

Heterogeneous but Functional Content Engineering Framework

Kirill Lisovsky
Patrick Baker
Stilo Corporation

Content Engineering

- Building and processing content in order to deliver it to customer
- Content Flow and Content Engineering Tasks
 - Acquire: import, metadata extraction, agregation
 - Enrich: transformation, linking, filtering
 - Deliver: rendering, publishing
 - ... validation, querying, chunking

OmniMark language

- Domain-specific Content Engineering Language
- The Streaming Paradigm
- Rules-based Programming
- Hierarchical Markup Parsing Model
- Powerful Pattern Matching
- Referents

Content engineering and Scheme

- Data Model
 - SXML
- Parsing
 - SSAX, HTMLPrag
- Querying
 - XPath , Sedna
- Transformation
 - XSLT, Direct evaluation
- Linking
 - SXPath, XPathLink

An XML document and its SXML representation

```
<?xml version='1.0'>
<di contract="728g">
  <wt refnum="345">
    <delivery>
      <date month="6" day="01" year="2001"/>
      <weight>783</weight>
    </delivery>
    <vehicle type="truck" number="A567TP99"/>
  </wt>
  <wt refnum="459">
    <vehicle type="car" number="25676043"/>
  </wt>
</di>
```

```
(*TOP* (*PI* xml "version='1.0'")
 (di (@ (contract "728g"))
  (wt (@ (refnum "345"))
   (delivery (date (@ (month "6")
                     (day "1") (year "2001"))
                 (weight "783")))
   (vehicle (@ (type "truck") (number "A567TP99"))))
 (wt (@ (refnum "459"))
  (vehicle (@ (type "car") (number "25676043")))) ) )
```

SXPath

```
`(*TOP*  
  (element "text1"  
    (nested (@ (attr "value"))  
      (element "it is in nested"  
        "text2"  
        (deep "First")  
        (deep "Second")  
        "text5"))))
```

```
(sxpath '(element nested))  
(sxpath "element/nested")
```

```
(node-join (select-kids (ntype?? 'element))  
           (select-kids (ntype?? 'nested)))
```

```
(sxpath `(element "nested/deep" *text* ,(lambda (nodeset)  
  (map  
    (lambda(x) (if (> (string-length node) 5) "Long" Short"))  
    nodeset))))
```

SXLink enabled SXPath

Go to web site of SICP at MIT Press, find the link to the full text of book and get the HTML head of the book's page"

```
((txpath/c "//a[contains(self::* , 'Full text')]/traverse::html/head")  
(xlink:documents "http://mitpress.mit.edu/sicp/"))
```

Why heterogeneous?

- You can do everything with Scheme...
 - but not necessary with one implementation
- Utilizing existing applications, tools and skills
 - XSLT stylesheets, latex classes, etc.
- Specialized tools
 - SGML or RTF parsers
 - PDF to XML transformation
 - LaTeX publishing

Recursive Composition of Web Services

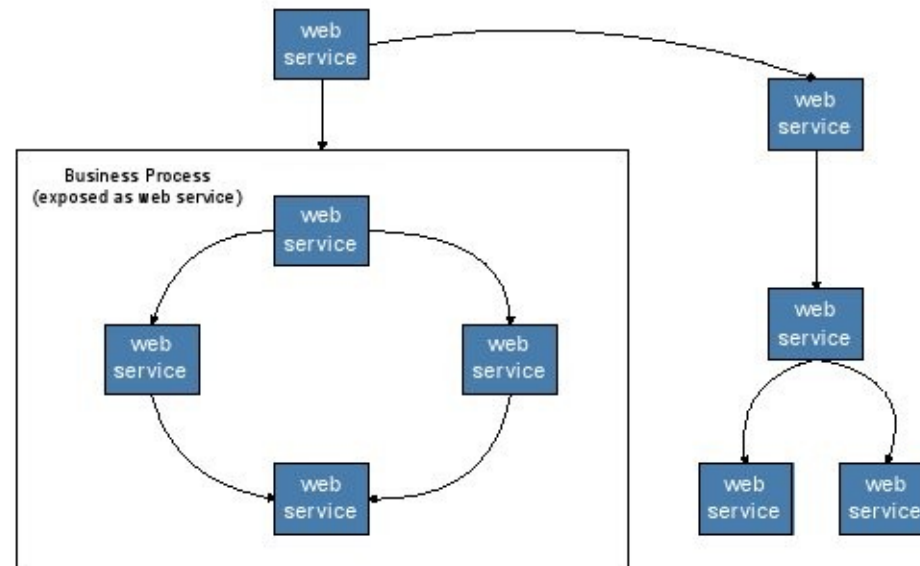


Figure 1. Recursive Composition of Web Services

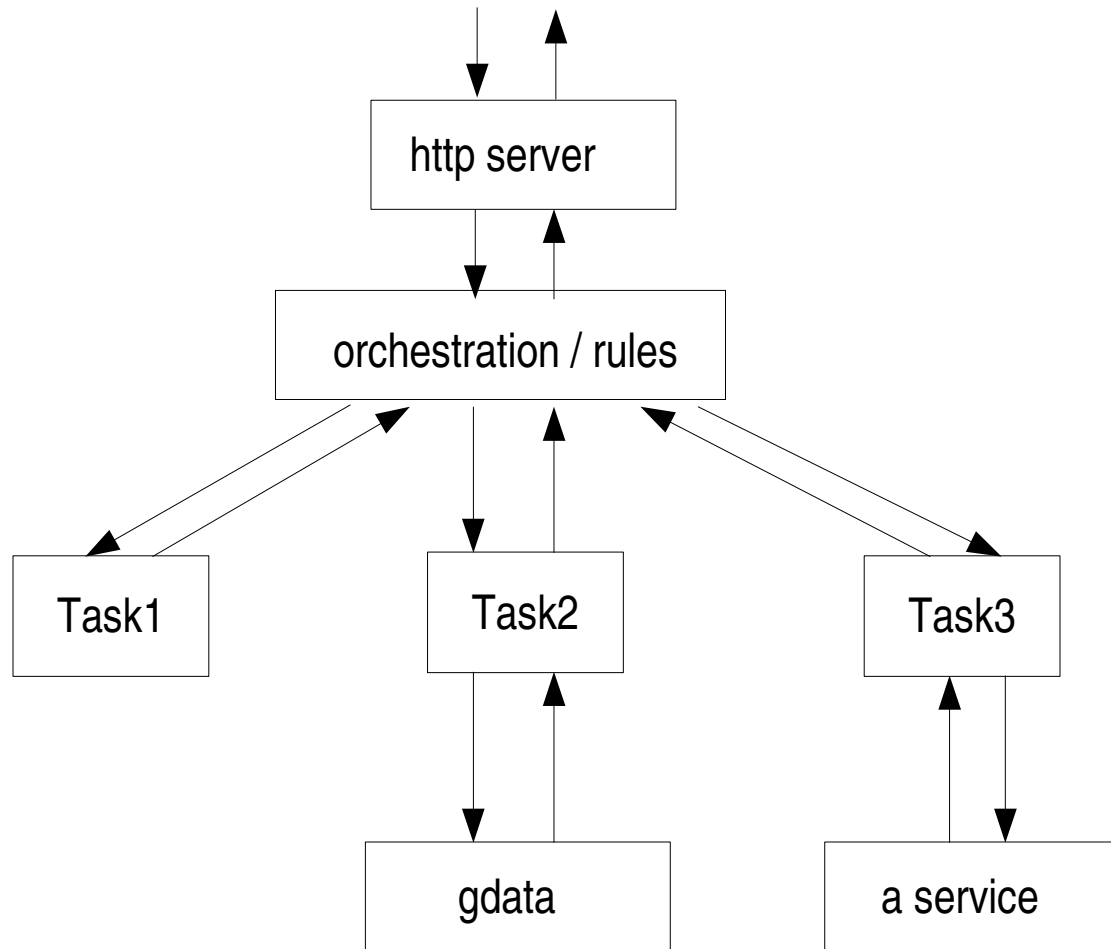
Framework Functionality

- Services Orchestration
- Content Processing (Business) Rules
 - Dependencies management
- Streaming Infrastructure

Orchestration

- Connecting services together to create higher-level business process
- Standards-based and open
 - Business Process Execution Language
 - BPEL4WS, BPML/BPMN
 - Web Service Choreography Interface
- Content engineering process – a special case of a business process

Orchestration and business rules

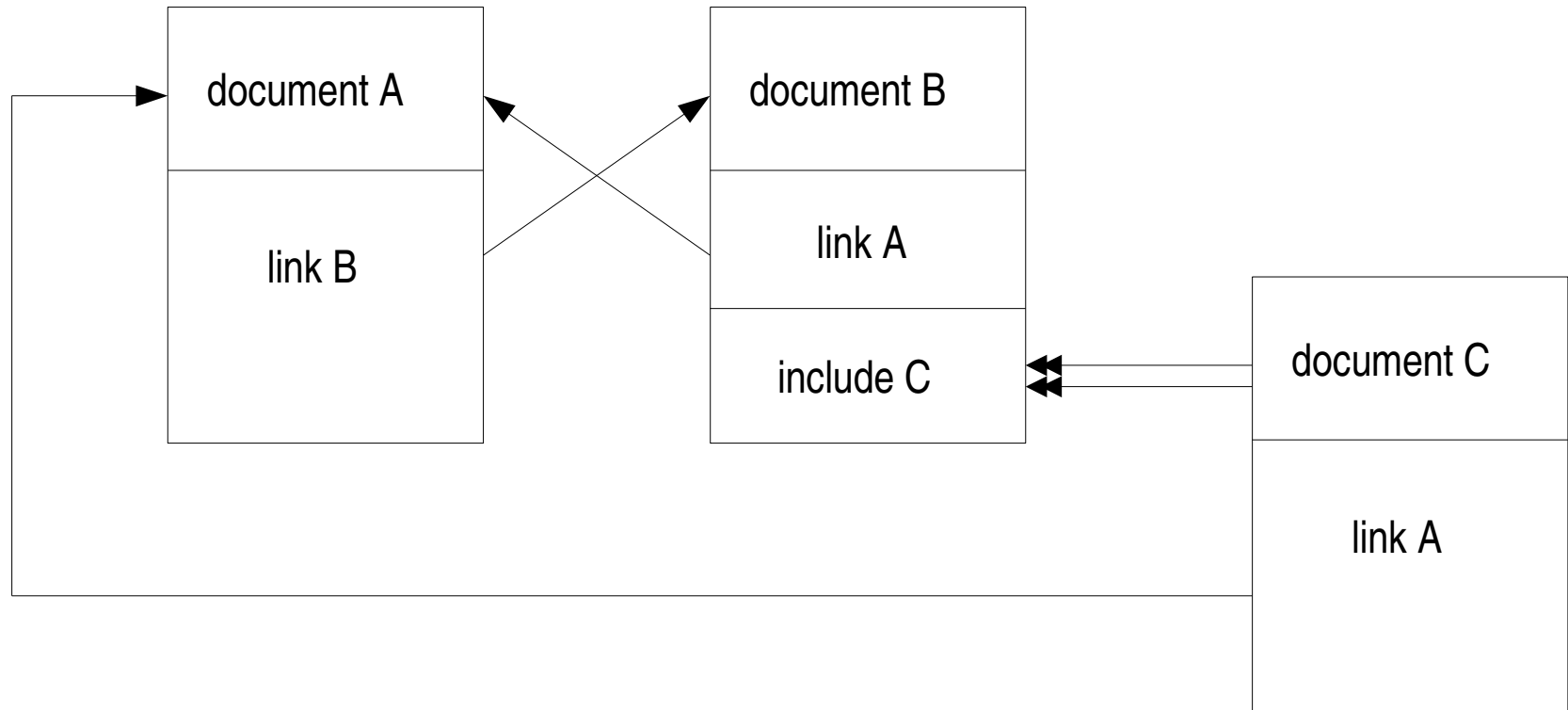


OmniMark Orchestration

```
find "convert" blank+ any-text++ => file-from blank+
    "to" blank + any-text++ => file-to
using output as (make-xml >> capitalise-names >>
    encode-gender >> file file-to)
output file file-from
```

- Non-linear topologies may be supported too
- Works fine, as long as you are OmniMark only

Document dependencies



Dependencies management

- Recursive, mutually recursive, and circular dependencies
- Impact of changes management
- Performance improvement
- *make* ?

What's wrong with *make*?

- Operates on files, not URIs
- Change detection: file modification time
- Temperamental syntax
- Ant, Maven, Cook, Scons ?
- Logical Rule Language

Logical *make*

```
%%%%%%%% rules %%%%%%%%%%  
depends_on(A,B) :- build_from(A,B).  
depends_on(A, B) :- depends_on(A,Z), build_from(Z,B).  
changed(A) :- rebuild(A).  
changed(A) :- modified(A).  
rebuild(A) :- depend_on(A,Z), changed(Z).
```

```
%%%%%%%% facts %%%%%%%%%%  
build_from('foo.o', 'foo.c').  
build_from('bar.o', 'bar.c').  
build_from('foo.exe', 'foo.o').  
build_from('foo.exe', 'bar.o').  
modified('foo.c').  
rebuild('foo.exe').
```

RuleML: the Rule Markup Language

- XML-based rule language
- RuleML backend – a deductive engine
 - Jess, jDREW
 - XSB, SWI
 - Kanren, Deviser

Power of RuleML

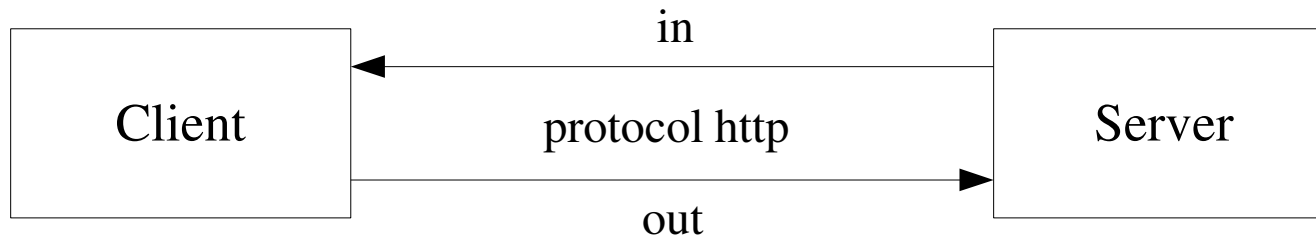
```
<?xml version="1.0" standalone="no"?>
<rulebase>
<imp>
  <_head>
    <atom>
      <_opr><rel>depends_on</rel></_opr>
      <var>A</var>
      <var>B</var>
    </atom>
  </_head>
  <_body>
    <and>
      <atom>
        <_opr><rel>build_from</rel></_opr>
        <var>A</var>
        <var>B</var>
      </atom>
    ...
  </_body>
</imp>
...
</rulebase>
```

```
depends_on(A,B):-build_from(A,B).
...
```

Streaming

- Delivering output data while consuming input data
 - Memory consumption does not depend on the data size
 - Natural paradigm for producer-filter-consumer applications
- Types of streaming:
 - Programming Language (OmniMark)
 - Operating System (Unix pipes)
 - Protocol (HTTP)

Client/server interaction



In \ Out	Small	Large
Small	login	document checkout
Large	document checkin	filter

Protocol Streaming

- HTTP Streaming
 - Chunking
 - Streaming response is typical
 - Streaming post is not
- FastCGI and load balancing
 - Apache – poor, but improving
 - Lighttpd – round robin
 - Zeus – session oriented

Functional RuleML

<Equal oriented="yes">

<Expr>

<Fun in="yes">home</Fun>

<Expr>

<Fun in="no">father-of</Fun>

<Ind>John</Ind>

</Expr>

</Expr>

<Ind>Mexico City</Ind>

</Equal>

home(father-of(John)) = Mexico City

<Expr>

<Fun in="yes" val='1">home</Fun>

<Ind>John</Ind>

<Ind>Mary</Ind>

</Expr>

Framework Functionality

- Services Orchestration
- Content Processing (Business) Rules
 - Dependencies management
- Streaming Infrastructure
- Framework: Rule Language + Streaming Protocol
 - RuleML / Kanren
 - HTTP
- Content Engineering: OmniMark, SXML, ...