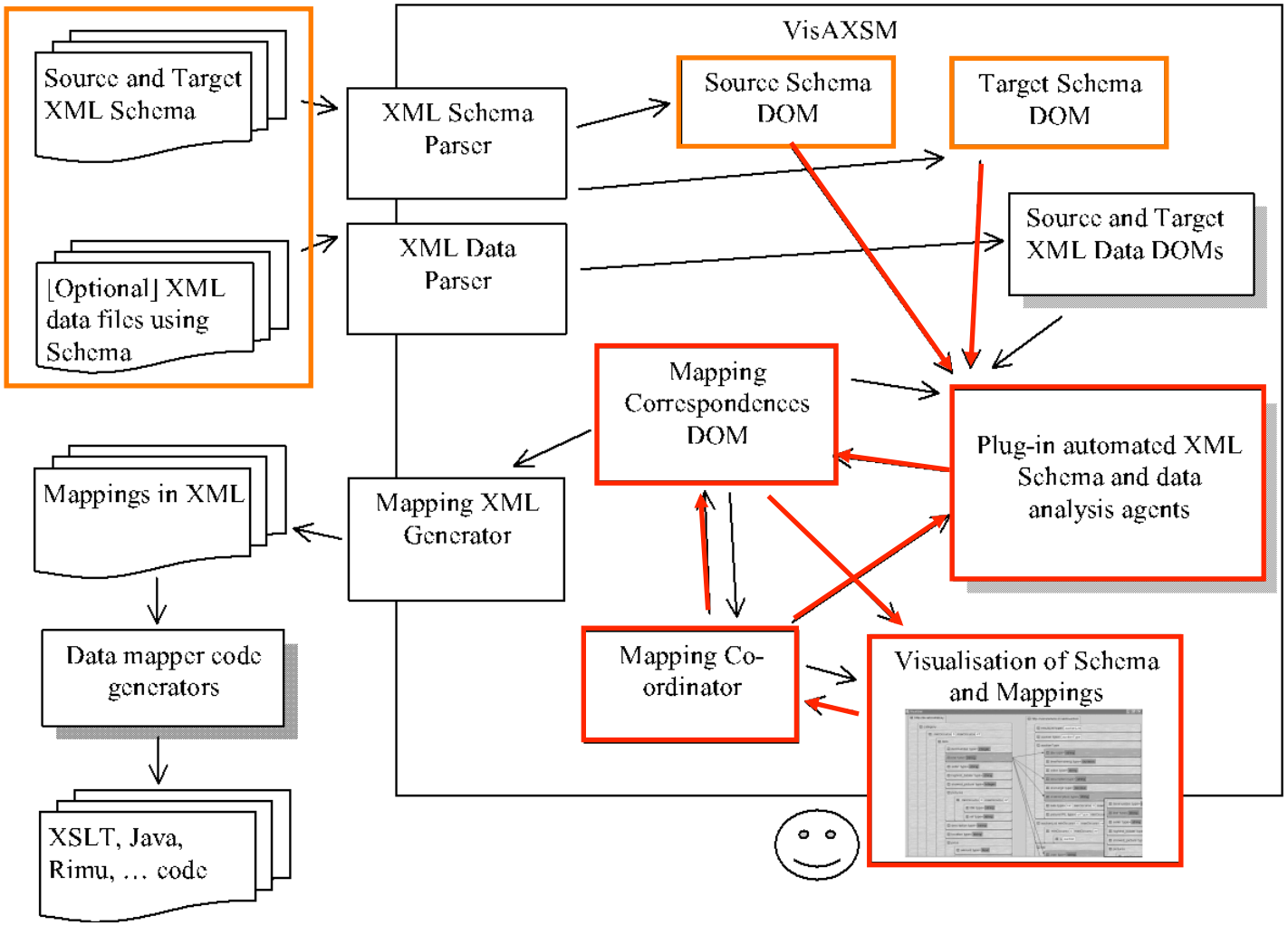
Record Type matcher:

- Need to look at record sub-types, and for partial matches of types to types. Expensive if over large part of schema

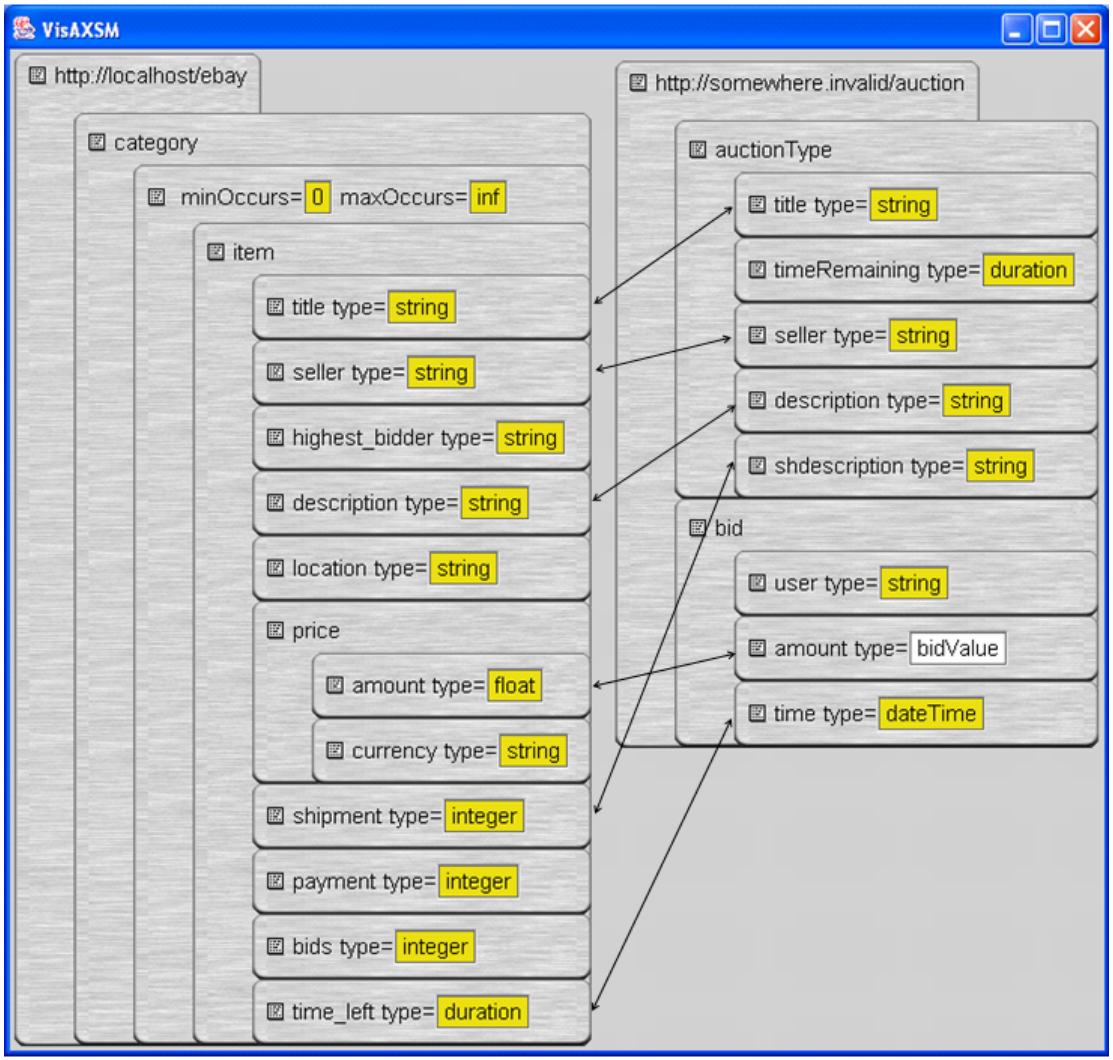
Exact Data Value matcher:

- Need example XML data for source/target; search for data values that match; can inform other matchers

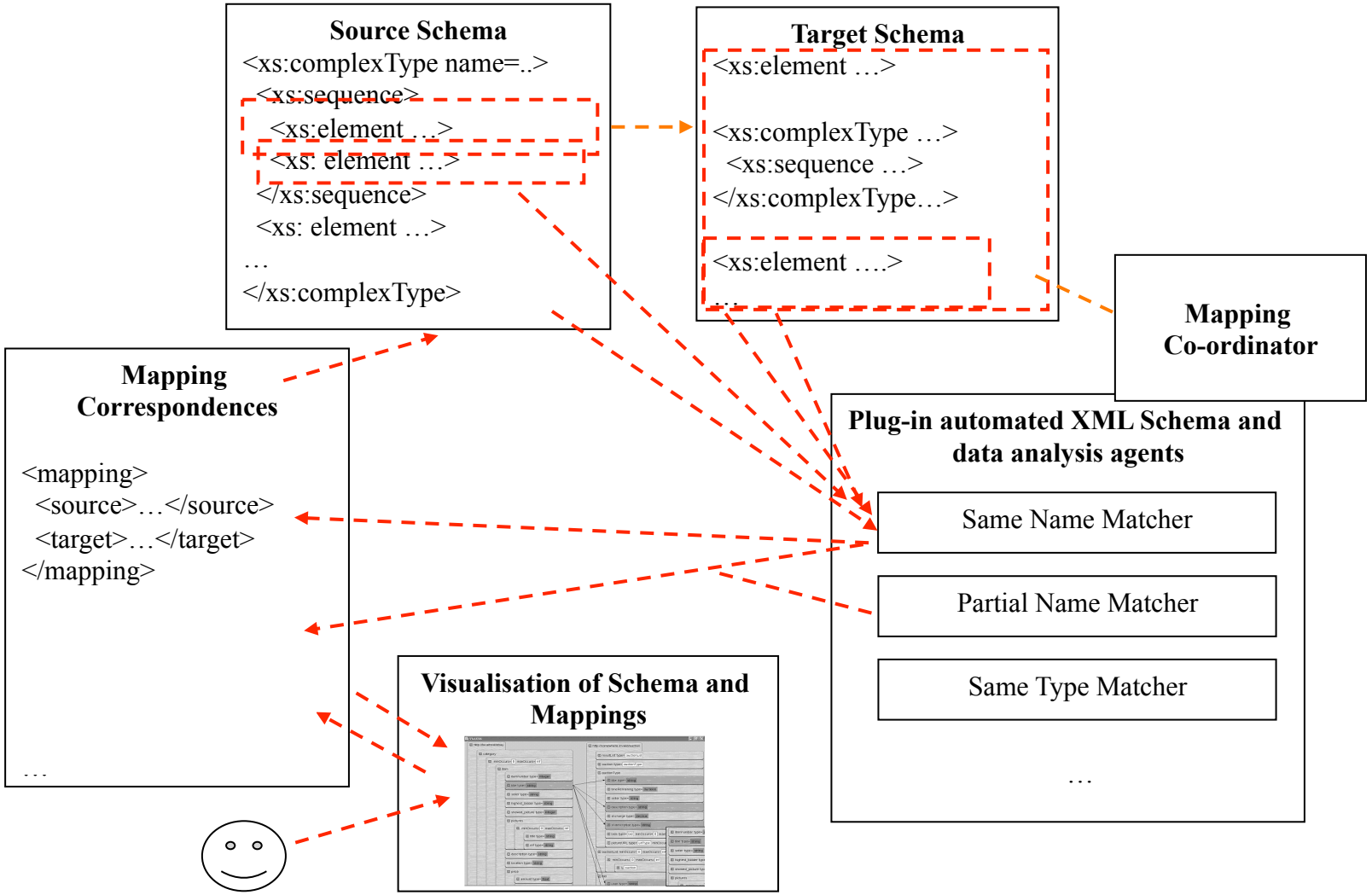
# VisAXSM IDE


VisAXSM
Source and Target XML Schema
[Optional] XML data files using Schema
XML Schema Parser
XML Data Parser
Source Schema DOM
Target Schema DOM
Source and Target XML Data DOMs
Mapping Correspondences DOM
Plug-in automated XML Schema and data analysis agents
Mapping XML Generator
Mappings in XML
Data mapper code generators
XSLT, Java, Rimu, … code
Mapping Co-ordinator
Visualisation of Schema and Mappings

# Example of using VisAXSM



# User, Mapping Agent and VisAXSM Interaction



# Features of VisAXSM

- Import XML Schema, XML data files
- Build up XML mapping correspondences data structure
- Elements in tree can be hidden, collapsed and expanded
- Tree displays are implemented as set of renderer plug-ins – tree, node, left, name/type, arrow, interactors (accept, reject, suggestions)
- Set of extensible mapping agent plug-ins
- Provides different filters on the schemas:
  - Show elements which are (not) resolved
  - Show elements with (no) suggestion
  - Auto-map if threshold of (set of) matching agents high

# Evaluation of VisAXSM

+ Can reduce the search space quickly after assigning high-level correspondences
+ Extensible set of agents and renderers which can have domain-specific knowledge
+ Ability to hide/show complex schema incrementally very useful
+ Even if only a few suggestions provided by agents, provides "nice" IDE to specify mappings manually
~ Problems with auto-generated names
  ~ except the data matchers can give some information
- Very different structures cause problems – few commonalities in tag names, types etc

# Conclusions/Future Work

- Implementing data mapping translators is hard
- VisAXSM uses combination of schema heuristics and user interaction to provide semi-automated specification and fully-automated generation

- Various future research in progress:
  - Extend the existing mapping agents with more sophisticated mapping heuristics
  - Provide plug-in (domain-specific) renderers for Schema elements/inter-schema mapping links
  - Integrate visual expression for more complex mapping (merging, converting,…) using VisAXSM's renderer plug-ins

# References


- Grundy, J.C, Hosking, J.G., Amor, R., Mugridge, W.B., Li, M. Domain-specific visual languages for specifying and generating data mapping system, Journal of Visual Languages and Computing, vol. 15, no. 3-4, June-August 2004, Elsevier, pp 243-263
- Grundy, J.C., Mugridge, W.B., Hosking, J.G. and Kendal, P. Generating EDI Message Translations from Visual Specifications, In Proceedings of the 16th International Conference on Automated Software Engineering, San Diego, 26-29 Nov 2001, IEEE CS Press, pp. 35-42.
- Grundy, J.C., Mugridge, W.B., Hosking, J.G. and Kendal, P., A Visual Language and Environment for EDI Message Translation, In Proceedings of Human-Centric Computing 2001, IEEE CS Press.
- Li, Y., Grundy, J.C., Amor, R. and Hosking, J.G. A data mapping specification environment using a concrete business form-based metaphor, In Proceedings of the 2002 International Conference on Human-Centric Computing, IEEE CS Press.
- White, P. and Grundy, J.C. Experiences Developing a Collaborative Travel Planning Application with .NET Web Services, In Proceedings of the 2003 International Conference on Web Services, Las Vegas, June 23-26 2003.
- Stockle, H., Grundy, J.C. and Hosking, J.G. Notation Exchange Converters for Software Architecture Development, ESEC'2003 Workshop on Tool Integration, September 2003.
- Stoeckle, H., Grundy, J.C. and Hosking, J.G. Approaches to Supporting Software Visual Notation Exchange, In Proceedings of the 2003 IEEE Conference on Human-Centric Computing, Auckland, New Zealand, October 2003, IEEE CS Press.
- Amor, R., Augenbroe, G., Hosking, J.G., Rombouts, W., Grundy, J.C., Directions in Modelling Environments, Automation in Construction, Vol. 4 (1995), Elsevier Science Publishers, 173-187.