LimeBits: Open-source community for sharing, recycling, customizing, and improving web code

Jonathan A. Marshall

2009-06-22

Summary

Code reuse decreases software development time and cost. Developers integrate existing components, instead of building them repeatedly from scratch. Reuse has succeeded widely for small chunks of code, like application component libraries.

However, larger-scale code reuse is perennially hobbled by practical problems. For example:

- When code is separated from its original development environment, context is lost [Reindel 2008] and must be reconstructed laboriously.
- Selecting good, reusable components requires significant effort and organization [Krueger 1992].
- Larger components are more specific [Krueger 1992] and need more custom tailoring for each reuse.
- Software components carry diverse, hard-to-understand, and often incompatible licensing terms.

LimeBits tries to solve several such problems and make code **recycling** – a form of reuse – practical for the first time. LimeBits provides code **in context** – in complete, running, modifiable programs. It combines development with hosting, so code modifications recycle to the repository automatically and effortlessly. It features open-source licensing, with tracking and compatibility guidance.

Introduction: Code recycling in LimeBits

LimeBits is a new platform for recycling of websites and web code. It facilitates a self-reinforcing cycle of code use, sharing, and reuse:

- Developers join LimeBits because it has a pool of websites and code that they want to reuse.
- They create modifications and improvements of the sites and code for their own benefit, and keep them live in LimeBits website hosting.
- Their modifications are automatically available to other developers. The pool thus grows and becomes more attractive.

Because LimeBits supports both development and hosting, developers need not break the cycle. They need not carry code in LimeBits elsewhere for further development; and need not carry their modified code elsewhere for live hosting. With participation of many developers, improvements can compound.

LimeBits vision

The LimeBits project springs from a vision of creative community. In general, members both benefit from a pool of community assets and incrementally improve the assets. Their improvements recycle back into the pool. This incremental creation cycle is a special form of *peer production* [Benkler 2002].

Creative community imposes special requirements. Copyright, access, attribution, derivation, and fair use are key concerns. Community members may want their creations to remain available to members, or to the public at large, rather than being arrogated for private benefit. Likewise, they may want acknowledgement and credit for their contributions.

LimeBits implements support for such requirements.

- LimeBits provides infrastructure and tools specifically for mass creation. It will automatically track authorship, versions, and derivation. It lets authors control access and usage for their contributions. It communicates license information between authors and users. It allows commenting, discussion, and notifications.
- LimeBits is standards-compliant. It interoperates with existing software platforms via standard, widely supported Internet protocols. It will mandate nothing proprietary or vendor-specific.
- LimeBits infrastructure is open-source. Members will be encouraged to improve and extend openly the community infrastructure itself.
- LimeBits is free of charge. It will be supported by charges for optional, ancillary services, so that LimeBits community members can participate without risk of bait-and-switch.
- Works in LimeBits reside by default in a commons, available to all. Although contributors may optionally set proprietary licensing and restrict file access, LimeBits encourages them to provide works on an open basis.

Creative community can ease and extend creativity. Because members share assets, they gain access to more resources than their own private creations. Members may need only to create incremental modifications, rather than whole new assets from scratch. The community may offer communication with peers, who can provide ideas, help, and specialized work.

LimeBits application: website code

The LimeBits concept applies best to domains of *incremental derivation*. That is, an author improves, extends, or customizes an existing contributed work, and then contributes the resulting derived work back to the community – for further derivation.

One such domain is *website code*. Most websites have similar underlying function and structure to other websites. In principle, it should be easy to develop a website by reusing and modifying an existing site.

However, a technical barrier has hindered incremental derivation of websites. For most websites, code is hidden, uncopyable, on the webserver. For reasons of cost, security, and competition, website owners have little incentive to reveal their code or allow modification.

LimeBits lowers the technical hurdle, via a nontraditional architecture in which all website code is inherently visible, open, copyable, and modifiable. LimeBits also provides incentives for owners and authors to allow derivation of their websites: thrift, vetting, improvement, and renown.

Another domain of incremental derivation is public knowledgebases. For example, participants in Wikipedia contribute and incrementally edit encyclopedia articles. The content is then available as a public commons. LimeBits naturally supports such content commons and provides wiki software tools.

Other domains of incremental derivation include visual design libraries and music/sound collections, when properly licensed. Images can be incrementally altered, processed, and reassembled. Likewise, music tracks can be remixed, filtered, sampled, and recomposed into derived works. LimeBits can potentially support incremental derivation in these domains too.

Code in context

LimeBits provides web code *in context* (fully working, pluggable components or complete websites), rather than isolated subroutines or snippets [Reindel 2008]. A developer merely picks a near-match for the desired website, and then edits it to suit. This may turn out far easier than traditional software development, for two reasons:

- the amount of coding is small, compared to building from scratch; and
- the existing, working code is always available as a test benchmark; any incremental change that breaks correct behavior can be detected immediately.

In earlier forms of reuse, developers *construct* software, by combining abstracted components [Krueger 1992].

By contrast, developers using LimeBits *deconstruct* software. They start from whole programs (websites), burrow in to find the parts to change, customize or extend incrementally, and test often to ensure that functionality is never lost.

What makes this form of reuse now possible is the explosive proliferation of network communications [Benkler 2002] and of webcode in recent years. Previously, a given software function might be implemented in a handful of instances. Now, it is common to find thousands of variant instances of a function embedded within sites across the web.

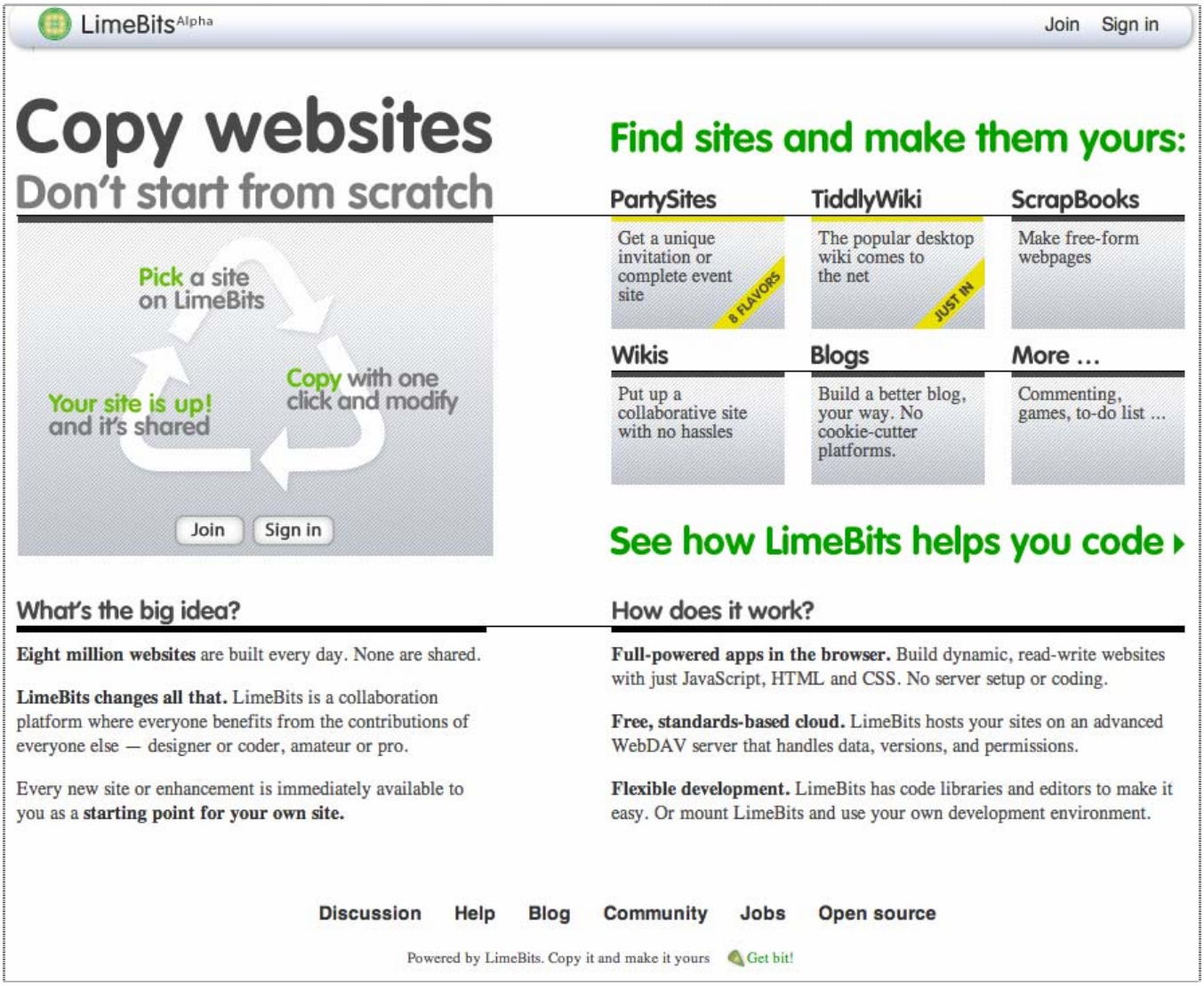
LimeBits is different

LimeBits acts simultaneously as a community website, storage server, code development environment, and webhost. It improves on traditional web services and technologies in several ways:

- *Community.* LimeBits organizes community members noncompetitively, to expand and encourage open, collaborative creation, yet allows project discipline and control. No fear of experimentation, no risk of exclusion.
- *Clarity.* LimeBits simplifies and clarifies the programming model that developers must learn, yet preserves the power and flexibility of dynamic websites. No backend coding, no PHP, CGI, SQL, Java, Perl, Ruby.
- *Context.* Developers using LimeBits can copy working programs and modify the code in context, keeping increments testable against functionality. No need to build from scratch.

How to use LimeBits

LimeBits is available to the public, currently as an alpha deployment, at the limebits.com website. Visitors see an introductory gallery of member-created websites and components, available to be fully copied. They sign up for a free account and then copy and modify existing websites and applications.



Users explore the gallery and discover sites that members have placed into LimeBits. A user might see a website and think "I like that." LimeBits provides a Copy button, which lets the user go to the next step – "I want that" – or even "I want something similar."

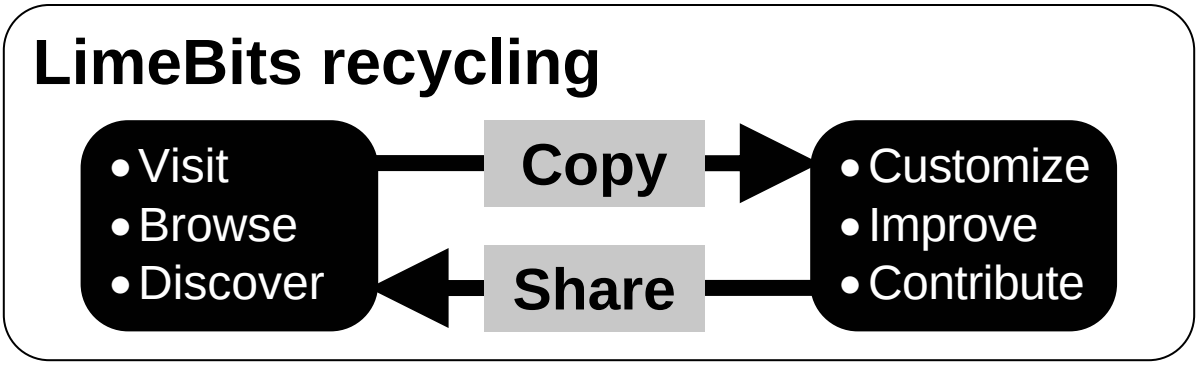
For example, a dentist browsing LimeBits might see websites that other dentists have created, find one that has desired features (e.g., appointment signup, interactive directions, checkup reminder), and then click Copy. LimeBits gives the dentist a fully working, customizable, modifiable copy of the website.

Using LimeBits, the dentist easily edits the website content and code, changing name, office address, images, look-and-feel, etc. Because the website was working to begin with, the dentist can easily keep it working while making the changes.

LimeBits includes some websites that are particularly easy to modify. For example, the ScrapBook web application lets LimeBits members quickly add and position components (text, images, blogs, wikis,

games, navigation, etc.) onto a webpage. ScrapBook also provides quick and simple editing access to text and styling.

Modifications are automatically shared back to the LimeBits community.



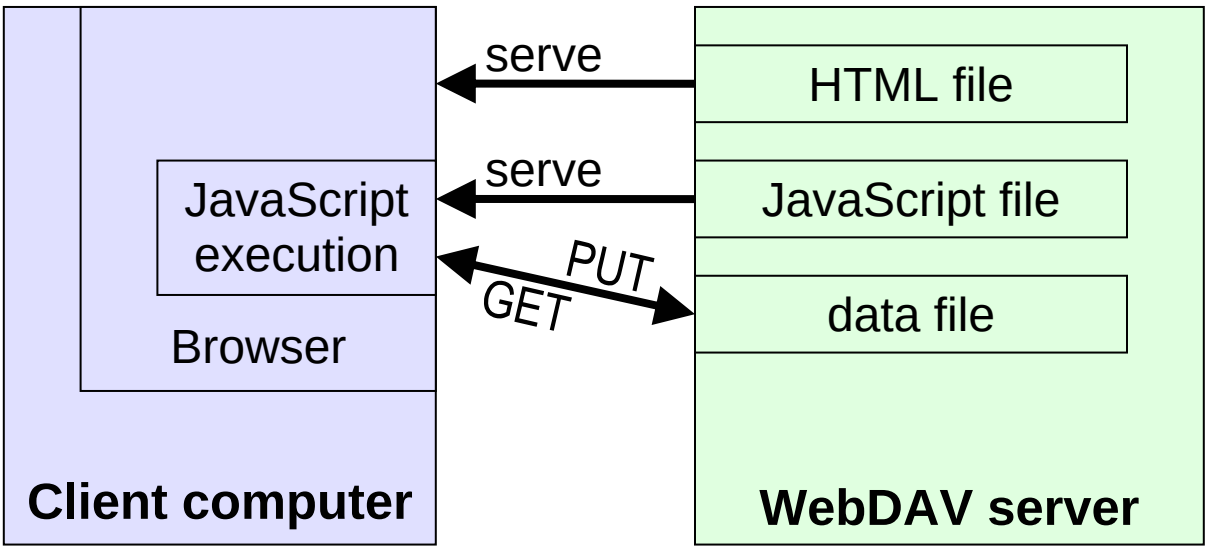
LimeBits unifies development and hosting into a single environment. The input and experience of both users and developers are available community-wide, better informing the development efforts. Improvements can be developed by any member and applied to production code easily and quickly.

Thin server architecture

LimeBits uses a system architecture that is both simple and novel. It starts with a "thin server." That is, the LimeBits server is designed to support only data storage, plus minimal, ancillary functions. Unlike traditional webservers, the LimeBits server does no application computing.

The LimeBits server implements the WebDAV protocol, a widely-used standard for the read-write web. WebDAV is a standard extension to HTTP which enhances REST storage, metadata (name-value properties), collaboration (version control, locking, overwrite protection, access-control groups, permissions), search queries, and other useful features. WebDAV is widely supported by software vendors including Adobe, Apple, GNU/Linux distributions, IBM, Microsoft, and Sun.

In the LimeBits architecture, application code runs on client computers, typically as JavaScript in web browsers. The client-side code, in turn, accesses server-side storage via a JavaScript library interface to WebDAV. The client code stores and retrieves state, variables, documents, etc. on the LimeBits server.



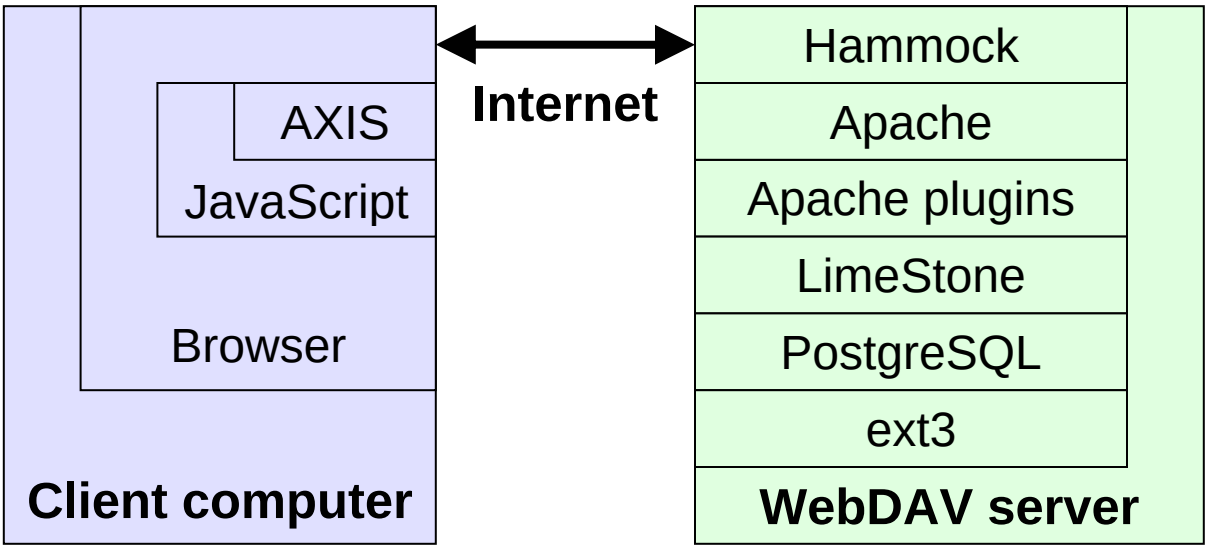
From the developer's viewpoint, LimeBits obviates the need to learn server-side programming; no PHP, Java, Perl, Ruby, ASP, CGI, etc. Developers make full-powered websites with only JavaScript, HTML, and CSS knowledge. Developers may use JavaScript libraries like jQuery, YUI, ExtJS, Prototype, Dojo, and MooTools. They may also integrate websites into LimeBits via a client-side HTML wrapper.

The thin-server architecture scales exceptionally well. LimeBits serves only static content; no server-side construction of dynamic webpages is required. Webpage dynamics execute on the client side, where computing power scales directly with number of users.

LimeBits implementation

LimeBits is based on Internet standards and open source, with no lock-in to proprietary technologies.

The LimeBits WebDAV server, named LimeStone, is a custom implementation, written in the C language and deployed as an Apache module. It is a substantial rewrite and extension of the open-source Catacomb WebDAV server. LimeStone also uses several Apache plugin modules, to manage request redirection and authentication.



LimeStone stores metadata in a SQL database (under PostgreSQL) and stores file contents as Linux ext3 files.

LimeBits has a custom front-end proxy named Hammock, which transparently gives WebDAV-noncompliant browsers full access to WebDAV.

LimeBits also includes some ancillary functions separately: emailing password-reset links, mapping external hostnames to internal URLs, and WebDAV compliance testing.

The LimeBits server runs on a Debian Linux distribution.

Bit architecture

Unlike widget collections, LimeBits offers free-standing, whole websites. It also provides a mix-and-match capability, allowing multiple sites to be combined into new compositions.

LimeBits client-side software is designed around the concept of a "**Bit**" – a relocatable unit of defined functionality. "Bit" is a general term that includes web sites, web apps, and web plugins.

A Bit should

- be working code,
- be copyable as a single unit, in a single operation, and
- be relocatable anywhere, continuing to work in its new location.

The Bit architecture requires no modification or special format for non-LimeBits code. An existing static website containing JavaScript can be placed into a directory in LimeBits. Unless the code interacts with a server, the site should work without change. Server interactions, for example via AJAX, may need to be modified for the LimeBits server.

LimeBits provides extra capabilities for Bits that adhere to the recommended format. For example, if the Bit contains an icon image file in a predefined location, LimeBits can show the icon in appropriate parts of its display. If the Bit stores its data content in a predefined location, LimeBits can automatically clear and reset the data on copy.

LimeBits is architected specifically for sharing, copying, and recycling. It offers complete access to the website code, not just configuration parameters. LimeBits makes it easy for developers to install configuration capabilities in their code.

LimeBits also lets developers mix Bits together – combine them on a webpage. Some Bits are designed to interact, extending or altering behavior.

Site functions

The limebits.com website provides several functions to members, by means of client-side JavaScript interacting with WebDAV server storage.

Basic site functions

- Member accounts. Signup, login/logout, secure cookie-based authentication, digest authentication, optional single-signin or user ID privacy.
- File and directory operations. Navigate, file upload, create, view, edit, copy/move, delete to Trash, metadata display.
- Remote mounting. LimeBits can be mounted as a WebDAV filesystem into other computers. This makes editing, uploading, and downloading files especially easy.
- Cross-subdomain copy. LimeBits protects members' data by respecting cross-domain restrictions in browsers. LimeBits creates a separate `member.limebits.com` subdomain for each member. In addition, LimeBits lets members copy files from other members, subject to security and privacy restrictions.
- External domain mapping. LimeBits lets members map their non-LimeBits domain names into their LimeBits folders. Visitors see the external URLs.
- Optional injection of toolbar. LimeBits adds a toolbar into every webpage for users exploring through the LimeBits site. External visitors see no toolbar.

- Code packaging. LimeBits makes websites easily copyable, via a single click. LimeBits simplifies much of the packaging that makes this possible, and makes it easy for developers to specify the copyable pieces.
- Metadata queries and tagging. LimeBits members may attach tag words to files. Members may query WebDAV for files matching the tags, as well as metadata such as modification date, file name, owner, and mimetype.

Security and privacy

- Fine grained WebDAV access permissions. LimeBits implements WebDAV file permission features, including user groups, access-control lists, and various levels of read and write access.
- Multi-level subdomain sandboxing. Each application in LimeBits will be accessible through its own subdomain, to protect users from wayward code. A program in `blog.joe.limebits.com` cannot delete files in `www.joe.limebits.com`. (Members may escape this restriction by marking individual apps as trusted.)
- Vetting. To inform members' trust judgments, LimeBits will allow community ratings and comments on code, as well as the ability to inspect source code.

Version derivation and lineage

- When members copy files from one another or modify their own files, LimeBits will record the version lineage. Members may view the derivation history tree, and see where and when changes were introduced, and by whom.

Community

- Monitoring and event triggers. LimeBits will let members subscribe to event notifications of interest. For example, a member may choose to be notified when a file is copied or modified, or when a program encounters an error.
- Autocommunity and notifications. LimeBits will automatically create and maintain a community around each Bit. Members will be entered into a Bit's community when they take certain actions, such as viewing or running the Bit, copying it, or modifying it. They may choose to receive notifications of events relevant to the Bit, such as views, copies, modifications, or errors.

Applications

- LimeBits provides several seed apps to demonstrate LimeBits functionality. These include wikis, blogs, commenting, documenting, and games. They are intended to be modified and improved by members.
- LimeBits provides several libraries to ease website building. *Axis* enhances the power of JavaScript by adding read-write storage, via WebDAV. *BitMix* will be a general container for mix-and-match components. *TextBox* is an HTML component with built-in editing. *Conversation* is for representing structured sequences of text.

LimeBits creative community

A crucial ingredient for incremental derivation is *community*. That is, value comes from the creative interactions *between* users, not just the contributions of individuals.

LimeBits supports creative community in several ways. It organizes community efforts noncompetitively, allowing multibranched development while preserving branch relationships, structure, and developer communication globally.

- **Copy lineage.** LimeBits will keep track of the version history of modifications. It will do so even between users. When one user copies and modifies a file, the version relationship will be preserved. On LimeBits an author can ask "What improvements have community members tried to make to my code?" A developer can ask "What changes did people make recently to this code?" and then check whether the changes introduced flaws.
- **Autocommunity and notifications.** On LimeBits, every work will be the nucleus of a mini-community; users who come into contact with the work become members automatically, according to the nature of their contact. As an example, for a blogging app, those who
    - copy the code become added to the app's community as copiers,
    - modify the code become added as modifiers,
    - write comments on the app become added as commenters,
    - run the app become added as users,
    - encounter error messages become added as error witnesses, etc.

  LimeBits will automatically notify the app's community members when specified events occur. For example, the author can be notified when someone creates a modification. Modifiers can be notified when error messages are encountered. Error witnesses can be notified when modifiers publish a fix.
- **Repository structure.** Works in LimeBits will be structured to minimize conflict and foster cooperation. No code branch need be considered universally authoritative. Instead, LimeBits will maintain all code branches in parallel, so developers can determine their own criteria for following authoritative code version updates. A developer can program any version of any code file to be loaded, using any criteria – e.g., version id, modification time, owner, modifiers, ratings, or number of copies.

  These repository features are intended to alleviate "fear of forking" [Moen 1999] – the social stress of suppressing alternatives for central control.
- **License tracking.** In recognition of authors' varying needs, LimeBits will give authors tools for optionally recording and integrating their license choices, as well as reasonable defaults. Choosing from a list of standard licenses, authors will indicate the terms under which their contributions are available to the community. LimeBits will preserve the license information on copies and display compatible license choices for derivatives.
- **Division of labor.** LimeBits broadens the range of functionality that front-end web developers can provide, and decreases reliance on back-end developers. The low overhead of LimeBits's copy-modify technology favors vertical specialization. Web developers and designers gain expertise and economies of scale in narrower application domains (e.g., Reno wedding websites).
- **Culture guidelines.** LimeBits is intended to encourage broader participation in open-source creation and sharing. Introductory documents will state expectations for cooperation, support, and courtesy between users.
- **Best-practices education.** LimeBits will provide tutorials, videos, and other documentation on best practices for LimeBits software development and deployment.

- **Authentication.** Individual identity is an essential ingredient of community. LimeBits provides file ownership and control via password authentication. Users are distinctly identified and are reliably protected from one another.
- **Access control.** LimeBits provides fine-grained read/write permissioning on the server. Authors define user groups and grant or revoke file access permissions for individuals or groups.
- **Security.** JavaScript code running in browsers is sometimes unsafe. Security flaws have made users vulnerable to loss and forgery of sensitive data. LimeBits will provide several security enhancements, including subdomain sandboxing, credentials restriction, and community vetting of code.

These interaction features are intended to encourage authors to participate in the LimeBits community. They represent an attempt to reduce the social obstacles to open-source creation and collaboration.

LimeBits benefits

The main benefit of LimeBits is lower cost of development. Lower cost translates to faster development, more products developed, higher quality, more customers served, and broader participation [Haefliger 2008].

LimeBits provides a new, effective model for code reuse. Code on LimeBits is not distributed as isolated subroutines or libraries [Mili 1995]. Instead, code is found in context: within working, whole websites. As a result, developers using LimeBits can focus on modification, rather than building from scratch.

LimeBits provides whole web designs, in the context of whole websites. Designers can find icon libraries, layouts, stylesheets, typography, color schemes, and images – integrated together rather than fragmented across separate repositories.

Testing and debugging are especially easy because developers start with fully working code. Any modification step can be tested immediately to see if it breaks functionality. Developers can instantly detect and isolate bugs.

LimeBits is a complete execution environment, not just a code repository. Developers need not install or configure code, nor learn the skills to do so.

LimeBits offers a simple software architecture. Developers need to know only one programming language: JavaScript. The Axis library enhances JavaScript, giving client-side programs server-side storage. Web developers can dispense with server-side technologies like PHP, Java, SQL, ODBC, Perl, CGI, ASP, and Ruby.

The LimeBits architecture scales exceptionally well. Each additional client brings its own computing power, relieving the server of the burden of generating dynamic webpages. Thus the server can be simple and can serve relatively many clients.

LimeBits gathers and focuses community. It will build community groups around each app automatically, based on user activities. Where other systems would fork and divide, LimeBits preserves community and lineage. LimeBits will let users work on parallel branch variants without losing cross-branch compatibility.

LimeBits draws code, content, and designs online. Rather than sequestering their creative products on private hard drives, creators can share by default, in a publicly organized repository.

LimeBits supports a community ecosystem beyond the apps. Developers and designers can offer services to other members. They can specialize, providing good quality and speed within vertical niches. They can cooperate and self-organize on projects.

LimeBits amplifies the benefits of sharing code. Because LimeBits reshares modified code automatically, sharing costs developers no effort. Even if developers contribute only small modifications, they benefit from many eyes seeking bugs, and many hands making improvement.

LimeBits risks and limitations

LimeBits's novel architecture poses some risks and bears some limitations.

Security

Shared code in LimeBits could become a vector for attacks. No single solution can eliminate the danger, just as nothing prevents users from copying and running malicious code by other means.

To mitigate the danger, LimeBits uses several approaches.

- *Community.* LimeBits will support discussing and rating of code, as well as trust and reputation ratings for community members.
- *Open source.* Users may view and analyze the code, rather than blindly executing compiled binaries.
- *Subdomain sandboxing.* Copied code will be accessed by default in a subdomain isolated from users' other files. The code may read and write within its own app-specific subdomain but not within other subdomains.
- *Education.* LimeBits will teach users about the risks of copying code, via onramp guides, documentation, and warning alerts.

Validation

JavaScript can easily validate data in the browser client, but the WebDAV standard defines only weak facilities for validating input data on the server. WebDAV allows simple datatypes (e.g., a 5-digit field for postal codes) to be defined for metadata properties and can reject nonmatching data.

Stronger validation is required on the server. For example, suppose that an application assumes that a data field contains only *valid* postal codes (a subset of all 5-digit sequences). Validity can be determined by lookup in a large table. WebDAV provides no mechanism for this sort of content validation.

A platform that lacks such validation cannot support applications requiring strong data integrity.

LimeBits will provide strong validation via a limited form of server-side code execution. The app developer's validation code will be written in JavaScript, stored in the server, triggered by WebDAV requests, and run in a secure sandbox in the server. The code will return a binary valid/invalid result that determines whether the server rejects the request. Side effects may be disallowed.

Crawlability

LimeBits webpages may contain text injected dynamically by JavaScript code. Current web search engine crawlers do not execute JavaScript and hence cannot index injected text.

LimeBits will alleviate this problem by executing JavaScript for crawlers (not for ordinary browsers), and serving the rendered DOM. LimeBits will detect search engine crawlers via their UserAgent header and IP address. The server-side execution environment will be properly isolated and sandboxed, for security.

Efficiency

Data flow over the Internet between server and client. LimeBits implements several measures to enhance efficiency of network communications.

- LimeBits will support aggregation of requests, so that multiple items may be transmitted at once. JavaScript code may organize data structures and may request entire files or component ranges.
- LimeBits supports client-side caching of XmlHttpRequest data. An app may choose to use cached data when available, to reduce network communications.
- LimeBits will implement server-side aggregation primitives. A client may request that the server transmit the result of basic primitives like SUM, MAX, and COUNT, rather than transmitting the data to be aggregated.
- Because the LimeBits server does essentially no application computing, it has low response latency. This keeps the request times short.
- LimeBits will support browser KeepAlive, to reduce the overhead of establishing TCP connections and transmit multiple requests per connection.
- LimeBits supports standard gzip, which shrinks the volume of data transmitted.

Vendor lock-in

Many proprietary systems for reducing software development cost exist. Some are available free of charge, but only under license terms that restrict freedom to reengineer, reuse, and redeploy.

LimeBits fully supports open source and will release its own code. Users will be able to set up their own, independent WebDAV server.

Business model and operations

A paramount goal of the LimeBits project is growing membership and useful community (member-member) interactions.

LimeBits supports (and because of its architecture must support) open source. The goal of growth implies that LimeBits must seek and deserve credibility in the open source community.

At the same time, LimeBits must become self-sustaining and even (because of its corporate sponsorship) profitable.

To accommodate these goals, the following structure is envisioned.

- LimeBits will provide a basic level of service at no charge, with generous but personal-sized storage and personal-speed bandwidth.
- Additional storage and bandwidth will be provided for a competitive, reasonable fee. Additional, premium services may also be provided for a fee.
- Members may seek and offer website/code-related services to one another on the LimeBits site via LimeExchange.com. Members who want to register a domain name may do so through LimeDomains.com. The LimeBits website may include links for these purposes, and LimeBits may receive referral fees from those sites.
- LimeBits may package Basic (free) and Pro (fee) versions of the LimeBits server for personal use.
- LimeBits may also sell enterprise installations, to provide the scalability benefits of the LimeBits platform to large organizations.
- All LimeBits-developed code and content will be open-source under appropriate terms, such as the Apache v2.0 license or the LGPL. Participation in the LimeBits code development team will be open to the public.
- Members will be able to choose the licenses under which they present their code and content, but the LimeBits Terms of Service will specify a default choice, such as Apache v2.0 or LGPL. LimeBits will provide tools to guide authors and users about licensing and compatibility.
- The limebits.com site will be operated by the corporate LimeBits team, observing strict privacy, so users can entrust data and private information to the server. Private account details (e.g., email address) will not be shared without user permission.
- Except for some necessary system functions, code deployed by LimeBits corporate team members will have no extra privileges above that by other LimeBits members.
- Ads and spam from LimeBits are viewed as inappropriate and will be avoided.
- Informative messages and interactions will be permitted. LimeBits may use email for feature updates, newsletters, password reset, and similar directly relevant, desired information. LimeBits will also provide for member-member communications like notifications and comments about apps.

Target market

LimeBits is aimed at several types of user. In general, it is for users who want to create websites easily and with low cost. More specifically, LimeBits targets several subcategories.

- **Experienced web developers.** Web developers using LimeBits need no back-end programming skills to build full-featured sites. They can thus do more, and they can reuse components to serve more customers.
- **Novice web developers.** LimeBits has a simple, unitary programming model: developers can persist data values without having to know the distinction between client and server. Several excellent JavaScript framework libraries are available, simplifying and regularizing the language and DOM access.
- **Web entrepreneurs.** LimeBits will make it easy for web service providers to show their wares and offer services. Developers and designers can attract attention to their work and build communities

using LimeBits. They can link to their service profiles and negotiate engagements on LimeExchange.

Entrepreneurs may use LimeBits not just to post their own services, but also as a hosting platform for any online business (stores, portals, communities, etc.).

- **Designers.** The WebDAV features of LimeBits make it easy for designers to share, reuse, and adapt their creations.
- **Collaborators.** WebDAV features (user groups, overwrite protection, versioning) make collaboration easy – on apps, documents, and designs. For example, members of a business team may have group read-write access to a document they are writing.
- **Social users.** Copyability and modifiability of source code make it easy for LimeBits users to enhance one another's web apps. The universality of browsers and visibility of web apps make incremental, viral one-upmanship easy to attain.
- **JavaScript enthusiasts.** LimeBits lets JavaScript programs store data – vastly increasing their power. JavaScripters can implement new classes of apps.
- **Enterprise.** Enterprise websites may need only few machines using the highly scalable, thin-server architecture of LimeBits, rather than a large server farm.

Potential market size

LimeBits offers three value propositions to users:

- *Context.* LimeBits reduces cost by making code reuse/recycling more attainable and practical than ever before. Code modules are provided not in a catalog, but in the context of already working, testable, modifiable websites.
- *Community.* LimeBits amortizes development costs across a (potentially large) community of users. LimeBits removes obstacles to cooperation: multiple threads of development coexist, yet may share one another's improvements. Enhancements are incremental, cumulative, and cross-fertile.
- *Clarity.* LimeBits reduces costs by obviating some specialists. The thin-server architecture is easier to understand, and requires fewer skills, than the traditional web server model. It can be learned faster and more widely.

These three propositions expand the market. Under an iceberg model, a small improvement in usability vastly increases the number of users. The relatively small population fluent in creating websites may buoy into a massive community.

The first browser to display text and images in the same page was Mosaic, in 1994. With simple HTML, people could easily mix-and-match heterogeneous content. The heterogeneity was popular and led to exponential growth of the early Web.

LimeBits has similar potential, with apps rather than just static content. People can easily mix-and-match heterogeneous apps in the same page. Dynamic websites may grow in popularity, becoming a medium of interaction, as well as information.

Development phases

Pre-alpha. Before public deployment in January 2008, the LimeBits team implemented several key components.

- LimeStone WebDAV server ✓
    - Core WebDAV ✓
    - Access control ✓
    - Quota and size properties ✓
    - BIND protocol ✓
    - Versioning ✓*partial*
    - MKCOL extensions ✓*partial*
    - Mounting ✓
    - Search ✓
- JavaScript security and authentication testing ✓
- Initial website ✓

Alpha. LimeBits is under development and has been deployed at limebits.com for initial testing, feedback, and debugging. Development is part way through the alpha phase. Some functionality is usable. Backward incompatibility may occur.

The goal of LimeBits alpha is to provide a minimal set of features for basic usability, to demonstrate viability of the thin-server architecture, and to sample opportunities for community and revenue growth.

- Authentication ✓
- Seed/demo programs in JavaScript ✓
- Axis JavaScript library ✓
- Single-file versioning ✓
- Usability improvements for the experiences of copying, finding, and creating. Interfaces for navigation, permissions, bind, redirect, and domain mapping
- Content: onramp, licensing, terms of service, mount instructions, team page, contact page, mailing list, newsletter, white paper, tech documentation
- Mixable sites and components, with unified display
- Commenting, community, notification
- Revenue demo sites, linking LimeBits to LimeDomains and LimeExchange
- Security sandboxing
- Server-side JavaScript for validation
- Additional WebDAV features
    - Redirect reference resources ✓
    - Versioning
    - Current Principal extension

Beta. When the basic LimeBits system is feature-complete and stable, the beta phase will begin.

- Scaling preparation
- Security hardening
- Server-side JavaScript for search engine crawlability
- Multifile versioning
- Version autofollowing
- Live Mode / passwordless accounts
- Additional WebDAV features
    - Collection synchronization
    - CalDAV calendaring
    - CardDAV contacts
    - PATCH

Future. LimeBits maintains a long wish list of desirable features, beyond the ones described here. It is hoped and intended that the LimeBits member community will take increasing responsibility for the open-source LimeBits software base.

Open-sourcing plan. LimeBits code is being released under open-source license in the following sequence:

- JavaScript functions
- RubyDAV – compliance test harness for WebDAV
- LimeStone – WebDAV server

Questions and challenges for LimeBits

As a new service and new technology, LimeBits poses several interesting questions and challenges.

- Will *code in context* actually prove to be more reusable than code in catalogs?
- Will front-end developers feel that the LimeBits *thin-server* architecture enables them to build more without back-end support?
- Will *novices* feel more encouraged and empowered to develop and share code?
- Can LimeBits help code *branches stay parallel*, rather than diverge?
- How can improvements or transformations made to one branch be *applied laterally* to another?
- Will parallel branches help *communities stay cooperative*?
- To what extent is it still necessary to abstract *functions* (as opposed to modifying *functionality* in place) in software reuse?
- In a system where *abstraction* and *integration* of software components are less important, how can *selection* and *specialization* [Krueger 1992] of components be enhanced and facilitated? What techniques might members need for finding the closest-matching sites in a large collection?

With the LimeBits service deployed and as experience with LimeBits grows, answers will soon become apparent. Adjustments and corrections will be deployed in response.

References

Benkler Y 2002 "Coase's Penguin." *Yale Law Journal* 112:369-446.

Haefliger S, von Krogh G, Spaeth S 2008 "Code reuse in open source software." *Management Science* 54:180-193.

Krueger CW 1992 "Software reuse." *ACM Computing Surveys* 24:131-183.

Mili H, Mili F, Mili A 1995 "Reusing software: Issues and research directions." *IEEE Transactions on Software Engineering* 21:528-562.

Moen R 1999 "Fear of forking essay." http://linuxmafia.com/faq/Licensing_and_Law/forking.html .

Reindel B 2008 "The code reuse myth." http://blog.reindel.com/2008/07/21/the-code-reuse-myth/ .

Acknowledgements

LimeBits team members (Vipul Bhasin, Angela Chen, Will DeWind, Kevin Faaborg, Rich Henning, Josh Kobrin, Ted Metcalfe, Tim Olsen, Sandro Pasquali, John Politowski, Chetan Reddy, Paritosh Shah, Wil Stuckey) have provided intellectual guidance and leadership, as well as development work. Their contributions and efforts for LimeBits and for this paper are deeply appreciated.

Staff members of Lime Labs, Lime Wire, and Lime Group have supported our efforts, including Kelsey Aahl, Jesse Callaway, Christine Cioffari, Rochelle DiRe, Luck Dookchitra, John Enright, Ben Livian, Jesse Rubenfeld, George Searle, Lucia Smith, EJ Wolborsky.

Mark Gorton's vision, involvement, and funding have been instrumental in driving LimeBits forward.

Jason Pelzer, Catherine Herdlick, Michael Nutt, Marc Herbert, Clayburn Griffin, Jonathan Lister, Dean Brettle, Sumreni Lala, Preethi Sampath, Ankit Shah, and Evan Korth have given us valuable feedback on LimeBits.

Jonathan A. Marshall, LimeBits group lead, is a computer scientist and senior software architect.