

# XML-based NaturalLanguageGeneration

Graham Wilcock  
DocentofLanguageTechnology  
UniversityofHelsinki  
[graham.wilcock@helsinki.fi](mailto:graham.wilcock@helsinki.fi)

## Acknowledgements

- Part1(WhatisNLG?)  
islargelybasedonStephan Busemann's  
*NaturalLanguageGeneration:AnOverview*  
(<http://www.dfki.de/~busemann/VL-SS99/>)
- Parts2(XML -basedNLG)and3(demos)  
arebasedonworkintheUSIX -Interactproject.  
(<http://www.mlab.uiah.fi/interact/>)



## Requirements

---

- For this tutorial, you need:
  - a general knowledge of XML
  - a basic idea of XSLT transformations
- You don't need:
  - any previous knowledge of natural language generation



## Tutorial Outline

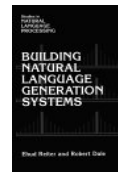
---

- Part 1: What is Natural Language Generation?
  - “Shallow” and “deep” NLG
- Part 2: XML -based NLG
  - Templates + trees + pipelines
  - A step -by- step example
- Part 3: Demos
  - Voice NLG with XML + Java speech
  - Web NLG with Cocoon servlets

## Part 1: What is NLG?

Natural language generation is the process of deliberately constructing a natural language text in order to meet specified communicative goals. [McDonald 1992]

- Recommended textbook :  
**Ehud Reiter & Robert Dale** ,  
*Building Natural Language Generation Systems* ,  
 Cambridge University Press, 2000.



## NLG inputs and outputs

- NLG input
  - a communicative goal, including
  - non-linguistic representation of information
- NLG output
  - a text, plain or formatted (HTML, JSML)
  - may be combined with graphics, tables etc.
- Knowledge sources required
  - domain-specific knowledge
  - language-specific knowledge
  - knowledge about human communication

## Template- v. plan -basedNLG

- “Template-based”NLG
  - Cannedtextswithvariableslots
  - Stringmanipulation(oftenin Perl)
  - Single-shotprocessing
- “Plan-based”NLG
  - Textplanningandsentenceplanning
  - Treestructuretransforms(ofteninJava)
  - Multi-stage,multi -levelprocessing

## Tasksofplan -basedNLG

- Contentdetermination
- Discourseplanning
- Lexicalization
- Referringexpressiongeneration
- Sentenceaggregation
- Surfaceralization

## NLG: Content Determination

- Deciding what to say
  - Construct a set of MESSAGES from the data source
  - A message may become a word, phrase or sentence
  - Messages are based on domain entities (concepts)

IDENTITY(NEXTSHIP,MS -LILLY) → *The next ship is the MS -LILLY.*

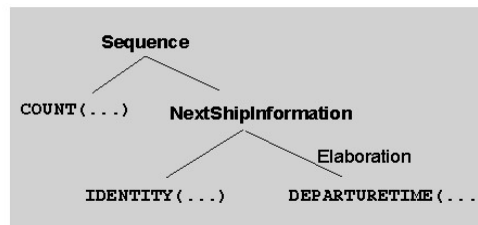
DEPARTURETIME(MS-LILLY,1000) → *The MS -LILLY departs at 10am .*

COUNT(SHIP,SOURCE(HAMBURG),DESTINATION(COPENHAGEN),5,PERDAY)  
→ *There are 5 ships daily from Hamburg to Copenhagen .*

## NLG: Discourse Planning

- Organize messages in a coherent text plan
  - A text is structured, not random sentences
  - Conceptual grouping, rhetorical relations

*There are five ships daily from Hamburg to Copenhagen. The next ship is the MS -LILLY. It departs at 10am.*



## NLG: Lexicalization

- Mapping from domain concepts to lexemes
- Determines the content words to be used
  - discourse focus - *buy vs sell*
  - collocations - *exert influence, administer punishment*
  - lexical semantics - *male unmarried adult vs bachelor*
  - basic level categories - *dog vs poodle*
  - attitude - *house vs home, father vs dad*
  - idioms - *kick the bucket*
- Partly determines the syntactic structure

## NLG: Referring Expressions

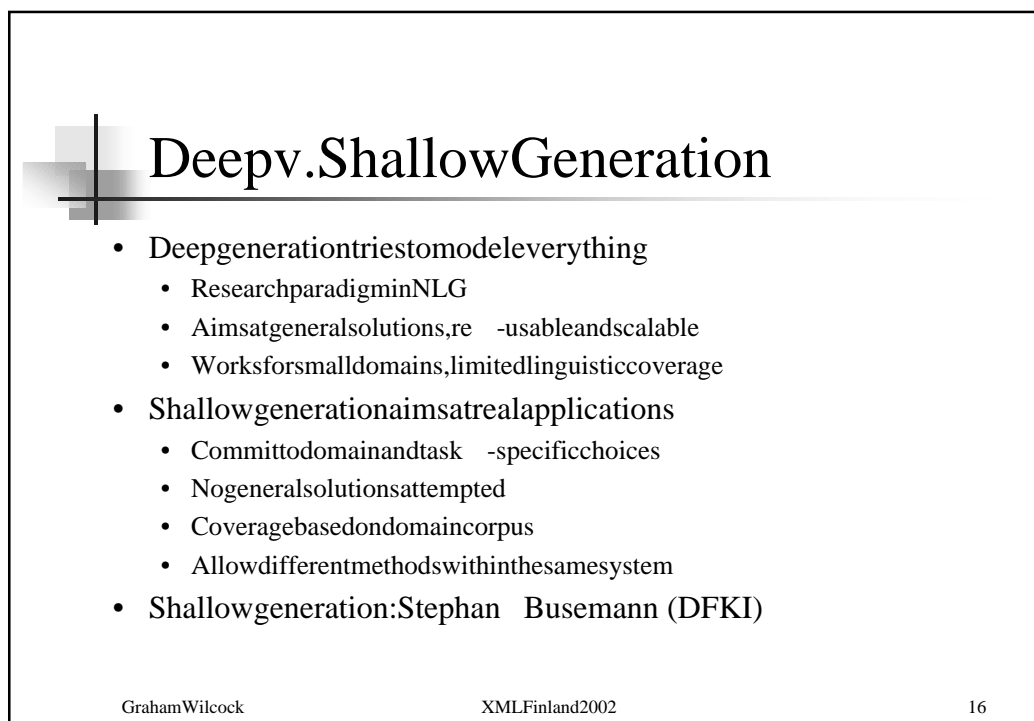
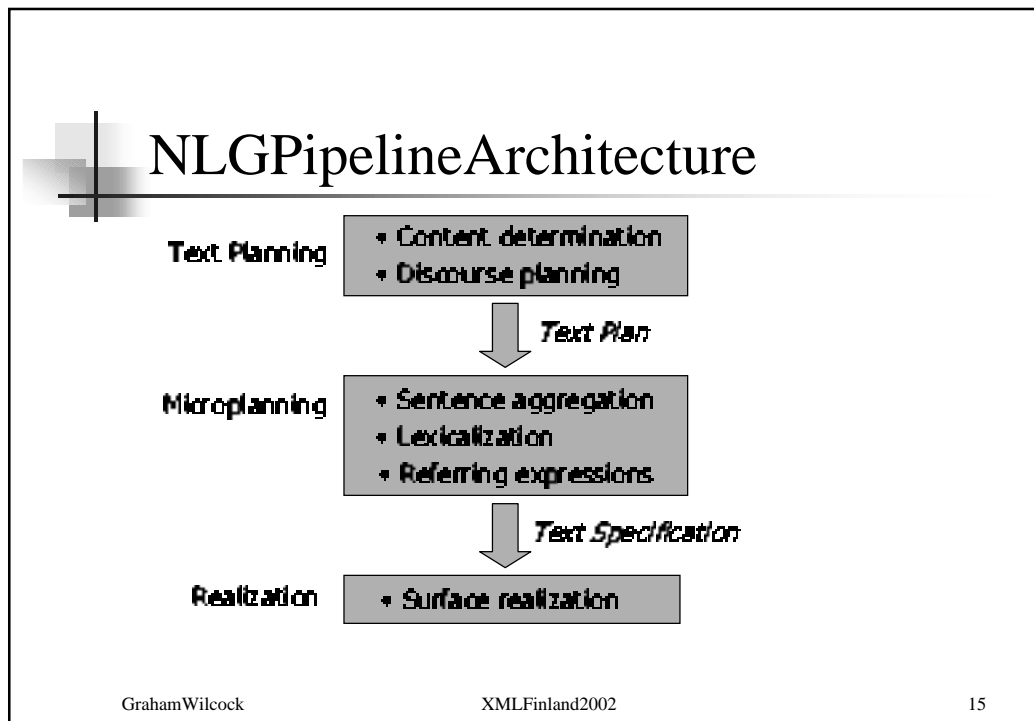
- Allow the hearer to identify discourse objects
- Kinds of referring expressions
  - Proper names - *Hamburg, The United States*
  - Definite descriptions - *the ship that leaves at 10*
  - Proforms - *it, later, there*
- Initial reference
  - use a full name - *the MS - LILLY*
  - relate to a salient object - *the ship's snack bar*
  - specify physical location - *the ship at pier 12*

## NLG: Sentence Aggregation

- Mapping from messages to sentences
  - One-to-one mapping results in poor text
  - Need to combine messages in complex sentences
- Without aggregation
  - *The next ship is the MS - LILLY. It leaves Hamburg at 10am. It has a restaurant. It has a snack bar .*
- With aggregation
  - *The next ship, which leaves Hamburg at 10am, is the MS - LILLY. It has a snack bar and a restaurant.*

## NLG: Surface Realization

- Convert text specifications into output text
- Generates grammatically correct text
  - insert function words - *he wants to book a ticket*
  - word inflection - *like + ed → liked*
  - grammatical word order
- Techniques for grammatical knowledge
  - declarative bidirectional grammars
  - grammars designed for generation
  - templates, easy and fast to implement

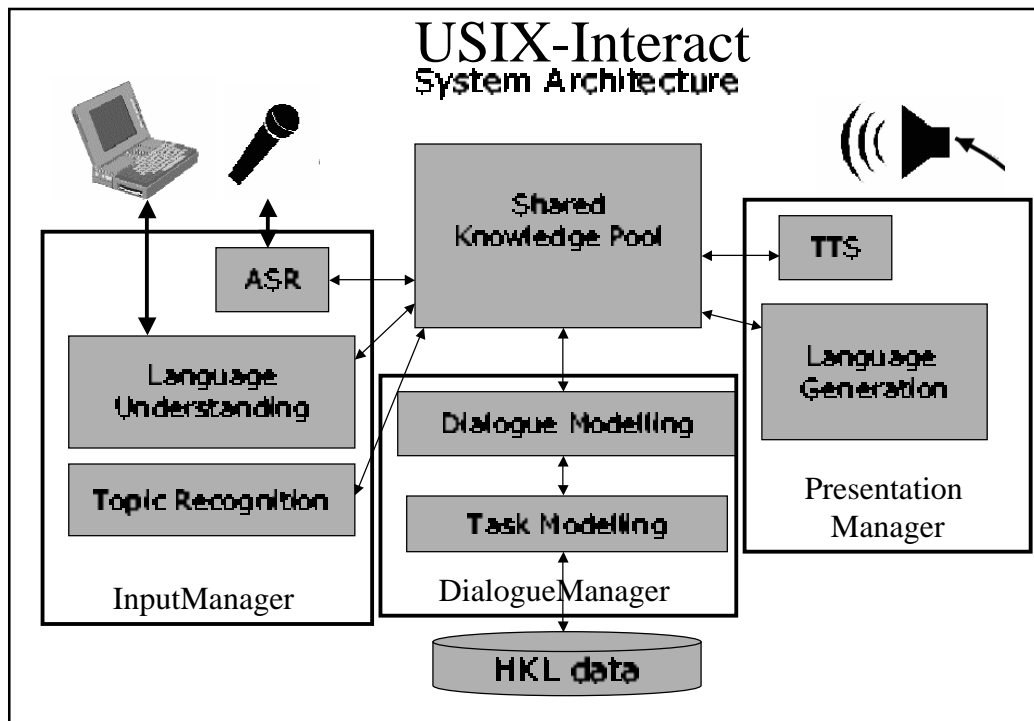


## Part2: XML-basedNLG

- Templates
  - ImplementNLGtemplate -basedgenerationusing predefinedXSLTtemplates
- Trees
  - ImplementNLGtreestructures(textplans,text specifications)usingXMLtreestructures
- Pipelines
  - ImplementNLGpipelinearchitectureusingpipeds equencesofXML -to-XMLtransformations

## A Step -by-Step Example

- Simplified(rapidprototyping)
  - Quickly-developed“firstprototype”generatorfor USIX-Interactspokendialogueproject
  - Dialogueresponseplanningandgeneration
- XML+XSLTimplementation
  - Agendaofdomainconcepts
  - Template-basedtextplans(“aggregations”)
  - PipelineofXSLTtransformations



## Responseplanning

- NewInformation( NewInfo)
  - Responseplanningstartsfrom NewInfo
  - Responsealwaysincludes NewInfo
- Topic
  - NewInfo maybe *wrapped* byTopiclink
  - Politeresponse:Topicand NewInfo
  - Ellipticalresponse: NewInfo only

## Generation from NewInfo (Ex.1)

- *Which bus goes to Malmi?*
  - Topic:transportation:bus
  - Topic:destination: Malmi
  - NewInfo:busnumber:74
- *Number 74.*

## Input: AgendainXML

- Unordered set of domain concepts
  - Marked as "Topic" or "NewInfo"
- Specified by Dialogue Manager
- Starting point for Generator
  - Shared XML representation

## Input:Agenda(Ex.1)

```

<agenda id="1">
  <concept info="Topic">
    <type>transportation</type>
    <value>bus</value>
  </concept>
  <concept info="Topic">
    <type>destination</type>
    <value>malmi</value>
  </concept>
  <concept info="Topic">
    <type>bus</type>
    <value>exists</value>
  </concept>
  <concept info="NewInfo">
    <type>busnumber</type>
    <value>74</value>
  </concept>
</agenda>

```

• *Number 74.*

## Template-basedNLGinXSLT

- CreatepredefinedXSLTnamedtemplates
  - DecideasetofpredefinedTextPlantreestructures(here,the structuresarecalled“aggregations”)
  - DefineonenamedtemplateperTextPlanaggregation
- XSLTprocessing
  - TesttheconceptvaluesgiveninAgenda
  - SelectaTextPlanbasedonconceptvalues
  - ThenamedtemplatefortheTextPlancopiesAgendaconcepts intoitsslotsusing< xsl:copy-of>

## TextPlanning:SelectthePlan

```

<xsl:template match="agenda">
  <xsl:choose>
    <xsl:when test="concept[@info='NewInfo']/type='transportation'">
      <xsl:call-template name="BY-TRANSPORT"/>
    </xsl:when>
    <xsl:when test="concept[@info='NewInfo']/type='bus'">
      <xsl:choose>
        <xsl:when test="concept[@info='NewInfo']/type='busnumber'">
          <xsl:call-template name="NUM-DEST-TIME"/>
        </xsl:when>
        ...
      </xsl:choose>
    </xsl:when>
    <xsl:when test="concept[@info='NewInfo']/type='busnumber'">
      <xsl:call-template name="NUMBER-ONLY"/>
    </xsl:when>
    ...
  </xsl:choose>
</xsl:template>

```

Graham Wilcock

XMLFinland2002

25

## TextPlanning:InsertConcepts

```

<xsl:template name="NUM-DEST-TIME">
  <aggregation>
    <subject cat="NP">
      <xsl:copy-of select="./concept[type='busnumber']"/>
    </subject>
    <predicate cat="VP">
      <xsl:copy-of select="./concept[type='bus']"/>
    </predicate>
    <complement cat="PP">
      <xsl:copy-of select="./concept[type='destination']"/>
    </complement>
    <adjunct cat="PP">
      <xsl:copy-of select="./concept[type='bustime']"/>
    </adjunct>
  </aggregation>
</xsl:template>

```

Graham Wilcock

XMLFinland2002

26

## TextPlanning:TextPlan(Ex.1)

```

<agenda id="1">
  <concept info="Topic">
    <type>transportation</type>
    <value>bus</value>
  </concept>
  <concept info="Topic">
    <type>destination</type>
    <value>malmi</value>
  </concept>
  <concept info="Topic">
    <type>bus</type>
    <value>exists</value>
  </concept>
  <concept info="NewInfo">
    <type>busnumber</type>
    <value>74</value>
  </concept>
</agenda>

```

- TextPlan

```

<aggregation id="1">
  <subject cat="NP">
    <concept info="NewInfo">
      <type>busnumber</type>
      <value>74</value>
    </concept>
  </subject>
</aggregation>

```

## NLGPipelineinXML

- Pipelinearchitecture
  - XMLinput(agenda),XMLoutput(JSML)
  - SequenceofXML -to-XMLtransformations
  - agenda → aggregation → speech markup
- TreetransformationsinDOMorXSLT
  - Modifytreestructuresandnodesinthem
  - Forexample(inReferringExpressionsstage):replace domainconceptsinaggregationtreebylinguisticreferring expressions

## Microplanning: NewInfo

```

<!-- REFERRING EXPRESSIONS: DESCRIPTIONS -->
<xsl:template match="concept[@info='NewInfo']">
  <xsl:choose>
    <xsl:when test="type='busnumber'">
      <word>number</word>
      <word><xsl:value-of select="value/text()"/></word>
    </xsl:when>
    <xsl:when test="type='destination'">
      <word>to</word>
      <word><xsl:value-of select="value/text()"/></word>
    </xsl:when>
    <xsl:when test="type='bustime'">
      <word>at</word>
      <word><xsl:value-of select="value/text()"/></word>
    </xsl:when>
  </xsl:choose>
</xsl:template>

```

## Microplanning: Pronouns

```

<!-- REFERRING EXPRESSIONS: PRONOUNS -->
<xsl:template match="concept[@info='Topic']">
  <xsl:choose>
    <xsl:when test="type='busnumber'">
      <word>it</word>
    </xsl:when>
    <xsl:when test="type='destination'">
      <word>there</word>
    </xsl:when>
    <xsl:when test="type='bustime'">
      <word>then</word>
    </xsl:when>
  </xsl:choose>
</xsl:template>

```

## Microplanning:TextSpec(Ex.1)

- TextPlan

```
<aggregation id="1">
  <subject cat="NP">
    <concept info="NewInfo">
      <type>busnumber</type>
      <value>74</value>
    </concept>
  </subject>
</aggregation>
```

- TextSpecification

```
<aggregation id="1">
  <subject cat="NP">
    <word>number</word>
    <word>74</word>
  </subject>
</aggregation>
```

## Realization:Output(Ex.1)

- TextSpecification

```
<aggregation id="1">
  <subject cat="NP">
    <word>number</word>
    <word>74</word>
  </subject>
</aggregation>
```

- Output

```
<jssml lang="en">
  <div type="sent">
    number 74
  </div>
</jssml>
```

(JavaSpeech Markup Language)

## AnotherExample(Ex.2)

- *HowdoIgetto Malmi?*
  - Topic:destination: Malmi
  - NewInfo:transportation:bus
  - NewInfo:busnumber:74
- *Bybus – number74goesthere.*



## Input:Agenda(Ex.2)

```

<agenda id="2">
  <concept info="NewInfo">
    <type>transportation</type>
    <value>bus</value>
  </concept>
  <concept info="Topic">
    <type>destination</type>
    <value>malmi</value>
  </concept>
  <concept info="NewInfo">
    <type>bus</type>
    <value>exists</value>
  </concept>
  <concept info="NewInfo">
    <type>busnumber</type>
    <value>74</value>
  </concept>
</agenda>

```

## TextPlanning:TextPlan(Ex.2)

```

<agenda id="2">
  <concept info="NewInfo">
    <type>transportation</type>
    <value>bus</value>
  </concept>
  <concept info="Topic">
    <type>destination</type>
    <value>malmi</value>
  </concept>
  <concept info="NewInfo">
    <type>bus</type>
    <value>exists</value>
  </concept>
  <concept info="NewInfo">
    <type>busnumber</type>
    <value>74</value>
  </concept>
</agenda>

```

```

<aggregation id="2">
  <adjunct cat="PP">
    <concept info="NewInfo">
      <type>transportation</type>
      <value>bus</value>
    </concept>
  </adjunct>
  <prosody cat="pause"/>
  <subject cat="NP">
    <concept info="NewInfo">
      <type>busnumber</type>
      <value>74</value>
    </concept>
  </subject>
  <predicate cat="VP">
    <concept info="NewInfo">
      <type>bus</type>
      <value>exists</value>
    </concept>
  </predicate>
  <complement cat="PP">
    <concept info="Topic">
      <type>destination</type>
      <value>Malmi</value>
    </concept>
  </complement>
</aggregation>

```

Graham Wilcock

XMLFinland2002

35

## Microplanning:TextSpec(Ex.2)

```

<aggregation id="2">
  <adjunct cat="PP">
    <concept info="NewInfo">
      <type>transportation</type>
      <value>bus</value>
    </concept>
  </adjunct>
  <prosody cat="pause"/>
  <subject cat="NP">
    <concept info="NewInfo">
      <type>busnumber</type>
      <value>74</value>
    </concept>
  </subject>
  <predicate cat="VP">
    <concept info="NewInfo">
      <type>bus</type>
      <value>exists</value>
    </concept>
  </predicate>
  <complement cat="PP">
    <concept info="Topic">
      <type>destination</type>
      <value>Malmi</value>
    </concept>
  </complement>
</aggregation>

```

```

<aggregation id="2">
  <adjunct cat="PP">
    <word>by</word>
    <word>bus</word>
  </adjunct>
  <prosody cat="pause"/>
  <subject cat="NP">
    <word>number</word>
    <word>74</word>
  </subject>
  <predicate cat="VP">
    <word cat="V">
      <lexeme>go</lexeme>
      <features>3sg</features>
    </word>
  </predicate>
  <complement cat="PP">
    <word>there</word>
  </complement>
</aggregation>

```

Graham Wilcock

XMLFinland2002

36

## Realization:Output(Ex.2)

```
<aggregation id="2">
  <adjunct cat="PP">
    <word>by</word>
    <word>bus</word>
  </adjunct>
  <prosody cat="pause"/>
  <subject cat="NP">
    <word>number</word>
    <word>74</word>
  </subject>
  <predicate cat="VP">
    <word cat="V">
      <lexeme>go</lexeme>
      <features>3sg</features>
    </word>
  </predicate>
  <complement cat="PP">
    <word>there</word>
  </complement>
</aggregation>
```

- Output

```
<jsml lang="en">
  <div type="sent">
    by bus
    <break size="large"/>
    number 74 goes there
  </div>
</jsml>
```

## Part3:Demos

- VoiceNLG
  - BilingualFinnish/Englishgenerator
  - FreeTTS Javaspcechsynthesizer
  - CurrentlyhasonlyEnglishvoice
- WebNLG
  - BilingualFinnish/Englishgenerator
  - CocoonXMLserverimplementation
  - TowardsstheSemanticWeb

## Demo: VoiceNLG

- Implemented in Java
  - Java API for XML (JAXP) to execute XSLT
  - Java Speech API to execute speech synthesis
- XML-based NLG
  - Input: Annotation Graph in XML
  - Output: bilingual Finnish and English in JSML
- Java speech synthesis
  - Input: (currently) English in JSML
  - Output: speech

## Annotation Graphs

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AGSet id="AGSet" version="1.0"
  xmlns="http://www ldc.upenn.edu/atlas/ag/"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:dc="http://purl.org/DC/documents/rec-dces-19990702.htm">

  <AG id="AGSet_AG1" type="transcription" timeline="AGSet_Timeline1">
  <Anchor id="AGSet_AG1_anchor1" offset="1" unit="index"/>
  <Anchor id="AGSet_AG1_anchor2" offset="2" unit="index"/>

  <Annotation type="DAct" id="AGSet1_AG1_annotation1"
    start="AGSet1_AG1_anchor2" end="AGSet1_AG1_anchor2">
  <Feature name="score">1.0</Feature>
  <Feature name="name">ALOITUS</Feature>
  </Annotation>
```

## FreeTTS

- AspeechsynthesizerinJava
  - FreeopensourcefromSun
  - Basedon Flite andFestival(Edinburgh)
  - Portable(needsJava1.4)
  - JSAPIinterface
- Currentlimitations( FreeTTS 1.1)
  - Suppliedvoices:English
  - JSMLacceptedbutnotapplied

## Demo:WebNLG

- NLGwebserver
  - XML-basedNLG
- Cocoonimplementation
  - Cocoonwebpublishingframework(Java)
  - Runsina servlet engine(Tomcat:Java)
  - UsesXSLTtransformers( Xalan:Java)
  - ConfigurableXSLTpipelines

## What is COCOON

“Apache Cocoon is an XML publishing framework that raises the usage of XML and XSLT technologies for server applications to a new level. Designed for performance and scalability around pipelined SAX processing, Cocoon offers a flexible environment based on a separation of concerns between content, logic, and style. To top this all off, Cocoon's centralized configuration system and sophisticated caching help you to create, deploy, and maintain rock-solid XML server applications.”

- Apache Cocoon website

## Configurable pipelines

- Pipelines specified in Cocoon configuration file

```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
  <!-- English examples: Jokinen & Wilcock (SIGDIAL-2001) -->
  <!-- Get an agenda, apply XSLT transforms, serialize as HTML -->
  <map:pipeline>
    <map:match pattern="generate(agendas/*.xml)">
      <map:generate src="agendas/{1}.xml"/>
      <map:transform src="transforms/SIGDIAL-2001/aggregation.xsl"/>
      <map:transform src="transforms/SIGDIAL-2001/lexicalization.xsl"/>
      <map:transform src="transforms/SIGDIAL-2001/realization.xsl"/>
      <map:serialize type="html"/>
    </map:match>
  </map:pipeline>
```

## Adding a Finnish pipeline

```

<!-- Finnish pipeline: Interact Demo 2002 -->
<!-- Get an A-Graph, apply XSLT transforms, serialize as HTML -->
<map:pipeline>
  <map:match pattern="suomeksi(agraphs/*.xml)">
    <map:generate src="agraphs/{1}.xml"/>
    <map:transform src="transforms/InteractDemo/AG-GetContent.xsl"/>
    <map:transform src="transforms/InteractDemo/AG-ResponsePlanner.xsl"/>
    <map:transform src="transforms/InteractDemo/FI-Lexicalization.xsl"/>
    <map:transform src="transforms/InteractDemo/FI-ReferringExps.xsl"/>
    <map:transform src="transforms/InteractDemo/FI-SurfaceRealizer.xsl"/>
    <map:transform src="transforms/displayverbatim.xsl"/>
    <map:serialize type="html"/>
  </map:match>
</map:pipeline>

```

## Adding an English pipeline

```

<!-- English pipeline: Interact Demo 2002 -->
<!-- Get an A-Graph, apply XSLT transforms, serialize as HTML -->
<map:pipeline>
  <map:match pattern="english(agraphs/*.xml)">
    <map:generate src="agraphs/{1}.xml"/>
    <map:transform src="transforms/InteractDemo/AG-GetContent.xsl"/>
    <map:transform src="transforms/InteractDemo/AG-ResponsePlanner.xsl"/>
    <map:transform src="transforms/InteractDemo/EN-Lexicalization.xsl"/>
    <map:transform src="transforms/InteractDemo/EN-ReferringExps.xsl"/>
    <map:transform src="transforms/InteractDemo/EN-SurfaceRealizer.xsl"/>
    <map:transform src="transforms/displayverbatim.xsl"/>
    <map:serialize type="html"/>
  </map:match>
</map:pipeline>

```

## FromDomainstoLanguages

- 1 Domain, 2 Languages

```
graph LR; DB[(Database)] --> TP[Text Planner]; TP --> MP1[Micro-planner]; TP --> MP2[Micro-planner]; MP1 --> FR[Finnish Realizer]; MP2 --> ER[English Realizer]; FR --> S1[😊]; ER --> S2[😊];
```

- 2 Domains, 1 Language

```
graph LR; DB1[(Database)] --> TP1[Text Planner]; DB2[(Database)] --> TP2[Text Planner]; TP1 --> MP1[Micro-planner]; TP2 --> MP2[Micro-planner]; MP1 --> FR[Finnish Realizer]; MP2 --> FR; FR --> S[😊];
```

Graham Wilcock XMLFinland2002 47

## TowardstheSemanticWeb

- 2 Ontologies, 1 Language

```
graph LR; SW((Semantic Web)); SW --> TP1[Text Planner]; SW --> TP2[Text Planner]; TP1 --> MP1[Micro-planner]; TP2 --> MP2[Micro-planner]; MP1 --> FR[Finnish Realizer]; MP2 --> FR; FR --> S[😊];
```

Graham Wilcock XMLFinland2002 48