

Google™



Google Wave: Under the Hood

David Wang, Alexandre Mah, Daniel Danilatos
and Casey Whitelaw
28 May, 2009



What are waves?

- Hosted content
- Live collaboration
- Robust and extensible platform

New technology that makes Google Wave possible:

- Concurrency Control for structured data
- The Wave Editor

New technology that Google Wave *makes* possible:

- Natural Language Processing in Wave



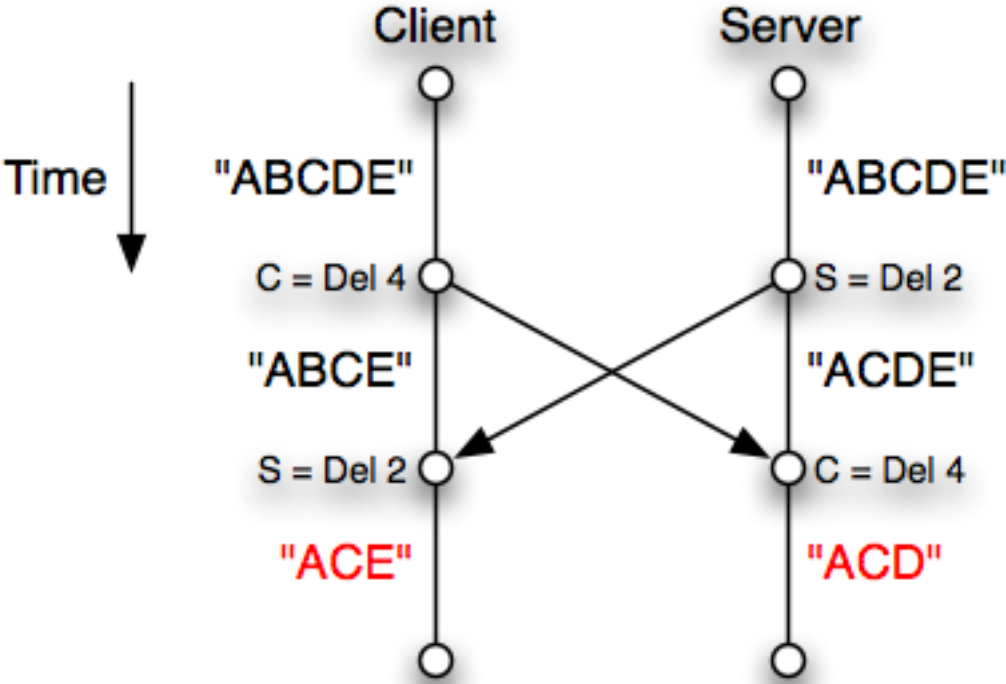
Concurrency Control



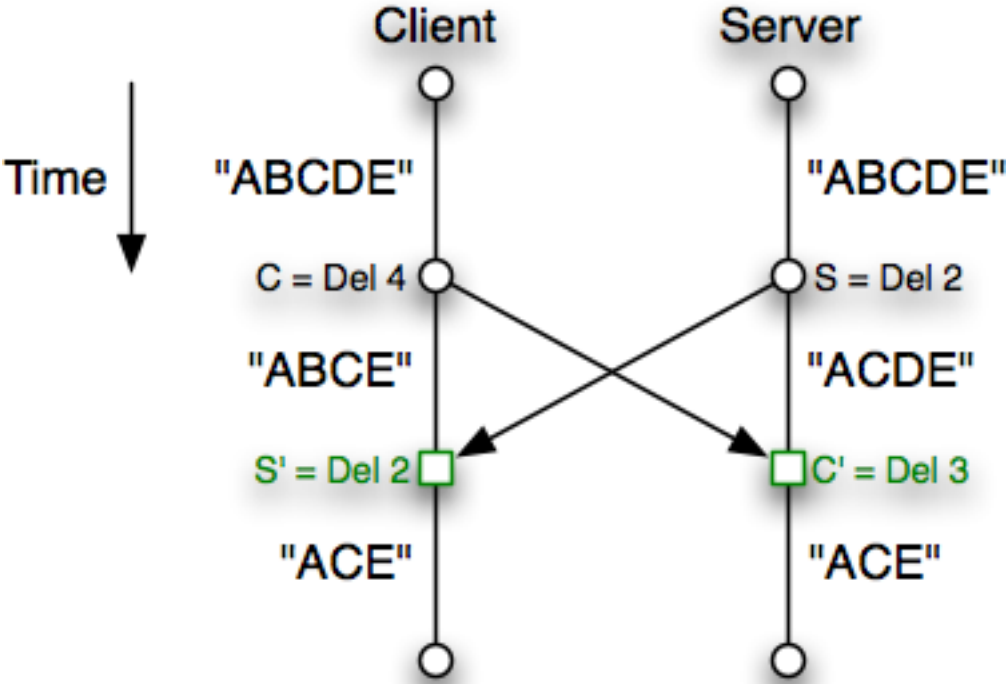
Introduction

- Concurrent rich text editor
- Existing concurrent editors
 - Google Docs, EtherPad, Subetha Edit
 - Jupiter System
- We want to have both live concurrent editing and rich text
- Operational Transformation
 - Our starting point, High-Latency, Low-Bandwidth Windowing in the Jupiter Collaboration System
(David A. Nichols, Pavel Curtis, Michael Dixon and John Lamping)

Concurrency (the naive way)



Concurrency (the correct way)



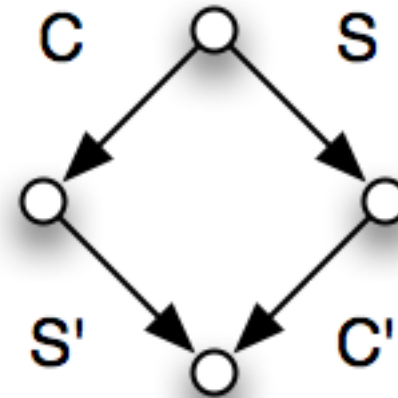
Operational Transformation

- Any changes to the shared object are described as an **operation**
 - e.g. insert character "a" at position x
- As long as there is a function **transform()** with the following behaviour, the states of the object in all the clients and the server will eventually converge.
 - S = Server Operation
 - C = Client Operation
 - S' = Transformed Server Operation
 - C' = Transformed Client Operation

$(S', C') = \text{transform}(S, C)$

where

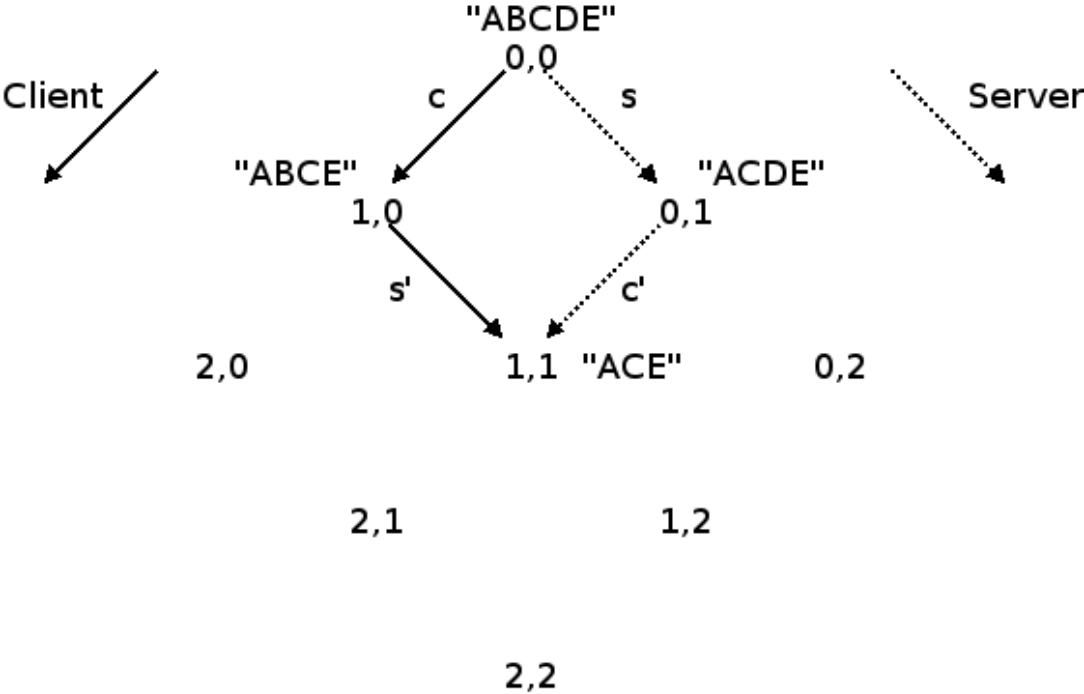
$$C' \cdot S = S' \cdot C$$



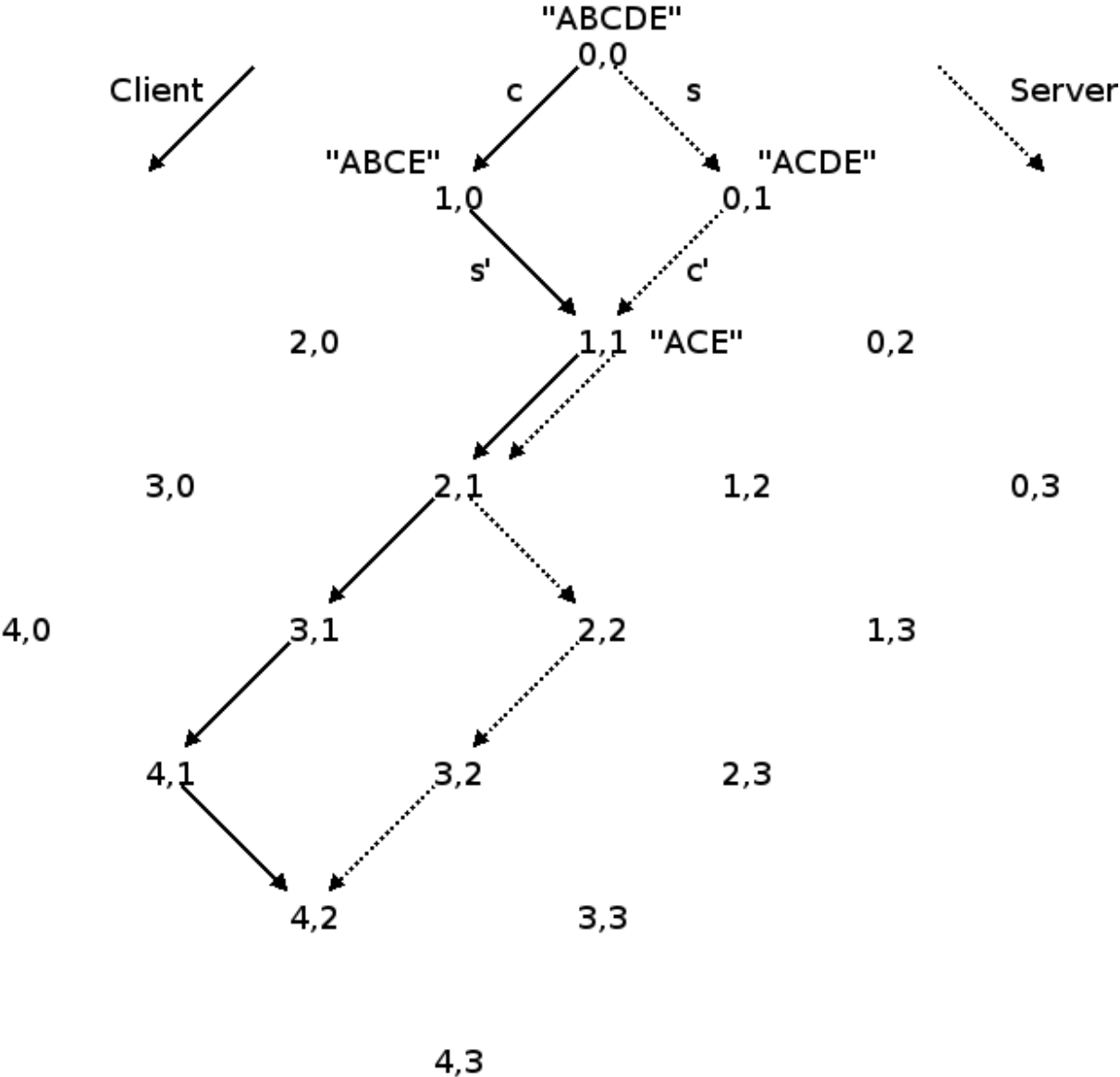
State space

The state space the client and server traverses through while processing the operations

- Any left traversal is caused by client operation
- Any right traversal is caused by server operation



State space



Changes to Operational Transformation

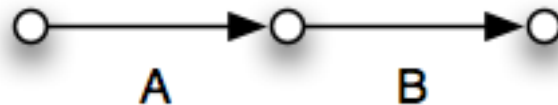
- Client waits for an ACK before sending more operations to the server. i.e. only 1 outstanding unacknowledged delta per client
 - Client keeps the inferred server OT path
 - Client transforms cached client operations before sending to server
- **Keeps the server state simple**
 - Server no longer needs to maintain multiple client states as clients always sends operations along the server's OT path

Added Support for Recovery

- Can be caused by
 - Client disconnect
 - Server crash (front end server or wave server)
 - Any form of communication failure
- Reconnect gracefully without user action

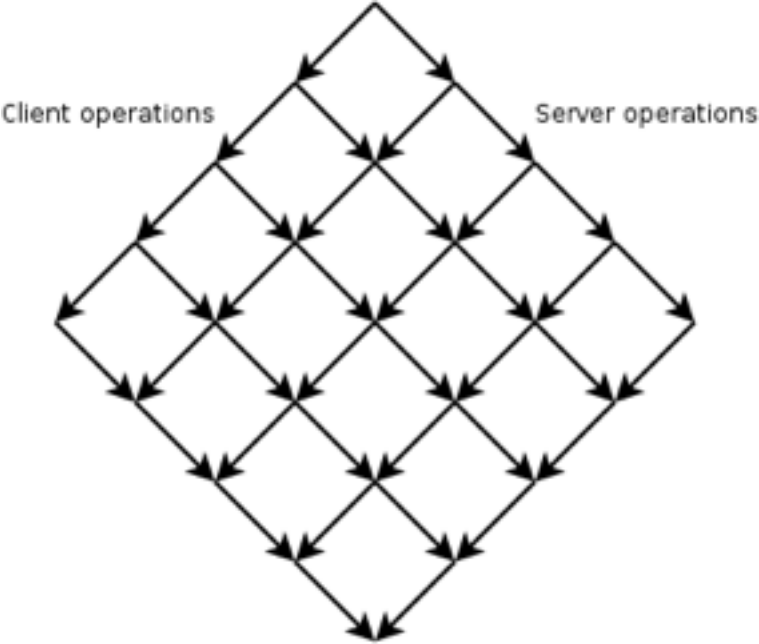
Wave Operations

We've introduced the ability to compose together any two consecutive operations and we've designed our operations in such a way that the composition of two operations is always another operation. That is, given two consecutive operations A and B , their composition $B \cdot A$ can be expressed as another operation.



Handling large transformations!

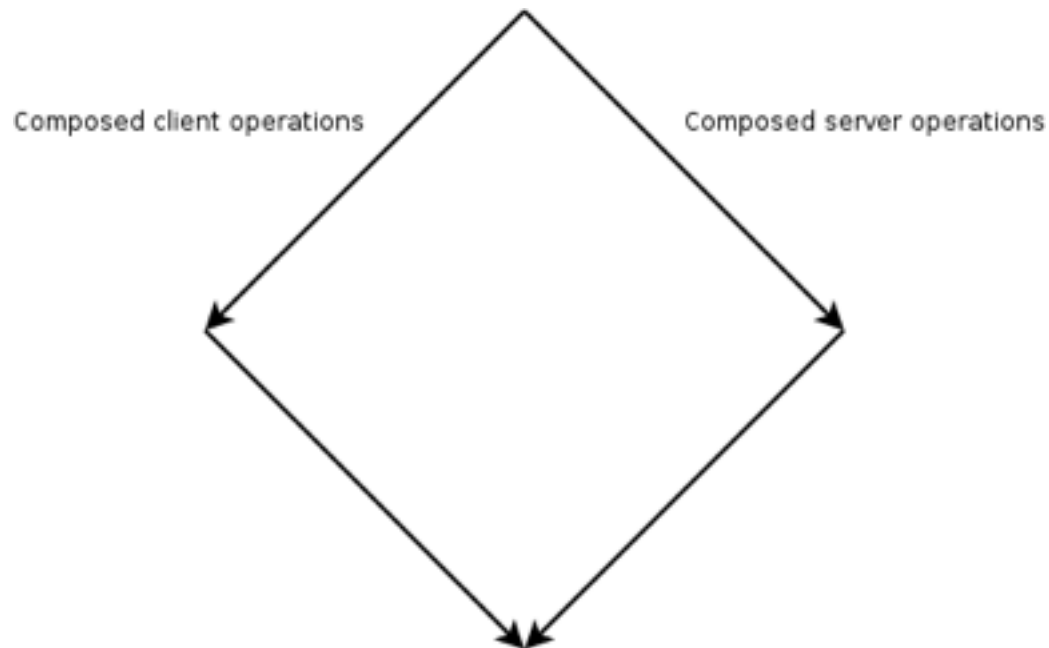
If the server and client have each accumulated a lot of concurrent operations, transformation can be expensive. This requires nm transformations, where n is the number of client operations and m is the number of server operations.



If efficient composition is possible...

Transforming many client operations with many server operations can be made efficient.

We can design composition to be efficient enough that we can cut the transformation running time to $O(n \log n + m \log m)$, where n is the total size of the client operations and m is the total size of the server operations.



The document interface

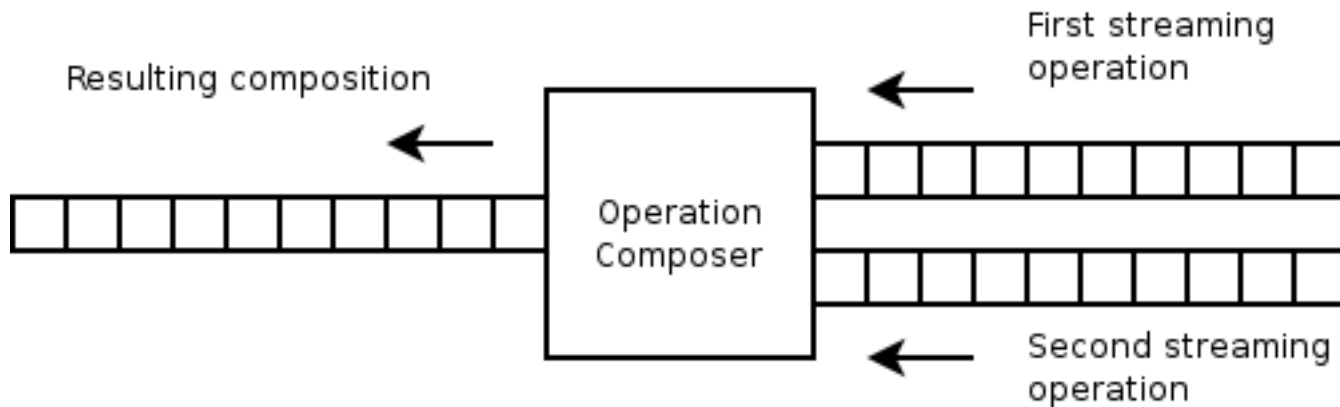
- A streaming interface.
- Traverses the operation linearly.

An example operation could perform the following sequence:

Skip 8	ElementStart tagName: "li"	Insert "hello"	ElementEnd	Skip 8	Delete 3
--------	-------------------------------	----------------	------------	--------	----------

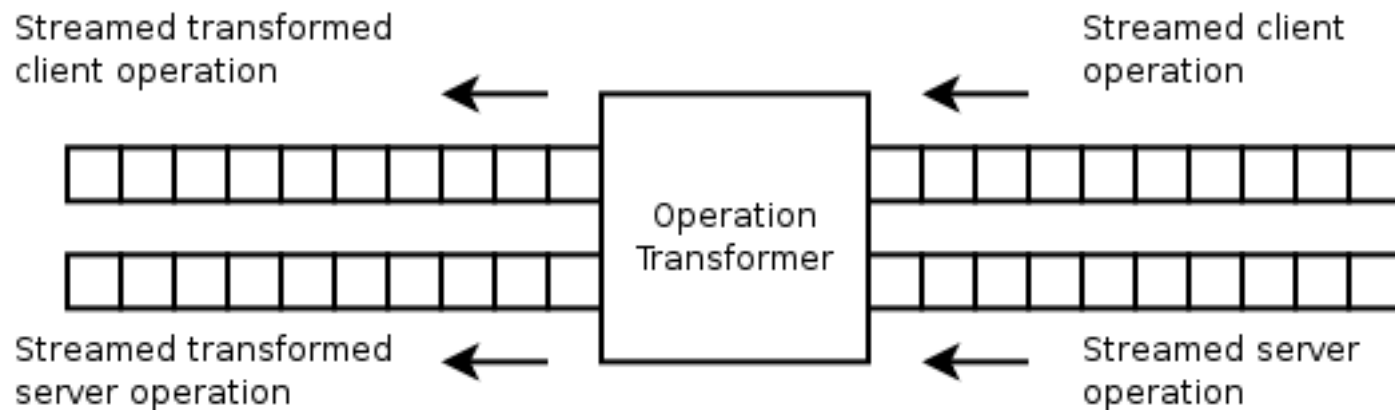
"Zipping"

The operation composer works by "zipping" two streaming operations into a single streaming operation.

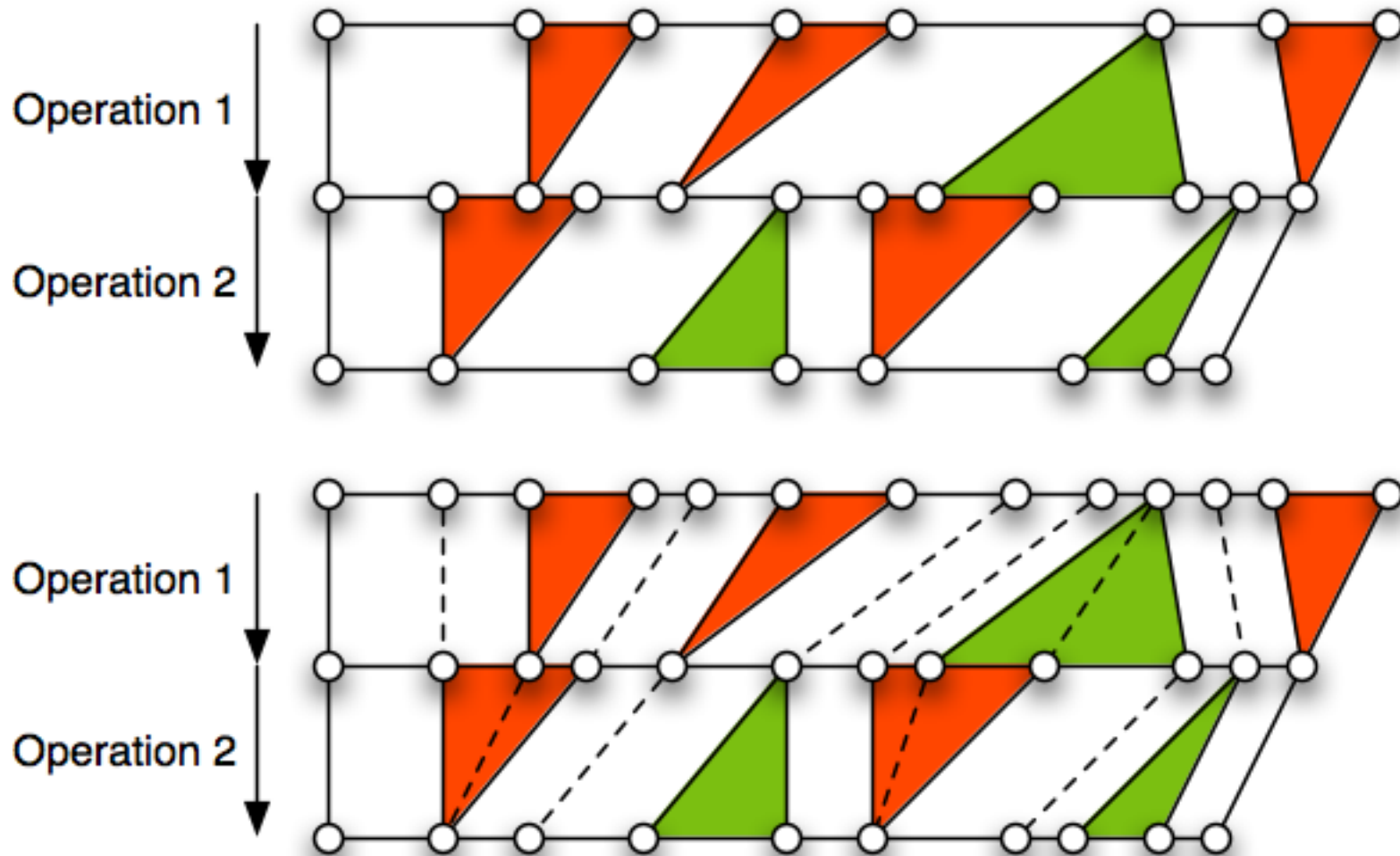


"Zipping" (continued)

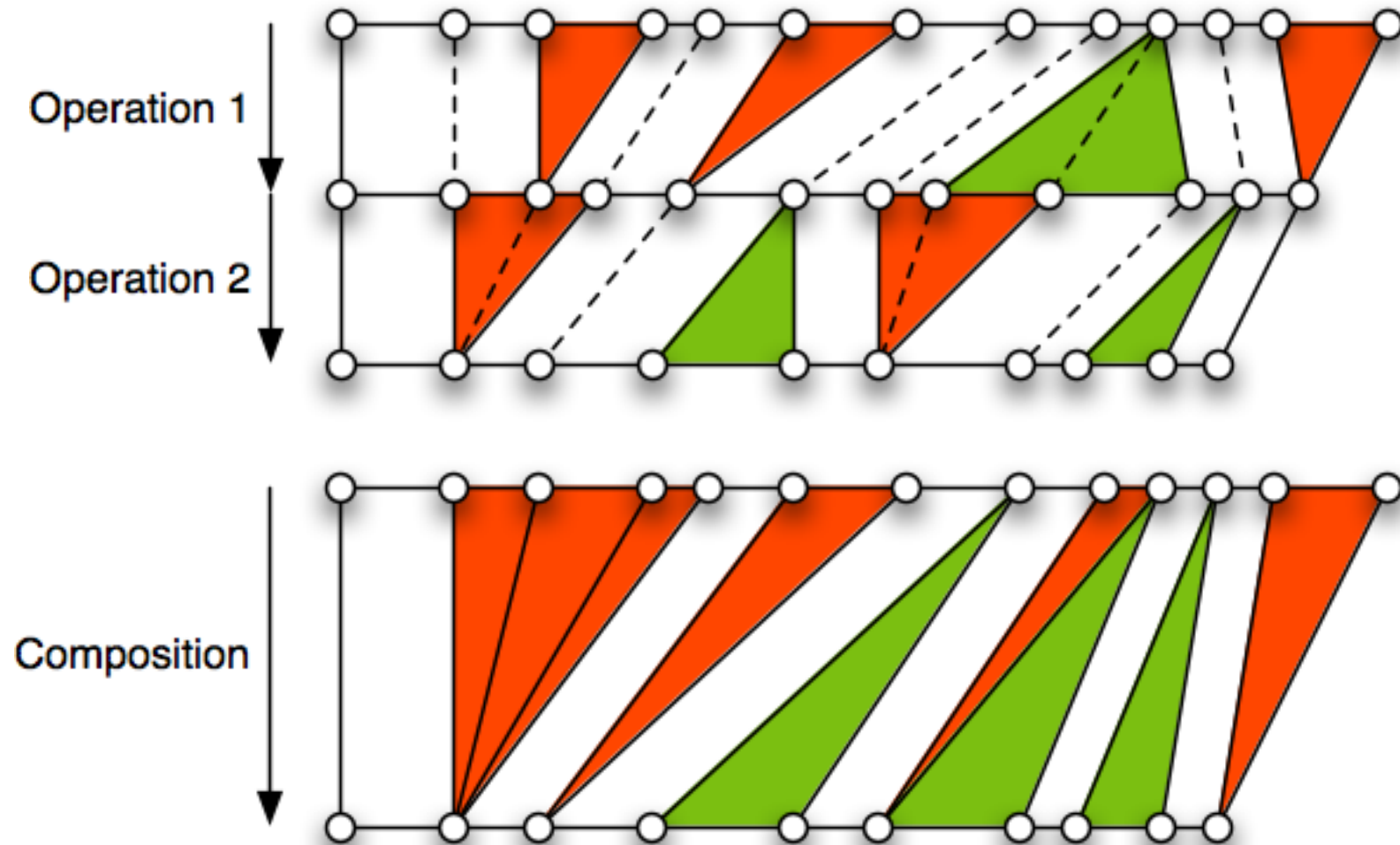
The operation transformer works by "zipping" two streaming operations into two streaming operations.



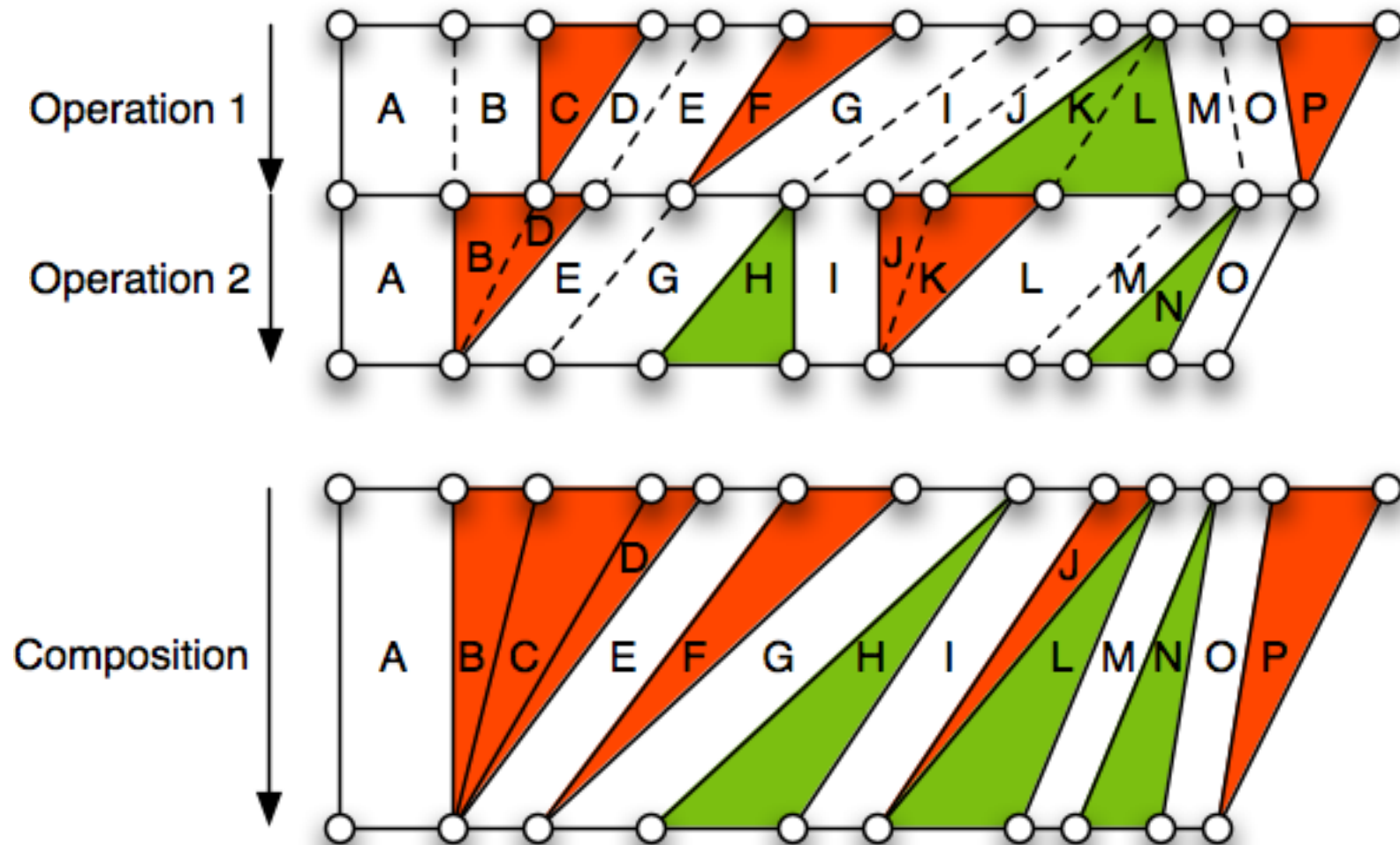
The composition algorithm illustrated



The composition algorithm illustrated



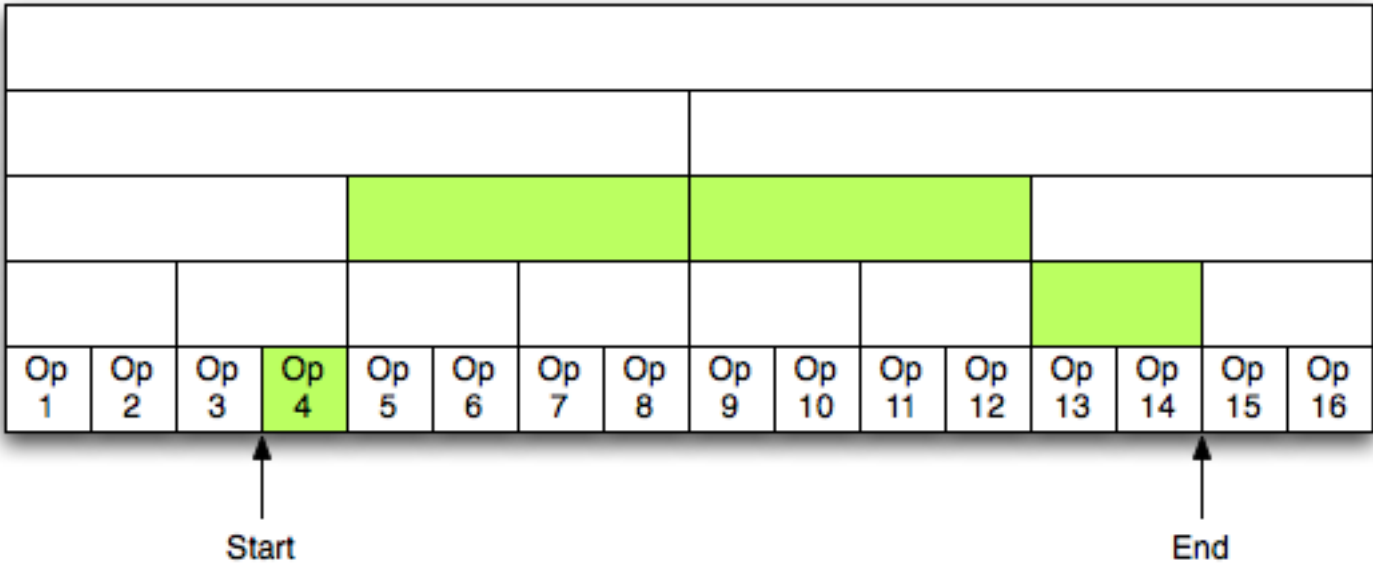
The composition algorithm illustrated



Composition tree

Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10	Op 11	Op 12	Op 13	Op 14	Op 15	Op 16

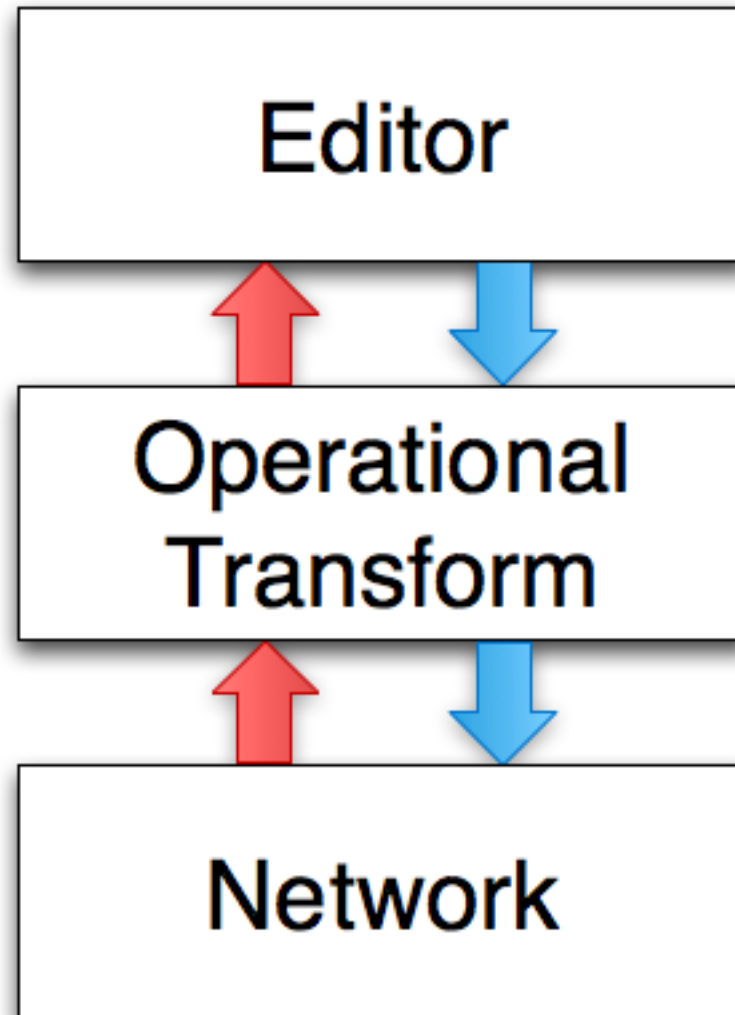
Composition tree





Editing Waves





Editor Goals

- High granularity extraction and application of operations
- Rich text and media
- Extensible (custom widgets, extensions)
- Mapping of an abstract Document Model to HTML
- Full control over concurrent rich text editing
- Multilingual (IMEs, RTL, etc)

Document Model

XML + Annotations

- Simple XML for structure
- Standoff annotations for style and metadata

<blip>

<p>

Hey have you seen this website?

</p>

<p>

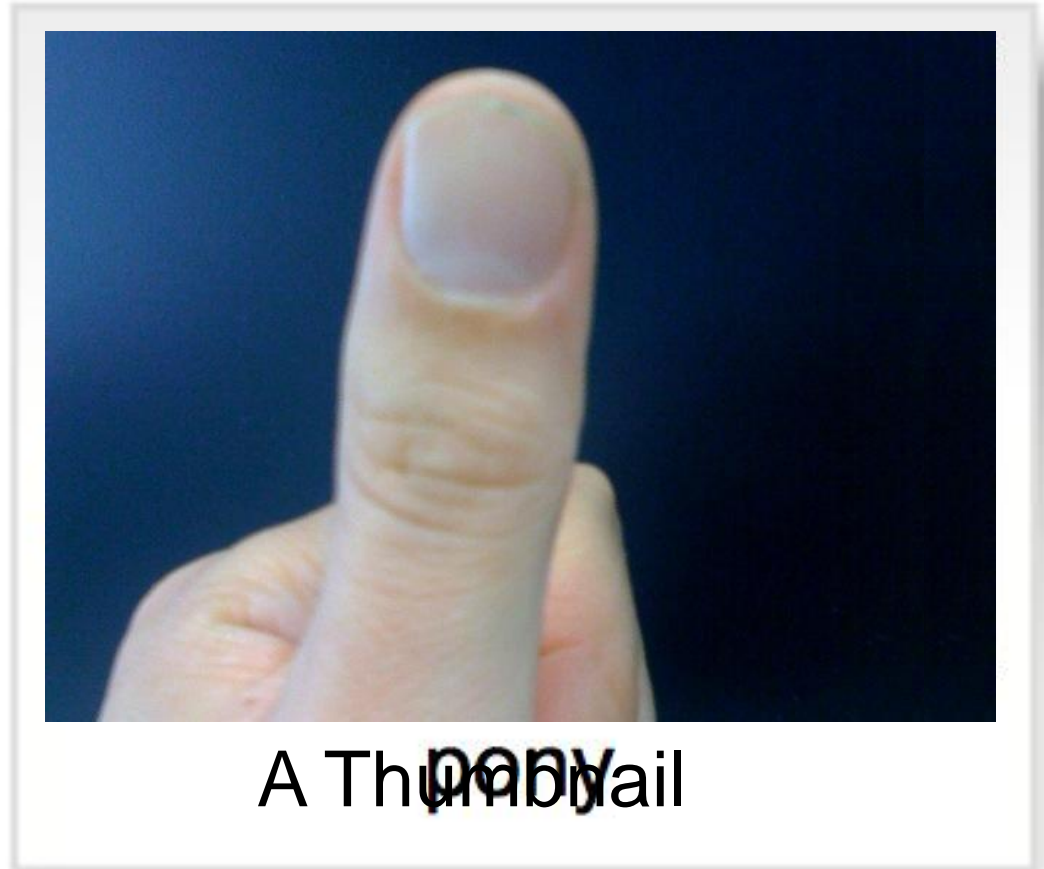
It's sweet!

</p>

</blip>

Image Thumbnail Example

```
<w:image attachment="...">  
  <w:caption>  
    A Thumbnail  
  </w:caption>  
</w:image>
```

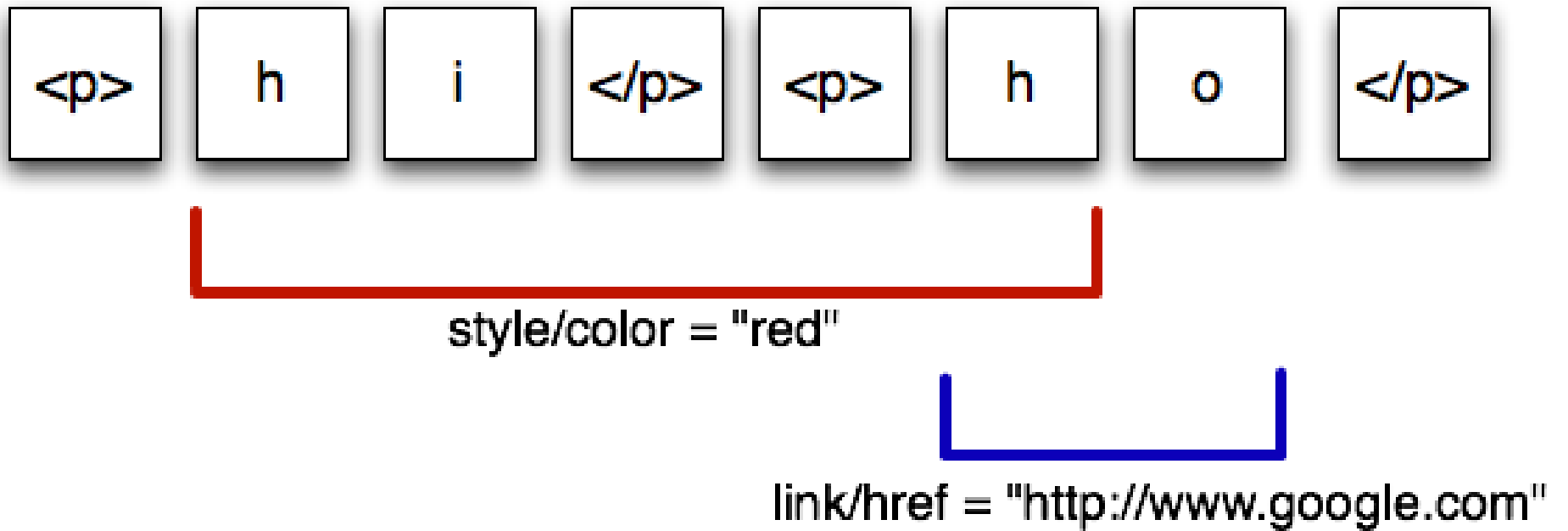


Content XML to Rendered HTML

```
<w:image  
attachment="...">  
  <w:caption>  
    pony  
  </w:caption>  
</w:image>
```

```
<table unselectable='on' cellpadding='0' class='ittt' cellspacing='0'>  
  <tbody>  
    <tr unselectable='on'>  
      <td unselectable='on' style='visibility: visible;' class='itco'>  
        <div unselectable='on' style='width: 120px; height: 79px;' class='itci'>  
          <a href='...' target='_blank'>  
            <img style='width: 120px; height: 79px;' src='...' class='gwt-Image J' />  
          </a>  
          <div style='display: none;' class='itcc' />  
          <div style='display: none;' class='pw'>  
            <div style='width: 0%;' class='pwa' />  
            <div style='width: 100%;' class='pwg' />  
          </div>  
        </div>  
      </td>  
    </tr>  
    <tr unselectable='on'>  
      <td unselectable='on' style='visibility: visible;'>  
        <div contenteditable='true'>pony<br /></div>  
      </td>  
    </tr>  
  </tbody>  
</table>
```

Annotations



Appears as:

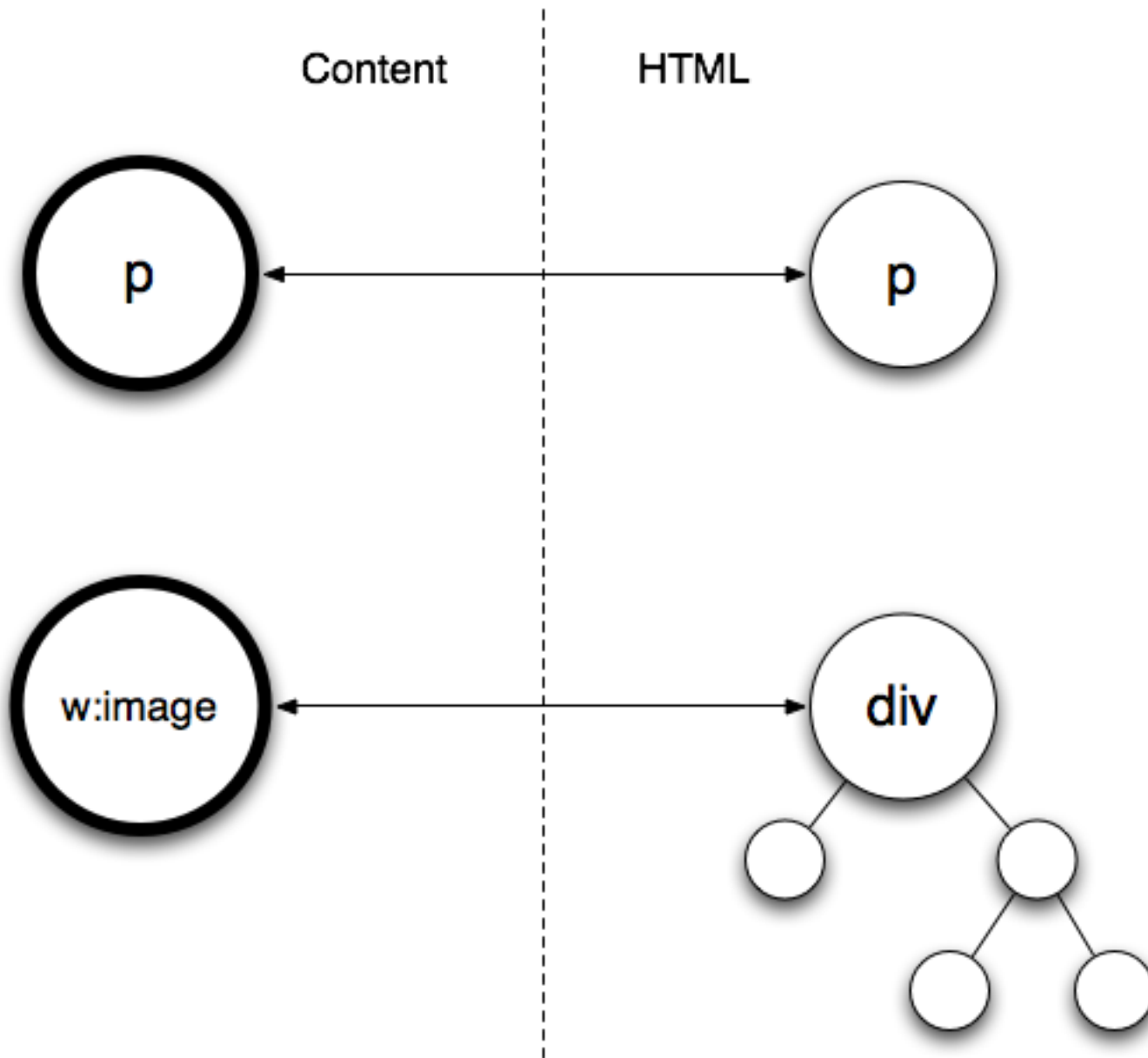
hi
ho

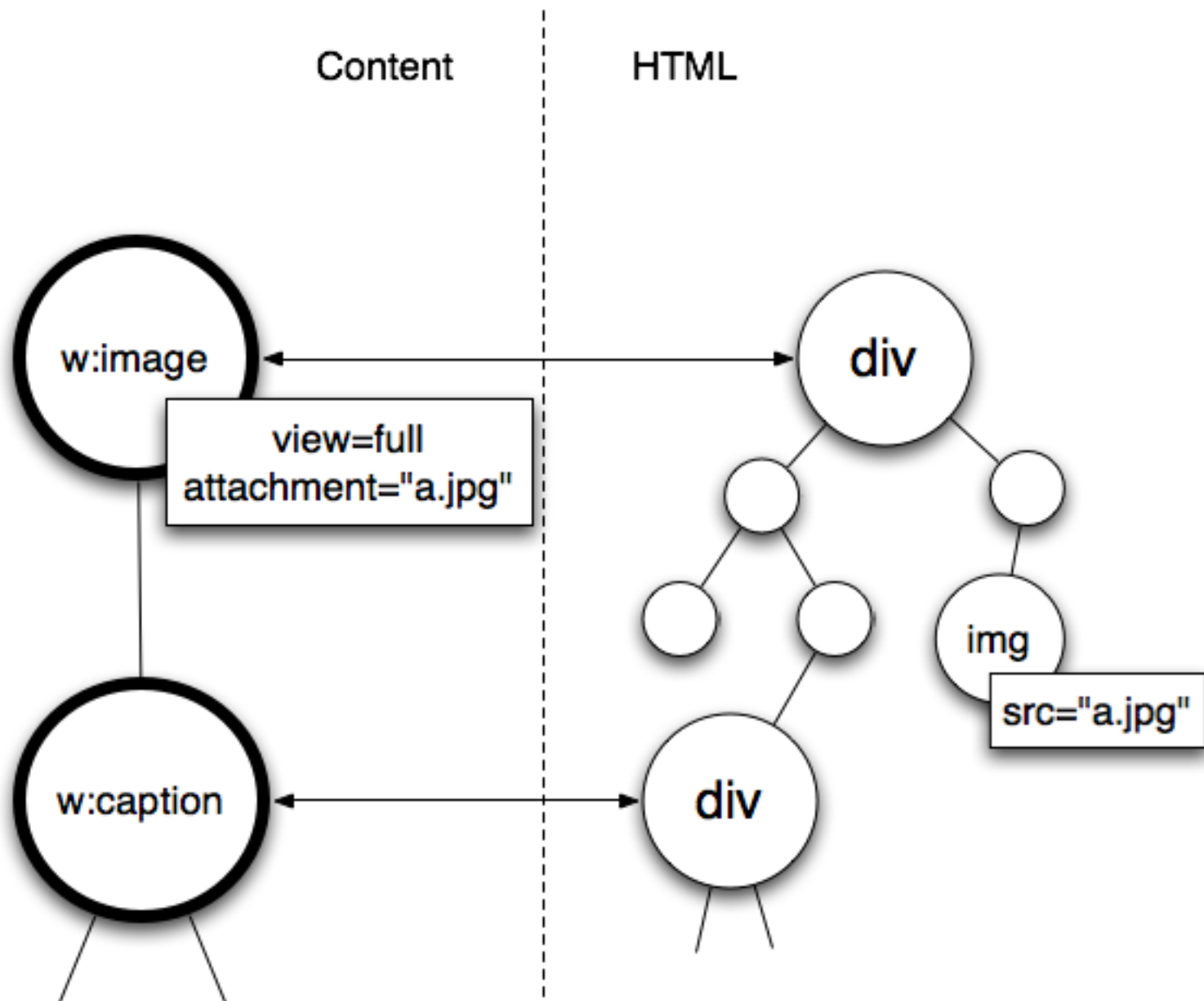
Properties of Annotations

- Don't affect the structural content or its operations
- Clients may choose to ignore some or all annotations

Some uses for annotations

- Other users' cursor position, selection
- Rich link annotations
- Robot-specific data
- Diff highlighting (using local annotations)
- May reference structured data in another document
 - E.g. spelling suggestions and state





<blip>

<p>

Hey haev you seen

</p>

<p>

this website? And some

</p>

<p>

styled text

</p>

</blip>

```
<blip>
  <p>
    Hey haev you seen
  </p>
  <p>
    this website? And some
  </p>
  <p>
    link/manual =
    "http://www.google.com/"
  </p>
  <p>
    styled text
  </p>
</blip>
```


spell = "id1"

style/fontWeight = "bold"

style/fontWeight = "italic"

Hey haev you seen
this website? And **some**
styled text

l:p - "paint"
l:xyz - "boundary"

Hey haev  you seen
this website? And **some**
styled text

<blip>

<p>

Hey <l:p hover="...">haev</l:p><l:spell/> you seen

</p>

<p>

<l:p href="...">this</l:p> website?

And <l:p fontWeight="bold">some</l:p>

</p>

<p>

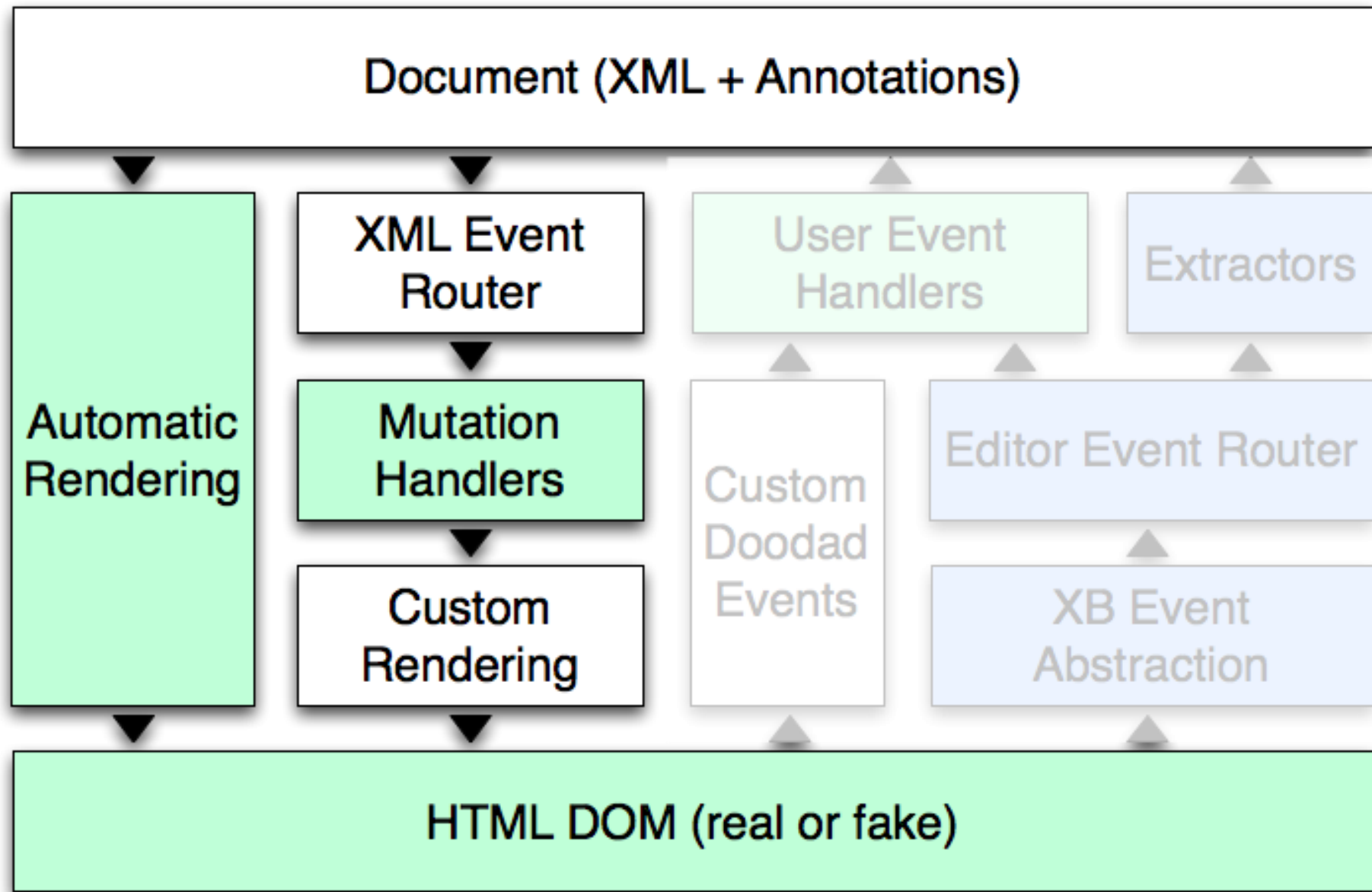
<l:p fontWeight="bold" fontStyle="italic">styled</l:p>

<l:p fontStyle="italic">text</l:p>

</p>

</blip>

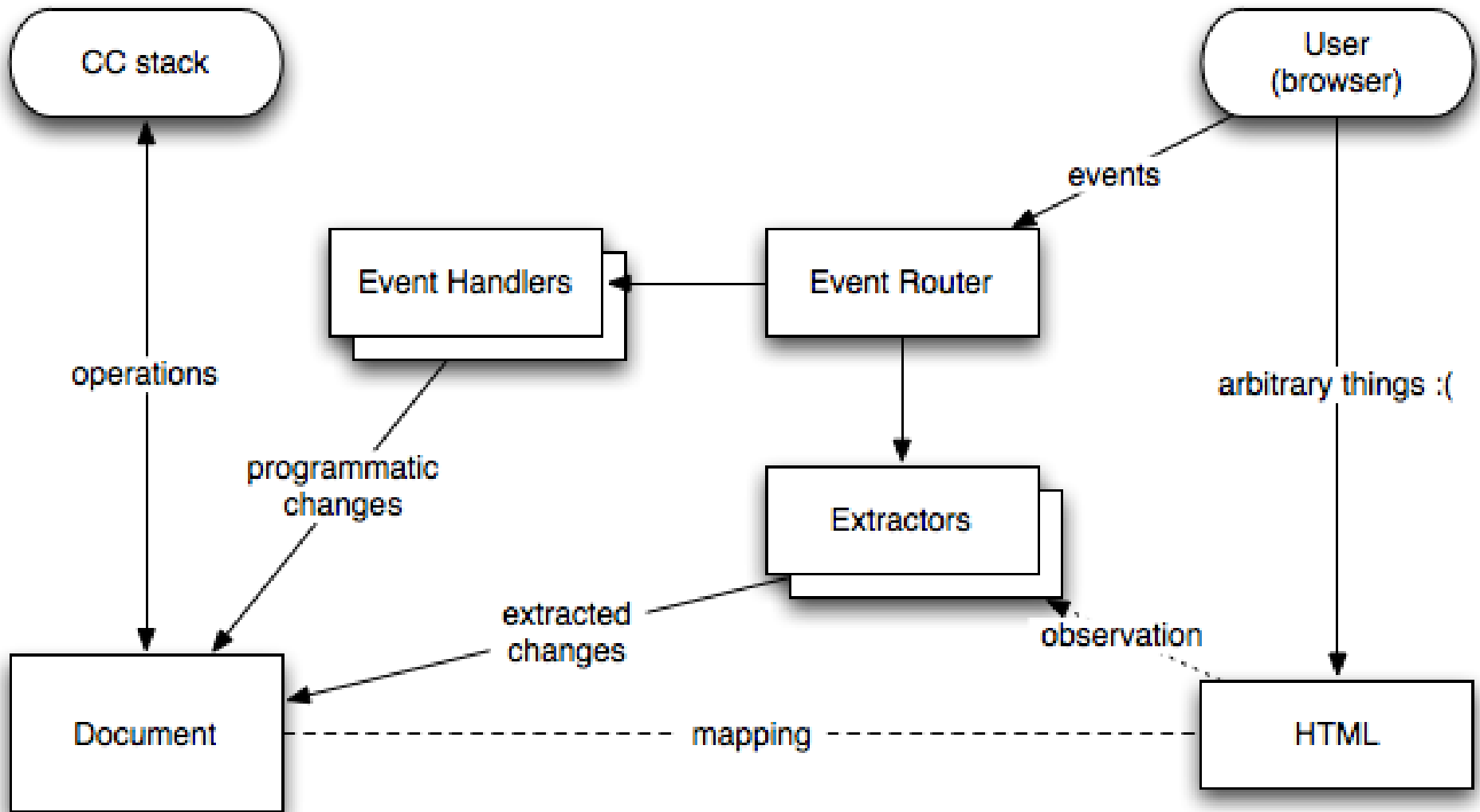
Rendering the Document Wave Model



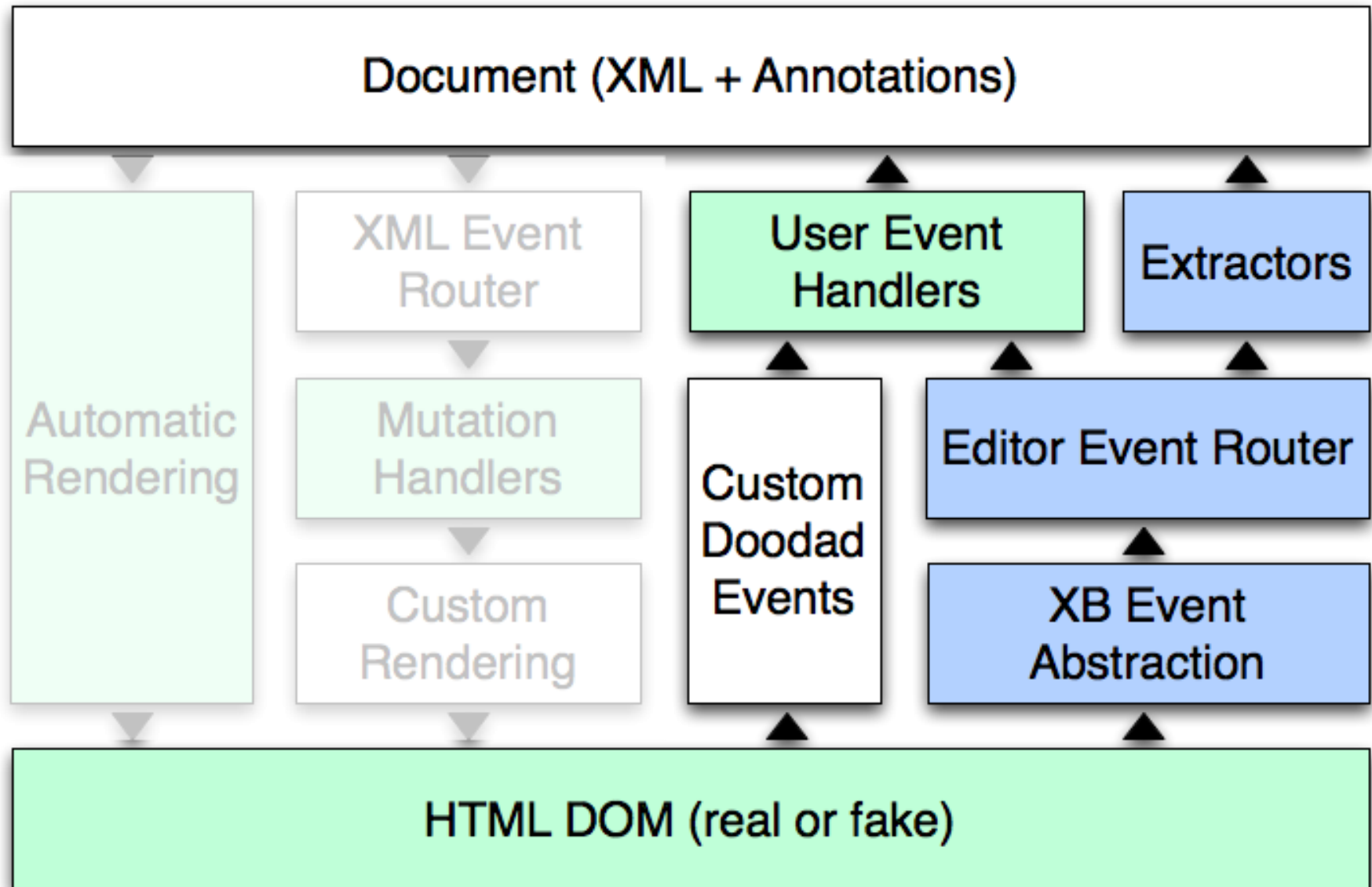
■ Editor ■ Used by editor ■ Non-editor




Extracting Operations

Editor Interactions Overview

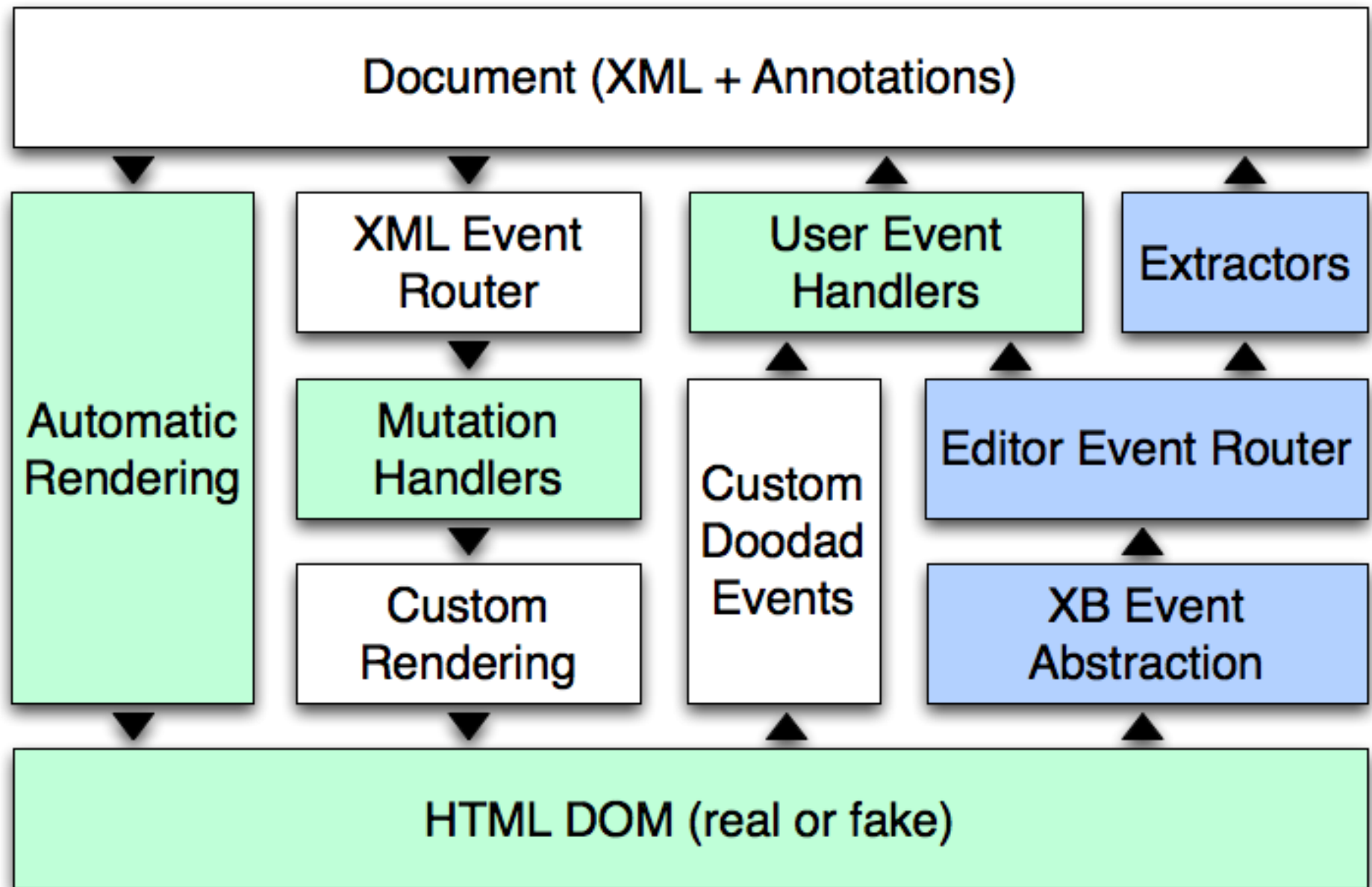


Editor within the Document Wave Model



 Editor  Used by editor  Non-editor

Editor within the Document Wave Model



■ Editor ■ Used by editor ■ Non-editor

Summary

- Arbitrary separation of document model from rendering
- Balance between maintaining control, and leveraging native support.



Natural Language Processing



Text: Not So Plain Anymore






Google Wave is an ideal platform for building smarter tools

- Structure
- Collaboration
- Liveness
- Hosting / The Cloud

Spelling Correction

The typist's dream: concentrate on writing, let the spelling mistakes sort themselves out!

Interacting With Spelling Suggestions

 me: It started  with a simple idea   

Draft **Done** Cancel

- started
- startled
- stared
- starled

Representing Spelling Suggestions

```
<spell>  
  <suggestion type="original">starled</suggestion>  
  <suggestion score="2.6">started</suggestion>  
  <suggestion score="1.2">startled</suggestion>  
  <suggestion score="0.6">stared</suggestion>  
</spell>
```

Spelly, the spelling robot

Listen to changes on waves

Check spelling

Add suggestions into the wave

Lunch tomorrow



Lunch tomorrow

2:23 pm

Hey how's it going? I was wondering if you were up for lunch?



me: Let's met tomorrow morning|

2:32 pm

Draft

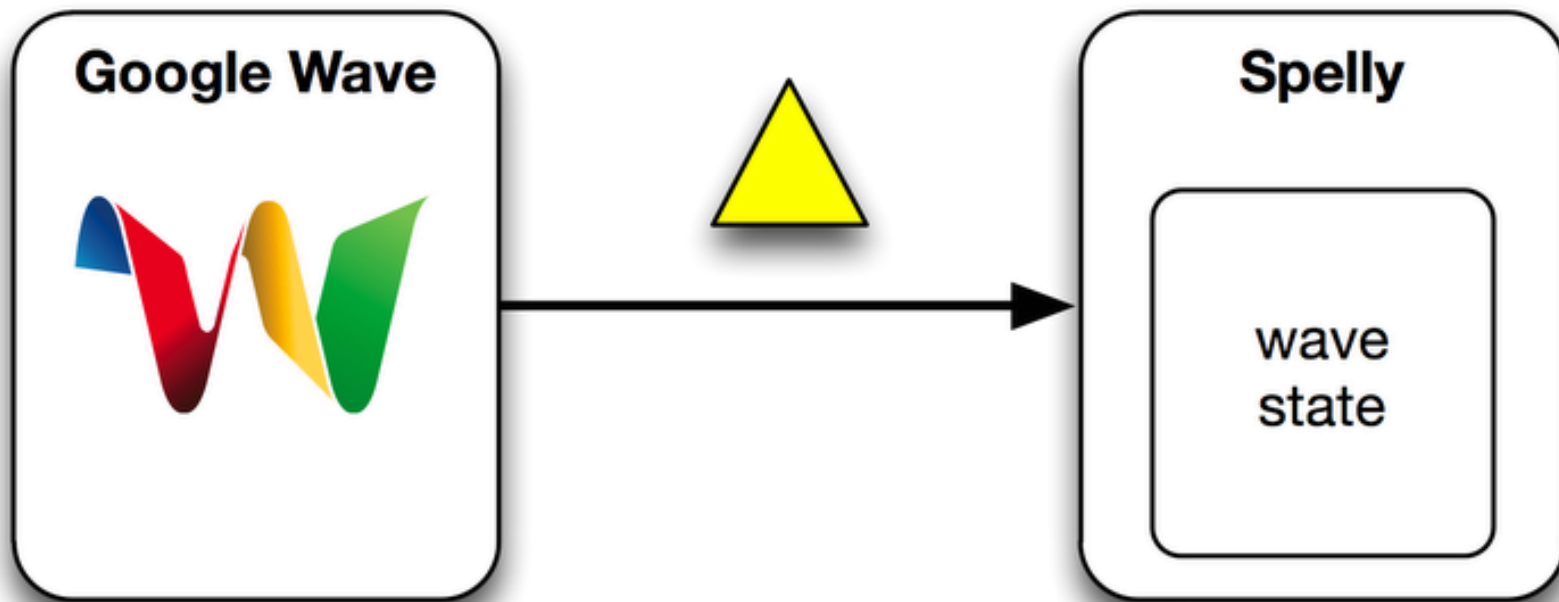
Done

Cancel



= insert: "et tomorrow mo"





Let's | met | tomorrow | mo

Let's **met** tomorrow

meat

meet

net

get

me

...

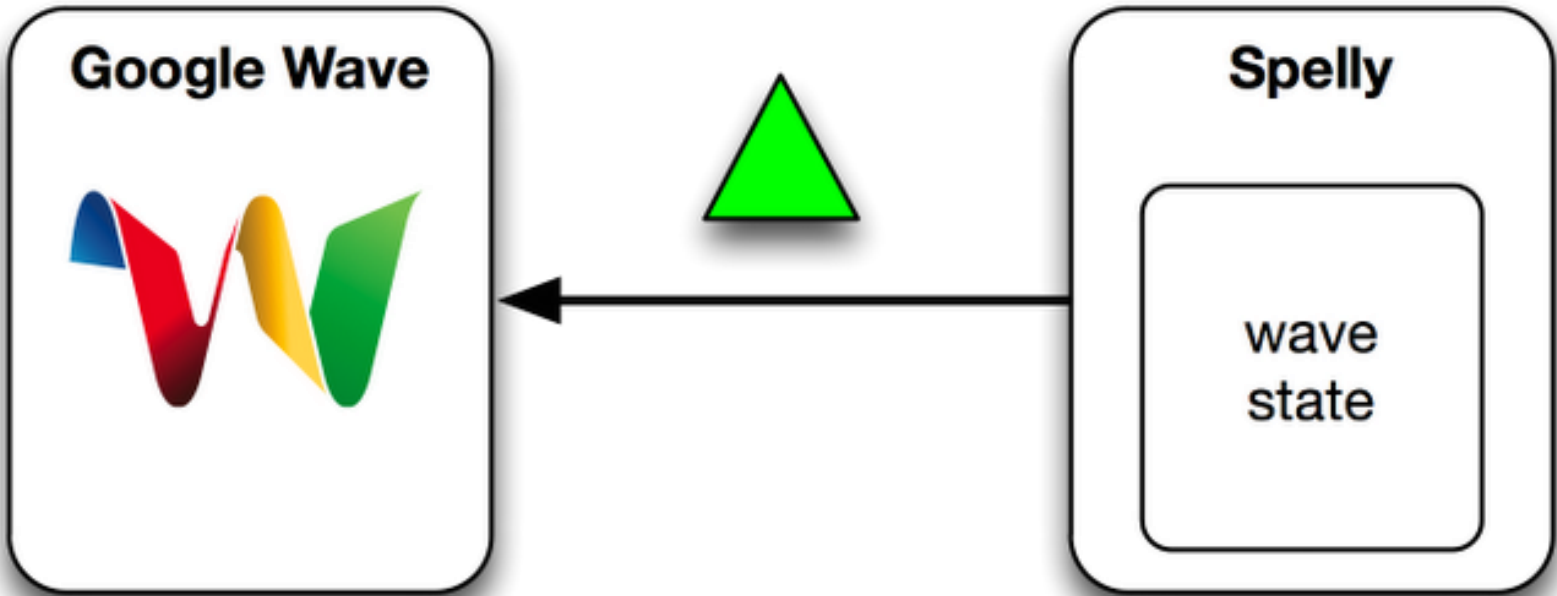
Let's met tomorrow
Let's meat tomorrow
Let's meet tomorrow
Let's net tomorrow
Let's get tomorrow
Let's me tomorrow
Let's ... tomorrow

Let's **meet** tomorrow



= annotate "met" with:

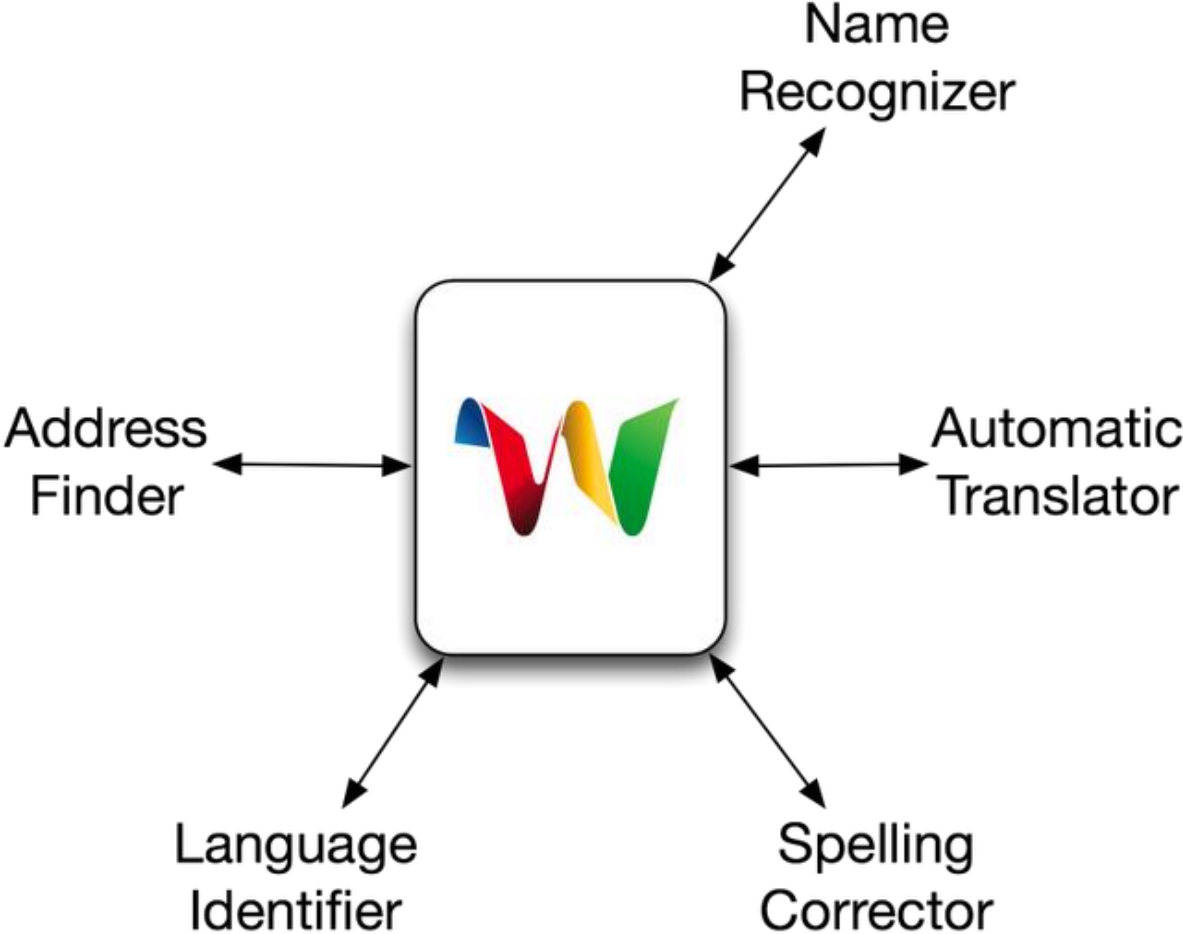
```
<spell>  
  <suggestion score="3.2">meet</suggestion>  
  <suggestion score="2.7">met</suggestion>  
  <suggestion score="1.2">set</suggestion>  
</spell>
```



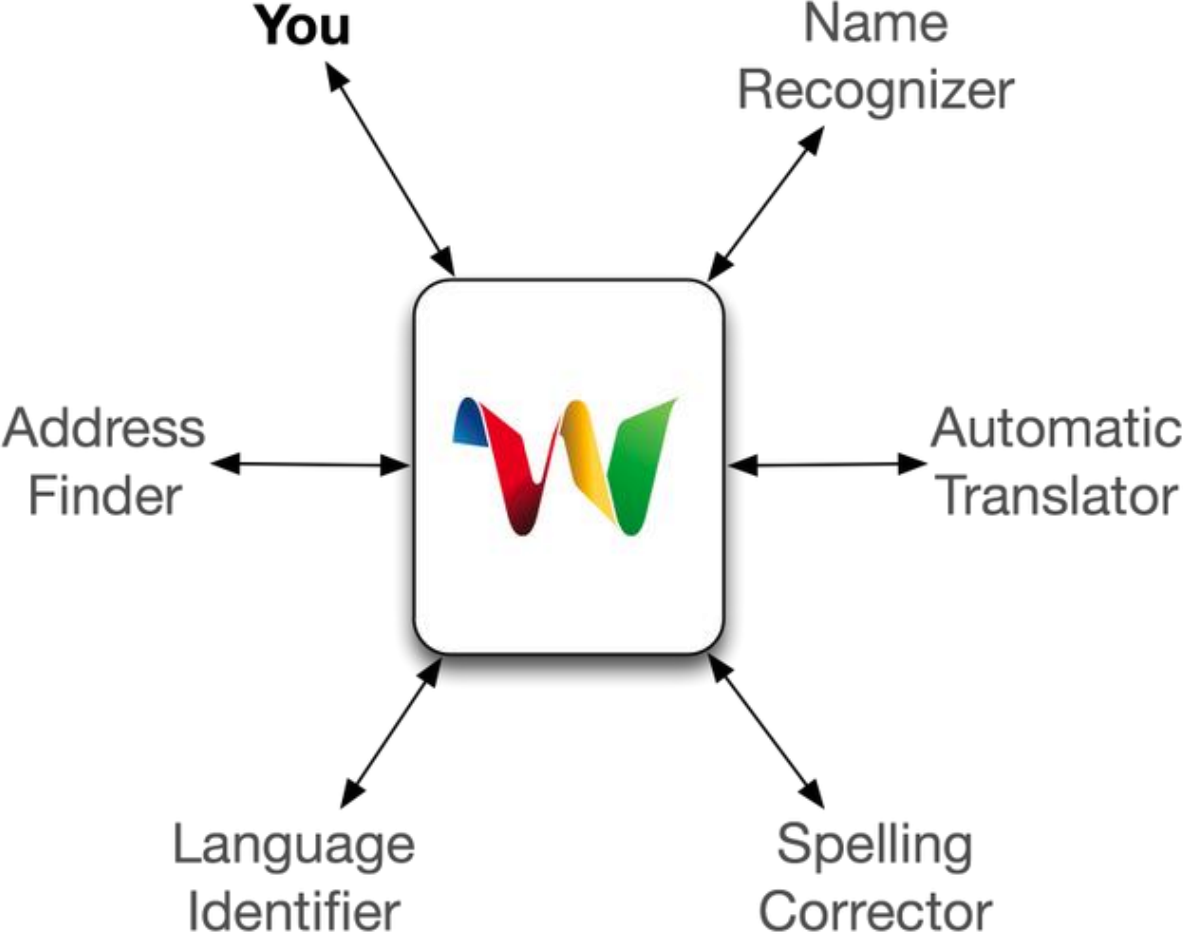
Robots in the Cloud

- available from any device
- more data → better quality
- shared resources → more efficient
- interactive → positive feedback and learning

Collaborative Robots



Collaborative System



Google™

