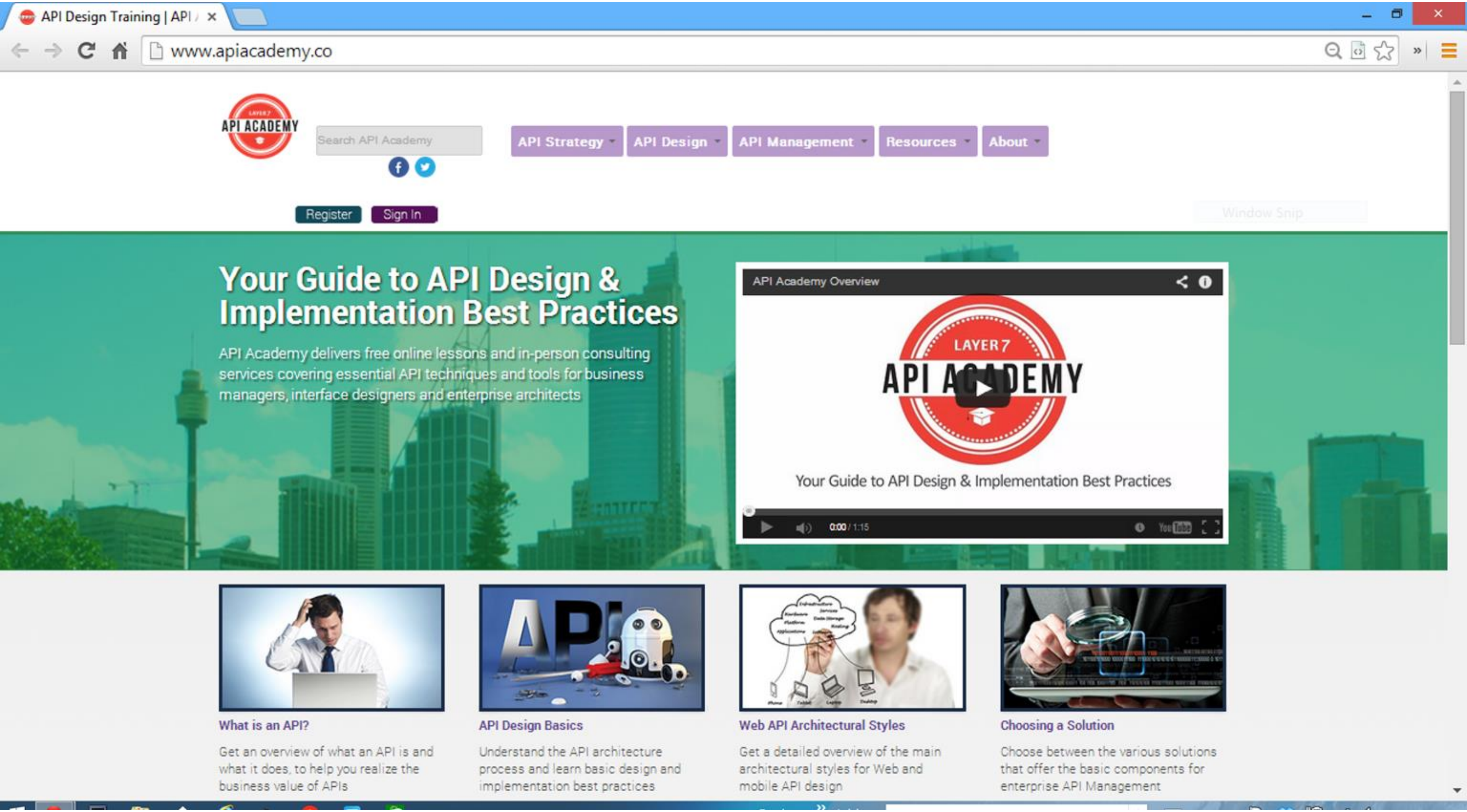
Web API Design Maturity Model

Mike Amundsen
@mamund
API Academy at CA Technologies

PgUp
Ctrl
Mike Amundsen
@mamund



Your Guide to API Design & Implementation Best Practices

API Academy delivers free online lessons and in-person consulting services covering essential API techniques and tools for business managers, interface designers and enterprise architects

Your Guide to API Design & Implementation Best Practices

What is an API?

Get an overview of what an API is and what it does, to help you realize the business value of APIs

API Design Basics

Understand the API architecture process and learn basic design and implementation best practices

Web API Architectural Styles

Get a detailed overview of the main architectural styles for Web and mobile API design

Choosing a Solution

Choose between the various solutions that offer the basic components for enterprise API Management

O'REILLY

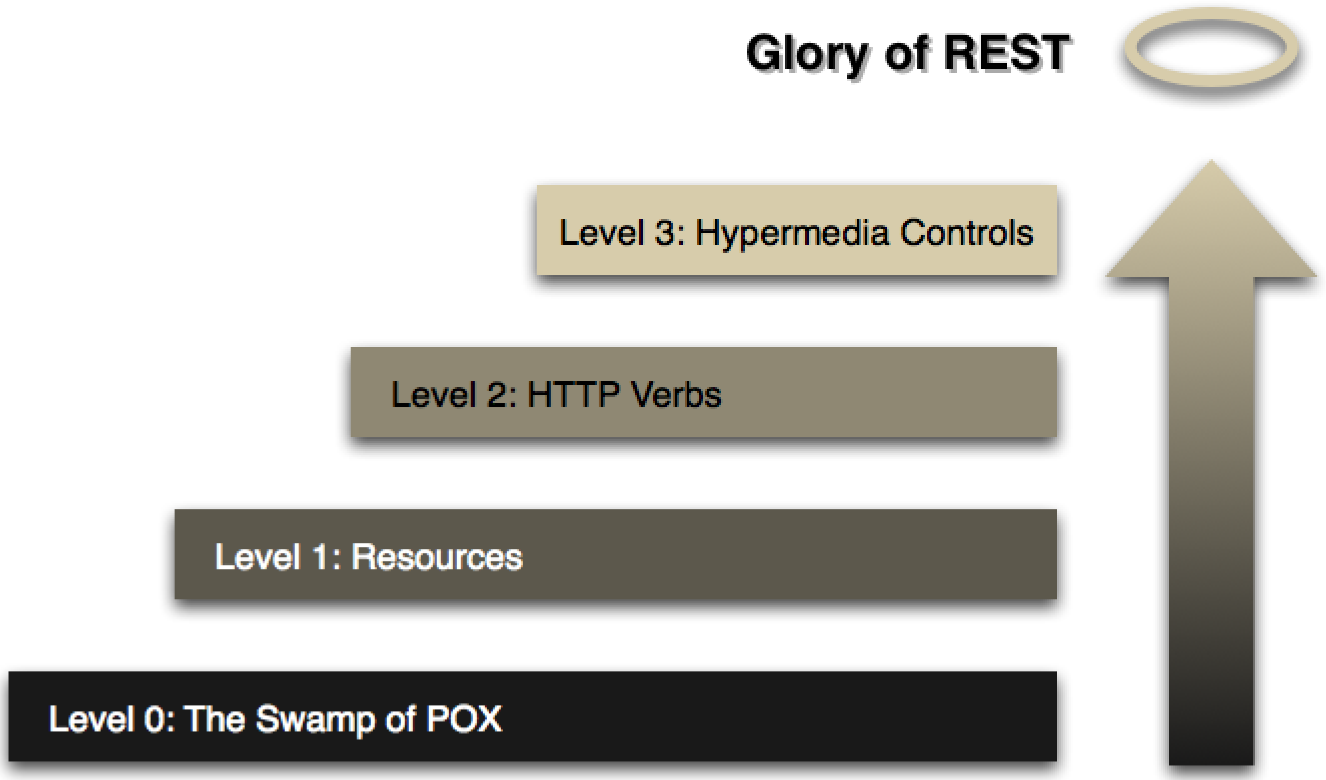
RESTful Web Clients

ENABLING REUSE THROUGH HYPERMEDIA

Mike Amundsen

Web API Design Maturity Model

Richardson Maturity Model (via Martin Fowler)

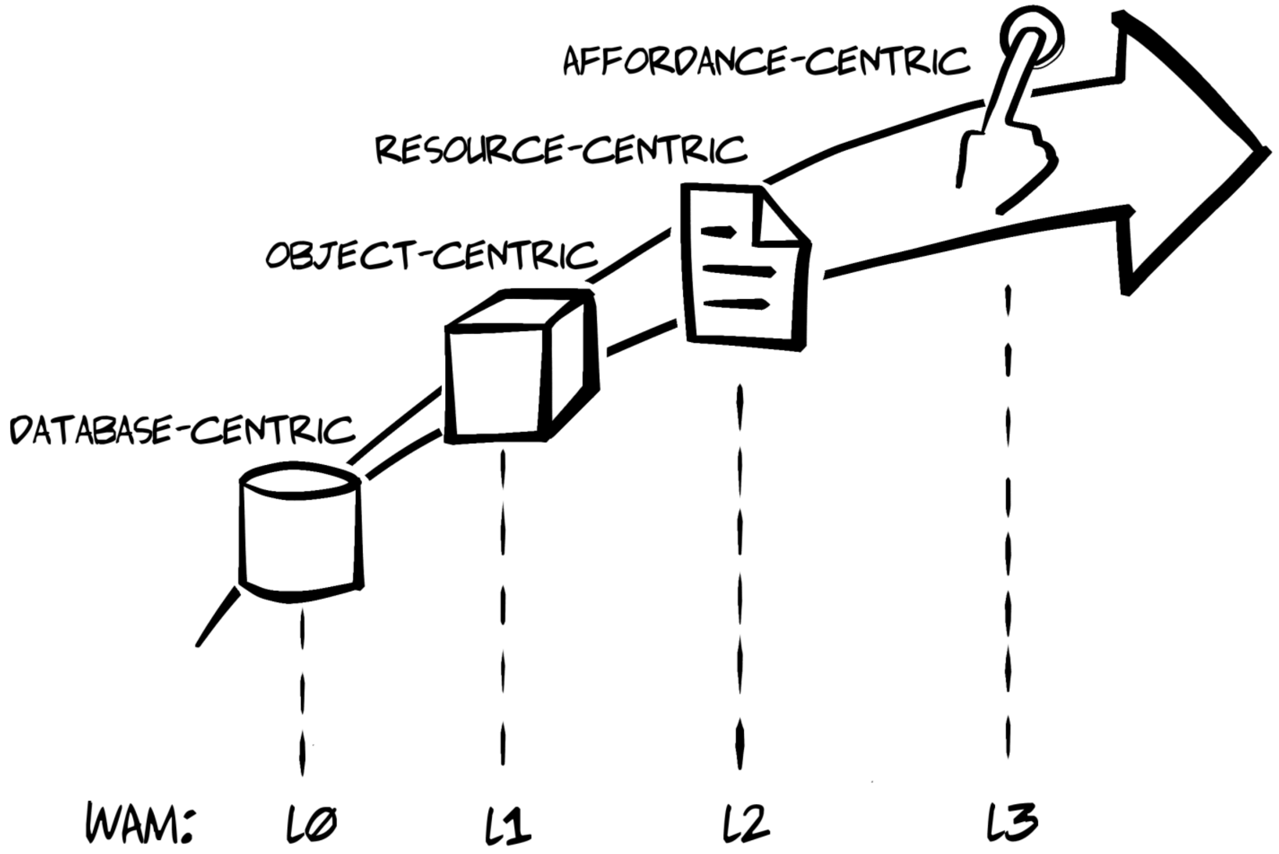


http://martinfowler.com/articles/richardsonMaturityModel.html

“I did RMM as a maturity model because I noticed that each 'step' corresponded to the adoption of a specific technology.”

Leonard Richardson, NYPL

Web API Design Maturity Model



“I did WADM as a maturity model because I noticed that each 'step' corresponded to the adoption of a specific model description to expose as the API.”

Mike Amundsen, 2016

Maturity Models

RMM

Focus on the API response documents.

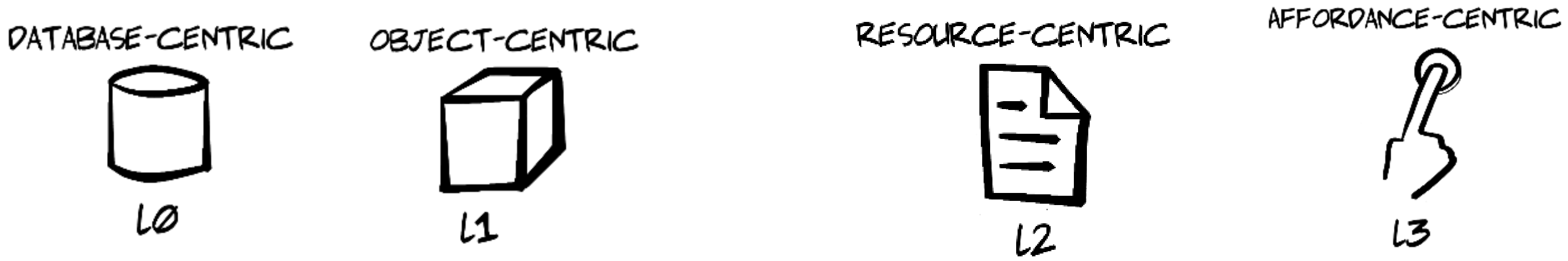
WADM

Focus on the API description documents.

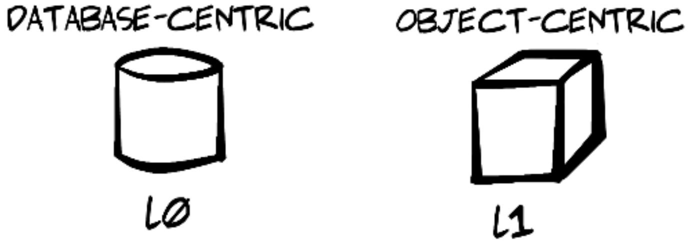
Web API Design Maturity Model

Internal Models

External Models



Internal Models

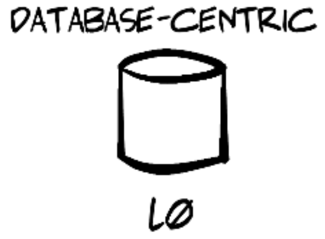


Data-Centric (WADM.L0)

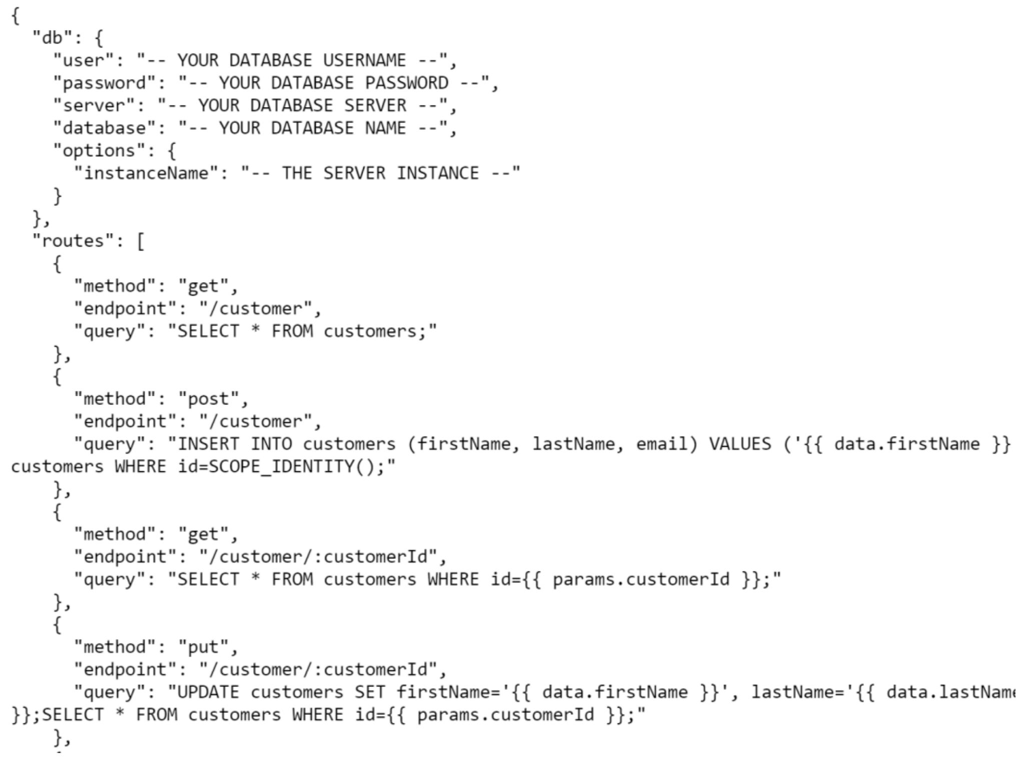
API is the exposed data model

The “go-to” approach for many enterprise IT

Lots of off-the-shelf and SaaS products available



Data-Centric (WADM.L0)



```
{
  "db": {
    "user": "-- YOUR DATABASE USERNAME --",
    "password": "-- YOUR DATABASE PASSWORD --",
    "server": "-- YOUR DATABASE SERVER --",
    "database": "-- YOUR DATABASE NAME --",
    "options": {
      "instanceName": "-- THE SERVER INSTANCE --"
    }
  },
  "routes": [
    {
      "method": "get",
      "endpoint": "/customer",
      "query": "SELECT * FROM customers;"
    },
    {
      "method": "post",
      "endpoint": "/customer",
      "query": "INSERT INTO customers (firstName, lastName, email) VALUES ('{{ data.firstName }}
customers WHERE id=SCOPE_IDENTITY();"
    },
    {
      "method": "get",
      "endpoint": "/customer/:customerId",
      "query": "SELECT * FROM customers WHERE id={{ params.customerId }};"
    },
    {
      "method": "put",
      "endpoint": "/customer/:customerId",
      "query": "UPDATE customers SET firstName='{{ data.firstName }}', lastName='{{ data.lastNam
}};SELECT * FROM customers WHERE id={{ params.customerId }};"
    },
```

https://www.npmjs.com/package/resquel

Data-Centric (WADM.L0)

Virtually NO design, so this is “level zero” on WADM scale

Upside:

Quick and easy

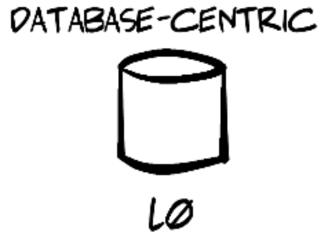
Downside:

Exposes IP

Tight-coupling to internal model

May depend on unique data-tech (GROUP-BY, etc.)

Provider push cost of change to consumers



“First step in breaking the data-centric habit, is to stop designing systems as a collection of data services, and instead design for business capabilities.”

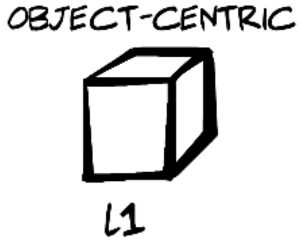
Irakli Nadareishvili, 2016

Object-Centric (WADM.L1)

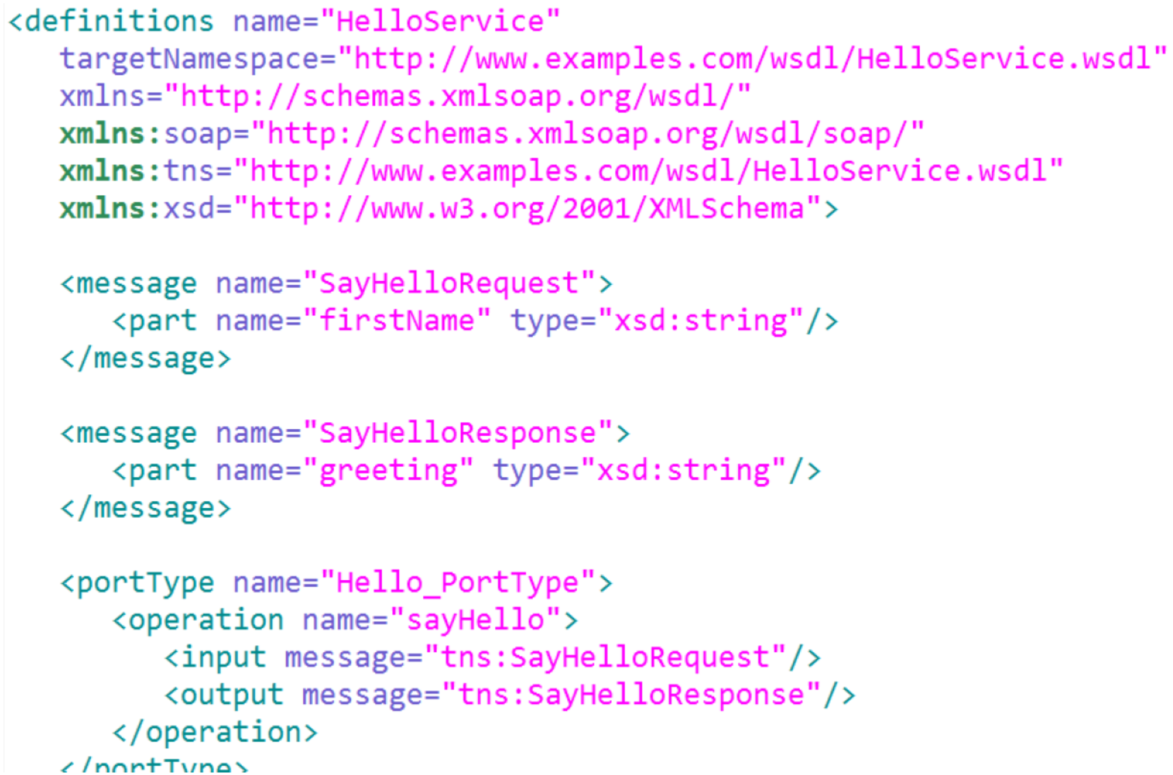
API is the exposed object model

Common for SOA or Canonical Model approach

Classic SOAP-style implementation pattern

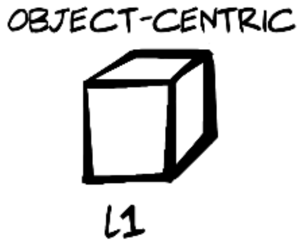


Object-Centric (WADM.L1)



```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>

   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>
```

http://www.tutorialspoint.com/wsdl/wsdl_example.htm



Object-Centric (WADM.L1)

Some design, so this get’s “level one” on the WADM scale

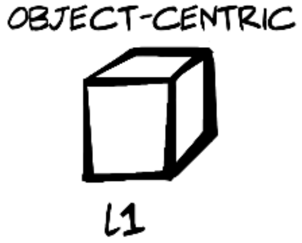
Upside:

- Lots of great tool support
- Models can be quick rich and targeted

Downside:

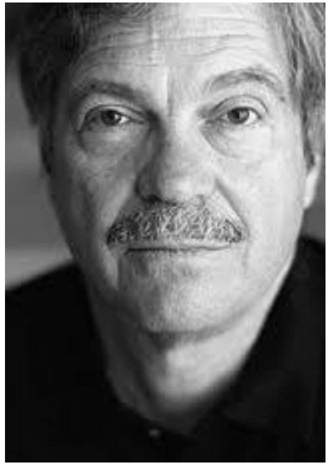
- Changes to internal models leak out to interface
- Often consumer model is not provider model (esp. mobile)

Coordinating consumer/provider models can be “heavy-handed”

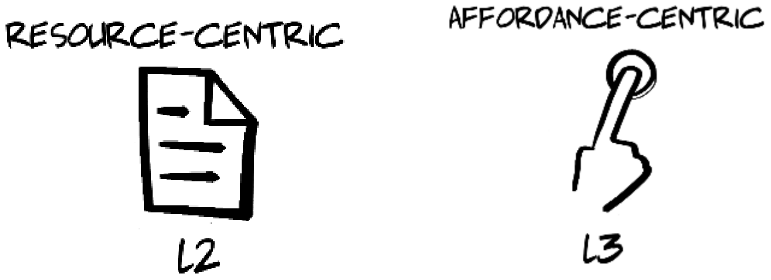


“I'm sorry that I long ago coined the term ***objects*** *for this topic because it gets many people to focus on the lesser idea. The big idea is* ***messaging****.”*

Alan Kay, 1998



External Models

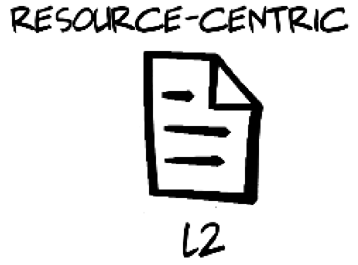


Resource-Centric (WADM.L2)

API is a set of HTTP-style resources

Common for Web and mobile development shops

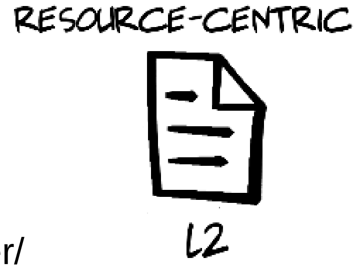
Lots of Resource-First products (Swagger/OAI, RAML, Blueprint, etc.)



Resource-Centric (WADM.L2)

```
### Edit A Product [PATCH]
Updates A Product

+ Request (application/json)

        {
        "id": "1",
        "name": "Product One",
        "description": "This is the full description of the product.",
        "url": "http://example.com",
        "image": "http://example.com/image.jpg",
        "thumbnailUrl": "http://example.com/image-thumb.jpg",
        "keywords": "western, cowboy",
        "brand": "Brand Name",
        "color": "Black",
        "itemCondition": "New",
        "manufacturer": "Manufacturer Name",
        "model": "Black",
        "sku": "SKU #",
        "weight": "12 pounds",
        "width": "12 inches",
        "height": "12 inches",
        "depth": "12 inches"
        }

+ Response 200

    [Product][]

### Delete A Product [DELETE]
+ Response 204
```

http://apievangelist.com/2014/03/08/hello-world-product-api-with-blueprint-raml-and-swagger/



Resource-Centric (WADM.L2)

External design earns this one “level 2”

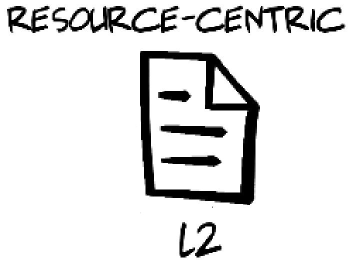
Upside:

Focus is on the interface
Often has a consumer focus (when done well)

Downside:

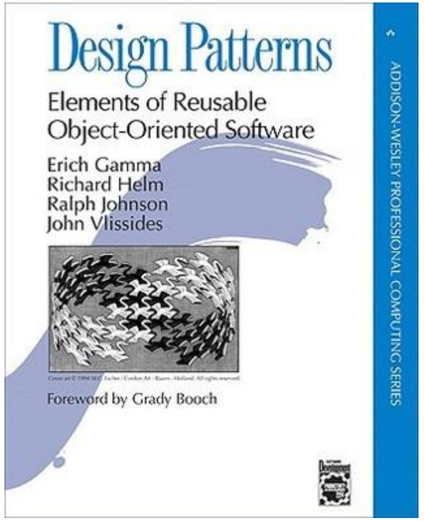
Sometimes just the internal object model (CRUD)
Usually HTTP-centric (WebSockets? Reactive? Thrift?)

Often still leaks internal objects and requires isomorphic models



“Program to an interface, not an implementation.”

Gamma, et al, 1992

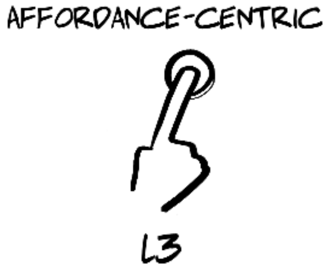

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Affordance-Centric (WADM.L3)

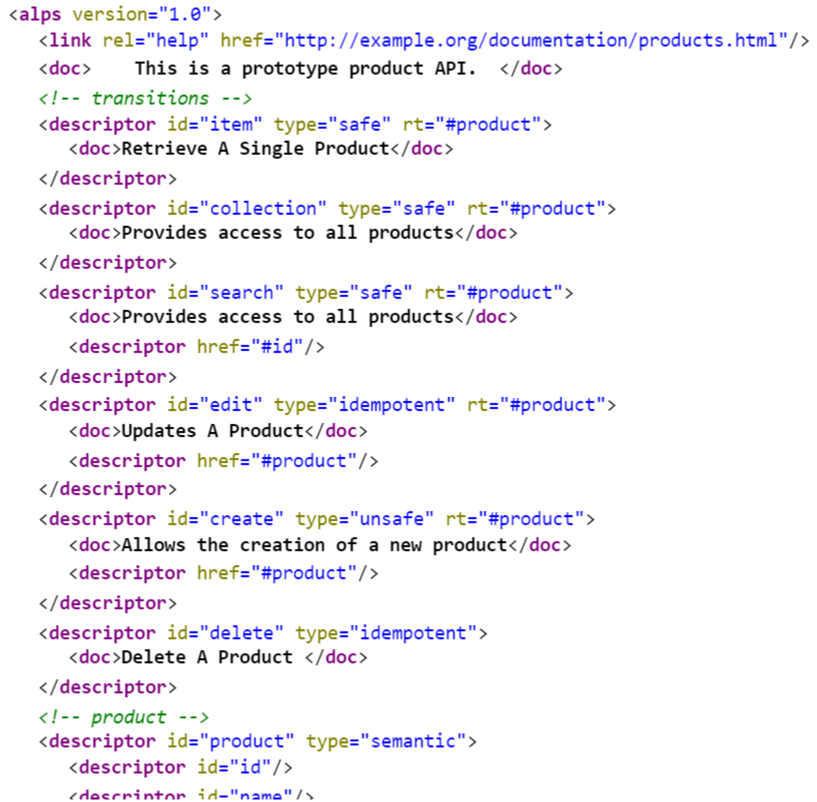
API is a set of action descriptions (e.g. hypermedia controls)

Common for hypermedia-style implementations

Several registered media types (HAL, Siren, Collection+JSON, UBER, etc.)

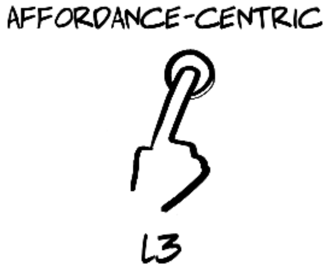


Affordance-Centric (WADM.L3)



```
<alps version="1.0">
   <link rel="help" href="http://example.org/documentation/products.html"/>
   <doc>    This is a prototype product API.   </doc>
   <!-- transitions -->
   <descriptor id="item" type="safe" rt="#product">
      <doc>Retrieve A Single Product</doc>
   </descriptor>
   <descriptor id="collection" type="safe" rt="#product">
      <doc>Provides access to all products</doc>
   </descriptor>
   <descriptor id="search" type="safe" rt="#product">
      <doc>Provides access to all products</doc>
      <descriptor href="#id"/>
   </descriptor>
   <descriptor id="edit" type="idempotent" rt="#product">
      <doc>Updates A Product</doc>
      <descriptor href="#product"/>
   </descriptor>
   <descriptor id="create" type="unsafe" rt="#product">
      <doc>Allows the creation of a new product</doc>
      <descriptor href="#product"/>
   </descriptor>
   <descriptor id="delete" type="idempotent">
      <doc>Delete A Product </doc>
   </descriptor>
   <!-- product -->
   <descriptor id="product" type="semantic">
      <descriptor id="id"/>
      <descriptor id="name"/>
```

https://gist.github.com/mamund/9443276



Affordance-Centric (WADM.L3)

External design independent of all internal models makes this one “level 3”

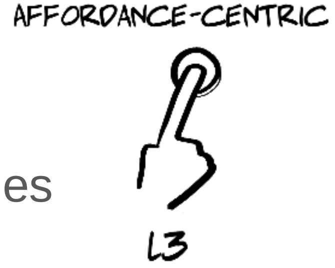
Upside:

Focus is on the use-cases, actions
Usually doesn’t restrict protocol, format, or workflow

Downside:

Very few tools/practices widely shared
for M2M cases, relies on custom code and/or vocabularies

Focus on actions over data means more reliance on shared dictionaries



“When I say hypertext, I mean the simultaneous presentation of information and controls such that the information becomes the affordance through which the user (or automaton) obtains choices and selects actions.”

Roy T. Fielding, 2008

So, what does this all mean?

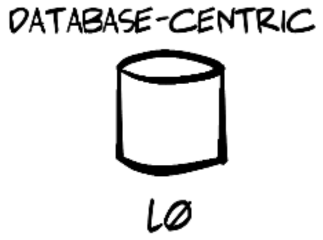
Modeling at different levels…

Data model may have:

Customer Table

Invoice Table

CustomerVisits Table



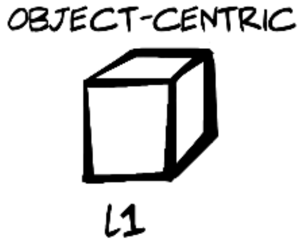
Object Model may have:

CustomerSummary

(basic info, summary of invoices, & visits)

`CustomerSummary.Read,`

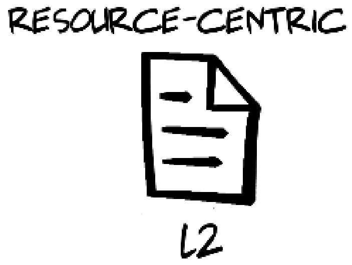
`.FilterByName, .Update, .Suspend`, etc.



Modeling at different levels…

Resource model may have:

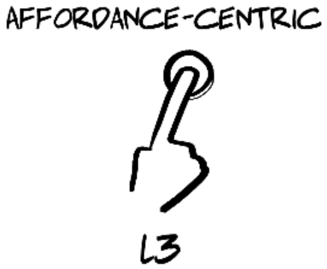
`/customersummary/{custid}`
with a LINK to `/invoices/{custid}`
and a LINK to `/visits/{custid}`



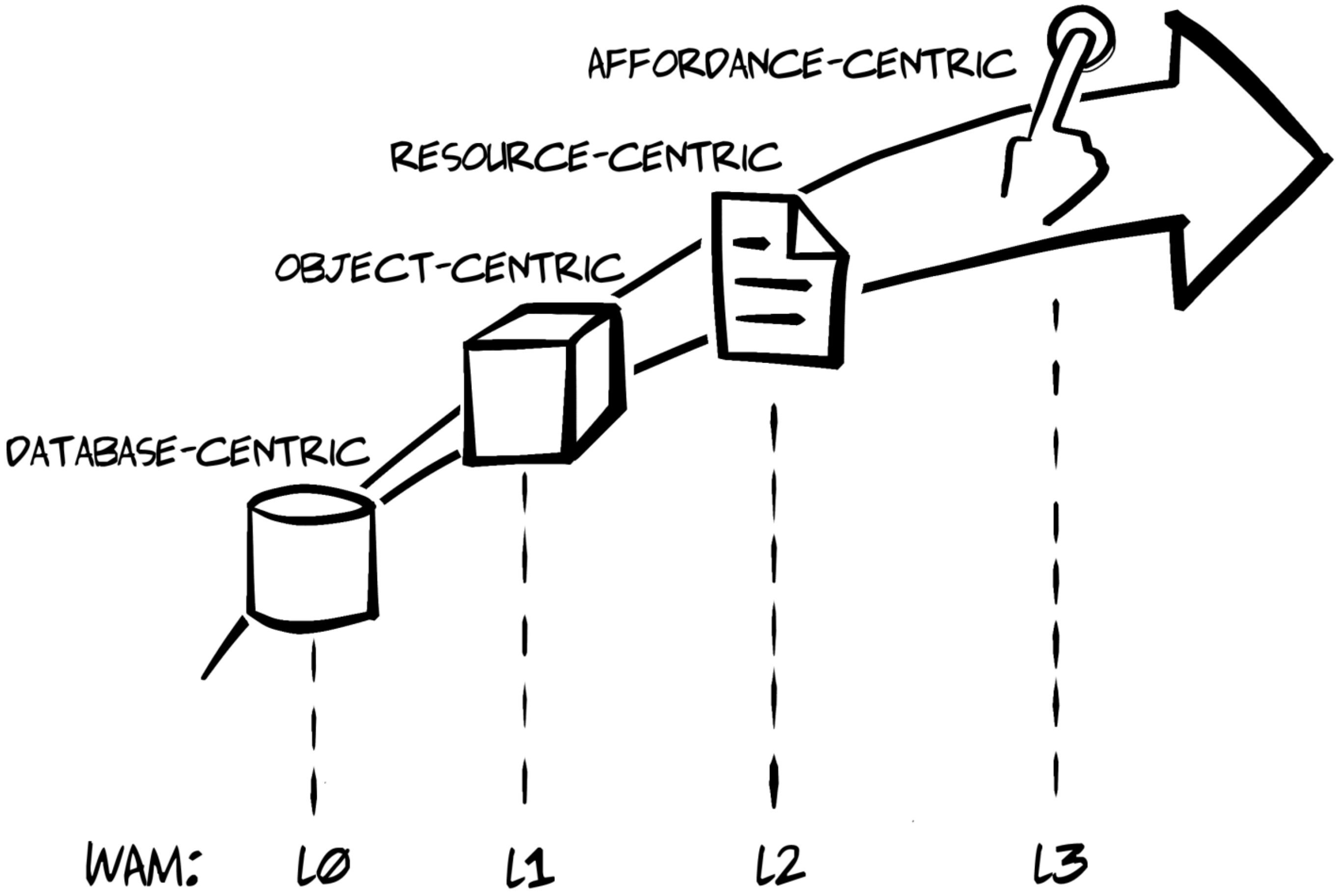
Affordance Model may have:

```
customerSummary
CustomerRead,
CustomerFilter,
CustomerSuspend,
CustomerSearch,
```

etc.



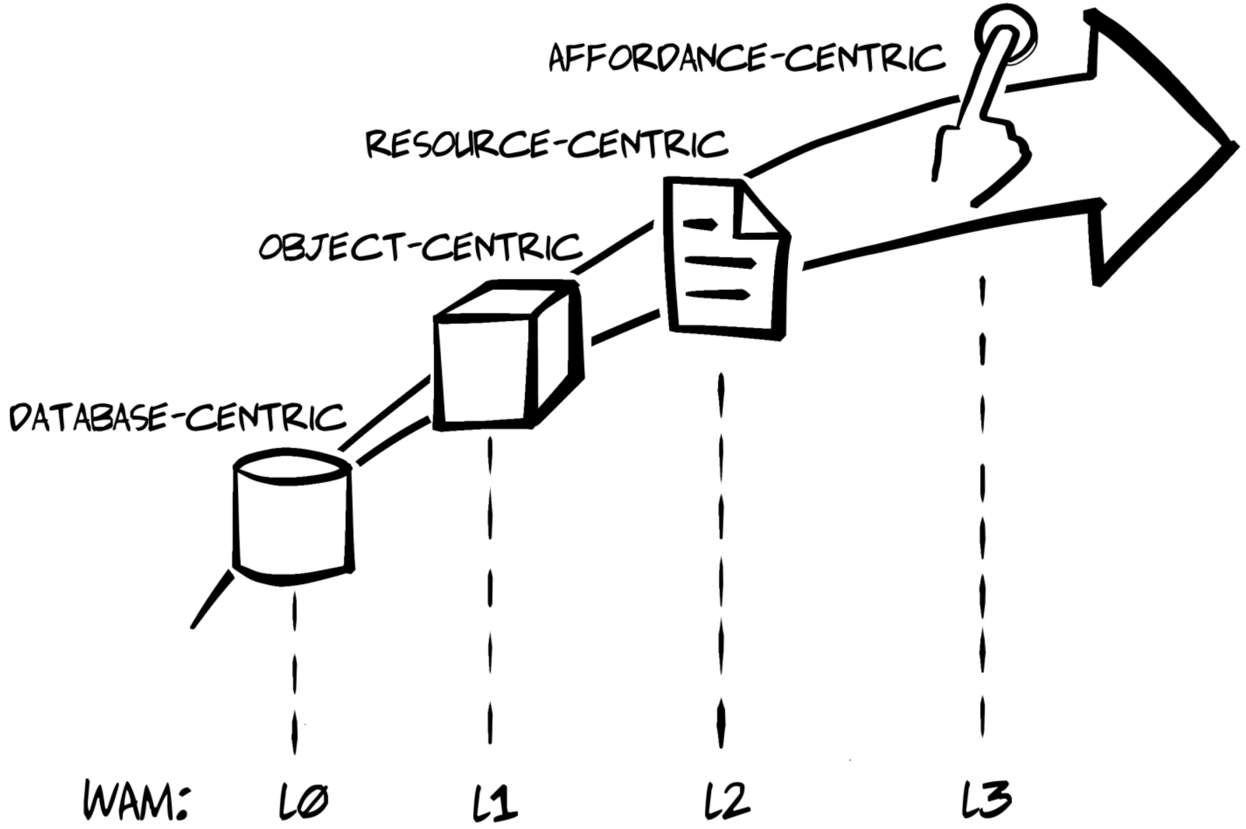
Web API Design Maturity Model



“Your data model is not your object model is not your resource model is not your affordance model.”

Mike Amundsen, 2016

QUESTIONS? COMMENTS?



Web API Design Maturity Model

Mike Amundsen
@mamund
API Academy at CA Technologies