Gradebook: Developing a Sakai Tool

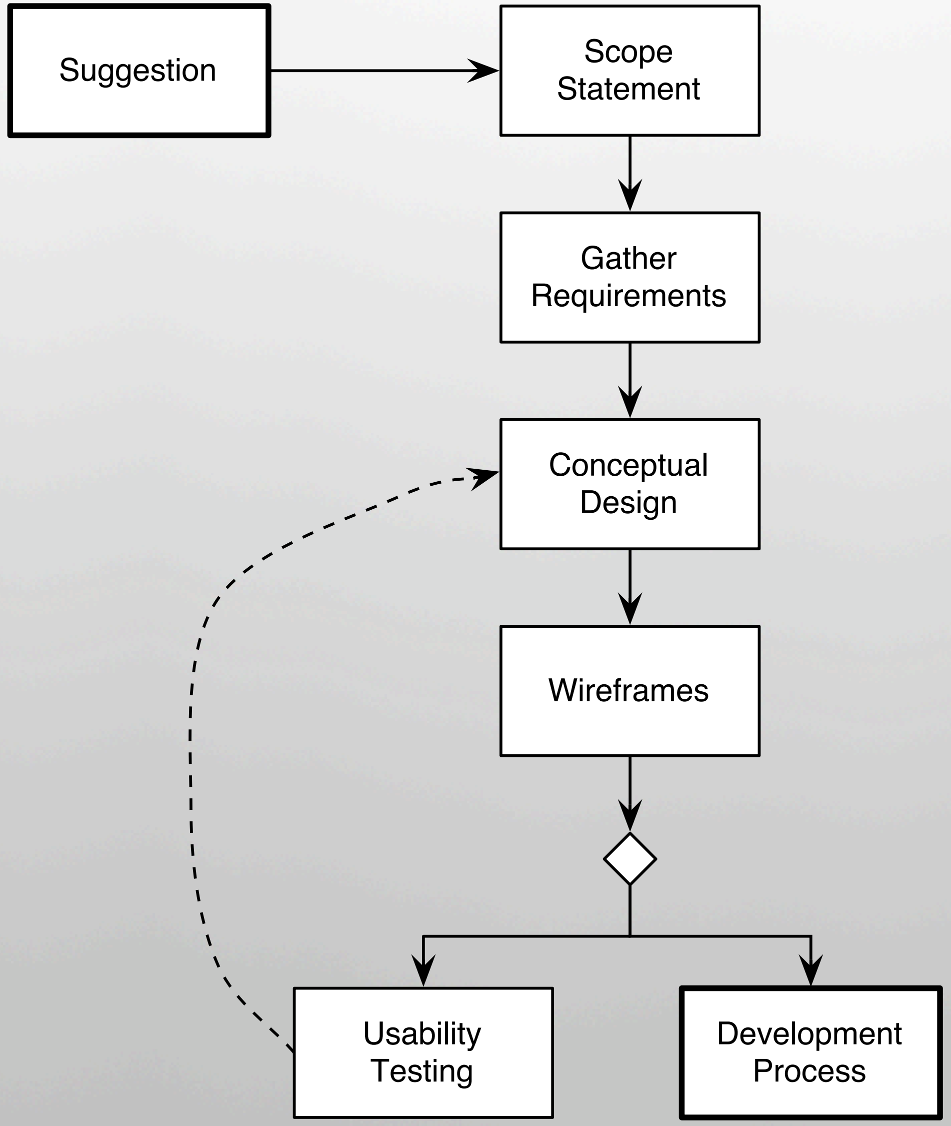
Ben Brophy
Massachusetts Institute of Technology

How We Build a Sakai Tool

Design	From suggestion to wireframes	3/04 - 5/04
Development	From wireframes to functional version	Underway
Launch	From functional version to release	Pilot Fall ‘04

Design

Design Process



Starting the Process

- There is a process for gathering suggestions coming - each institution probably has their own.
- MIT Faculty had been asking for a course-wide gradebook in Stellar.
- Samigo will need a gradebook to plug into.
- Most importantly, we need to figure out how to build a tool using the Sakai framework.

Our Methods

- User centered design, bringing users into the process early
- The application is programmed from the UI, not the other way around
- This approach stems from our experience designing tools for Stellar, MIT's current Course Management System

Requirements

- Gathered from MIT faculty and TAs, including 24 one on one interviews
- Instructors from all the schools at MIT, and some from other institutions
- We got help from students in the Bentley Human Factors and Information Design program
- Got samples of what the instructors used for grading now (usually Excel spreadsheets)

Sample Requirements

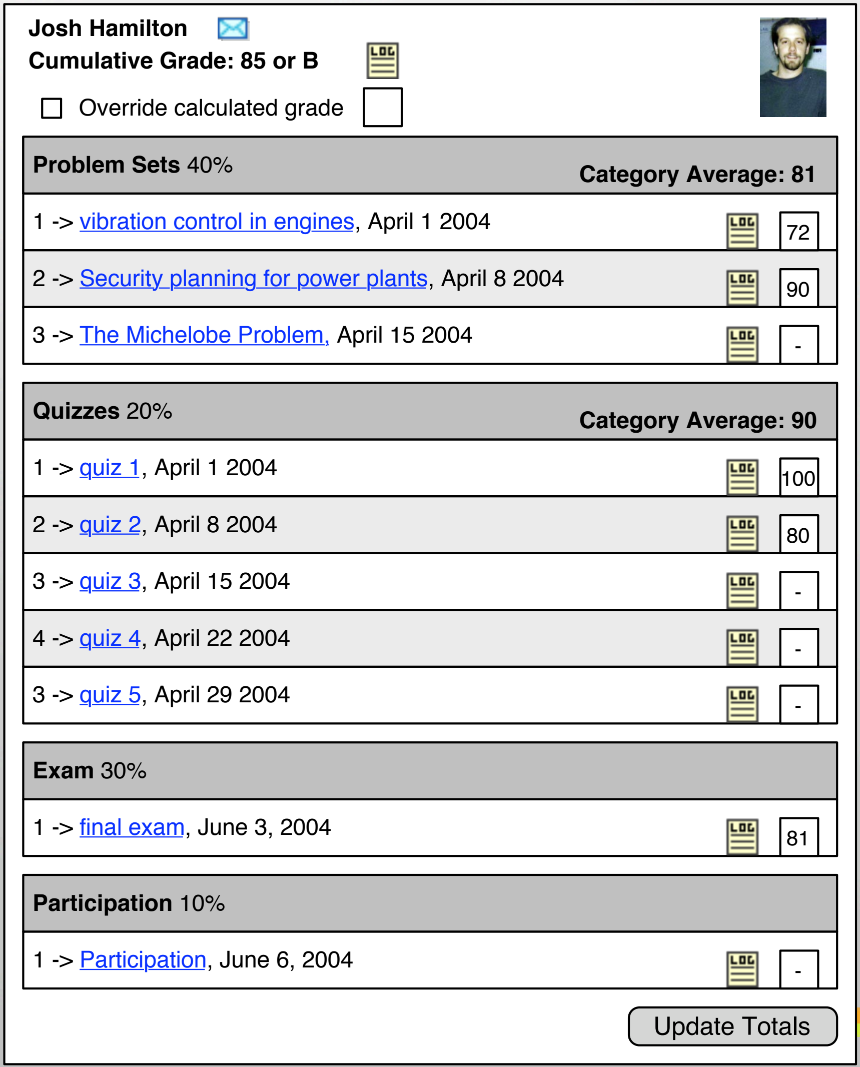
- Submit grades to registrar button
- Instructor can override calculated grades
- Adjustable grade scales
- Instructor can set student viewing options
- Show statistics for each assignment
- Ability to automatically drop worst grades

Existing Applications

- Similar functionality in Stellar's Homework tool, and similar security concerns
- Gathered grade related functionality gaps, principally from Indiana's Course Management System
- Reviewed commercial applications
- Access to the requirements gathered by UC Berkeley

Conceptual Design

- Based on the requirements
- Start with the overview pages
- Build a flow chart at the same time
- Build every page
- Iterative process



Josh Hamilton

Cumulative Grade: 85 or B

□ Override calculated grade □

Problem Sets 40%	**Category Average: 81**
1 -> vibration control in engines, April 1 2004	72
2 -> Security planning for power plants, April 8 2004	90
3 -> The Michelobe Problem, April 15 2004	-

Quizzes 20%	**Category Average: 90**
1 -> quiz 1, April 1 2004	100
2 -> quiz 2, April 8 2004	80
3 -> quiz 3, April 15 2004	-
4 -> quiz 4, April 22 2004	-
3 -> quiz 5, April 29 2004	-

Exam 30%	
1 -> final exam, June 3, 2004	81

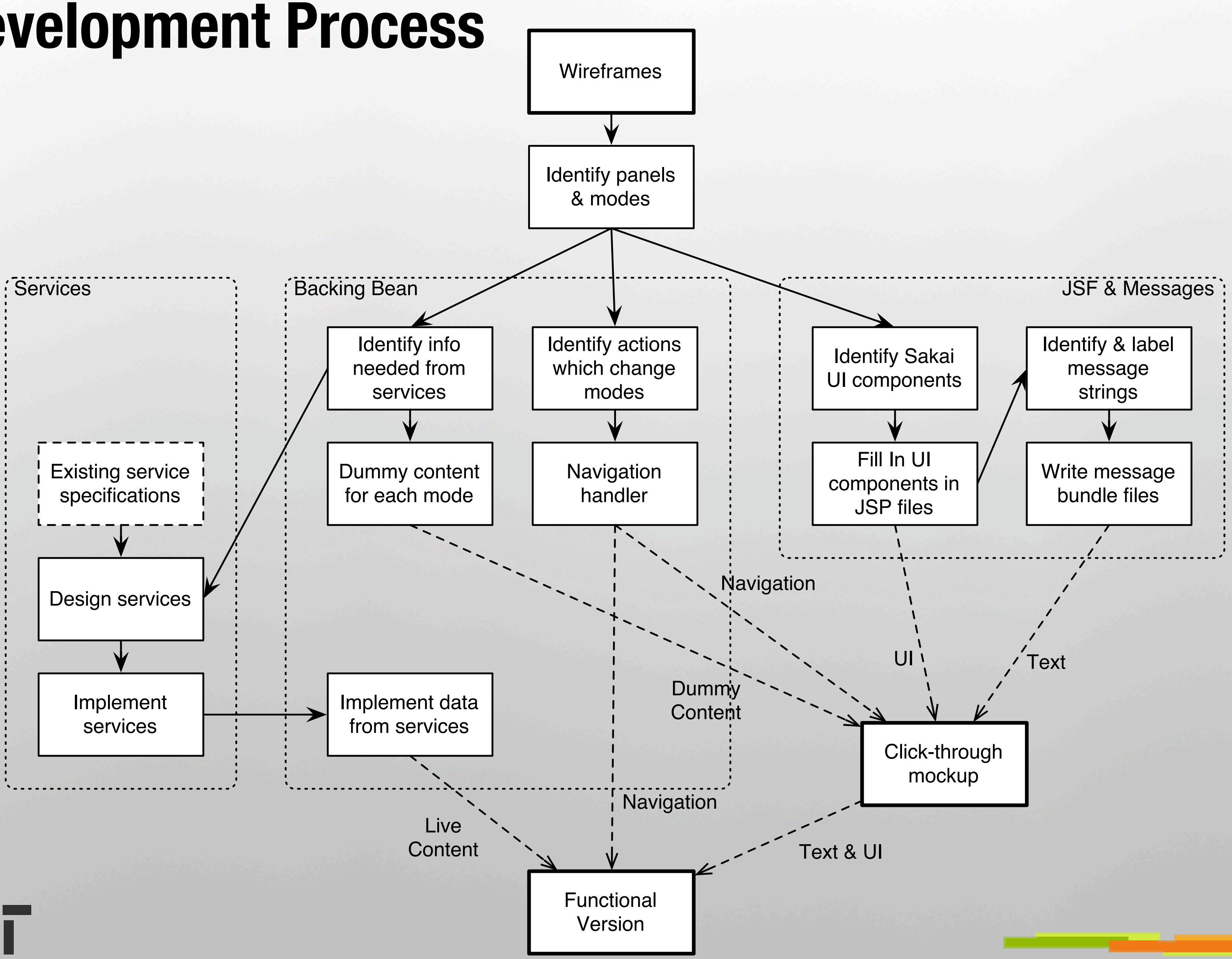
Participation 10%	
1 -> Participation, June 6, 2004	-

Usability Testing

- Used the wireframes as paper prototypes
- First round of testing a heuristic review by the usability team
- Second round with instructors who contributed to the requirements
- Revised wireframes after each test

Development

Development Process


Wireframes
Identify panels & modes
Services
Backing Bean
JSF & Messages
Identify info needed from services
Identify actions which change modes
Identify Sakai UI components
Identify & label message strings
Existing service specifications
Dummy content for each mode
Navigation handler
Fill In UI components in JSP files
Write message bundle files
Design services
Navigation
UI
Text
Implement services
Implement data from services
Dummy Content
Click-through mockup
Navigation
Live Content
Text & UI
Functional Version

Using JSF Tags

- A JSF tag produces an element of the user interface
- Example: <sakai:button_bar> makes a row of buttons
- Very simple for any one used to editing XML
- A minimal Sakai tag set is available now
- Developers may be able to build custom tags
- The O'Reilly book was a big help

Using Message Bundles

- Start with a particular string in the JSF file
- Decide on abstract, clear, understandable label for what that string represents
- Put label and string in bundle: *gb_addflag_fieldset_flag_startdate=Check starting*
- Use label in JSF files

Navigation

- While the UI Designer is working on the JSF files, Programmers are working on the Backing Bean
- The UI Designer identifies where all of the links go, and what the buttons do.
- Programmers design the backing bean to supply actions that control navigation between pages.

Dynamic Content

- The UI designer identifies what data is needed from the database
- The programmer determines what services to use, and designs, implements services to call that data
- While the work on the services is going on, dummy content is provided to each mode by the backing bean

Bringing it together

- The UI, messages, navigation and dummy content create a click-through mock-up.
- This mock up can be used for an additional round of usability testing.
- Small adjustments to the UI and messages can be made now without affecting the programmer's work.
- When the services are implemented, we're ready for final QA and testing.

Development Issues

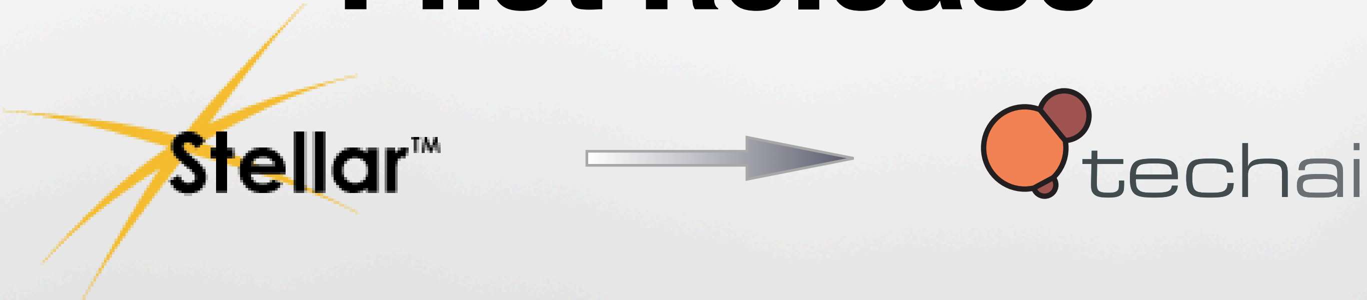
- We’ve been working while the framework is being developed
- The Sakai Style Guide will be a great aid to conceptual design
- We need a way for tool developers to create custom JSF tags, and for those tags to become part of the Sakai tag library
- Documentation for tool developers

Launch

Where we are now

- Developing new custom JSF tags
- Working on the Sakai APIs
- Recruiting pilot users at MIT

Pilot Release



- We plan to do a pilot release of the gradebook at MIT for the Fall 2004 semester
- 3-5 courses, instructors willing to use beta version of the tool
- We’ll make improvements to the tool based on the pilot release

Future Development

- The source code will be available to SEPP members during our Fall ‘04 pilot release
- Begin a new cycle of usability testing and requirements gathering
- We want to benefit from UC Berkeley’s experience with their gradebook
- Integration of other Sakai tools with the grade book