When Should You Consider Meta-Architectures?

"Using Meta to Scale the Cloud"

Joseph W. Yoder
joe@refactory.com

JDD

Overview

- What is the Cloud?
- Ways to evolve systems
- What meta really allows
- Meta-architectures and DSLs
- Meta in the Cloud
- Case studies of systems that scale

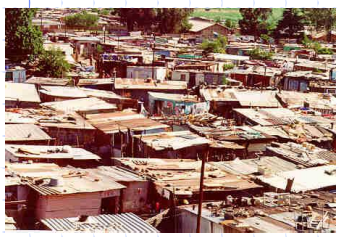
Evolved from UIUC SAG

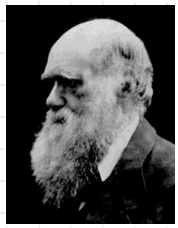
In the early 90's we were studying objects, frameworks, components, meta, refactoring, reusability, patterns, "good" architecture.

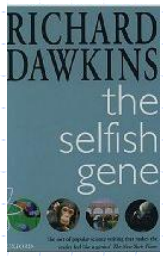
However, in our SAG group we often noticed that although we talk a good game, many successful systems do not have a good internal structure at all.

Selfish Class

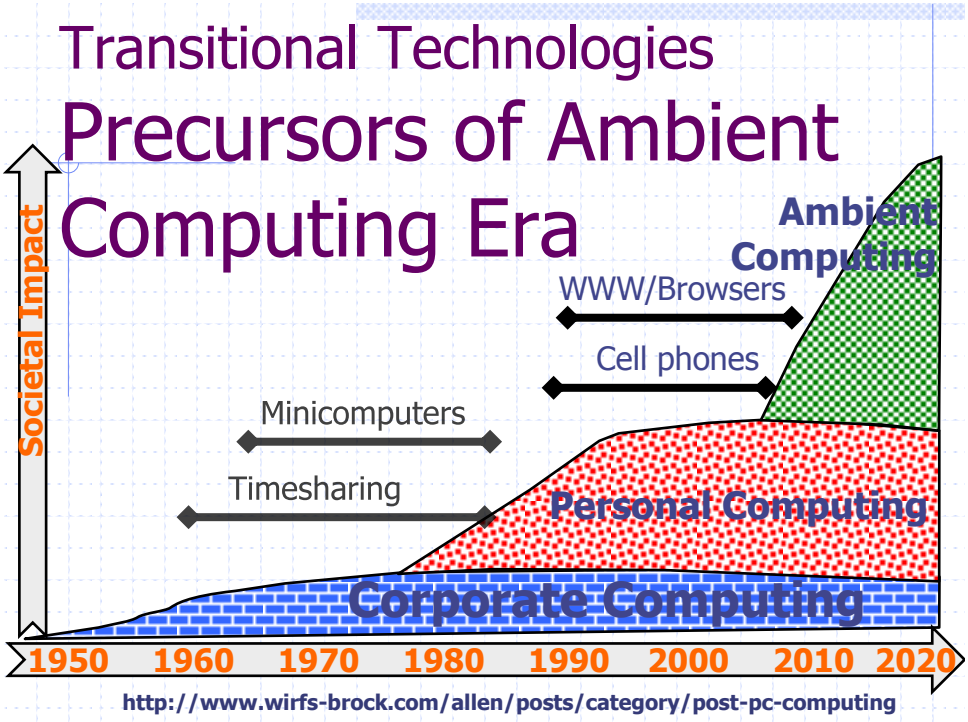
Brian and I had just published a paper "Selfish Class" which takes a *code's-eye view of software reuse and evolution.*







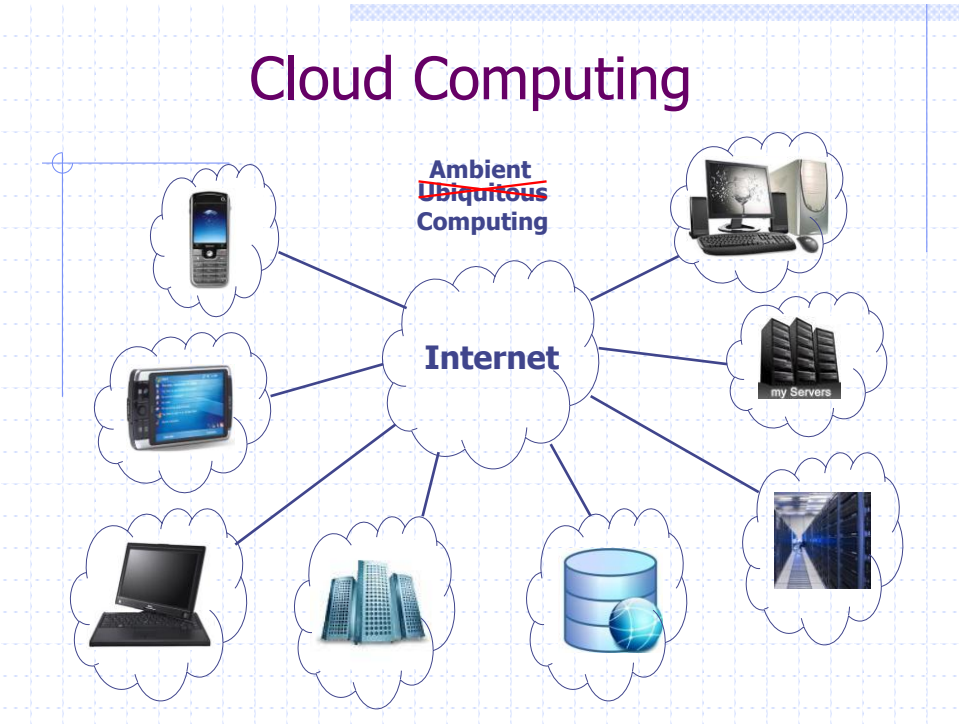
In contrast, our BBoM paper noted that in reality, a lot of code was hard to (re)-use.



Apps on Browser Platforms

Apps are becoming the norm running on whatever platform

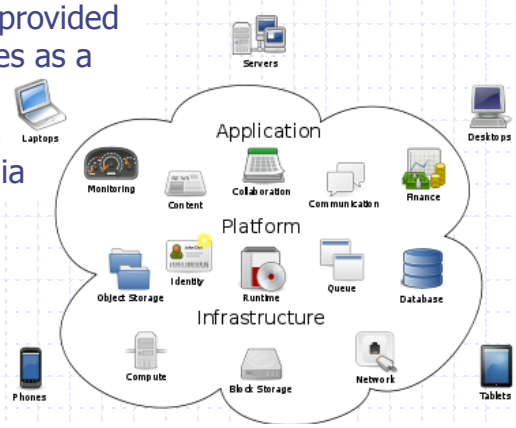

Evolving Needs
my Servers


Cloud Computing
Ambient
Ubiquitous
Computing
Internet
my Servers

What is Cloud Computing?

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software and information are provided to computers and other devices as a utility (like the electricity grid) over a network (i.e. Internet).

--Wikipedia



Traditional vs Cloud Computing

- Traditional - install and maintain
  - Applications, Servers, Networks
  - Integration issues, upgrade headaches
  - Expensive with a lot of overhead
- Cloud Computing
  - Purchase resources (services) from the Cloud
  - Provides a way to scale systems (overnight)
  - No direct maintenance
  - Highly Configurable Systems (glue together)

Cloud Computing

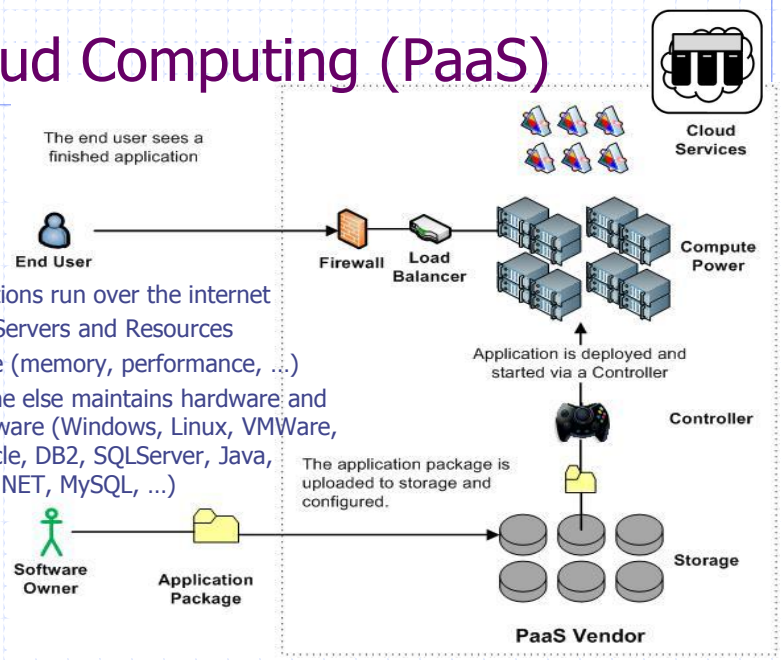
- **Platform as a Service (PaaS)**
- **Software as a Service (SaaS)**
- **Infrastructure as a Service (IaaS)**

Cloud Computing (IaaS)

Infrastructure as a Service (IaaS)

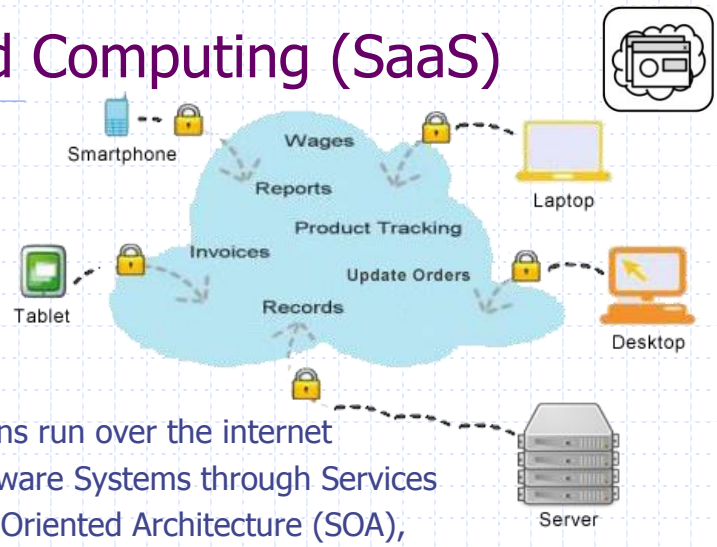
- Hardware (Servers)
- Networking
- Backup and UPS
- Power Sources

Cloud Computing (PaaS)



Applications run over the internet
Virtual Servers and Resources
Scalable (memory, performance, ...)
Someone else maintains hardware and software (Windows, Linux, VMWare, Oracle, DB2, SQLServer, Java, ASP.NET, MySQL, ...)

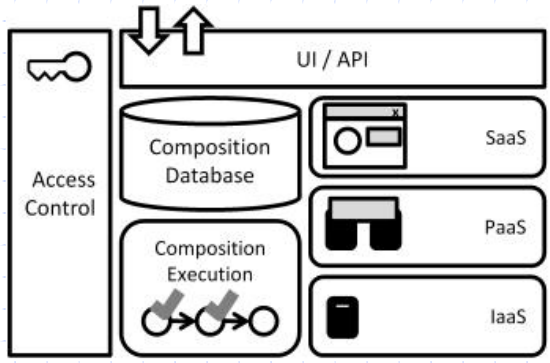
Cloud Computing (SaaS)



Applications run over the internet
Build Software Systems through Services

- Service Oriented Architecture (SOA),
- Mail Services, Web Services, ...
- Credit Card Processing, Invoicing, ...

Cloud Computing (CaaS)



Compositions of Provider Supplied Services residing on the Software, Platform, or Infrastructure layer…

http://cloudcomputingpatterns.org/

What do we want to scale?

- Increased
  - performance
  - throughput

cloud

  - **adaptability/flexibility**
  - **rate of supporting new requirements**
  - **time to deploy**
  - **support for configuration options**
- Reduced code bulk
- Higher-leverage code (frameworks, tools…)

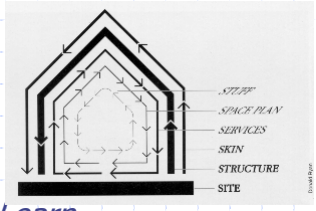
Metadata is powerful technique for scaling

Motivation: Need to Quickly Adapt to Change

- Business Rules or Domain Elements and Resource Needs are changing quickly:
  - New calculations for insurance policies and new types of policies offered
  - Online store catalog with new products and services and rules applying to them
  - New cell phone product and services…
- Need quick ways to **develop** and **adapt** to these changing requirements.

Stuart Brand's Shearing Layers

- Buildings are a set of components that evolve in different timescales.
- Layers: site, structure, skin, services, space plan, stuff. Each layer has its own value, and speed of change (pace).
- Buildings adapt because faster layers (services) are not obstructed by slower ones (structure).



—Stuart Brand, *How Buildings Learn*

Yoder and Foote's Software Shearing Layers

- "Factor your system so that artifacts that change at similar rates are together."—Foote & Yoder, Ball of Mud, PLoPD4

Layers

Slower

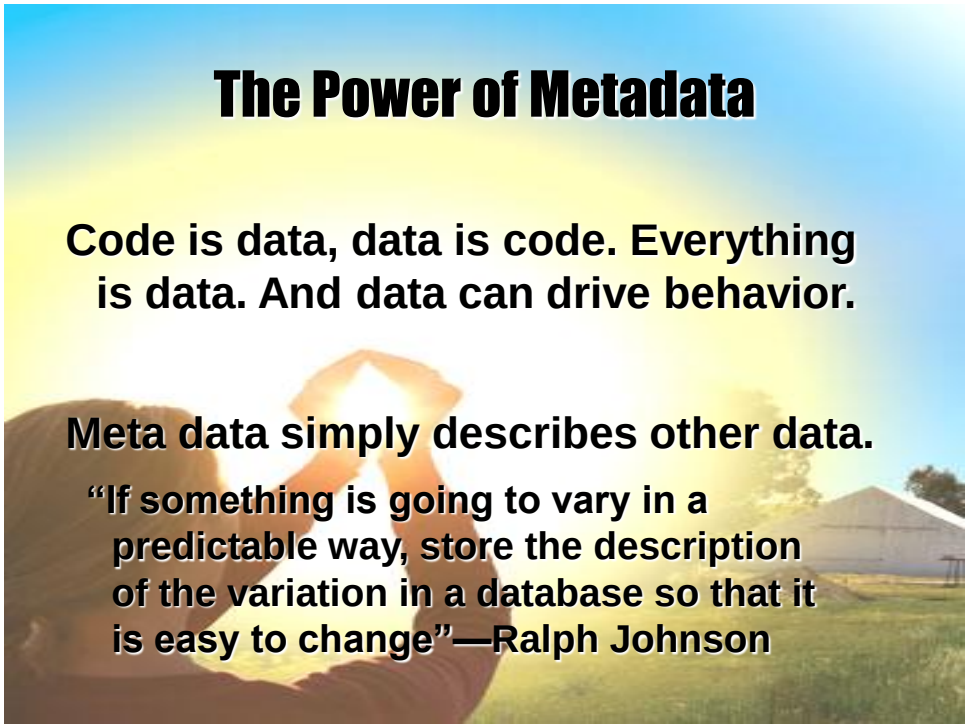
- The platform
- Infrastructure
- Data schema
- Standard frameworks, components, and modules
- Abstract classes and interfaces
- Classes
- Code
- Data

Faster

In my youth…two bad words

M and R words

Metadata and Reflection



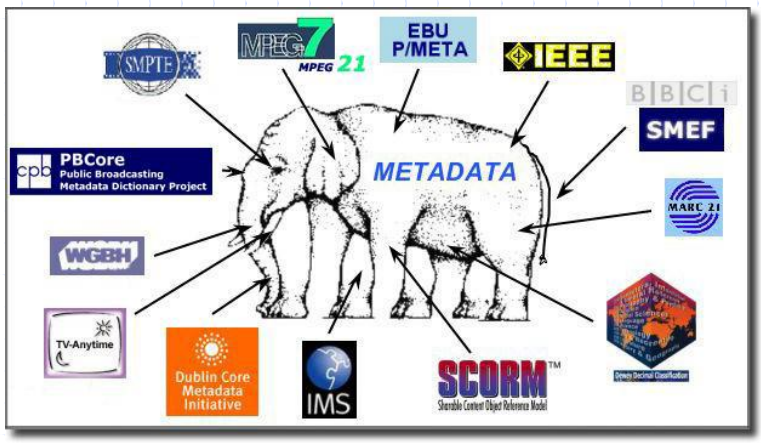
Meta-Architecture
Where are they Useful?

- Meta information about the domain model and business rules is interpreted at runtime.
  - Entities, attributes, behaviors and relationships described by *metadata*
  - *Rules, Processes, and Resource Needs*
  - *Descriptions* stored in a database or in files and interpreted (can be XML/XMI).
- When you change the metadata, system behavior changes and needed resources.
- Domain experts typically make changes, not programmers.

Using Meta to Scale: A Disclaimer

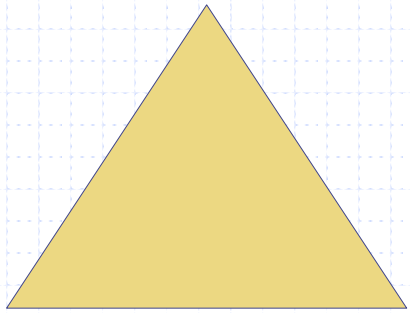
- I am **not** considering:
  - Using metadata to generate code.
  - Building a "stack" of meta-models and/or transforming between various models.
- Will focus on practical techniques and for using metadata to scale solutions and making them more adaptable!

Metadata



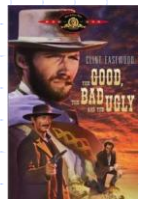
Metadata – the Three "M"s

Metaobjects



Markup

Manifest



Brian Foote: Metadata, The Good, The Bad, The Ugly
Mini-SugarLoaf PLoP, São Paulo, Brazil - September 2011

Metadata – Markup


```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/
    1999/XSL/Transform" version="1.0">
<xsl:key match="/publisher/books/book" use="author-ref/@ref"
    name="books-by-author"/>
<xsl:template match="/"><bestsellers-list> <xsl:apply-templates
    select="/publisher/authors/author"> <xsl:sort select="sum(key('books-
    by-author', @id)/sold)"/> <xsl:sort select="last_name"/> </xsl:apply-
    templates> </bestsellers-list></xsl:template>
<xsl:template match="author"> <copy> <name> <xsl:value-of
    select="last_name"/>, <xsl:value-of select="first_name"/> </name>
    <total_publications> <xsl:value-of select="count(key('books-by-author',
    @id))"/> </total_publications> <total_sold> <xsl:value-of
    select="sum(key('books-by-author', @id)/sold)"/>
    </total_sold> <rank> <xsl:value-of select="position()"/>
    </rank> </copy> </xsl:template>
</xsl:stylesheet>
```

Metadata – Annotations

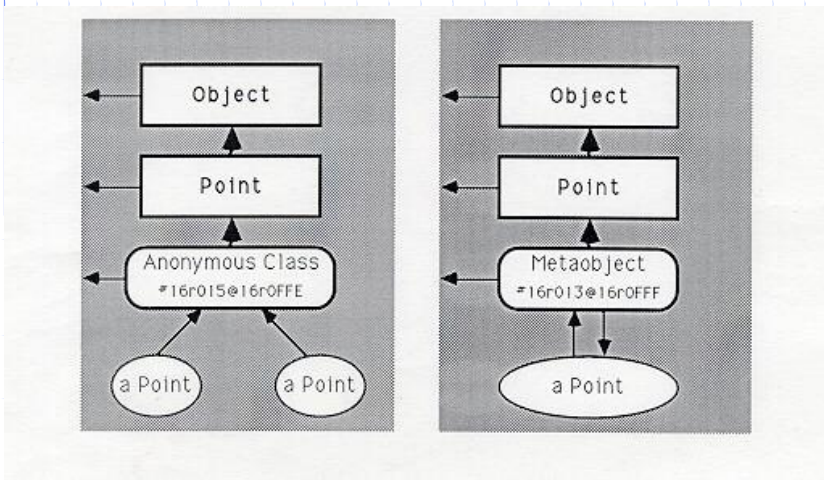

```
public class Car {
    // Injectable constructor
    @Inject public Car(Engine engine) { ... }
    // Injectable field
    @Inject private Provider<Seat> seatProvider;
    // Injectable package-private method
    @Inject void install(Windshield windshield, Trunk trunk) { ... }
}
```

Metadata – Manifest

DTD

```
<!DOCTYPE note [
    <!ELEMENT note (to,from,heading,body)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT heading (#PCDATA)>
    <!ELEMENT body (#PCDATA)>
]>
```

Metadata – MetaObject



Metadata – Most use 2*M

Markup and Manifest are "good" enough for most successful meta-architectures…

Though sometimes you need to go all the way!!!

→ Meta the Meta the Meta ☺

"Anything You can do I can do Meta"

Example Meta-Architecture: Adaptive Object-Model

- Represent classes, attributes, behaviors and relationships as *metadata*
- Experts change the *metadata* (object model) to reflect changes in the domain.
- *Object-Model* stored in a database or in files and interpreted (can be XML/XMI).

Consequently, the object model is adaptable without writing code. When you change the metadata, the system behavior changes.

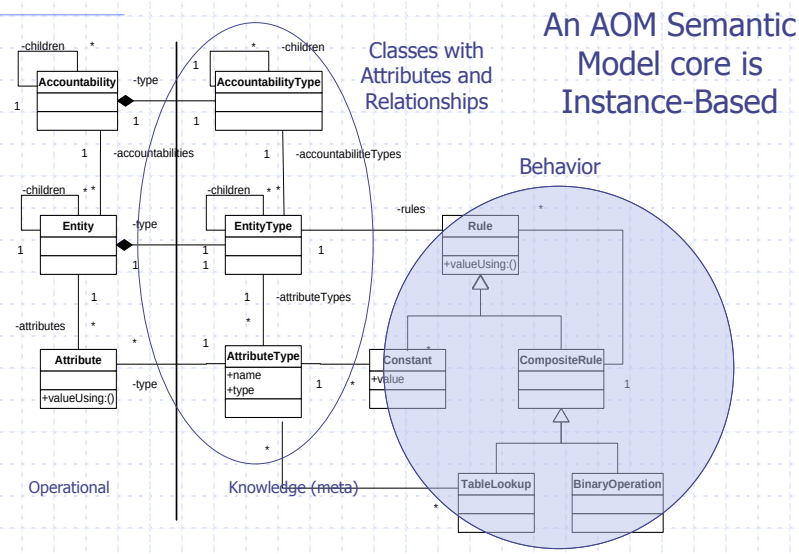
www.adaptiveobjectmodel.com

Architecture Elements of Adaptive Object-Models

- Metadata
- TypeObject
- Properties
- Type Square
- Strategy/RuleObjects
- Entity-Relationship
- Interpreters/Builders
- Editors/GUIs

If you want something to change quickly, push it into data and build tools geared towards changemakers' needs.

Adaptive Object Model "Core Domain Architecture"



AOM and TypeSquare (code example)

```
public class EntityType {
  private string _typeName;
  private Type _type;
  public EntityType (string name, Type type) {
    _typeName = name;
    _type = type; }
  public string Name {
    get{return _typeName;}
    set{_typeName = value;} }…}
```

http://www.codeproject.com/
Christopher G. Lasater

Adaptive Object Model "Describing Your Entities"

```
<entities>
    <entity name="President"
        entityType="TopLevelExecutiveType"
        actualAttributes="EmployeeId"
        />
    <entity name="VicePresident"
        entityType="ExecutiveType"
        actualAttributes="EmployeeId"
        />
    ....
</entities>
```

Adaptive Object Model "Describing Your Relationships"

```
<accountibilities>
    <!-- Hiring //-->
    <accountability name="PresidentHiresVicePresident"
        accountabilityType="TopLevelExecutiveHiresExecutive"
        actualParent="President"
        actualChildren="VicePresident" />
    <accountability name="VicePresidentHiresSalesManager"
        accountabilityType="ExecutiveHiresManager"
        actualParent="VicePresident"
        actualChildren="SalesManager" />
    ...
</accountibilities>
```

Systems Support Four Different Types of Behavior

- If behavior varies, specify rules in meta-data and interpret. Typical behaviors:
  - Constraints on values, relationships, state changes…(discrete values, ranges, permissible associations)
  - Functional
  - Workflow
  - Event based

Where Metadata Techniques Have been Successfully Used:

- Representing Insurance Policies
- Telephony Marketing Systems
- Telephone Billing and Back Office Systems
- Workflow Systems
- Medical Observations
- Banking and Trading
- Validate Equipment Configuration
- Documents Management System
- Gauge Calibration Systems
- Simulation Software
- Invoicing Systems
- Manufacturing Systems
- …

(some can be found in papers)

www.adaptiveobjectmodel.com

Contrast: Two Ways to Evolve a Working System

	Traditional SW Architecture	Meta-Data Based Architecture
Who implements a requirements change?	A programmer.	A domain expert might, if it is a domain object or business rule change.
How are changes verified?	Programmer writes tests, QA writes and runs acceptance tests, end-user approves.	Still have to run all unit and acceptance tests. But can also build into end-user tool checks that model changes won't "break" the existing working system.
How often can the system be updated in production?	At the end of an iteration.	Whenever changes are verified. Not tied to dev cycle.
What's the big deal?	Still have to go through some release cycle. Programming and deployment can become bottlenecks.	Significant changes can be made by end-users. Releasing can be simply updating production metadata.

Different Types of Cloud Metadata

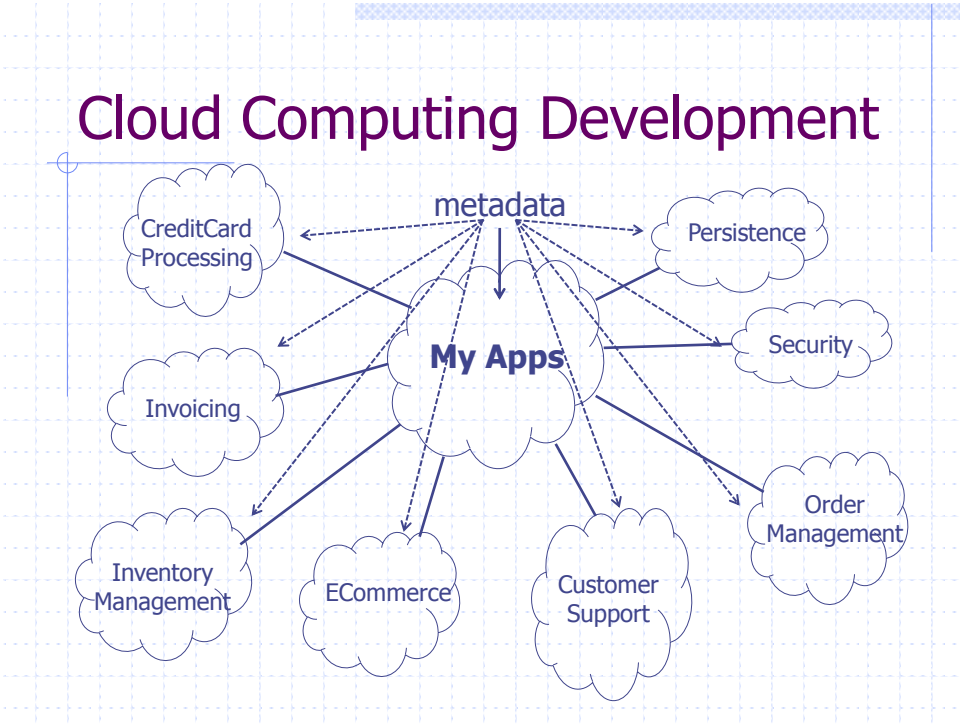
- **Basic metadata** - low level data - e.g. block level information about where data is stored and how often it is accessed. (PaaS)
- **File-level metadata** - more complex data from file systems. (IaaS)
- **Content-level metadata** - metadata that might be found in content management systems such as file type and other more meaningful data such as: "Is this email from the CEO?" or … "Is the mammogram positive?" (SaaS)

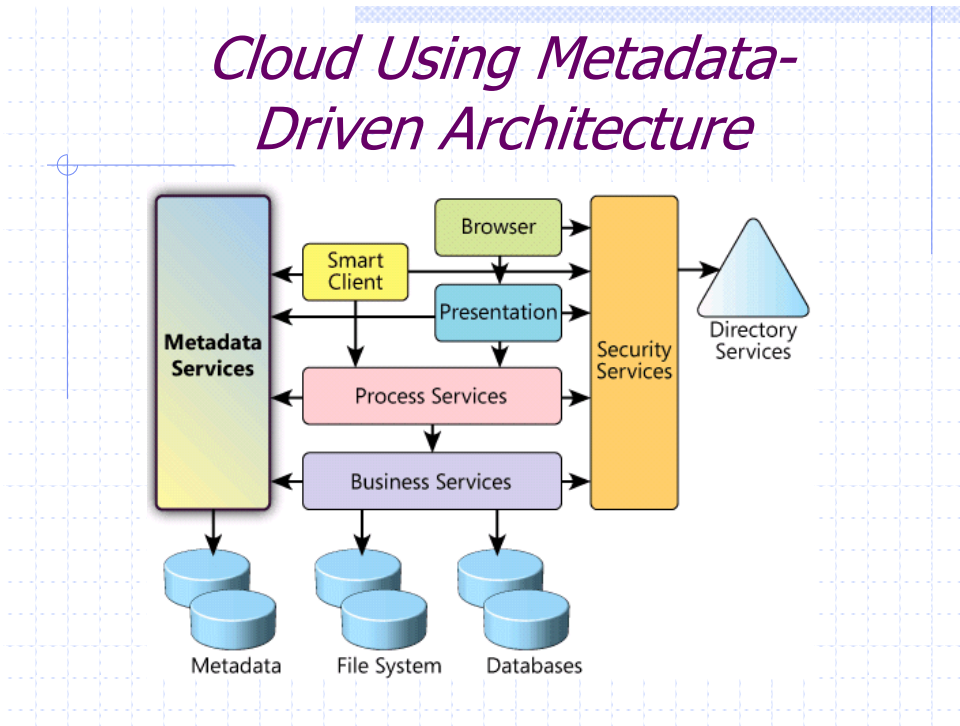
Custom Metadata

Cloud Computing and Metadata

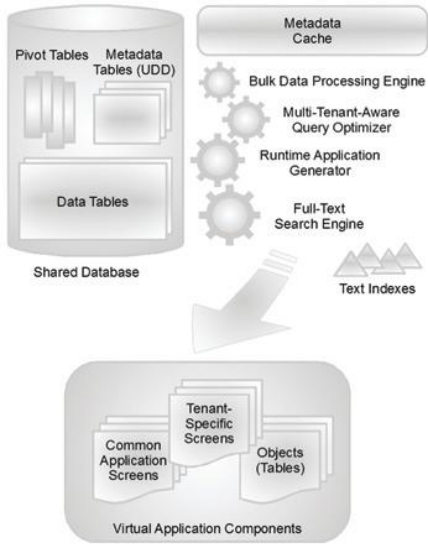
Cloud Components/Services are configurable

- Use Metadata to describe
  - Resources (computational, memory, …)
  - Configuration (frameworks etc)
  - Security and Roles
  - Types of Entities
  - Business Rules
  - Processes
  - …


Cloud Computing Development
metadata
CreditCard Processing
Persistence
My Apps
Security
Invoicing
Order Management
Inventory Management
ECommerce
Customer Support


Cloud Using Metadata-Driven Architecture
Browser
Smart Client
Presentation
Metadata Services
Security Services
Directory Services
Process Services
Business Services
Metadata
File System
Databases

Force.com's Metadata-Driven Architecture



http://www.salesforce.com/au/assets/pdf/Force.com_Multitenancy_WP_101508.pdf

Issues that need addressed: Metadata Based Architectures

- Requires infrastructure for storing, building, and interpreting metadata.
- Requires strong design skills to create.
- Trust Relationships (Security, Validity, Tested)

Issues that need to addressed: Metadata Based Architectures

- Interpreting metadata may result in lower performance (this can be remedied by well-known techniques)
- Versioning – keeping things consistent, evolution, history, backwards compatibility, etc..

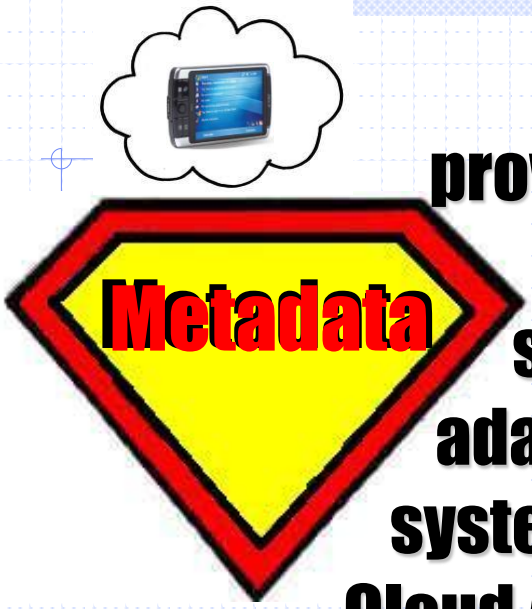
Things are Changing

Social Networking surpasses Email in 2009

Smartphones and Tablets pass PCs in sales

Gartner Reports for 2012:

- 20% of Businesses will own no IT Assets
- Cloud Computer will be as influential as E-Commerce



Metadata has proven to be very powerful and successful for adapting various systems including Cloud Architectures

Summary

- Cloud is becoming the norm…
  - can help with shared resources and services available on the Internet
- Meta-architectures enable rapid changes that are domain specific
- Core application code less than with conventional programming
- Scale in two dimensions:
  - support for rapid requirements changes
  - design leverage (team size ~10^1)

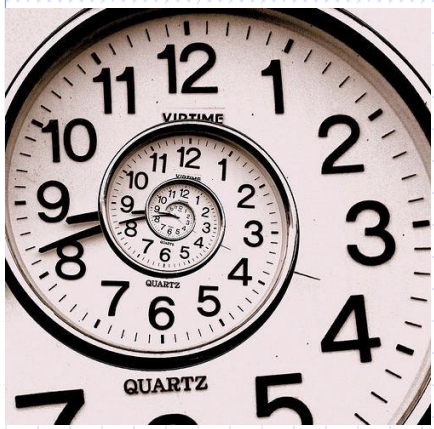
Meta Collaborators

- Ademar Aguiar
- Francis Anderson
- Ali Arsanjani
- Jean Bezivin
- Paulo Borba
- Filipe Correia
- Krzysztof Czarnecki
- Ayla Dantas
- Martine Devos
- Hugo Ferreira
- Brian Foote
- Martin Fowler
- Richard Gabriel
- Atzmon Hen-Tov
- Ralph Johnson
- David H. Lorenz
- Lena Nikolaev
- Jeff Oaks
- Reza Razavi
- Nicolas Revault
- Dirk Riehle
- Lior Schachter
- Dave Thomas
- Michel Tilman
- Leon Welicki
- Rebecca Wirfs-Brock ...

Resources

- Adaptive Object Models
  - www.adaptiveobjectmodel.com
- Agile Software
  - Refactoring www.refactory.com
  - Agile Alliance: www.agilealliance.org
  - The Agile Manifesto
- Cloud Computing www.cloudcomputingpatterns.org
- MetaPLoP.org
- Force.com and SalesForce.com

That's All … **Dzięki!!!**



http://www.refactory.com

joe@refactory.com
Twitter: @metayoda