CASCON [2007]

The Future of User Interfaces in Web 2.0

Wolfgang Stuerzlinger
York University, Toronto

CASCON [2007]

Outline

- **Analysis & extrapolation of a few trends**
  - Web 2.0: the new Desktop?
  - Collaborative computing in Web 2.0
  - Offline applications & storage in Web 2.0
- **Implications & predictions**
  - Changes to development process
  - A few observations & predictions

CASCON [2007]

Why is Web 2.0 successful?

- **Web 2.0 offers services**
  - Not applications
- **Any time, any platform, any location**
- **Users are in control**
  - Control access to their data
  - Choose their services, add to them, collaborate via them,…
- **Harness collective intelligence**
  - E.g. transition from Taxonomy ⇒ "Folksonomy" & Tagging
- **Offers real advantages to many people**

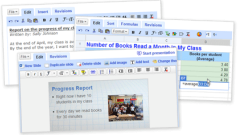
CASCON [2007]

Web 2.0: The New Desktop?

CASCON [2007]

Web 2.0: The New Desktop?

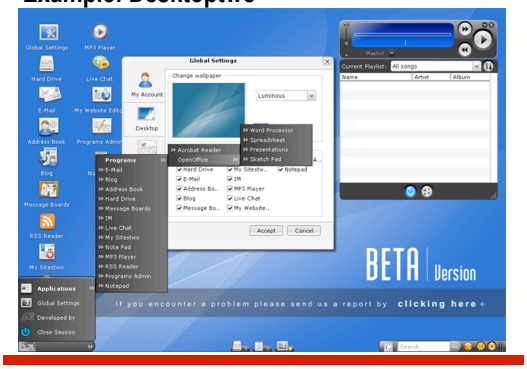
- **Most traditional applications available in Web 2.0**
  - Gmail, Google Calendar
  - Google Docs
  - …



- **Seems to obviate need for traditional "desktop"**
- **Web Operating System**
  - Operating system under browser still necessary
    - But a blurry boundary (iPhone)

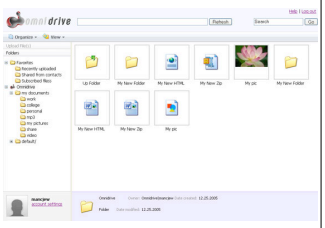
CASCON [2007]

Example: Desktoptwo



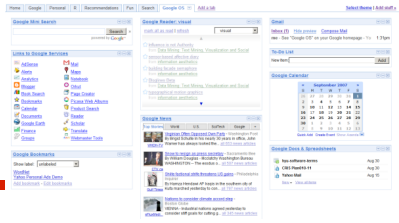
Web Desktop with Files

- **Clone Desktop Idea**
  - Advantage: familiarity
  - Disadvantages
    - UI performance hit
    - Speed performance hit
    - Files - again?
    - Not really "Web 2.0"
      - Web 2.0 is about *services*
      - Also no collaboration support - desktop is inherently single user



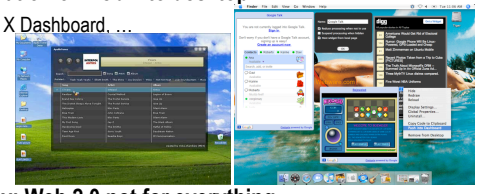
Web 2.0 Equivalent: Service Aggregation

- **Different mental model**
  - Clean break better (less confusion)
- **Example: iGoogle**
  - Configurable by user, everybody's needs are different!



Another way: Desktop Widgets

- **Integration of Web into desktop**
  - OS X Dashboard, …



- **Finally: Web 2.0 not for everything**
  - Most 3D games, high-performance computing, big simulation, visualization, etc.

Collaborative Computing in Web 2.0

Collaborative Systems - CSCW / Groupware

- **Collaboration**
  - Group of people with common goal
  - Goal large enough so that it can't be tackled individually
- **Computer Supported Collaborative Work (CSCW)**

- **Lot's of *collaborative* services on Web 2.0**
  - Email, encyclopedia, scheduling, office applications, …
  - CVS, bug tracking, feature requests, …
  - Internal company information systems, …

CSCW/Groupware Development

- **Different from software for individuals**
  - Developers can *never* know all potential users & their jobs
    - Many different target users
    - Managers are biased, too
- **Critical mass issue**
- **CSCW apps must support**
  - Information sharing (easy on Web)
  - Awareness (hard on Web)
  - Coordination (hard on Web)

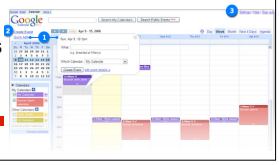
Four CSCW/Groupware Design Principles

- **Maximize Personal Acceptance**
  - Visible benefits, no training, …
- **Minimize Requirements**
  - Any time, location, platform, unobtrusive collaboration, …
- **Minimize Constraints**
  - Support roles, allow exceptions, …
- **External Integration**
  - Data exchange, …

Based on [Cockburn94, Grudin94]

Maximize Personal Acceptance (1)

- **Address disparity in work and benefit**
  - CSCW often more work for those who don't benefit
  - Example: group calendars
    - Manager able to control your time
      - ⇒ People don't use it (rejection)
      - Or people block days at a time (subversion)
  - Reduce burden to non-beneficiaries
    - Easy-to-use, efficient to use
  - Generate personal (!) benefits
    - E.g. private use of Blackberry



Maximize Personal Acceptance (2)

- **Facilitate adoption**
  - Example: hospital, only 1 of 5 nurses uses SW ⇒ disaster
  - Low entrance threshold
    - Easy-to-learn design
- **If training required, make it painless**
  - On site, multiple times, user's schedule
  - Alternative: Screen videos/tutorials
    - Need to be done well, too

Minimize Requirements & Overhead

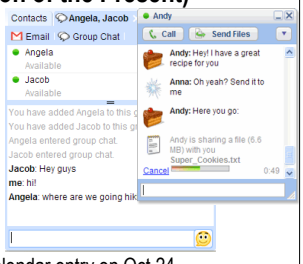
- **Any time, any location, any platform, …**
  - Web 2.0 good at that (except not offline)
- **Unobtrusive CSCW features**
  - Information sharing *and* editing
    - Mostly reading, but *encourage* contribution
  - Awareness
    - What are others currently doing?
  - Coordination
    - Identification of ownership, changes, etc.
- **Features must be present but *never* predominant!**

Information Sharing & Editing

- **Example: Wikis**

Awareness (Coordination of the Present)

- **Well known in Chat**
  - "Jill is typing …"
- **Helps avoid conflicts**



- **But can be much more**
  - Jack is currently adding calendar entry on Oct 24
  - John is currently editing part X of project Y
  - Jim started editing "Future plans" on current page at 3pm

Coordination of the Past

- **Who did what, when?**
  - Not just retrieval, but also comparisons, etc.

Coordination of the Future

- **Who is planning what?**
  - Example: Wiki discussions, roadmaps, …

Minimize Constraints

- **Support roles and exceptions**
  - Organizations have both formal & informal work patterns
- **CSCW applications tend to flatten hierarchies**
  - May disadvantage some users & threaten their "domains"
- **Exception handling**
  - "This needs to be authorised by Ms. S but she is away for 3 weeks" … "I'll see if I can get Mr. J to sign for her"
  - Humans good at exception handling & improvisation
  - Flexibility is key here!

External Integration

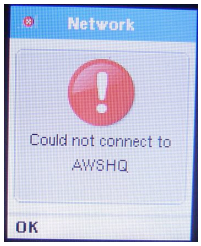
- **Permit other tools**
  - Bi-directional data exchange with other systems
    - Another way to ease adoption
    - Helps with emergency back-out
      - If your system is fulfills user's needs well, this will never happen
      - Reassurance factor
    - Web 2.0 services must allow users to retrieve *all* their data
      - Otherwise: lot's of workarounds

More Reflections on Collaboration in Web 2.0

- **One of the key features of Web 2.0**
  - Increasingly more collaborative features
    - In the future more: reviewing, commenting, …
- **What works**
  - Leave individual user in control
    - Flexibility is key
    - Otherwise abandonment or lot's of workarounds
  - Do one service well
    - Don't try to do everything at once
  - Support user fully in their tasks

Offline Web 2.0 Applications

Offline Web Applications



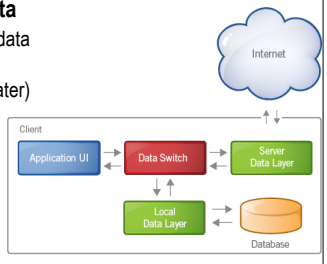
- **Main deficiency of Web 2.0**
  - No support for *offline* work
    - I.e. when no network connectivity
      - Algonquin park
      - Subway
      - Foreign country
      - No network
      - Insecure environment
      - …
- **Offline apps already available today**
  - E.g. OS X dashboard "widgets" are Javascript!
- **Problem: no persistency**

Source of Inspiration

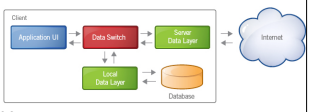
- **Mobile Computing Platform**
  - PDA's & Smartphones
    - Palm/Treo, WM6, Blackberry, iPhone, etc.
  - Addition of wireless networks serious "boost"
    - "Crackberry" addiction
    - Some people use mobile computing more than desktop!
  - Best user interfaces: iPhone, Palm

How to Enable Offline Work

- **Keep local copy of data**
  - Usually only "touched" data
    - Cache
  - Better a bit more (see later)
- **If network**
  - (Upload local updates)
  - Use network directly
    - Cache requests
- **If no network**
  - Use local copy
  - Store updates



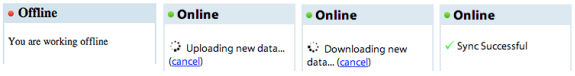
Synchronization



- **Local copy**
  - User edits data when offline
  - Changes to be uploaded on re-connect
  - Also download recent changes by others
- **Conflict resolution**
  - Easy if only one side updated since "time of last sync"
    - Newest change wins
    - Better ideas … (later)
- **Maps well to Web 2.0 - AJAX!**

User Interface for Synchronization (1)

- **User's view syncing as a bug, not a feature.**
- **Need basic UI that sync is occurring**
  - Diagnosis when network is acting up or sync is failing



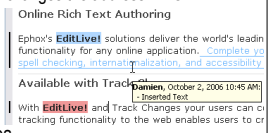
- **Sync should happen automatically**
  - On first load, network (re-)connect, periodically (autosave)
  - Users rarely can predict when they will be offline
    - Local copy of "essential", "owned", and "touched" data

User Interface for Synchronization (2)

- **Users don't understand diff/merge interfaces**

User Interface for Synchronization (3)

- **System needs to make *reasonable* sync decisions**
  - Ideally fine-grained
    - E.g. A changes title of contact, B changes the address ⇒ OK!
  - For longer documents
    - Annotate each change
      - E.g. "Track Changes" in Word



  - In conflicts, last change survives
    - However, overwritten version must be stored in history
      - Note: history anyways necessary for collaboration!

Problems with Synchronization

- **Issues to think about**
  - Network delays (async!)
  - Many changes (async!)
  - Partial syncs due to network drop, crashes (both sides)
  - New changes can occur during a sync

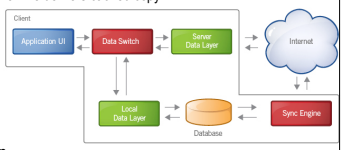
- **User should never loose data**
  - System never in unusable or inconsistent state

Offline Technologies: "Hacks"

- **Store data in Cookies**
  - Well-known problems
- **Store data via Adobe Flash (shared object), ActiveX (COM), XPCOM (Firefox), …**
  - Limited amounts, not platform/browser independent
- **Form data storage**
  - Limited amounts
- **Internet Explorer DHTML feature**
  - Limited to 60k
- **Store application/data as URL's**
  - Fairly good, up to limit of browser URL cache

Offline Technologies: Google Gears

- **Three main parts**
  - Local Server
    - Stores all Web requests, if offline delivers cached copy
  - Database
    - SQL lite (variant)
  - Worker Pool
    - for long(er)-running tasks



  - Also: Sync Engine
    - Worker for synchronization
- **Available for PC, Mac, Linux**
  - Currently not yet on Palm, iPhone, WM6, …

Offline Technologies: Others

- **Dojo Offline Toolkit**
  - Part of Dojo AJAX framework
  - Support for offline storage
    - DojoSQL and DojoStorage (key-value pairs)
    - Encryption for stored data
  - Support for synchronization
- **HTML 5 (draft standard)**
  - Offline storage of key-value pairs as well as SQLite
  - Will be available in Firefox 3, …
    - Most probably also in IE
  - No support for synchronization, yet

Changes on SW Development in Web 2.0

Effects on Development

- **Shift from applications to services**
  - More users ⇒ more beginners
- **Hence, usability #1 priority, not more features**
  - Design for user's needs
  - Do not maximize functionality, *minimize* it
    - More features make *everything* harder to find
    - User has no time for tutorials, documentation,…
      - "Ballons" may work well
    - Alternatively, 3 different levels of service
      - Beginner, intermediate (most users), expert (a few)
  - Back button = undo for everything
    - Store changes even across sessions
  - Good visual design, avoid web terminology, KISS, …
    - Allow bookmarking of screens

Safety / Security / Availablility

- **Services should be always available**
  - Google already "world-wide critical infrastructure" :-)
  - 99.999+% availability
    - Offline support helps here
  - Ideally no crashes (server & client)
- **Encrypt data**
  - Transmission
  - Local cache
  - Never share user's data without explicit permission

Technical Web 2.0 Trap

- **Update everybody's "program" at almost any time**
  - Advantage & disadvantage
    - Fix everybody's service immediately
    - Break everybody's service immediately
      - Hard for user to revert in this case, as no local copy (!)
        - » Disaster
    - Another problem: caches not always updated immediately
      - Can be months: 4 week vacation, 2 week conference, 2 week sick
- **Necessitates good version management**
  - Must always allow upgrade "old" data versions
    - With *no* user effort (or at the worst minimal effort)

Web 2.0: Opportunity

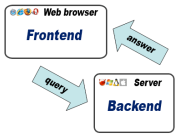
- **Get immediate feedback from actual use**
  - Continuous feedback
  - Fast identification of problems
    - Is a feature actually used?
      - Monitor successes
    - When & where do users abort / fail?
      - Monitor partial "transactions"
  - Fast adaptation possible
- **Key aspect of Web 2.0 business model**

Future Web 2.0 Toolkits

- **Collaboration support**
- **Offline & sync support**
- **Encapsulate Web 2.0 UI "look & feel"**
- **Current examples**
  - Dojo
  - Google Web Toolkit + Google Gears
- **Future**
  - Web 2.0 UI toolkits
  - Web 2.0 CSCW toolkits
  - Web 2.0 Offline & sync toolkits

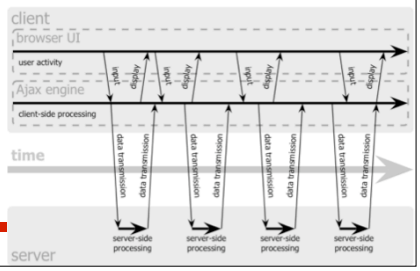
Effects on Development Teams

- **Web 2.0 necessitates "back-end" & "front-end"**
  - Front-end: Focus on users & their tasks
    - Mix of IT, UI, CSCW designers
    - Typically more code here than at backend
  - Back-end *supports* front-end
    - Must not dictate front-end development
    - Focus on system stability, data security, performance
    - Often shared among different projects
  - Both need to collaborate to succeed
    - Special needs back-end code similar to "middleware"



Web 2.0 SW Development Model Change

- **AJAX fundamentally changes code**
  - In particular: "Asynchronous"
    - Hence, back-end cannot control UI



Closing Thoughts

My Advice for Web 2.0

- **#1: Test your user interface**
  - Even more important than on desktop
- **#2: Get the basics right (standard Web UI guidelines)**
  - Simple design, consistent UI, clear navigation, good feedback, error resistant, as simple as possible, CSCW, offline & sync, …
- **#3: Communicate clearly**
  - Users want to spend minimum time
    - Quickly & easily convince them that your site is worthwhile
- **#4: Provide what users need**
  - Match their expectations & needs
    - Research requirements most carefully

Technological Advice for Web 2.0

- **Technical advice**
  - None (on purpose)
    - Technology is just a means, not an end!
  - Support collaboration
  - Support off-line storage & synchronization
    - Both will become common in Web 2.0
- **Remember: the "invisible" computer wins**
  - See your *parents* car, cell-phone, etc.
  - Our job to make Web 2.0 as invisible as possible

Disclaimer

The IBM Center for Advanced Studies (CAS) regularly publishes technical documents that are aimed at facilitating an exchange of information. These reports are written by individuals or groups of authors and represent their opinions and do not necessarily reflect those of IBM. In the event of questions or concerns regarding individual reports or presentations, email addresses have been provided for most authors. Alternatively, please feel free to contact CAS at casinfo@ca.ibm.com. CAS is dedicated to providing a forum for IBM employees, research affiliates, and university students to publish results of their work, their views on technical issues, book reviews, and workshop reports.

Trademarks