WOS(Web of Services)

- Open API and Mashup -

2007.3.6

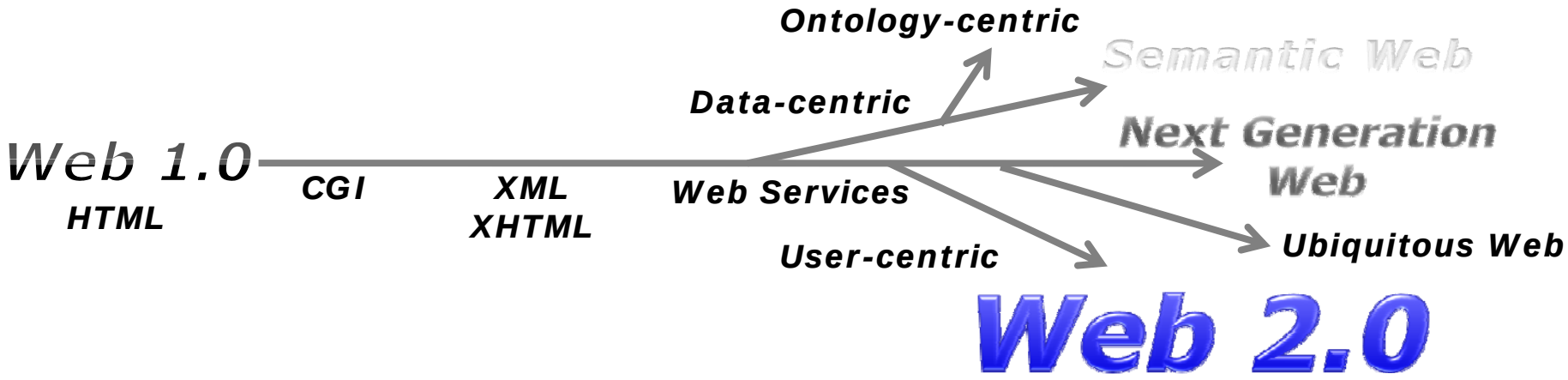
ETRI
www.etri.re.kr

한국전자통신연구원 표준연구센터 서비스융합표준연구팀
전 종 홍

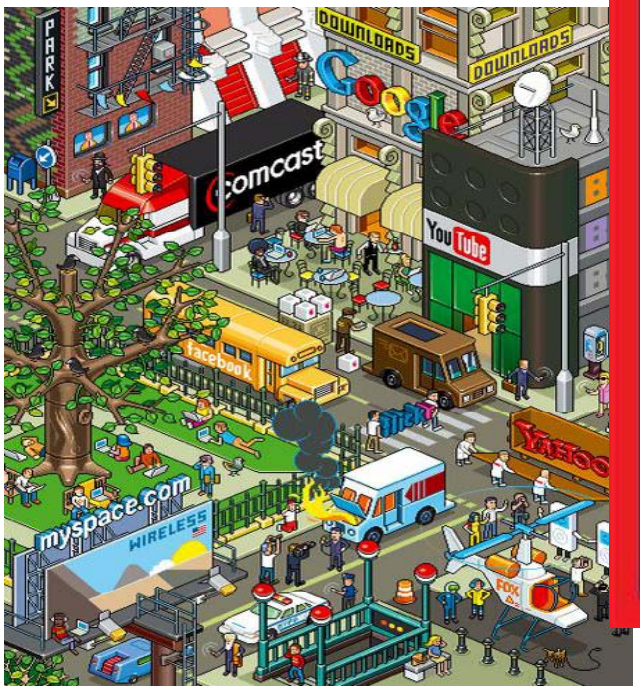
Email: hollobit@etri.re.kr
Blog: http://blog.webservices.or.kr/hollobit
Tel : 042-860-5333

http://blog.webservices.or.kr/hollobit/presentation/TTA-mashup-hollobit.pdf

Age of Ubiquitous Web

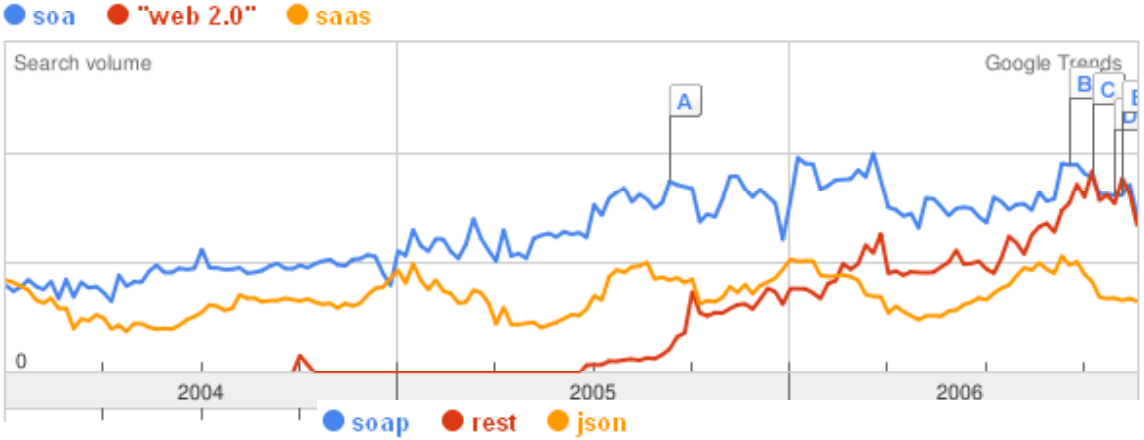


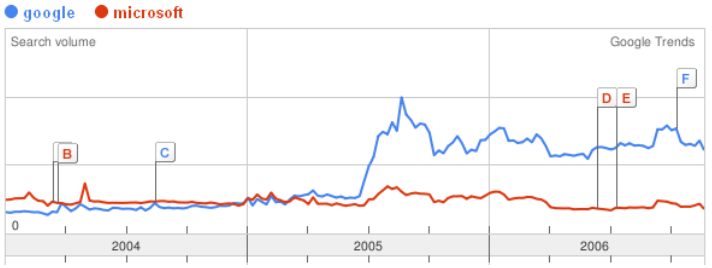
Age of “2.0”

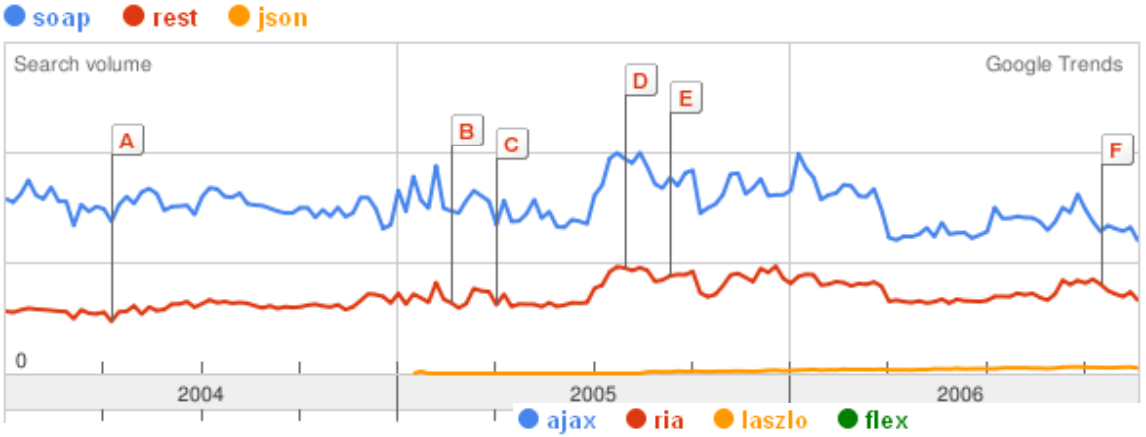


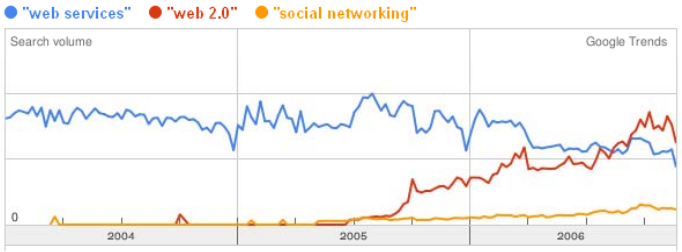
1996 2006

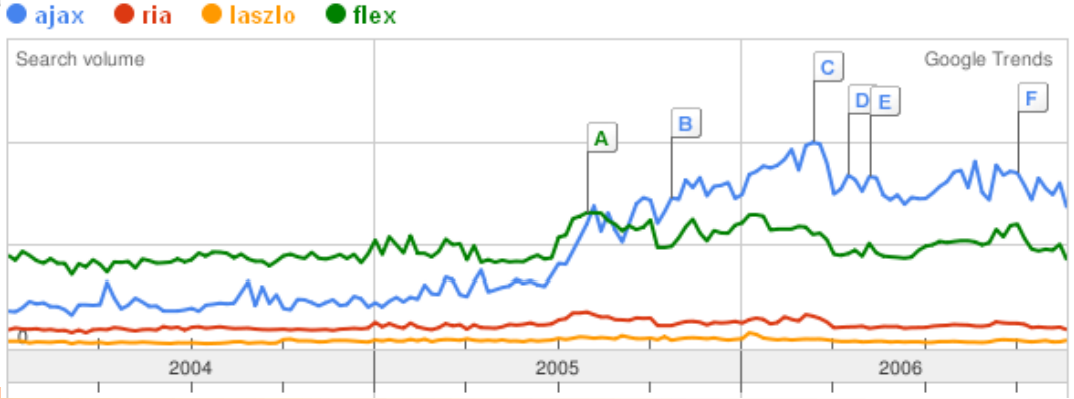
Age of “2.0”









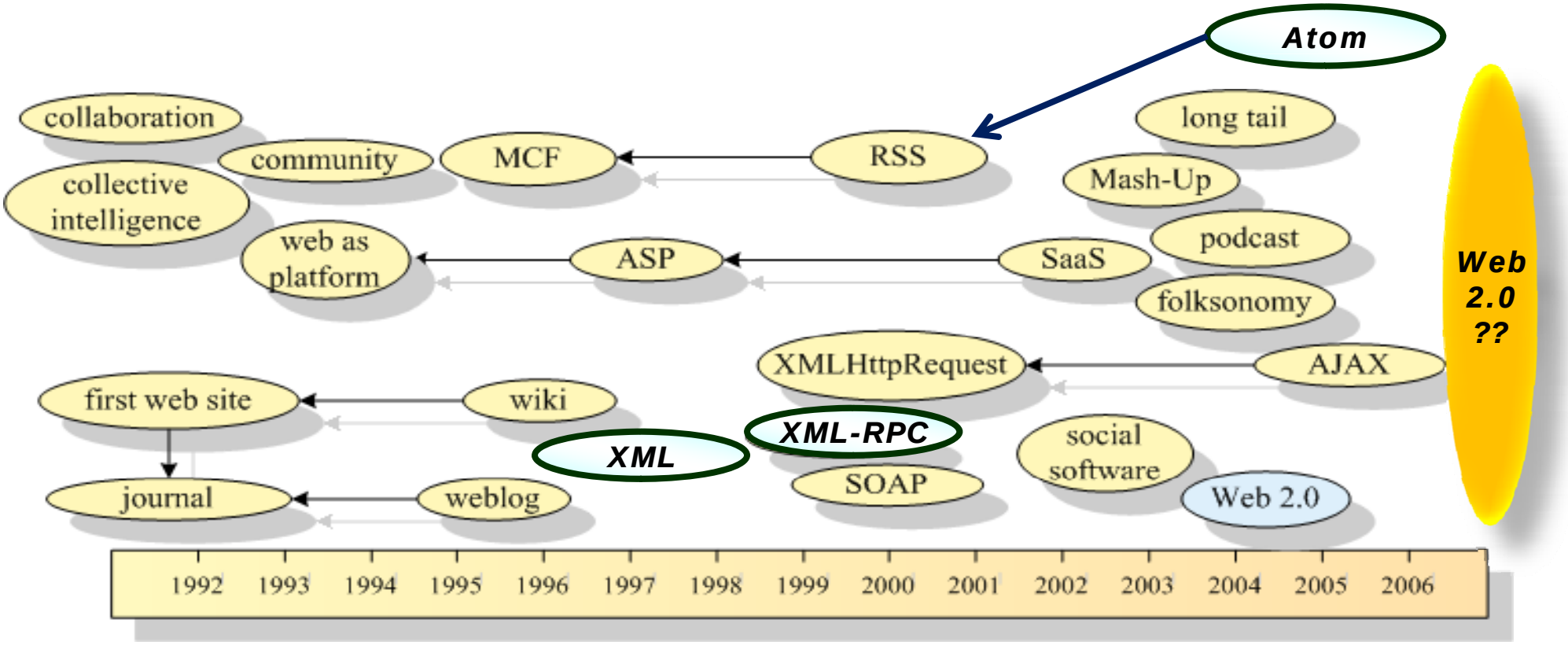


http://www.google.com/trends

웹2.0 변화를 일으키는 6가지 원동력

- 글로벌화된 고객들
  - 2005년말 인터넷 사용자는 10억명 (8억5천은 늘 사용)이며, 북미23%, 유럽24%, 아시아 36%로 전세계적으로 증가 추세임.
- 초고속 인터넷의 확산과 인터넷의 생활화
  - 전세계적으로 초고속 인터넷 환경이 50%에 육박하고 있음. (2006년부터 광대역 통신망의 시대로 전환 중)
- 언제 어디서든 접속할 수 있는 환경
  - 2006년 초를 기준으로 PC 인터넷 사용자의 두배에 해당하는 20억명의 휴대폰 사용자가 있으며, 이중 28%는 어디서든 인터넷 접속이 가능
- 고객은 접속하는 것뿐 아니라, 직접 참여하고 있음
  - 미국의 성인의 50% 정도는 온라인 콘텐츠를 새롭게 만드는데 기여하고 있음 (2006년4월 기준으로 5천만개 블로그가 있으며, 매일 17만개의 새로운 블로그가 생기고 있음. 동영상 공유 사이트인 유튜브에서는 하루 1억개 이상의 비디오를 제공)
- 인터넷 산업의 생산비용은 급격하게 줄고 있음
  - IT 인프라 단가는 지난 6년 동안 72% 이하로 싸졌음. (과거 고가 워크스테이션과 고가 운영체제 및 소프트웨어를 사용하던 환경에서 PC 서버와 오픈 소스 소프트웨어 환경으로 바뀌고 있음)
- 새로운 수익 모델과 기회의 등장
  - 미국 내에서 2006년 한해 온라인 광고 시장 성장률은 37%

Web 2.0, Is it new technology ?



http://en.wikipedia.org/wiki/Image:Web20buzz.png

Directions for Technical Innovation

1. 콘텐츠 유통과 상거래 방식의 변화
   - Blog, RSS/Atom, UCC

2. 브라우징 방식의 변화
   - Tagging, Social Browsing, Microformat

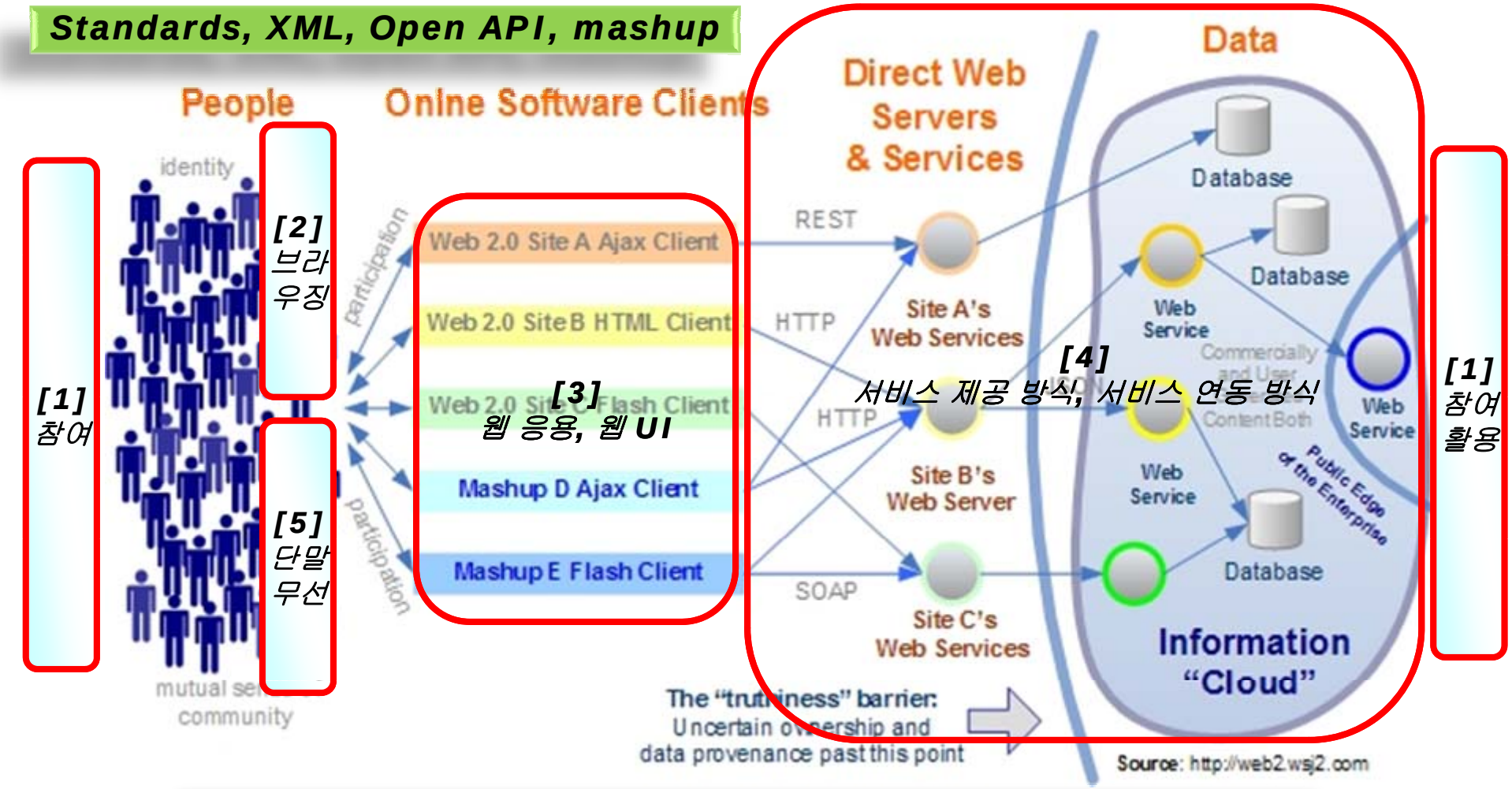
3. 웹 응용 환경의 변화
   - Weblication, RIA, AJAX, Browser 확장 기술, XUL

4. 서비스 제공 방식의 변화
   - Open API, 웹서비스, REST, Mashup, SOW

5. 디바이스의 변화
   - MWI, UMPC, Mobile Web 2.0

➢ Standards, XML, Open API, mashup 이 핵심 키워드

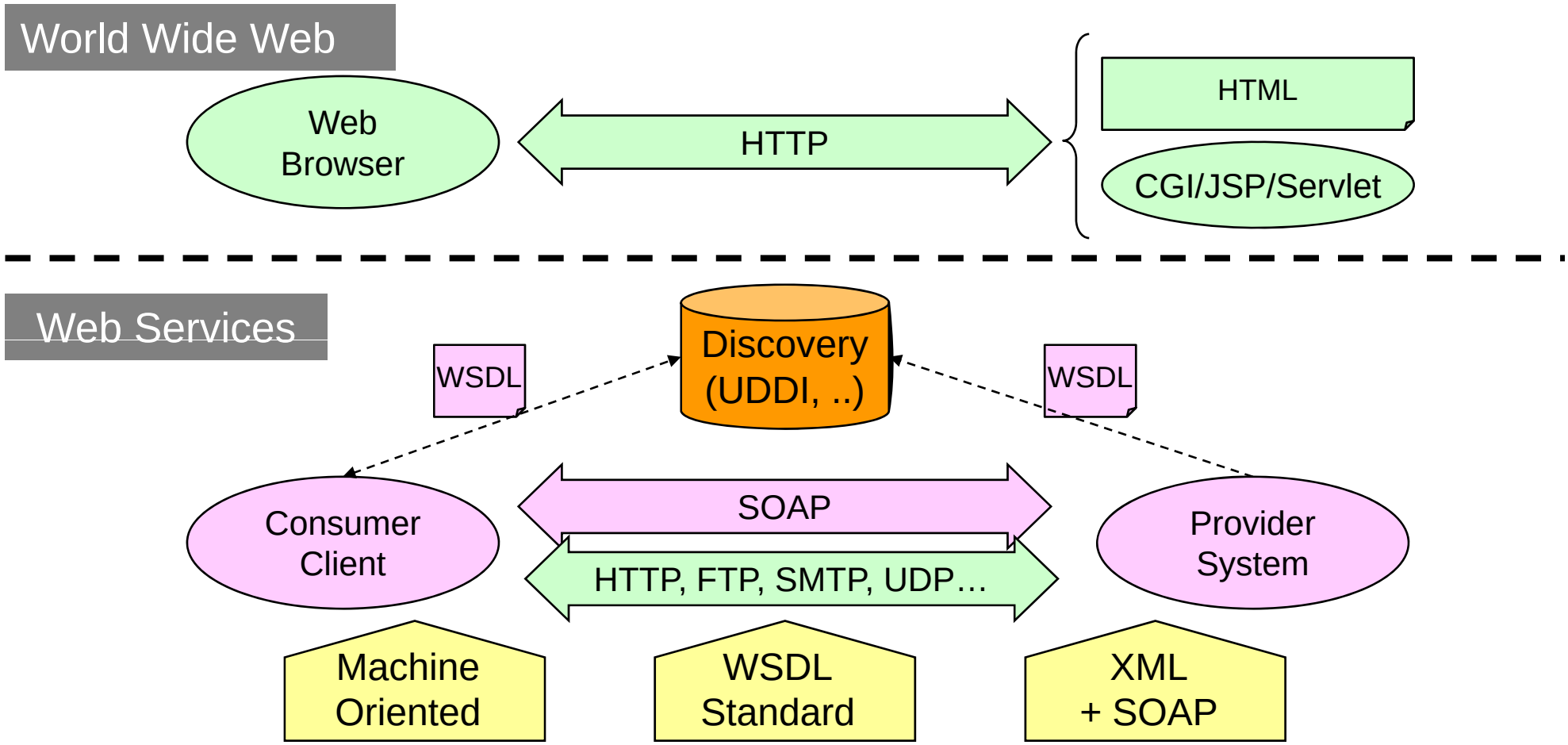
Directions for Technical Innovation



Internet Technology, Platform Technology, IP Convergence …
(Google Platform, GRID, Skype, Youtube, …)

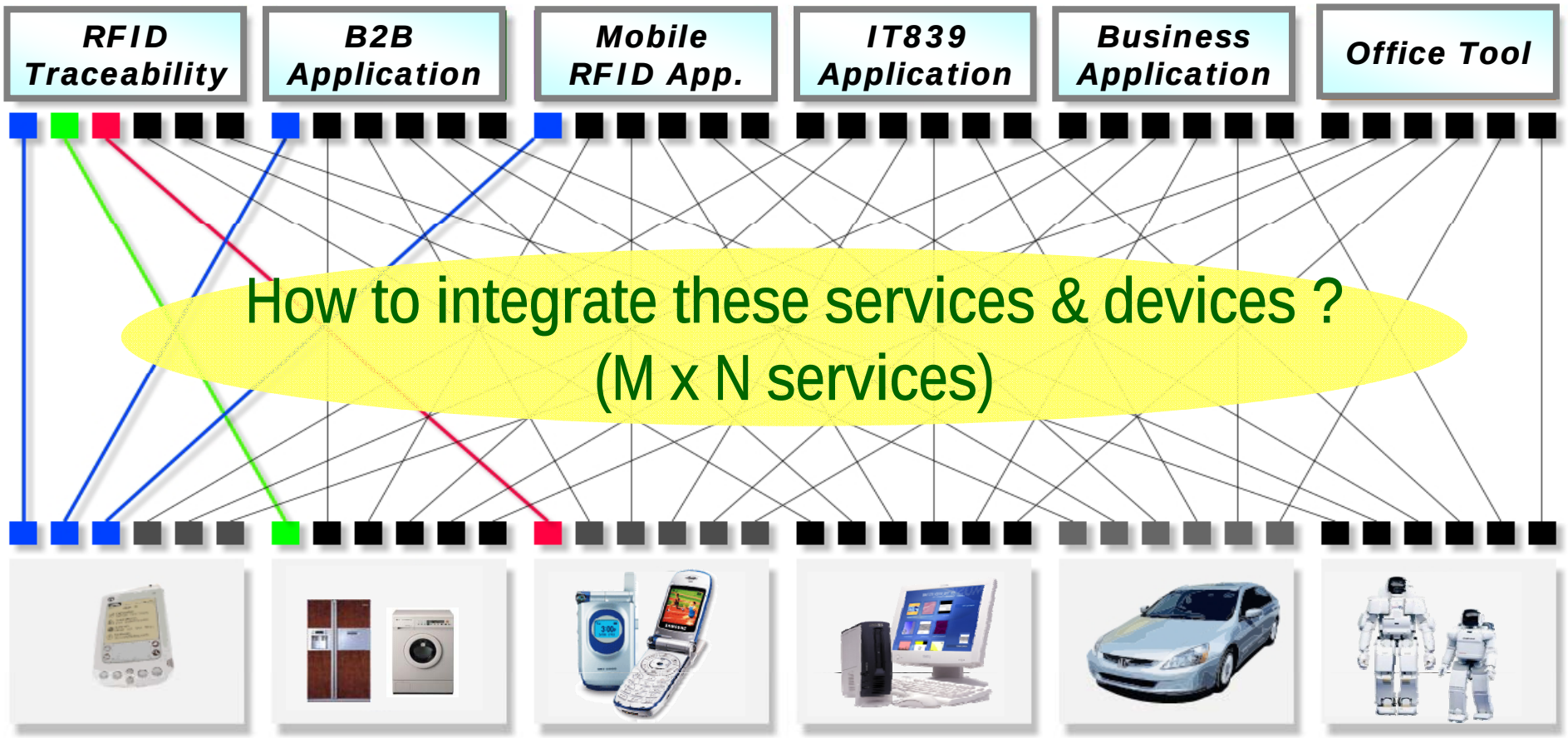
Why Web Services ?

Web service & Web Services



Why we need Web Services ?

M applications...



N devices

어떻게 많은 연결들을 가능하도록 할 것인가?

Why Web Services ?

- Extend the reach of your Service
  - Enabling users to consume data in their own way e.g. Flock
  - Beyond the browser: mobile, desktop, etc.
  - Integration (Internal / External / Partners)
  - Prepare for future use cases

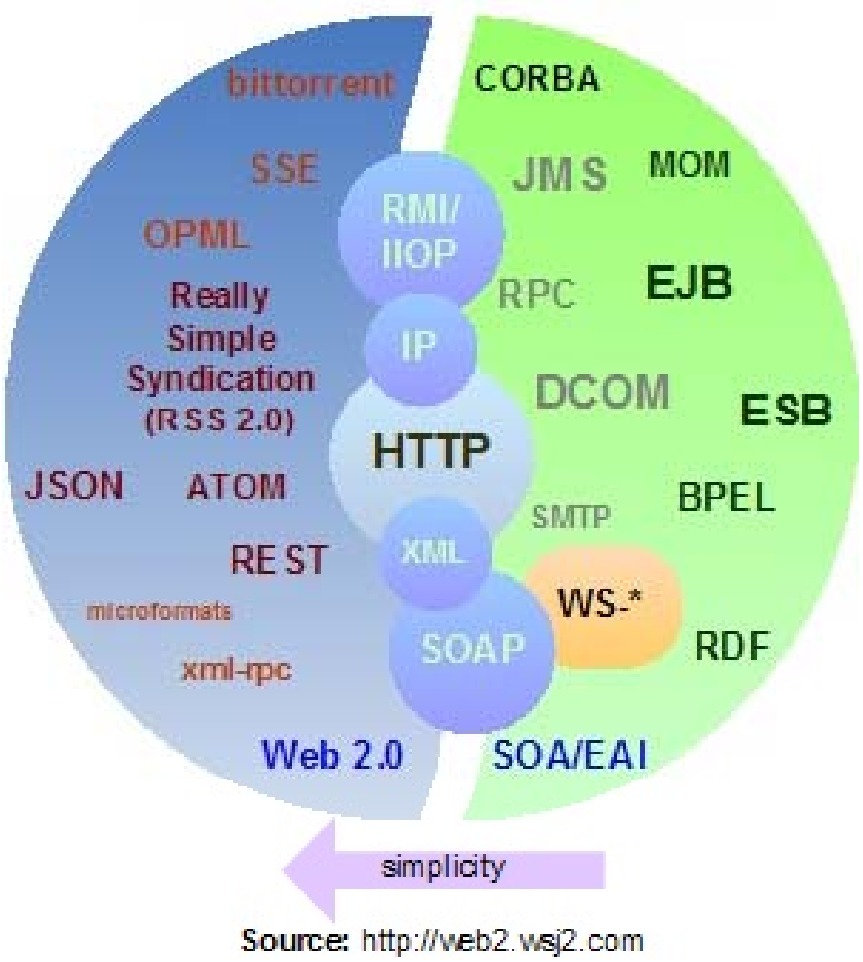
"The electric outlet in the wall is, to stretch the metaphor, an API. A manufacturer making a product that uses electricity can equip it with a plug that fits into the socket."

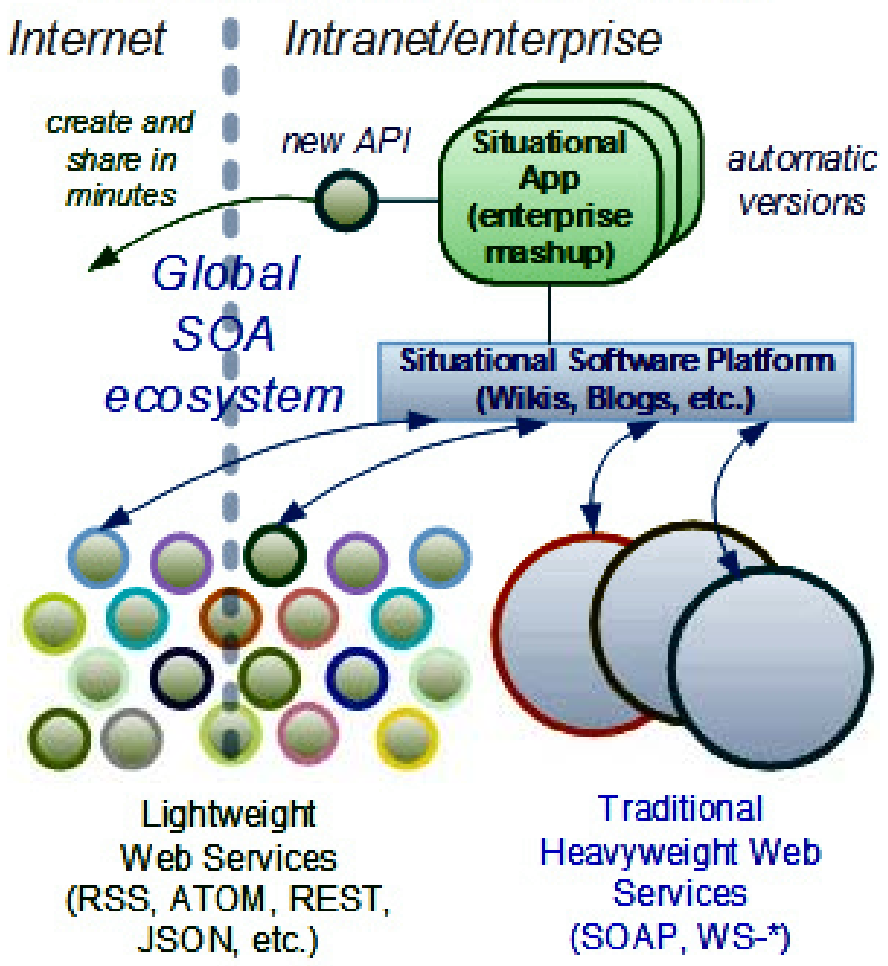
- Dan Gilmoor

Traditional Web APIs or Web 2.0 APIs

	Traditional Web APIs	Web 2.0 APIs
선도 기업	IBM, Microsoft	Google, Yahoo, Amazon, eBay,...
주요 기술	SOAP, Web Services	REST, XML
데이터 유형	HTML + (일부 XML 교환용 데이타)	다양한 XML (RSS, Atom, RDF, Microformat ...), JSON
기술 선도	기업	사용자
개발 프레임워크	Business Application Framework	Web Application Framework
개발 참여자	소수 개발자	다수 사용자
기술 복잡도	복잡	단순
개발 속도	저속	고속
기술 관점	Application Centric	Platform Centric
연동 방식	System Integration	Mash up
통합 방식	수직적 통합	협력적 활용

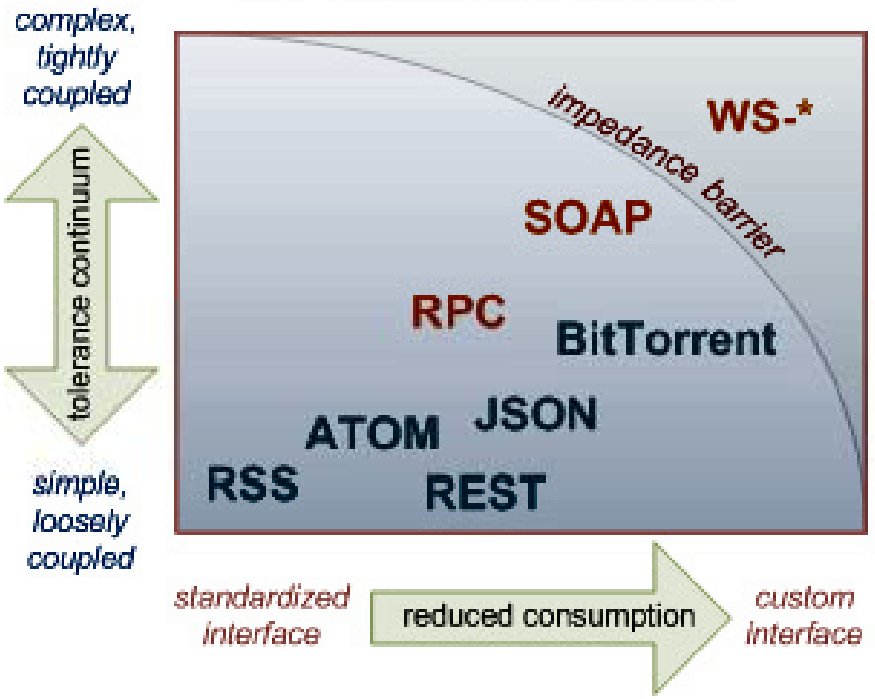
Traditional Web APIs or Web 2.0 APIs





Styles of APIs and Web Services

- REST, yREST or RESTful
- RSS/Atom
- SOAP (+ WSDL, WS-*, Web Services ...)
- JavaScript, ActionScript APIs
- HTTP GET or POST



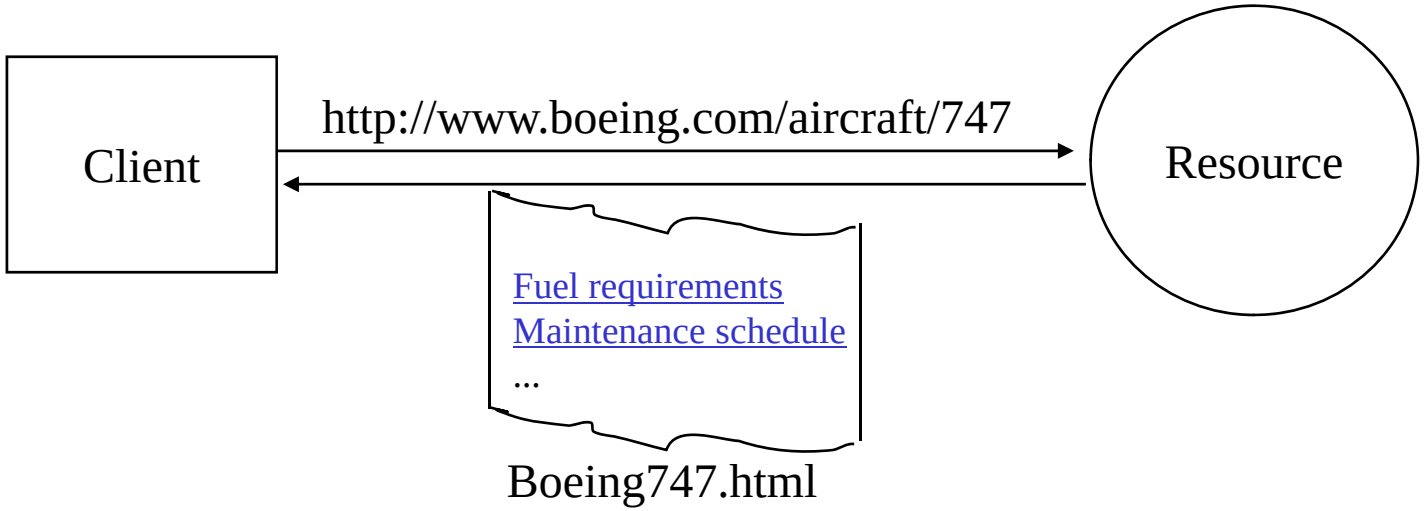
Source: http://web2.wsj2.com/

Styles of APIs - REST

- REST = Representational State Transfer
  - a collection of architectural principles for large scale distributed systems
  - first presented in Roy Fielding's doctoral dissertation about the web
  - the principles underlying HTTP (Roy Fielding was one of the principal authors of HTTP)
- An architectural style is
  - Not a protocol, Not a specification
- Who is Roy Fielding - Co-author of the HTTP and URI RFCs
- Structured around verbs (only a few) and nouns (many)
  - nouns identified a resource (URI or URL in http)
  - verbs are "generic" actions of those resources: GET, POST, PUT, DELETE
- Two Types of REST
  - Pure REST : Based on Roy Fielding's principles
  - Popular REST : Generic XML over HTTP (not SOAP)

Styles of APIs - REST's Actions

- REST's Actions
  - GET: obtained the state of a resource
  - POST: update the state of a resource (non idempotent)
  - PUT: replace the state of a resource with a new version (idempotent)
  - DELETE: delete the resource
- Example REST Services
  - Amazon : http://developer.amazonwebservices.com/
  - Yahoo! : http://developer.yahoo.com/search/rest.html



Styles of APIs - Web Services

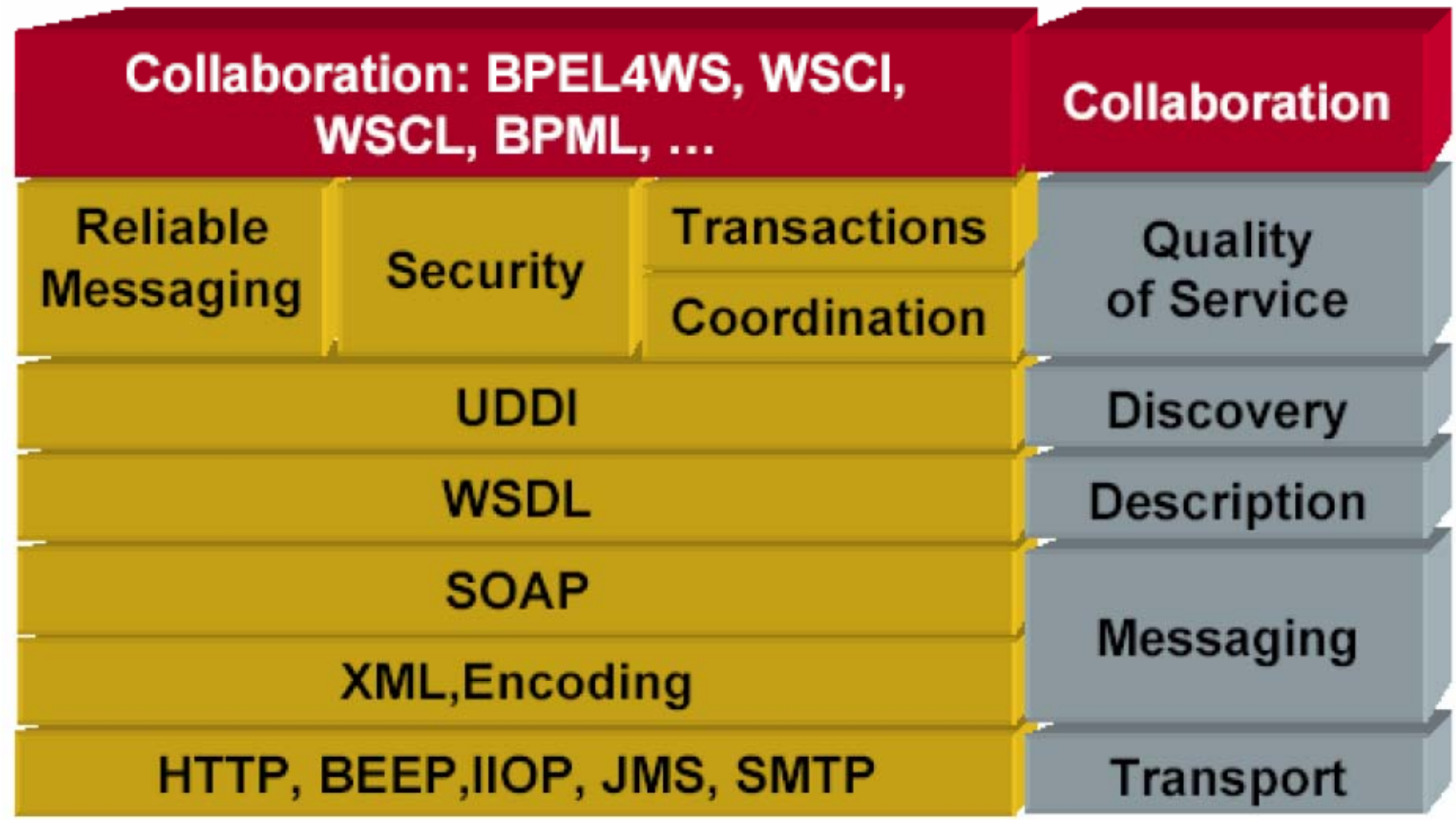
- Web Services
  - According to Wikipedia:
    - According to the W3C a Web service is a software system designed to support interoperable machine-to-machine interaction over a network.
- Web Services Characteristics
  - Language/platform independent
  - Messaging focused on the interface
  - XML over HTTP + web-related standards
- Services
  - Well-defined
  - Self-contained
  - No reliance upon the state of other services

Styles of APIs - SOAP

- Protocol for exchanging XML messages
- Foundation of Web Services
- Relies on WSDL
- SOAP Primer
  - http://www.w3.org/TR/soap12-part0/
- Example SOAP Services
  - Google : http://www.google.com/apis/
  - NOAA National Weather Services
    - http://www.weather.gov/xml/

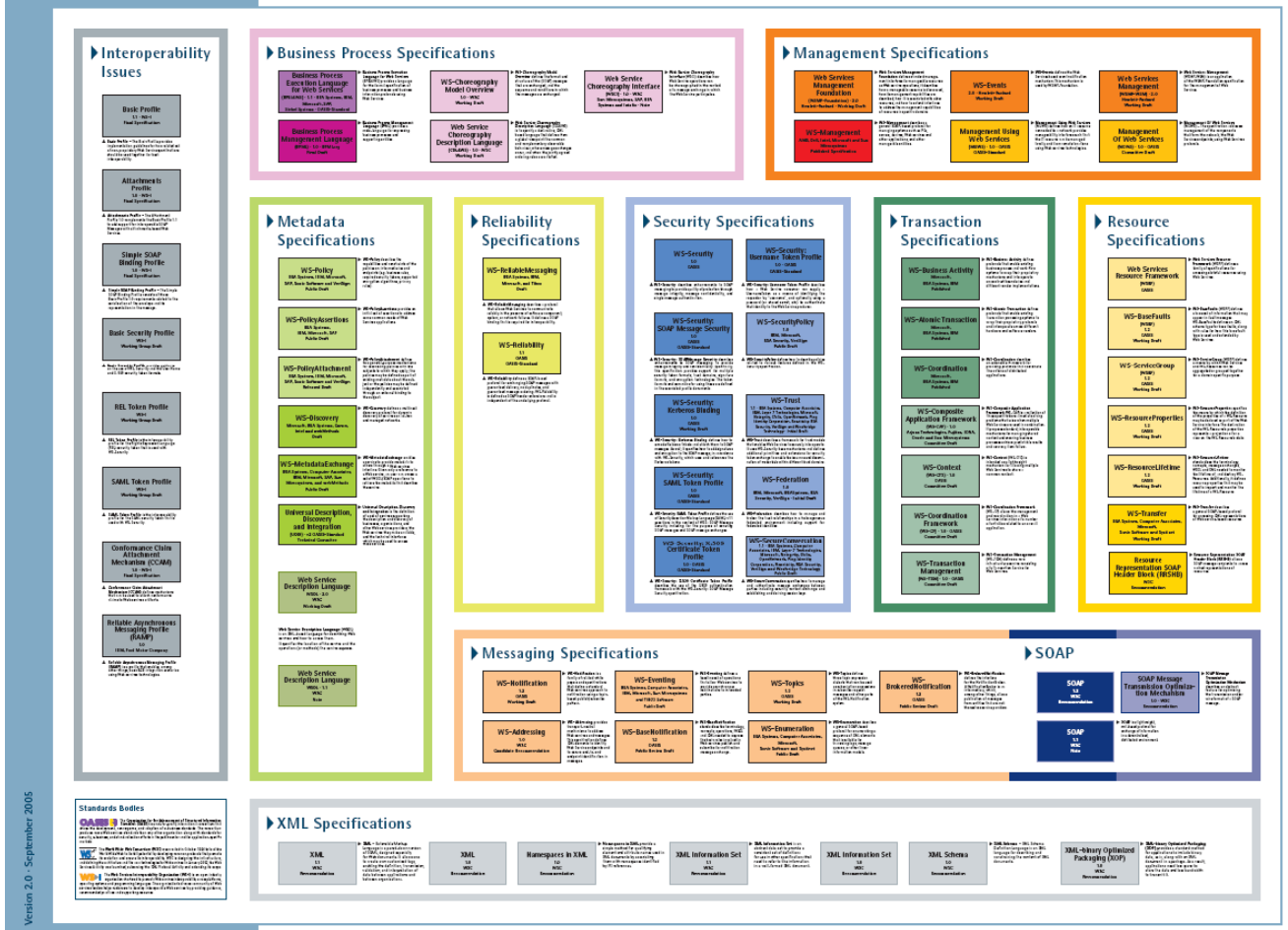
```
<soap:Envelope
    xmlns:soap='http://www.w3.org/2003/05/soap-envelope'>
  <soap:Body>
    <echo xmlns='http://example.com/foo'>
      <bar>Hello World!</bar>
    </echo>
  </soap:Body>
</soap:Envelope>
```

Web Services Technology



기타: http://blog.webservices.or.kr/hollobit/roadmap/ws-specs/

Web Services Technology



Source: http://www.innoq.com/soa/ws-standards/poster/

REST vs SOAP based Web Services

REST based web services	SOAP based Web Services
REST is tied to HTTP only	SOAP is transport neutral (FTP, SMTP, MQ)
REST utilizes URL to identify the desired resources to be accessed	SOAP uses XML message to identify the desired web resource or procedure to be invoked.
Utilizes the web transportation methods to manipulate the resources	Utilizes the Web transportation methods to exchange messages between clients and services
Security in REST can be implemented by Standard and traditional solutions for authorized access to certain web resources	Security in SOAP based web services requires additional infrastructure in web to enable message/Transport level security concerns
In REST every entity in the web is centered around resources	Every entity is centered around interfaces and messages that are channeled to the interface
A better approach for open systems	Is a good approach for closed system
Is an architectural style	Is a RPC/Document oriented architectures
Utilizes the web caching mechanisms to its full potential	Ignores the Web Caching mechanism.
WRDL provides the description to the web resources	WSDL provides the description to Services interfaces which can receive and deliver SOAP Messages
WS-* does not apply to REST	SOAP includes a whole stack of “composable” WS-* specifications

REST vs SOAP [1]

- Example of a REST Web Service
  - **GET /weatherforecast/02110 HTTP/1.1**
    - Get the weather forecast for Daejeon
  - **POST /weatherforecast HTTP/1.1**
    - Upload a new weather forecast for Seoul by sending up an XML document which conforms to the appropriate Schema
    - Response is a “201 Created” and a new URI

      ```
      201 Created
      Content-Location: /weatherforecast/95101
      ```
  - **PUT /weatherforecast/95101 HTTP/1.1**
    - Update an existing resource representation
  - **DELETE /weatherforecast/02110 HTTP/1.1**
    - Delete the resource representation

REST vs SOAP [2]

- Contrast with a SOAP weather service
  - `POST /weatherforecast.asmx HTTP/1.1`
    - Send a SOAP message to get the weather in Daejeon
  - `POST /weatherforecast.asmx HTTP/1.1`
    - Send a different SOAP message to create a forecast for Seoul
    - Response is a custom SOAP response message

```
POST /weatherforecast.asmx HTTP/1.1

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<SOAP-ENV:Envelope xmlns:SOAP-
   ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >

<SOAP-ENV:Body>

<wns:getWeather xmlns:wns="urn:weather" SOAP-
   ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<zipCode xsi:type="xsd:string">02110</zipCode>

</wns:getWeather>

</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

SOAP and REST have in common

- WSDL 2.0 (formerly known as WSDL 1.2) allow services to be defined as both REST and SOAP style services
- SOAP 1.2 supports both REST (HTTP GET) and SOAP style services
- Vendor tools such as Microsoft Visual Studio .NET create Web Services that have both REST and SOAP interfaces
- Public Web Service providers such as Amazon and Google provide both REST and SOAP style Web Services
  - Google Deprecates Their SOAP Search API (AJAX Search API)

Styles of APIs – Web Services

They HATE SOAP!

But LOVE The Web !!!

Styles of APIs - JSON

- What is JSON?
  - Lightweight data-interchange format
    - Compared to XML
  - Simple format
    - Easy for humans to read and write
    - Easy for machines to parse and generate
  - JSON is a text format
    - Programming language independent
    - Uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python
  - Resources
    - Introducing JSON
      - ❖ http://www.json.org/
    - JSON in JavaScript
      - ❖ Http://www.json.org/js.html

Styles of APIs - JSON

- JSON Structures
  - A collection of name/value pairs
    - In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array
  - An ordered list of values
    - In most languages, this is realized as an array, vector, list, or sequence
- Example: JSON Object
  - A JSON object is an unordered set of name/value pairs

```
var myJSONObject =
{       "web":[ {       "name": "html",
                        "years": "5"
                },
                {       "name": "css",
                        "years": "3"
                }]
        "db":[  {       "name": "sql",
                        "years": "7"
                }]
}
```

JSON vs. XML

- JSON is like XML because
  - They are both 'self-describing' meaning that values are named, and thus 'human readable'
  - Both are hierarchical. (i.e. You can have values within values.)
  - Both can be parsed and used by lots of programming languages
  - Both can be passed around using AJAX (i.e. httpWebRequest)
- JSON is UNlike XML because
  - XML uses angle brackets, with a tag name at the start and end of an element: JSON uses squiggly brackets with the name only at the beginning of the element.
  - JSON is less verbose so it's definitely quicker for humans to write, and probably quicker for us to read.
  - JSON can be parsed trivially using the eval() procedure in JavaScript
  - JSON includes arrays {where each element doesn't have a name of its own}
  - In XML you can use any name you want for an element, in JSON you can't use reserved words from javascript

Why JSON over XML?

- Lighter and faster than XML as on-the-wire data format
- JSON objects are typed while XML data is typeless
  - JSON types: string, number, array, boolean,
  - XML data are all string
- Native data form for JavaScript code
  - XML data needed to be parsed and assigned to variables through tedious DOM APIs
    - Data is readily accessible as JSON objects in your JavaScript code
  - Retrieving values is as easy as reading from an object property in your JavaScript code

Key Technical Issues

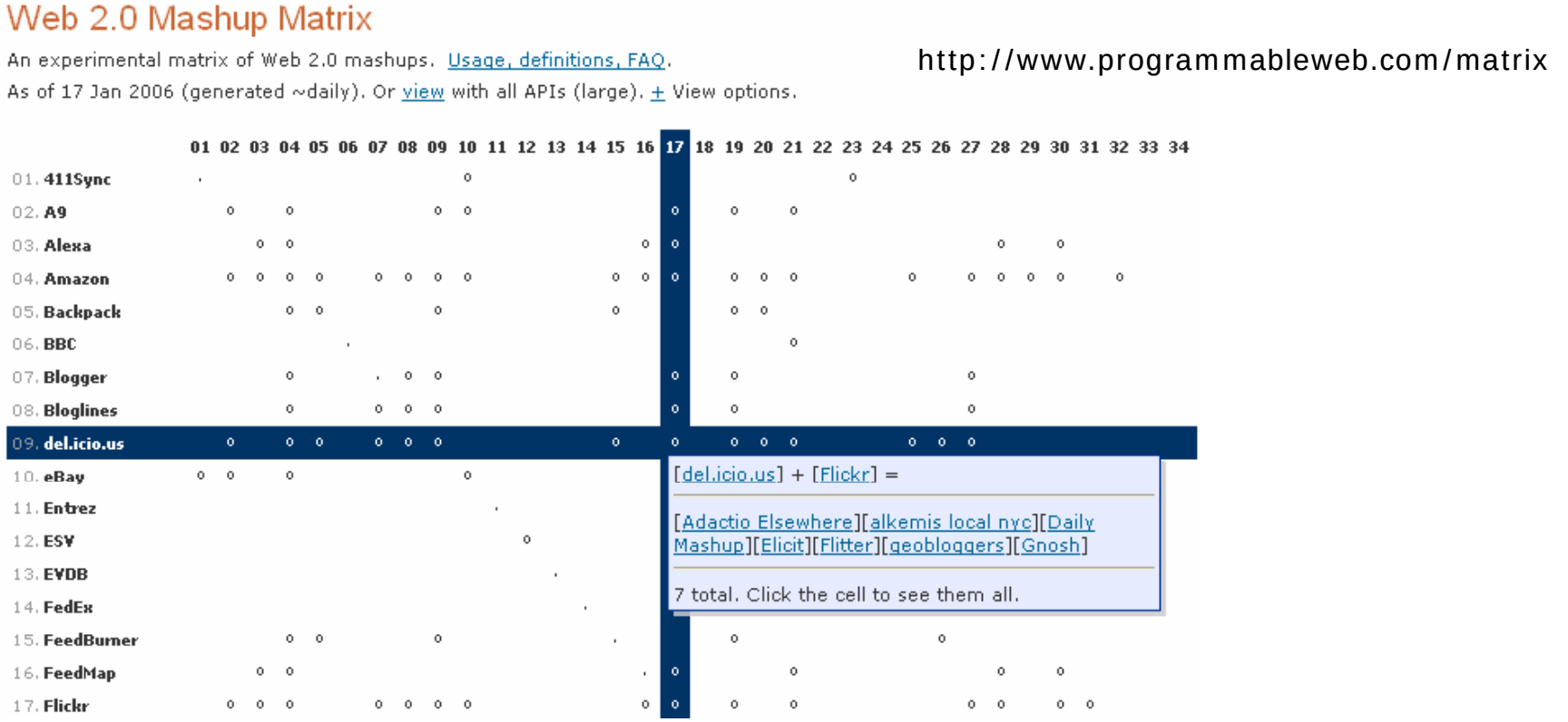
- Data Exchange Format
  - JSON, XML, RSS, Atom
- Transfer Protocol
  - HTTP, SMTP, FTP
- Service Discovery / Service Registry
  - Dynamic Discovery, Service Repository ..
- Service Description
  - WSDL, XML-RPC, RSS
- Messaging
  - SOAP, REST/XML
- Change Mng, Notification, Service Level Agreement, Transaction Mng, Reliable Messaging, Addressing, Resource Mng., Metadata
  - WS-*
- Security / Authentication / Encryption / Federation / Trust
  - WS-*
- Interoperability
  - WS-I Profiles

Mashup & Open API

Mashup & Open API

- A **mashup** is :
  - a website or application that seamlessly combines content from more than one source into an integrated experience.

(**From Wikipedia, the free encyclopedia**)



Mashups are HOT

- Mashups are Growing Fast
  - Ubiquitous web service API
    - Google Maps, Yahoo! Maps, Amazon, Flickr, del.icio.us, etc.
  - People can create new applications by reusing the existing parts
    - The whole is more than the sum of its parts
  - Maps are intuitive UI interface.

3 mashups are added a day

Quick Stats

Total Mashups

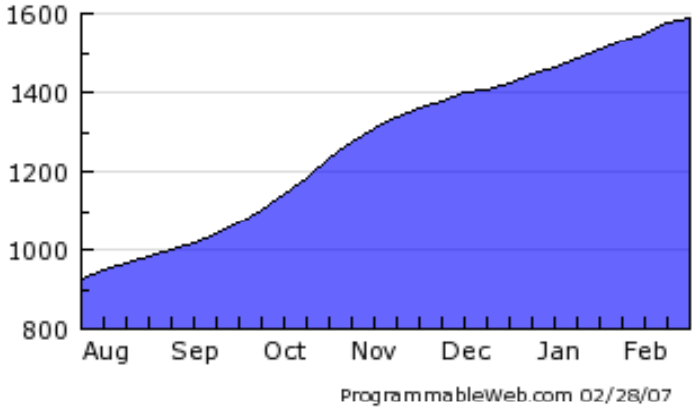
1595

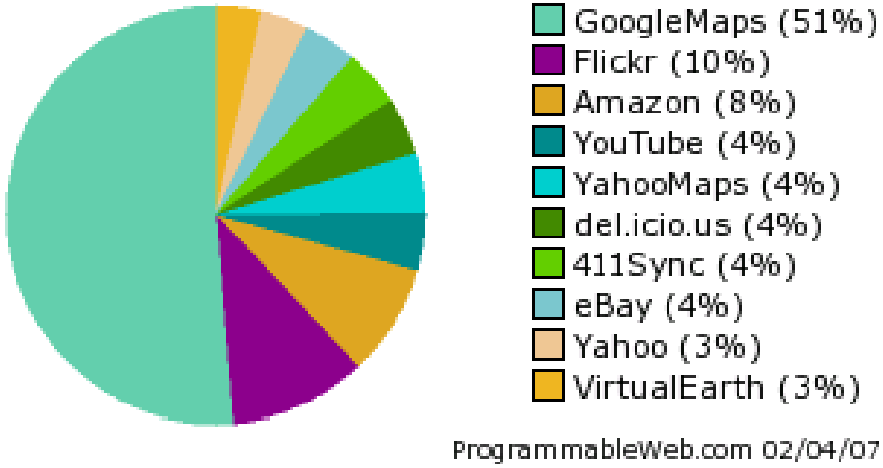
Total APIs

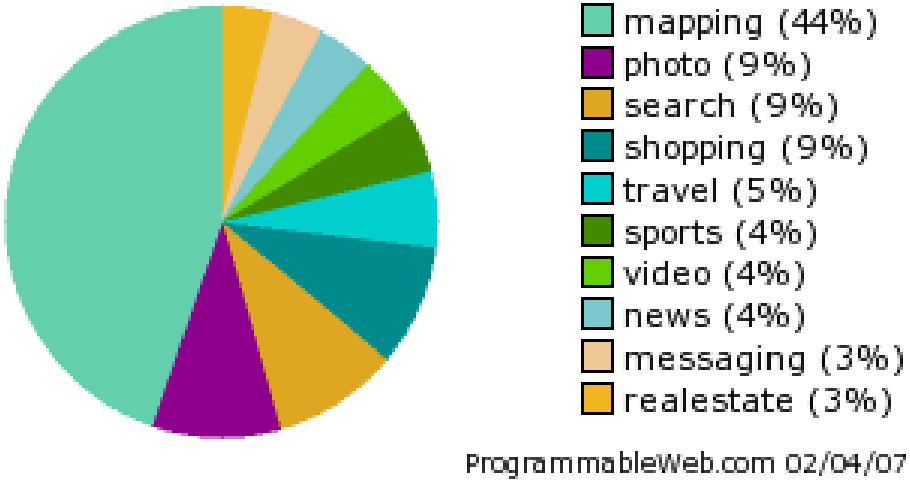
388

Mashups/Day

3.01



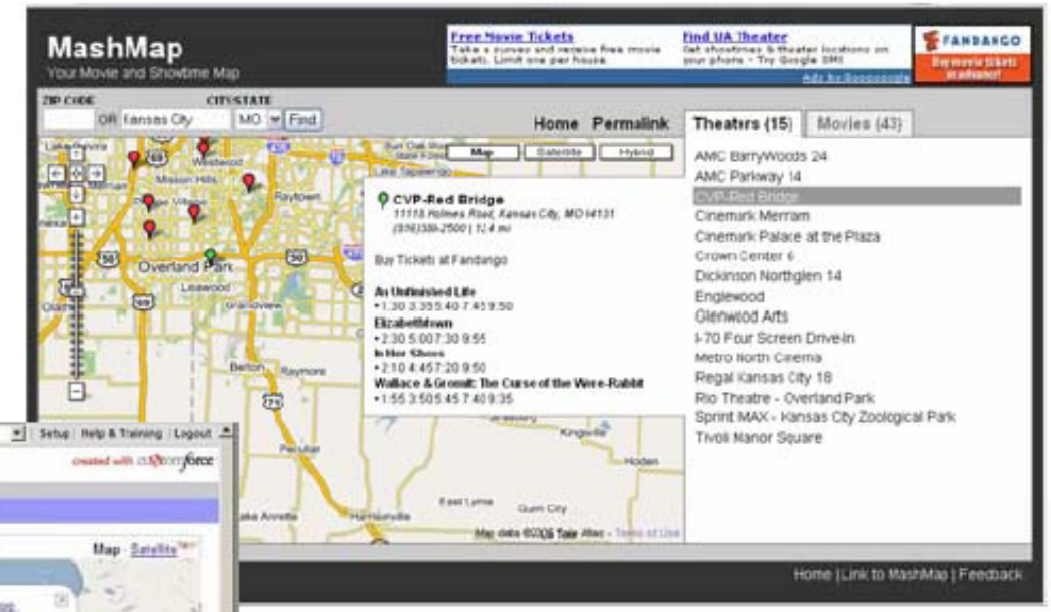




Mashups: Composite Model

Google Maps + Fandango =

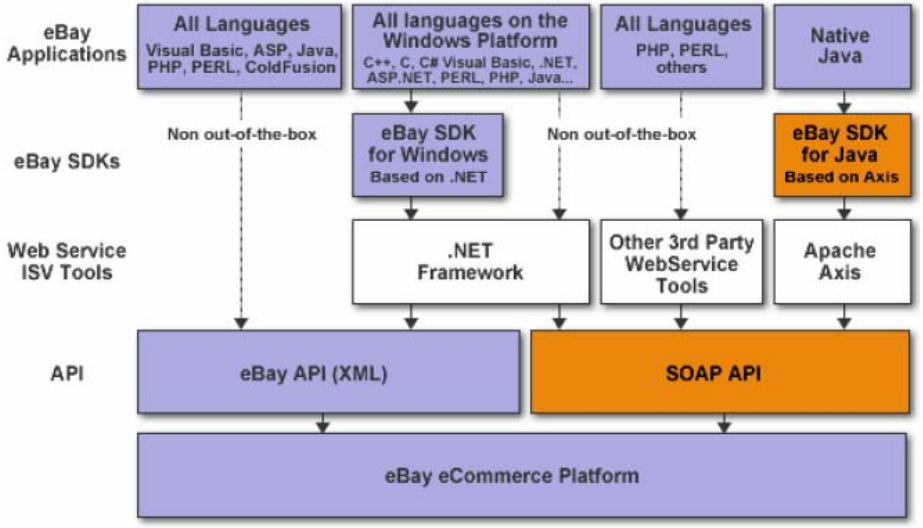
Mashmap.com



Google Maps + Salesforce.com = smashforce

Gartner

Mashup, Hacking & Open API



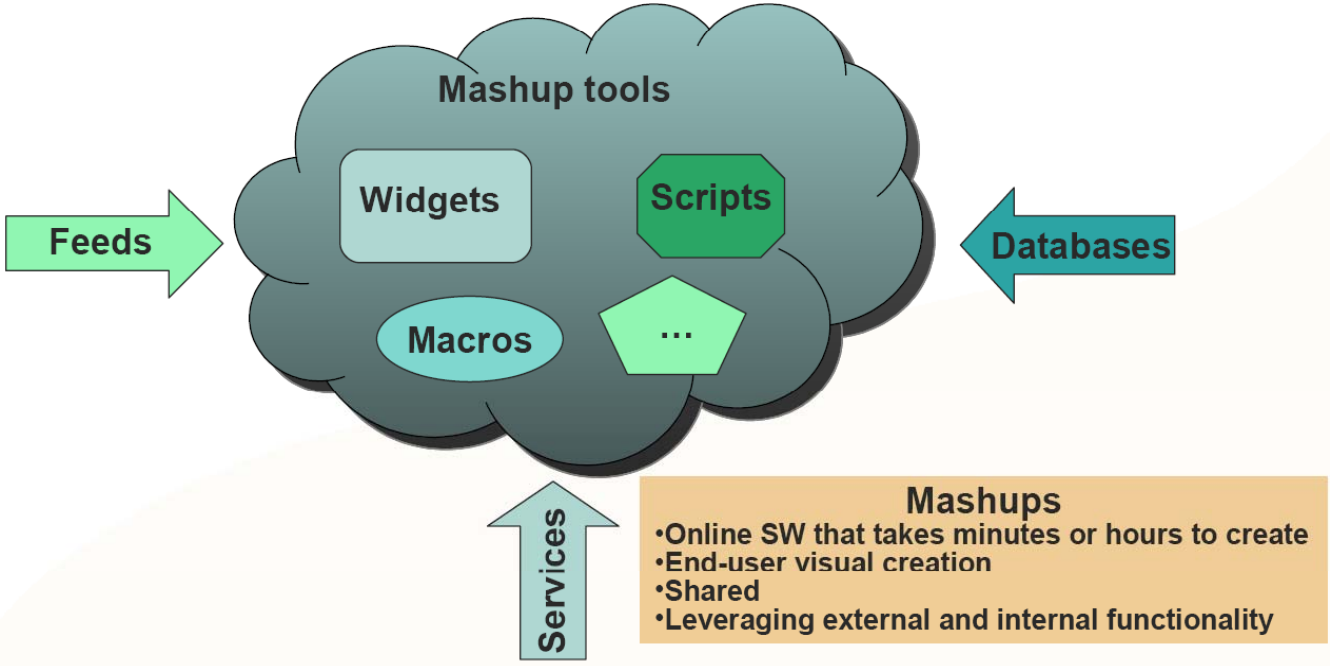
〈Ebay의 전자 상거래 웹 서비스 플랫폼〉

〈플랫폼을 이용한 비즈니스 방법을 다룬 책들〉

Source: KRNet2006, 윤석찬 “Introduction to Web 2.0 Technology” 중

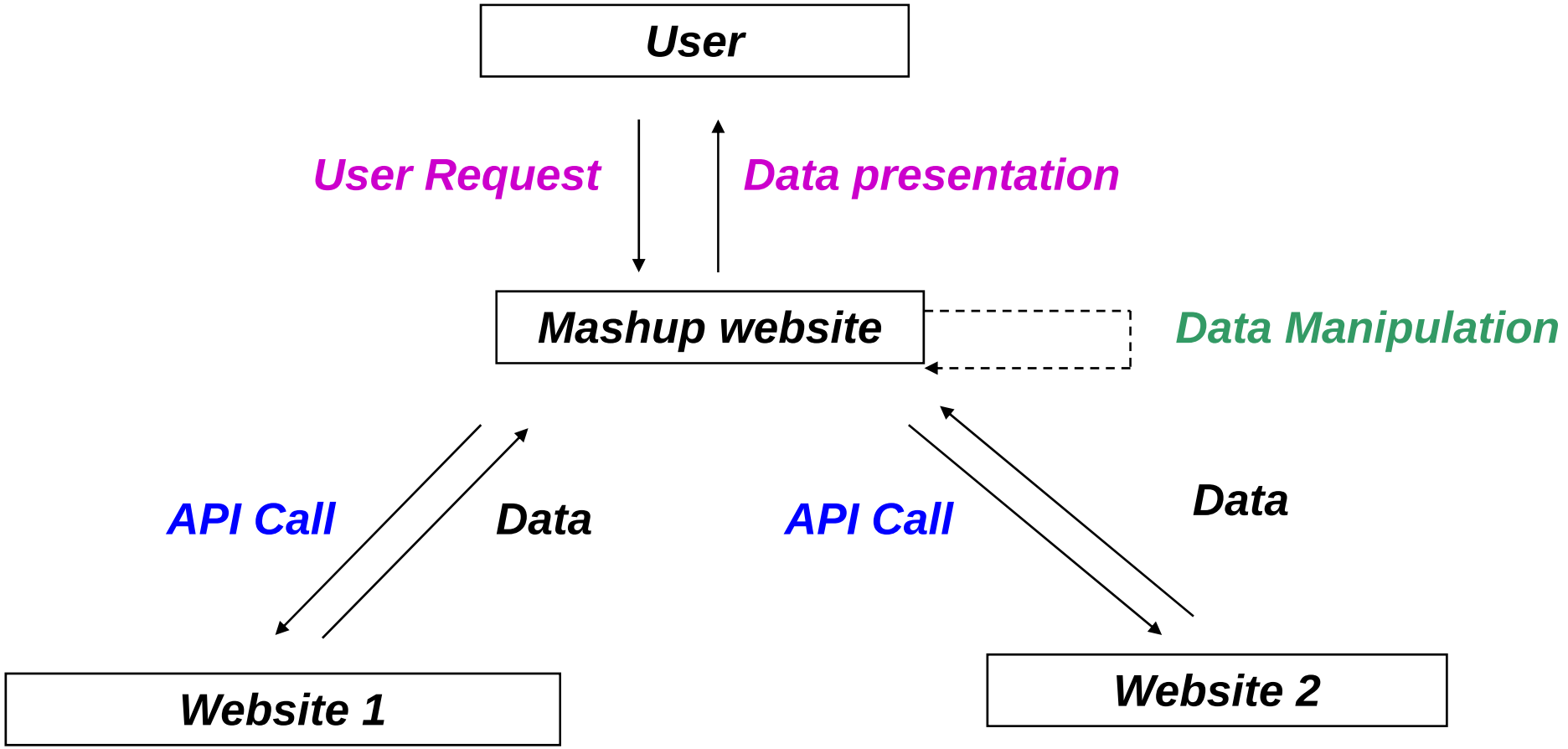
Mashup Ecosystem

- Open Data
  - Usage statements, copyright information
- Open set of services
  - Programmatic access to draw down the access from each
- Small pieces loosely joined

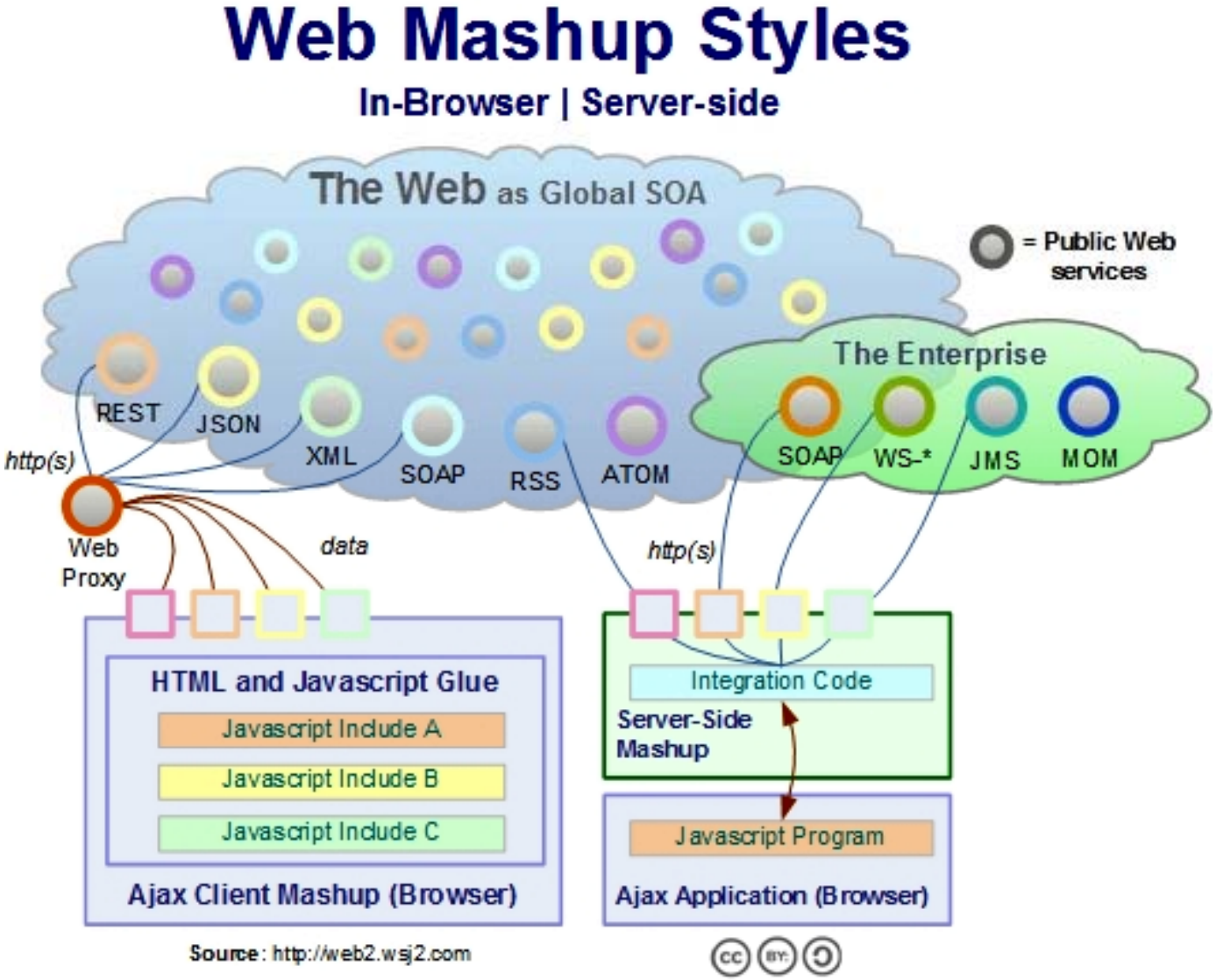


Putting everything together

➢ Your Mashup = API calls + Data Manipulation + UI

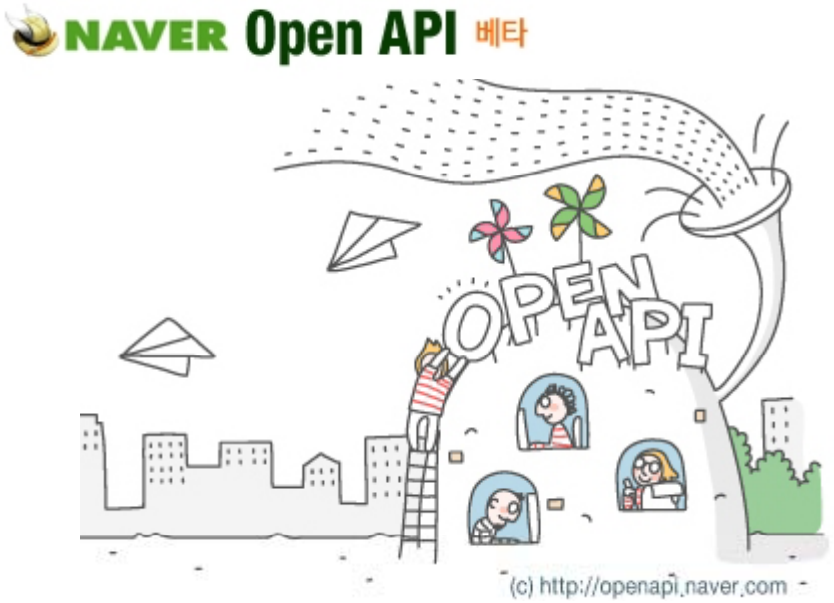


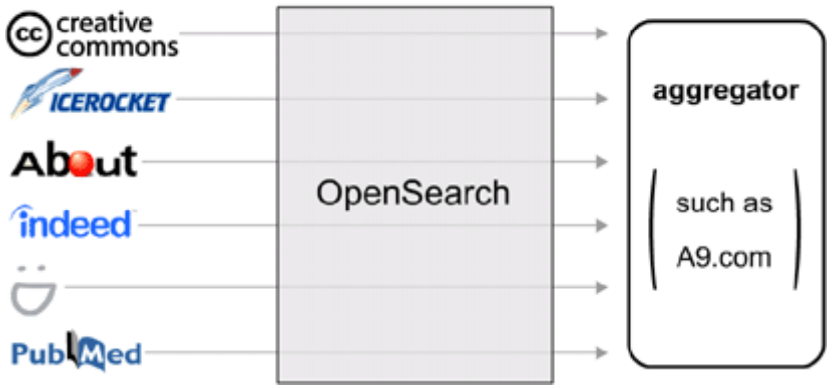
Web Mashup Styles



Web 2.0 Mashup & Open API

- 네이버 http://openapi.naver.com
- Amazon (http://www.amazon.com/gp/aws/landing.html)
- Yahoo (http://developer.yahoo.net)
- eBay (http://developer.ebay.com/rest)
- Flickr (http://www.flickr.com/services/)
- Del.icio.us (http://del.icio.us/doc/api)
- Programmable Web (http://www.programmableweb.com/apis)





국내 **Open API**

- Naver Open API (http://openapi.naver.com/index.nhn)
  - 검색결과 서비스
    - 지식In검색, 블로그 검색, 전문자료 검색, 한국 웹 문서 검색, 책 검색, 쇼핑 검색, 국어사전 검색, 일어사전 검색, 영어사전 검색, 내 PC검색
  - 검색관련 기능
    - 실시간 검색어, 추천 검색, 성인검색어판별, 오타변환, 바로가기
  - 서비스API
    - 지식스폰서 API, 지도 API
- Daum API (http://dna.daum.net/apis)
  - 검색 API
    - 신지식 검색, 게시판 검색, 카페 검색, 블로그 검색, 뉴스 검색, 도서 검색, 일본어 사전
  - UCC API - 블로그
  - 디앤샵 API
    - 상품 정보, 상품 검색
  - 여행 API
    - 할인 항공권 검색, 해외호텔 검색, 국내호텔 및 숙박 검색, 여행상품 검색
  - 인증 API – 어플리케이션 인증

50 Things to do with Google Maps Mashups

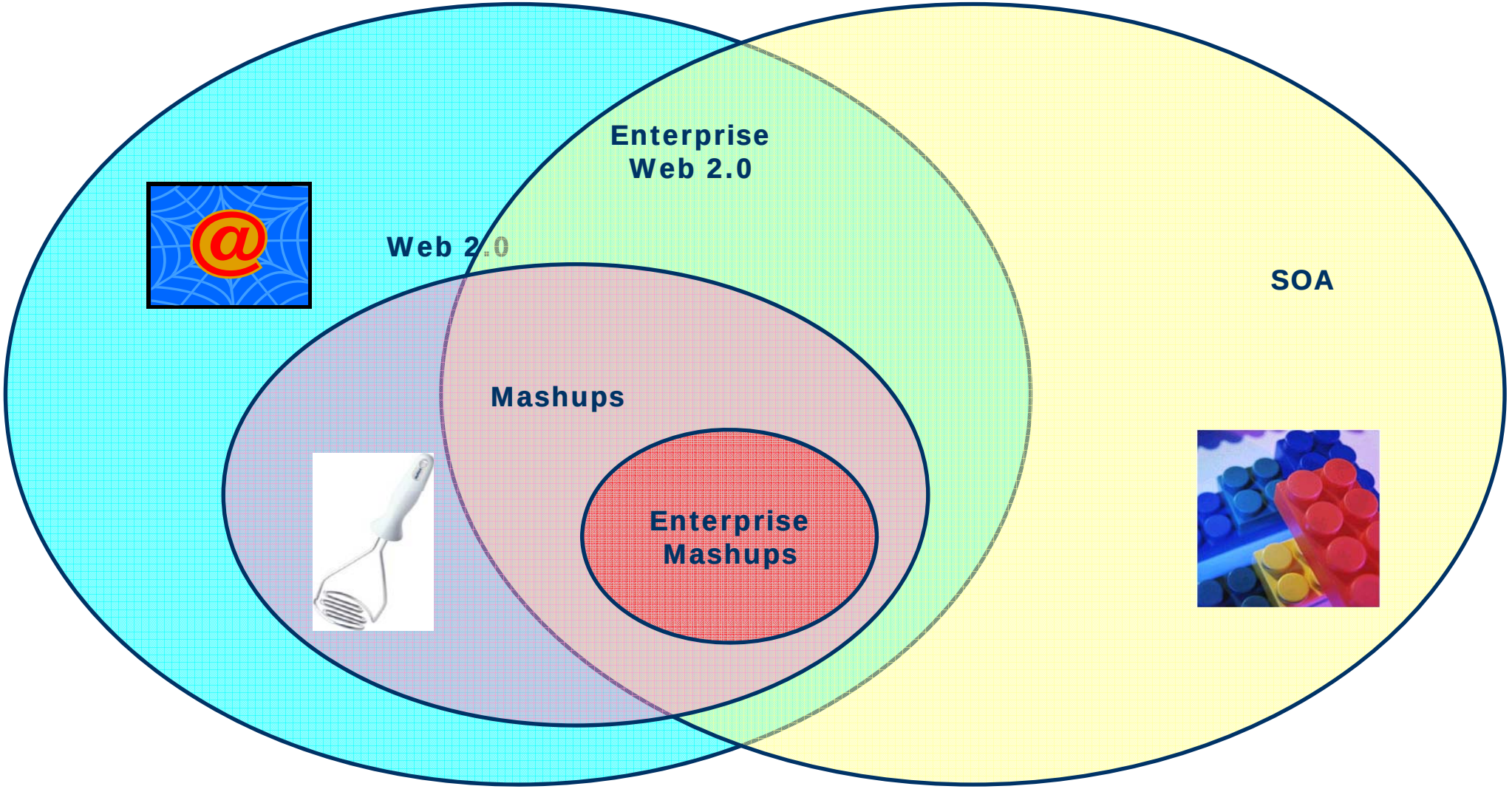
1. run route의 측정
2. 세계 속의 시간 체크
3. 미국내 우편번호 검색
4. 영역 재기
5. 자신만의 지도 만들기
6. 사진맵 만들기
7. 세계 곳곳에 무엇이 있는지 확인하기
8. 미국내에 사람찾기
9. 날씨 체크
10. 가까운 스타벅스 찾기
11. eBay 부동산 거래 찾기
12. 미국/영국내 공항 주차장 찾기
13. 달리기 코스 만들기
14. 여행 비디오 보기
15. 미국내 패스트푸드점 찾기
16. 온타리오의 맥주 사기
17. 우편번호로 뉴스 확인하기
18. 발병 경로 피하기
19. 트래픽 피하기
20. 나만의 구글맵 매쉬업 만들기
21. 감시 카메라 피하기
22. 값싼 주유소 찾기
23. 일출/일몰 시간 체크
24. 통화 교환
25. 미국내 살 곳 찾기
26. 지금 대낮인 곳 찾기
27. 세계 스키리조트 찾기
28. 구글맵 상에서 항공편 추적하기
29. 두 공항 사이의 거리 계산하기
30. 해발 고도 찾기
31. 세계 7대 경이 지도
32. 세계 골프 코스 찾기
33. 어딜 클릭하든 여행 정보 얻기
34. 홍수 피해 예상 시뮬레이션
35. 비행기 조종
36. 전체화면으로 구글맵 검색
37. 세계 웹캠 찾기
38. 미국/캐나다 도서관 찾기
39. 세계 항구 찾기
40. 탱크 속에 생선 지도
41. 현대 해적 위험 지역 찾기
42. 세계 잠수 사이트 찾기
43. 위치별 위키피디아 기사 지도
44. UFO 관측 지점 보기
45. 메카 방향으로 기도하기
46. 미국 우편번호로 이베이 물품 검색
47. 비행기 소란과 공해 지도
48. 런던 지하철 여행 계획
49. 뉴욕내 택시 요금 계산
50. 미국과 캐나다의 영화 촬영지 지도

http://hollobit.tistory.com/49

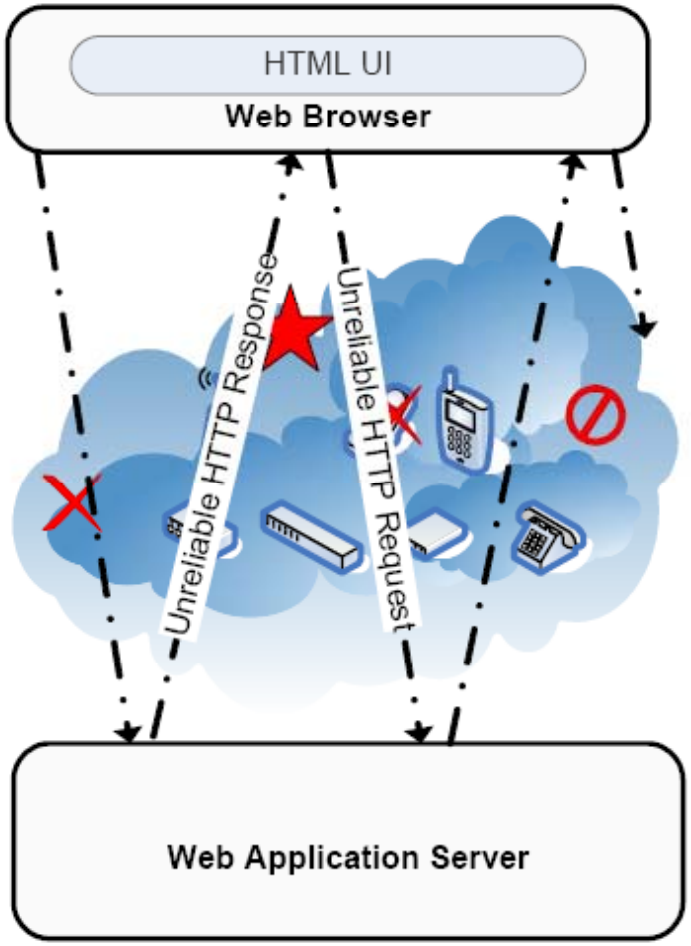
Type of Mashups

- (Internet or Web) Mashup
- Geographical Mashup
- Mobile Mashup
- Enterprise Mashup
- Ubiquitous Mashup
- Offline Mashup

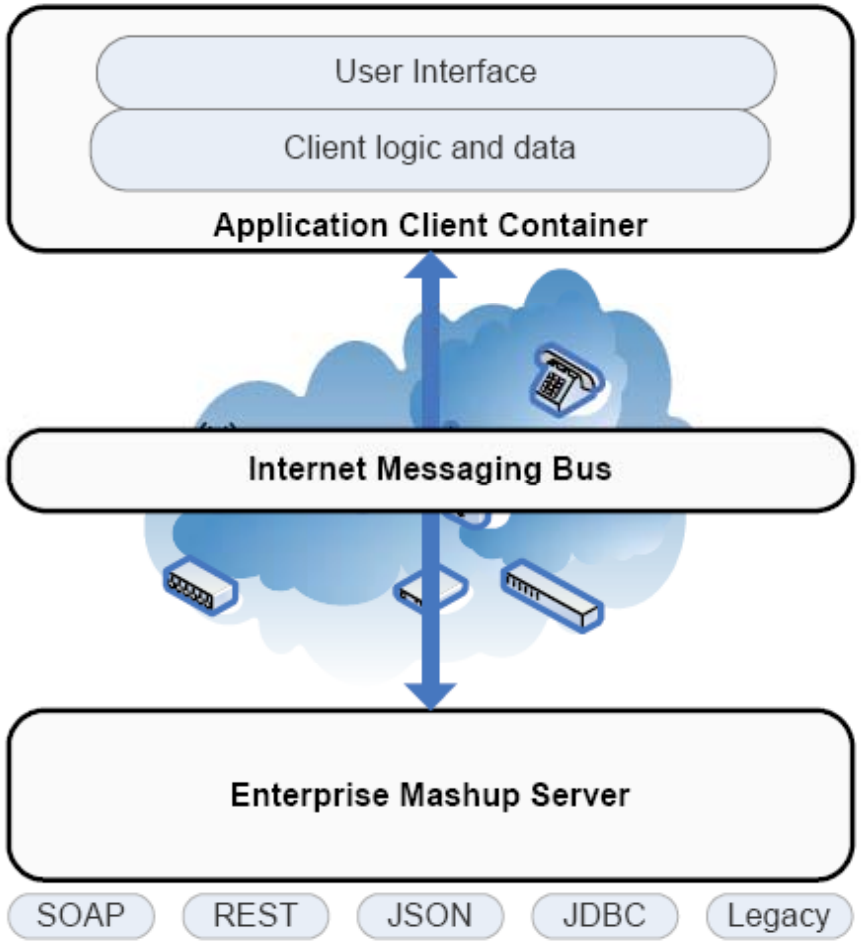
Web 2.0 vs. SOA



From Web 1.0 to Web 2.0: The Evolution of Technology Stack

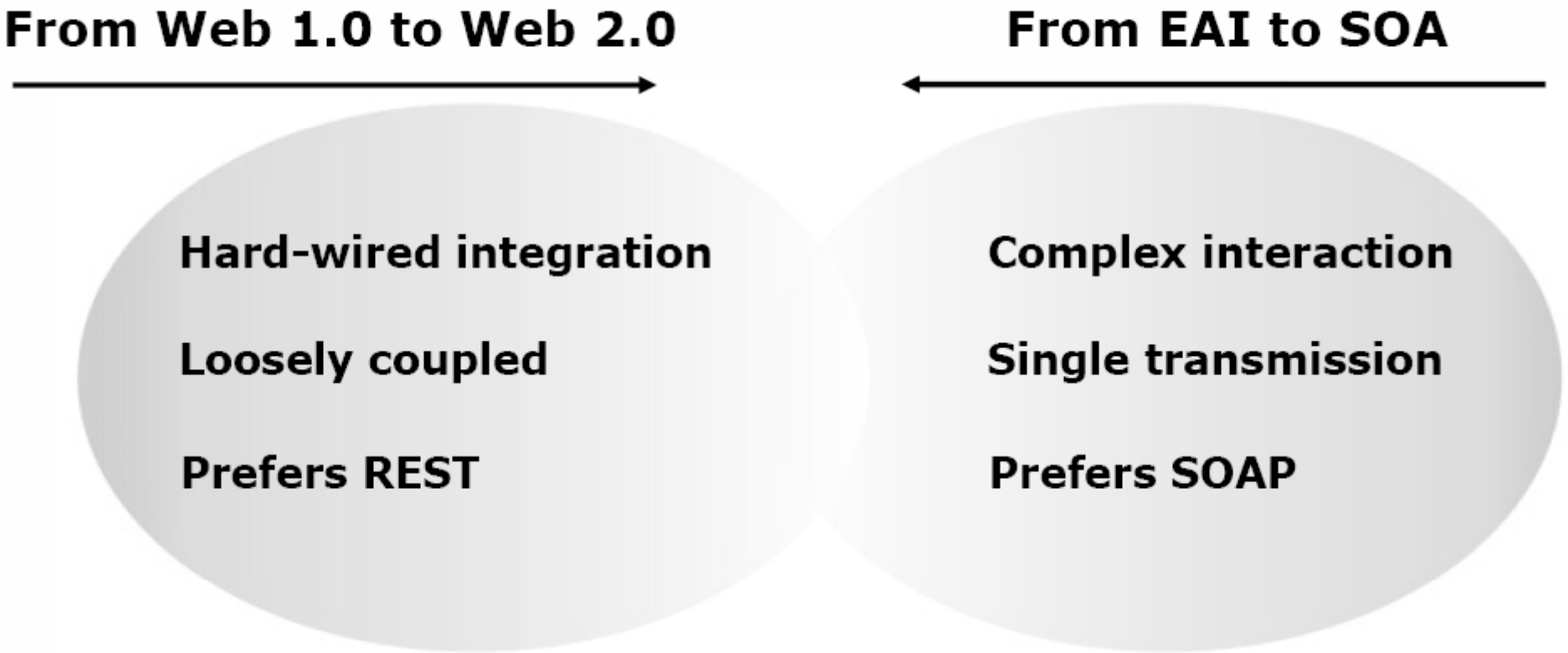


Web 1.0 Technology Stack



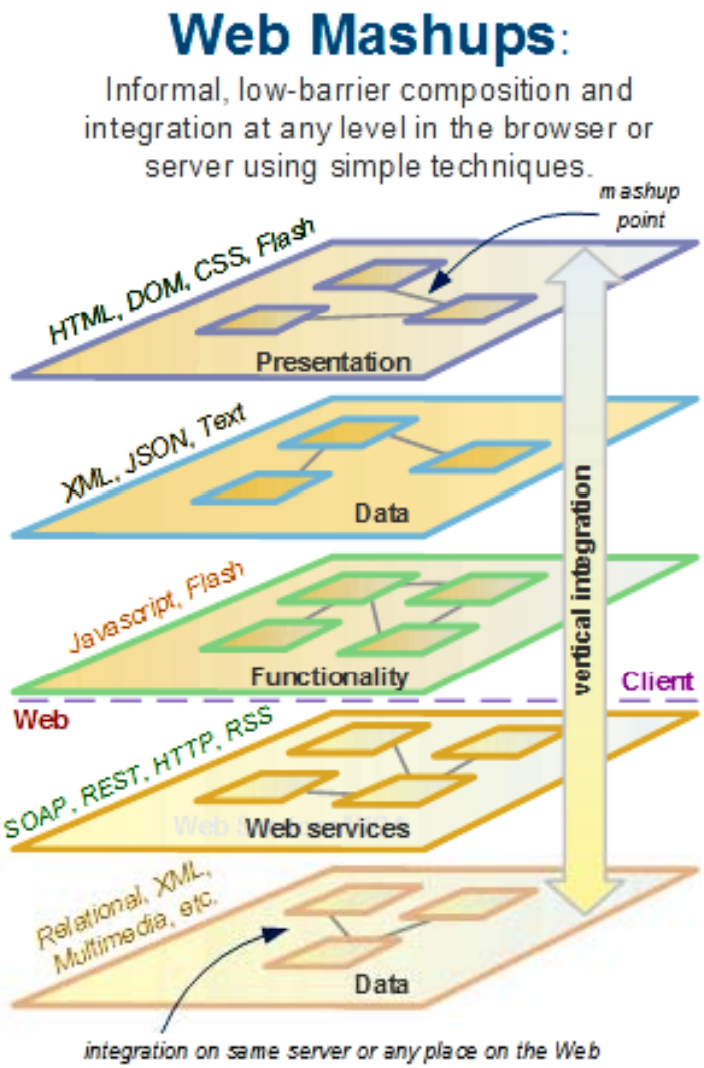
Web 2.0 Technology Stack

Effects of a Merge



The styles of enterprise mashups

- **Presentation Mashup**
  - where information and layout is retrieved and either remixed or just placed next to each other.
- **Client-Side Data Mashup**
  - takes information from remote web services, feeds, or even just plain HTML and combines it with data from another source.
- **Client-Side Software Mashup**
  - where code is integrated in the browser to result in a distinct new capability.
- **Server-Side Software Mashup**
  - where software is recombined on the server since web services can more easily use other web services and there are less security restrictions and cross domain issues.
- **Server-Side Data Mashup**
  - uses relatively powerful mechanisms to join or mashup data from databases on the server-side.



http://blogs.zdnet.com/Hinchcliffe/?p=49&tag=nl.e622

Enterprise Mashups

- You have to distinguish between Internet Mashups and Enterprise Mashups

  - Internet Mashups
    - Enabling web communities to create new business solutions reusing and combining functionality (services) into new challenging products.
  - Enterprise mashups
    - Enabling small communities within a company to create adapted tools for their situation

SaaS

Enterprise Web 2.0

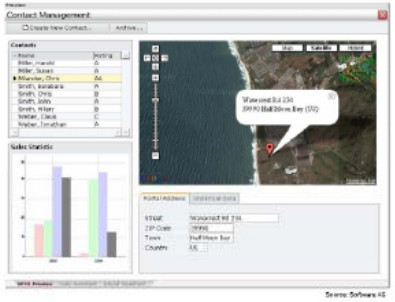
Enterprise 2.0

Enterprise Mashups can be seen as “Situational Solutions”

Conceptual architecture for Enterprise Mashups

Mashup Toolbox

Service Directory



Mashup

Services

Web (External IT)

Internal IT

Where to Start

- Point & click, cut, paste & publish
- “Cloning” - simple source code edits
- Just program it
  - Developer’s toolkits and API documentation
- Using APIs/Web Services
  - Step 1 – Get an idea
  - Step 2 – Sign up for a developer token
    - http://aws.amazon.com/
    - http://www.google.com/apis/maps/
    - http://api.search.yahoo.com/webservices/register_application
  - Step 3 – Read the fine print
  - Step 4 – Create your first mashup

Issues

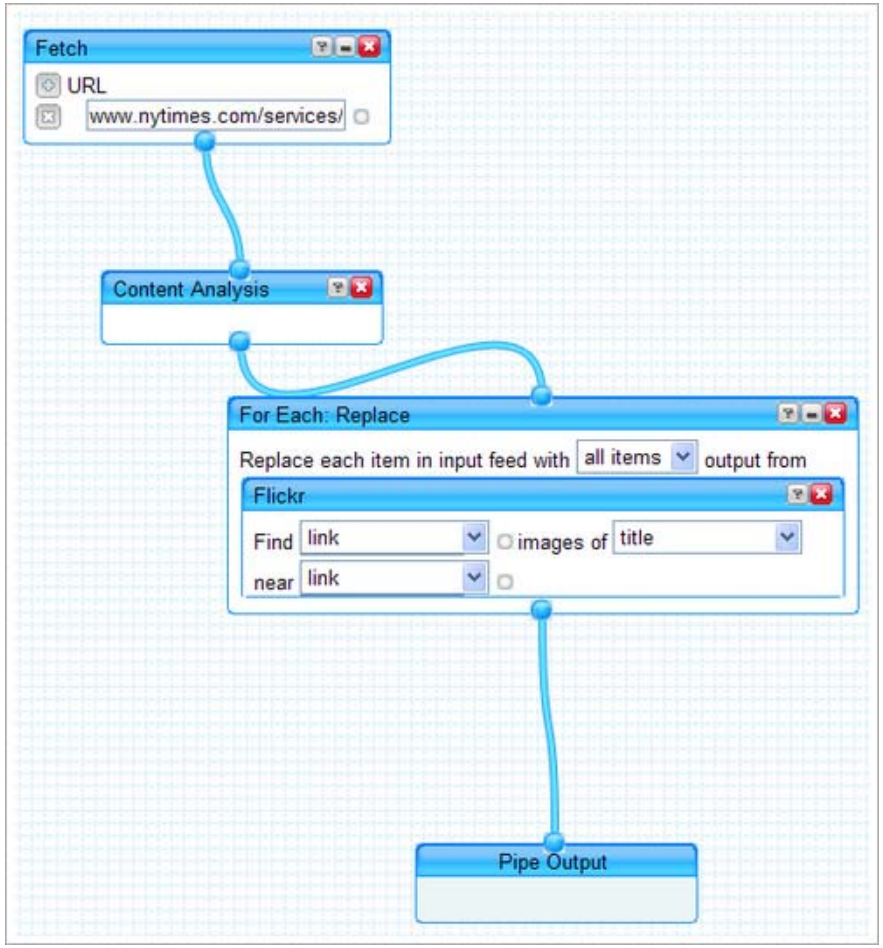
- Technical Issues
  - In its infancy
  - Mashup tools are fall short of the ideal
  - Better universal registry for mashup APIs
  - Scale and dependencies issues
  - How much to invest?
    - Not licensing but using “free” data
  - Development is quicker with languages like Ruby
- Social Issues
  - Intellectual property issues
    - “right to remix”
  - Provenance
    - Origin or source
    - Authority

Web Platform APIs — A Partial List

- 411sync - SMS messaging
- **Amazon - Online retailer, search, queuing service**
- AmphetaRate - News aggregator
- Backpack - Online information manager
- BBC - Multimedia archive database
- Blogger - Blogging services
- Bloglines - Online feed aggregator
- Buzznet - Photo sharing
- CDYNE - Data delivery services
- **cPath - Medical database lookup**
- Creative Commons - Licensing engine integration
- **Data On Call - Fax services**
- del.icio.us - Social bookmarking
- Digital Podcast - Podcast search
- **eBay - Online marketplace**
- EVDB - Events database
- **FedEx - Package shipping**
- FeedBurner - Blog promotion tracking service
- FeedMap - Blog geo-coding
- Findory - Personalized news aggregation
- Flickr - Photo-sharing service
- Freedb/CDDB - Online CD catalog service
- geocoder - Geographic lookup services
- Gigablast - Search service
- **Google - Adwords, advertising, search, maps**
- GraphMagic - Graph and chart services
- Internet Archive - Non-profit Internet library
- JotSpot - Wiki-style collaboration tools
- **Library of Congress SRW - Information search**
- **Microsoft - Mapping (MapPoint, Virtual Earth)**
- **NASA - Satellite mapping images**
- NCBI Entrez - Life sciences search services
- NewsGator - Feed aggregation
- **NOAA Weather Service - Weather forecast database**
- **PayPal - Online payments**
- Plazes - Location discovery service
- **Skype - VoIP software**
- **StrikeIron - Web services marketplace**
- Tagalag - Email tagging
- Tagyu Tag - Recommendation service
- Technorati - Blog search
- **Telcontar - Location-based services**
- Trekmail - Messaging services
- TypeKey - Authentication framework
- Upcoming.org - Collaborative event calendar
- **UPS - Package shipping**
- **Yahoo! - Ads, ad mgt, maps, search, shopping**
- **ZipCodes - Zip code lookup service**
- Zvents - Events ecosystem

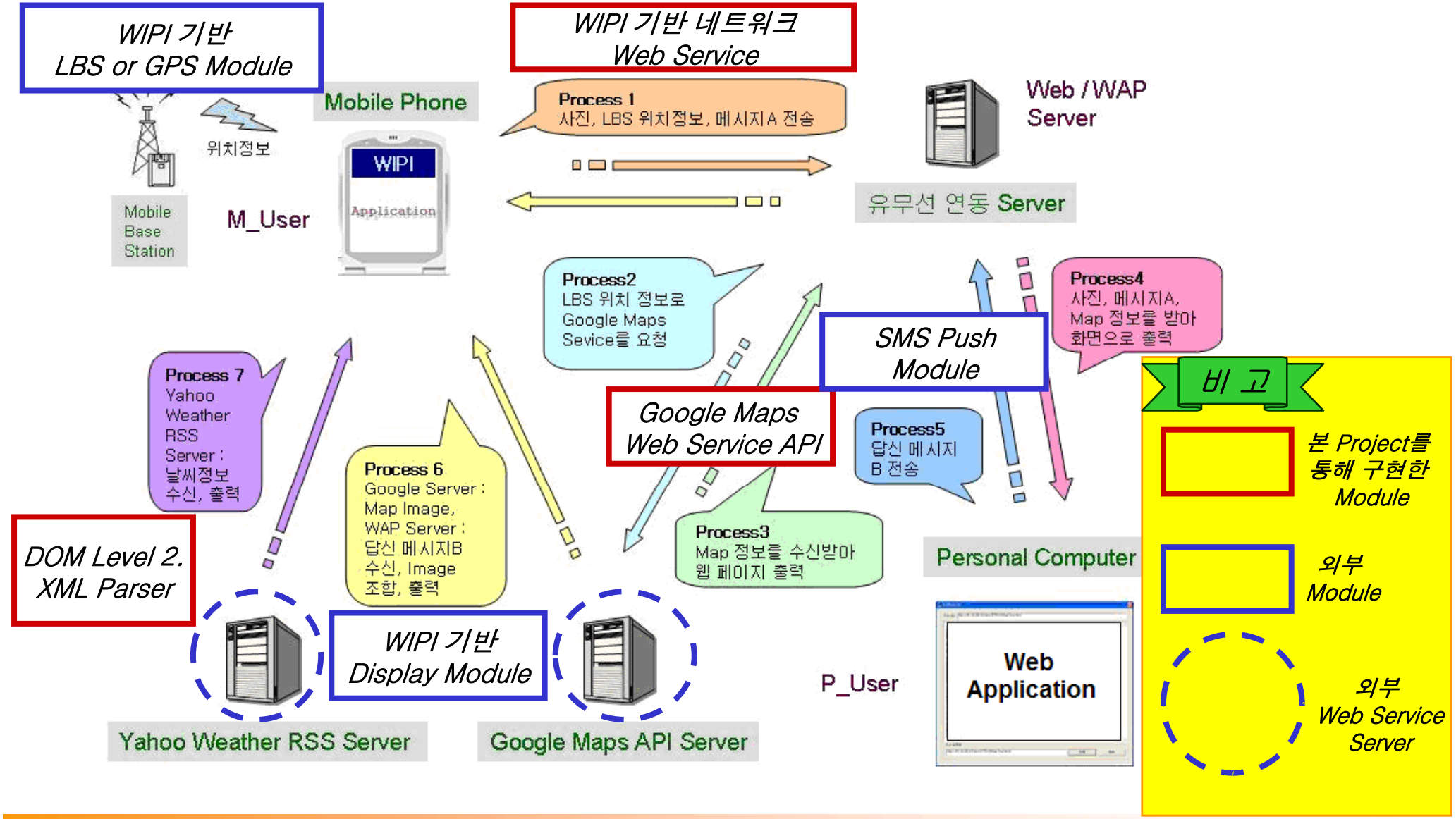
Mashup Builder

- Yahoo Pipe
  - http://pipes.yahoo.com/
  - Rewire the Web
  - RSS와 Atom밖에 대응하고 있지 않지만 앞으로 대응 데이터 소스의 수가 늘어날 예정
  - 예: 뉴욕타임즈 홈페이지의 글을 분석해서 키워드를 추출한 뒤, flickr에서 해당 키워드로 검색되는 이미지들을 보여주는 파이프

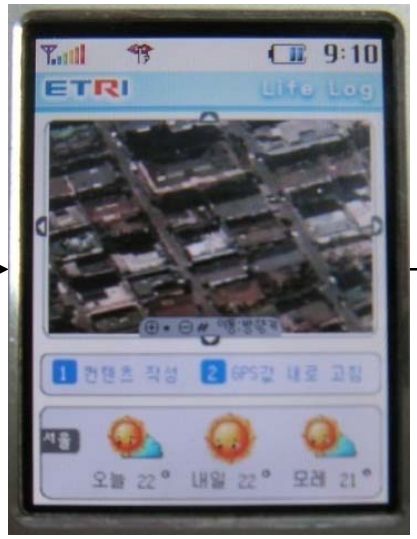


Ubiquitous Web Mashup

WIPI & Mobile Mashup



WIPI & Mobile Mashup



- 어플리케이션 동작

- 현재 위치 값 얻어오기 시도
- 현재 위치 값과 매칭되는 Google earth 표시
- Yahoo weather의 데이터 파싱하여 표시

- 사진 촬영 및 편집
- 메시지 작성
- 사진 및 메시지 web으로 업로드

WIPI 기반
LBS or GPS Module

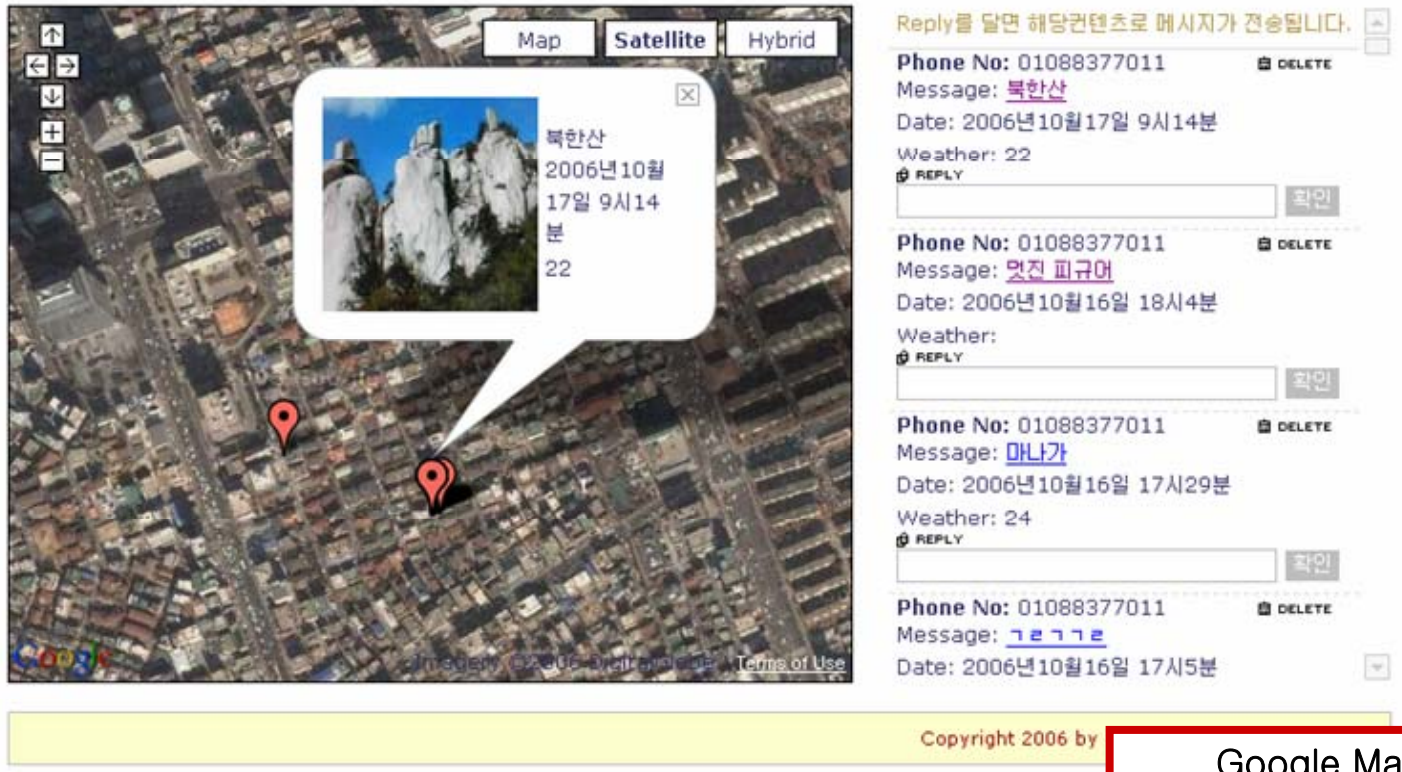
Google Maps
Web Service API

DOM Level 2.
XML Parser

Mobile Phone
Resource 구동 모듈

WIPI 기반 네트워크
Web Service

WIPI & Mobile Mashup

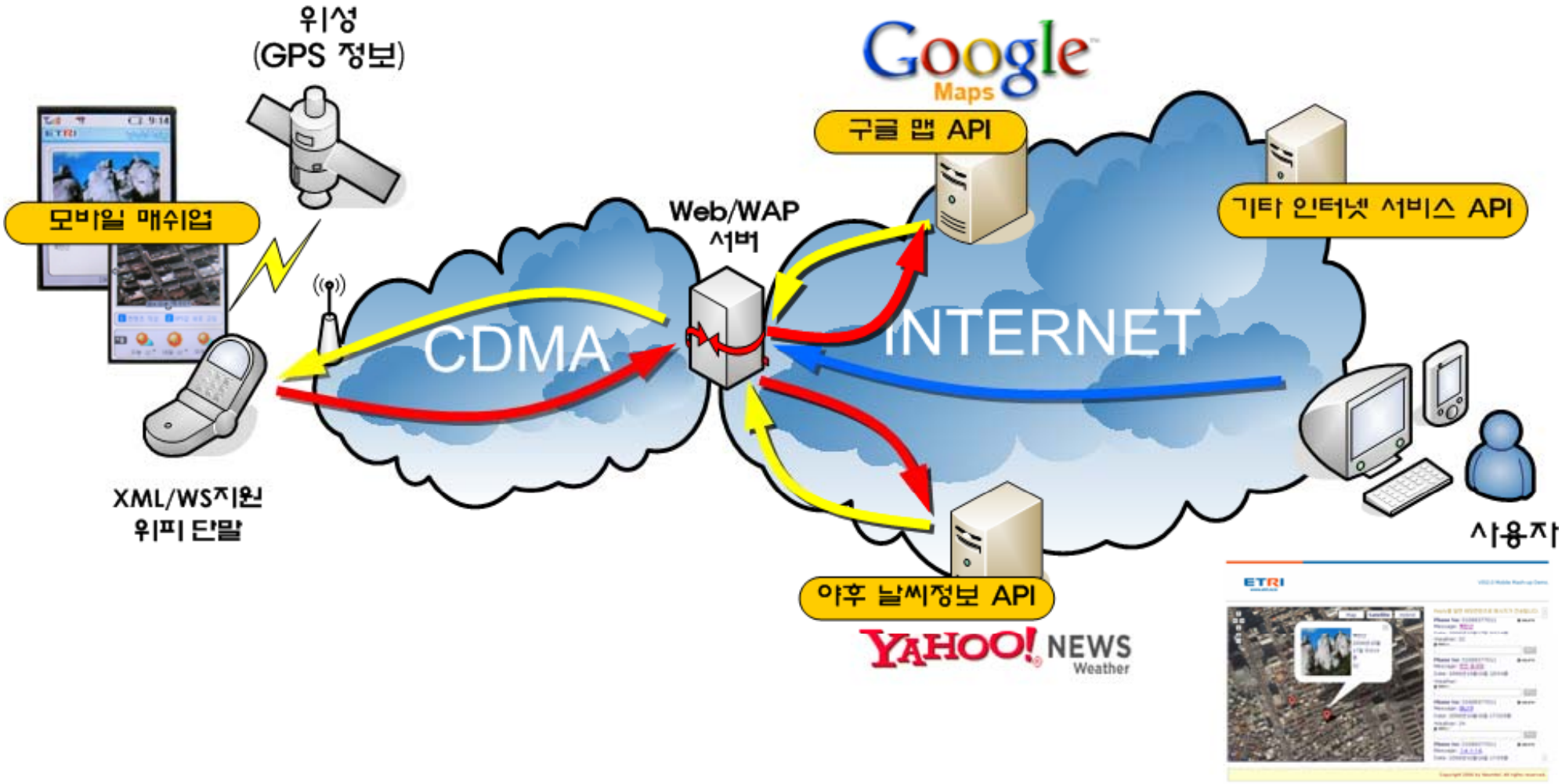


- Google Earth 및 컨텐츠 목록으로 구성된 웹페이지
- 컨텐츠가 전송된 위치가 지도 위에 표시되고 컨텐츠 목록에 등록
- 지도 위의 표시 클릭하면 사진 및 메시지, 부가정보 표시
- Reply에 메시지 입력 후 전송하면 해당 컨텐츠를 전송한 단말로 메시지 전송

Google Maps
Web Service API

SMS Push
Module

WIPI & Mobile Mashup



Mobile Barcode & Mashup

Mobile RFID Mashup

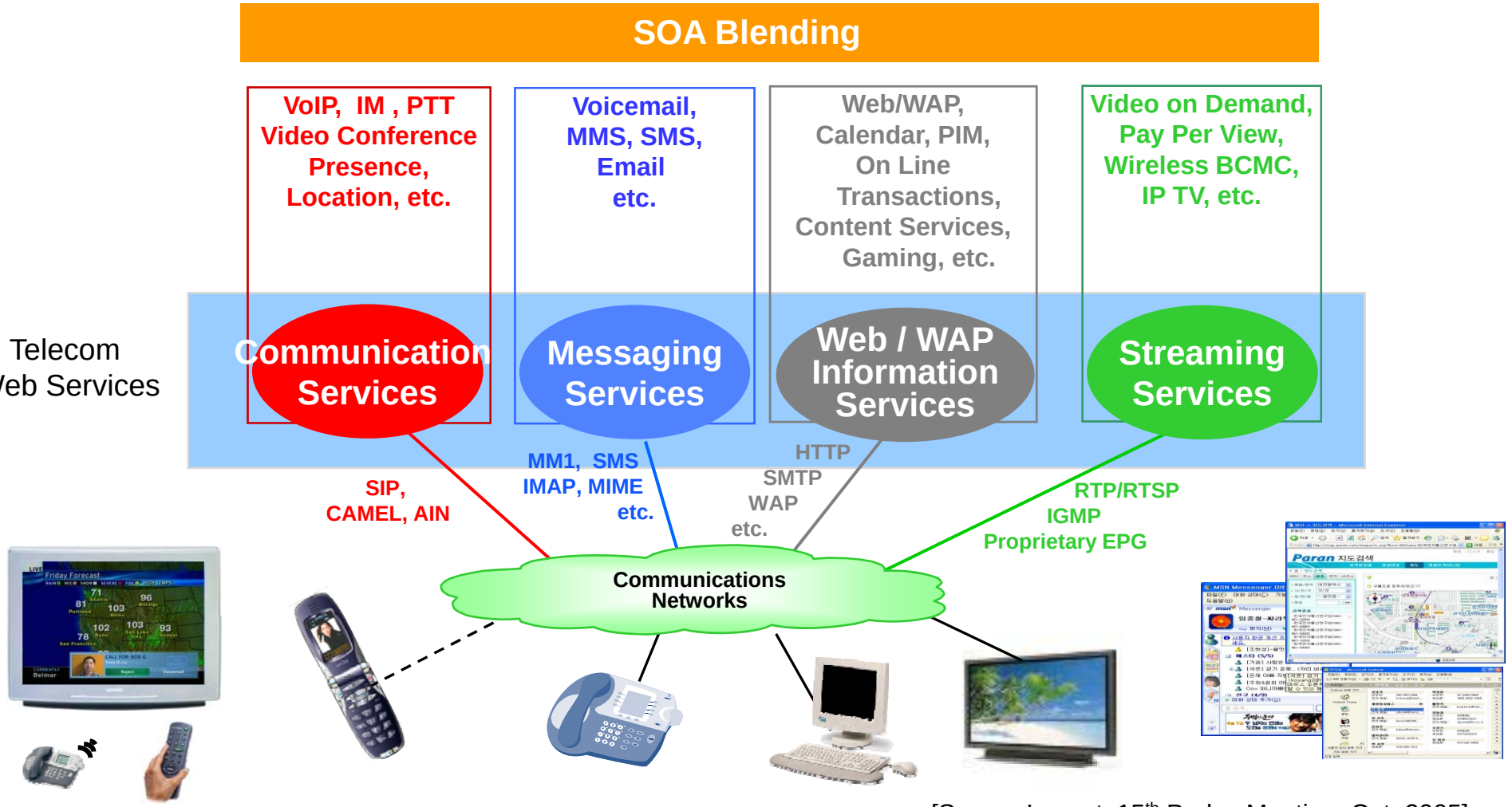
- RFID는 기존 웹2.0의 매쉬업 기능과 결합하여 개별 제품의 상세 정보를 제공할 수 있는 대단한 장점을 가질 수 있음

- Google Maps / Yahoo Maps + RFID = Location, Tracking
- VirtualEarth + RFID = Location, Tracking
- eBay + RFID = Auction Library, sell/buy
- Flickr + RFID = Image Library
- Amazon + RFID = Personal Library, Auto Shopping
- Yahoo Geocode + RFID = Item Location
- Del.icio.us + RFID = Tag-marking, Tag-ID share
- Technorati + RFID = Tagging, semantic clustering
- A9 + RFID = Search, Comparison, Relate Information
- Google + RFID = Search, Comparison, Ads
- YouTube + RFID = Search, Movie
- Shopping.com + RFID = Price Comparison, Shopping

- EX: GoogleMaps + Flickr + eBay + Amazon + RFID + Google

Converged Mashup Services

➢ Opportunities to blended services across the bundle



[Source:Lucent, 15th Parlay Meeting, Oct. 2005]

Developer's Paradise

Developer Ecosystem

Yahoo Developer Candy

- Javascript User Interface Library
  - A collection of JavaScript libraries you can use in your Web development
  - http://developer.yahoo.com/yui/
- Design Patterns
  - A pattern describes an optimal solution to a common problem within a specific context.
- Application Gallery
  - http://gallery.yahoo.com
  - Unifies the many separate collections of applications
    - Flickr, Search, Widgets, Maps, etc., etc.
    - For Widgets, Web sites, plug-ins, etc.

Developer Candy

- http://dna.daum.net/

- http://openapi.naver.com/

Google Code

- http://code.google.com/
  - Google Account Authenticatio
  - AdSense API
  - AdWords API
  - Google AJAX Search API
  - Google Base Data API
  - Blogger Data API
  - Google Calendar Data API
  - Google Code Search Data API
  - Google Data APIs
  - Google Desktop SDK
  - Google Earth KML
  - Google Gadgets API
  - Gmail Atom Feeds
  - Google Apps APIs
  - Google Checkout API
  - Google Web Toolkit
  - Google Groups Feeds
  - Google Maps API
  - Google News Feeds
  - Google Notebook Data API
  - Google Related Links
  - Google Search Appliance APIs
  - Google Search History Feeds
  - Google Sitemaps
  - Google Spreadsheets Data API
  - Google Talk XMPP
  - Google Toolbar API
  - Google SOAP Search API
  - YouTube API

- 대한민국 매쉬업 경진대회
  - 주최 - 다음, NHN, 장소 - 연세대, KAIST
  - 기간 – 2007.2.1 – 2007.3.31
  - 경진대회 - http://mashupkorea.com/
  - 주요 프로그램
    - Daum API 소개 - 인증, 검색, 블로그, 디앤샵, 여행
    - Naver API 소개 - 검색, 데스크탑, 지도
    - 멘토링 (10대 과제)

Conclusions

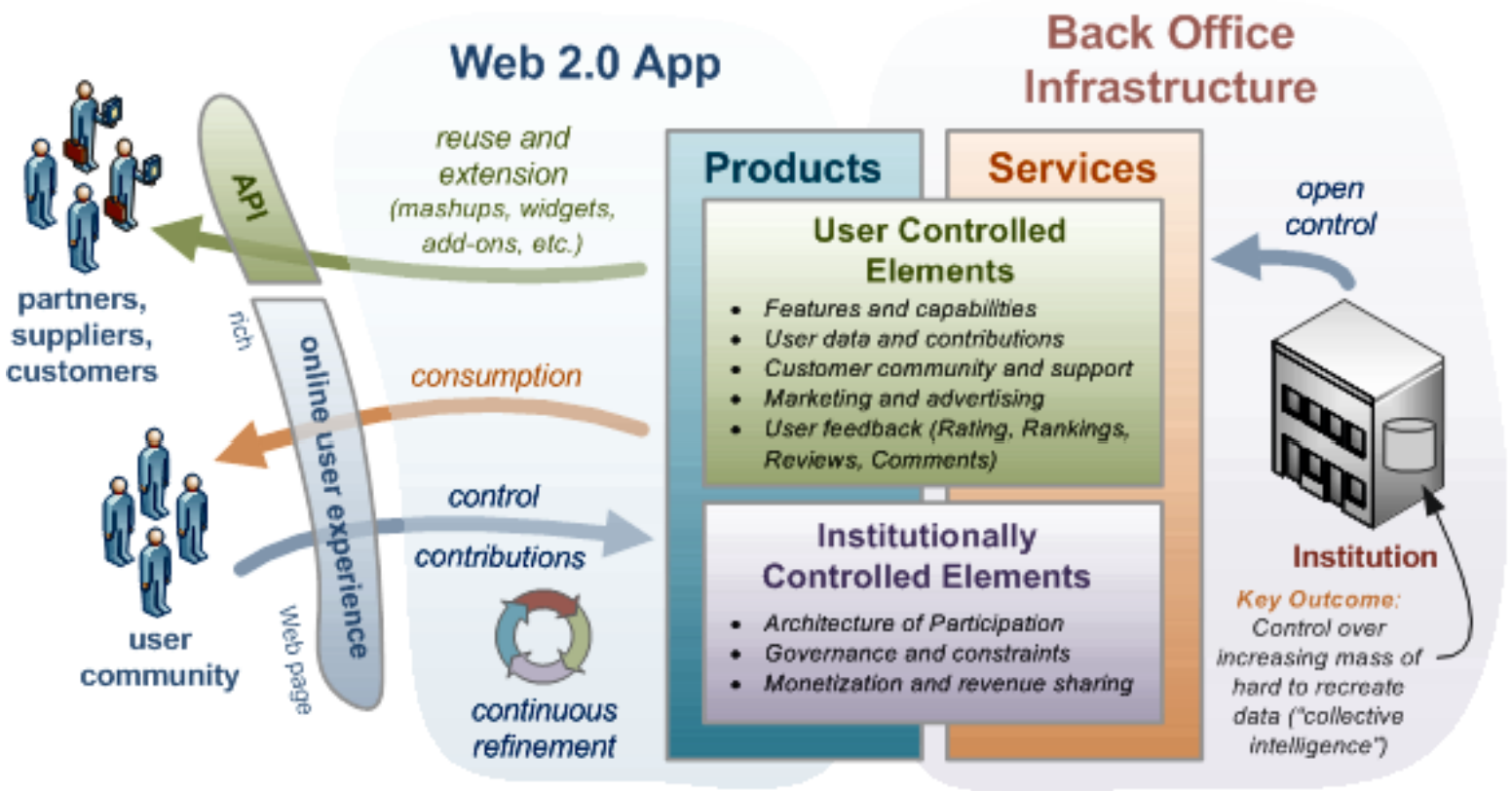
- Openness, Open Technology, Open Architecture, Open Source
  - 개방형 체계를 통한 위험의 분산
- Open API
  - API 개방을 통한 활용성의 증대
  - Open API의 범위와 권한 : Internal, External..
- Mashup
  - 서로 다른 서비스를 합친 새로운 서비스
  - NOT (mashup == Open API)
  - 다양한 매쉬업 시도의 증가 : Enterprise, Mobile....
- Mashup + Open API
  - 빠른 서비스 개발 가능
  - 기업간 이용을 위해서는 해결해야 할 이슈들 : 신뢰, SLA, 지원 ...
- Lightweight Framework + Mashup + Open API
  - 빠른 경쟁 환경에 적응하기 위한 시도 !!!

Web of Services for Enterprise Computing

- W3C Workshop on Web of Services for Enterprise Computing
  - 27 to 28 February 2007, MITRE, Bedford, MA, USA
  - 25 Position Papers
  - Program
    - http://www.w3.org/2007/01/wos-ec-program.html
    - Day 1: What's missing from the picture - new stuff to consider
      - ❖IONA, MITRE, Xerox, Hartford, Redhat, Gestalt, WSO2, FSTC,
    - Day 2: Separate or Together? (i.e. one Web or two architectures)
      - ❖BT, Coactus Consulting, Yahoo, HP, BEA, Progress Software, IBM, W3C TAG

The Move to Product Development 2.0

Product Development 2.0: Using the Web to Put Users in Control and Co-Create Better, Richer Products Faster



Source: http://web2.wsj2.com

http://web2.wsj2.com/product_development_20.htm

The Move to Product Development 2.0

	Product Development 1.0	Product Development 2.0
Primary Customer Interaction Channel:	Telephone, Mail, Face-to-Face, One Way Media (Print, TV, Radio, etc.), e-mail	World Wide Web, e-mail, IM
Source of Innovation:	Organizations	Customers
Innovation Cycle:	Months, Years	Minutes, Hours, Days, Weeks
Content Creators:	Internal Producers	External Producers
Feedback Mechanisms:	Market research, satisfaction surveys, complaints, focus groups	Analytics, online requests, user contributed changes
Customer Engagement Style:	Controlled, well-defined process	Spontaneous and chaotic
Product Development Process:	Upfront design	Less upfront, much more emergent
Product Architecture:	Closed, not designed for easy extension or reuse by others; walled garden	Open, very easy to extend, refine, change and add on to, ecosystem friendly, designed (and legal) for widespread remixing and mashups
Product Development Culture:	Hierarchical, centralized, Not Invented Here, somewhat collaborative, expert-driven	Egalitarian, decentralized, remix instead of reinvent, highly collaborative, Wisdom of Crowds
Product Testing:	Internal, dedicated test groups, hand-picked select customers	Users as testers
Customer Support:	Customer Service	User Community
Product Promotion:	One-Way Marketing and Advertising	Viral propagation, explicit leveraging of network effects, word of mouth, user generated and other two-way advertising
Business Model:	Product Sales, Customer Service and Support Fees, Service Access Charges, Servicing High Demand Products	Advertising, Subscriptions, Product Sales, Servicing All Product Niches (The Long Tail), Unintended Uses
Customer Relationship:	External Buyer (Consumer)	Partner and -- increasingly remunerated -- Supplier (Consumers as Producers)
Product Ownership:	Institution, particularly executive management and shareholders	Entire User Community
Partnering Process:	Formal, explicit, infrequent, mediated	Ad hoc, thousands of partners online, disintermediated
Product Development and Integration Tools:	Heavyweight, formal, complex, expensive, time-consuming, enterprise-oriented	Lightweight, informal, simple, free, fast, consumer-oriented
Competitive Advantage:	Superior products, legal barriers to entry (IP protections), brand name advantage, price, popularity, distribution channel agreements	#1 or #2 market leader, leveraging crowdsourcing effectively, mass customization, control over hard-to-create data, end-user sense of ownership, popularity, cost-effective customer self-service, audience size, best-of-breed architectures of participation

http://web2.wsj2.com/product_development_20.htm

Thank you

For more discussion :

JongHong Jeon (hollobit@etri.re.kr)
+82-42-860-5333

Blog : http://blog.webservices.or.kr/hollobit

OR