API Design Workshop

Mike Amundsen,
API Academy / CA
@mamund

Introduction


Ctrl
PgUp
Mike Amundsen
@mamund

API Design Training | API

www.apiacademy.co

API ACADEMY

Search API Academy

API Strategy | API Design | API Management | Resources | About

Register | Sign In

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

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

Creating Evolvable Hypermedia Applications

Building

Hypermedia APIs with HTML5 & Node

O'REILLY®

Mike Amundsen

Services for a Changing World

RESTful Web APIs

O'REILLY®

Leonard Richardson,
Mike Amundsen & Sam Ruby

O'REILLY®

Designing APIs for the Web

Mike Amundsen

VIDEO

Part One:
Design Elements of an API

meth·od·ol·o·gy

/ˌmeTHəˈdäləjē/

noun

a system of methods used in a particular area of study or activity.
"a methodology for investigating the concept of focal points"

Actually, we have a methodology already...

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Term	Description
Authority	A URI component that identifies the party with jurisdiction over the namespace defined by the remainder of the URI.
Collection	A resource archetype used to model a server-managed *directory* of resources.
Controller	A resource archetype used to model a procedural concept.
CRUD	An acronym that stands for the four classic storage-oriented functions: create, retrieve, update, and delete.
Developer portal	A Web-based graphical user interface that helps a REST API acquire new clients.
Docroot	A resource that is the hierarchical ancestor of all other resources within a REST API's model. This resource's URI should be the advertised entry point.
Document	A resource archetype used to model a singular concept.
Forward slash separator (/)	Used within the URI path component to separate hierarchically related resources.
Opacity of URIs	An axiom, originally described by Tim Berners-Lee, that governs the visibility of a resource identifier's composition.
Parent resource	The document, collection, or store that governs a given subordinate concept by preceding it within a URI's hierarchical path.
Query	A URI component that comes after the path and before the optional fragment.
Resource archetypes	A set of four intrinsic concepts (document, collection, store, and controller) that may be used to help describe a REST API's model
Store	A resource archetype used to model a client-managed resource repository.
URI path segment	Part of a resource identifier that represents a single node within a larger, hierarchical resource model.
URI template	A resource identifier syntax that includes variables that must be substituted before resolution.

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)

Term	Description
`DELETE`	HTTP request method used to remove its parent.
`GET`	HTTP request method used to retrieve a representation of a resource's state.
`HEAD`	HTTP request method used to retrieve the metadata associated with the resource's state.
`OPTIONS`	HTTP request method used to retrieve metadata that describes a resource's available interactions.
`POST`	HTTP request method used to create a new resource within a collection or execute a controller.
`PUT`	HTTP request method used to insert a new resource into a store or update a mutable resource.
`Request-Line`	RFC 2616 defines its syntax as `Method SP Request-URI SP HTTP-Version CRLF`
Request method	Indicates the desired action to be performed on the request message's identified resource.
Response status code	A three-digit numeric value that is communicated by a server to indicate the result of a client's request.
`Status-Line`	RFC 2616 defines its syntax as: `HTTP-Version SP Status-Code SP Reason-Phrase CRLF`
Tunneling	An abuse of HTTP that masks or misrepresents a message's intent and undermines the protocol's transparency.

Design Guidelines

Craft [good/pretty/usable/stable] URIs

Map domain actions to HTTP methods (CRUD)

Use the proper HTTP Status Codes

Code	Name	Meaning
400	Bad Request	Indicates a nonspecific client error
401	Unauthorized	Sent when the client either provided invalid credentials or forgot to send them
402	Forbidden	Sent to deny access to a protected resource
404	Not Found	Sent when the client tried to interact with a URI that the REST API could not map to a resource
405	Method Not Allowed	Sent when the client tried to interact using an unsupported HTTP method
406	Not Acceptable	Sent when the client tried to request data in an unsupported media type format
409	Conflict	Indicates that the client attempted to violate resource state
412	Precondition Failed	Tells the client that one of its preconditions was not met
415	Unsupported Media Type	Sent when the client submitted data in an unsupported media type format
500	Internal Server Error	Tells the client that the API is having problems of its own

Design Guidelines

Craft [good/pretty/usable/stable] URIs
Map domain actions to HTTP methods (CRUD)
Use the proper HTTP Status Codes
Document serialized objects as HTTP bodies

Term	Description
Field	A named slot with some associated information that is stored in its value.
Form	A structured representation that consists of the fields and links, which are defined by an associated schema.
Format	Describes a form's presentation apart from its schematic.
Link	An actionable reference to a resource.
Link formula	A boolean expression that may serve as HATEOAS calculator's input in order to determine the availability of state-sensitive hypermedia within a form.
Link relation	Describes a connection between two resources.
Schema	Describes a representational form's structure independent of its format.
State fact	A Boolean variable that communicates a condition that is relevant to some state-sensitive hypermedia.

Design Guidelines

Craft [good/pretty/usable/stable] URIs
Map domain actions to HTTP methods (CRUD)
Use the proper HTTP Status Codes
Document serialized objects as HTTP bodies
Use HTTP headers responsibly

Code	Purpose
`Content-Type`	Identifies the entity body's media type
`Content-Length`	The size (in bytes) of the entity body
`Last-Modified`	The date-time of last resource representation's change
`ETag`	Indicates the version of the response message's entity
`Cache-Control`	A TTL-based caching value (in seconds)
`Location`	Provides the URI of a resource

Design Guidelines

Craft [good/pretty/usable/stable] URIs
Map domain actions to HTTP methods (CRUD)
Use the proper HTTP Status Codes
Document serialized objects as HTTP bodies
Use HTTP headers responsibly
Describe edge cases (async, errors, authN/Z)

```
# Response
HTTP/1.1 202 Accepted ❶
Content-Type: application/xml;charset=UTF-8
Content-Location: http://www.example.org/images/task/1
Date: Sun, 13 Sep 2009 01:49:27 GMT

<status xmlns:atom="http://www.w3.org/2005/Atom">
  <state>pending</state>
  <atom:link href="http://www.example.org/images/task/1" rel="self"/>
  <message >Your request has been accepted for processing.</message>
  <ping-after>2009-09-13T01:59:27Z</ping-after> ❷
</status>
```

```
# Response
HTTP/1.1 409 Conflict
Content-Type: application/xml;charset=UTF-8
Content-Language: en
Date: Wed, 14 Oct 2009 10:16:54 GMT
Link: <http://www.example.org/errors/limits.html>;rel="help"

<error xmlns:atom="http://www.w3.org/2005/Atom">
  <message>Account limit exceeded. We cannot complete the transfer due to
  insufficient funds in your accounts</message>
  <error-id>321-553-495</error-id>
  <account-from>urn:example:account:1234</account-from>
  <account-to>urn:example:account:5678</account-to>
  <atom:link href="http://example.org/account/1234"
             rel="http://example.org/rels/transfer/from"/>
  <atom:link href="http://example.org/account/5678"
             rel="http://example.org/rels/transfer/to"/>
</error>
```

```
# Request to obtain a request token
POST /request_token HTTP/1.1 ❶
Host: www.example.org
Authorization: OAuth realm="http://www.example.com/photos",
                     oauth_consumer_key=a1191fd420e0164c2f9aeac32ed35d23,
                     oauth_nonce=109843dea839120a,
                     oauth_signature=d8e19bb988110380a72f6ca33b2ba5903272fe1,
                     oauth_signature_method=HMAC-SHA1,
                     oauth_timestamp=1258308730,
                     oauth_version=1.0 ❷
Content-Length: 0

# Response containing a request token and a secret
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=0e713d524f290676de8aff4073b1bb52e37f065c
  &oauth_token_secret=394bc633d4c93f79aa0539fd554937760f05987c ❸
```

Designing Consistent RESTful Web Service Interfaces

REST API

Design Rulebook

O'REILLY®

Mark Massé

Solutions for Improving Scalability and Simplicity

RESTful Web Services Cookbook

O'REILLY® | YAHOO! PRESS

Subbu Allamaraju

But there's a problem here...

Those are not *design* guidelines..

They are *implementation* guidelines!

im·ple·men·ta·tion

/ˌimpləmənˈtāSHən/

noun

the process of putting a decision or plan into effect; execution.
"she was responsible for the implementation of the plan"

Ok, so what is a *design methodology*, then?

de·sign

/dəˈzīn/

noun

1. a plan or drawing produced to show the look and function or workings of a building, garment, or other object before it is built or made.
   "he has just unveiled his design for the new museum"
   synonyms: plan, blueprint, drawing, sketch, outline, map, plot, diagram, draft, representation, scheme, model More

2. purpose, planning, or intention that exists or is thought to exist behind an action, fact, or material object.
   "the appearance of design in the universe"
   synonyms: intention, aim, purpose, plan, intent, objective, object, goal, end, target; More

Services for a Changing World

RESTful Web APIs

O'REILLY®

Leonard Richardson,
Mike Amundsen & Sam Ruby

Here's a simple seven-step procedure...

Seven-Step API Design Guide

Seven-Step API Design Guide

List Semantic Descriptors

Seven-Step API Design Guide

List Semantic Descriptors
Draw State Diagram

Seven-Step API Design Guide

List Semantic Descriptors
Draw State Diagram
Reconcile Names

Seven-Step API Design Guide

List Semantic Descriptors
Draw State Diagram
Reconcile Names
Produce API Profile

Seven-Step API Design Guide

List Semantic Descriptors
Draw State Diagram
Reconcile Names
Produce API Profile
Choose a Message Model

Seven-Step API Design Guide

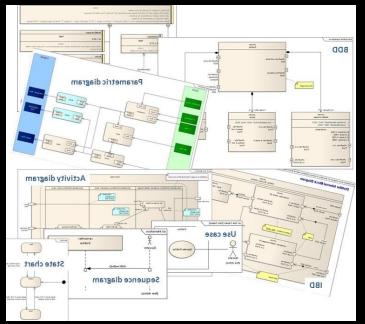
List Semantic Descriptors
Draw State Diagram
Reconcile Names
Produce API Profile
Choose a Message Model
Implement the API!

Seven-Step API Design Guide

List Semantic Descriptors
Draw State Diagram
Reconcile Names
Produce API Profile
Choose a Message Model
Implement the API!
Publish the Billboard URL

Discovery Phase for API Design

Collect Info from Stakeholders
Define Shared Actions and Vocabulary
Iterate until "Done"



Collect Info from Stakeholders

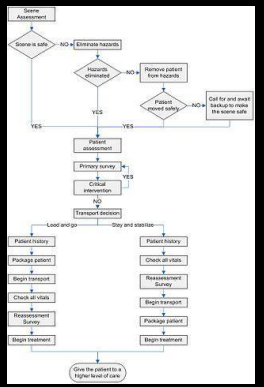
Collect Info from Stakeholders

What to Collect
One-on-One
Group

What to Collect

- Identify data elements
  - names, required, optional, etc.
- Identify actions & data needed for each
- Identify critical paths, required order, etc.
- Recording can help (video/audio)

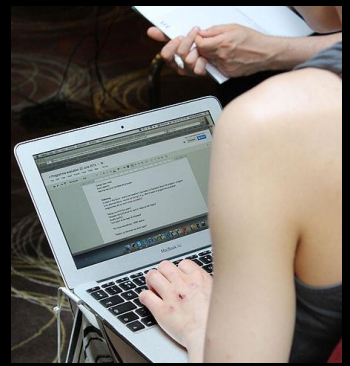
Don't evaluate, just collect



One-on-One

Come armed with questions
Ask person to describe their process/work
Use the "echo technique“

Share your notes

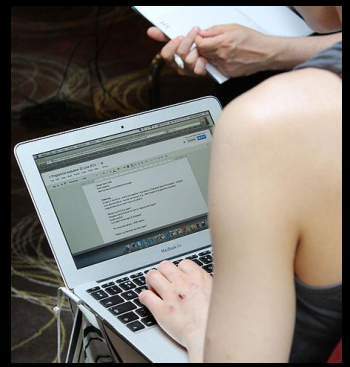


One-on-One

to-do-collect - Notepad

File Edit Format View Help

```
To Do List
==========

We keep track of:
- Action (Finish Q3 Summary, Call Barb in Accounting, etc.)
- Due Date
- Category (Sales, Marketing, Operations, etc.)

We share a single list with the team.
We add items throughout the day
We sort by Due Date to figure out what to do next
We remove them from the list when they are completed
```



One-on-One

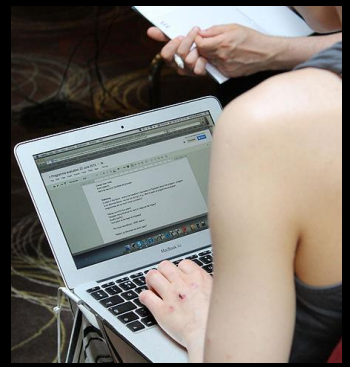
task-mgmt-collection.txt (Windows7_OS (C:) \Users\mamundsen\Dro...Private\2013\2...

File Edit View Search Tools Documents Help

task-mgmt-collection.txt

```
1 Task Management
2
3 Everyone keeps their own list.
4 Add new items throughout the day.
5 Pull up today's tasks, order by priority
6 Mark them completed when done.
7 Print completed report at the end of the day
8
9 Data collected:
10  * name
11  * priority (high, medium, low)
12  * date/time due
13
```

Plain Text | Tab Width: 2 | Ln 1, Col 1 | INS



One-on-One

Punch List

Below is how we manage the weekly roll-out punch list.

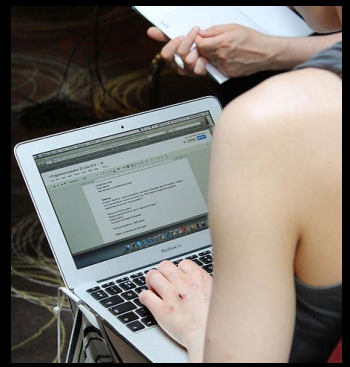
Adding Processes

If you have an item to add to our weekly roll-out, send it to Josh. We can go over it in the daily meetings and, if approved, he can add it to the list.

1. Be sure to include a title, description, which parts are affected and when the work needs to appear in the punch list.
2. Keep in mind that we have "sub-lists", too. It could be you want to add something to those, and not to the primary work process.
3. Before suggesting a new item for the punch list, be sure to go over all existing items in all the sub-lists, too. It's probably already there somewhere!
4. Once approved, you will be responsible to monitoring and policing the item. If you need to work w/ others on the team to get this completed, be sure to include that in the description.
5. If this is a "fail-fast" item, you'll need to be on-hand at the weekly roll-out to make sur eyou can fix whatever breaks. While this may happen the first few times, you should be able to let things run on their own eventually.

Removing Processes

1. If you think we no longer need a process in the rollout, bring it to the group at dailies.
2. If the removal is approved, you'll need to be on-hand at roll-out to make sure all still goes smoothly without the missing step. Once success is confirmed, you do not need to be on-hand every week.

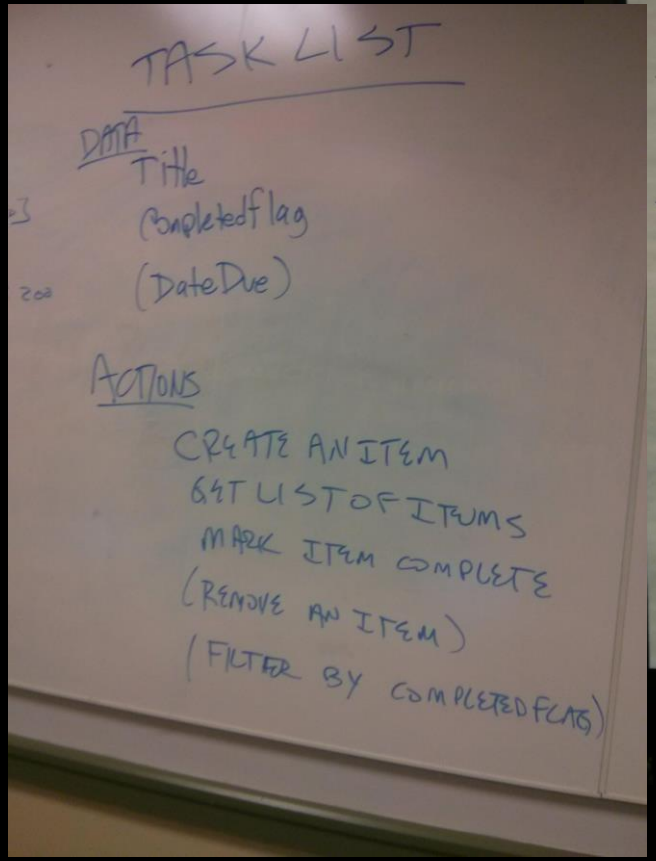


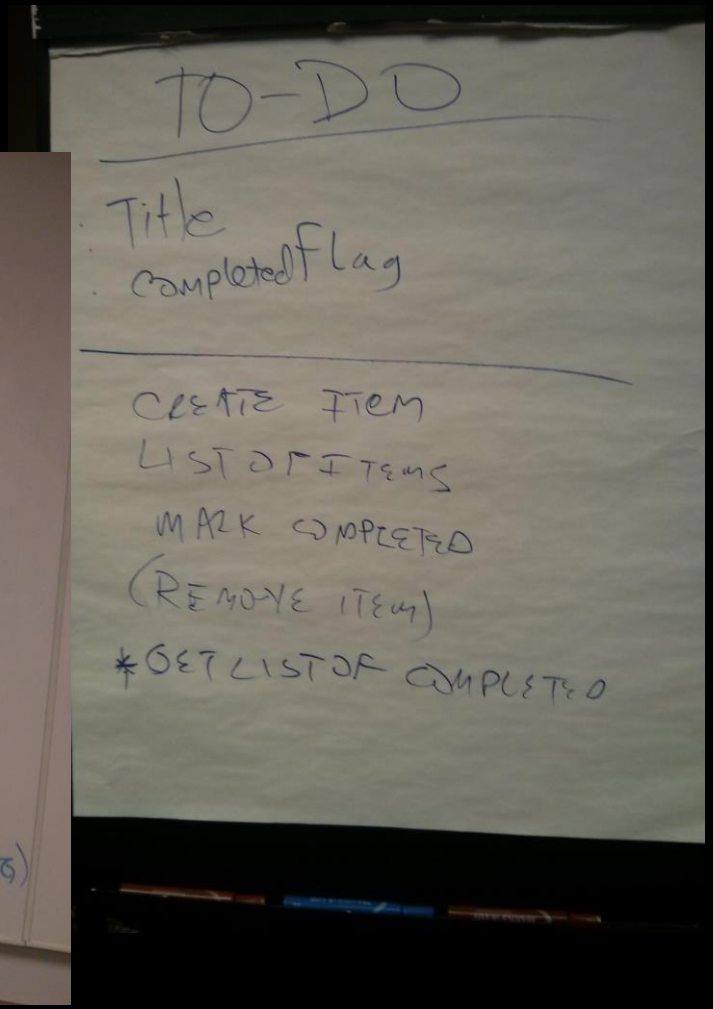
Group

Focus on shared notes from One-on-Ones
Invite discussion, collaboration
Work to distill the information
Moderate, don't edit

Share the results

Group


TASK LIST
DATA
Title
completedflag
(DateDue)
ACTIONS
CREATE AN ITEM
GET LIST OF ITEMS
MARK ITEM COMPLETE
(REMOVE AN ITEM)
(FILTER BY COMPLETED FLAG)


TO-DO
Title
completedflag
CREATE ITEM
LIST OF ITEMS
MARK COMPLETED
(REMOVE ITEM)
*GET LIST OF COMPLETED

Workshops

Turn the tables
Show stakeholders what you've learned
Invite them to amend, adjust your material
Present for comment, don't dictate

Share corrected results

Workshops

Marketing

Marketing has a shared list with categories and removes items when they are completed.

Table 1. Data

Data
[Action]
[DueDate]
[Category]

Table 2. Action

Action
[ViewList]
[FilterList

Sales

In sales everyone keeps their own list with priorities. They mark them completed and produce reports at the end of the day.

Table 3. Data

Data
[Name]
[DateTimeDue]
[Priority]

Table 4. Actions

Action
[ViewList]
[FilterByPrior

Operations

The Operations team has a single, shared list (Punch List) of all the work that is scheduled for the weekly roll-outs. Only one person has access to editing the list and there is a process for getting things approved to adds & removes.

Table 5. Data

Data	Description
[Title]	Thing to do
[Description]	Brief explanation
[AffectedPart]	What will be altered/updated
[ExecutionOrder]	Order in which this is done

Table 6. Actions

Action	Data Elements
[ViewList]	NONE (always in order)

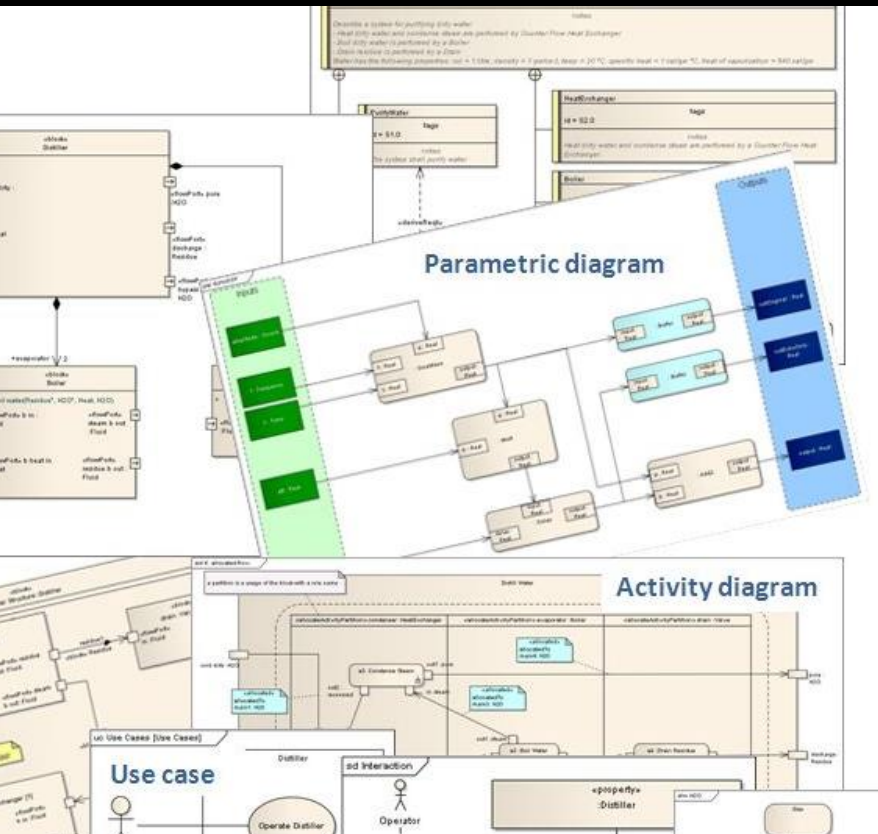
Collect Info from Stakeholders

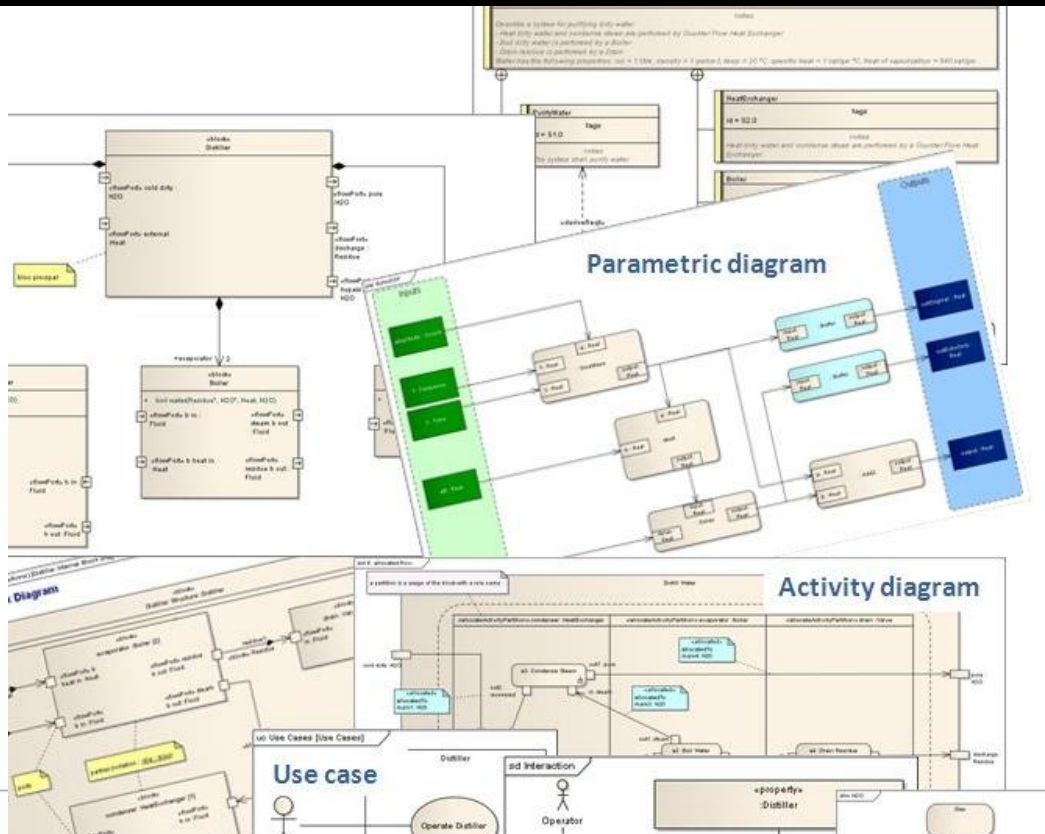
- Collect
  - Don't evaluate!
- One-on-One
  - Use the "echo technique"
- Group
  - Moderate, don't edit

Exercise One:
Produce a List of Use-Cases

Part Two:
Data and Actions

Identify Actions and Vocabulary

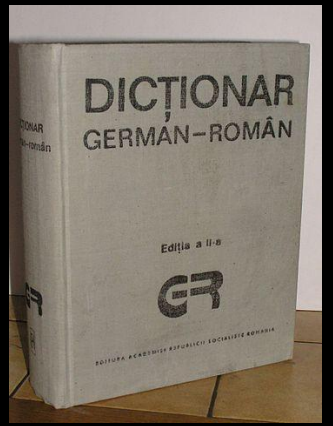




What you collected

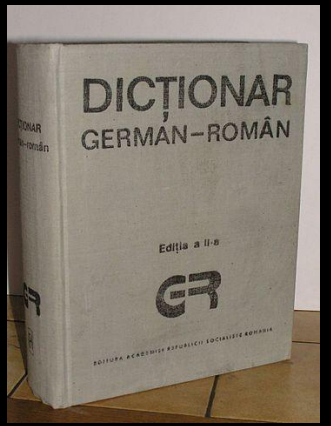
- Data elements
  - names, required, optional, etc.
- Actions & data needed for each
- Critical paths, required order, etc.

Now let's evaluate



Focus on Vocabulary

Work to reconcile names
Don't try to define *objects*
Create a single vocabulary
Link vocabulary words to *categories*



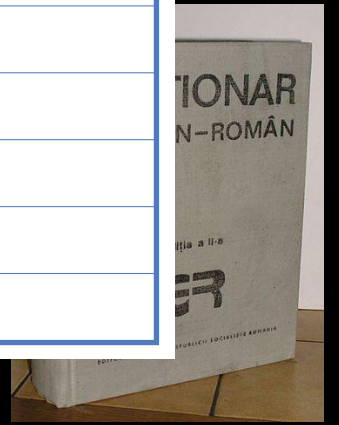
Focus on Vocabulary

Shared Vocabulary

Below is the list of shared vocabulary names for the Task Service.

Table 7. Shared Vocabulary

Data	Description
[Title]	Thing to do
[Description]	Brief explanation
[Category]	Sales Marketing Accounting
[Priority]	High Medium Low
[DateTimeDue]	Date/Time string
[AffectedPart]	What will be altered/updated
[ExecutionOrder]	Order in which this is done



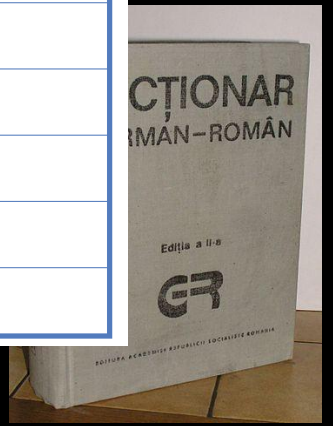
Focus on Vocabulary

Shared Vocabulary with Tags

Below is the list of shared vocabulary names for the Task Service with association tags.

Table 8. Shared Vocabulary

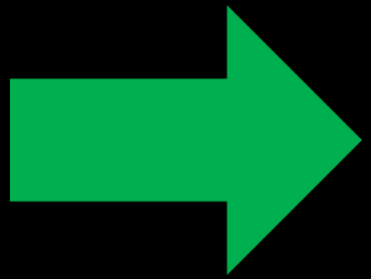
Data	Description	Tag
[Title]	Thing to do	[Marketing] [Sales] [Operations]
[Description]	Brief explanation	[Operations]
[Category]	Sales Marketing Accounting	[Marketing]
[Priority]	High Medium Low	[Sales]
[DateTimeDue]	Date/Time string	[Marketing] [Sales]
[AffectedPart]	What will be altered/updated	[Operations]
[ExecutionOrder]	Order in which this is done	[Operations]



Describe Actions

Work to fully describe actions
Don't try to consolidate too much
Allow for a wide set of actions
Link actions to *categories*

Making actions specific improves the DX/UX over the long term



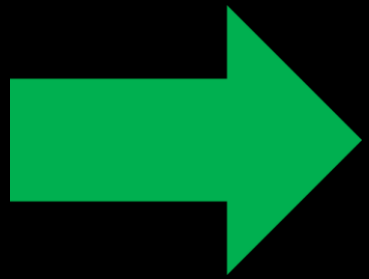
Describe Actions

Task Service Actions with Tags

Below is a list of all the actions currently defined for the Task Service with association Tags.

Table 9. Defined Actions

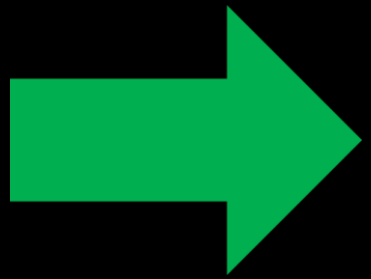
Action	Data Elements	Tag
[ViewList]	NONE	[Marketing] [Sales] [Operations]
[FilterListByTitle]	Title-Fragment	[Marketing]
[FilterListByPriority]	Priority	[Sales]
[Add]	Title Description Priority DateTimeDue AffectedPart ExecutionOrder	[Marketing] [Sales] [Operations]
[Remove]	Title	[Marketing] [Sales] [Operations]
[MarkCompleted]	Title	[Sales]
[ReportCompletedToday]	DateTime	[Sales]



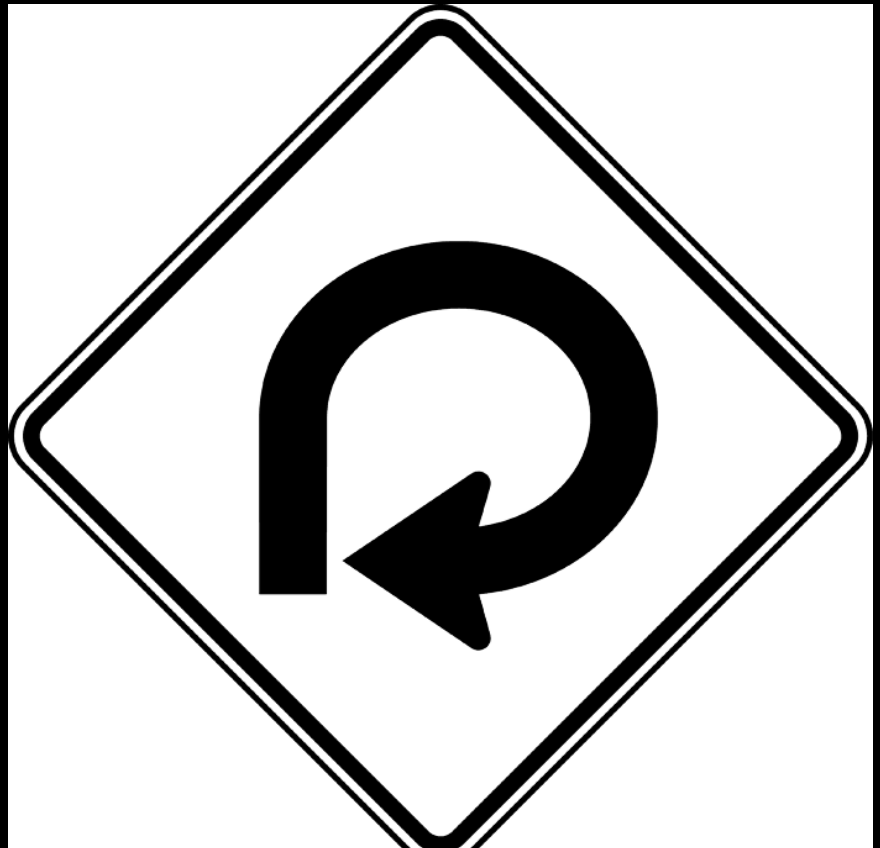
Actions and Vocabulary

Consolidate Vocabulary
Allow for a wide set of actions
Organize by Tags, not Object Hierarchy

Be sure to review w/ stakeholders



Iterate until “Done”



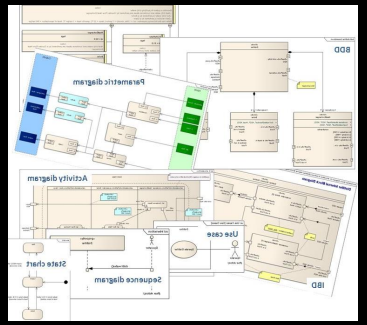
Iterate until "Done"

- Be prepared to iterate on the pattern
  - One-on-One, Group, Workshop
  - Defined Actions and Vocabulary
- Know when you are "Done"
  - Often "close enough" is close enough

Don't get caught in an endless loop

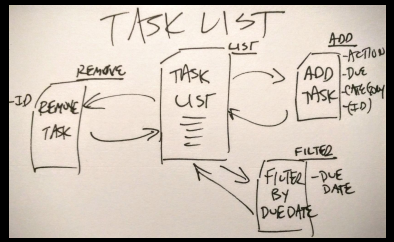
Discovery Phase for API Design

- Collect Info from Stakeholders
  - One-on-One, Group, Workshop
- Define Shared Actions and Vocabulary
  - Consolidate on Vocabulary
  - Support a wide range of Actions
- Iterate until "Done"
  - Aim for 80%



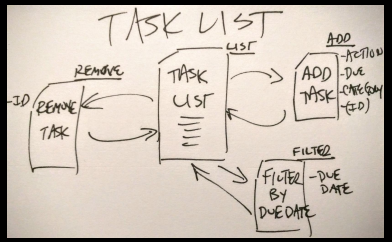
Exercise Two:
Produce List of All Data/Actions

Part Three:
State Diagrams



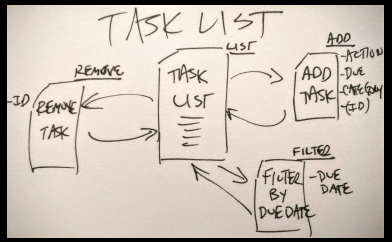
Visualize Your Interactions

Lists are limited to words



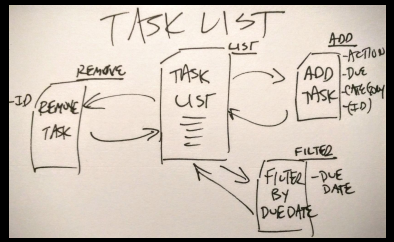
Visualize Your Interactions

Lists are limited to words
80% of communication is non-verbal



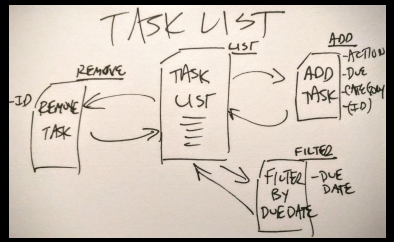
Visualize Your Interactions

Lists are limited to words
80% of communication is non-verbal
Drawing can uncover missing elements



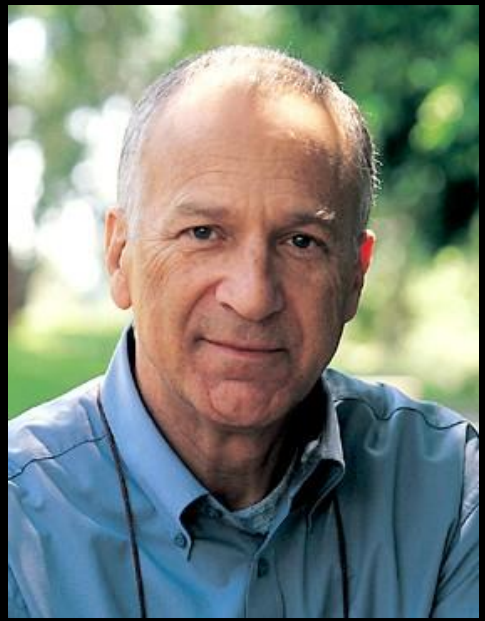
Visualize Your Interactions

Lists are limited to words
80% of communication is non-verbal
Drawing can uncover missing elements
Drawing can be easy to throw away

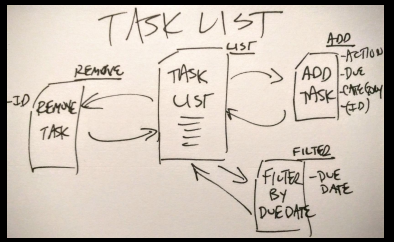


David Harel’s State Charts

“The Harel statechart is equivalent to a state diagram but it improves the readability of the resulting diagram.”

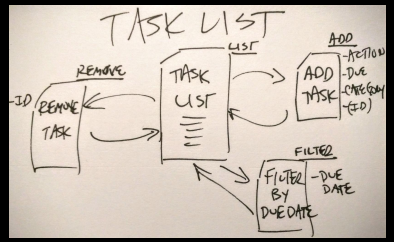


What to include…

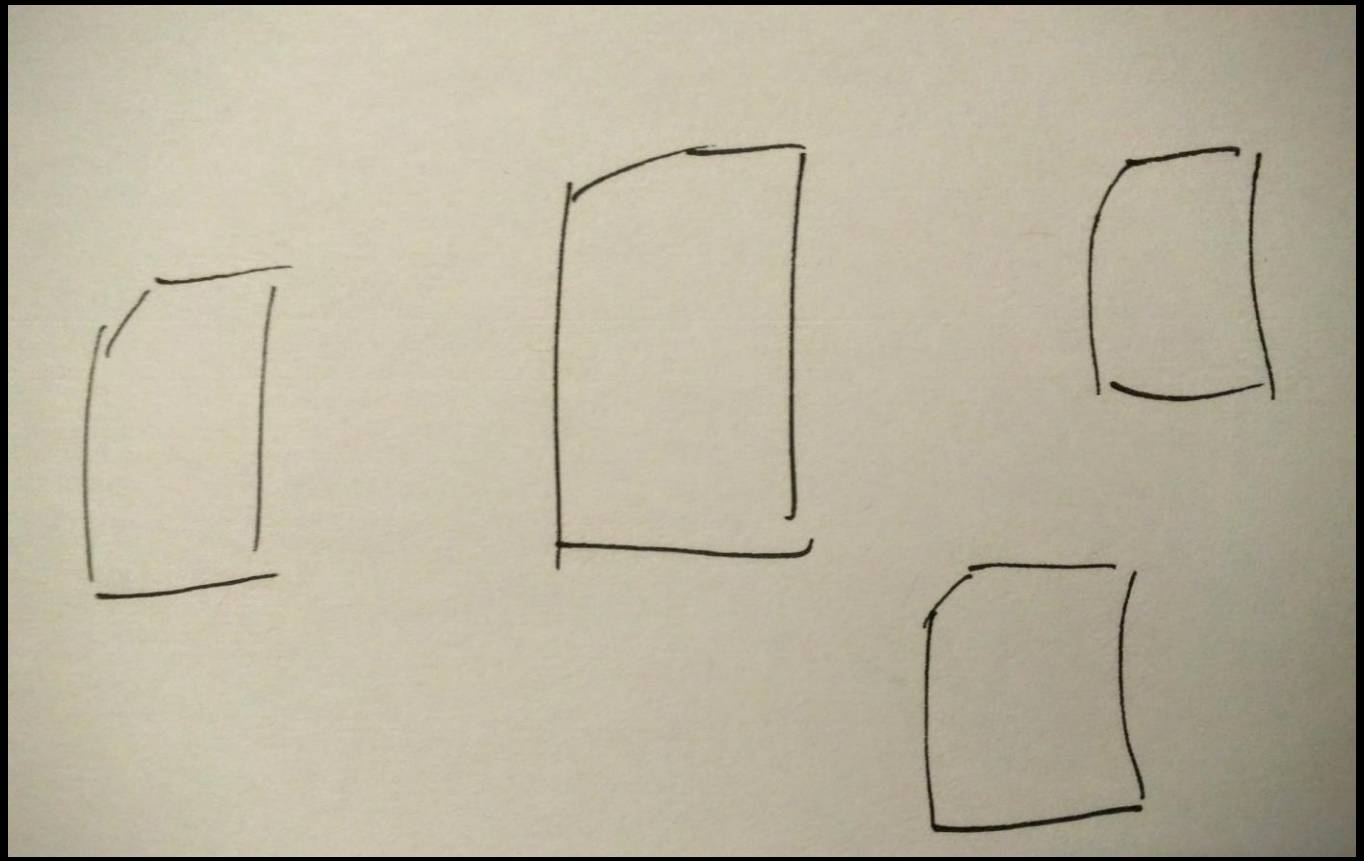


What to include…

Each state is a “thing”

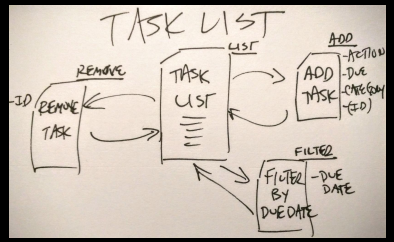


What to Include…

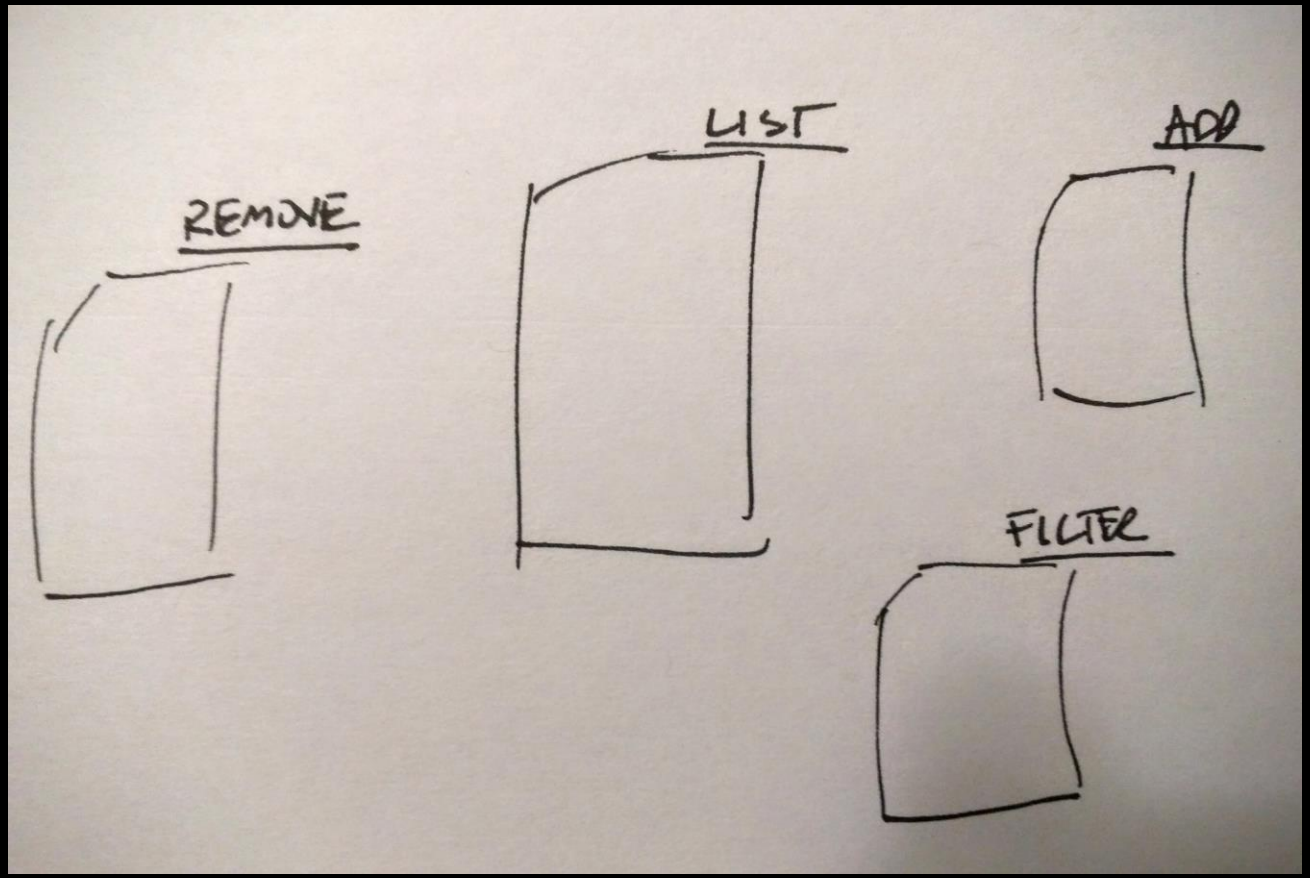


What to include…

Each state is a “thing”
Each state has a “name”

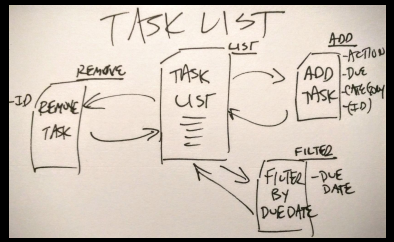


What to Include…

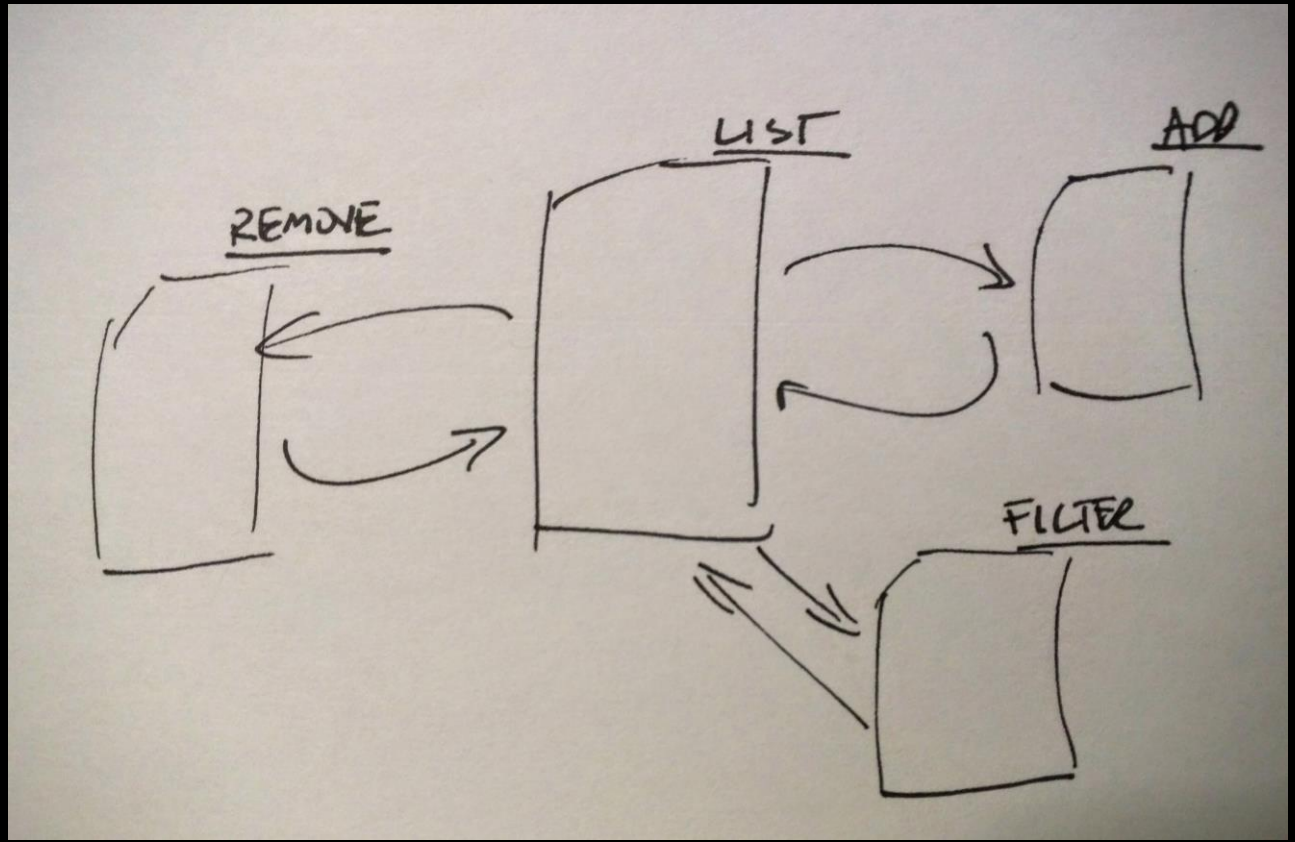


What to include…

Each state is a “thing”
Each state has a “name”
Each state is connected to other states

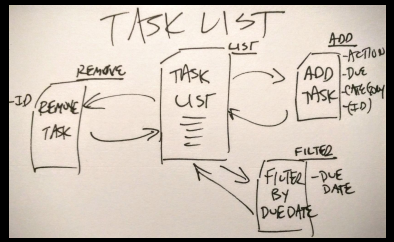


What to Include…

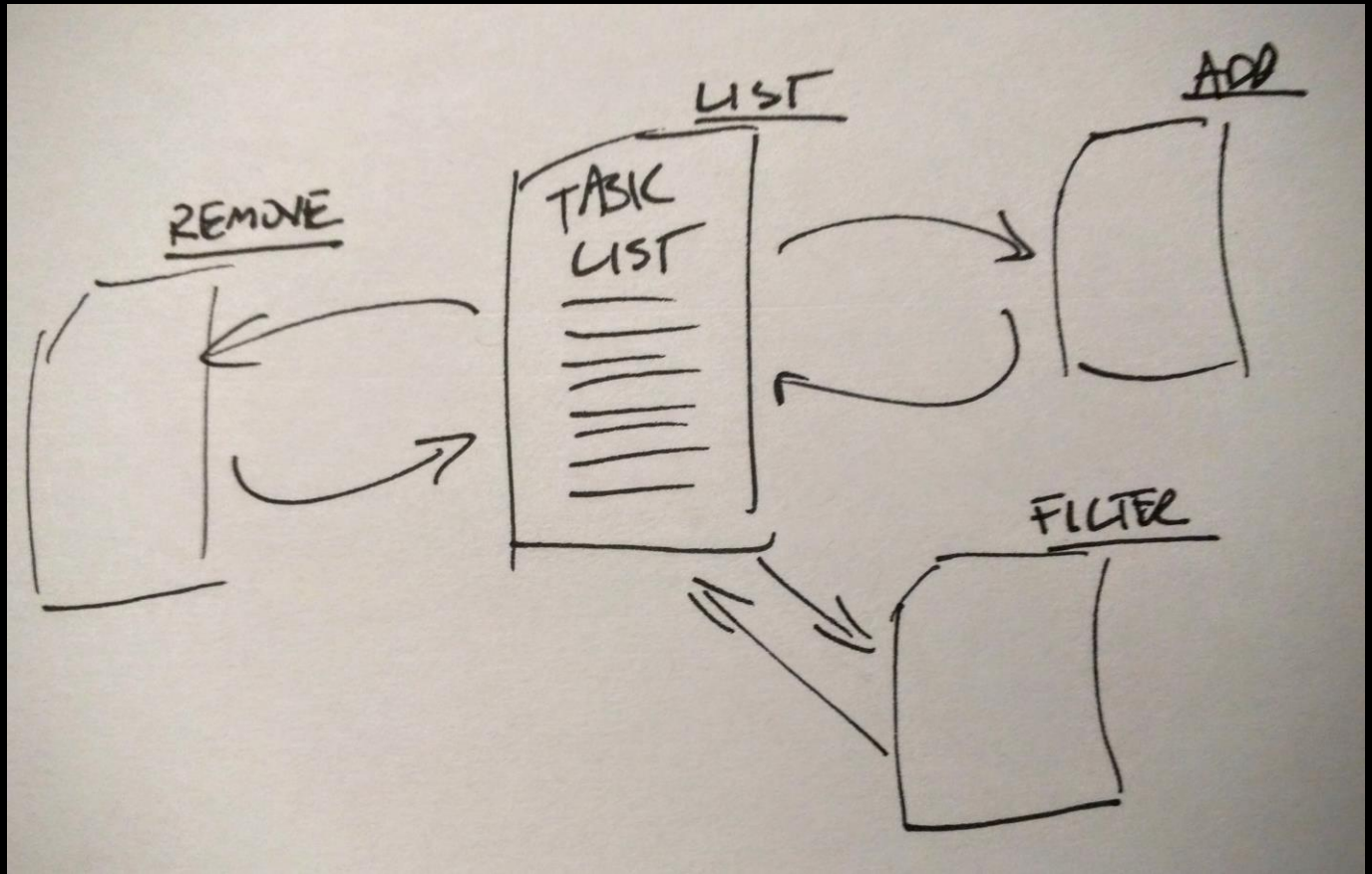


What to include…

Each state is a “thing”
Each state has a “name”
Each state is connected to other states
Each state can have multiple “data points”

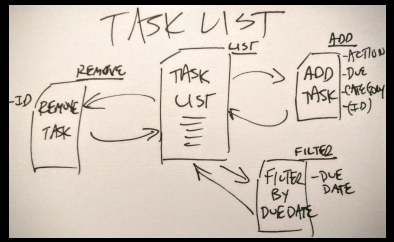


What to Include…

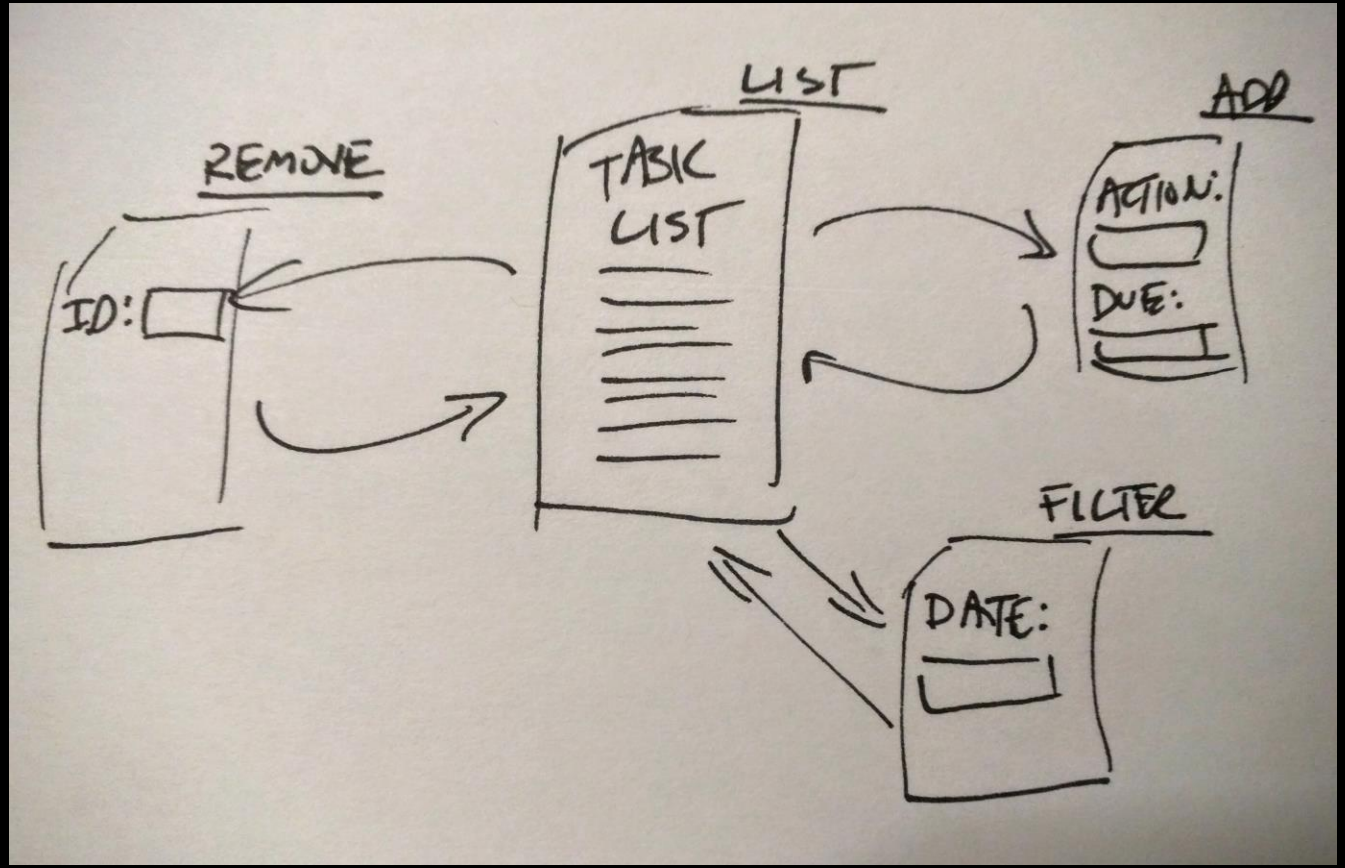


What to include…

Each state is a “thing”
Each state has a “name”
Each state is connected to other states
Each state can have multiple “data points”
Some states handle actions (FORMs)

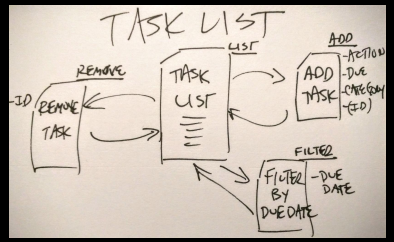


What to Include…

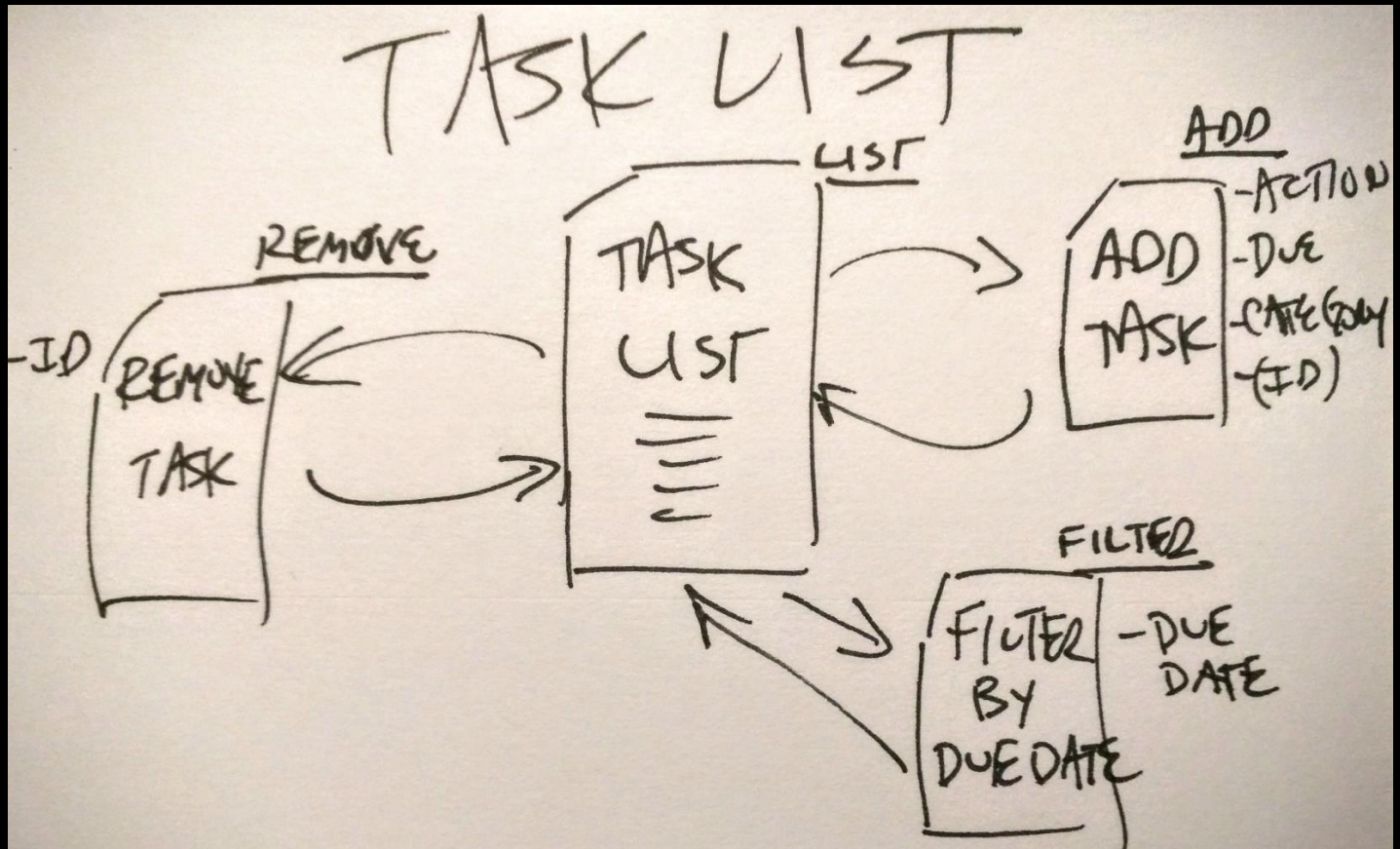


What to include…

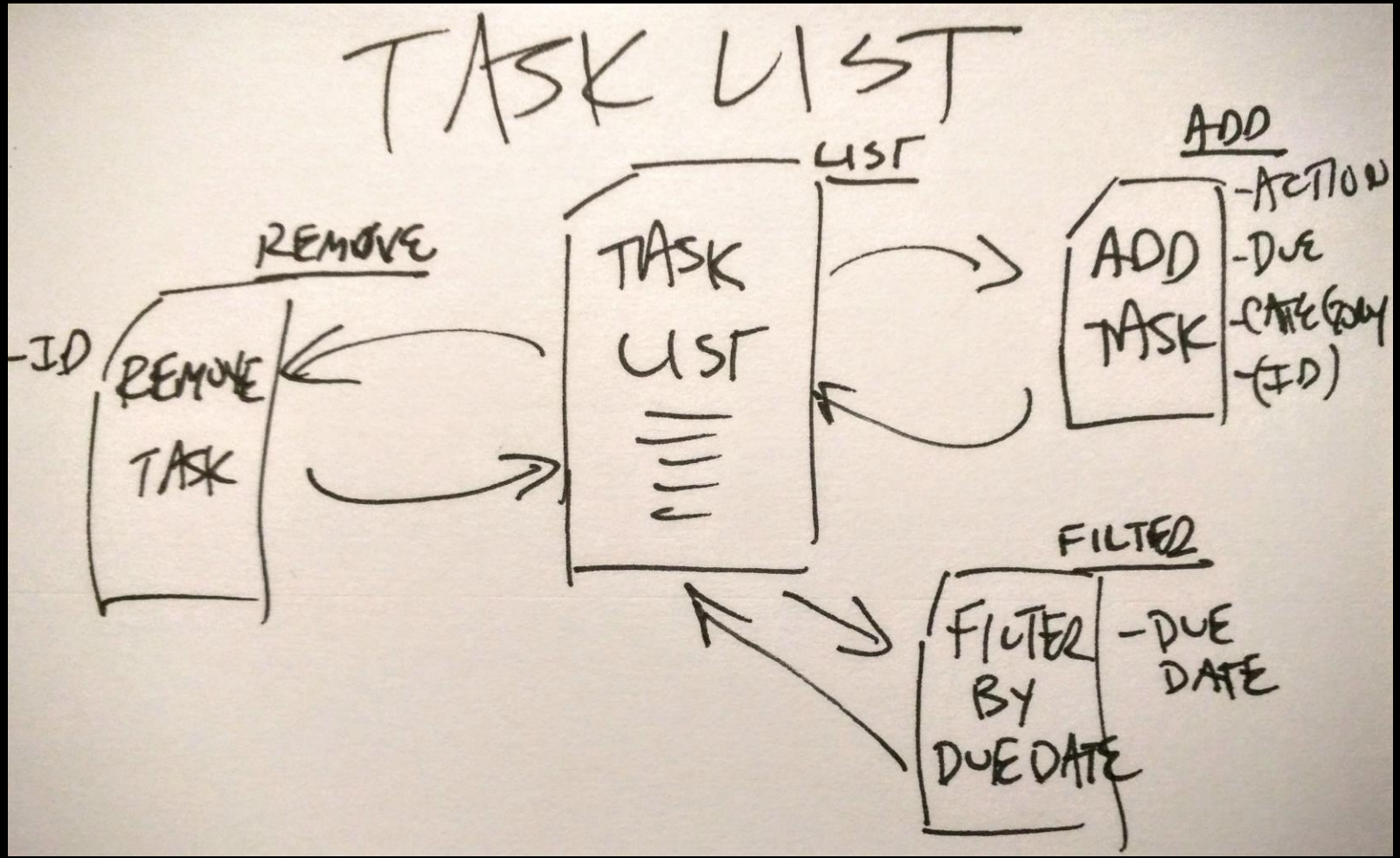
Each state is a “thing”
Each state has a “name”
Each state is connected to other states
Each state can have multiple “data points”
Some states handle actions (FORMs)
Each use-case as a set of “states”



What to Include…



Exercise Three:
Draw Your API Diagram


TASK LIST
LIST
TASK LIST
ADD
-ACTION
ADD TASK
-DUE
-CATEGORY
-(ID)
REMOVE
-ID
REMOVE TASK
FILTER
FILTER BY DUE DATE
-DUE DATE

BREAK

Part Four:
Reconcile Names

Reconcile Names

Interfaces are public
Object models are private
Object models change over time
Interface names rarely change

Roy Fielding on names…

“It is far easier to standardize representation and relation types than it is to standardize objects and object-specific interfaces.”

Roy Fielding

Reconcile Names

Reconcile Names

IANA Link Relation Values

Reconcile Names

Relation Name	Description	Reference
about	Refers to a resource that is the subject of the link's context.	[RFC6903], section 2
alternate	Refers to a substitute for this context	[http://www.w3.org/TR/html5/links.html#link-type-alternate]
appendix	Refers to an appendix.	[http://www.w3.org/TR/1999/REC-html401-19991224]
archives	Refers to a collection of records, documents, or other materials of historical interest.	[http://www.w3.org/TR/2011/WD-html5-20110113/links.html#rel-archives]
author	Refers to the context's author.	[http://www.w3.org/TR/html5/links.html#link-type-author]
bookmark	Gives a permanent link to use for bookmarking purposes.	[http://www.w3.org/TR/html5/links.html#link-type-bookmark]
canonical	Designates the preferred version of a resource (the IRI and its contents).	[RFC6596]
chapter	Refers to a chapter in a collection of resources.	[http://www.w3.org/TR/1999/REC-html401-19991224]
collection	The target IRI points to a resource which represents the collection resource for the context IRI.	[RFC6573]
contents	Refers to a table of contents.	[http://www.w3.org/TR/1999/REC-html401-19991224]
copyright	Refers to a copyright statement that applies to the link's context.	[http://www.w3.org/TR/1999/REC-html401-19991224]
create-form	The target IRI points to a resource where a submission form can be obtained.	[RFC6861]
current	Refers to a resource containing the most recent item(s) in a collection of resources.	[RFC5005]
derivedfrom	The target IRI points to a resource from which this material was derived.	[draft-hoffman-xml2rfc]
describedby	Refers to a resource providing information about the link's context.	[http://www.w3.org/TR/powder-dr/#assoc-linking]
describes	The relationship A 'describes' B asserts that resource A provides a description of resource B. There are no constraints on the format or representation of either A or B, neither are there any further constraints on either resource.	[RFC6892]
disclosure	Refers to a list of patent disclosures made with respect to material for which 'disclosure' relation is specified.	[RFC6579]
duplicate	Refers to a resource whose available representations are	[RFC6249]

Reconcile Names

IANA Link Relation Values
schema.org

Reconcile Names

Property	Expected Type	Description
Properties from Thing		
additionalType	URL	An additional type for the item, typically used for adding more specific types from external vocabularies in microdata syntax. This is a relationship between something and a class that the thing is in. In RDFa syntax, it is better to use the native RDFa syntax - the 'typeof' attribute - for multiple types. Schema.org tools may have only weaker understanding of extra types, in particular those defined externally.
alternateName	Text	An alias for the item.
description	Text	A short description of the item.
image	ImageObject or URL	An image of the item. This can be a URL or a fully described ImageObject.
mainEntityOfPage	URL or CreativeWork	Indicates a page (or other CreativeWork) for which this thing is the main entity being described. See background notes for details. Inverse property: mainEntity.
name	Text	The name of the item.
potentialAction	Action	Indicates a potential Action, which describes an idealized action in which this thing would play an 'object' role.
sameAs	URL	URL of a reference Web page that unambiguously indicates the item's identity. E.g. the URL of the item's Wikipedia page, Freebase page, or official website.
url	URL	URL of the item.

Reconcile Names

IANA Link Relation Values
schema.org
microformats

Reconcile Names

Keyword	Effect on `link`	Effect on `a`, `area`	Brief description (from the relevant specification where possible)	Link to defining specification
acquaintance	not allowed	external relation	the person represented by the current document considers the person represented by the referenced document to be an acquaintance	XFN
alternate	external resource	external relation	Designates substitute versions for the document in which the link occurs. When used together with the `lang` attribute, it implies a translated version of the document. When used together with the `media` attribute, it implies a version designed for a different medium (or media).	HTML4 Link types
appendix	allowed	allowed	Refers to a document serving as an appendix in a collection of documents.	HTML4 Link types
bookmark	not allowed	allowed	Refers to a bookmark. A bookmark is a link to a key entry point within an extended document. The `title` attribute may be used, for example, to label the bookmark. Note that several bookmarks may be defined in each document.	HTML4 Link types
chapter	allowed	allowed	Refers to a document serving as a chapter in a collection of documents.	HTML4 Link types
child	not allowed	external relation	the referenced person is a child of the person represented by the current document	XFN
colleague	not allowed	external relation	the referenced person is a colleague of the person represented by the current document	XFN
contact	not allowed	external relation	the person represented by the current document considers the person represented by the referenced document to be a contact	XFN
contents	allowed	allowed	Refers to a document serving as a table of contents. Some user agents also support the synonym ToC (from "Table of Contents").	HTML4 Link types
copyright	allowed	allowed	Refers to a copyright statement for the current document.	HTML4 Link types
co-resident	not	external	the referenced person lives in the same residence as the person	XFN

Reconcile Names

IANA Link Relation Values
schema.org
microformats
Dublin Core

Reconcile Names

Term Name: contributor	
URI:	http://purl.org/dc/elements/1.1/contributor
Label:	Contributor
Definition:	An entity responsible for making contributions to the resource.
Comment:	Examples of a Contributor include a person, an organization, or a service. Typically, the name of a Contributor should b
Term Name: coverage	
URI:	http://purl.org/dc/elements/1.1/coverage
Label:	Coverage
Definition:	The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which th
Comment:	Spatial topic and spatial applicability may be a named place or a location specified by its geographic coordinates. Temp range. A jurisdiction may be a named administrative entity or a geographic place to which the resource applies. Recom vocabulary such as the Thesaurus of Geographic Names [TGN]. Where appropriate, named places or time periods can b as sets of coordinates or date ranges.
References:	[TGN] http://www.getty.edu/research/tools/vocabulary/tgn/index.html
Term Name: creator	
URI:	http://purl.org/dc/elements/1.1/creator
Label:	Creator
Definition:	An entity primarily responsible for making the resource.
Comment:	Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used
Term Name: date	
URI:	http://purl.org/dc/elements/1.1/date

Reconcile Names

IANA Link Relation Values
schema.org
microformats
Dublin Core
Activity Streams

Reconcile Names

Class	Description	
Object	URI:	`http://www.w3.org/ns/activitystreams#Object`
	Notes:	Describes an object of any kind. The Object class serves as the base class for most of the other kinds of objects defined in the Activity Vocabulary, include other Core classes such as `Activity`, `IntransitiveActivity`, `Actor`, `Collection` and `OrderedCollection`.
	Disjoint With:	`Link`
	Properties:	`alias` | `attachment` | `attributedTo` | `content` | `context` | `displayName` | `endTime` | `generator` | `icon` | `image` | `inReplyTo` | `location` | `preview` | `published` | `replies` | `scope` | `startTime` | `summary` | `tag` | `title` | `updated` | `url` | `to` | `bto` | `cc` | `bcc`
Link	URI:	`http://www.w3.org/ns/activitystreams#Link`
	Notes:	A Link is an indirect, qualified reference to a resource identified by a URL. The fundamental model for links is established by [RFC5988]. Many of the properties defined by the Activity Vocabulary allow values that are either instances of `Object` or `Link`. When a `Link` is used, it establishes a qualified relation connecting the subject (the containing object) to the resource identified by the `href`.
	Disjoint With:	`Object`

Reconcile Names

IANA Link Relation Values
schema.org
microformats
Dublin Core
Activity Streams

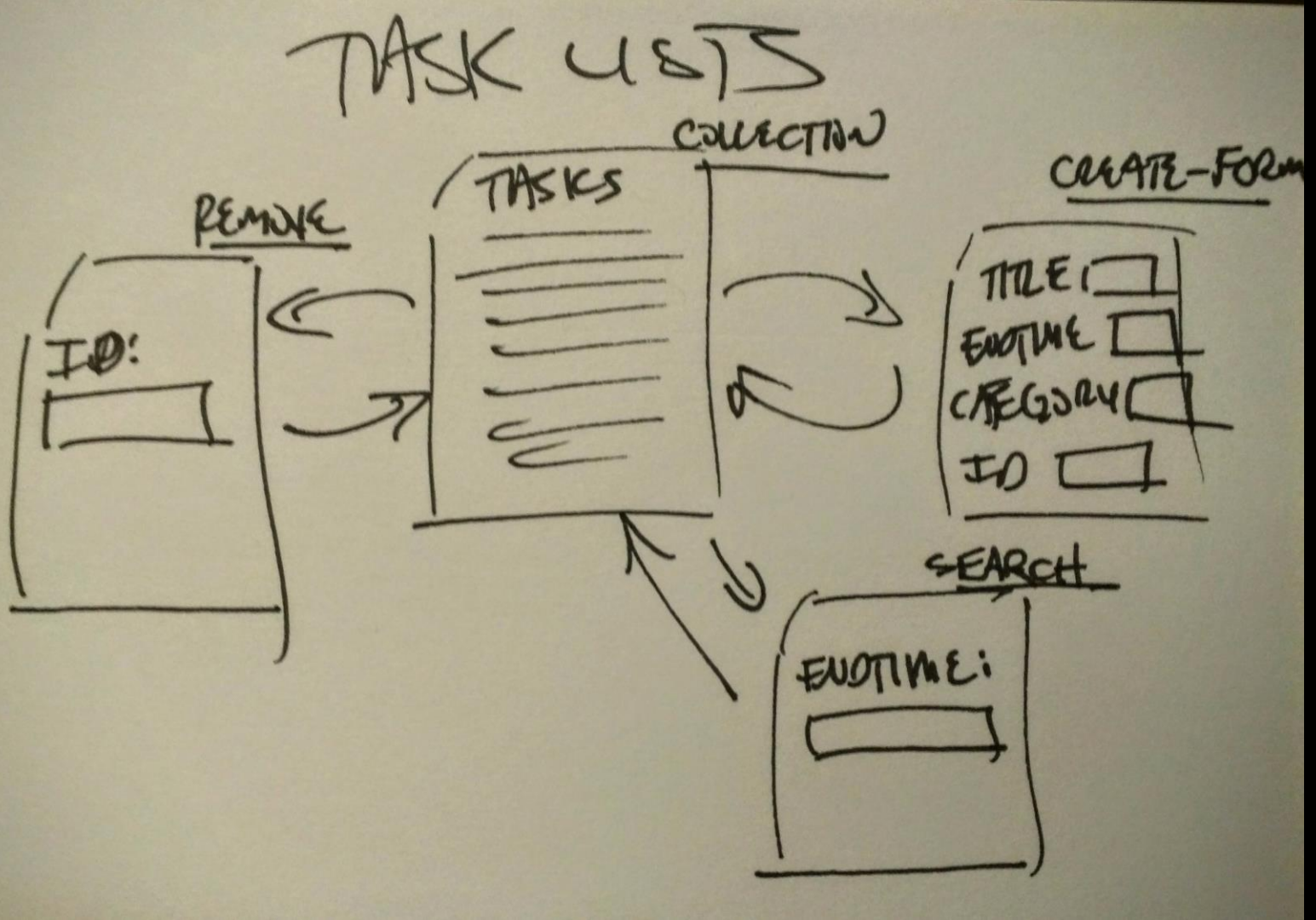
Review/Revise Interface Names

ID
Action
DueDate
Category
List
Filter
Add
Remove

Review/Revise Interface Names

ID (`http://example.org/rels/id` -- RFC5988)
Action (`title` – Activity Streams)
DueDate (`endTime` – schema.org)
Category (`category` – microformats)
List (`collection` – IANA)
Filter (`search` – IANA)
Add (`create-form` – IANA)
Remove (`http://example.org/rels/remove` -- RFC5988

Exercise Four:
Reconcile your interface names


TASK UIS
COLLECTION
TASKS
REMOVE
ID:
CREATE-FORM
TITLE:
ENDTIME
CATEGORY
ID
SEARCH
ENDTIME:

Part Five:
Write an API Profile

Describing Your API

Web interfaces are…

- Loosely-Typed
- Varying URLs
- Varying Inputs
- Varying Workflow

A Brief History of Web Interface Description Languages (IDLs)

- How far does this go back?
  - 2001
- How many do we currently have?
  - a dozen so far!

WSDL (2001)

- Web Service Definition Language
- Specific to SOAP Services

```
<?xml version="1.0"?>

<!-- root element wsdl:definitions defines set of rel
<wsdl:definitions name="EndorsementSearch"
  targetNamespace="http://namespaces.snowboard-info.c
  xmlns:es="http://www.snowboard-info.com/Endorsement
  xmlns:esxsd="http://schemas.snowboard-info.com/Endo
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

  <!-- wsdl:types encapsulates schema definitions of
  <wsdl:types>

    <!-- all type declarations are in a chunk of xsd
    <xsd:schema targetNamespace="http://namespaces.sn
      xmlns:xsd="http://www.w3.org/1999/XMLSchema">

      <!-- xsd definition: GetEndorsingBoarder [manuf
      <xsd:element name="GetEndorsingBoarder">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="manufacturer" type="st
            <xsd:element name="model" type="string"/>
```

AtomSvc

- Atom Service Document (2007)
- Specific to Atom format/services

```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://www.w3.org/2007/app"
         xmlns:atom="http://www.w3.org/2005/Atom">
 <workspace>
   <atom:title>Main Site</atom:title>

   <collection href="http://example.org/reilly/main" >
     <atom:title>My Blog Entries</atom:title>
     <categories href="http://example.com/cats/forMain.c
   </collection>

   <collection href="http://example.org/reilly/pic" >
     <atom:title>Pictures</atom:title>
     <accept>image/png</accept>
     <accept>image/jpeg</accept>
   </collection>
 </workspace>

 <workspace>
   <atom:title>Side Bar Blog</atom:title>
```

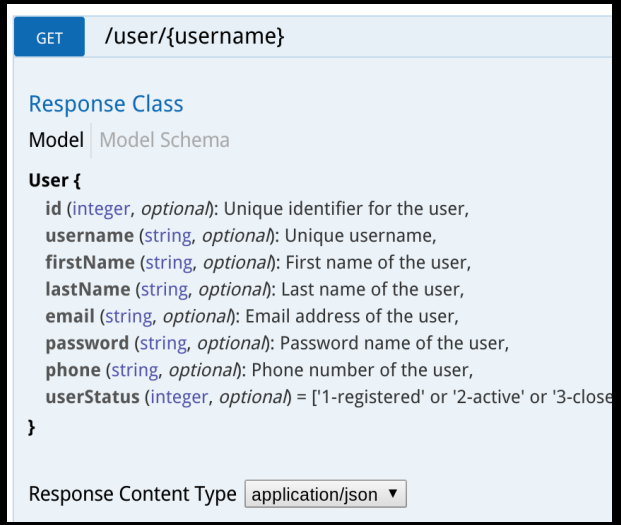
WADL (2009)

- Web Application Description Language
- For "HTTP-based" Services

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
    <doc xmlns:jersey="http://jersey.dev.java.net/"
                jersey:generatedBy="Jersey: 1.0-ea-SNAPSHOT
    <resources base="http://localhost:9998/storage/">
        <resource path="/containers">
            <method name="GET" id="getContainers">
                <response>
                    <representation mediaType="application/x
                </response>
            </method>
            <resource path="{container}">
                <param xmlns:xs="http://www.w3.org/2001/XMLS

                <method name="PUT" id="putContainer">
                    <response>
                        <representation mediaType="applicati
                    </response>
                </method>
                <method name="DELETE" id="deleteContainer"/>
                <method name="GET" id="getContainer">
```

Swagger (2009)

- Swagger is a JSON format from Wordnik
- Support both machine and human descriptions



RESTDesc (2010)

- Semantic descriptions for Hypermedia APIs
- Based on RDF

```
@prefix : <http://example.org/image#>.
@prefix http: <http://www.w3.org/2011/http#>.
@prefix dbpedia: <http://dbpedia.org/resource/>.
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>.
{
  ?image :smallThumbnail ?thumbnail.
}
=>
{
  _:request http:methodName "GET";
            http:requestURI ?thumbnail;
            http:resp [ http:body ?thumbnail ].
  ?image dbpedia-owl:thumbnail ?thumbnail.
  ?thumbnail a dbpedia:Image;
             dbpedia-owl:height 80.0.
}.
```

ioDocs (201?)

- Runtime description/managment
- JSON format

API Blueprint (2012)

- Describe API and stand up proxy
- based on MarkDown

```
--
REST Hello
Use the machine-readable instructions to
--

GET /hello/
> accept-type: application/xml
> Authorization: Basic 1324325364487
< 200
<hello>
    <greetings>Mike</greetings>
    <say href="/hello/" method="get">

        <data id="name" value="" />
    </say>
    <add href="/hello/" method="post">
        <data id="name" value="" />
    </add>
</hello>
```

Google Discovery Document (2012)

- The Discovery Document describes the surface for a particular version of an API.

```
{
  "kind": "discovery#restDescription",
  "discoveryVersion": "v1",
  "id": {string},
  "name": {string},
  "version": {string},
  "revision": {string},
  "title": {string},
  "description": {string},
  "icons": {
    "x16": {string},
    "x32": {string}
  },
  "documentationLink": {string},
  "labels": [
    {string}
  ],
  "protocol": "rest",
  "baseUrl": {string},
  "basePath": {string},
  "rootUrl": {string},
  "servicePath": {string},
  "batchPath": "batch",
  "parameters": {
    {(key)}: {
      "id": {string},
```

JSON Home (2013)

- A Discovery format
- Based on JSON

```
{
  "resources": {
    "http://example.org/rel/widgets": {
      "href": "/widgets/"
    },
    "http://example.org/rel/widget": {
      "href-template": "/widgets/{widget_id}",
      "href-vars": {
        "widget_id": "http://example.org/param/wi
      },
      "hints": {
        "allow": ["GET", "PUT", "DELETE", "PATCH"
        "formats": {
          "application/json": {}
        },
        "accept-patch": ["application/json-patch"
        "accept-post": ["application/xml"],
        "accept-ranges": ["bytes"]
      }
    }
  }
```

JSON Hyperschema (2013)

- Based on JSON Schema and JSON Path
- Primarily a design-time description

```
            }
        }
    },
    "required" : ["id", "title", "authorId"],
    "links": [
        {
            "rel": "full",
            "href": "{id}"
        },
        {
            "rel": "author",
            "href": "/user?id={authorId}"
        }
    ]
```

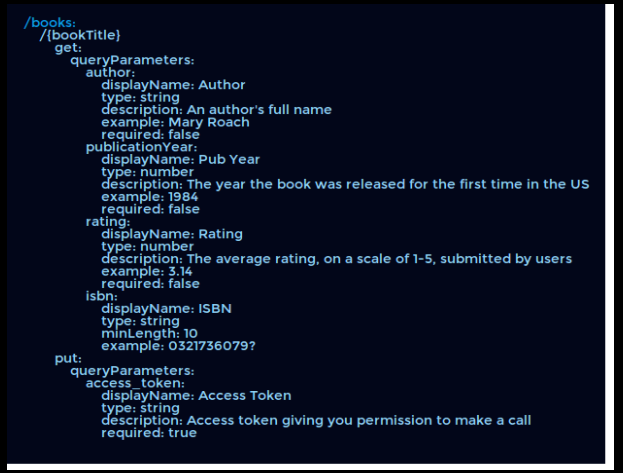
RSDL (2013)

- RESTful Service Description Language
- Resource-oriented descriptions

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="rsdl2ht
<service name="Documents" identity-provider-re
  xmlns="http://identifiers.emc.com/rsdl"
  xmlns:html="http://www.w3.org/1999/xhtml/">

  <documentation> This REST service provides a
    created, modified, or deleted. </documenta

  <start ref="res-home"/>

  <media-types>
    <media-type id="med-document" name="applic
      <documentation> The media type for the s
      <description href="example.com/mediatype
    </media-type>
    <media-type id="med-home-xml" name="applic
```

RAML (2013)

- RESTful API Modeling Language
- "It's a way of describing practically-RESTful APIs in a way that's highly readable by both humans and computers."



```
/books:
  /{bookTitle}
    get:
      queryParameters:
        author:
          displayName: Author
          type: string
          description: An author's full name
          example: Mary Roach
          required: false
        publicationYear:
          displayName: Pub Year
          type: number
          description: The year the book was released for the first time in the US
          example: 1984
          required: false
        rating:
          displayName: Rating
          type: number
          description: The average rating, on a scale of 1-5, submitted by users
          example: 3.14
          required: false
        isbn:
          displayName: ISBN
          type: string
          minLength: 10
          example: 0321736079?
    put:
      queryParameters:
        access_token:
          displayName: Access Token
          type: string
          description: Access token giving you permission to make a call
          required: true
```

A Brief History of Web IDLs

- Why so many?
- Are we missing something?
- Is there another way to do this?

What-if...

- We could describe a class of services, not just an instance?
- We could describe the service independent of media type?
- We could describe the service independent of protocol (HTTP, XMPP, CoAP, etc.)
- We could describe the service without describing the workflow?

What-If..

- We could describe affordances instead of resources?
- We could define vocabularies instead of object trees?

- What would be possible then?

We could..

- Allow implementers to select their preferred media type(s)
- Allow implementers to select their preferred protocol(s)
- Allow implementers to establish their own workflow

We could...

- Allow servers to own their own URL space
- Allow clients to focus on the hypermedia and not the URL

We could...

- Focus on a shared vocabulary on the outside without modifying our storage, logical, or business model on the inside.

And clients and servers can still inter-operate.


AIPS
Application-Level
Profile Semantics

Network Working Group	M. Amundsen
Internet-Draft	Layer 7 Technologies
Expires: September 19, 2013	L. Richardson
	March 18, 2013

TOC

Application-Level Profile Semantics (ALPS) draft-amundsen-richardson-alps-00

Abstract

This document describes ALPS, a data format for defining simple descriptions of application-level semantics, similar in complexity to HTML microformats. An ALPS document can be used as a profile to explain the application semantics of a document with an application-agnostic media type (such as HTML, HAL, Collection+JSON, or Siren). This increases the reusability of profile documents across media types.

```
<alps version="1.0">
    <doc format="text">
        A list of contacts that also supports search
    </doc>

    <!-- a hypermedia control for returning contacts -->
    <descriptor id="search" type="safe" rt="contact">
        <doc>
            Simple hypermedia control for getting a list of contacts
        </doc>
        <descriptor id="name" type="semantic">
            <doc>
                Input for search form
            </doc>
        </descriptor>
    </descriptor>

    <!--  a contact: one or more of these may be returned -->
    <descriptor id="contact" type="semantic">
        <doc>
            Individual Contact
        </doc>
        <descriptor id="link" type="safe">
            <doc>
                Link to individual contact
            </doc>
        </descriptor>
        <descriptor id="givenName" type="semantic" href="http://schema.org/givenName"/>
        <descriptor id="familyName" type="semantic" href="http://schema.org/familyName" />
        <descriptor id="email" type="semantic" href="http://schema.org/email" />
        <descriptor id="telephone" type="semantic" href="http://schema.org/telephone"/>
    </descriptor>
</alps>
```

Exercise Five:
Create an ALPS Document

```
<alps version="1.0">
  <doc>A simple task management interface</doc>
  <!-- data -->
  <descriptor id="id" type="semantic"/>
  <descriptor id="title" type="semantic" />
  <descriptor id="endTime" type="semantic" />
  <descriptor id="category" type="semantic" />
  <!-- actions -->
  <descriptor id="collection" type="safe" />
  <descriptor id="create-form" type="unsafe">
    <descriptor href="#title" />
    <descriptor href="#endTime" />
    <descriptor href="#category" />
  </descriptor>
  <descriptor id="search" type="safe">
    <descriptor href="#endTime" />
  </descriptor>
  <descriptor id="remove" type="unsafe">
    <descriptor href="#id" />
  <descriptor>
</alps>
```

Part Six:
Choose a Message Model

Message Models Power the Web

On the Web

we pass ***messages***

Not **objects**

Message Models Power the Web

We standardize message models:

HTML
CSV
JPG
XML
JSON

Message Models Power the Web

Once you determine your application interface,

you can select one or more message models.

Media Types

Last Updated

2014-06-10

Registration Procedure(s)

Expert Review for Vendor and Personal Trees.

Expert(s)

Ned Freed, primary; Mark Baker, secondary; Bjoern Hoehrmann, secondary

Reference

[RFC6838][RFC4855]

Note

Per Section 3.1 of [RFC6838], Standards Tree requests made through IETF documents will be reviewed and approved by the IESG, while requests made by other recognized standards organizations will be reviewed by the Designated Expert in accordance with the Specification Required policy. IANA will verify that this organization is recognized as a standards organization by the IESG.

Note

[RFC2046] specifies that Media Types (formerly known as MIME types) and Media Subtypes will be assigned and listed by the IANA.

Media Types

www.iana.org/assignments/media-types/media-types.xhtml

Name	Template	Reference
1d-interleaved-parityfec	application/1d-interleaved-parityfec	[RFC6015]
3gpdash-qoe-report+xml	application/3gpdash-qoe-report+xml	[Ozgur_Oyman][ThreeGPP]
3gpp-ims+xml	application/3gpp-ims+xml	[John_M_Meredith]
activemessage	application/activemessage	[Ehud_Shapiro]
activemessage	application/activemessage	[Ehud_Shapiro]
alto-costmap+json	application/alto-costmap+json	[RFC-ietf-alto-protocol-27]
alto-costmapfilter+json	application/alto-costmapfilter+json	[RFC-ietf-alto-protocol-27]
alto-directory+json	application/alto-directory+json	[RFC-ietf-alto-protocol-27]
alto-endpointprop+json	application/alto-endpointprop+json	[RFC-ietf-alto-protocol-27]
alto-endpointpropparams+json	application/alto-endpointpropparams+json	[RFC-ietf-alto-protocol-27]
alto-endpointcost+json	application/alto-endpointcost+json	[RFC-ietf-alto-protocol-27]
alto-endpointcostparams+json	application/alto-endpointcostparams+json	[RFC-ietf-alto-protocol-27]
alto-error+json	application/alto-error+json	[RFC-ietf-alto-protocol-27]
alto-networkmapfilter+json	application/alto-networkmapfilter+json	[RFC-ietf-alto-protocol-27]
alto-networkmap+json	application/alto-networkmap+json	[RFC-ietf-alto-protocol-27]
andrew-inset	application/andrew-inset	[Nathaniel_Borenstein]
applefile	application/applefile	[Patrik_Faltstrom]
atom+xml	application/atom+xml	[RFC4287][RFC5023]
atomcat+xml	application/atomcat+xml	[RFC5023]
atomdeleted+xml	application/atomdeleted+xml	[RFC6721]
atomicmail	application/atomicmail	[Nathaniel_Borenstein]
atomsvc+xml	application/atomsvc+xml	[RFC5023]
auth-policy+xml	application/auth-policy+xml	[RFC4745]
bacnet-xdd+zip	application/bacnet-xdd+zip	[ASHRAE][Dave_Robin]
batch-SMTP	application/batch-SMTP	[RFC2442]
beep+xml	application/beep+xml	[RFC3080]
calendar+json	application/calendar+json	[RFC7265]
calendar+xml	application/calendar+xml	[RFC6321]
call-completion	application/call-completion	[RFC6910]
cals-1840	application/cals-1840	[RFC1895]
cbor	application/cbor	[RFC7049]
ccmp+xml	application/ccmp+xml	[RFC6503]

Choosing a Media Type…

Select media type(s)s based on the features you need.

What you need are ***H-Factors…***

H-Factors (2010)

Link Factors

- LO
- LE
- LT
- LN
- LI

```
<a href="http://www.example.org/search" title="view search page">Search</a>
```

```
<img src="http://www.example.org/images/logo" title="company logo" />
```

```
<form method="get">
  <label>Search term:</label>
  <input name="query" type="text" value="" />
  <input type="submit" />
</form>
```

```
<form method="post" action="http://www.example.org/my-keywords"/>

  <label>Keywords:</label>
  <input name="keywords" type="text" value="" />
  <input type="submit" />
</form>
```

```
function delete(id)
{
  var client = new XMLHttpRequest();
  client.open("DELETE", "/records/"+id);
}
```

H-Factors

Control Factors

- CR
- CU
- CM
- CL

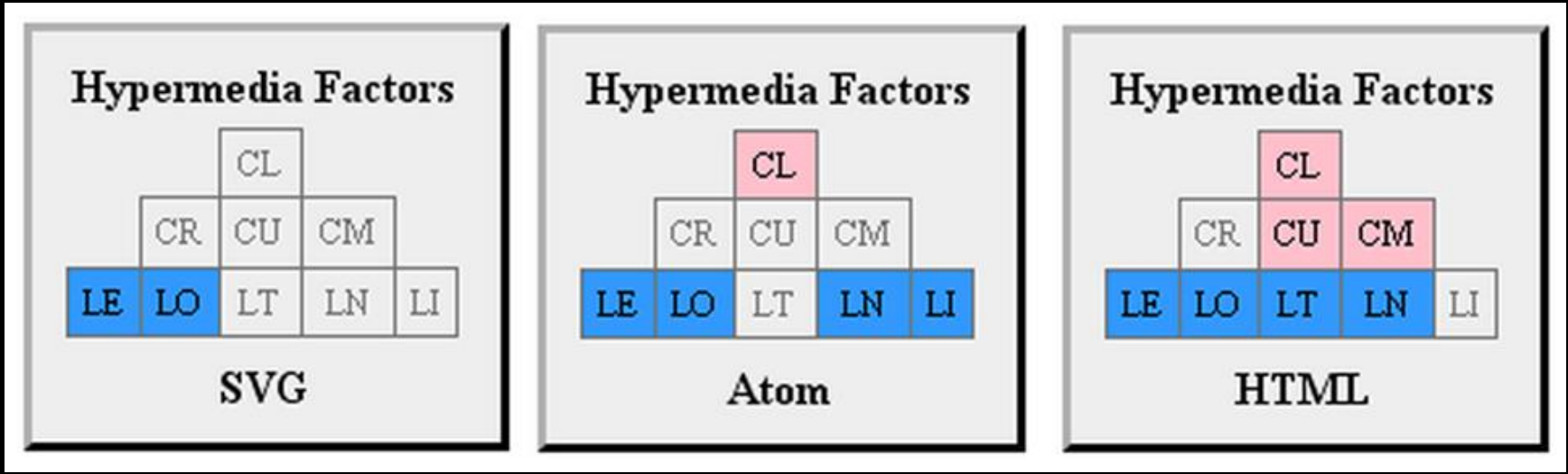
```
<xsl:include href="http://www.exmaple.org/newsfeed" accept="application/rss" />
```

```
<form method="post" action="http://www.example.org/my-keywords" enctype="application/x-www-form-urlencoded" />

  <label>Keywords:</label>
  <input name="keywords" type="text" value="" />
  <input type="submit" />
</form>
```

```
<form method="post" action="http://www.example.org/my-keywords" />
  <label>Keywords:</label>

  <input name="keywords" type="text" value="" />
  <input type="submit" />
</form>
```

```
<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>Some text.</content>
  <link rel="edit" href="http://example.org/edit/first-post.atom"/>
</entry>
```

H-Factors


Hypermedia Factors
CL
CR CU CM
LE LO LT LN LI
SVG
Hypermedia Factors
CL
CR CU CM
LE LO LT LN LI
Atom
Hypermedia Factors
CL
CR CU CM
LE LO LT LN LI
HTML

Choose a Media Type

Use application/json, application/xml
Collection type: Atom, OData, Collection+JSON
Free-form: HTML, Siren, HAL, JSON-LD
Invent your own semantic type

Choose a Media Type

~~Use application/json, application/xml~~
Collection type: Atom, OData, Collection+JSON)
Free-form: HTML, Siren, HAL, JSON-LD)
Invent your own semantic type

Choose a Media Type

~~Use application/json, application/xml~~

Collection type: Atom, OData, Collection+JSON

Free-form: HTML, Siren, HAL, JSON-LD

~~Invent your own semantic type~~

Choose a Media Type

~~Use application/json, application/xml~~

Collection type: Atom, OData, Collection+JSON

~~Free-form: HTML, Siren, HAL, JSON-LD~~

~~Invent your own semantic type~~

Choose a Media Type

~~Use application/json, application/xml~~

Collection type: Atom, OData, Collection+JSON

~~Free-form: HTML, Siren, HAL, JSON-LD~~

~~Invent your own semantic type~~

Excercise Six:
Select a Media Type for your API

Implementation

ta-da!

Publication

Publication

Publish your "billboard" URL
Publish your profile
Register new rel values and/or media types
Publish the documentation
Consider "well-known" URIs

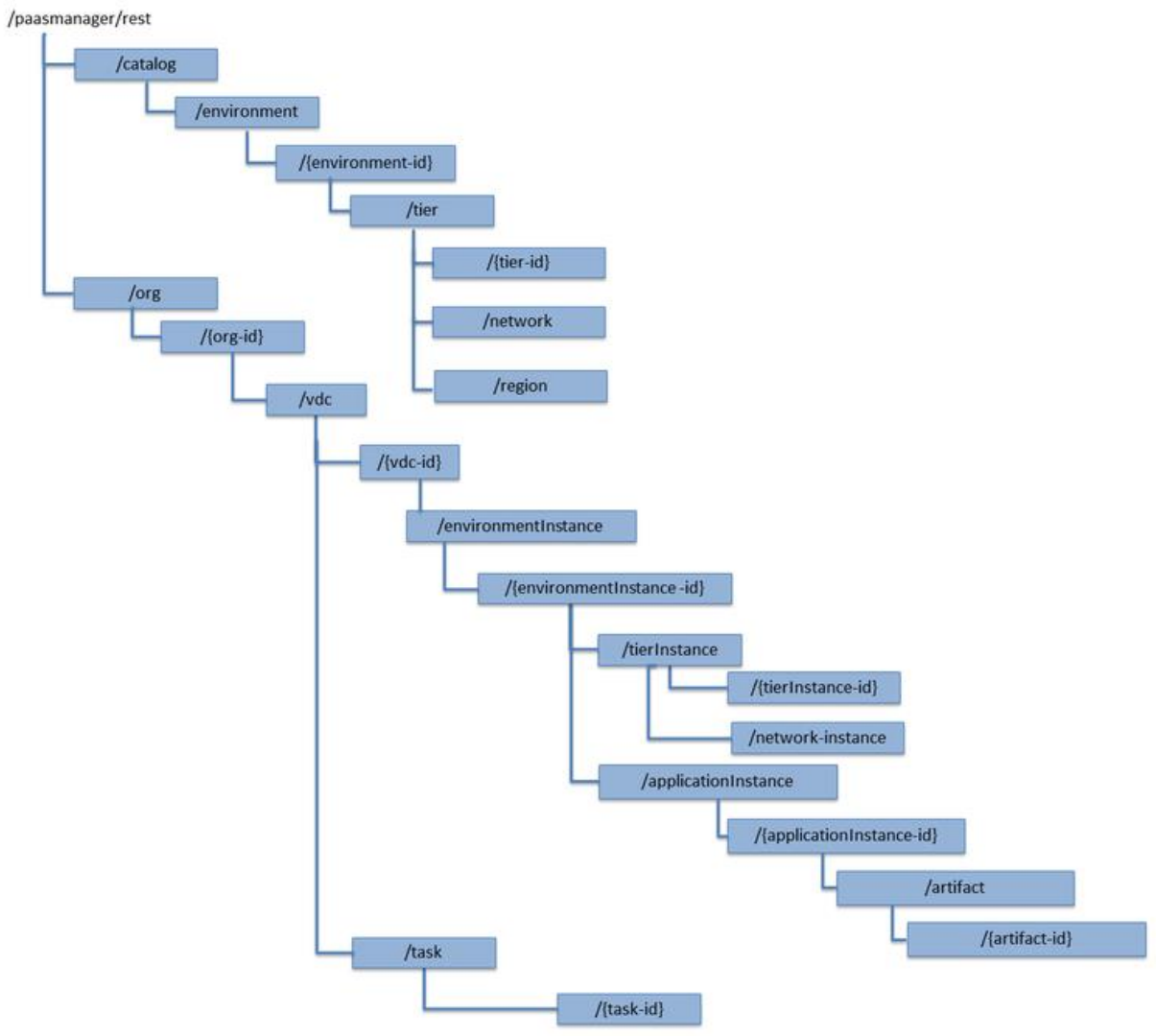
Now, you're done!

Seven-Step API Design Guide

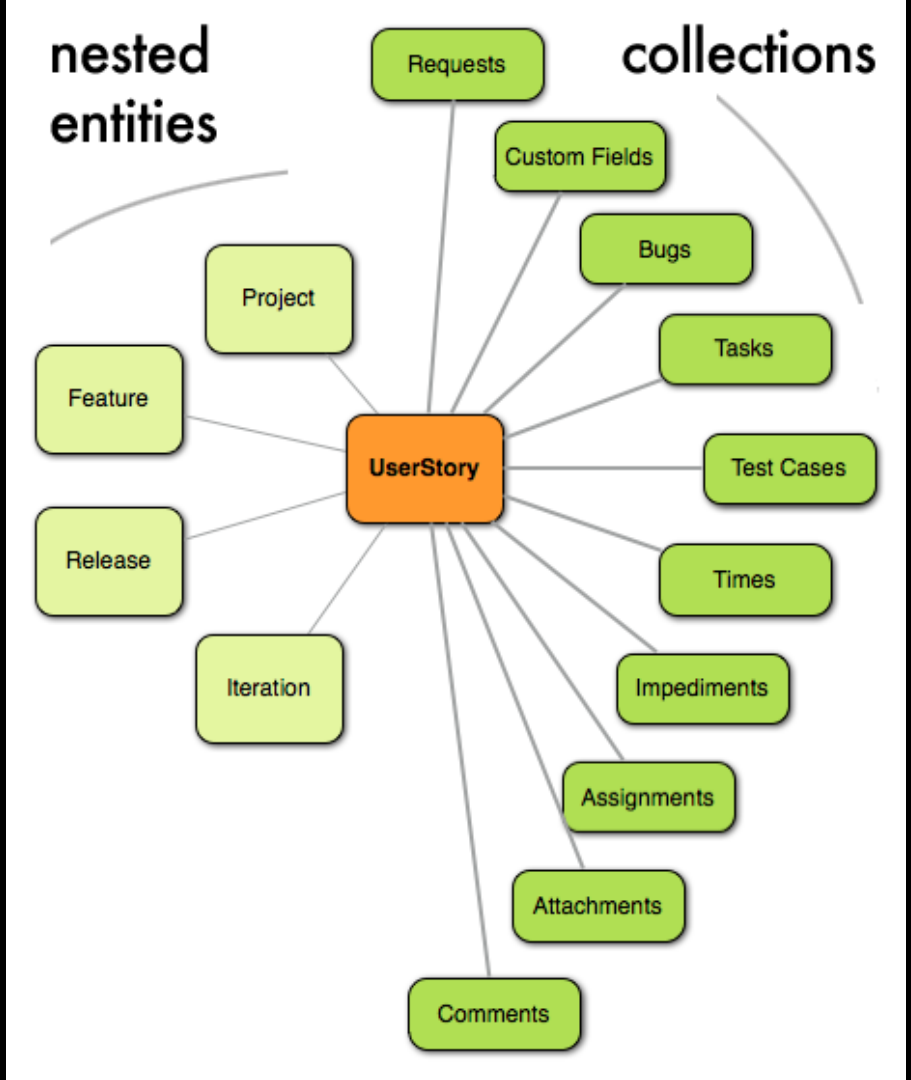
List Semantic Descriptors
Draw State Diagram
Reconcile Names
Produce API Profile
Choose a Message Model
Implement the API!
Publish the Billboard URL

Some Final Advice

Resources are an implementation detail


/paasmanager/rest
/catalog
/environment
/{environment-id}
/tier
/{tier-id}
/network
/region
/org
/{org-id}
/vdc
/{vdc-id}
/environmentInstance
/{environmentInstance -id}
/tierInstance
/{tierInstance-id}
/network-instance
/applicationInstance
/{applicationInstance-id}
/artifact
/{artifact-id}
/task
/{task-id}

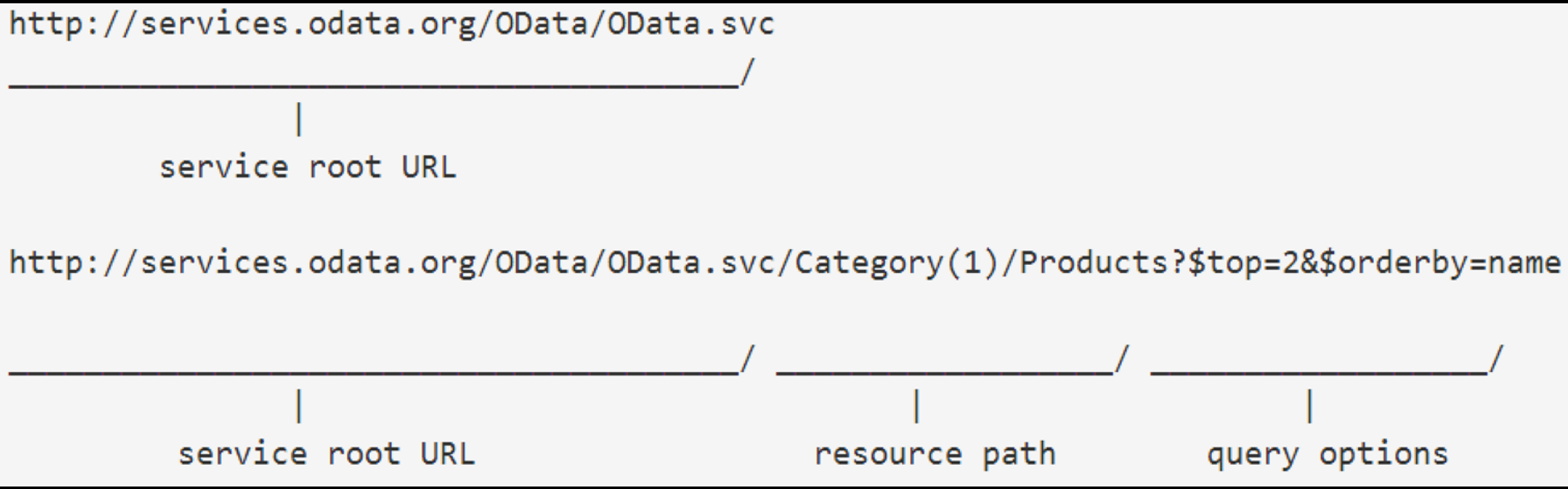
Don't fall into the collection trap


nested entities
collections
Requests
Custom Fields
Bugs
Tasks
Test Cases
Times
Impediments
Assignments
Attachments
Comments
Iteration
Release
Feature
Project
UserStory

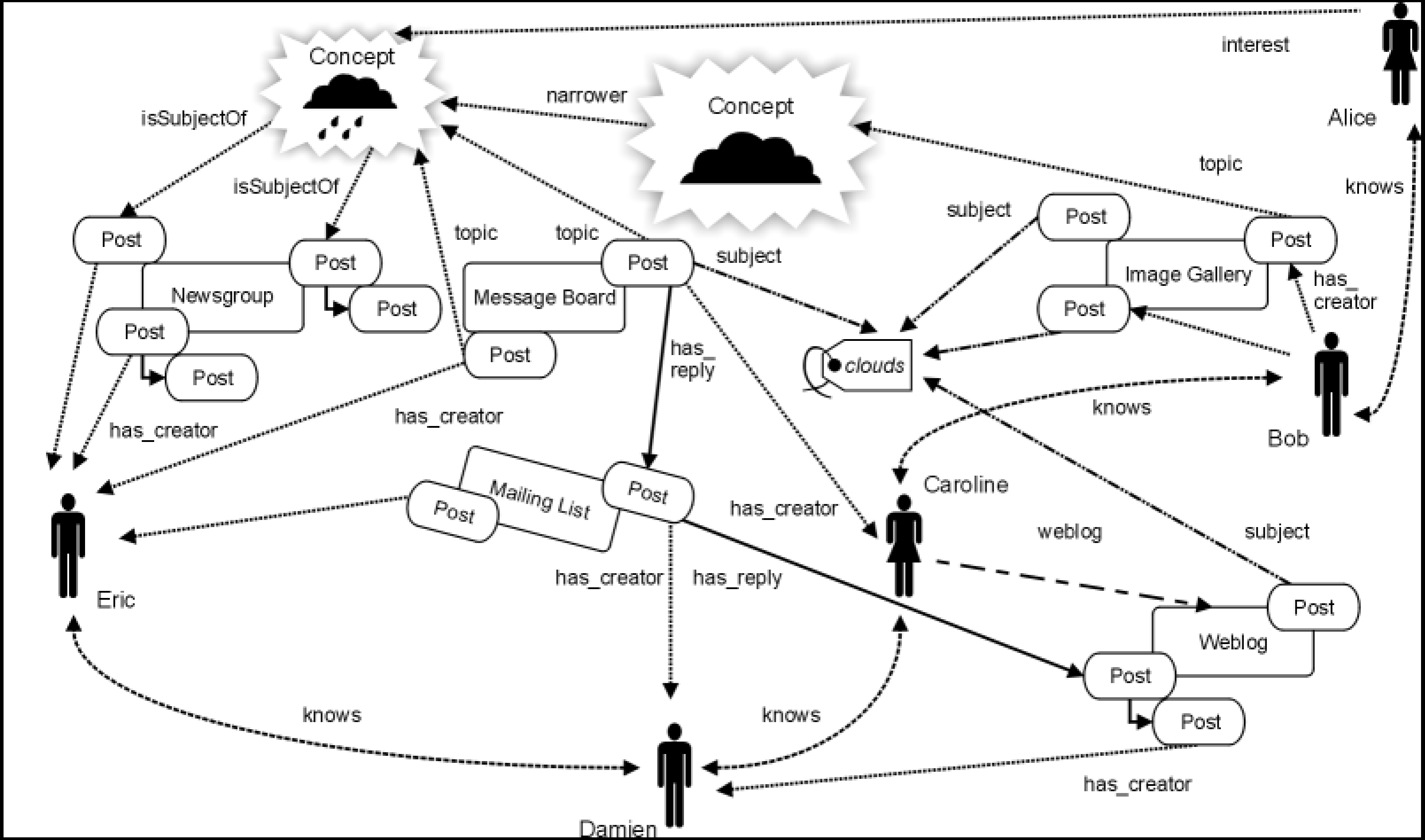
Don't start with the representation format

```
{ "collection" :
  {
    "version" : "1.0",
    "href" : "http://example.org/friends/",

    "links" : [
      {"rel" : "feed", "href" : "http://example.org/friends/rss"},
      {"rel" : "queries", "href" : "http://example.org/friends/?queries"},
      {"rel" : "template", "href" : "http://example.org/friends/?template"}
    ],

    "items" : [
      {
        "href" : "http://example.org/friends/jdoe",
        "data" : [
          {"name" : "full-name", "value" : "J. Doe", "prompt" : "Full Name"},
          {"name" : "email", "value" : "jdoe@example.org", "prompt" : "Email"}
        ],
        "links" : [
          {"rel" : "blog", "href" : "http://examples.org/blogs/jdoe", "prompt" : "Blog"},
          {"rel" : "avatar", "href" : "http://examples.org/images/jdoe", "prompt" : "Avatar", "render" : "image"}
        ]
      }
    ]
  }
}
```

URL design doesn't matter

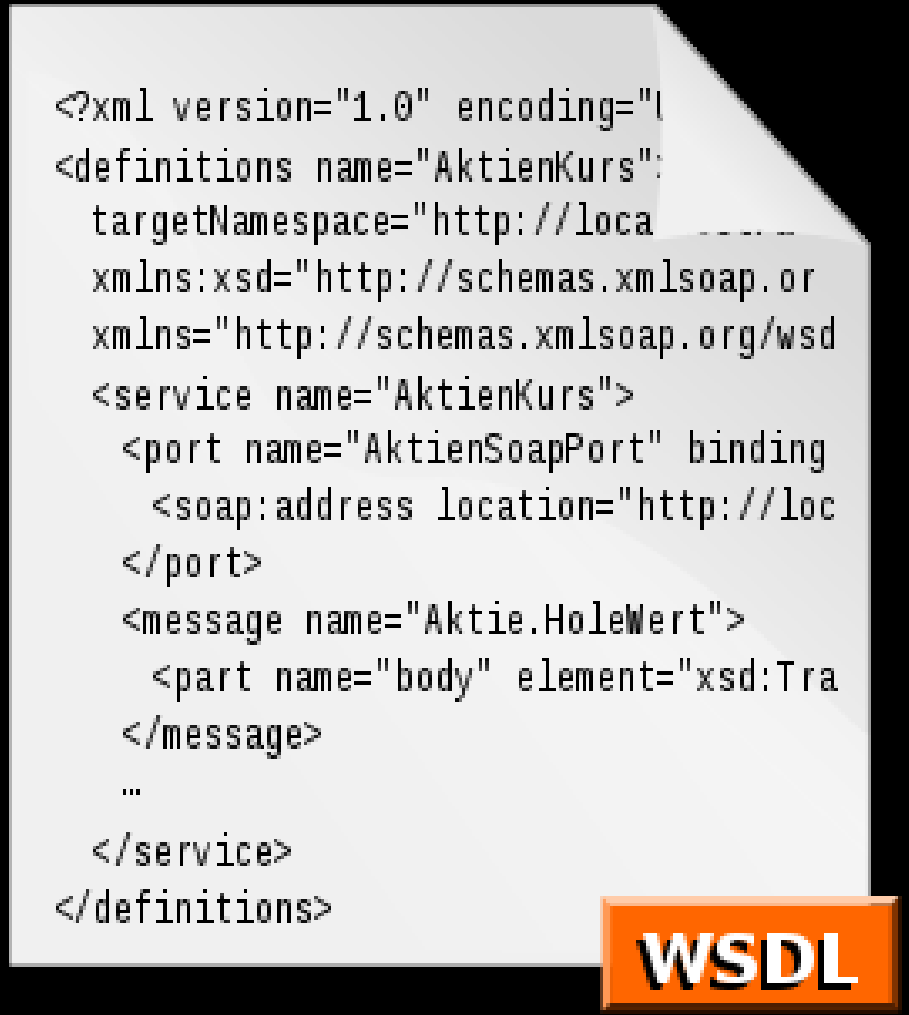

http://services.odata.org/OData/OData.svc
service root URL
http://services.odata.org/OData/OData.svc/Category(1)/Products?$top=2&$orderby=name
service root URL
resource path
query options

Standard names are probably better than yours.



Concept
Concept
narrower
interest
Alice
isSubjectOf
isSubjectOf
topic
topic
topic
knows
subject
subject
Post
Post
Post
Post
Post
Post
Post
Post
Newsgroup
Message Board
Image Gallery
has_
creator
has_
reply
clouds
has_creator
has_creator
knows
Bob
Mailing List
Post
Post
Caroline
has_creator
Eric
weblog
subject
has_creator
has_reply
Post
Weblog
Post
Post
knows
knows
has_creator
Damien

Don't keep all the hypermedia in one place


<?xml version="1.0" encoding="
<definitions name="AktienKurs"
targetNamespace="http://loca
xmlns:xsd="http://schemas.xmlsoap.or
xmlns="http://schemas.xmlsoap.org/wsd
<service name="AktienKurs">
<port name="AktienSoapPort" binding
<soap:address location="http://loc
</port>
<message name="Aktie.HoleWert">
<part name="body" element="xsd:Tra
</message>
…
</service>
</definitions>
WSDL

```
Blog entry {
          _id: 1
          _rev: 2-36d81850033da710262c21462f293703
          body: JSON Schema (application/schema+json) has several purposes, one
                instance validation. The validation process may be interactive o
                interactive. For instance, applications may use JSON Schema to b
                interface enabling interactive content generation in addition to
                checking, or validate data retrieved from various sources. This
                describes schema keywords dedicated to validation purposes.
        schema: blog
         title: JSON Schema
        author:  Author  Mike
      comments: [
                     Comment   comment  {
                        author:  Author  Mike
                            id: 2
                         title: json
                    }
                ]
          date: Invalid Date
          tags: [
                     Tag  json
                     Tag  jsonschema
```

Some Final Advice

Resources are implementation details
Don't fall into the collection trap
Don't start w/ the representation format
URL design doesn't matter
Standard names are probably better than yours
Don't keep all the hypermedia in one place

In Conclusion...


LOST
CONFUSED
UNSURE
UNCLEAR
PERPLEXED


HIGH VALUE
HIGH QUALITY


STOP
WRONG WAY

SUCCESS

Because you too can own this face of pure accomplishment

In Conclusion...

Don't confuse implementation w/ design
Design is the hard part (high value)
Implementation is the easy part (high speed)
Avoid common design mistakes
Go out and make lots of APIs!

API Design Workshop

Mike Amundsen,
API Academy / CA
@mamund