

NVDL— a Breath of Fresh Air for Compound Document Validation

Petr Nálevka

Jirka Kosek

University of Economics, Prague
*Dept. of Information and Knowledge
Engineering*

University of Economics, Prague
*LISp (Laboratory for Intelligent Systems
Prague)*

petr@nalevka.com

jirka@kosek.cz



<http://jnvdL.sourceforge.net>

NVDL—a Breath of Fresh Air for Compound Document Validation

This presentation is available at:

<http://www.nalevka.com/xtech2007.pdf>



<http://jnvdl.sourceforge.net>

NVDL— a Breath of Fresh Air for Compound Document Validation

- Agenda
 - Compound documents
 - Validation of compound documents
 - Relax NG, XML Schema
 - NVDL Introduction
 - Validation dispatching process
 - Why is NVDL better?
 - NVDL implementations, JNVDL
 - JNVDL extensions, versioning
 - NVDL in use



Compound Documents

- CD = XML document, made of elements and attributes from different mark-up vocabularies
- Language extensions – Monolithic X Compound
- Example: The Web / HTML evolution story
 - Monolithic → Modularized → Compound



Compound Document Examples

- Templating languages
 - XSLT, JSP, ...
- XML protocols
 - SOAP, ...
- Office documents
 - ODF, ...
- Semantic Web
 - RDF → RDFS → OWL
- Web
 - xH (XHTML + SVG + MathML)
- Other
 - XInclude
 - XLink
 - XML default attributes
 - xml:lang



Compound Document Validation

- Validation = interoperability
- Compound documents are more difficult to interpret than standalone documents
- Example: RDF in XHTML, eligible context
- **NO ADOPTION WITHOUT VALIDATION!**



Namespace aware schema languages

Relax NG, XML Schema

- Namespace support = Compound document validation
- Relax NG – CD constructs
 - ns="namespace" attribute
 - anyName, nsName
 - Modularity support
 - Definition: interleave, choice
 - externalRef



Namespace aware schema languages Problems!

- Example: schema for XHTML + RDF + SVG + MathML
 - 1 Download XHTML DTDs and convert them to Relax NG
 - 2 Create definitions for `head`, `inline` and `block` context within the schema
 - 3 Download schemas for RDF, SVG and MathML
 - 4 In case the schemas are in a different language than Relax NG, they need conversion
 - 5 Converted schemas need further adjustments to make them modules of the XHTML parent schema



Namespace aware schema languages Problems!

- Problems
 - Incorporating a new language
 - May require adjustments in the parent language schema
 - Requires a special module to be created for the new combination of languages
 - Changing the parent language (e. g. from XHTML to SVG)
 - Requires large-scale changes across all involved schemas
 - The parent schemas need to be duplicated



Namespace aware schema languages Problems!

- To conclude:
 - Existing schemas cannot be reused as they are
 - Conversion, adjustment
 - Duplicating schemas is error prone (propagating changes)
 - Difficult to create new compound schemas
 - Extra knowledge needed (schema languages, implementation details)
 - Compound language constraints not separated from individual vocabulary constraints
 - Difficult to read and understand



NVDL

- Namespace-based Validation Dispatching Language
 - “Part 4 of ISO/IEC 19757 DSDL” (Document Schema Definition Languages) international standard
- NVDL is not new (evolution)
 - Relax Namespace 2001 (RNS)
 - Modular Namespaces 2003 (MNS)
 - Namespace Switchboard 2003 (NSSB)
 - Namespace Routing Language 2003 (NRL)
 - NVDL 2006

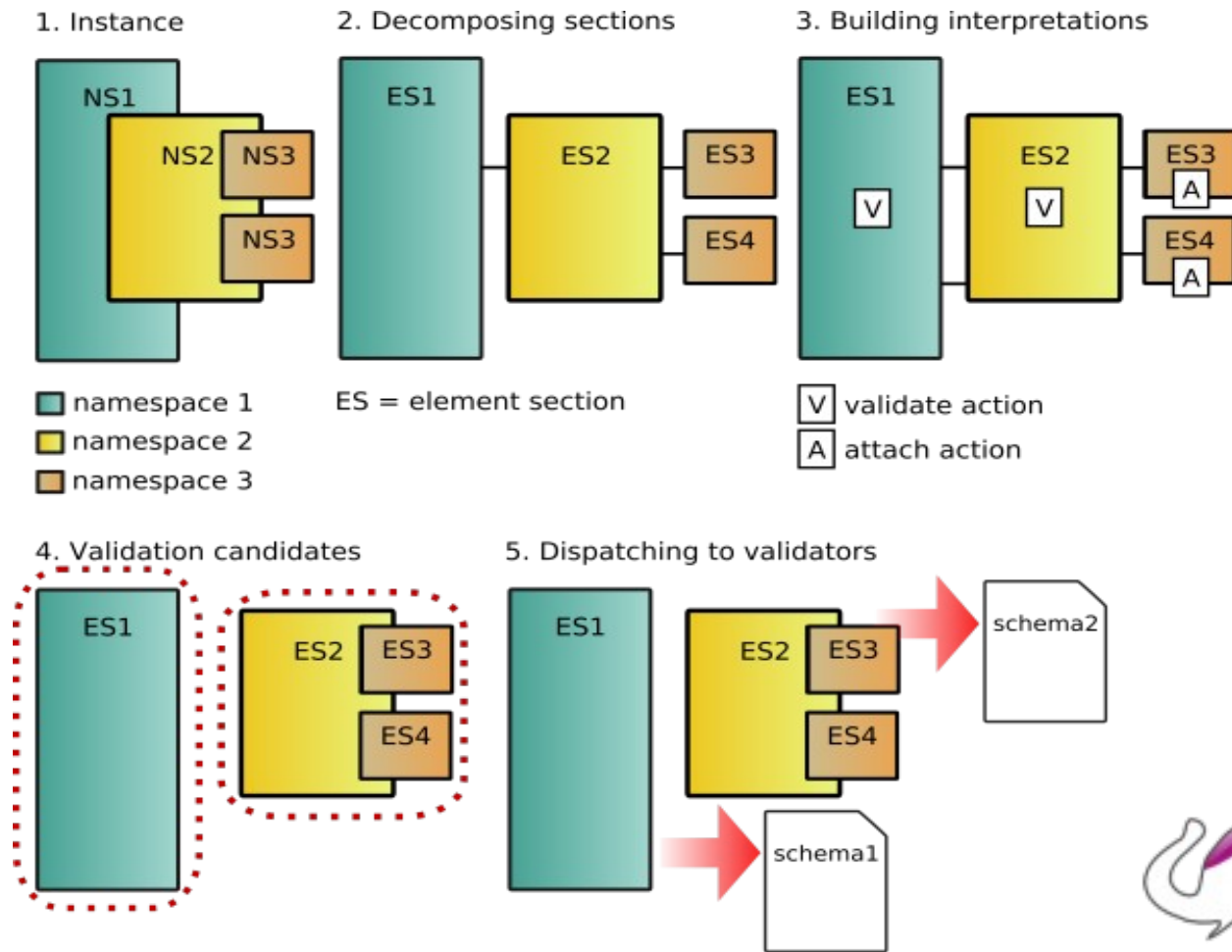


The validation dispatching process

- Principle
 - divide compound documents into single namespace fragments and send them individually for validation
- NVDL process input
 - Compound document instance, NVDL schema
- NVDL process phases
 - 1 Section decomposition
 - 2 Constructing interpretations
 - 3 Validation candidates
 - 4 Dispatching to validators



Example validation dispatching process

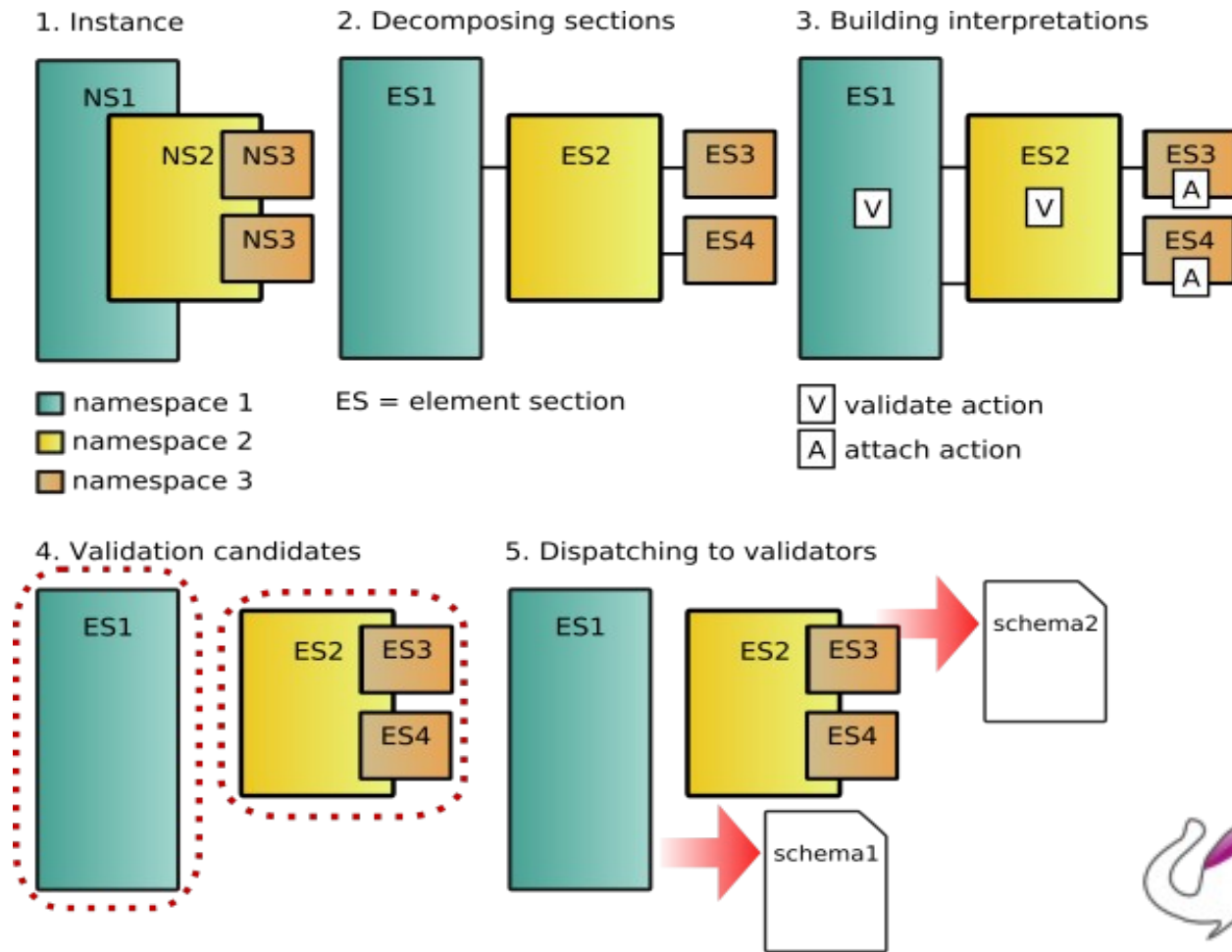


Adjusting the validation dispatching process

- NVDL schema
 - Rules
 - namespace, anyNamespace
 - Actions
 - validate, allow, reject
 - attach, unwrap, ...
 - Modes (different rules in different context)
 - Context handling
 - Parent namespace
 - Context path – fine grained



Example validation dispatching process



Instance example

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>1984</title>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description dc:title="1984 Nineteen Eighty-Four">
    <dc:creator>George Orwell</dc:creator>
    <dc:identifier>ISBN 014118776X</dc:identifier>
  </rdf:Description>
</rdf:RDF>
</head>

<body><p>It was a bright cold day in April,
and the clocks were striking thirteen...</p>
</body>

</html>
```



Section decomposition

Element section 1

```
<html xmlns="...">
<head>
<title>1984</title>
  ○
</head>
<body><p>...</p>
</body>
</html>
```

Element section 2

```
<rdf:RDF ...>
  <rdf:Description>
    ○
  </rdf:Description>
</rdf:RDF>
```

Element section 3

```
<dc:creator>George
Orwell</dc:creator>
```

Element section 4

```
<dc:identifier>ISBN
014118776X</dc:identifier>
```



NVDL schema example

```
<rules xmlns="http://purl.oclc.org/dsdl/nvd1/ns/structure/1.0">
<namespace ns="http://www.w3.org/1999/xhtml">
<validate schema="xhtml.xsd">
<context path="/html/head">
  <mode>
    <namespace ns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <validate schema="rdf.rng">
        <mode>
          <anyNamespace>
            <attach/>
          </anyNamespace>
        </mode>
      </validate>
    </namespace>
  </mode>
</context>
</validate>
</namespace>
</rules>
```



Executing actions

validate

Element section 1

```
<html xmlns="...">
<head>
<title>1984</title>
  ○
</head>
<body><p>...</p>
</body>
</html>
```

validate

Element section 2

```
<rdf:RDF ...>
  <rdf:Description>
    ↙
  </rdf:Description>
</rdf:RDF>
```

attach

Element section 3

```
<dc:creator>George
Orwell</dc:creator>
```

attach

Element section 4

```
<dc:identifier>ISBN
014118776X</dc:identifier>
```



Validation dispatching

Validation fragment 1

```
<html xmlns="...">
<head>
<title>1984</title>
</head>
<body><p>...</p>
</body>
</html>
```

xhtml.xsd

Validation fragment 2

```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description dc:title="1984 Nineteen Eighty-Four">
<dc:creator>George Orwell</dc:creator>
<dc:identifier>ISBN 014118776X</dc:identifier>
</rdf:Description>
</rdf:RDF>
```

rdf.rng



NVDL advantages

- Schema language neutral
 - Different schema languages within single validation process
- No adjustment to schemas required
 - Single namespace schemas can be fully reused
- Simple schema language to define CD (readability)
 - Compound language definition is separate from vocabulary schemas



The power of NVDL

- Example: schema for XHTML + RDF + SVG + MathML
 - Created in a matter of minutes
 - Existing schemas can be fully reused at their original locations
 - No knowledge about different vocabulary schema implementation required
 - Adjusting the NVDL schema for a different compound language in minutes



The power of NVDL

```
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0" startMode="root">
<mode name="root">
<namespace ns="http://www.w3.org/1999/xhtml">
  <validate schema="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <context path="head" useMode="head"/>
    <context path="div|li|p...all block level elements" useMode="block_inline"/>
    <context path="a|em|span|...all inline elements" useMode="block_inline"/>
  </validate>
</namespace>
</mode>
<mode name="block_inline">
  <namespace ns="http://www.w3.org/2000/svg">
    <validate schema="http://www.w3.org/TR/2002/SVG.xsd"/>
  </namespace>
  <namespace ns="http://www.w3.org/1998/Math/MathML">
    <validate schema="http://www.w3.org/Math/XMLSchema/mathml2/mathml2.xsd"/>
  </namespace>
</mode>
<mode name="head">
  <namespace ns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <validate schema="http://www.w3.org/2000/07/rdf.xsd"/>
  </namespace>
</mode>
</rules>
```



NVDL implementations

- JNVDL (<http://jnvdل.sf.net>)
 - open-source Java implementation written from scratch
 - supports many schemas through Java 1.5 validation API
- oNVDL (<http://www.oxygenxml.com/onvdل.html>)
 - open-source Java implementation based on Jing
 - also bundled with oXygen XML editor
- enovdl
 - open-source Mono/.NET implementation



JNVDL pros

- Out-of-the-box support
 - DTD, Relax NG, Relax Core, Relax Namespace, XML Schema, Trex, Schematron 1.5, ISO Schematron and Schematron embedded in Relax NG
- Extensions
 - Round tripping
 - Enhanced rule conditions, versioning
 - Namespace prefix filtering



JNVDL cons

- No streaming mode supported
 - Resource inefficiency in case of huge documents



NVDL extensions in JNVDL

- Round tripping
 - Validator report line numbers
 - Problems
 - Line numbering changes in validation fragments
 - Insignificant whitespaces in the instance are lost when parsed



NVDL extensions in JNVDL

- Namespace may not be sufficient to map instance fragments to schema
- Different versions of a language have same namespace
 - Additional conditions at rules
 - `jnvd1:useWhen="@version = '1.0' "`
 - `jnvd1:useWhenPublicId="-//W3C//DTD XHTML 1.0 Strict//EN"`
 - `jnvd1:useWhenSystemId="..."`
- Example:
 - HTML Strict/Transitional/Frameset
 - XSLT 1.0 / XSLT 2.0



NVDL extensions in JNVDL

- Namespace prefix filtering
 - DTD (no support for namespaces)
 - Example: `<h:html>` → `<html>`



Standards using NVDL

- W3C SVG Tiny 1.2
(<http://www.w3.org/TR/SVGMobile12/>)
- Ecma-376 Office Open XML
(<http://www.ecma-international.org/publications/standards/Ecma-376.htm>)
- W3C Internationalization Tag Set
(<http://www.w3.org/TR/its/>)
- DocBook V5.0 (<http://docbook.org>)



Thank you for your attention



Try <http://jnvdl.sourceforge.net> yourself!