

Cornell University CIT-IWS Discussion Paper Series # 2

Practicing Web 2.0

Dirk Swart*, Al Gonzalez, Barbara Friedman, Tonya Miles, David DeMello, Tony Lombardo, Lisa Cameron-Norfleet, Nathan Reimer, Robert Miller, Corey Merrell.

Abstract:

CU Web 2.0 is envisioned as a phased initiative to develop approaches, templates and tools that will allow Cornell University faculty, students and staff to quickly develop websites and to move interactive content online so as to enable collaboration and scholarship, enhance learning and teaching, facilitate recruiting and retention and, most importantly significantly lower the barriers to web content production, presentation and retention. This paper introduces the web 2.0 initiative, proposes some fundamental architecture principles, and looks at tools that could be used to deliver a useful system to campus stakeholders. Ease of use and reducing barriers to entry are emphasized. A system complexity assessment test is proposed.

Challenge:

Deploy a web tool on campus that allows someone to create a good looking web presence in 30 minutes or less.

* Corresponding author: dps42@cornell.edu

Table of Contents

Introduction.....	3
A possible web 2.0 initiative at Cornell University.....	4
Creating a site builder tool.....	8
Digital Repository options.....	10
Considerations and possible next steps.....	11
Appendix I: User Stories.....	12
Appendix I: User Stories.....	13
Appendix II: Web-creation tool complexity assessment.	14
Appendix III: Institutional Repository Notes.....	15

Introduction

The idea that the web is ultimately about harnessing network effects and the collective intelligence of users to build applications is a powerful one. In order for it to be realized, it must first be possible for users to easily and efficiently add information to the web. In the past, we have relied on web professionals to publish content online, but this process is cumbersome and slow, sapping momentum and resulting in content not being put online, or drifting out of date. This problem can be partially solved through the use of traditional content management systems (CMS), which fulfill an essential niche in delivering web content by decentralizing content authoring, while maintaining control of fundamental design decisions and editorial workflow and approvals.

Traditional CMSs work best for large web presences with many stakeholders and constraints upon focused message, identity, consistent editorial style, and centralized design and organizing principles. CMSs tend to be expensive and require end user training, as well as expert resources for implementation and maintenance of complex, proprietary infrastructures.. For small projects with few stakeholders and more localized control, CMSs can be too high a hurdle for campus users.

Many existing web experiences designed around user-generated content, such as Facebook, Flickr, Ning, Pownce, etc. have demonstrated a compelling value proposition through making it easy to upload information that is integrated into the daily fabric of users' lives as they work and play through information radiators, often just by clicking a link or two. We can learn from this approach.

This ability to very easily add to an existing framework and have metadata automatically attached to content by virtue of how it is added has resulted in an explosion of content and a devoted user base.

What is Web 2.0?

In 2005 Tim O'Reilly explained what he meant by web 2.0. He made it clear that there was no hard boundary to the term, and listed some basic characteristics:

- The web as a platform.
- Harnessing collective intelligence and mass publishing.
- Control over hard to reach rich data sources.
- Trusting users as co-developers and the significance of customer self service.
- Lightweight business models and programming.
- Rich user experiences.
- Software written above the level of the device.

In applying each of these principles, the goal is to harness the network effect of users who are empowered to build the web their way.

Cornell's Outreach Portal is an example of a Web 2.0 application.

Web 2.0 is a moving target. One way to see what people think it means now is to look at del.icio.us: <http://tinyurl.com/6ky5yd>

Tim O'Reilly, Design Patterns and Business Models for the Next Generation of Software, 9/30/2005.
<http://tinyurl.com/cr5p9>

CU Web 2.0 is a phased initiative to develop templates and tools that will allow Cornell University faculty, students, and staff to quickly develop websites and interactive content online. These tools could also be shared with K-12 teachers and students as part of CU's outreach efforts. They emphasize abstracting complexity away from the end user and hold ease of use as the paramount design requirement.

Any web content developed with CU Web 2.0 templates would be initially compliant with Cornell's Web Identity and Section 508 accessibility requirements. Users would be made aware of how to maintain compliance to Section 508 accessibility requirements and CU Web Identity Guidelines.

Mandate, brief, and background.

Cornell is committed to scholastic and research excellence. We need to provide our community with tools that deliver an environment that allows them to be effective. This means more than just delivering technology.

This paper investigates three questions:

- How can we create an environment that makes it trivially easy for any Cornellian to put up a website, regardless of their web literacy level?
- How can we help blur the border between producers and consumers of information?
- How can we facilitate the use of existing technologies and services?

As we do this, can we make sure that content contributors are not disenfranchised, and their content is safe over the long term and has appropriate policies and rights management attached to it?

You don't have to be a technologist to understand web 2.0. While technologists are obviously essential—for example to design the information architecture, put the tools in place and operate the network—significant portions of the problem space are sociological and usability related. It is therefore an application of technology issue, not a tool selection issue.

A possible web 2.0 initiative at Cornell University

Many faculty members, classes, clubs, and other organizations frequently want to have a Web presence of their own, but they have insufficient web expertise to build one with traditional tools. It is difficult for them to allocate budget or resources to meet the need, or the process of doing so is more effort than its worth. Their time is extremely limited. Even

Challenge - Restatement:

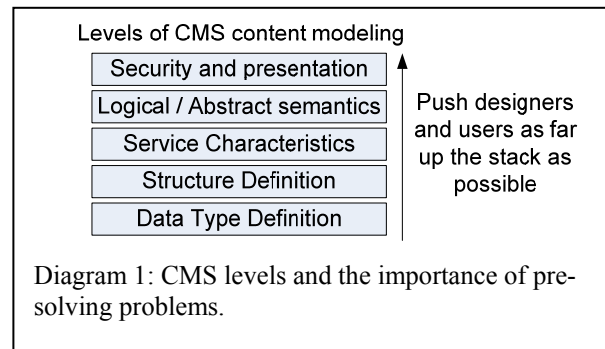
Deploy a web tool on campus that allows someone to upload a wide variety of content types and create a good looking web presence in 30 minutes or less.

This should not require communicating with anybody else and require only the user's netID. Ease of use is the primary design goal. The tool should conform to the principles of a best practice system, but should not expose the complexity of those practices to the first time user.

It should be possible to maintain the site using the same interface with which it was created with and information should be stored in a Cornell University controlled repository. Data should be stored in an institutional repository and be fully searchable and available for repurposing and reuse.

if they are technically capable of putting information on the web, they may not want to shift mindshare from the problem at hand to that of getting information onto the web and research what is essentially a new problem space. We need to provide tools and underlying infrastructure that makes it intellectually easy and technically and ‘safe’ for people to easily put information onto the web.

There are several challenges with this approach. In many cases, the difficulty of putting content on the web does not lie with uploading the actual content (the technical problem), but with the overhead of having to make decisions about information collection, information architecture and site design (the conceptual problem). Unfortunately these problems typically manifest at the same point, that of uploading the content. The result is the technical problem being recognized as a roadblock, but the conceptual problem is not. When using traditional website tools that allow easy uploading, the users often find themselves stuck at or just after content uploading. One of the successes of 2.0 tools like Facebook is that, because they are solving a specific problem, they can provide a predetermined workflow and associated metadata for any uploaded content, removing this decision point while still allowing content to be added intelligently. This pushes the user up the web stack away from design and implementation details toward the content itself, lowering the usage hurdle. So one solution is to constrain the options that a user has and provide a preset framework for each type of content.



Secondly, cost is a factor. Each workflow and content type added increases the overall cost of the system. Because systems like this have high startup costs but near zero marginal costs, the overall audience size is important. Cornell has a limited audience, and it is therefore impractical to spend development effort beyond a certain point. A solution to this is to partner with other institutions to develop for a larger user base.

Finally, the system should integrate with a user’s existing web presences, data sources, and web tools as much as possible, leveraging these tools where appropriate.

While this document is not intended to provide specific technological recommendations, the overall approach is implementation-technology independent. We envision four phases to solving this:

1. Articulate the problem space, audience and value proposition.
2. Create a suite builder tool.
3. Push data repositories closer to creators and consumers.
4. Integrate across other data sources.

1. The problem space, audience, and value proposition

Problem space and audience: The area between that served by traditional CMSs / web content and the ad-hoc CU People-like environment where data is currently stored in a

wide variety of hard to reach and hard to find places. Content that requires structure and context, but that will benefit from being easily reused. This includes bringing data out from computers under people's desks. Appendix 1 lists three hypothetical user stories.

The problem space does not include:

- Publishing of institutional information within the university administration framework like official department websites. This area is well served by traditional CMSs such as CommonSpot. We would like to be able to share data with these websites.
- Official publishing of research: conference proceedings, monographs, etc. There are more formal ways to publish. Generally faculty will work with established publishers and given the choice would rather be in a more prestigious location than an institutional repository. We do include publishing student research, class research and work in progress within the problem space.

The value of providing a tool in this space is significant-much wider acceptance, usability and availability as information that is currently not published or kept in hard to find ephemeral data stores is brought out from 'under the desk', and out of people's inboxes. Ease of use and pre-solving conceptual problems make the act of publishing information to a means to an end rather than an end in itself¹.

"Ease of use dominates everything else. A simple site creation tool raises the level of web competency for all of us."

Tom Hirschl
Professor of Development Sociology

We have identified six essential principles to delivering value in this problem space:

1. Align technologies with practices and use. The necessary technologies are already present and more mature than user expectations, but need to be brought into alignment with users. This requires soft skills such as sociology, psychology, and an understanding of how users work. Once we understand how users work, we can tailor the presentation of existing technologies to fit that set of user expectations and needs.
2. Program to lightweight models that allow for loosely coupled systems. Use simple, widely accepted protocols and approaches rather than sophisticated, proprietary protocols, application stacks and systems. This requires collaborative, open, standards-based strategies from implementation partners, rather than centralized, closely controlled proprietary systems, bound by institutional assumptions.
3. The target is not colleges or departments of the university machine putting up content. It is individual faculty and staff trying to publish and share as a means to accomplish tasks they are faced with, and should allow for individual ownership.

¹ This is not always true (for example publishing to ArXiv could be considered an end in itself, but we believe it will be sufficiently true for this problem space.

4. Conceptually decouple adding content to the web from consuming that content. They are fundamentally different tasks. Of course, within the UI a site creation tool may incorporate these into one step, as when a user clicks a link to add data to a page, but in the background the platform needs to keep track of the content added, and provide alternative means to access it. A corollary of this is that it is equally important to design for the persistent storage of information as a separate function.
5. Design for large collections of simple tools to work together, rather than a single sophisticated tool. Facebook has executed this well—what you are presented with is an agglomeration of individual tools.
6. Increase responsiveness to create a rich user experience. Facilitate on-the-fly feedback and interactivity, offering visual cues to users that allow them to more accurately guess at functionality.

2. Create a suite builder tool.

A tool that makes it easy to create and maintain a website is needed. We need to execute our vision by purchasing a solution, modifying an existing solution or developing our own solution using off the shelf tools. This will be a major deliverable and form the core of Cornell's solution in this problem space. See the next section for details. The first step will be to create a minimally usable subset.

3. Push data repositories closer to creators and consumers.

Content creators use a wide variety of tools to store content on the web. Many of these are existing web tools which are integrated into their work practices, such as Zotero, del.icio.us and Twitter. This content is conceptually 'close to the user': easily accessible and changeable, but also distributed and shared. It is a significant store of value. Other tools like Open Calais are focusing on interoperability and solving metadata problems.

There are two types of functionality we need to provide:

First, mechanisms to integrate with and publish content that users already have on the web, allowing them to harness collective intelligence and mass publishing through a site builder tool.

Second, members of the Cornell community should be able to easily share data they have created using such a tool with others in a secure, rights-controlled way.

This is central to trusting users as co-developers and providing self service ways to innovate. In doing so, we give up some control over this data, but our choices are to include it (and accept less control) or miss out.

4. Integrate across other data sources.

The Ithaca College Experience

This problem is not unique to Cornell. Ithaca College recognized a similar, related problem. They developed an in house tool which allows users to easily create websites. A key lesson we can learn is not to tackle the entire problem at once.

Although there are problems, for example with code maintainability, this tool has been very successful and has matured well.

It is no longer realistic to expect users to faithfully upload all their content into the next new system. Cornell already has significant web-based data repositories, Vivo and the library eCommons for example, which store significant amounts of data. We need to harvest from and share with these systems.

Part of the current focus on machine-readable forms of information (the semantic web) allows us to interconnect data and services in a meaningful way. Institutional repositories already deliver functionality in this space, but it is not at a maturity level that facilitates easy end user use. The final phase of a successful system will be to facilitate sharing with other institutional systems

Creating a site builder tool

There are two key technology areas in the solution, a site builder tool and a repository to store data. The repository needs to be separate from the site builder tool. While the site builder focuses on ease of use in a rapidly changing web landscape, the repository needs to focus on more pedestrian issues of robustness, durability, scalability etc. We want to think about these complicated questions so our users don't have to. They should not need to be aware of the complexity, but should be able to tap into it when needed.

Many tools have done an incredible job with front-end ease of use: Wordpress, Confluence, Facebook, Webnode, and others. These tools make it easy to add information to the web. None of them concentrate on being an information repository, and many address only a limited problem space—blogs, microsites, etc.—and findability is often a problem. Perhaps the most comprehensive and best executed solution is Facebook. It includes a sophisticated permissions system, handles many types of data, has rich metadata, and has cleared the ease of use hurdle.

There is a need at Cornell University for a tool that makes it easy to create websites. A significant amount of work has already been done to help web users on campus—for example the identity templates, assistance with Section 508 compliance, and a host of IT and university communications services. It is important to leverage these assets in delivering a site builder tool, which should also adhere to the following requirements:

Ease of use requirements:

- 1) No training required for users such as those described in the user stories (Appendix I).
- 2) Should not require set-up work by people other than the end user creating the site.

Using a site creation tool is like seeing a special effect in a movie. The minute a person watching a movie stops and thinks 'what a cool special effect' you have lost something. They should be thinking 'what is the hero going to do next?'

In the same way, when a person is creating a website they should be focused on the problem they are trying to solve – the reason they want a website – and not be thinking about the tools and technical complexities of what is after all a very complex exercise. It is our job as technologists to solve these problems for them.

Feature requirements:

- 1) The user needs to have control over look and feel, preferably by choosing amongst provided templates, all of which are acceptable and adhere to basic design principles.
- 2) Layout needs to be modifiable with drag-and-drop and not require coding.
- 3) The toolset should allow inclusion of content from multiple sources, not just its own repository, ideally including places where people are already storing their content.
- 4) The toolset should be flexible enough to add new components and content sources over time.
- 5) The toolset needs to allow rich content (video, audio, images, possibly applications including chat, blogs, forums, etc.) as droppable components.
- 6) It should be possible for external developers from within Cornell to add components.
- 7) Output should be Section 508 compliant
- 8) Output needs to be Cornell branded
- 9) The application code must be maintainable.

There are a number of tools that deliver a significant portion of this functionality and can be expanded to deliver a site builder. Confluence can be seen as a prototype of such a tool, provided effort could be devoted to substantially increasing ease of use, and the paradigm could be expanded beyond that of wiki and document management.

Depending on budget and time constraints, we could also choose a more modular based on emerging open standards like OpenSocial and, creating a framework and set of cooperative components would be a more robust and scalable solution. This is the approach taken by Facebook, and has allowed for developers to independently create a very wide set of tools. Of course, it has an additional overhead of creating the interoperable framework. It also has the risk of failure: the same approach is taken by uPortal which, although it offers a uniform delivery infrastructure for individual channels of content, does not offer compelling metaphors, paradigms, and behaviors that foster creative ease of use and connections between content channels.

Hosting requirements:

- 1) Must be hosted at Cornell and allow inclusion of content hosted at Cornell.
- 2) Needs a permissions structure to manage ownership and visibility.
- 3) Information must be sharable. It should be possible to access information via web services.
- 4) The data should be searchable and enduring.

Institutional repositories deliver all these and more.

“A university-based institutional repository is a set of services that a university offers to the members of its community for the management and dissemination of digital materials created by the institution and its community members. It is most essentially an organizational commitment to the stewardship of these digital materials, including long-term preservation where appropriate, as well as organization and access or distribution.”

Lynch, C.A. Institutional Repositories: Essential Infrastructure for Scholarship in the Digital Age, 2003.

Digital Repository options

People create digital assets, which are used as components in web pages. It is the role of the digital repository to store these assets in a way that allows them to be reorganizable, discoverable, reusable and preserved over the long term. They deliver this functionality in a way that allows maximum flexibility to change design and architecture decisions later.

There are three ways of thinking about a digital repository:

- Repository as an architecture. The conceptual framework of how you organize and manage digital objects.
- Repository as an application. What kind of software you choose to implement this architecture, and the tradeoffs implicit in this choice.
- Repository as a service. What kind of services do you provide? How do you gather information, and how are outputs shared and harvested? How copyright clearance is handled, what you choose to expose to search engines, how you protect patentable information, etc.

Cornell University is actively involved in the forefront of research into digital repositories and also maintains a number of repositories. The largest of these, the ArXiv preprint server, has approximately 450,000 articles. Like ArXiv, most of these repositories are aimed at a well defined problem space and perform well in this space, but are not designed to address our problem, a more general, accessible institutional repository.

Two institutional level repositories were evaluated, Dspace and Fedora. Both are instantiated within Cornell, and Fedora is supported by CIT and available for anyone to use. Appendix III tabulates some important features.

There are many differences, but most notably DSpace offers a single integrated solution focused on preservation, while Fedora decouples each aspect of the solution through a more modular approach, and handles a wider problem space. Each of these approaches offer compelling advantages.

Finally, our goal here is to integrate and share data across campus with other systems, not to integrate our system with other systems.

Fedora

Fedora is an open source repository produced by the Fedora-Commons, a Cornell University sponsored spin off.

It is designed to be extended through adding tools, which makes it particularly flexible, and it is particularly suited for the kind of storage needs a solution would have. However, its modular approach raises the bar to implementing a full solution. For example, it is bedeviled by not having an obvious, easy to use front end, a problem DSpace has overcome well. The site builder tool would solve this problem. It would not be necessary for an end user to interact directly with Fedora.

"You are able to design for continuous change.

Your digital assets stay safe while Fedora allows you to move gracefully into the future as you build new systems."

- Dan Davis, Fedora-Commons, on the benefits of using Fedora.

Additional reading:

Fedora-Commons: www.fedora-commons.org contains information and numerous references to other resources.

DSpace

Developed by MIT and HP Labs, DSpace is a sophisticated, mature product, and is more widely used than Fedora. It is aimed at digital collections (libraries) and is optimized for the size, usage style and request style of digital libraries with the associated focus on preservation. It integrates all aspects of the problem space into one end-to-end solution. This makes it conceptually relatively easy to understand and support.

The current disadvantage of DSpace is that it makes many assumptions about the type of content being preserved in order to optimize functionality and efficiency. It closely couples the items in the repository with their location in an arbitrary organizational taxonomy or categorization scheme. Fedora can be configured easily to support a loosely coupled metadata layer that allows the same content objects to exist in multiple contexts and categorizations.

Additional reading:

DSpace: www.dspace.org

Eternal Bits, IEEE Spectrum Online July 2005. <http://www.spectrum.ieee.org/jul05/1568>

Considerations and possible next steps

The goal of this document has been to begin a discussion on the principles and practice of a user centric-web capability. This included looking at the needs of campus and what constitutes a best practice system. Six basic principles of system design were proposed.

Important focus points were:

- The components of a successful system are here, and recognizable. It is not necessary to create new tools from scratch. We have design principles we can use to evaluate alternatives, and choose the best from what is available. Once that is done, we need to apply resources to making these components work together.
- Ease of use. This was the overriding message we heard from faculty and staff. Any system should be usable (in a limited way) with no training.
- A set of loosely coupled components will need to connect into existing university system and software. Therefore, we should build our current systems with interoperability in mind.

We believe that a carefully crafted system, building existing components into a modular system, can significantly improve productivity at Cornell. No solution exists to do this right now, but there are many promising components. This gives us four choices: adapting existing solutions within Cornell, cooperating with other universities, building our own system from existing components, or leaving the problem for the future. As technologists, it is our role to sort through the technical complexity and present finished work that is simple without being simplistic.

This document is the first part of the four step process we outlined. As we work on those steps, we should:

- Focus on a minimally marketable subset of features within and work to deliver that quickly and efficiently. Do not attack the entire problem space at once.
- Look at creating a set of templates on an existing tool such as Confluence and focusing on increasing the ease of use.
- As technologists, we need to ensure that data is stored in institutional repositories, while abstracting the complexity this entails.

There is a lot of talk about web 3.0. For most people this is just more of web 2.0: focusing on extended convergence, the internet of things, multiplatform (especially mobile) delivery and simpler syndication, for example. If it will include something very different is an interesting topic, but beyond the scope of this paper other than to note that in the last 10 years we have seen very little innovation in the browser client. Chrome and IE8 seem to be trying to change this by delivering solutions that can do real work, not just browse the web.

The authors would like to thank the many people who helped us make this research a reality: Clare van Den Blink and the staff of CIT; Tom Hirschl; Cornell Libraries; the Fedora and Dspace teams, especially Rich Marisa, Sandy Payette and Dan Davis; Diane Kubarek and Tommy Bruce of University Communications.

Appendix I: User Stories.

Below is a brief description of specific problems faced by three hypothetical users. They have been created as archetypal models of situations which fall within the problem space.

User 1: Faculty member Josh Perry.

Josh Perry is an assistant professor who spends a great deal of time working on his research. He is not afraid of technology and is aware of its possibilities, but doesn't have time to be an expert. In addition to having his bio on the department web site, he wants his own site, which would contain his CV, information about his research, links to sites related to his research, and an area where he can share his research with people who are interested in the subject matter but are generally not going to be reading the current journals. He also needs to put information on the web to support his classes, and often wishes he had an easy way to do that. He has specific ideas about the ideal layout and color on the site he would create.

User 2: Staff member Andrew Tucker.

Andrew Tucker is a staff member in the Performing Arts department who creates web sites for different clients within his academic department. He produces his own media, including images and video, which he is willing to share with others. He is comfortable with technology, and has up-to-date equipment, but he does not have a great deal of time for producing individualized websites.

User 3: Staff member Judy Snyder

Judy Snyder is a secretary for an academic department in the Arts College. She knows basic Office applications, but has no web skills, and her unit has no resources to pay anyone to work on a web site and cannot afford CommonSpot. She has desktop support on call but no other technical support is available. People within her department depend on her to maintain the department web site, which is an old site that needs an updated look. She has some opinions on how a new site would look and wants flexibility in layout. She needs to include biographies and events, plus videos and images in the new site. She is consistently working on a deadline.

Appendix II: Web-creation tool complexity assessment.

In addition to usability, we should consider the power and flexibility of a system as an important secondary goal. The questions below are intended to be a first order approximation of what should be considered when assessing how complex a system is, represented by a score out of 13. The goal of this assessment is to better understand what users consider complex, focus attention on adding maximum utility and minimum complexity, and explicitly recognize that utility is linked to both complexity and ease of use. This is not intended to be a complete or rigorous evaluation, and clearly further investigation is needed.

A regular user is someone who is not able to write any code, who uses off the shelf tools, dialogs, and wizards, and who has not modified their computer in any way. Any solution must be designed for use by a regular user. Usage can be evaluated according to existing tools, such as the System Usability Scale (SUS).

Web tool complexity assessment.

- Questions are assessed on a Likert scale.

Is it accessible through a regular, human readable URL?

Is it possible for a regular user to do the following without help:

1. Add content to a page.
2. Create a new page.
3. Add a new element: blog entry, form, photo gallery, etc.
4. Add new content to an existing element.
5. Place or rearrange elements on a page.
6. Link to a data source.

When conducting a search, is it clear which content is being searched?

Is it possible to search across all content and present results in a contextually meaningful way?

Is the user able to select his/her own look and feel when adding a page?

Does the system constrain design changes to an acceptable subset?

Does this look and feel comply with Cornell's identity and section 508 requirements

Is content accessible agnostically from the product and context in which it is presented.

Appendix III: Institutional Repository Notes.

The table below is a *subjective* evaluation of Fedora and DSpace. Source data is taken from the web and interviews with system experts.

Feature	Fedora	DSpace
What types of objects is the repository best at holding?	Fedora is extremely flexible and can be instantiated to hold just about any type of objects and support any discovery method / access level.	DSpace is designed to hold scholarly knowledge. That is, papers, articles, etc. It is not particularly good at holding snippets and frequently accessed items, like web page content.
Does the repository simply hold the objects, or does it provide for editing and manipulation of objects?	It holds objects and extensive manipulation is provided for.	It holds objects. Not much manipulation is provided for.
How are the objects organized in the repository?	Objects are composed of data streams. These can be fragments of XML inline, or they can be files stored on the server, or they can be external / on another server and can be (a) delivered through a URL or (b) Fedora can proxy the content on another server so it looks like it is local even though its not. Finally, the content can be dynamically generated or fetched from a web service.	DSpace provides a classification scheme.
Can objects be classified in multiple ways?	Yes.	Not really.
Are subject-level classifications separate from administrative and security classifications?	Yes.	Yes.
Can external metadata managed in other systems be applied to the objects in the repository?	Since metadata is just a stream, you can do this either explicitly or undetectably.	This is in the works, but right now, no.
Is security on the controlled vocabularies distributed?	Security is a big issue. There are common policies to manage stewardship of the system, and it is distributed.	Security is a big issue. There are common policies to manage stewardship of the system, and it is distributed.
Does the repository have a well-developed client interface for administering security, classifications systems, users and groups, and objects themselves?	Not really. There are a number of 3rd party interfaces. Fedora does not include this in the core.	Yes.
Does the repository have a well-developed API for interacting with its content from other information systems? Can the repository, through its API, be used as a persistent storage layer for other applications?	Yes.	This is an area of active research. Right now, not really.
Do objects have unique and persistent identifiers that are reliable for archival purposes?	Yes.	Yes.
Can the repository use pluggable security enabling single sign-on with Cornell NetID or other authentication systems or integration with external authorization management systems such as grouper?	Yes. (Single sign on is not part of the Fedora core.)	Yes.
Is the repository available as a product that can be hosted at Cornell, or is it a proprietary off-site service with which we contract?	Yes. It is available for use and supported by CIT.	Yes. We would need to create an instance of it.

