GOTO 2012

Linked Data as a new model for Application Integration

Martin Nally, VP & IBM Fellow

CTO, IBM Rational

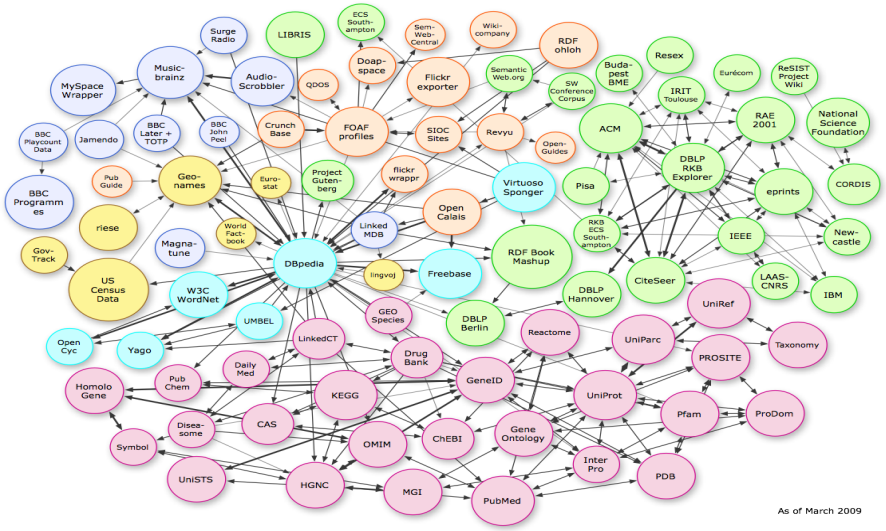
What is Linked Data?

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.

HTTP web of DATA “resources”, instead of HTML pages

Linked Data is not new

Publish Data on the Internet



- Use it to provide new value

But our usage of Linked Data – Application Integration - is a bit different

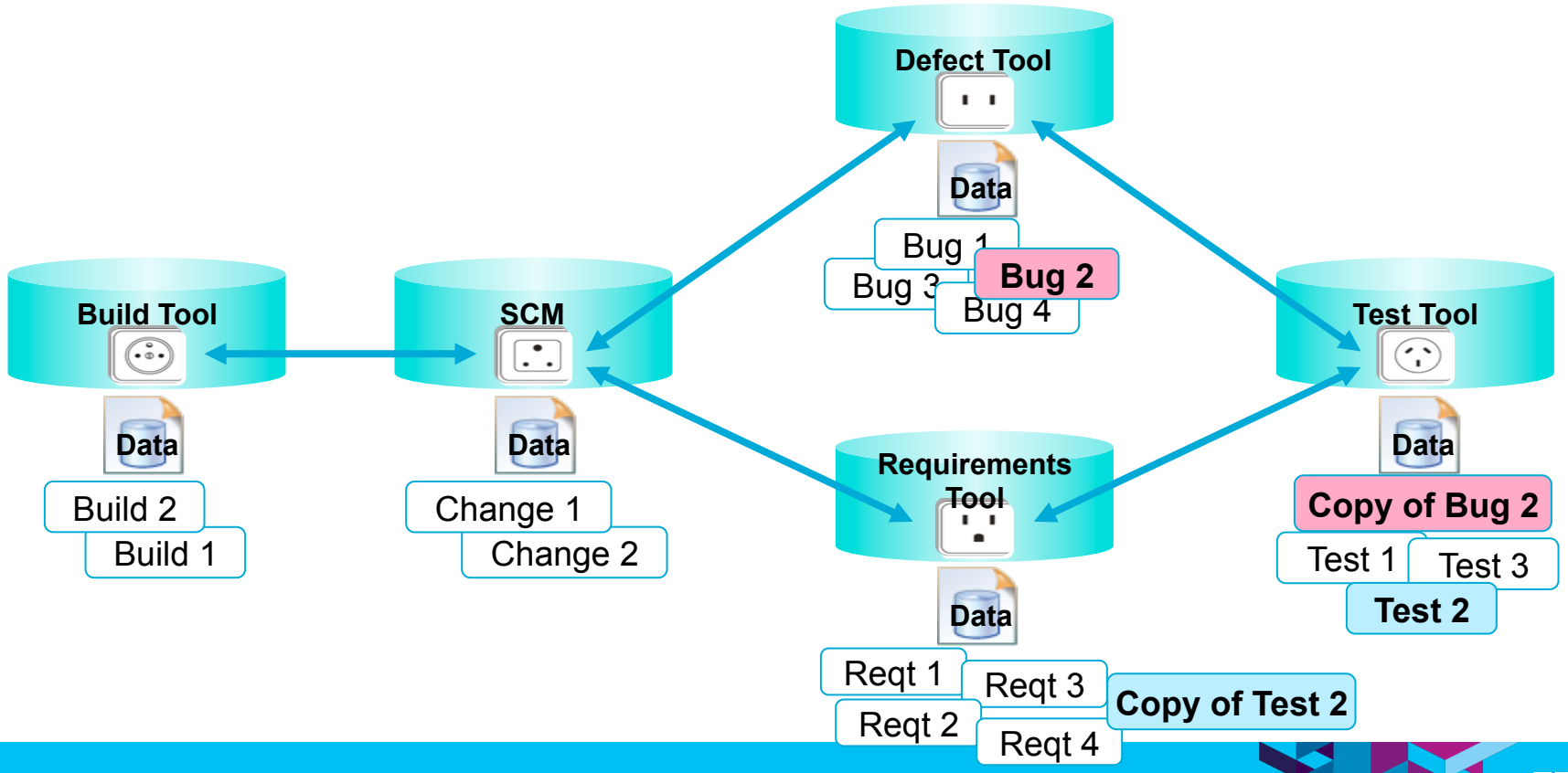
Why do we need a new approach to integration?

Top 3 reasons Application Lifecycle Management (ALM) fails to deliver promise

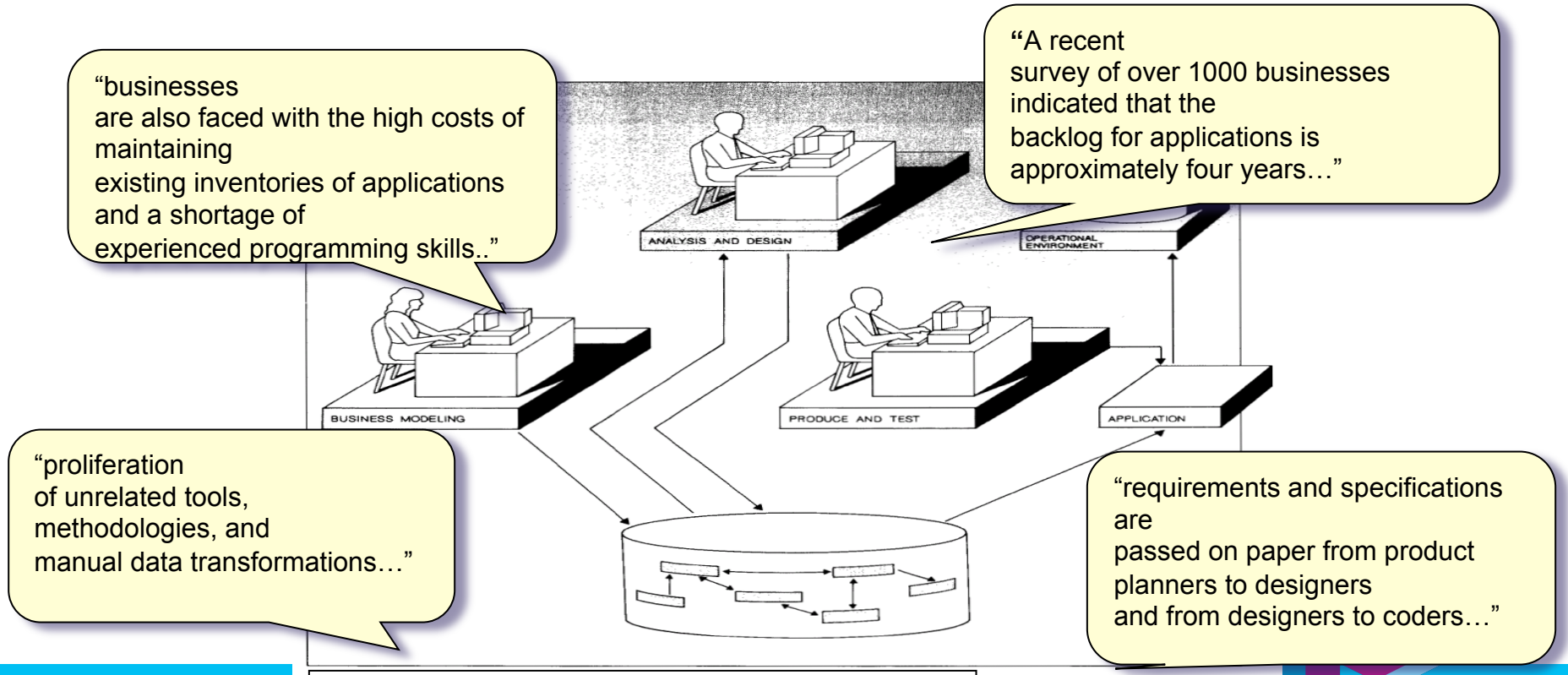
- Distracted by day-to-day delivery pressures – 78%
- Tools don't integrate properly – 62%
- Lack the necessary internal expertise – 56%

Source: Forrester study commissioned by Wipro, 2008

Tool integration today



What did we say about this 20 years ago?



Source: Presentation on IBM's AD/Cycle, circa **1990!**

What is the state-of-the-art today?

Most other vendors still trying to build AD/Cycle

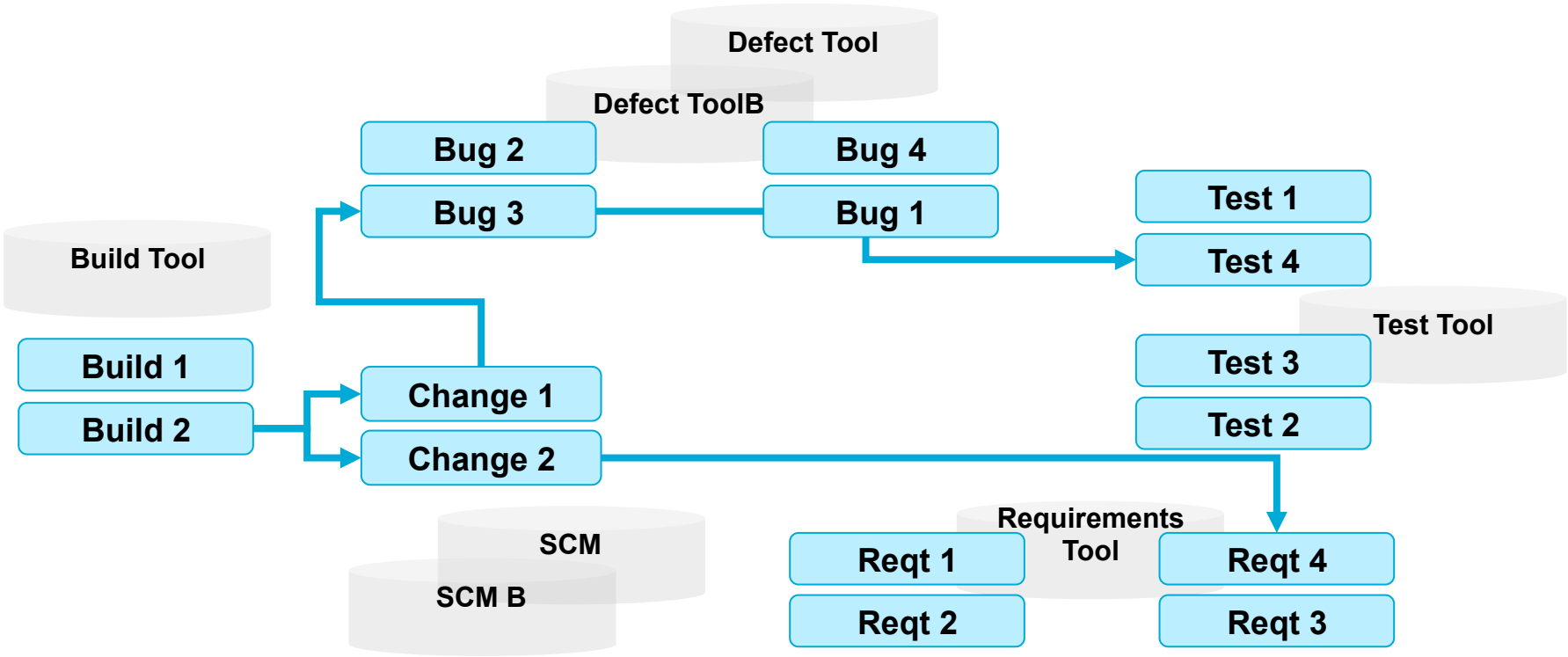
Requires all tools to integrate around centralized repository

- Data import (duplication) for foreign tools

Works as well as other centrally-planned economies have worked

- Do your company's needs match a fixed, pre-planned solution, or is an open, integrated economy a better model?

A new approach: Linked [Lifecycle] Data



What is RDF?

A "universal" data representation for the web

- Relational, IMS, COBOL, XML, object, … data can all be expressed in RDF

A very simple model and syntax for representing data on the world wide web

- RDF is like property, value pairs
- RDF adds "subject" – what is it the property of – so triples, not pairs
- RDF properties are themselves resources with URLs.

That's about it – most of the rest is hype and pretention, or detail

- RDF also can describe containers and collections
- RDF has the notion of type, but it's not similar to OO type, it's like type in the natural world.
- There is a language for querying over RDF, called SPARQL. (SPARQL adds graphs, so quadruples, not triples)
- You can write down RDF data in XML, as a twisted experiment of no value, but there are much nicer, more natural formats

RDF example - http://example.com/r1

@prefix oslc-rm: <http://open-services.net/ns/rm#>.

@prefix dc: <http://purl.org/dc/terms/>.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# >

<http://example.com/r1> rdf:type oslc-rm:Requirement.

< http://example.com/r1 > dc:title "Requirement 1".

< http://example.com/r1 > dc:description "The system should respond within 1 second".

RDF example - http://example.com/tc1

@prefix oslc-qm: <http://open-services.net/ns/qm#>.

@prefix dc: <http://purl.org/dc/terms/>.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# >

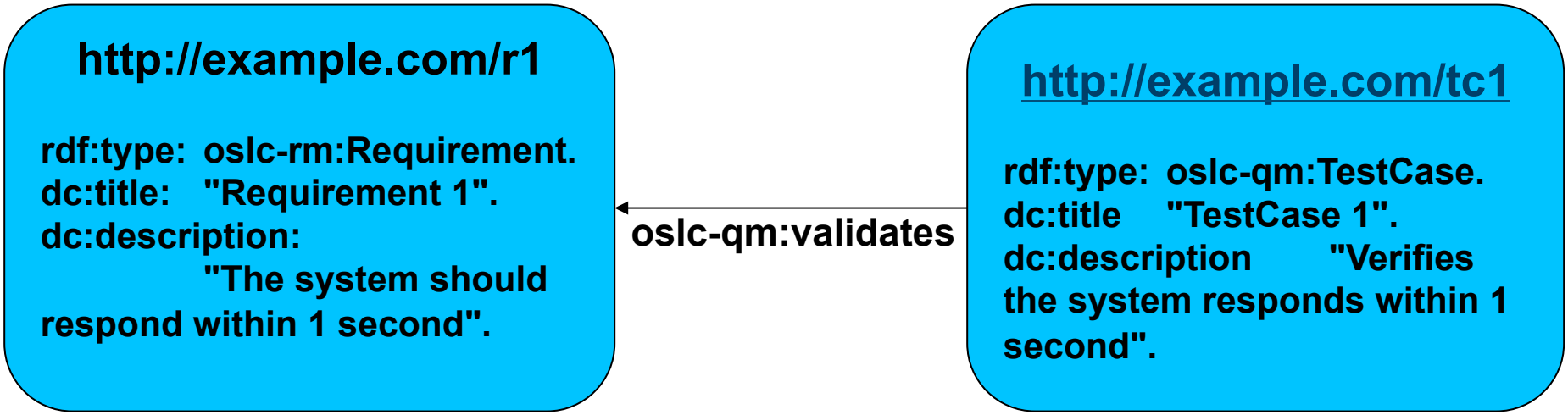
<http://example.com/tc1> rdf:type oslc-qm:TestCase.

<http://example.com/tc1> dc:title "TestCase 1".

<http://example.com/tc1> dc:description "Verifies the system responds within 1 second".

<http://example.com/tc1> oslc-qm:validates <http://example.com/r1>.

RDF – example http://example.com/r1



RDF example2 - http://example.com/notr1ortc1

```
@prefix oslc-rm: <http://open-services.net/ns/rm#>.
@prefix oslc-qm: <http://open-services.net/ns/qm#>.
@prefix dc: <http://purl.org/dc/terms/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# >
<http://example.com/r1>          rdf:type      oslc-rm:Requirement.
< http://example.com/r1 >        dc:title                "Requirement 1".
< http://example.com/r1 >        dc:description          "The system should respond within 1s".
<http://example.com/tc1>         rdf:type                oslc-qm:TestCase.
<http://example.com/tc1>         dc:title                "TestCase 1".
<http://example.com/tc1>         dc:description          "Verifies the system responds within 1s".
<http://example.com/tc1>         oslc-qm:validates       <http://example.com/r1>.
```

http://example.com/r1 and http://example.com/tc1 are completely empty!

RDF – example http://example.com/notr1ortc1

http://example.com/notr1ortc1

```
@prefix oslc-rm: <http://open-services.net/ns/rm#>.
@prefix oslc-qm: <http://open-services.net/ns/qm#>.
@prefix dc: <http://purl.org/dc/terms/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# >
<http://example.com/r1>     rdf:type          oslc-rm:Requirement.
< http://example.com/r1 >   dc:title          "Requirement 1".
< http://example.com/r1 >   dc:description    "The system should respond within 1s".
<http://example.com/tc1>    rdf:type          oslc-qm:TestCase.
<http://example.com/tc1>    dc:title          "TestCase 1".
<http://example.com/tc1>    dc:description    "Verifies the system responds within 1s".
<http://example.com/tc1>    oslc-qm:validates <http://example.com/r1>.
```

Understanding RDFS and OWL

- RDFS introduces classes and properties
- They are not the same as the classes and properties you know from OO
- Everything about RDFS ad OWL is backwards from what you know
  - In OO, classes come first, then you have instances. In RDF, instances come first and then you can (optionally) have classes
  - In OO, instances have 1 type. In RDF, instances can have lots of types, or none at all
  - In OO, classes constrain instances (e.g. properties, multiplicity). In RDF classes can only infer new information, cannot verify existing info
- The challenge with RDFS/OWL is unlearning what you know.

Linked Data – a potential major transformation

There have only been two major model shifts in my 30+ year career.

- First was shift to client server from mainframe. This is the second.

Adopting this simple model turns everything on its head.

- The HTTP resources are central, your application a minor detail
- The HTTP URLs are permanent reality, the data in the database a detail
- Closed, fixed in scope -> open, extensible scope
- Fixed in time -> everything evolves over time
- Don’t import data – address it where it is

But there is a problem

Existing experience with Linked Data is read-only.
For writing, many basic questions need to be answered

- It seems obvious that you POST to create, but what do you POST to?
- How do I find the things that already exist?
- What media types should I use?
- What about resources that cannot be represented in RDF?
- What primitive types should I use?
- What standard vocabularies should I use?
- Links - What if links have properties? Back-links?
- How do you describe “shapes” or “schemas”?
- PUT or Patch for update?

Solution

Good news

- There are simple answers to most of these questions using technologies that already exist – little or no invention required

Bad News

- The answers are often not obvious and are definitely not well-known

Action

- New W3C workgroup, aiming at new W3C recommendation
  - “Linked Data Platform”
  - http://www.w3.org/2012/ldp/wiki/Main_Page

Basic Profile Resources and Containers

- Observation - 90% of all applications are just records and lists
  - Lists answer the questions “where do I POST to and how do I find things that already exist?”
- Conclusion – if we describe how you do records and lists with Linked Data, we will have enabled 99% of applications in this style
- IBM wrote a specification for LD records and lists and offered it to the W3C
  - Being used as the basis for the LDP spec

Atom Publishing - Lists for XML programmers

```
POST /collection/1 HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn
Slug: First Post
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <title>Atom-Powered Robots Run Amok</title>
  <id>urn:uuid:1225c695-80da344efa6a</id>
  <updated>2003-12-13T18:30:02Z</updated>
  <author><name>John Doe</name></author>
  <content>Some text.</content>
</entry>
```

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
<title>Example Feed</title>
<link href="http://example.org/"/>
<updated>2003-12-13T18:30:02Z</updated>
<author><name>John Doe</name></author>
<id>urn:uuid:60a76c80-0003939e0af6</id>
  <entry xmlns="http://www.w3.org/2005/Atom">
    <title>Atom-Powered Robots Run Amok</title>
    <id>urn:uuid:1225c695-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <author><name>John Doe</name></author>
    <content>Some text.</content>
  </entry>
</feed>
```

APP Accommodating RDF? First try

```
POST /collection/1 HTTP/1.1
  Host: example.org
  Content-Type: application/atom+xml;type=entry
  Content-Length: nnn
  Slug: First Post

  <?xml version="1.0"?>
  <entry xmlns="http://www.w3.org/2005/Atom">
   <title>Atom-Powered Robots Run Amok</title>
   <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
   <updated>2003-12-13T18:30:02Z</updated>
   <author><name>John Doe</name></author>
   <content type="application/rdf+xml">
     <-- RDF/XML content here -->
   </content>
  </entry>
```

APP Accommodating RDF. Second try.

```
@prefix atom: <http://www.w3.org/2005/Atom/>.
  <http://example.org/myData/collection1>
     a atom:Feed;
     atom:title "Example feed";
     atom:updated "2003-12-13T18:30:02Z";
     atom:author [atom:name "John Doe"];
     atom:id <urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6>;
     atom:entries
        ([a atom:Entry;
           atom:title "Atom-Powered Robots Run Amok";
           atom:alternate <http://example.org/2003/12/13/atom03>;
           atom:id <urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a>;
           atom:updated "2003-12-13T18:30:02Z";
           atom:author [atom:name "John Doe"];
           atom:content [ #more RDF here for the triples that define the content]
        ]
        [#another Atom entry]).
```

This simply is not how you design with RDF

- Atom “entries” are just getting around the lack of a “subject” in RDF
- Custom media types are not a good idea
- Atom title, author, updated are redundant with other RDF vocabularies
- uuid not needed/used in Linked Data (use http urls)
- Atom “collection” redundant with other RDF concepts (and actually even those concepts are not needed anyway for this problem)

OK, so how do you design with RDF?

How do you express a collection with RDF?

- You could use rdf:Seq or rdf:List (not query-friendly)
- But RDF predicates automatically define collections

<subjectURL>
<predicateURL>
<firstElementURL>,
<secondElementURL>,
… ,
<lastElementURL>.

APP equivalent for RDF with no new concepts!

- POST to a “collection resource” to create resource and add a new triple
- Delete and element to delete resource and remove triple
- GET to see the existing triples/resources

```
#representation of collection resource
<collectionURL>
<subjectURL>
  <predicateURL>
    <firstElementURL>,
    <secondElementURL>,
    ... ,
    <lastElementURL>.
```

Basic writable collection in RDF – APP example

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
<http://example.org/collection1>
    rdfs:member <http://example.org/1225c695-80da344efa6a>;
    rdfs:member <http://example.org/1225c695-80da344efa6b>.
```

Use whatever predicate you like –
http://www.w3.org/2000/01/rdf-schema#member is just an example

But APP does much more

- With APP, data for the entries was in the feed – I did not have to GET each entry to see their data, I can just GET the feed.
- You want data about the entries – RDF lets you put it there!
  - No invention required

Writable RDF collection with “entry data”

```
# representation of <collectionURL>
<subjectURL>
   <predicateURL>
   <firstElement>,
   <secondElement>,
   ... ,
   <lastElement>.
# triples about firstElement begin
<firstElement>
   <predicate1> value1;
   <predicate2> value2;
   #more triples
   <predicateN> valueN.
# triples for other elements
<lastElement>
   <predicate1> value1l;
   <predicate2> value2l;
   #more triples
   <predicateN> valueNl.
```

Minor problem

- Which triples define the collection?
  - Subject URL may not be collection URL
  - “collection predicate” may not be rdfs:member
- Reluctantly, we do finally have to invent something new

```
@prefix bp: <http://open-services.net/ns/basicProfile#>.
<http://example.org/netWorth/nw1/assetContainer>
  a bp:Container;
  bp:membershipSubject <http://example.org/netWorth/nw1>;
  bp:membershipPredicate <http://example.org/vocab#asset>.
```

But APP has more

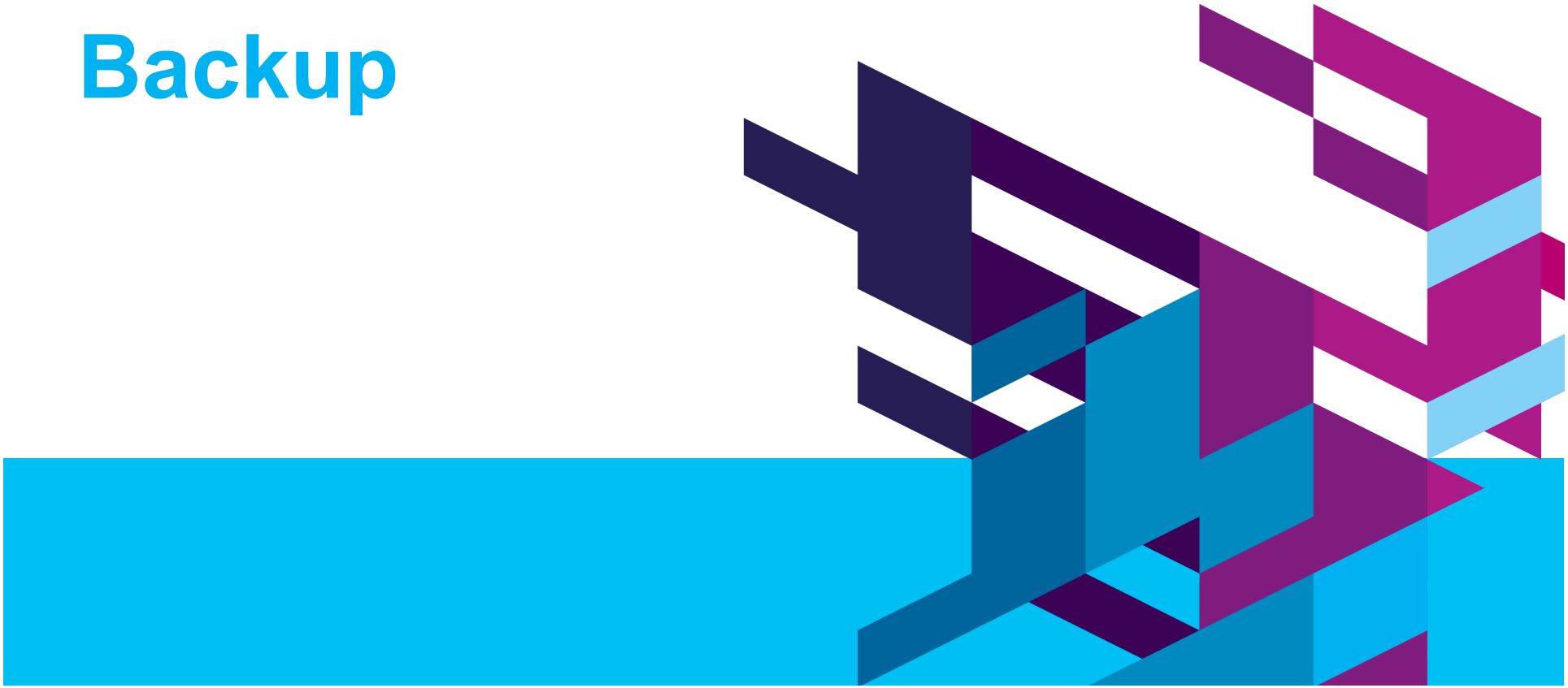
- Read the LDP spec if you want to see how pagination and ordering are done
- As before RDF already knows how to do almost all of it already – someone just has to standardize the pattern

Basic Profile Resources

- That is the “list” part, what about the “record” part
- As before, mostly just a few “conventions” or “patterns of use” of existing technologies - not much to say
- Perhaps one item is worth discussion - update

PUT, PATCH, documents and data

Backup



An example showing why subject is important -

http://vh1.example.com/testcases/defects?oslc.where=oslc_cm:inprogress="true"

@prefix oslc-qm: <http://open-services.net/ns/qm#>.

@prefix determs: <http://purl.org/dc/terms/>.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns# >

<> oslc:nextPage <http://vh1.example.com/testcases/defects/..../page2**>.**

<http://vh1.example.com/defects**>**

rdf:member <http://vh1.example.com/defects/00001**>;**

rdf:member <http://vh1.example.com/defects/00002**>;**

rdf:member <http://vh1.example.com/defects/00003**>.**

Another example showing use of subject - http://members.cox.net/martin_nally

```
@prefix dbpp: <http://dbpedia.org/property/>.
@prefix dbpr: <http://dbpedia.org/resource/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix determs: <http://purl.org/dc/terms/>.

<> dcterms:description “a web site for Martin Nally”.

<http://martin-nally.name>
        dbpp:birthname "Martin Paul Nally";
        dbpp:birthDate "1957-01-05"^^xsd:date;
        dbpp:birthPlace dbpr:Scotland;
        foaf:depiction <http://members.cox.net/martin_nally/Martin_Nally.6.email.jpg>.
```

Open Services for Lifecycle Collaboration

Specifications for linked lifecycle data

An open community of individuals from industry, commercial tools vendors, systems integrators, open source projects, and academia.

Focusing on sharing of lifecycle data (requirements, test cases, change requests) between tools and across the lifecycle.

Taking a technology-neutral approach based on Internet standards and protocols.

Operating at open-services.net

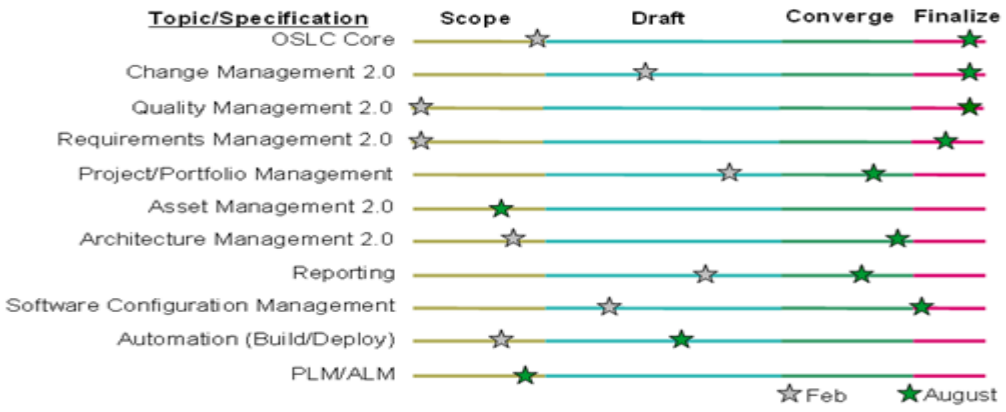
OSLC Community

Eleven workgroups operating at open-services.net

- Intensive focus in 2010 on Core and CLM related specs (CM, RM, QM, Arch Mgmt, SCM)
- PLM/ALM workgroup defining cross-cutting scenarios and driving a systems perspective

Continuing to grow

- 345+ registered community members (up from 70 people at RSC 2009)
- Individuals from 34+ different companies have participated in OSLC workgroups (up from 5 companies at RSC 2009)



Accenture
APG
Black Duck
Boeing
BSD Group
Citigroup
EADS
Emphasys Group
Empulsys
Ericsson
Fokus Fraunhofer
Galorath
General Motors
Health Care Services Corp
IBM
Institut TELECOM
Integrate Systems
Lender Processing Services
Northrop Grumman
Oracle
QSM
Rally Software
Ravenflow
Shell
Siemens
Sogeti
SourceGear/Teamprise
State Street
Tasktop (Eclipse Mylyn)
Thales
Tieto
TOPIC Embedded Systems
UrbanCode
WebLayers

OSLC Core Spec

Applies to all resources in an OSLC system.

Tries to answer some simple questions on how to use linked data

- What URLs can I POST to create new resources?
  - What properties could/should I set when POSTing to these URLs?
- How do I query the resources already POSTed at an URL?
  - What properties might be available to query on a set of resources?
- How is pagination of large representations handled?
- How can I delegate to the UI of another service, instead of dealing with its data?
- Best practices for expressing hyper-links between resources (e.g. link properties)
- Partial Update (there is a reason that SQL has no equivalent of PUT, only PATCH)

A bit like a superset of APP, except …

- Linked data compatible
- Generic - doesn't require you to model your domain as a blog (feed, entry)
- Simpler, Solves more problems

Other OSLC specs

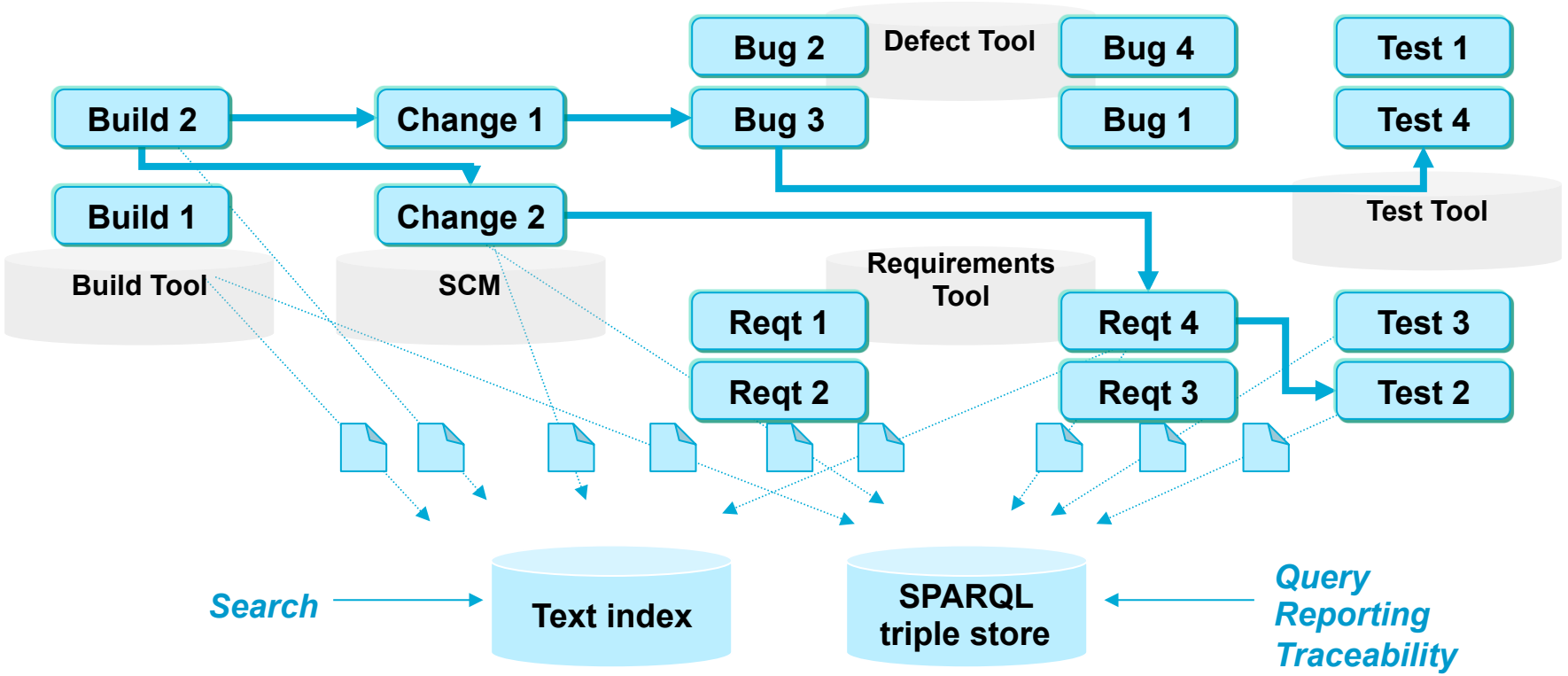
Adhere to Core spec and add domain-specific vocabularies

- Change Management
- Requirements
- Assets
- Tests
- Estimation
- Source Code Management/ versioning
- Reporting
- Architecture
- Project/portfolio
- Automation (e.g. build)

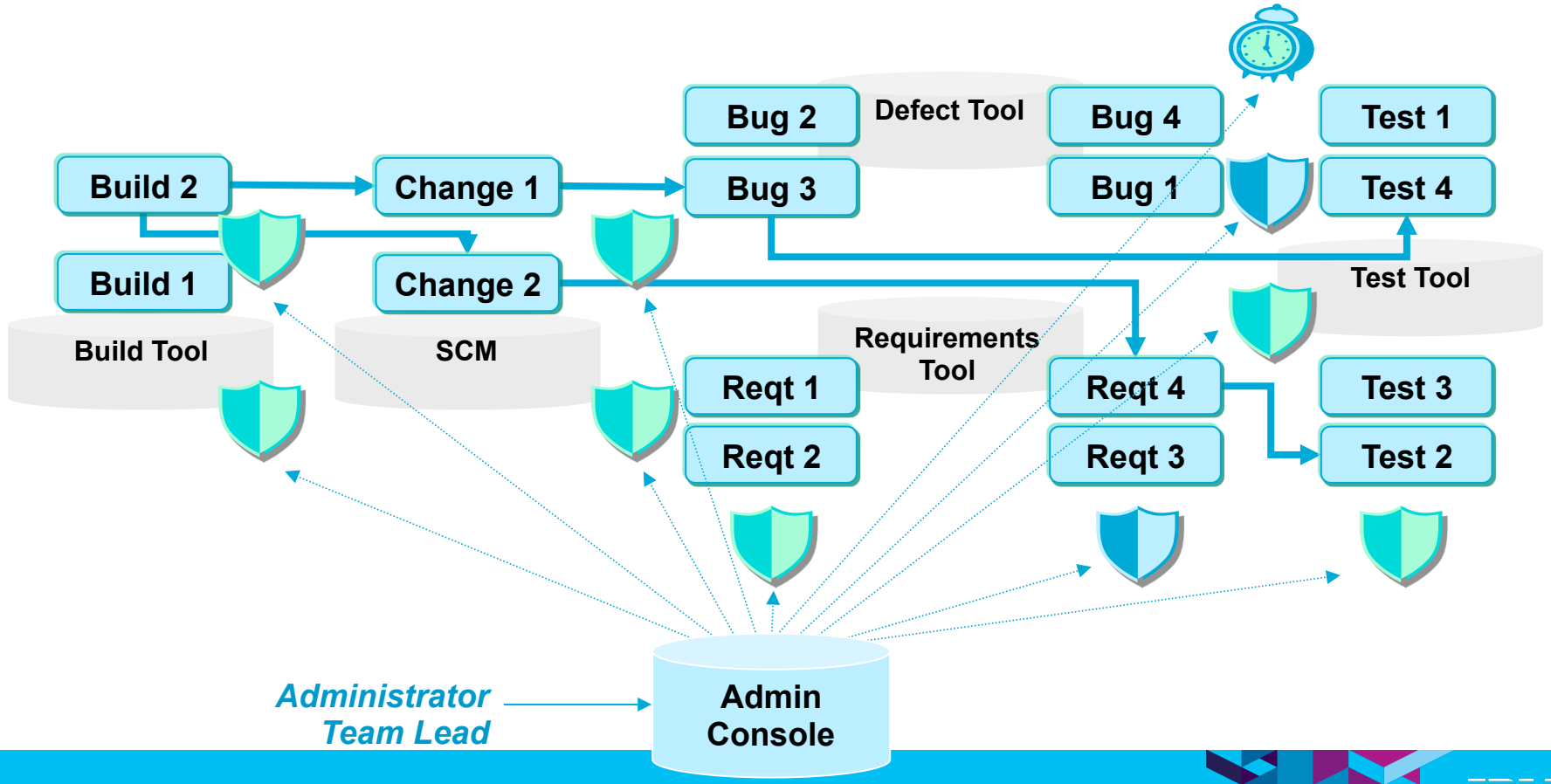
Ontologies (odious, pretentious word)

- Need to agree on common terms like “name”, “type”, “title”, “identifier”
  - Want to query across all resources, not just within types
  - We like Dublin Core. Maybe rdfs (for label). Maybe foaf for Person.
- Need to agree on some domain-specific terms
  - Don’t try to define all the properties of a resource like defect
    - Every team/organization wants different ones
  - Focus on those properties that are important for integration scenarios
    - E.g. Is a defect closed?
    - E.g. What requirement does this test-case test?

Finding and analyzing data



Defining process rules



friends don't let friends do …

XML

- OSLC core spec has some "features" to keep the XML zombies happy ☹
  - RDF/XML mandated
  - forcing "blank nodes" to ease XML parsing ("Local Resource") Web Services

ATOM Publishing Protocol

Data formats or interfaces specified in programming language technologies

- Data specified with object-oriented concepts (classes, instances)

Think conceptually, don't think like a programmer

"a list of bugs", "the first page of a list of bugs" and "bugs whose id is 8" are independent resources – not one resource with "arguments".

- http://example.com/bugs
- http://example.com/bugs?oslc.where=dcterms:identifier=%228%22
- http://example.com/bugs?oslc.paging=true

"oslc:pagination=true" is also "thinking like a programmer". Better would be

- http://example.com/bugs?oslc.firstPage

Most of the current web is read-only

Most content created “conventionally” and then published

- Blogs, tweets, wikis are exceptions
- APP is a protocol for blogs (?)
- Doing “authoring on the web” for a new domain requires learning

Cool URLs last forever

Don’t assume you can “move” data

- Use virtual host names, not ones ties to machines

Don’t put any “meaning” into URLs

- It will change

Security

- **Web authentication protocols are embryonic – e.g. OAuth**
  - Google – everything is public
  - Enterprise search – typically everyone in enterprise can see

What is scope of "user"?

What is language for ACLs?

Miscellanea

Don’t write back-links (they will get out of synch)

Don’t assume closed schema

- Ideally, let others add properties to existing types

Don’t assume what is at the other end of a “link”

Avoid “local resources”

- Users are global, not defined by an application (accounts can be local)
- “Type descriptions” are global (defects, requirements, …)

Does the application own storage?

Typical Application model:

- Data access is through an application, application controls integrity
- Storage is an application concern, totally private and fixed

Traditional IDE model:

- Data in files, multiple tools work on the files, files may be all screwed up
- Permanent Storage (e.g. SCM versioning) is a peer application - file system is just a temporary shared cache between applications and permanent storage

What is web equivalent of IDE model?

Linked Data Challenges - detail

Think of "data policy that changes with time" not "inherent characteristics"

- E.g. Defects must have a priority between 1 and 3 – changeable policy.

Assume applications are "black boxes" – use protocols, not frameworks to integrate (c.f. Eclipse)

Fat client or resource-oriented UI?

You can write fat clients in the browser too

- That is exactly what most experienced programmers will do

Fat clients have good support for specific workflows

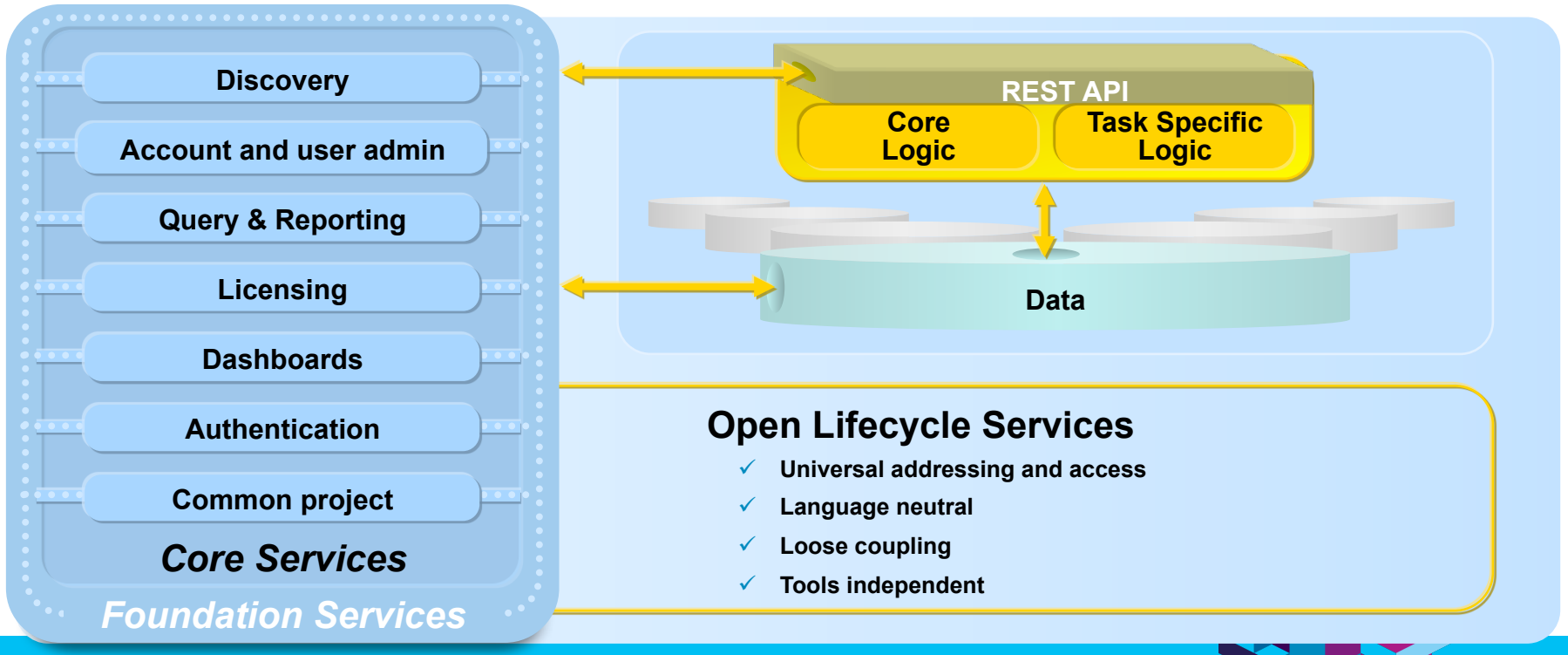
Fat clients have problems

- Long load times
- Closed systems (what do you do when a link leads to pdf, or html or other?)
- More difficult to evolve when workflows change

Another option is “page per resource” UIs

- Embrace page switches

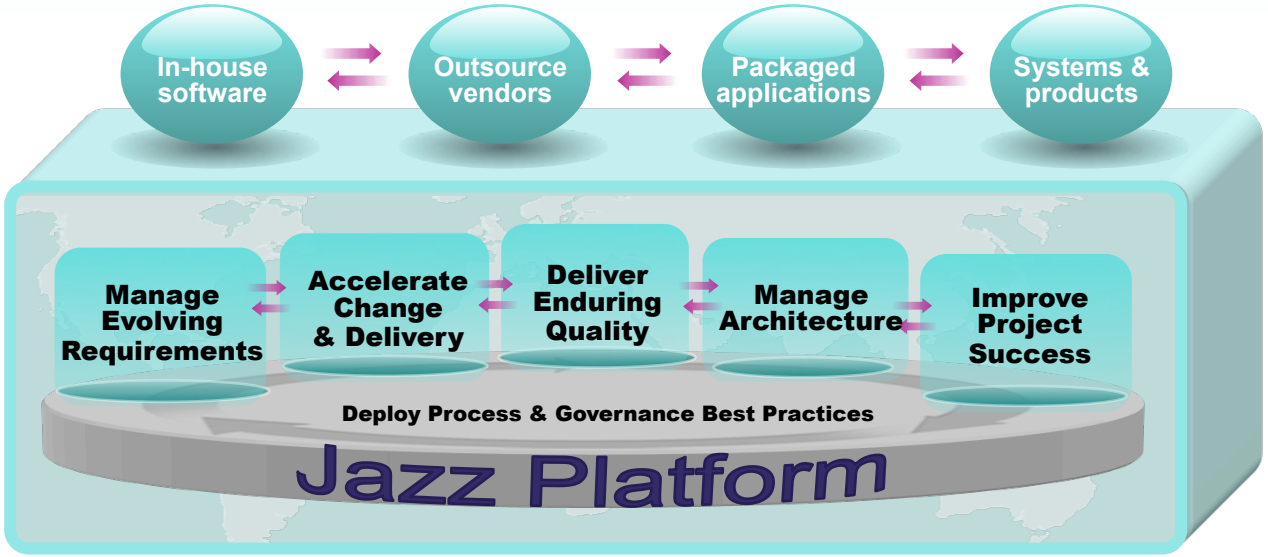
Jazz: Open, extensible, web-centric, integration platform



IBM Rational Software Delivery Platform

Rational. software

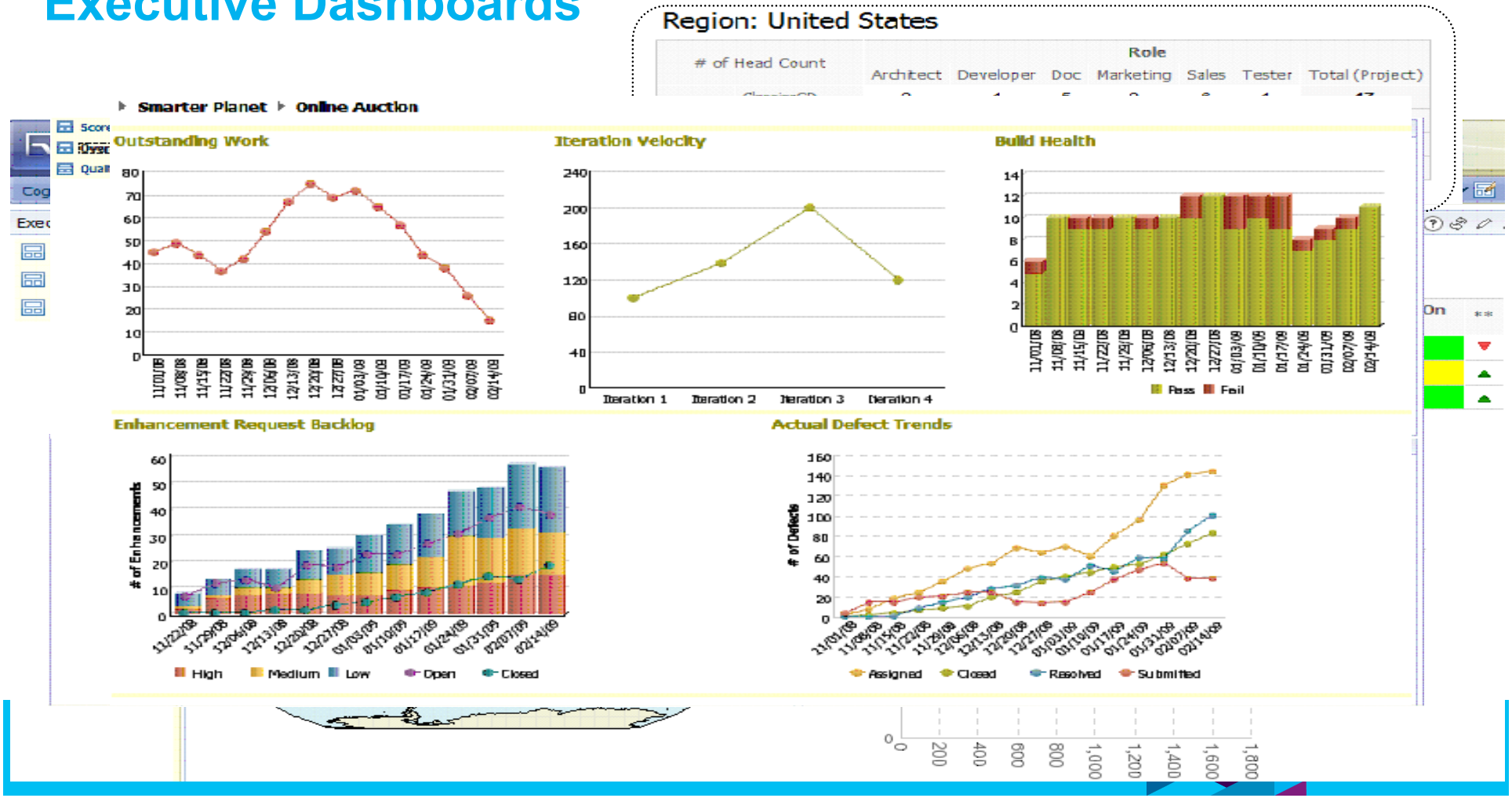
***Solutions** to help customers achieve greater value and performance from their investments in delivering software*



- *Enterprise Modernization and Transformation*
- *Organizational Governance*
- *Skill Development and Community*
- *Implementation Services*

Governance and Control of Software Delivery

Executive Dashboards



Something completely different

The future is uncertain – reason about probabilities

- Pierre de Fermat and Blaise Pascal, correspondence (1654)
- Christian Huygens (1657)
- Jacob Bernoulli (1713)
- Abraham de Moivre (1718)
- Thomas Bayes (1763)
- Pierre-Simon Laplace (1774)

Financier – using probability distributions to evaluate project and portfolio value