Semantic Web

Semantic Web Architecture
### Where are we?

<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td><strong>Semantic Web Architecture</strong></td>
</tr>
<tr>
<td>3</td>
<td>Resource Description Framework (RDF)</td>
</tr>
<tr>
<td>4</td>
<td>Web of data</td>
</tr>
<tr>
<td>5</td>
<td>Generating Semantic Annotations</td>
</tr>
<tr>
<td>6</td>
<td>Storage and Querying</td>
</tr>
<tr>
<td>7</td>
<td>Web Ontology Language (OWL)</td>
</tr>
<tr>
<td>8</td>
<td>Rule Interchange Format (RIF)</td>
</tr>
<tr>
<td>9</td>
<td>Reasoning on the Web</td>
</tr>
<tr>
<td>10</td>
<td>Ontologies</td>
</tr>
<tr>
<td>11</td>
<td>Social Semantic Web</td>
</tr>
<tr>
<td>12</td>
<td>Semantic Web Services</td>
</tr>
<tr>
<td>13</td>
<td>Tools</td>
</tr>
<tr>
<td>14</td>
<td>Applications</td>
</tr>
</tbody>
</table>
Overview

• Introduction and motivation
• Technical solutions
  – Semantic Web architecture
  – Uniform Resource Identifier
  – Extensible Markup Language
  – XML Schema
  – Namespaces
• Extensions
• Illustration by a large example
• Summary
• References
INTRODUCTION AND MOTIVATION
Web Architecture

- The Semantic Web extends the Web
- Web architecture principles
  - Simplicity
  - Orthogonality of specifications
    - Identification vs. interaction vs. representation
    - Better manageability, independent evolution
  - Extensibility
    - Extending and subsetting
- Web of documents
  - Wealth of data
  - Mostly in HTML, XML – *data islands*
Semantic Web

• Semantic data
  – Powerful queries, inference, reasoning

• Linked data
  – Data integration and reuse
  – Combining data from different sources

• Semantic automation
  – Mediation of different vocabularies
  – Data mining
  – Automated use of Web services
Semantic Web Architecture

• Formalized components and their relationships
  – What technologies make up Semantic Web
  – What are the dependencies between components

• Roadmap for steps of developing the Semantic Web
The Semantic Web architecture and its foundations

TECHNICAL SOLUTION
Search and Query the Web I

- The Web is a constantly growing network of distributed resources.
  - More than 760 million web sites
  - Check most updated data on:

- User needs to be able to efficiently search resources/content over the Web
  - When I Google “Jaguar” do I find info on the car or the animal?

- User needs to be able to perform query over largely distributed resources
  - When is the next performance of the rock band “U2”, where it will be located, what are the best way to reach the location, what are the attractions nearby…
On2Broker is the evolution of Ontobroker, a system that aims at providing a solution to the problems discussed in the previous slides by adopting Semantic Technologies.

On2Broker is a system that processes distributed information sources and that provides intelligent information retrieval, query answering.

On2Broker relies on components of the Semantic Web Architecture: analysis of such systems driven the definition of the Semantic Web Architecture.

On2Broker: Architecture

[Adapted from Fensel et al., On2broker: Semantic-Based Access to Information Sources at the WWW]
• **Query Interface**
  – Provides a structured input structure that enable users to define their queries without any knowledge of the query language
  – Input queries are then transformed to the query language (e.g. SparQL)

• **Repository**
  – Decouples query answering, information retrieval and reasoning
  – Provide support for materialization of inferred knowledge
On2Broker Components II

• Crawlers and Wrappers (or Info Agent)
  – Extract knowledge from different distributed and heterogeneous data sources
  – RDF-A pages and RDF repositories can be included directly
  – HTML and XML data sources requires processing provided by wrappers to derive RDF data

• Inference Engine
  – Relies on knowledge imported from the crawlers and axioms contained in the repository to support query answers
  – Adopts horn-logic and closed world assumption
On2Broker: Example

1. Who is known by Dieter Fensel?

2. SELECT DISTINCT ?s ?o
   WHERE { ?s foaf:knows ?o . } ...

3. Extract RDF from: fensel.com dblp ...

4. Extends KB:
   if "x dblp:coauthor y"
   then "x foaf:knows y"
   if "y foaf:knows x" then
   "x foaf:knows y"

5. John Domingues and Frank van Harmelen are known by Dieter Fensel
SemWeb Architecture: Requirements

- **Extensibility**
  - Each layer should extend the previous one(s)

- **Support for data interchange**
  - Using data from one source in other applications

- **Support for ontology description with different complexity**
  - Including rules

- **Support for data query**

- **Support for data provenance and trust evaluation**

see the Semantic Web Roadmap: [http://www.w3.org/DesignIssues/Semantic.html](http://www.w3.org/DesignIssues/Semantic.html)
Semantic Web Stack

Adapted from http://en.wikipedia.org/wiki/Semantic_Web_Stack

User interface and applications

Trust

Proof

Unifying logic

Querying: SPARQL

Ontologies: OWL

Rules: RIF

Taxonomies: RDFS

Data: RDF

Syntax: XML

Identifiers: URI

Characters: UNICODE

Cryptography
UNICODE, URI and XML

• UNICODE is the standard international character set
  – E.g. used to encode the data in the repository

• Uniform Resource Identifiers (URIs) identify things and concepts
  – E.g. used to identify resources on the Web and in the repository

• eXtensible Markup Language (XML) is a markup language used for data exchange
  – E.g. format that can be wrapped into RDF and imported into the repository
RDF, RDFS and OWL

• Resource Description Framework (RDF) is the HTML of the Semantic Web
  – Simple way to describe resources on the Web
  – Based on triples <subject, predicate, object>
  – Various serializations, including one based on XML
  – A simple ontology language (RDFS)
  – E.g. language used to store the data in the repository
  – More in Lecture 3

• Web Ontology Language (OWL) is a more complex ontology language than RDFS
  – Layered language based on DL
  – Overcomes some RDF(S) limitations
  – E.g. ontology language used to define the schemas used in the repository
  – More in Lecture 7
SPARQL and Rule Languages

- **SPARQL**
  - Query language for RDF triples
  - A protocol for querying RDF data over the Web
  - E.g. language used to query the repository from the user interface
  - More in lecture 6

- **Rule languages (esp. Rule Interchange Format RIF)**
  - Extend ontology languages with proprietary axioms
  - Based on different types of logics
    - Description Logic
    - Logic Programming
  - E.g. used to enable reasoning over data to infer new knowledge
  - More in lecture 8
Logics, Proof and Trust

- **Unifying logic**
  - Bring together the various ontology and rule languages
  - Common inferences, meaning of data

- **Proof**
  - Explanation of inference results, data provenance

- **Trust**
  - Trust that the system performs correctly
  - Trust that the system can explain what it is doing
  - Network of trust for data sources and services
  - Technology and user interface

- Many open problems, topics for future research
More than a-z, A-Z

UNICODE
Character Sets

• ASCII – 7 bit, 128 characters (a-z, A-Z, 0-9, punctuation)

• Extension code pages – 128 chars (ß, Ä, ñ, ø, Š, etc.)
  – Different systems, many different code pages
  – ISO Latin 1, CP1252 – Western languages (197 = Å)
  – ISO Latin 2, CP1250 – East Europe (197 = Í)

• Code page is an interpretation, not a property of text

• Thus if we do not interpret correctly the code page, the result visualized will not be the expected one
UNICODE: a unambiguous code

- We need a solution that can be unambiguously interpreted, i.e. where to a code correspond a single character and viceversa

- That’s why UNICODE was created!

$Å L Æ pheric U+0024 U+00C5
U+0139 U+00C6 U+03AE

5%8 ♥ Жып U+0643 U+215D U+2665 U+0416 U+0E0D
• ISO standard
  – About 110'000 characters, space for 1'000'000
  – Unique code points from U-0000 through U-FFFF to U-10FFFF
  – Well-defined process for adding characters
  – Most recent version 6.3 (Sep 2013)

• When dealing with any text, simply use UNICODE
  – Character code charts: http://www.unicode.org/charts/

• See also:
  – http://tbray.org/ongoing/When/200x/2003/04/06/Unicode
How to identify things on the Web

**URI: UNIFORM RESOURCE IDENTIFIERS**
Identifier, Resource, Representation

URI
http://weather.example.com/oaxaca

Resource
Oaxaca Weather Report

Representation
Metadata:
Content-type: application/xhtml+xml

Data:
<!DOCTYPE html PUBLIC "...
"http://www.w3.org/... 
<html xmlns="http://www... 
<head> 
<title>5 Day Forecaste for Oaxaca</title> 
... 
</html>

Taken from http://www.w3.org/TR/webarch/
A Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource on the Internet.

A URI can be a URL or a URN.

A Uniform Resource Name (URN) defines an item's identity:
- The URN `urn:isbn:0-395-36341-1` is a URI that specifies the identifier system, i.e. International Standard Book Number (ISBN), as well as the unique reference within that system and allows one to talk about a book, but doesn't suggest where and how to obtain an actual copy of it.

A Uniform Resource Locator (URL) provides a method for finding it:
- The URL `http://www.sti-innsbruck.at/` identifies a resource (STI's home page) and implies that a representation of that resource (such as the home page's current HTML code, as encoded characters) is obtainable via HTTP from a network host named `www.sti-innsbruck.at`.
• **Examples**
  - mailto:John.Doe@example.com
  - news:comp.infosystems.www.servers.unix
  - telnet://melvyl.ucop.edu/

• **URI Syntax**  **scheme: [/authority] [/path] [?query] [#fragid]**
  - The scheme distinguishes different kinds of URIs
  - Authority normally identifies a server
  - Path normally identifies a directory and a file
  - Query adds extra parameters
  - Fragment ID identifies a *secondary resource*
URI Syntax cont’d

- Reserved characters (like /:?#@&+* )
- Many allowed characters
- Rest percent-encoded from UTF-8
  - http://google.com/search?q=technikerstra%C3%9Fe

- IRI – Internationalized Resource Identifier
  - Allows whole UNICODE
  - Specifies transformation into URI – mostly UTF-8 encoding
URI Schemes

- Schemes partition the URI space into subspaces
- Schemes can add or clarify properties of resources
  - Ownership (how authorities are formed)
  - Persistence (how stable the URIs should be)
  - Protocol (default access protocol)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Description</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>Host-specific file names</td>
<td>[1738]</td>
</tr>
<tr>
<td>ftp</td>
<td>File Transfer Protocol</td>
<td>[1738]</td>
</tr>
<tr>
<td>http</td>
<td>Hypertext Transfer Protocol</td>
<td>[2616]</td>
</tr>
<tr>
<td>https</td>
<td>Hypertext Transfer Protocol Secure</td>
<td>[2818]</td>
</tr>
<tr>
<td>im</td>
<td>Instant Messaging</td>
<td>[3860]</td>
</tr>
<tr>
<td>imap</td>
<td>internet message access protocol</td>
<td>[5092]</td>
</tr>
<tr>
<td>ipp</td>
<td>Internet Printing Protocol</td>
<td>[3510]</td>
</tr>
<tr>
<td>iris</td>
<td>Internet Registry Information Service</td>
<td>[3981]</td>
</tr>
<tr>
<td>ldap</td>
<td>Lightweight Directory Access Protocol</td>
<td>[4516]</td>
</tr>
<tr>
<td>mailto</td>
<td>Electronic mail address</td>
<td>[2368]</td>
</tr>
<tr>
<td>mid</td>
<td>message identifier</td>
<td>[2392]</td>
</tr>
</tbody>
</table>

From http://www.iana.org/assignments/uri-schemes.html
How to exchange structured data on the Web

XML: EXTENSIBLE MARKUP LANGUAGE
eXtensible Markup Language

• Language for creating languages
  – “Meta-language”
  – XHTML is a language: HTML expressed in XML
• W3C Recommendation (standard)
  – XML is, for the information industry,
    what the container is for international shipping
  – For structured and semistructured data
• Main plus: wide support, interoperability
  – Platform-independent
• Applying new tools to old data
Structure of XML Documents

• Elements, attributes, content
• One root element in document
• Characters, child elements in content
XML Element

• Syntax `<name>contents</name>`
  – `<name>` is called the opening tag
  – `</name>` is called the closing tag

• Examples
  `<gender>Female</gender>`
  `<story>Once upon a time there was…. </story>`

• Element names case-sensitive
Attributes to XML Elements

- Name/value pairs, part of element contents
- Syntax
  
  \(<\text{name attribute\_name=}"\text{attribute\_value}"\text{>}\text{contents}</\text{name}>\)
- Values surrounded by single or double quotes

- Example
  
  \(<\text{temperature unit=}"F"\text{>}64</\text{temperature}>\)
  \(<\text{swearword language=}"fr"\text{>}\text{con}</\text{swearword}>\)
Empty Elements

- Empty element: `<name></name>`
- This can be shortened: `<name/>`
- Empty elements may have attributes

- Example
  `<grade value='A'/>`
Comments

- May occur anywhere in element contents or outside the root element
- Start with <!--
- End with -->
- May not contain a double hyphen
- Comments cannot be nested

- Example:
  <element>content
    <!-- a comment, will be ignored in processing -->
  </element>
  <!-- comment outside the root element -->
Nesting Elements

- Elements may contain other (child) elements
  - The containing element is the parent element
- Elements must be properly nested

- Example with improper nesting:
  <b>bold <i>bold-italic</b> italic?</i>

- The above is not XML (not well-formed)
Special Characters in XML

• < and > are obviously reserved in content
  – Written as &lt; and &gt;

• Same for ' and " in attribute values
  – Written as &apos; and &quot;

• Now & is also reserved
  – Written as &amp;

• Any character: &#223; or &#xdf; → ß
  – Decimal or hexa-decimal unicode code point

• Elements and attributes whose name starts with “xml” are also special
Uses of XML

• Document mark-up – XHTML
  – HTML is a language, so it can be expressed in XML

• Exchanged data
  – Scalable vector graphics – SVG
  – E-commerce – ebXML
  – Messaging in general – SOAP
  – And many more standards

• Internal data
  – Databases
  – Configuration files

• Etc.
Why XML?

• For semistructured data:
  – Loose but constrained structure
  – Unspecified content length
• For structured data:
  – Table(s) or similar rows
  – Well-defined structure, data types
  – Good interoperability
  – But: requirements for quick access, processing
XML Parsers

• Document Object Model (DOM) builder
  – Creates an object model of XML document
  – In-memory representation, random access
  – DOM complex, simpler JDOM etc.

• Simple API for XML parsing (SAX)
  – Views XML as stream of events
  – el_start("date"), attribute("day", "10"), el_end("date")
  – Calls your handler
  – DOM builder can use SAX

• Pull parsers
  – You call source of XML events
How to distinguish categories of resources

NAMESPACE
The Problem

- Documents use different vocabularies
  - Example 1: CD music collection
  - Example 2: online order transaction

- Merging multiple documents together
  - Name collisions can occur
    - Example 1: albums have a `<name>`
    - Example 2: customers have a `<name>`
  - How do you differentiate between the two?
The Solution: Namespaces!

- **What is a namespace?**
  - A syntactic way to differentiate similar names in an XML document

- **Binding namespaces**
  - Uses Uniform Resource Identifier (URI)
    - e.g. “http://example.com/NS”
  - Can bind to a named or “default” prefix
Namespace Binding Syntax

• Use “xmlns” attribute
  – Named prefix
    • `<a:foo xmlns:a='http://example.com/NS'/>`
  – Default prefix
    • `<foo xmlns='http://example.com/NS'/>`

• Element and attribute names are “qualified”
  – URI, local part (or “local name”) pair
    • e.g. `{ “http://example.com/NS” , “foo” }`
Example Document I

• Namespace binding

<?xml version='1.0' encoding='UTF-8'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <desc xmlns:html='http://www.w3.org/1999/xhtml'>
    </desc>
  </item>
</order>
• Namespace scope

```xml
<?xml version='1.0' encoding='UTF-8'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <desc xmlns:html='http://www.w3.org/1999/xhtml'>
    </desc>
  </item>
</order>
```
• Bound elements

```xml
<?xml version='1.0' encoding='UTF-8'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <desc xmlns:html='http://www.w3.org/1999/xhtml'>
    </desc>
  </item>
</order>
```
How to define XML document structures

XML SCHEMA
What is it?

• A grammar definition language
  – Like DTDs but better
    • Uses XML syntax
  – Defined by W3C

• Primary features
  – Datatypes
    • e.g. integer, float, date, etc…
  – More powerful content models
    • e.g. namespace-aware, type derivation, etc…
XML Schema Types

• Simple types
  – Basic datatypes
  – Can be used for attributes and element text
  – Extendable

• Complex types
  – Defines structure of elements
  – Extendable

• Types can be named or “anonymous”
Simple Types

• DTD datatypes
  – Strings, ID/IDREF, NMTOKEN, etc…

• Numbers
  – Integer, long, float, double, etc…

• Other
  – Binary (base64, hex)
  – QName, URI, date/time
  – etc…
Deriving Simple Types

• Apply facets
  – Specify enumerated values
  – Add restrictions to data
  – Restrict lexical space
    • Allowed length, pattern, etc…
  – Restrict value space
    • Minimum/maximum values, etc…

• Extend by list or union
A Simple Type Example

• Integer with value (1234, 5678]

  <xsd:simpleType name='MyInteger'>
    <xsd:restriction base='xsd:integer'>
      <xsd:minExclusive value='1234'/>
      <xsd:maxInclusive value='5678'/>
    </xsd:restriction>
  </xsd:simpleType>
A Simple Type Example II

- Validating integer with value (1234, 5678]
  
  `<data xsi:type='MyInteger'></data>` INVALID
  `<data xsi:type='MyInteger'>Andy</data>` INVALID
  `<data xsi:type='MyInteger'>-32</data>` INVALID
  `<data xsi:type='MyInteger'>1233</data>` INVALID
  `<data xsi:type='MyInteger'>1234</data>` INVALID
  `<data xsi:type='MyInteger'>1235</data>` INVALID
  `<data xsi:type='MyInteger'>5678</data>` INVALID
  `<data xsi:type='MyInteger'>5679</data>` INVALID
Complex Types

• **Element content models**
  – Simple
  – Mixed
    • Unlike DTDs, elements in mixed content can be ordered
  – Sequences and choices
    • Can contain nested sequences and choices
  – All
    • All elements required but order is not important
A Complex Type Example I

• Mixed content that allows `<b>`, `<i>`, and `<u>`
  
  ```xml
  <xsd:complexType name='RichText' mixed='true'>
    <xsd:choice minOccurs='0' maxOccurs='unbounded'>
      <xsd:element name='b' type='RichText'/>
      <xsd:element name='i' type='RichText'/>
      <xsd:element name='u' type='RichText'/>
    </xsd:choice>
  </xsd:complexType>
  ```
• Validation of RichText

<content xsi:type='RichText'></content>
<content xsi:type='RichText'>Andy</content>
<content xsi:type='RichText'>XML is <i>awesome</i>.</content>
<content xsi:type='RichText'>XML is <i>awesome</i>.</content>
<content xsi:type='RichText'><B>bold</B></content> INVALID
<content xsi:type='RichText'><foo/></content> INVALID
Building On The Foundations

- RDF for semantic data
  - Graphs of linked data
  - Semantic Web
- Any XML or HTML can support translation to RDF
  - GRDDL: a pointer to a transformation
  - RDFa: RDF in XHTML
  - Makes existing data part of the Semantic Web
- XML has encryption and digital signature
  - Necessary technologies for data provenance, trust
Web of Linked Data

As of March 2009
• **Resource Description Framework**
  – Metadata: about Web resources
  – But also any other data
• **Graphs of resources interlinked with properties**
  DieterFensel teaches SemanticWebCourse
  knows FedericoFacca
  FedericoFacca teaches WebEngineeringCourse
• **Ontology languages for data schemas**
  – Various properties: teaches, knows
  – Classes of resources: Person, Professor, Course
• **SPARQL for querying the data**
The Semantic Web architecture in practice

ILLUSTRATION BY A LARGER EXAMPLE
• All the data about the conference is part of the SemWeb
  – Date, location
  – Organizers, peer-review committees
  – Articles (papers), their authors
  – Detailed program schedule

• Each SemWeb architecture layer plays a role

• ISWC is annotating conference data using Semantic Web technologies
  – Currently available data regards only papers and authors
  – This could be extended to support features discussed above
Foundation Layers

• UNICODE
  – All participants' names should be in UNICODE because they are international: Denny Vrandečić, Diego Meroño, François Maué
  – Same for paper titles: "α-decay and β-decay of heavy atoms"

• URI: All important things must have identifiers, for example:
  – Participant: http://data.semanticweb.org/person/tom-heath
  – Participant's affiliation: http://data.semanticweb.org/organization/talis-information-limited
Data Layers

- **XML**
  - The HTML pages should be in XHTML
  - The RDF data (below) should be in RDF/XML
  - News feed should be in Atom (an XML format)

- **RDF**
  - The conference dataset, and any useful subsets, should be published in RDF for download; for example:
Ontologies, Query

• **RDFS, OWL**
  - The conference would use various vocabularies and ontologies, such as:
  - FOAF (Friend of a friend) for talking about the attendees and authors/presenters
  - Dublin Core for paper metadata
  - Calendar ontology for the program

• **SPARQL**
  - The conference server should have a public SPARQL endpoint that can be used for queries over the conference data
  - [http://data.semanticweb.org/snorql/](http://data.semanticweb.org/snorql/)
Browsing ISWC Data

http://data.semanticweb.org/person/tom-heath/html
Querying ISWC Data

http://data.semanticweb.org/snorql/
That’s almost all for today…

SUMMARY
Things to Keep in Mind

- Semantic Web builds on the Web
- For any text, use UNICODE, probably UTF-8
- URIs can identify anything
  - Not only documents on the Web
- XML helps with data exchange, interoperability
- XML languages are distinguished with namespaces
References

• Mandatory:
  – http://www.w3.org/TR/webarch/
  – http://www.w3.org/DesignIssues/Architecture.html

• Further reading:
  – http://www.w3.org/Provider/Style/URI
  – http://www.ietf.org/rfc/rfc3986.txt
  – http://www.unicode.org/charts/
  – http://tbray.org/ongoing/When/200x/2003/04/06/Unicode
  – http://www.w3.org/TR/xml/
  – http://www.w3.org/TR/xml-names/
  – http://www.w3.org/TR/xmlschema-1/
  – Fensel et al., On2broker: Semantic-Based Access to Information Sources at the WWW
  – Fensel et al.: Ontobroker in a Nutshell
References

• Wikipedia links:
  • http://en.wikipedia.org/wiki/Semantic_Web_Stack
  • http://en.wikipedia.org/wiki/URI
  • http://en.wikipedia.org/wiki/Unicode
  • http://en.wikipedia.org/wiki/XML
  • http://en.wikipedia.org/wiki/XML_Namespaces
  • http://en.wikipedia.org/wiki/RDF_Schema
  • http://en.wikipedia.org/wiki/Web_Ontology_Language
  • http://en.wikipedia.org/wiki/SPARQL
  • http://en.wikipedia.org/wiki/Rule_Interchange_Format
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Semantic Web Architecture</td>
</tr>
<tr>
<td>3</td>
<td>Resource Description Framework (RDF)</td>
</tr>
<tr>
<td>4</td>
<td>Web of data</td>
</tr>
<tr>
<td>5</td>
<td>Generating Semantic Annotations</td>
</tr>
<tr>
<td>6</td>
<td>Storage and Querying</td>
</tr>
<tr>
<td>7</td>
<td>Web Ontology Language (OWL)</td>
</tr>
<tr>
<td>8</td>
<td>Rule Interchange Format (RIF)</td>
</tr>
<tr>
<td>9</td>
<td>Reasoning on the Web</td>
</tr>
<tr>
<td>10</td>
<td>Ontologies</td>
</tr>
<tr>
<td>11</td>
<td>Social Semantic Web</td>
</tr>
<tr>
<td>12</td>
<td>Semantic Web Services</td>
</tr>
<tr>
<td>13</td>
<td>Tools</td>
</tr>
<tr>
<td>14</td>
<td>Applications</td>
</tr>
</tbody>
</table>
Questions?