



Analysis on Inference Mechanisms for Schema-driven Forms Generation

“From XSD to XForms”

Josef Spillner (josef.spillner@tu-dresden.de),
with Alexander Schill

Berliner XML-Tage, Berlin, Germany, 25.09.2007



01 Scenario

Dynamic web service invocation

- Service discovery and selection
- Conventional use: code generation
- Interactive use: ad-hoc invocation
 - WSDL handling
 - XSD handling
 - UI handling

01 Scenario

Web Services Graphical User Interfaces (WSGUI)

- Inference mechanisms
- Implicit semantic hints
- Explicit GUI hints
- Layout and embedding
- Applications built on services

02 Background

Inference basics

- Inference (our vague definition): to transfer information derived from schema domain into visual representation
- Schemas describe data structures, types and constraints
- Declarative GUI descriptions are about structured information retrieval and presentation
- Model-driven approach: Generate GUIs from data schemas
- Properties:
 - type safety
 - completeness, schema exploitation
 - usability, perfection

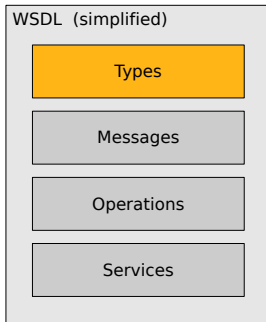
02 Background

Specific inference domain

- XML Schema
 - describes XML grammar
 - includes type system
- XForms
 - describes user interfaces for forms
 - data processing is based on XML Schema (model, instance)
- Goal: schema-exploiting XForms representation for interactive instance production

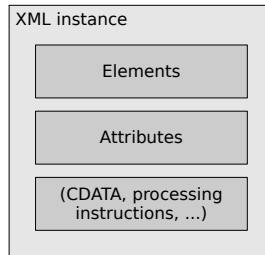
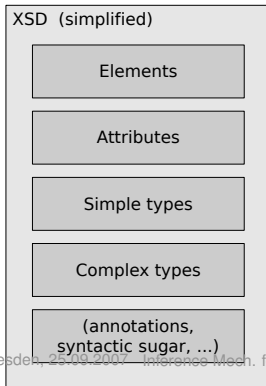
02 Background

Tackling the Scenario: WSDL processing



02 Background

XSD processing



03 Categorisation

Analysis of XForms type support

- Type dependencies
 - unrestricted input fields (xs:string and derivatives)
 - numeric input or ranges (xs:decimal and derivatives)
 - date input (e.g. xs:dateTime)
 - upload of encoded binary data (e.g. xs:base64Binary)
 - checkboxes (xs:boolean)
 - URL input fields (xs:anyURI)
- XForms has type-bound widgets
 - some inference power remains unused (e.g. xf:range with stepsize 1, xf:select1)
 - not all widgets can be inferred (e.g. xf:secret, xf:textarea)
- Initial smart instance creation

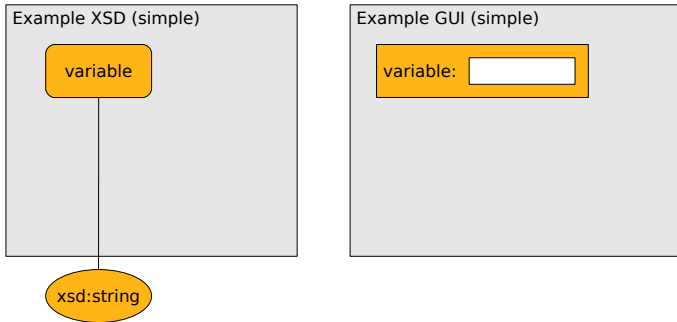
03 Categorisation

Inference issue list

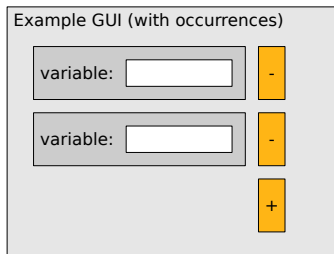
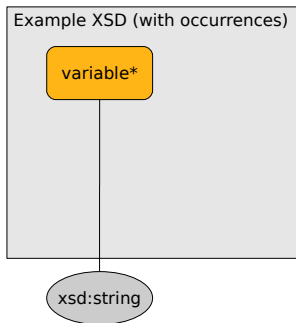
- Enumerations (xs:enumeration -> xf:select1)
- Ranges (xs:{min,max}{In,Ex}clusive -> xf:range)
- Unions (xs:union)
- Regular expressions (xs:pattern)
- Simple lists (xs:list)
- Complex lists/occurrences
- Restricted lists

04 Examples

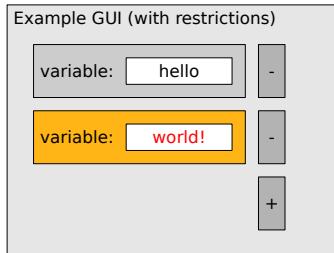
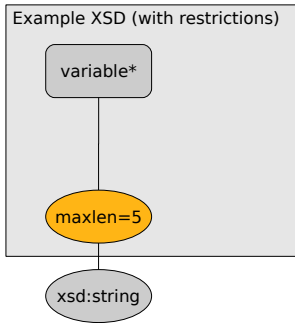
Seeing the inference mechanism in action



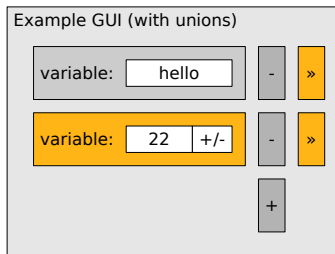
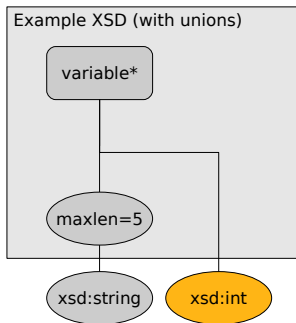
04 Examples



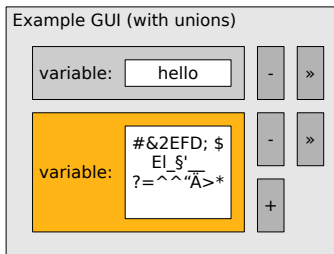
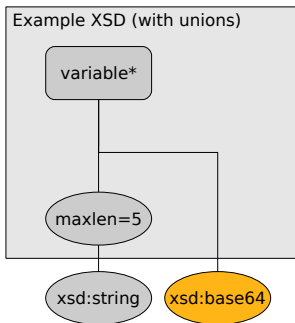
04 Examples



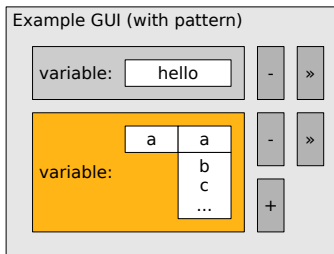
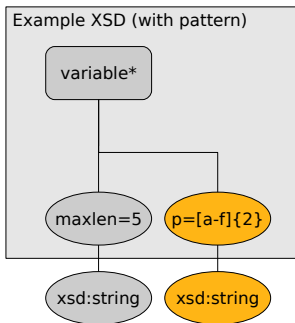
04 Examples



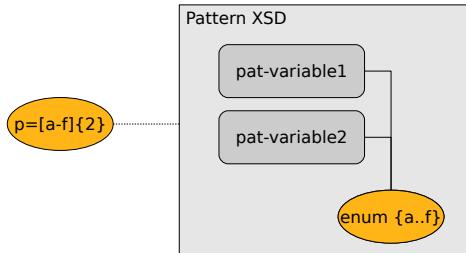
04 Examples



04 Examples



04 Examples



05 Mapping

Simple type constraints

- Numerical constraints
 - `xs:{total,fraction}Digits`: XForms
 - `xs:{min,max}{In,Ex}clusive` -> `xf:range`
- String constraints
 - `xs:[{min,max}]Length`: XForms
 - `xs:whiteSpace`: XForms
 - user-toggle
- Enumeration constraints
 - `xs:enumeration` -> `xf:select1`
- Unions (`xs:union`)
 - user-toggle



05 Mapping

User-toggle facility

- Toggle between `xf:input`, `xf:secret`, `xf:textarea`
- Toggle between union constituents
- Realisation
 - `xf:choice`

05 Mapping

Lists

- Simple lists (xs:list) and mapping
 - one widget per item, with `relevant="false"`
 - one visible widget for aggregation
 - `calculate="concat(..w1,..w2)"`
 - limits: only works for fixed-size lists or known upper boundary
- Complex lists
 - `xf:repeat`
- Restricted complex lists
 - shadow model needed in XForms
 - each complex list element converted to complex type
 - each such type amended with two elements for addition/removal buttons
 - restriction by binding buttons to occurrences
- XForms 1.1 removes several complex list limitations

05 Mapping

Regular expressions

- Similarities
 - schema describes grammar of XML
 - regex describes grammar of a string
- Mapping
 - transformation of regex to schema
 - rendering the schema
 - using `xpath:concat` to combine values again
- Improved user experience
 - `\d{5}-\d{8}`



05 Mapping Namespaces

- Many schema processing applications get them wrong
- Incorrect: rely on prefix parts
- Correct: only honour namespace parts
- Improvement: honour prefix suggestions
- Usage in web services especially crucial

06 Results

- Advantages of auto-generated GUIs: type safety, schema exploitation
- Further review needed: smart instance data, user influence
- Implementation available: Dynvoker XFormsAdapter based on XSD4J and RegExpInstantiator
- Issues: evolving XSD and XForms specs; immature XForms implementations
- Suggestion: initial instantiation and more inference within XForms

Implementation:

<http://dynvocation.selfip.net/xsd4j/>
<http://dynvocation.selfip.net/dynvoker/>