UNC
SCHOOL OF MEDICINE

UNC - School of Medicine Curriculum Management

Charlie Hitlin, Manager of Web Application Development

Walt Martin, Plone Developer and Systems Analyst

UNC Chapel School of Medicine

Kapil Thangavelu, Plone Developer

Cignex Technologies

Overview

- UNC Chapel Hill School of Medicine Background
- Existing curriculum system
- Problems and motivation for change
- Why Plone
- Why Cignex
- Implementation
- Post Mortem
- Demo and Q&A

UNC SCHOOL OF MEDICINE

School of Medicine - Background

- Established in 1879, the School of Medicine is a top-ranked public medical school and oldest in NC
- Medical student body: 649
- 4 year undergraduate medical program that leads to a MD
- 2,972 applications were received for the 160 places in the Class of 2008
- Full-time faculty: 1238
- Approximate 1/3 of faculty are very active in the MD curriculum
- Ranks 17th of the NIH funding list
- Largest single source of NC physicians

UNC SCHOOL OF MEDICINE

Existing Curriculum System

- Users include medical students, faculty and administrators
- The first two years of the curriculum are entirely online
- All medical students have a required laptop to access online curriculum materials
- Key elements presented online are the course schedule and course resources
- Resources can be owned by multiple types of users
- Resources can be documents (.doc, .pdf, .txt), web resources (urls) and multimedia files (.mpg, .mov, .rm)

Existing Curriculum System

- 2000 plus statically linked web pages
- Most page design accomplished through Dreamweaver
- Pages and resources added on request
- New or updated resources are uploaded manually and linked in by web designers
- Some resources (e.g. quiz answer sheets) have a finite life span and must be removed from the site manually

Problems and Motivation for Change

- Inefficient Content Management:
  - Multiple tools used to create content
  - Manual management of broken links and integrity of information was error prone and time consuming
  - Content delivery was varied and lacked workflow
- Inadequate Infrastructure:
  - Ever growing number of digital learning resources had become difficult due to inefficiencies in the process
  - Resource delivery took anywhere from a few hours to days
  - Delivery of resources was by physical transport due to email attachment restrictions

Problems and Motivation for Change

- Dissatisfied User Base:
  - » Increasing push to have an on demand publishing system to speed up content delivery
  - » Expectation that learning materials be immediately available for each course
  - » Demands that content always be the most up to date version available
- High Maintenance Costs:
  - » Consistent look and feel to the existing site was maintained for all courses by manual processes
  - » Mixed content and presentation limited the re-use of resources and introduced a good deal of duplication
  - » High human and system costs resulted from the manual maintenance of this static environment

Why Plone

OR - The process that lead to Plone

- School of Medicine CMS committee formed to conduct needs analysis
  - Make up of team
  - Discovery process
  - Initial needs analysis
- Internal requirements for the system included:
  - Integration of existing LDAP
  - Integration with Oracle DB
  - Ability to run on current UNIX environment - Sun Solaris
  - Highly scalable application framework
  - Cost effective
- Looked at several proprietary and open source solutions:
  - Proprietary: Vignette, Oracle Portal - *Expensive*
  - Open Source: Various PHP solutions, Bricolage, Mason, OpenCMS - *Cost effective, but none exactly right*

UNC SCHOOL OF MEDICINE

Why Plone

- Fit all system needs
  - LDAP
  - Oracle integration
  - Runs on existing UNIX environment
- Strong security and workflow model
- Open source, so the price was right (e.g. cost effective)
- Use what you need rather than buy unused features
- Don't pay for software licenses, pay for development to meet specific business model

Why Plone

- Easy to get started
- Have a site up and running in 5 minutes - Great for proof of concept
- Large and active user community group
- Completely customizable look and feel
- Other local knowledge bases on campus

Why Cignex

- Introduced to Cignex via Zope and Plone developer training
- Cignex presented a well defined process
  - Concept development
  - Planning and analysis
  - Solutions design
  - Development
  - Integration and deployment
- Cignex project management and documentation framework
  - Requirements document
  - Design document
  - Issue Tracker
  - Weekly status reports
- Forced us to closely examine our entire process and business logic
- Because Cignex was a paid consultant, they got answers and decisions were made

Implementation

- Resource Manager:
  - Provides multiple resource types… file, link and references
  - Asset library for resource reuse
  - Tracks revisions of resources
  - File content stored on local file system, not the ZODB and indexed for search, with support for Powerpoint, Word, Excel, and PDF formats.
  - Builds on ATManagedFile, ATVocabularyManager, PortalTransforms, and ATReferenceBrowser.

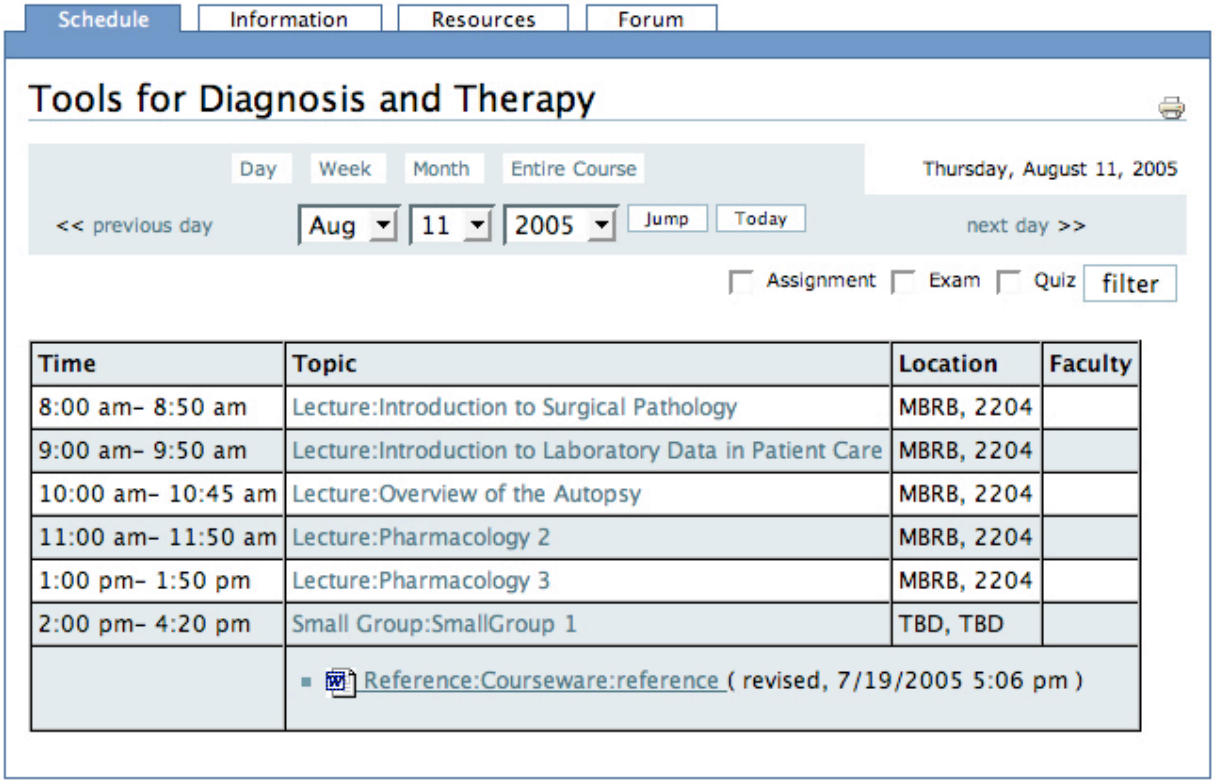
Implementation

- CurriculumManager
  - The Core of the system presents multiple calendar views displaying courses and topics
    - Topics are extended system events
  - Courses and topics support delegation to users with appropriate roles for management.
  - LDAP Backed User Search Widget
  - Basic iCalendar Integration.
  - Builds on Calendaring, ATVocabularyManager, and python-icalendar projects.

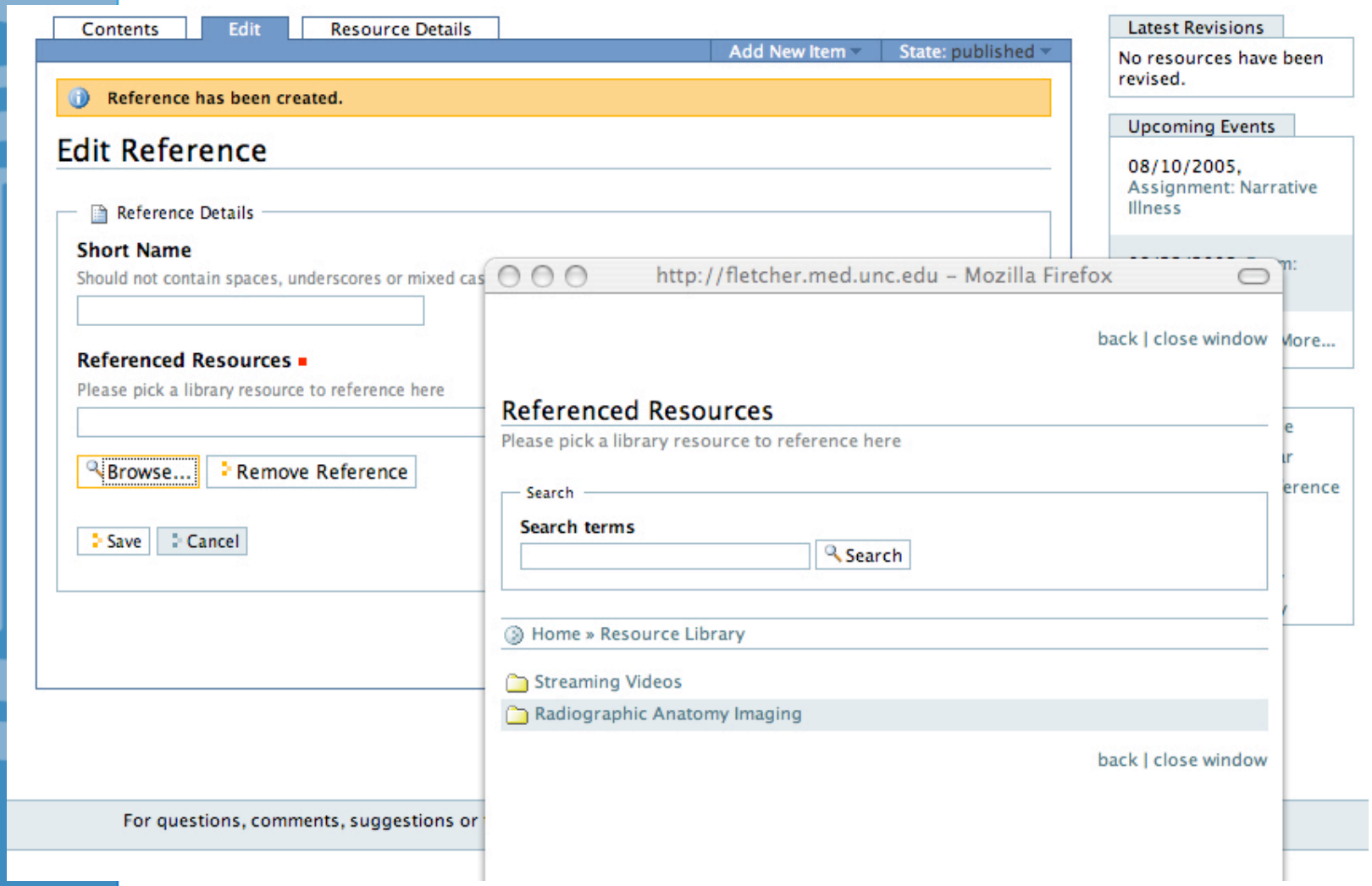
Implementation

- Development Lessons Learned:
  - Good reuse is crucial - focus on the application and not the plumbing
  - ATVocabularyManager - great for admin managed vocabularies, ui needs work though.
  - ATManagedFile - lots of functionality, but the code looks bad for future extensions. The blob project at plope.com, looks promising.
  - CalendarX versus Calendaring
    - Pretty UI versus Extensibility
    - W/ Calendaring, every custom calendar view reduced to five lines of code.
  - Catalog security filtering inadequate out of the box when containers are private

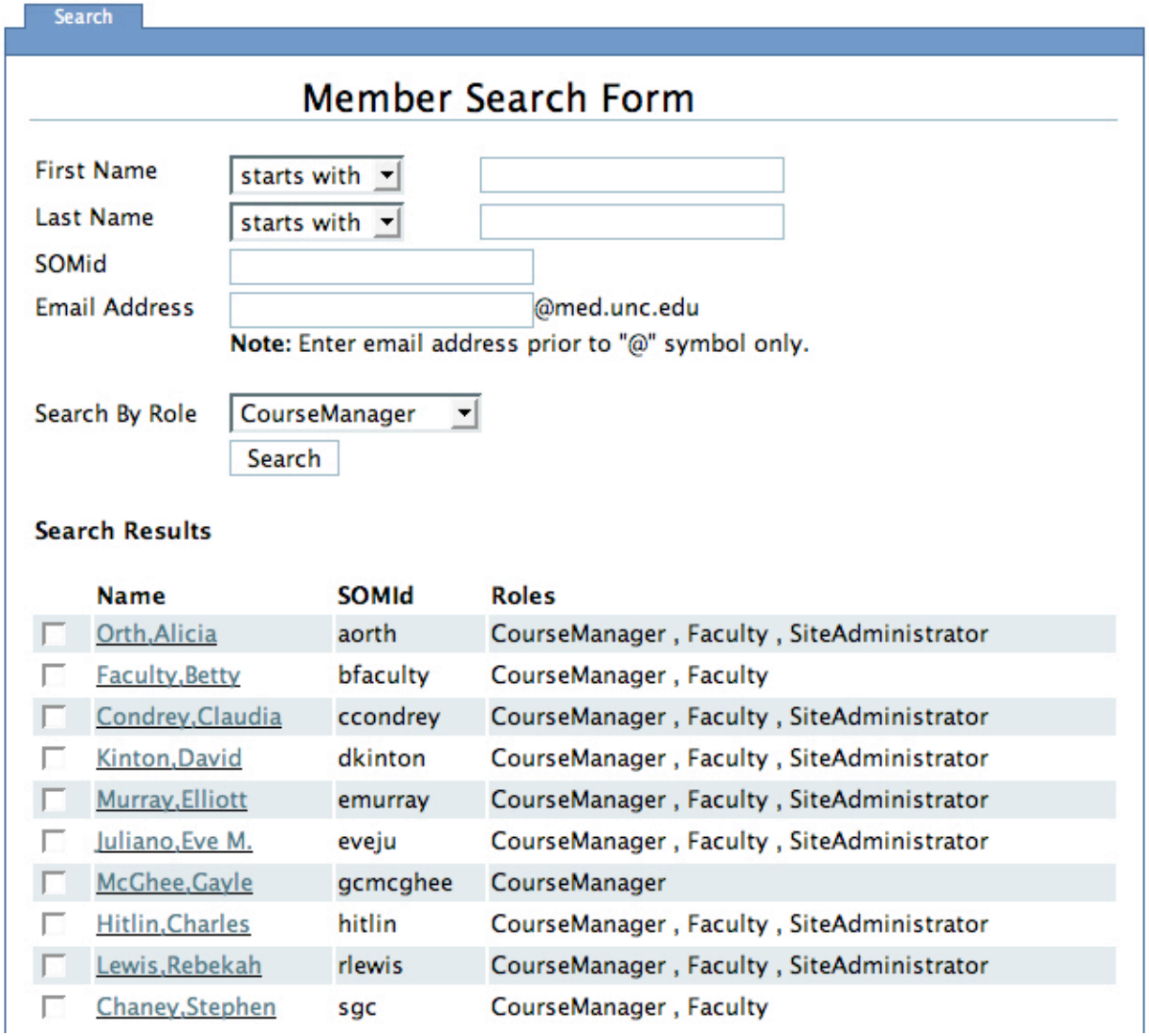
Topic Day View



Referenced Resources

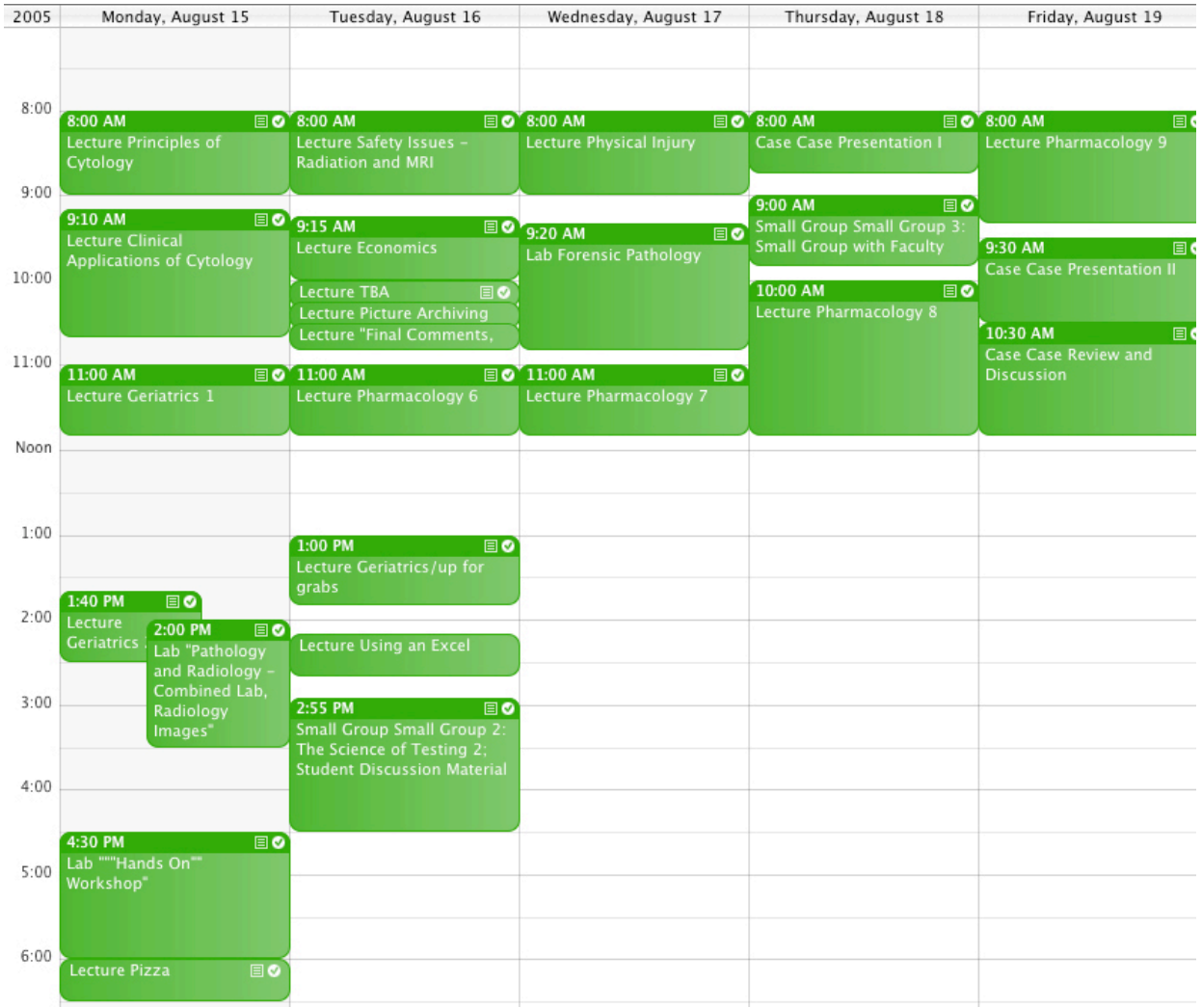


Manage Members



UNC SCHOOL OF MEDICINE

iCalendar Integration



UNC SCHOOL OF MEDICINE

Post Mortem - Lessons Learned

- Initial User Interaction
  - Faculty and administrative response to demonstrations of the beta software was very favorable
- Looking Backwards:
  - We did not know our own business process as well as we thought
  - The problem we needed to solve was discovered to be more complex then originally believed
  - We were not able to complete everything we initially wanted to do
  - Difficult issues or decisions were sometimes tabled for future discussion
  - Initial needs analysis had a tendency to focus on feature lists and screen designs rather than business process

UNC SCHOOL OF MEDICINE

Post Mortem -

The Good, The Bad, The Ugly

The Good, The Bad, The Ugly; A Retrospective

- The Good:
  - Initial user reaction was extremely favorable
  - Plone (and Zope/Python) have proven to be flexible and rich tools
  - The development process proceeded smoothly and mostly on schedule
  - The delivered system meets and exceeds specifications
  - Costs were almost exactly inline with quoted time and effort estimates months earlier
  - The analysis and design of this system has given focus to other projects
  - The end user response has been excellent
  - Cignex proved to be an invaluable partner and delivered a great application

The Good, The Bad, The Ugly; A Retrospective

- The Bad:
  - The project was severely under specified. The requirements document could have been expanded by 100%
  - Use Cases were never done
    - These should have been done early in the process with user's input
    - At one point *post hoc* use cases were done on screen mock-ups to resolve process issues
  - Communication between internal staff broke down at times
  - Mindset switch when transitioning from static web pages to designing dynamic applications built on business process is key - This is still occurring

The Good, The Bad, The Ugly; A Retrospective

- The Bad Continued:
  - Without the upfront training for web developers in Plone or Object Oriented design, misconceptions existed
  - Hardware requirements should have been investigated and tested early in the process
  - Performance goals and systems level software configurations should have been fully mapped out in the design phase

The Good, The Bad, The Ugly; A Retrospective

- The Ugly:
  - Some core Plone UI semantics were mixed in order to satisfy user needs
  - Some requirements proved unworkable upon use of live system resulting in 11th hour fixes
  - Regression of the final beta ran right into production time lines - one month over goal
- BUT……

Plone Works!

- “The ‘Bad’ and ‘Ugly’ are learning experiences for a green staff. Despite our deficiencies Plone provided the flexibility and power we needed to produce a content management system that met our requirements in a reasonable amount of time and effort.”
  - Walt Martin - UNC Chapel Hill School of Medicine
  - Plone Developer and Systems Analyst

UNC SCHOOL OF MEDICINE

Demo and Questions