Information Sharing for e-Government

Information integration. Semantic Web. Ontologies.

Pablo Fillottrani   Elsa Estévez

Center for Electronic Governance
United Nations University - International Institute for Software Technology

July 2010

Information Integration. Semantic Web. Ontologies.

Data strategy is needed

- data is a primary asset in most organizations, but it is common to have no plan for it.
- the value of data is not well understood.
- choices of databases, platforms and applications are made based on small local needs without enterprise-wide guidelines
- an open door to those who want to pursue own objectives (e.g. introducing new technologies or tools without validation)
- exercise: comparison with financial assets

Data strategy vision

- a data strategy must conform overall IT strategy, which in turn must conform the business strategy.
- every organization has strategic goals, which are unique because the organization is unique. Without these goals there is little support for decision making.
- examples of IT strategy goals:
  - deliver systems faster
  - improve quality of systems
  - reduce costs
  - attract and retain good people
- these goals should not be accepted without extensive discussion, they all have costs
- for data strategy, a supporting infraestructure or architecture must be in place

Data strategy components

- data integration
- data quality
- metadata
- data modelling
- organizational roles and responsibilities
- performance and measurement
- security and privacy
- business intelligence
- business value of data

Developing a data strategy

- data stratetegy developing team: CTO (leader), a data administrator, a strong DBA, and a business analyst.
- current data environment assessment: existing DBMSs, internal skills and culture, leqacy systems, etc.
- exercise: apply questionnaire to assist your organization in gathering basic information about its current data environment.
- costs of developing a data strategy:
  - profiling tools for data quality needs extensive effort
  - capturing metadata both automatically and manually involves costs
  - as it becomes apparent that there are holes in security and privacy, you need to assign staff to administer and audit
  - most organizations find data modelling activities insufficient

Integrated view of data

- information integration may mean from two systems passing data back and forth, to a shared environment in which all data elements are unique and non-redundant and are used by several applications
- data elements come from multiple sources, and are usually duplicated and often inconsistent
- for example, an integrated view of a customer may come from
  - what the customer fills out on a loan application
  - transactions the customer performs with the organization
  - demographic data on the customer that is available from external vendors
  - data from order management applications
  - information gathered when the customer calls a customer service department
- exercise: record all data elements that you must provide when asking for a given service, and how many times each

Ineffective technology solutions

- companies often look for silver-bullets solutions when they realise that data integration is the key to thriving in the fast-paced information-oriented economy
- however, they quickly discover that the real issues underlying current disintegrated data cannot be solved only with technology.
- a fundamental change in the way data is managed is needed
- we will analise three silver-bullet technologies:
  1. Enterprise Resource Planning (ERP)
  2. Data warehousing (DW)
  3. Customer relationship management (CRM)

Enterprise Resource Planning

- ERP is a collection of functional modules used to integrate operational data to support seamless operational business process for the enterprise
- were meant to solve the redundant and inconsistent operational data mess by consolidating this data
- extensive research including analysing all data elements for its definitions, contents, semantics and business rules was necessary but rarely done due to time constraints
- ERP data conversion does not include
  - finding data elements with multiple meanings
  - finding missing relationships between business entities
  - finding and resolving duplicate keys
  - validating data content among dependent data elements across files
  - finding and extracting processing logic embedded in the data values

Enterprise Resource Planning

- market share ERP vendors 2005

Vendor	**Revenue** (millions u$s)	**Market share**
SAP	1949	30.33
Oracle Applications	1.374	21,38
The Sage Group	1.121	17,44
Microsoft Dynamics	916	14,25
SSA Global Technologies	464	7.22

ERP evaluation

ERP promises	**ERP realities**
data integration	cross-organizational integration is limited to the operational functions converted to the ERP packages.
no more redundancy	reduced data redundancy is limited to the converted operational systems.
consistency of data content	data content is almost as consistent or inconsistent as it was on legacy systems, because of rarely changed or cleaned during conversion.
improved data quality	only limited or no data cleansing was performed during conversion.
easy reporting	poor quality or unusable reports (often the reason for a data warehouse initiative).
easy maintenance	complicated and costly maintenance.

Data Warehousing

- a DW delivers a collection of integrated data used to support the decision making process for the enterprise
- properly implemented meant extensive analysis of the operational data to find the data redundancy and data inconsisten problems, and to correct them
- this type of analysis cannot be magically performed by a tool, it requires business analysts
- in the end, few DW projects have the necessary user involvement to perform such rigurous analysis
- some DW products are: IBM DB2, Oracle, Microsoft SQL Server, Sun MySQL, SAND/DNA Access, and Teradata

DW evaluation

DW promises	DW realities
data integration	stove-pipe data marts, each with ts own extract/transform load staging area.
no more redundancy	continued, sometimes even increased data redundancy.
consistency of data content	data is still inconsistent among data marts (no central staging area, no reconciliation).
improved data quality	little improvement of data quality.
historical data	historical data limited to departmental views.
unlimited, ad-hoc reporting	limited, ad-hoc reporting (too complicated, missing relationships, and poor performance).
reliable trend analysis reporting	inconsistent trend analysis report among data marts.
faster data delivery and data access	drill-down capabilities are slow.
busines intelligence capabilities	compromised by inconsistent and unreliable key performance indicators.

Customer relationship management

- CRM attempts to integrate customer information with product information through related business functions, such as sales, marketing, and order fulfillment
- over more than two decades, CRM has evolved into a sophisticated set of tools and applications
- unfortunately, CRM conversions follow the traditional habits of source-to-target mapping without extensive data analysis and little data cleansing. Data is usually moved “as-is”
- market share CRM vendors 2008

Vendor	**Revenue** (millions u$s)	**Market share**
SAP	2.055	22,5
Oracle	1.475	16,1
Salesforce.com	965	10,6
Microsoft	581	6,4
Amdocs	451	4,9

CRM evaluation

CRM promises	CRM realities
data integration	more stove-pipe systems; most CRM modules are not integrated, especially when purchased from different vendors.
no more customer redundancy	still no single, cross-organizational view of customer; different department continue to keep and utilize redundant customer files and databases.
improved data quality	customer data is often still as dirty as it was on legacy files.
customer profitability analysis and customized product pricing	customer often still have multiple keys, which make profitability analysis difficult.
customer intimacy	privacy issues and government regulations override customer intimacy requirements.
geographic market potential and increased customer wallet share	customer wallet share is difficult to estimate; external customer data is incomplete or dirty.

Gaining management support

- integrating information is difficult and expensive, therefore it requires top management support
- common goals for integrated information are:
  - reducing overall costs integrated data minimizes supplier costs, reduces programming costs associated with building interfaces and maintaining redundant applications, reduces fraud, and minimizes efforts to reconcile inconsistent reports
  - increasing revenue integrated data gives a better perspective of products and services, and how much money you can make or lose on them. Comparative analysis of data about competitors exposes opportunities for marketing and sales strategy.
  - improving the supply chain a popular business intelligence (BI) application at manufacturing is the supply chain intelligence (SCI) which proactive alerts about low inventory or shipment delays. This requires the integration of product data, inventory levels, order placement and shipments.

Gaining management support

- common goals for integrated information are:
  - reducing product development times increasing the efficiency of your business process and your daily activities ultimately decreases the time to market. BI applications like business performance management (BPM) and business activity monitoring (BAM) need integrated set of metrics about your business process and your organization.
  - providing better customer service integrated customer and sales data provides a more accurate view of customers and purchases, generating the opportunity to a more tailored solution, which should raise customer satisfaction. Thus, customers receive the right product at the right time using the right channel while also resolving customer's inquiry or complaint. Customer do not have to deal with multiple points of contact or receive multiple solicitations and legal notifications.

Gaining management support

- common goals for integrated information are:
  - complying with information legislation a number of law mandate complete, consistent, and accurate information that can be realized only with integrated data. For example, the Data Protection Act gives individuals the right to know what information is held about them or financial regulations require reports defining responsibilities for providing inaccurate information.
  - improving control of assets a pool of integrated information about corporate assets is a necessary components of a organization's business quality. Market analysis use business quality metrics in their evaluation.
  - being a driver of business transformation (most important!) effective use of information should lead to business process improvements, such as moving to a real-time business environment, developing productive partnerships with suppliers and customers, streamlining internal operations, and actually driving up productivity and superior customer service

Business case for data integration

- data integration is costly, and it also requires that tha integration process goes to completion
- justifications are numerous and compelling, but must be reiterated at every meeting with senior management
- some compelling reasons are:
  - reduced risks of litigation patient data is a prime example for the need for accurate, consistent and complete information.
  - reduced risks of compliance failure for information required by regulatory entities
  - retirement of legacy systems an integrated database can provide the opportunity to retire old legacy files and databases, which also allows less costly maintenance of these items.
  - increased business agility a more flexible and responsive IT infrastructure, which gives the opportunity to respond faster to the changing business environment.
  - fraud detection potential losses from undetected fraud and abuse can easily justify any cost associated with information integraton

Opportunities for data integration

- integrating information is much more than connecting a few databases, building bridges among applications or consolidating disparate database
- it also addresses data inconsistency and redundancy found in each organizational
- this process produces an inventory of data (either as logical models or physical databases) in which all information is integrated whithin a common buisiness context
- there are numerous data integration opportunities in each organization. Example: in a university, data from students, instructors, courses, enrollments, drop off, classroom, facilities, exams
- example: in a hospital, data from patients, guests, services, stays, facilities, providers
- exercise: record the data integration opportunites for federal and local governments

Challenges of business data integration (I)

- know your business entities business entities are at the heart of organizations' processes. The information about these entities must allow the organization to take clever and more effective actions.
- merges, acquisitions and reorganizations these processes are usually taken considering the potential for cost savings. From the IT point of view this potential is very difficult to estimate, and almost imposible to achieve without information integration.
- data redundancy it is not uncommon to find data repeated 10, 20 or more tiime within the organization, and nobody knows which file is the system of record and which copy of data most accurately reflects the real world. Redundant data results in a loss of control, misunderstandings and a continuing bad reputation for IT.

Challenges of business data integration (II)

- data lineage it's the process of tracking the descendent data elements from their origins to the current instantiations. Documenting the data lineage for each data element in a data dictionary or a metadata repository provides the origin and subsequent alterations and duplications. Remember the immense effort of Y2K impact because of absence of data lineage.
- multiple DBMSs can be an obstacle to data integration, because of incompatible formats and data definitions, and inefficient use of optimizers.

Data integration prioritization

- data integration is performed in two layers: logical and physical.
- logical data integration is the process of building an enterprise logical data model
- physical data integration is the process of filtering redundant data, retiring redundant files and databases, and combining data elements fro the same business entity into one physical database.
- an enterprise-wide information integration effort must be carved up into small iterative projects, starting with the most critical data.
- some data might not be suitable for integration at all. For example, department specific data, or highly secured data.
- the business people working with the data integration team must determine which data is most appropriate for integration

Issues for data integration prioritization

- how will the data be used?
- political issues some data owners might not want the data they control to be integrated or made available to the entire organization. This can be solved by the CEO who must clearly state that these data sources will be integrated.
- security issues managers might justify withholding the data based on security considerations, legitimate or not.
- regulatory and legal issues some industries require a firewall between the vertical functions of their business. For example, financial services and healthcare industries.

Risks of data integration

- management commitment senior management must be committed to a long-run project, and must supportingthe continued sustenance of the initiative. Short-term oriented management must understand that information integration is a prerequisite to execute BI applications.
- cost and effort information integration requires HW and SW, but also dedicated and smart internal personnel with strong analytical and technical skills. It is possible to bring in consultants for the initial stages, but they cannot be the subject matter experts.
- sustainability integrated data must be kept reasonably current, based on how the data is used and the user requirements for currency.
- external data most likely it is necessary to bring data from suppliers, distributors and partners. This is often incomplete, less than clean, difficult to match to own data, out-of-date, and poorly documented.
- data selection and validation validation is relevant in selecting which data to use for the integration process. The dirtier the data the more difficult it is to integrate.

Consolidation and federation

- consolidation and federation are alternatives to true data integration
- Data consolidation means gathering data elements that describe the same business entity (e.g. CUSTOMER) from different source databases and storing them in another database. Integration goes beyond that, enforcing data uniqueness, enforcing business rules and eliminating data redundancy. Integration also means that data is formally named, unambiguosly defined, well architected, and its lineage properly documented.

Consolidation and federation

- Data federation data does not have to be consolidated or moved. A simplistic move is to install a middleware (such as EAI) and providing metadata to make people aware of the data, how it is and how to get to it. This eliminates the need to convert, match, filter, and merge the data, however requieres complete and current metadata as well as clean and consistent business data. This approach is not mutually exclusive with integration.

Data integration capability maturity model

Level 1	limited data federation; often with redundant and inconsistent data
Level 2	limited data consolidation; documenting redundancies and inconsistencies
Level 3	data integration initiated; new “disintegration” is discouraged
Level 4	data integration widely adopted; “disintegration” is penalized
Level 5	enterprise-wide data integration and other data strategy principles practiced by all departments in the organization

Actions towards information integration (I)

1. measure the cost of “disintegrated” data
2. educate show this cost to executives and managers
3. get sponsorship initiatives can be sustained only with strong executive sponsorship, not only CIO or CTO
4. prioritize find out which information is the most critical, who uses it and the regulations affecting it
5. research integration requires a in-depth knowledge of business data: semantics, lineage, owners, domains, business rules, security and privacy considerations

Actions towards information integration (II)

6. recruit recruit full-time business people and technicians who will actively participate on data integration process if not lead them
7. plan the team must asses and discuss the different approaches, tools and architectures to determine how to best integrate data
8. execute execute the plan, measure the results, and report results to the management
9. adapt as you learn more about the business data, be flexible and adapt your plan as well as your approaches

Recognizing dirty data

- incorrect data values must adhere to valid domains, usually can be programatically enforced
- inaccurate data accuracy of dependent data values is difficult to enforce programatically
- business rules violations for example,
- inconsistent data uncontrolled data redundancy results in inconsistencies.
- incomplete data many data elements in systems have missing values or default values that are not correct
- nonintegrated data storing data redundantly and inconsistently across many sistems, primary keys that don’t match, are not unique or even don’t exist, or when development or maintenance is outsourced

Categories of data heterogenity

- business entities rules about business objects
- buisness attributes rules aboust specific attributes
- data dependency rules that apply to relationships between two or more entities or attributes
- data validity rules that govern data values

Business entity rules

1. uniqueness
2. cardinality
3. optionality

Uniqueness

1. every instance of a business entity haas its own unique identifier (unique primary key)
2. this identifier must always be known (it can't be null)
3. whenever the key is composite (it is composed of two or more attributes), it must be minimal
4. composite primary key may contain foreign keys

Cardinality

- referes to the degree of a relationship, ie the maximun number of times one business entity can be related to another
- for example: one-to-one, two-to-many, or many-to-one

Opcionality

- specifies the minimun cardinality of an entity participation in a relationship. It can be mandatory (at least one), or optional (may or may not be related)
- optionality can be combined with cardinality, to define for example one-to-one optional relationship
- every instance of an entity being referenced by another entity in a relationship must exist

Business attribute rules

1. inheritance
   - all generalized business attributes of the supertype are inherited by all its subtypes
   - the unique identifier of the supertype is the same unique identifier of its subtypes
   - all attributes of a subtype must be unique to that subtype only
2. domains domain can be a range of values, a set of values, a constrained set of values or a pattern. Each attribute domain must contain only those values that are valid for the attribute

Data dependency rules

1. entity-relationship dependency
2. attribute dependency

Entity-relationship dependency

1. the existence of a relationship depends on the state of another entity that participates in the relationship. For example, orders cannot be placed bor a customer whose status is “delinquent”
2. the existence of a relationship mandates that another relationship also exists. For example, when a customer places an order, then a salesperson must also be associated with that order
3. the existence of a relationship prohibits the existence of another relatiosnip. For example, an employee who is assigned to a project cannot be enrolled in a training program

Attribute dependency

1. the value of one attribute depends on the state of the entity in which the attribute exists. For example, when the state of a license is “granted” then the number and type of the license cannot be null
2. the correct value of an attribute depends on, or is derived from, the values of two or more other attributes. For example, the age of a person depends on the current date and the birth date
3. the allowable value of one attribute is constrained by the value of one ore more attributes in the same business entity or in a different but related entity. For example, when the category is “manager” then the age must be greater than 25
4. the existence of one attribute value prohibits the existence of another attribute value in the same entity, or in a different but related entity. For example whe For example, if age is under 18 then commission rate must be null

Data validity rules (I)

1. completeness
   a. entity completeness requires that all instances exist for all business entities
   b. relationship completeness referes to the condition that referential integrity exists among all referenced instances
   c. attribute completeness state that all attributes in an entity must exist for all instances
   d. domain completeness demands that all attribute contain allowable values and that null values can be differentiated from missing values
2. correctness requires that all data values for an attribute must be correct and representative of the attribute's definition
3. accuracy states that all data values for an attribute must be accurate in terms of the attribute's dependency rules and its state in the real world

Data validity rules (II)

4. precision specifies that all data values must be as precise as required by the attribute's requirements, business rules and intended usage
5. uniqueness
   a. every entity instance must be unique
   b. every entity must have a unique identifier
   c. every attribute must have a unique definition (no homonyms)
   d. every attribute must have a unique name (no synonyms)
   e. every attribute must have one unique domain (no overloaded attribute)
6. consistency
   a. data values for an attribute must be consistent when the attribute is duplicated for performance reasons or data distribution issues. Data should never be stored redundantly because you don't trust the data from another user or because of departamental politics
   b. duplicated data values of an attribute must be based on the same domain and on the same business rules

Origin of the Semantic Web

- Tim Berners Lee, a british physicist and a pioneer of the Internet, originally had an extended vision of the web

  "... a goal of the Web was that, if the interaction between person and hypertext could be so intuitive that the machine-readable information space gave an accurate representation of the state of people's thoughts, interactions, and work patterns, then machine analysis could become a very powerful management tool, seeing patterns in our work and facilitating our working together through the typical problems which beset the management of large organizations."

- this vision has become known as the Semantic Web
- this is probably too hard for the moment, but there is a plan to gradually achieve this goal

The "syntactic" web of today

- a graph of resources (web pages, images, sound, video) connected by links
- (Goble 2003) a place where computers do the presentation (easy) and people do the linking and interpreting (hard).
  - a hypermedia, a digital library a library of documents called (web pages) interconnected by a hypermedia of links
  - a database, an application platform a common portal to applications accessible through web pages, and presenting their results as web pages
  - a platform for multimedia for music, movie trailers, videos
  - a naming scheme unique identity for those documents
  - why not get computers to do more of the hard work?

Almost impossible to...

- complex queries involving background knowledge
  - find information about "articles about IS in countries with population less than 10 million"
- locating information in data repositories
  - travel enquiries
  - prices of goods and services
  - results of human genome experiments
- finding and using "web services"
  - visualise surface interactions between two proteins
- delegating complex tasks to web "agents"
  - book me a holiday next weekend somewhere warm, not too far away, and where they speak Spanish

The search problem (I)

- (H.Boley 2004) the World Wide Web currently includes more than four billion (often 'scrollable') pages
- when you search it for a particular page you have to "sieve" through pages more thoroughly than if you were searching for a specific cubic grain having 1mm side length tightly packed in two boxes of such grains each having, as cubes, 1000mm = 1m side length.
- search engines should be able to "understand" the "semantics" – the meaning – of Web pages far enough to enable "sensible" queries, but at the moment such "semantic search engines" only exist for specialized areas of knowledge
- most search engines use a so-called crawler, i.e. a program that periodically and automatically navigates across as many of the currently existing Web pages as possible.

The search problem (II)

- for every page the crawler mainly analyses the text components. Basically, it enters the central and frequent words of a page into a huge address book
- every word in this "address book" thus refers to a list of all the pages *in which this word was discovered by the crawler* More specifically, this list contains a summary of each page together with its URL with which you, as the inquirer, can click through to the complete result page if desired
- imagine you're looking for pages that include the (composite) word "wonder drug"to see if there is one that helps against head pain

  `== Google Search: "wonder drug" ==`
- a hard-to-penetrate silt of 24,000 pages comes

The search problem (III)

- the search result rates too low in its so-called "precision": in particular, the word "drug" is ambiguous in this composition – it can mean medicine or narcotics or perhaps both at the same time
- imagine you're looking for pages that contain the word "aspirin" in order to check it as a remedy for head pain.

  `== Google Search: Aspirin ==`
- for prevalent isolated words like this you receive much too many pages. In this case: 640,000. Despite the unambiguousness of "Aspirin" the search result is not precise enough either
- besides the grains that are precious for your search, you still receive too many irritating grains, here, e.g., pages about Aspirin for dogs.

The search problem (IV)

- however, because a Crawler enters *all the important words of an analysed page* into the address book, you can now narrow down the search by typing a whole combination of words in the search line. Then you continue to receive a page only if the crawler has discovered in it at least *all of these search words*

  `== Google Search: Aspirin “head pain” ==`
- the precision has noticeably improved: we now only get the 82,800 pages in which the words “Aspirin” and “head pain” appear together.
- but wait: Have we perhaps cut out pages because we only wrote “head pain” but not the word “migraine” which means the same thing?

The search problem (IV)

- so, as not to exclude any interesting pages, you would have to connect the words meaning the same thing with an OR combination

  ```
  == Google Search: Aspirin
                    “head hurt” OR “migraine”==
  ```
- still, there are a lot of synonyms for “head hear” that are not included in this search. Why is it necessary to know all synonyms of a word in order to obtain sensible results?

The data problem

- a typical web page consists of
  - markup for rendering information (e.g., font size and colour)
  - hyper-links to related content. Semantic content is accessible to humans but not (easily) to computers. . .
- (Rector and Horrocks 2004) what informatio
  n can we see

```
Escuela de Ciencias Informáticas 2010
Del 26 al 31 de julio
UBA-Fac. de Ciencias Exactas y Naturales
Inscripción                   Cursos
```

The data problem

- a typical web page consists of
  - markup for rendering information (e.g., font size and colour)
  - hyper-links to related content. Semantic content is accessible to humans but not (easily) to computers. . .
- what information can a machine see

```
⋆++fl2adf=?:@|5~þdf¬%l"%v⋆^(p½e]
|#°vb2ñ]@fµ“æðđ”et6%)0~{-+çæ@»“ww1!
»c°ð⋆■4gaf=¿dhkñpwwełþøeiföþ←zcx3
%%þø|2#5¬a7[q|g-<&+đ■jkł~lkoty]
```

The data problem

- a typical web page consists of
  - markup for rendering information (e.g., font size and colour)
  - hyper-links to related content. Semantic content is accessible to humans but not (easily) to computers. . .
- solution? XML markup with “meaningful” tags

```
<school>⋆++fl2adf=?:@|5~þdf¬%l"%v⋆^(p½e]</school>
|#°vb2ñ]@fμ“æðđ<date>”et6%)0~{-+çæ@</date>»“ww1!
<place>»c°ð⋆■4gaf=¿dhkñpwwełþøeiföþ←zcx3</place>
%%þø|2#5¬a7[<courses>q|g-<&+đ■jk</courses>ł~lkoty]
```

The data problem

- a typical web page consists of
  - markup for rendering information (e.g., font size and colour)
  - hyper-links to related content. Semantic content is accessible to humans but not (easily) to computers. . .
- but still the machine only sees

```
<rb¬jð>⋆++fl2adf=?:@|5~þdf¬%l"%v⋆^(p½e]</rb¬jð>
|#°vb2ñ]@fμ“æðđ<½#rv>”et6%)0~{-+çæ@</½#rv>»“ww1!
<1@“fq>»c°ð⋆■4gaf=¿dhkñpwwełþøeiföþ←zcx3</1@“fq>
%%þø|2#5¬a7[<l32x■o>q|g-<&+đ■jk</l32x■o>ł~lkoty]
```

Need to add semantics

- external agreement on meaning of annotations
  - e.g., Dublin Core for annotation of library/bibliographic information. Agree on the meaning of a set of annotation tags
  - problems with this approach: inflexible, limited number of things can be expressed
- use ontologies to specify meaning of annotations
  - ontologies provide a vocabulary of terms
  - new terms can be formed by combining existing ones
  - meaning (semantics) of such terms is formally specified
  - can also specify relationships between terms in multiple ontologies

...but it is hard

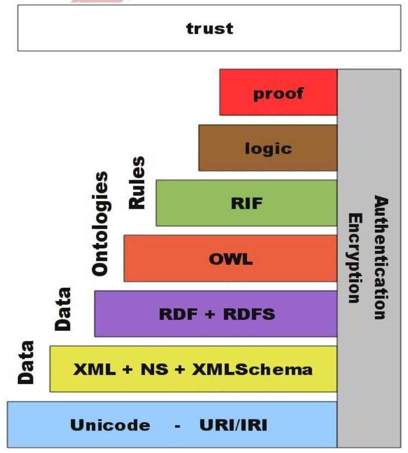
- ontology languages are tricky

  All tractable languages are useless; all useful languages are intractable

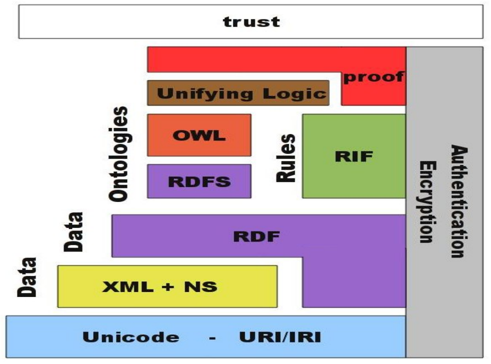
- ontologies are tricky
  - people do it too easily; but intuitions hard to formalise
- the problem has been about for 3000 years: ontologies in philosophy, ontologies in linguistics
- the semantic web means knowledge representation matters

The vision

- Tim Berners-Lee envisions the semantic web as being machine processable
- evolution can be obtained by a series of technologies and markup langauges



Current Semantic Web



- the W3C consortium `http://www.w3.org/` issues recommendations for languages in the Semantic Web

Definition

- official deffinition

  The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. (Berners-Lee et al., Scientific American, May 2001)

Principles

- the following characteristics of the Web must be preserved
  - the Web is distributed many sources, varying authority, inconsistency
  - the Web is dynamic representational needs may change
  - the Web is enormous systems must scale well
  - the Web is open-world and incomplete
- therefore the Semantic Web propones
  - clear semantics for information integration
  - flexible mechanisms for updating representations
  - efficient reasoning, able to cope with large data
  - tolerant to incomplete information; a gradual transition from “syntactic” web

URI

- a Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource on the Internet
- such identification enables interaction with representations of the resource over a network (typically the World Wide Web) using specific protocols. Schemes specifying a concrete syntax and associated protocols define each URI.
- URI syntax consists of a URI scheme name (such as `http`, `ftp`, `mailto` or `file`) followed by a colon character, and then by a scheme-specific part

URN

- a URI may be classified as a locator (URL), or a name (URN), or both
- a Uniform Resource Name (URN) functions like a person's name, while a Uniform Resource Locator (URL) resembles that person's passport number
- in other words: the URN defines an item's identity, while the URL provides a method for finding it
- example: `urn:isbn:9781420090505` is a book, while `http://www.semantic-web-book.org/index.html` is its web page

IRI

- the Internationalized Resource Identifier (IRI) is a generalization of the Uniform Resource Identifier (URI)
- while URIs are limited to a subset of the ASCII character set, IRIs may contain characters from the Universal Character Set (Unicode/ISO 10646), including Chinese or Japanese kanji, Korean, Cyrillic characters, and so forth
- e.g. `http://www.españa.com/`
- IRI provides to the Semantic Web a basic naming mechanism, able to handle any international need
- through IRI/URI-s we can link any data to any data. The "network effect" is extended to the (Web) data
- it's the Web's only basic datatype

What is XML?

- XML stands for eXtensible Markup Language, it's a W3C recommendation
- it's a simplified version of SGML, designed so that information can be delivered over the Web
- it's a markup language much like HTML, but it's more flexible and adaptable
- it's designed to structure, store and transport data, not to display data
- XML tags are not predefined, you must define your own tags
- XML is designed to be self-descriptive, but machine-procesable at the same timeliness
- XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML

XML little example

```
<email> <from>prf@cs.uns.edu.ar</from>
<to>ece@cs.uns.edu.ar</to>
<date> <year>2010</year>
<month>May</month><day>20</day></date>
<subject>Reminder</subject>
<body>Don't forget about ECI course slides!</body>
</email>
```

- XML tags are "invented" by the author. They have no meaning.
- tags must be properly parenthesized
- XML data is structure has a tree shape

XML vs. HTML

HTML	XML
mark up text so it can be displayed to users	mark up data so it can be processed by computers
describes both structure (e.g. <p>, <h2>, <em>) and appearance (e.g. , <font>, <i>)	XML describes only content
fixed, unchangeable set of tags	you make up your own tags
browsers ignore and/or correct as many HTML errors as they can, so HTML is often sloppy	rules are strict and errors are not allowed

- more flexible and adaptable than HTML
- XML is Not a Replacement for HTML; XML is a complement to HTML.
- XHTML: a reformulation of HTML 4 in XML 1.0

Objectives

- XML simplifies Information Sharing; in the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.
- this makes it easier to create data that applications can share
- XML simplifies data transport with XML, data can easily be exchanged between incompatible systems.
- exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.
- since XML is independent of hardware, software and application, XML can make your data more available and useful.
- different applications can access your data. With XML, your data can be available to all kinds of "reading machines" (handheld computers, voice machines, etc), and make it more available for blind people, or people with other disabilities.

XML elements

- elements are XML building blocks, identifying the content they surround
- are delimited by starting and ending tags within angle brackets
- general format: `<element> ... </element>`
- empty element: `<element/>`
- elements are related as parents and children, defining the tree-structure of an XML document
- elements can have different content: other elements, text, mixed, empty

XML attributes

- attributes are name-value pairs that occur inside start-tags after element name, like:

```
<element attribute="value"> . . . </element>
```

- provide additional information about elements that often is not a part of data
- attributes and elements are somewhat interchangeable, like in

```
<name>
  <first>Juan</first><last>Pérez</last>
</name>
<name first="Juan" last="Pérez"></name>
```

- in general, metadata (data about data) should be stored as attributes, and that data itself should be stored as elements. But since there is no semantics, this separation is not neat

Well-formed XML documents

- a well-formed XML document has correct XML syntax, according to the following rules:
  - must have one and only one root element
  - each elements must have a closing tag
  - elements must be properly nested
  - tags are case sensitive
  - attribute values must be quoted
  - document must start with an XML declaration, like

```
<?xml version="1.0" encoding="UTF-8"
  standalone="yes"?>
```

Valid XML documents

- a valid XML document is a well-formed XML document whose elements also conform to the rules of a Document Type Definition (DTD) declared in the documentclass
- DTD part specifies some structure for the document, by giving necessary relationships and attributes for elements
- can be inline in a XML file or as an external reference
- but DTD language has several drawbacks (for example, its syntax is not XML), so W3C supports an XML-based alternative, called XML Schema:

DTD example

```
<!DOCTYPE email [
<!ELEMENT email (to,from,date,subject,body)>
<!ELEMENT to (#PCDATA)><!ELEMENT from (#PCDATA)>
<!ELEMENT date (year,month,day)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT date (#PCDATA)>
]>
```

Namespaces

- element and attribute names in XML are defined by the user, so there is a possible conflict when combining data from different users
- XML namespaces is a simple mechanism for creating globally unique names for the elements and attributes of your markup language.
- benefits: disambiguation of the meaning of identical names in different XML-based markup languages
- namespaces are implemented by requiring every XML name to consist of two parts: a prefix and a local part, separated by a colon:

```
<uns:student uns:id="34223">
  Juan Pérez
</uns:student>
```

Prefixes

- before using prefixes in names, they must be declared
- prefixes are declared by associating it to and URI in the start tag of an element

```
<uns:student
     xlmns:uns="http://www.uns.edu.ar/">
```

- this definition is local to the element (including child elements), thus the same prefix can have several meaning in one XML documentclass
- but the URI is global, all prefixes associated to the same URI in one or several XML documents are synonyms
- the URI is not used by XML parsers, it plays only the role of an identification

Default namespaces

- defining a default namespace for an element saves us from using prefixes in all the child elements and attributes
- it has the following syntax:

```
<student xmlns="http://www.uns.edu.ar/">
```

- default namespace apply to all prefix-less element and attribute names in the definition

DTD drawbacks

- DTD language is weak
  - you can't put any restrictions on text content
  - you have very little control over mixed content (text plus elements)
  - you have little control over ordering of elements
  - you have little control on element iteration
- DTDs are written in a strange (non-XML) format

XML Schema Definition language

- W3C's XML Schema Definition language solves these problems by giving you much more control over structure and content, and being XML-based
- in addition, this language supports namespaces and includes basic datatype definitions
- an XML Schema is an XML document. The constraints on the data are expressed in a document.
- by expressing the data constraints in a document (and using XML to express the constraints) then the schema itself becomes information! Not only is the XML instance data being checked information, but the schema itself is information. Thus, the schema can be shipped around, mined, morphed, searched, etc.

XML Schema association

- you can associate the schema of an element with

```
<email xmlns ="http://www.uns.edu.ar"
   xmlns:xsi=
     "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
      "http://www.uns.edu.ar/InternalSchema.xsd">
   ...
</email>
```

- the xsi prefix is required to uniquely identify schema related attributes
- `InternalSchema.xsd` is the location of the file with the schema definition

XML Schema type declarations

- element may have predefined built-ins simple type (`xs:string, xs:integer`, etc) or user-defined types. There are, in fact, two user-definable types: `ComplexType` and `SimpleType`
- `SimpleType`d elements are elements that only contain data, no other structure
- they may not contain attributes or sub-elements
- new simple types are defined by deriving them from existing simple types (built-in's and derived)
- simple type definitions are used when a new data type needs to be defined, where this new type is a modification of some other existing simpleType-type

XML Schema type declarations

- `ComplexType`d elements are elements that allow sub-elements and/or attributes
- complex types are defined by listing the elements and/or atributes nested within them
- there are four kinds of complex elements:
  - empty elements
  - elements that contain only other elements
  - elements that contain only text
  - elements that contain both other elements and text

Example of schema definition

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.cs.uns.edu.ar">
<xs:element name="email">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="to" type="xs:string"
         minOccurs="1" maxOccurs="unbounded"/>
   <xs:element name="from" type="xs:string" />
   <xs:element ref="date" minOccurs="0" />
   <xs:element name="heading" type="xs:string" name="date"
         minOccurs="0" />
   <xs:element name="body" type="xs:string" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="date">
 <xs:complexType>  ...  </xs:complexType>
</xs:element>
</xs:schema>
```

XML related technologies

- XML defines the minimun requirement for a language for data exchange and interoperability. It is the basis for several more specific languages
- some XML-based languages are RDF, SOAP, XHTML, SVG, GML, MusicML
- there are also other related technologies for querying, processing, formatting, transforming, and relating XML documents and information
- we will shortly review a query language XPath/XQuery, and two processing standards SAX and DOM
- other XML technologies: XSLT, XLink, XPointer, XSL-FO, XForms

XPath

- XPath is a language for finding information in an XML document
- it's a language for defining parts of an XML document, and it's a W3C recommendation
- XPath uses path expressions to navigate in XML documents, selecting nodes or node-sets in the tree-structure of the document
- these path expressions look very much like the expressions you see when you work with a traditional computer file system
  `email/date`
- also contains a library of over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more.
- XPath is used in other languages like XSLT, XQuery and XPointer

XQuery

- XQuery is designed to query XML data - not just XML files, but anything that can appear as XML, including databases.
- XQuery for XML is like SQL for relational databases
- XQuery can be used to extract information to use in a Web Service, generate summary reports, transform XML data to XHTML, and search Web documents for relevant information
- XQuery is built on XPath expressions
- XQuery is supported by all major databases, and it's a W3C recommendation
- example: "Select all email subjects from emails that prf send to ece"

```
doc("emailRecords.xml")/
   email[from='prf' and to='ece']/subject
```

SAX

- there are two categories of libraries for processing XML documents: SAX and DOM
- both are platform- and language-independent
- Simple API for XML (SAX) is a serial access parser API for XML. SAX provides a mechanism for reading data from an XML document in a sequential way
- it's a lexical, event-driven interface

DOM

- Document Object Model (DOM) defines a standard way for accessing and manipulating a XML document, by creating its tree-structure
- DOM defines the standard object model for XML, and a standard programming interface for XML, ie defines the objects and properties of all XML elements, and the methods (interface) to access them
- in other words: it is a standard for how to get, change, add, or delete XML elements
- the main difference between these two models is that access in SAX is sequential, whilst in DOM is random. But this makes it necessary for DOM to have all the document information loaded in memory at the same time for processing

XML and XML Schema is not enough (I)

- XML Schema is not a language for semantic definitions (e.g. cannot say that `hasChild` is the inverse of `hasParent`, or that `hasParent` is the same as `childOf`)
- XML + XML Schema works on fixed tree-like text documents. URIs can be referred but are not inherent to their data model
- validation against a XML Schema is just a syntactic check, no implicit knowlege can be extracted from the data

XML and XML Schema is not enough (II)

- XML Schema specifies syntactic properties, the semantics is left to the programs that process the information
- this is bad, because semantics may change and evolve making it necessary to rewrite these programs
- XML data integration is difficult (XML Schema does not help)
- a language for writing semantic definition in order to also make them machine processable is needed

RDF introduction

- the Resource Description Framework (RDF) is framework for describing resources on the web and its relationships
- RDF is designed for being machine procesable, not for being displayed to people
- basically it is a graphical data model, with syntax in XML and a formal semantics
- it is a W3C Recommendation since 2004

RDF design goals (I)

- the design of RDF was driven to meet the following goals:
  - having a simple data model easy for applications to process and manipulate, independent of any specific syntax
  - having formal semantics and provable inference provides a basis for reasoning about the meaning, in particular supports rigorously defined notions of entailment
  - using an extensible URI-based vocabulary URI references are used for naming all kinds of things in RDF, literals are the only other kind of data value
  - using an XML-based syntax which can be used to encode the data model for exchange of information among applications

RDF design goals (I)

- the design of RDF was driven to meet the following goals:
  - supporting use of XML schema datatypes assisting the exchange of information between RDF and other XML applications.
  - allowing anyone to make statements about any resource to facilitate operation at the Web, RDF is an open-world framework that allows anyone to make statements about any resource. In general, it is not assumed that complete information about any resource is available and RDF does not prevent anyone from making assertions that are nonsensical or inconsistent with other statements. Designers of applications that use RDF should be aware of this and may design their applications to tolerate incomplete or inconsistent sources of information.

Uses of RDF

- Web metadata providing information about Web resources and the systems that use them
- applications that require open rather than constrained information models
- to do for machine processable information (application data) what the World Wide Web has done for hypertext to allow data to be processed outside the particular environment in which it was created, in a fashion that can work at Internet scale.
- data interoperability among applications combining data from several applications to arrive at new information

Uses of RDF

- automated processing of Web information by software agents the Web is moving from having just human-readable information to being a world-wide network of cooperating processes. RDF provides a world-wide lingua franca for these processes
- concrete examples:
  - describing properties for shopping items, such as availability
  - describing time schedules for web events, organizational process description
  - describing information about web pages (content, author, created and modified date)
  - describing content and rating for web pictures
  - describing content for search engines, content rating and privacy preferences systems
  - describing electronic libraries

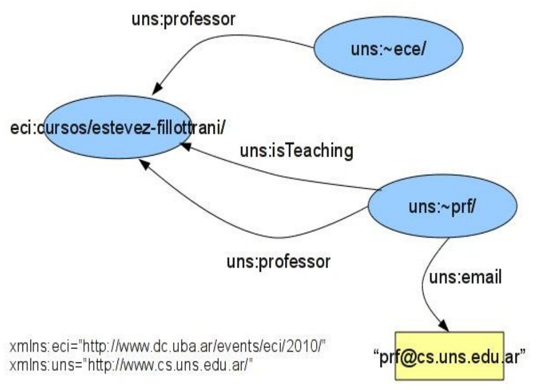
Uses of RDF

- RDF is designed to represent information in a minimally constraining, flexible way
- it can be used in isolated applications, where individually designed formats might be more direct and easily understood, but RDF's generality offers greater value from sharing
- the value of information thus increases as it becomes accessible to more applications across the entire Web

RDF data model

- RDF data model describes directed labelled pseudo-graphs. This means:
  - directed every arc has a direction
  - labelled every arc has a label
  - pseudo-graph there can be more then one arc between the same two nodes
- both arc and start node labels can be URIs; end node labels can be URIs or literals
- URIs denote resources, anything we can talk about
- a RDF database is the just an unordered collection of statements describing these arcs
- a statement is also known as a triple because it's composed of three things: a subject (stard node), a predicate (arc) and an object (end node)

Example of a RDF database


uns:professor
uns:~ece/
eci:cursos/estevez-fillottrani/
uns:isTeaching
uns:~prf/
uns:professor
uns:email
xmlns:eci="http://www.dc.uba.ar/events/eci/2010/"
xmlns:uns="http://www.cs.uns.edu.ar/"
"prf@cs.uns.edu.ar"

Example of a RDF database (cont)

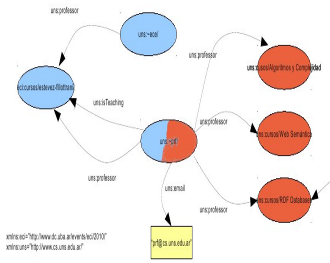
- equivalent set of statements

```
uns:~prf---uns:professor ->eci:cursos/e-f/
uns:~ece---uns:professor-->eci:cursos/e-f/
uns:~prf---uns:isTeaching->eci:cursos/e-f/
uns:~prf---uns:email------>"prf@cs.uns.edu.ar"
```

- note the use of namespaces
- predicates are URIs, so we can describe them in RDF like any other resource

RDF database integration

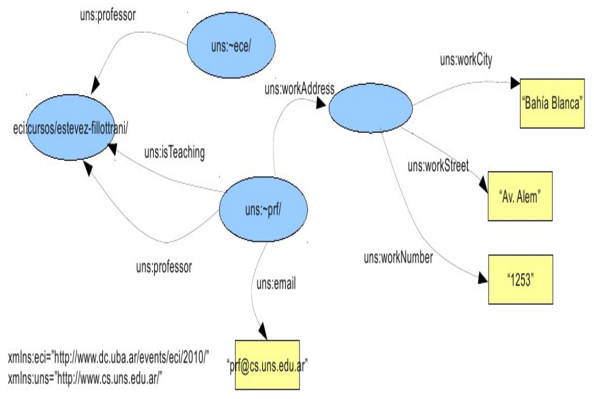
- RDF is designed with a strongly decentralized system in mind, there must be a way to identify those parts of the statements so that they can be reused, either in the same model or in other models
- a fundamental property of the RDF data model is that it is possible to combine any two RDF databases



Blank nodes

- RDF also allows blank nodes as subject and object in triples
- a blank node is a node that is not a URI reference or a literal. It's just a unique node that can be used in one or more RDF statements, but has no intrinsic name (ie no URI)
- when graphs are merged, their blank nodes must be kept distinct if meaning is to be preserved
- blank node identifiers are not part of the RDF abstract syntax, and the representation of triples containing blank nodes is entirely dependent on the particular concrete syntax used

Example of blank node



XML syntax

- so in order to transfer and integrate RDF databases, it is necessary to serialise it. There are several options for this
  - RDF/XML the official W3C serialization. Application-oriented, too verbose.
  - Notation3 (N3) more human-oriented syntax
  - Turtle
  - N-triples

RDF/XML syntax main elements (I)

- `rdf:RDF` is the root element of every RDF/XML document
- it contains a series of `rdf:Description` elements, each one describing one or more triples
- `rdf:Description` elements may be annidated
- `rdf:about` attribute of `rdf:Description` is used to specify the URI of the resource being described
- properties are named with URIs, therefore they can also be described

RDF/XML syntax main elements (II)

- objects may be other `rdf:Description` elements or just text for literals
- the attribute `rdf:resource` can be used to describe objects with URIs
- the attribute `rdf:nodeID` can be used to identify blank nodes in cases when these nodes are referred in several triples
- in summary, one RDF data model has several (too much!) corresponding XML encodings; one XML encoding has one and only one corresponding RDF data model

RDF/XML syntax example

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:uns="http://www.cs.uns.edu.ar/">
 <rdf:Description rdf:about="http://www.cs.uns.edu.ar/~prf">
  <uns:professor rdf:resource="http://www.dc.uba.ar/events/eci/2010/"/>
  <uns:isTeaching rdf:resource="http://www.dc.uba.ar/events/eci/2010/"/>
  <uns:email>prf@cs.uns.edu.ar</uns:email>
  <uns:workAddress>
   <rdf:Description>
    <uns:workCity>Bahía Blanca</uns:workCity>
    <uns:workStreet>Av. Alem</uns:workStreet>
    <uns:workNumber>1253</uns:workNumber>
   </rdf:Description>
  </uns:workAddress>
 </rdf:Description>
 <rdf:Description rdf:about="http://www.cs.uns.edu.ar/~ece">
  <uns:professor rdf:resource="http://www.dc.uba.ar/events/eci/2010/"/>
 </rdf:Description>
</rdf:RDF>
```

N3

```
@prefix uns:<"http://www.cs.uns.edu.ar/">.
@prefix eci:<"http://www.dc.uba.ar/events/eci/2010/">.

uns:~prf
  uns:professor  eci:cursos/estevez-fillottrani;
  uns:isTeaching eci:cursos/estevez-fillottrani;
  uns:email "prf@cs.uns.edu.ar";
  uns:workAddress [ uns:workCity "Bahía Blanca";
                    uns:workStreet "Av. Alem";
                    uns:workNumber "1253" ] .

uns:~ece
   uns:professor  eci:cursos/estevez-fillottrani.
```

Comparison of RDF notations

- RDF/XML is good to use as the output of XSLT when transforming XML datasets into an equivalent RDF representation
- but it is not for writing by hand or when using scripting. N3 is much more adequate for these tasks
- tools like Jena (java) or CWM (python) can do the transformations for you from one syntax to another, since they have the same expressive power
- exercise: describe the graph model of a RDF database with information from this course
- exercise: serialize this database in RDF/XML
- exercise: validate RDF/XML files and generate graph models with the W3C RDF Validation Service `http://www.w3.org/RDF/Validator/`

RDF/XML extended vocabulary

- one of the main problems of RDF is that its RDF/XML official serialization includes several names without a proper semantics
- RDF containers are used to describe group of things
  - the `rdf:Bag` element is “intended” to describe a list of values that does not has to be in a special order. Values may be repeated.
  - the `rdf:Seq` element is “intended” to describe an ordered list of values. Values may be repeated
  - the `rdf:Alt` element is “intended” to describe a list of alternative values (the user can select only one of the values)
- the element `rdf:type` can be used to type nodes to these container types
- RDF also include vocabulary for reification of triples, ie describing metadata of statements. These are the `rdf:ID` attribute and the `rdf:subject`, `rdf:property`, and `rdf:object` properties

Example of a RDF bag

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:cd="http://www.recshop.fake/cd#">
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:artist>
    <rdf:Bag>
      <rdf:li>John</rdf:li>
      <rdf:li>Paul</rdf:li>
      <rdf:li>George</rdf:li>
      <rdf:li>Ringo</rdf:li>
    </rdf:Bag>
  </cd:artist>
</rdf:Description>
</rdf:RDF>
```

Example of a RDF alt

```
 <rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:cd="http://www.recshop.fake/cd#">
<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Beatles">
  <cd:format>
    <rdf:Alt>
      <rdf:li>CD</rdf:li>
      <rdf:li>Record</rdf:li>
      <rdf:li>Tape</rdf:li>
    </rdf:Alt>
  </cd:format>
</rdf:Description>
</rdf:RDF>
```

RDF Schema

- RDF describes resources with properties, and values, identifying them with URIs
- in addition, RDF also need a way to define application-specific classes (sets of resources that have common characteristics) and the properties that apply to all their instances. These must be defined using extensions to RDF.
- RDF Schema is basically a set of RDF statements that define classes and properties. You can think of RDFSchema as metadata for your statements.

RDF Schema vocabulary

- the kind of things you can say with a mix of RDF and RDFSchema vocabularies are:
  - this URI should be considered `rdf:type` a class `rdfs:Class` or a property `rdf:Property`
  - indicate a human readable label `rdfs:label` or comment `rdfs:comment`. These are very useful for visualizing RDF in more presentation-friendly ways and defining metadata
  - this URI is defined by `rdfs:isDefinedBy`. Also metadata about resource's author
  - this class is a subclass of this other `rdfs:subClassOf`
  - this property is subproperty of this other `rdfs:subPropetyOf`
  - this property connects this class of subjects `rdfs:domain` with this class of objects `rdfs:range`

Classes in RDFS

- classes in RDF Schema are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes
- but a resource may belong to several classes, `rdf:type` is just a property
- ie, it is not like a datatype!
- the type information may be very important for applications, it may be used for a categorization of possible nodes
- the name RDF Schema is unfortunate because it does not play to RDF the same role as XML Schema plays to XML: XML Schema is a language to specify the structure of a XML document; RDF Schema is a vocabulary to define semantics of terms
- thus, entailment in RDFS is much more difficult than in RDF.

Example of RDF Schema

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xml:base="http://www.cs.uns.edu.ar/">
<rdf:Description rdf:ID="teachingPersonal">
  <rdf:type rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:ID="professor">
  <rdf:type rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="teachingPersonal"/>
</rdf:Description>
</rdf:RDF>
```

XML vs. RDF

- could we use plain XML instead of RDF? Yes
- can XML encode a graph? Use URIs within one single document
- can XML support higher-order statements? Establish naming and linking convention
- oops, you've just re-wrote RDF specification
- knowledge from an XML document is completely explicitly specified; in a RDF database there is implicit knowledge
- it is necessary entailment procedures for querying RDF databases; a simple linear search algorithm is enough for a XML document

RDF in real life (I)

- Sigma data aggregation application from DERI in National University of Ireland, Galway(NUIG)
- Creative Commons uses RDF to embed license information in web pages and mp3 files
- FOAF (Friend of a Friend) designed to describe people, their interests and interconnections
- DOAC (Description of a Career) supplements FOAF to allow the sharing of résumé information
- MusicBrainz publishes information about Music Albums

RDF in real life (II)

- NEPOMUK an open-source software specification for a Social Semantic desktop uses RDF as a storage format for collected metadata. NEPOMUK is mostly known because of its integration into the KDE4 desktop environment
- RDF Site Summary (RSS 1.0) one of several RSS"languages for publishing information about updates made to a web page; it is often used for disseminating news article summaries and sharing weblog content
- Simple Knowledge Organization System (SKOS) an KR representation intended to support vocabulary/thesaurus applications

Dublin Core Metadata Initiative

- the Dublin Core Metadata Initiative (DCMI) has created some predefined properties for describing documents
- the first Dublin Core properties were defined at the Metadata Workshop in Dublin, Ohio in 1995 and is currently maintained by the Dublin Core Metadata Initiative
- it consists of predefined properties for describing documents like

Property	Definition
Contributo	an entity responsible for making contributions to the content of the resource
Coverage	the extent or scope of the content of the resource
Creator	an entity primarily responsible for making the content of the resource
Format	the physical or digital manifestation of the resource

Dublin Core Metadata Initiative

- it consists of predefined properties for describing documents like

Property	Definition
Date	a date of an event in the lifecycle of the resource
Description	an account of the content of the resource
Identifier	an unambiguous reference to the resource within a given context
Language	a language of the intellectual content of the resource
Publisher	an entity responsible for making the resource available
Relation	a reference to a related resource
Rights	information about rights held in and over the resource

Dublin Core Metadata Initiative

- it consists of predefined properties for describing documents like

Property	Definition
Source	a reference to a resource from which the present resource is derived
Subject	a topic of the content of the resource
Title	a name given to the resource
Type	the nature or genre of the content of the resource

- RDF is the language for representing Dublin Core term definition

DC example in RDF

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc:    <http://purl.org/dc/elements/1.1/> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix dcp:   <http://dublincore.org/usage/documents/principles#> .
@prefix dch:   <http://dublincore.org/usage/terms/history#> .
dc:creator
  rdf:type       rdf:Property ;
  rdfs:label     "Creator"@en-US ;
  rdfs:comment   "An entity primarily responsible for
                 making the content of the resource."@en-US ;
  dc:description "Examples of a Creator include a person,
                 an organisation, or a service. "@en-US ;
  rdfs:isDefinedBy  <http://purl.org/dc/elements/1.1/> ;
  dct:issued     "1999-07-02" ;
  dct:modified   "2002-10-04" ;
  dc:type        dcp:element ;
  dct:hasVersion dch:creator-004 .
```

SPARQL introduction

- SPARQL is a RDF Query language with a SQL-like syntax
- it's a W3C recommendation since January 2008
- queries consist of the following clauses
  - `PREFIX` prefix mechanism for namespaces
  - `SELECT` identifies the variables to be returned in the query answer
  - `FROM` name of the graph/s (URI/s of RDF database/s) to be queried
  - `WHERE` query pattern as a list of triple patterns in Turtle syntax
- query results are those triples entailed by the merge of the designated graphs

SPARQL example (I)

- data

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

- query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

SPARQL example (II)

- query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

- result

```
     name                      mbox
"Johnny Lee Outlaw"  <mailto:jlow@example.com>
"Peter Goodguy"   <mailto:peter@example.org>
```

Jena introduction

- Jena is a Java framework for building Semantic Web applications
- it provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine
- Jena is open source and grown out of work with the HP Labs Semantic Web Programme
- the Jena Framework includes:
  - a RDF API
  - reading and writing RDF in RDF/XML, N3 and N-Triples
  - an OWL API
  - in-memory and persistent storage
  - SPARQL query engine

Jena example

```
// create a model
 Model model=new ModelMem();
 Resource subject=model.createResource("URI_of_Subject")
 // 'in' refers to the input file
 model.read(new InputStreamReader(in));
 StmtIterator iter=model.listStatements(subject,null,null);
 while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty();
    o = st.getObject();
    do_something(p,o);
 }
```

From vocabularies to ontologies

- data integration needs agreements
  - on terms, ie, “translator”, “author”
  - on categories used “Person”, “literature”
  - on relationships among those “an author is also a Person”, “historical fiction is a narrower term than fiction”
  - in a way that new relationships can be deduced
- there is a need for “languages” to define such vocabularies and to assign clear semantics on how new relationships can be deduced: these are called ontologies
- formally, there are two definitions for ontology
  - formal explicit specification of a shared conceptualization of a domain
  - a specification consisting of Classes, Relations between classes, Individuals, and Axioms.

Ontologies and the Semantic Web

- ontologies form the backbone of the Semantic Web, define the basic vocabulary for the annotations, enable reasoning with background knowledge
- based on formal languages, interweave meaning for humans and machines
- based on limitations of RDFS, three technologies have emerged
  - to re-use thesauri, glossaries, etc: SKOS
  - to define more complex vocabularies with a strong logical underpinning: OWL
  - generic framework to define rules on terms and data: RIF

RDFS is not enough!

- indeed RDFS is an ontology language
- there is typing, subtyping, properties can be put in a hierarchy datatypes can be defined
- RDFS is enough for many vocabularies, but not for all
- expressive limitations of RDF(S)
  - only binary relations
  - characteristics of Properties (e.g. inverse, transitive, symmetric)
  - local range restrictions (e.g. for Class Person, the property hasName has range xsd:string)
  - complex concept descriptions (e.g. Person is defined by Man and Woman)
  - cardinality restrictions (e.g. a Person may have at most 1 name)
  - disjointness axioms (e.g. nobody can be both a Man and a Woman)

Ontology components

- classes : grouping of individuals with common properties
  - intentional classes e.g. Person, Car, University
  - extensional classes e.g Color$= \{blue, red, yellow, green\}$
- relations : connections between individuals, may be attached to classes e.g. hasName, hasChild, hasColor, owns
- individuals : objects in the domain, may be instances of classes
- axioms : additional statements about the domain, specified in logical language e.g. “hasName has one value”

Motivation

- complex applications may want more possibilities:
  - characterization of properties
  - identification of objects with different URI-s
  - disjointness or equivalence of classes
  - construct classes, not only name them
  - more complex classification schemes
  - can a program reason about some terms? eg, “if Person resources «A» and «B» have the same «foaf:email» property, then «A» and «B» are identical”
- languages should be a compromise between rich semantics for meaningful applications and feasibility, implementability

History

- OWL is an extra layer, a bit like RDF Schemas
- own namespace, own terms
- it relies on RDF Schemas
- there is a 2004 version of OWL (OWL 1) and there is an update (“OWL 2”) published in 2009
- OWL is a large set of additional terms

OWL design goals

- shareable
- changing over time
- interoperability
- inconsistency detection
- balancing expressivity and complexity
- ease of use
- compatible with existing standards
- internationalization

OWL ontologies

- ontologies are object on the Web with their own meta-data, versioning, etc...
- ontologies are extendable
- with XML syntax

OWL vocabulary: term equivalence

- for classes:
  - `owl:equivalentClass` two classes have the same individuals
  - `owl:disjointWith` no individuals in common
- for properties:
  - `owl:equivalentProperty`
  - `owl:propertyDisjointWith`
- for individuals:
  - `owl:sameAs` two URIs refer to the same concept (“individual”)
  - `owl:differentFrom` negation of previous one

OWL vocabulary: property characterization

- in OWL, one can characterize the behavior of properties (symmetric, transitive, functional, inverse functional, etc)
  - example: `foaf:email` may be defined as inverse functional ie, two different subjects cannot have identical object
- OWL also separates data and object properties
  - “datatype property” means that its range are typed literals
- in OWL 2 properties may also be characterized as reflexive or irreflexive
- there may be an inverse relationship among properties
  - example:

```
<somebook> ex:author <somebody>.
ex:author owl:inverseOf ex:authorOf.
```

OWL vocabulary: property chains

- properties, when applied one after the other, may be subsumed by yet another one:

  if a person «P» was born in city «A» and «A» is in country «B» then «P» was born in country «B»

  more formally:

```
ex:born_in_country owl:propertyChainAxiom
          (ex:born_in_city ex:city_in_country)
```

- more than two constituents can be used
- there are some restrictions to avoid “circular” specifications

OWL vocabulary: classes

- in RDFS, you can subclass existing classes. That's all
- in OWL, you can construct classes from existing ones:
  - enumerate its content
  - through intersection, union, complement, etc
- OWL makes a stronger conceptual distinction between classes and individuals

OWL vocabulary: classes

- there is a separate term for owl:Class, to make the difference
- individuals are separated into a special class called owl:Thing Eg, a precise classification would be:

```
ex:Person rdf:type owl:Class.
uns:~prf  rdf:type owl:Thing;
          rdf:type owl:Person.
```

OWL vocabulary: cardinality constraints

- in a cardinality restriction, the number of relations with that property is restricted
- "a book being on offer" could be characterized as having at least one price property (ie, the price of the book has been established)

```
ex:Book_on_sale rdfs:subClassOf [
     rdf:type             owl:Restriction;
     owl:onProperty       p:price;
     owl:minCardinality "1"^^xsd:integer  ].
```

OWL vocabulary: cardinality constraints

- there can be also qualified cardinality restrictions, combining cardinality and the "all value" restriction
- "there should be exactly two listed price tags with currency value"

```
ex:Listed_Price rdf:type owl:Class;
   rdfs:subClassOf [
     rdf:type     owl:Restriction;
                  owl:onProperty                p:currency;
                  owl:onClass  ex:Currency;
                  owl:qualifiedCardinality   "2"^^xsd:integer    ].
```

OWL evaluation

- the OWL features listed so far are already fairly powerful
- many inferred relationship can be found using a traditional rule engine
  - logical expressions (and, or, not)
  - (in)equality
  - local properties
  - required/optional properties
  - required values
  - enumerated classes
  - symmetry, inverse

OWL evaluation

- the combination of class constructions with various restrictions is extremely powerful
- however, a full inference procedure is hard and not implementable with simple rule engines, for example

OWL dialects

- OWL Lite
  - Classification hierarchy
  - Simple constraints
- OWL DL
  - Maximal expressiveness, while maintaining tractability
  - Standard formalization in a DL
- OWL Full
  - Very high expressiveness, losing tractability
  - All syntactic freedom of RDF (self-modifying)

OWL dialects comparison

OWL Lite

- (sub)classes, individuals
- (sub)properties, domain, range
- conjunction
- (in)equality
- (unqualified) cardinality 0/1
- datatypes
- inverse, transitive, symmetric properties
- someValuesFrom
- allValuesFrom

OWL DL

- Negation
- Disjunction
- (unqualified) Full cardinality
- Enumerated classes
- hasValue

OWL Full

- Meta-classes
- Modify language

OWL DL usage

- very large ontologies can be developed that require precise procedures eg, in the medical domain, biological research, energy industry, financial services (eg, XBRL), etc
- the number of classes and properties described this way can go up to the many thousands
- OWL DL has become a language of choice to define and manage formal ontologies in general even if their usage is not necessarily on the Web

OWL notations

- OWL/RDF official exchange syntax, hard for humans and also RDF parsers are hard to write!
- OWL/XML not the RDF syntax, still hard for humans, but more XML than RDF tools available
- abstract syntax not defined for OWL Full, more human readable

```
Class( firstYearCourse partial
  restriction (isTaughtBy allValuesFrom ( Professor )))
Class(mathCourse partial
  restriction (isTaughtBy hasValue (949352)))
Class(academicStaffMember partial
  restriction (teaches someValuesFrom (undergraduateCourse)))
Class(course partial  restriction (isTaughtBy
     minCardinality (1)))
Class(department partial restriction (hasMember
     minCardinality(10))
 restriction (hasMember maxCardinality(30)))
```