# Hypermedia-Oriented Design

## An approach for supporting evolvable distributed network applications

Mike Amundsen
amundsen.com
@mamund

2011-10-19/20
W3C Workshop on Data and Services Integration

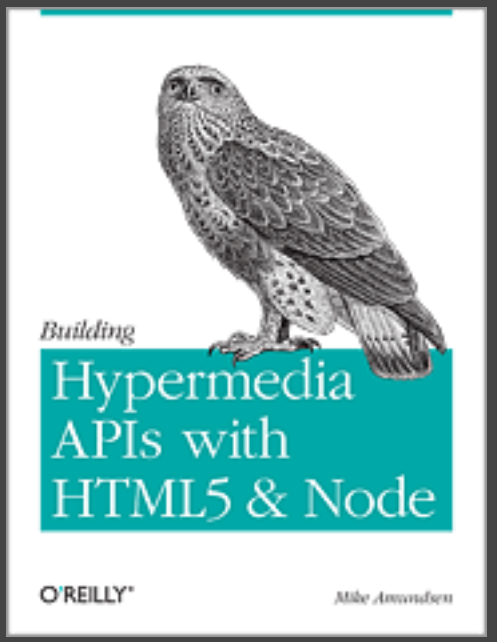
# Mike Amundsen

- Developer/Architect
- Presenter
- Author
- @mamund

# Mike Amundsen

- Developer/Architect
- Presenter
- Author
- @mamund

*My current focus is on defining and exploiting hypermedia to implement long-lived evolvable solutions for the WWW.*



"Affordances" - Donald Norman, 1994

"The value of a well-designed object..."

***- Donald Norman, 1994***

# "The value of a well-designed object..."

"...is when it has such a rich set of affordances

***- Donald Norman, 1994***

# "The value of a well-designed object..."

"...is when it has such a rich set of affordances that the people who use it

***- Donald Norman, 1994***

# "The value of a well-designed object..."

"...is when it has such a rich set of affordances
that the people who use it
can do things with it

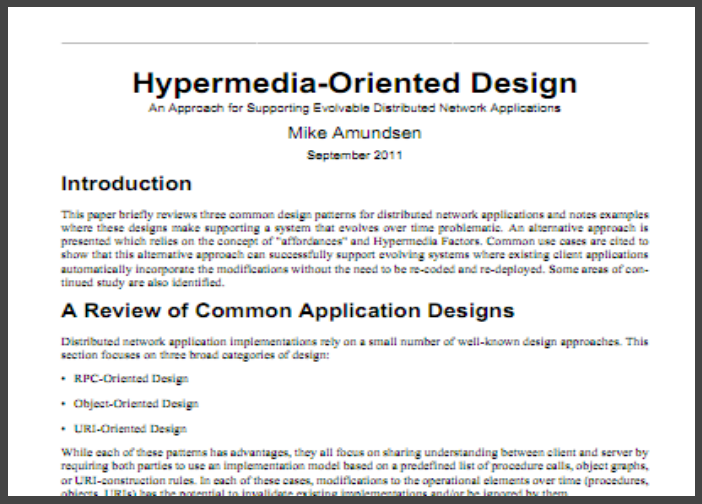
***- Donald Norman, 1994***

# "The value of a well-designed object..."

"...is when it has such a rich set of affordances
that the people who use it
can do things with it
that the designer never imagined."

***- Donald Norman, 1994***

# The Paper



## Hypermedia-Oriented Design

An Approach for Supporting Evolvable Distributed Network Applications

Mike Amundsen

September 2011

### Introduction

This paper briefly reviews three common design patterns for distributed network applications and notes examples where these designs make supporting a system that evolves over time problematic. An alternative approach is presented which relies on the concept of "affordances" and Hypermedia Factors. Common use cases are cited to show that this alternative approach can successfully support evolving systems where existing client applications automatically incorporate the modifications without the need to be re-coded and re-deployed. Some areas of continued study are also identified.

### A Review of Common Application Designs

Distributed network application implementations rely on a small number of well-known design approaches. This section focuses on three broad categories of design:

- RPC-Oriented Design
- Object-Oriented Design
- URI-Oriented Design

While each of these patterns has advantages, they all focus on sharing understanding between client and server by requiring both parties to use an implementation model based on a predefined list of procedure calls, object graphs, or URI-construction rules. In each of these cases, modifications to the operational elements over time (procedures, objects, URIs) has the potential to invalidate existing implementations and/or be ignored by them.

# Common designs used to implement

# Common designs used to implement distributed applications over the WWW

Common designs used to implement distributed applications over the WWW make integration and evolvability

Common designs used to implement distributed applications over the WWW make integration and evolvability a challenge

Common designs used to implement distributed applications over the WWW make integration and evolvability a challenge

# Context for this presentation:

**The medium is the message**

# Common Designs for Dist-Net Apps

- **RPC-Oriented**

# Common Designs for Dist-Net Apps

- **RPC-Oriented**

```
<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:y="http://example.org/math/"
  xmlns:ns="http://example.org/math/types/"
  targetNamespace="http://example.org/math/"
>
  <types>
    <xs:schema
      targetNamespace="http://example.org/math/types/"
      xmlns="http://example.org/math/types/"
    >
     <xs:complexType name="MathInput">
       <xs:sequence>
      <xs:element name="x" type="xs:double"/>
      <xs:element name="y" type="xs:double"/>
       </xs:sequence>
     </xs:complexType>
     <xs:complexType name="MathOutput">
       <xs:sequence>
         <xs:element name="result" type="xs:double"/>
       </xs:sequence>
     </xs:complexType>
     <xs:element name="Add" type="MathInput"/>
     <xs:element name="AddResponse" type="MathOutput"/>
     <xs:element name="Subtract" type="MathInput"/>
     <xs:element name="SubtractResponse" type="MathOutput"/>
     <xs:element name="Multiply" type="MathInput"/>
     <xs:element name="MultiplyResponse" type="MathOutput"/>
     <xs:element name="Divide" type="MathInput"/>
     <xs:element name="DivideResponse" type="MathOutput"/>
    </xs:schema>
  </types>
    ...
</definitions>
```

# Common Designs for Dist-Net Apps

- **RPC-Oriented**

*In RPC-Oriented models, the client is "bound" to the procedure list.*

```
<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:y="http://example.org/math/"
  xmlns:ns="http://example.org/math/types/"
  targetNamespace="http://example.org/math/"
>
  <types>
    <xs:schema
      targetNamespace="http://example.org/math/types/"
      xmlns="http://example.org/math/types/"
    >
     <xs:complexType name="MathInput">
       <xs:sequence>
      <xs:element name="x" type="xs:double"/>
      <xs:element name="y" type="xs:double"/>
       </xs:sequence>
     </xs:complexType>
     <xs:complexType name="MathOutput">
       <xs:sequence>
         <xs:element name="result" type="xs:double"/>
       </xs:sequence>
     </xs:complexType>
     <xs:element name="Add" type="MathInput"/>
     <xs:element name="AddResponse" type="MathOutput"/>
     <xs:element name="Subtract" type="MathInput"/>
     <xs:element name="SubtractResponse" type="MathOutput"/>
     <xs:element name="Multiply" type="MathInput"/>
     <xs:element name="MultiplyResponse" type="MathOutput"/>
     <xs:element name="Divide" type="MathInput"/>
     <xs:element name="DivideResponse" type="MathOutput"/>
    </xs:schema>
  </types>
    ...
</definitions>
```

# Common Designs for Dist-Net Apps

- RPC-Oriented
- **OO-Oriented**

# Common Designs for Dist-Net Apps

- RPC-Oriented
- **OO-Oriented**


```
{
   "data": [
      {
         "id": "501072349_10150317427757350",
         "from": {
            "name": "Mike Amundsen",
            "id": "501072349"
         },
         "picture": "http://photos-f.ak.fbcdn.net/hphotos-ak-ash4/3026
         "link": "http://www.facebook.com/photo.php?fbid=1015031742808
         "name": "Mobile Uploads",
         "icon": "http://static.ak.fbcdn.net/rsrc.php/v1/yb/r/FzSuxp0E
         "privacy": {
            "description": "Friends",
            "value": "ALL_FRIENDS"
         },
         "type": "photo",
         "object_id": "10150317428082350",
         "application": {
            "name": "Facebook for Android",
            "id": "350685531728"
         },
         "created_time": "2011-09-12T08:14:05+0000",
         "updated_time": "2011-09-12T08:14:05+0000",
         "comments": {
            "count": 0
         }
      },
      {
         "id": "501072349_10150317174857350",
         "from": {
            "name": "Shannon Lee",
            "id": "12913651"
```

# Common Designs for Dist-Net Apps

- RPC-Oriented
- **OO-Oriented**

*In Object-Oriented designs, the client is "bound" to the object graph.*

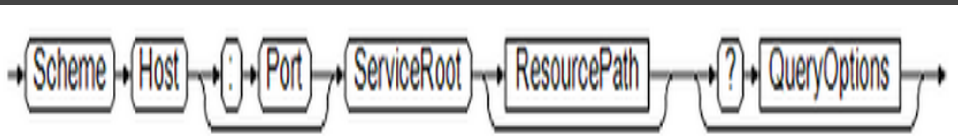

```
{
   "data": [
      {
         "id": "501072349_10150317427757350",
         "from": {
            "name": "Mike Amundsen",
            "id": "501072349"
         },
         "picture": "http://photos-f.ak.fbcdn.net/hphotos-ak-ash4/3026
         "link": "http://www.facebook.com/photo.php?fbid=1015031742808
         "name": "Mobile Uploads",
         "icon": "http://static.ak.fbcdn.net/rsrc.php/v1/yb/r/FzSuxp0E
         "privacy": {
            "description": "Friends",
            "value": "ALL_FRIENDS"
         },
         "type": "photo",
         "object_id": "10150317428082350",
         "application": {
            "name": "Facebook for Android",
            "id": "350685531728"
         },
         "created_time": "2011-09-12T08:14:05+0000",
         "updated_time": "2011-09-12T08:14:05+0000",
         "comments": {
            "count": 0
         }
      },
      {
         "id": "501072349_10150317174857350",
         "from": {
            "name": "Shannon Lee",
            "id": "12913651"
```

# Common Designs for Dist-Net Apps

- RPC-Oriented
- OO-Oriented
- **URI-Oriented**

# Common Designs for Dist-Net Apps

- RPC-Oriented
- OO-Oriented
- **URI-Oriented**




The following are two example URIs broken down into their component parts:

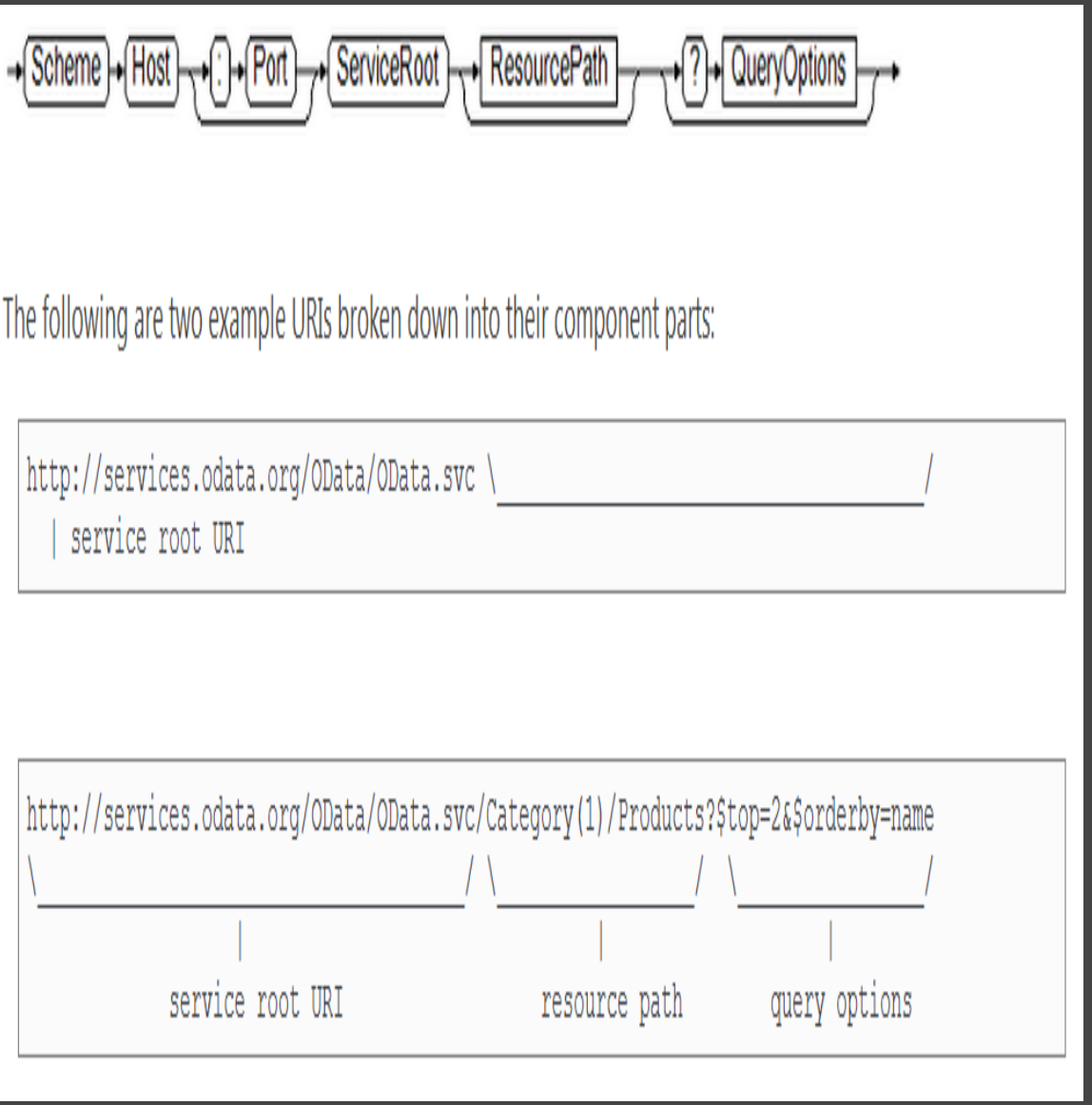
```
http://services.odata.org/OData/OData.svc \_____________________________/
  | service root URI
```

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

# Common Designs for Dist-Net Apps

- RPC-Oriented
- OO-Oriented
- **URI-Oriented**

*In URI-Oriented designs, the client is "bound" to URI construction rules*



# Versioning instead of Evolving

# Versioning instead of Evolving

- Clients are "bound" to first-class domain elements
  - Procedures
  - Objects
  - Identifiers (URIs)

# Versioning instead of Evolving

- Clients are "bound" to first-class domain elements
  - Procedures
  - Objects
  - Identifiers (URIs)
- Changing domain elements may "break" the client

# Versioning instead of Evolving

- Clients are "bound" to first-class domain elements
  - Procedures
  - Objects
  - Identifiers (URIs)
- Changing domain elements may "break" the client
- Common strategy is to use "versioning" to isolate clients from changes in domain elements

# Versioning instead of Evolving

- Clients are "bound" to first-class domain elements
  - Procedures
  - Objects
  - Identifiers (URIs)
- Changing domain elements may "break" the client
- Common strategy is to use "versioning" to isolate clients from changes in domain elements

*Every "version" is an evolutionary "dead end."*

# The Hypermedia Alternative

# Hypermedia-Oriented Design

- RPC-Oriented
- OO-Oriented
- URI-Oriented
- **Hypermedia**

# Hypermedia-Oriented Design

- RPC-Oriented
- OO-Oriented
- URI-Oriented
- **Hypermedia**

**LE** Support for embedded links (HTTP GET)

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

**LO** Support for out-bound navigational links (HTTP GET)

```
<a href="http://www.example.org/search" title
```

**LT** Support for templated queries (HTTP GET)

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

**LN** Support for non-idempotent updates (HTTP POST)

```
<form method="post" action="http://www.exampl

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

**LI** Support for idempotent updates (HTTP PUT, DELETE)

```
function delete(id)
{
  var client = new XMLHttpRequest();
```

# Hypermedia-Oriented Design

- RPC-Oriented
- OO-Oriented
- URI-Oriented
- **Hypermedia**

*In Hypermedia-Oriented designs, the client is "bound" to the affordances.*

**LE** Support for embedded links (HTTP GET)

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

**LO** Support for out-bound navigational links (HTTP GET)

```
<a href="http://www.example.org/search" title
```

**LT** Support for templated queries (HTTP GET)

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

**LN** Support for non-idempotent updates (HTTP POST)

```
<form method="post" action="http://www.exampl

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

**LI** Support for idempotent updates (HTTP PUT, DELETE)

```
function delete(id)
{
  var client = new XMLHttpRequest();
```

# Hypermedia-Oriented Designs can Evolve

# Hypermedia-Oriented Designs can Evolve

- Clients are bound to *affordances*, not domain-specifc elements

# Hypermedia-Oriented Designs can Evolve

- Clients are bound to *affordances*, not domain-specifc elements
- Changes in domain elements do not affect clients directly
  - New procedures, objects, URIs

# Hypermedia-Oriented Designs can Evolve

- Clients are bound to *affordances*, not domain-specifc elements
- Changes in domain elements do not affect clients directly
  - New procedures, objects, URIs
- Servers maintain the same set of affordances *even when domain elements change.*

# Hypermedia-Oriented Designs can Evolve

- Clients are bound to *affordances*, not domain-specifc elements
- Changes in domain elements do not affect clients directly
  - New procedures, objects, URIs
- Servers maintain the same set of affordances *even when domain elements change.*

*The solution can "evolve" over time as needed.*

# Hypermedia-oriented design includes more than "data" in responses.

# Affordances, Factors, & Types

# Affordances

James Gisbon, 1977

"The affordances of the environment are what it offers ... what it provides or furnishes, either for good or ill. The verb 'to afford' is found in the dictionary, but the noun 'affordance' is not. I have made it up."

# Affordances

Donald Norman, 1988

"[T]he term affordance refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used."

# Affordances

Roy T. Fielding, 2008

"When I say Hypertext, I mean the simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions."

# Affordances

Roy T. Fielding, 2008

"When I say Hypertext, I mean the simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions."

# Affordance Aspects

# Affordance Aspects

- **Safety**

# Affordance Aspects

- **Safety**

**The HTTP protocol supports a number of "safe" actions such as HEAD, and GET.**

**The HTTP methods PUT, POST, and DELETE are categorized as "unsafe" actions.**

# Affordance Aspects

- Safety
- **Idempotence**

# Affordance Aspects

- Safety
- **Idempotence**

**In HTML when a FORM element has the METHOD property set to "get" this represents an idempotent action.**

**When the same property is set to "post" the affordance represents a non-idempotent action.**

# Affordance Aspects

- Safety
- Idempotence
- **Mutability**

# Affordance Aspects

- Safety
- Idempotence
- **Mutability**

**In HTML, the FORM element affords mutability.**

**The LINK element does not.**

# Affordance Aspects

- Safety
- Idempotence
- Mutability
- **Presentation**

# Affordance Aspects

- Safety
- Idempotence
- Mutability
- **Presentation**

**In HTML, the A element affords navigation.**

**The IMG element affords transclusion.**

# Hypermedia-oriented design includes more than "data" in responses.

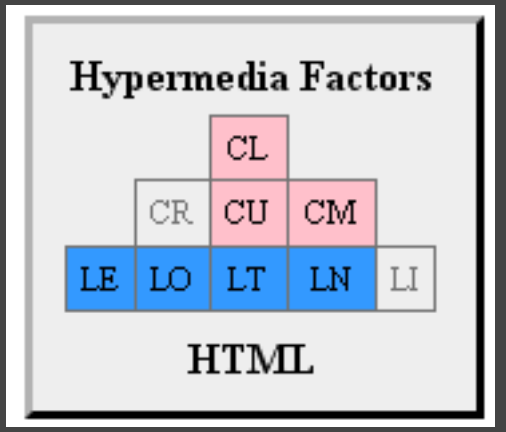
Hypermedia-oriented design
includes more than "data" in responses.

Responses also include
hypermedia controls
that tell the client
what the data
"affords."

# H-Factors

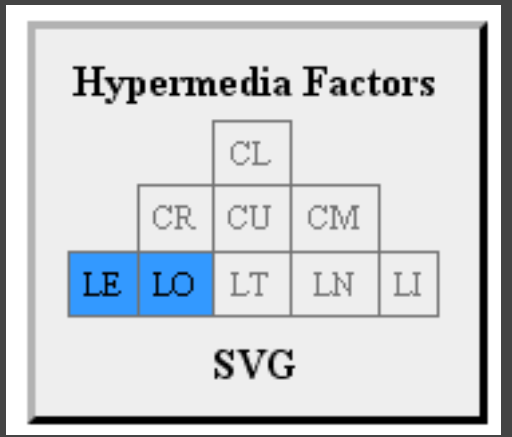
- Hypermedia controls are the affordances "through which the user obtains choices and selects actions."

- "Hypermedia Factors" or "H-Factors"
- There are five Link Factors
- There are four Control Factors



# H-Factors

## Link Factors

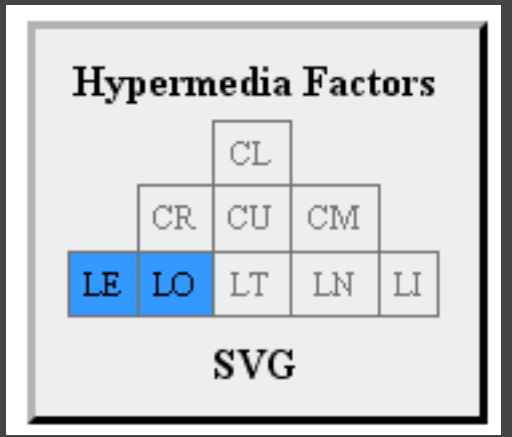


# H-Factors

## Link Factors

- LO (outbound links)

```
<html:a href="..." title="...">...</a>
```

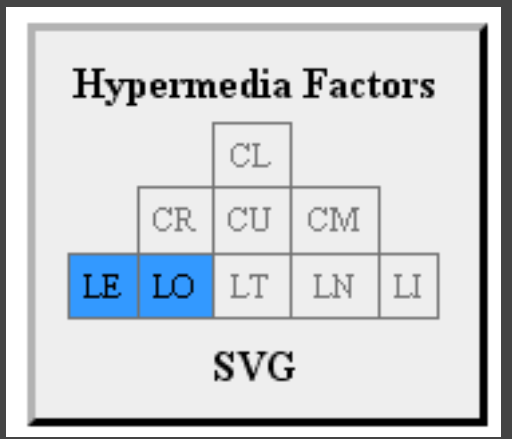


# H-Factors

## Link Factors

- LO (outbound links)
- **LE (embedded links)**

```
<x:include href="..." />
```

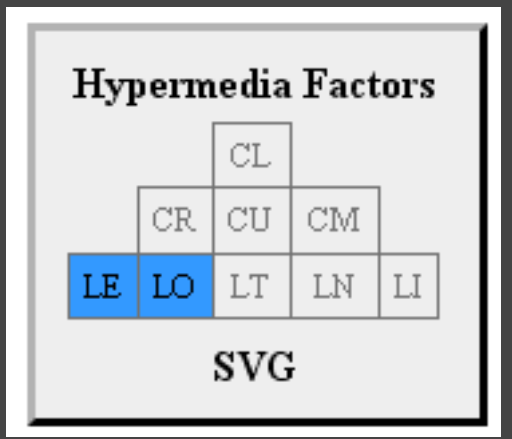


# H-Factors

## Link Factors

- LO (outbound links)
- LE (embedded links)
- **LT (templated links)**

```
<html:form method="get" action="...">
...
<html:form>
```

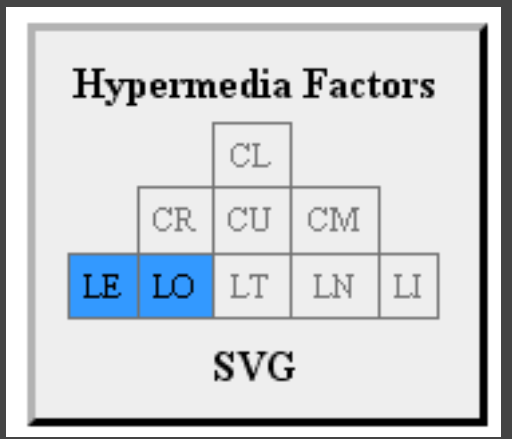


# H-Factors

## Link Factors

- LO (outbound links)
- LE (embedded links)
- LT (templated links)
- **LN (non-idempotent links)**

```
<html:form method="post" action="...">
...
<html:form>
```

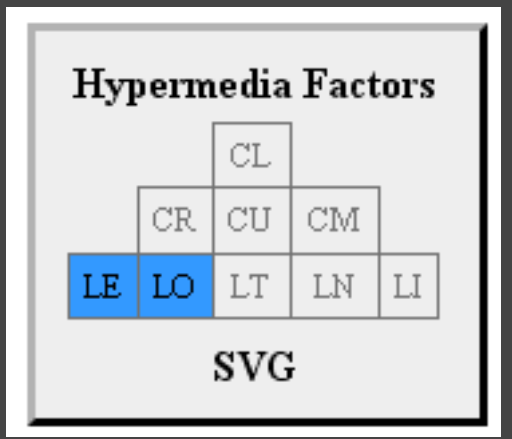


# H-Factors

## Link Factors

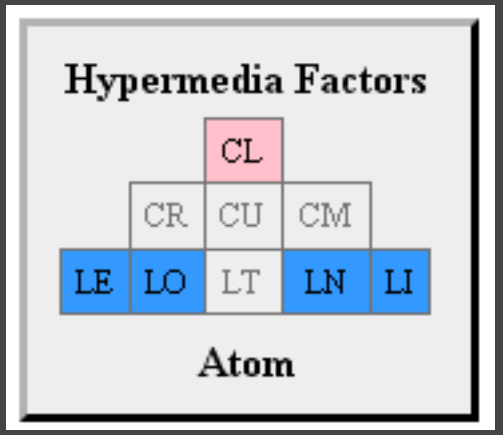
- LO (outbound links)
- LE (embedded links)
- LT (templated links)
- LN (non-idempotent links)
- **LI (idempotent links)**

```
<atom:link href="..." rel="edit" />
```



# H-Factors

## Control Factors

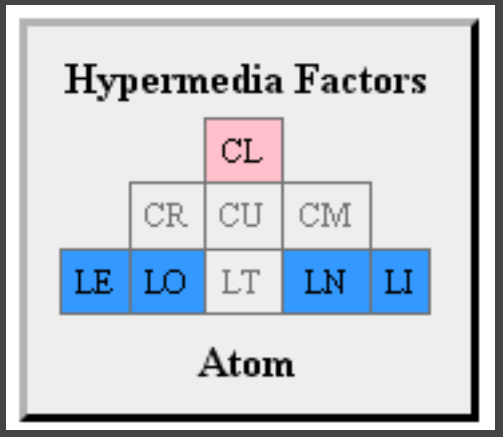


# H-Factors

## Control Factors

- CR (request control values)

```
<xsl:include href="..."
   accept="application/rss" />
```

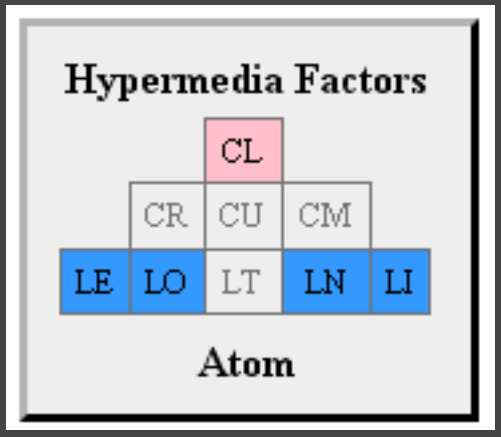


# H-Factors

## Control Factors

- CR (request control values)
- **CU (update control values)**

```
<html:form method="..." action="..."
  enctype="text/plain" />
```

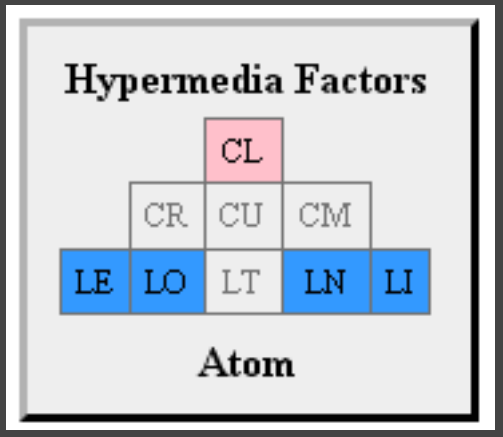


# H-Factors

## Control Factors

- CR (request control values)
- CU (update control values)
- **CM (method control values)**

```
<html:form method="post" href="...">
...
</html:form>
```

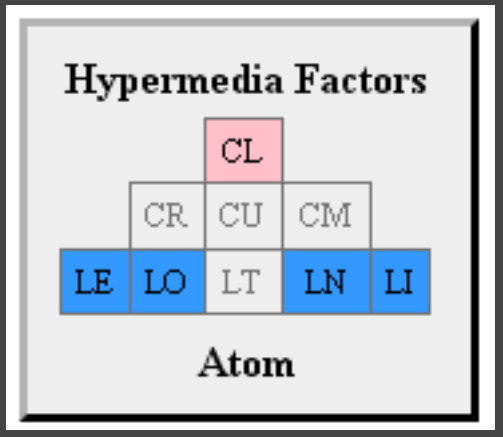


# H-Factors

## Control Factors

```
<html:link href="..." rel="stylesheet" />
```

- CR (request control values)
- CU (update control values)
- CM (method control values)
- **CL (link control values)**



# (Media) Types

# (Media) Types

- A collection of H-Factor definitions constitutes a "type"

# (Media) Types

- A collection of H-Factor definitions constitutes a "type"
- HTTP uses IANA-registered "Media Types" to represent these hypermedia collections.

# (Media) Types

- A collection of H-Factor definitions constitutes a "type"
- HTTP uses IANA-registered "Media Types" to represent these hypermedia collections.
- (Hyper)Media Types "afford" data

Hypermedia-oriented design
includes more than "data" in responses.

Responses also include
hypermedia controls
that tell the client
what the data
"affords."

Hypermedia-oriented design
includes more than "data" in responses.

Responses also include
hypermedia controls
that tell the client
what the data
"affords."

Each response integrates
data and services

or, looking at it another way...

# Using hypermedia-oriented designs

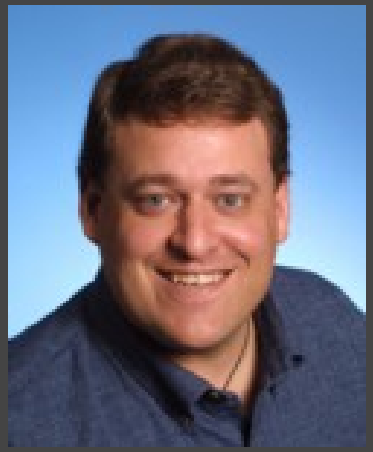
# Using hypermedia-oriented designs to implement distributed applications

# Using hypermedia-oriented designs to implement distributed applications for the WWW

Using hypermedia-oriented designs
to implement distributed applications
for the WWW
is an example of

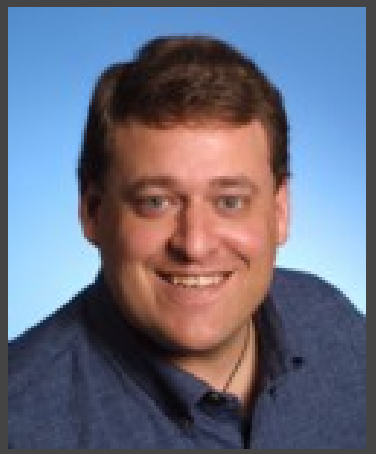
Using hypermedia-oriented designs
to implement distributed applications
for the WWW
is an example of
data and services integration

"The value of a well-designed media type..."



# "The value of a well-designed media type..."

"...is when it has such a rich set of affordances
that the people who use it
can do things with it
that the designer never imagined."



# Hypermedia-Oriented Design

## An approach for supporting evolvable distributed network applications

Mike Amundsen
amundsen.com
@mamund

**http://go.mamund.com/xtjw**