Course Tools

Kuali Days VIII continued Friday, December 11th, 2009

These are my notes from the conference sessions; they've taken a while to get written up. They're focused on Kuali Student rather than what Course Tools is building for Cambridge, so read on if it's Kuali Student you're interested in.

Conference details

Ian Boston (CARET CEO), John Norman (CARET Director) and Amyas Phillips (Course Tools project manager) attended to improve contacts with Kuali for Course Tools and Sakai.

Conference details [http://www.kuali.org/kd8] and schedule.
[h ttp://kuali.org/files/schedule/schedule.xml

Pre-conference Course Tools summit

We met with Leo Fernig and Scott Thorne, KS lead service architects, to try to knock out some apparent blocking issues, particularly regarding how KS associates 'learning units' (LUs) with 'organisations'. One way of handling the permissions necessary for a course being delivered jointly by five or six departments, it had been suggested, was to create a 'virtual department' comprising all of them. We were worried we might have to maintain n^2/2 or more different virtual departments to handle joint courses, but it turned out this needn't be the case. In fact, KS has a concept of a 'primary' organisation and 'others' for any particular role, which works well for our NST courses where many departments might be involved but only one holds the role of course organiser. Besides that, more than one kind of relationship is supported in KS – at MIT, Scott has successfully represented their system of multiple overlapping hierarchies for different areas of management.

Documentation for the canonical KS deployment process at an institution is still in development, including feedback from early implementers such as ourselves, but we identified some straightforward next steps for Course Tools.

Conference Keynotes

Daniel Greenstein, University of California, VP Academic Planning and Programs, and Dr. Raul Rodriquez, San Joaquin Delta College both delivered keynote presentations. They are responsible for successfully introducing Kuali systems at their respective institutions, and both took an interesting line on 'selling' Kuali to HE administrators.

HE-based IT development has struggled to shake off the impression "that the best is allowed to be the enemy of the good, that it panders to departmental special pleading, that it repudiates versioning, that it is not accountable for cost and function, and that it costs more to develop". Commercial software vendors have been preferred in consequence, and latterly commercial providers of IT services also, perhaps costing more but avoiding labour relations and being consequently more flexible. In this context, Daniel suggested that being open source is not the best selling point for HE-based projects like Kuali, especially now that open source is now an established part of the iT landscape. A better selling point is that Kuali addresses the academic part of the university organisation – an area generic sysems are afraid to engage with. In contrast the Kuali Foundation's motto is "Open source administrative software, by higher education, for higher education", and Kuali Student's unofficial motto is "your practice, not 'best' practice".

Introducing effective tools in this domain makes new things feasible – UC Berkeley for instance created a general course catalogue which it used to manage the pooling of resources across three campuses so that it could continue to offer classics. This kind of thing is important, because Institutions are being forced to focus more than they ever have on their core mission, and teaching is at least half of what HEIs do. Without the ability to integrate a view of their overall position, institutions are unable to make strategic changes.

Raul pointed up the success of the Kuali Finance System, now complete and live at a number of institutions, as proof of the community-source model in administrative computing. Rather than people asking "you did WHAT with open source?", the question will soon be "you paid how much, to whom, for what?" or even more pertinently, "how many scholarships did that decision cost?". Whereas commercial solutions try to capture ever more of the enterprise, open-source projects have different success metrics. To those who prefer to deal with trusted suppliers, he argued that community-source offers more security, not being dependent on a single entity. He saw it as a trust-building strategy – "KFS is exciting, but KS is the mothership".

Kuali Student Project Updates

Two sessions, one by Rajiv Kaushik, MIT, Kuali Student Project Manager; the other a KS Roundtable panel session.

Alpha (i.e. not feature-complete) release 1 (known as milestone 1, or just M1) is now available to implementers. This is the very first deployable release and although it is not feature-complete, the user-facing elements are really only the tip of the iceberg – the amount of thought and effort that has gone into the underlying service architecture means that the bulk ('90%') of the total work has already been done. Releasing M1 involved the

dev team in what is hopefully the last major piece of this – a considerable refactoring of the data orchestration layer, which translates from the highly abstract (hence reusable) services to concepts which are readily usable in the application domain. In the original conception, there was no need for this 'impedance matching' layer, but practicalities more or less demand it.

The dev effort is proceeding in six-week release cycles. M2 has subsequently been released to QA on December 7th and will be available to implementers after 2-3 weeks. Milestone 3 code freeze is scheduled for Dec 18, ready for QA in early January and released to implementers in late January. Pressure on institutions to cut back has meant that development resource is 27% below what was expected, and although the public 1.0 release is still slated for March 2010 it will now be followed by incremental released in May and July.

Each alpha milestone release is adding functionality, but following feedback from the implementers' user group it is recognised that there needs also to be a sustained focus on the actual implementation of configurability. There are many layers of configurability: CSS, labels on fields, moving fileds on screen, moving and adding new fields, setting up the data dictionary, swapping out entire services.

SIS operators often spend much time working around the mismatch between the SIS and their institution. Kuali Student is more configurable than this – it should be able to model each institution faithfully. The UI is configured in Java code, but the data model is configured in the 'data dictionary'. R1.x will include enhancements to configurability, 'award management' which is about approval processes, front and back matter for the course catalogue, and support for the IMS Learning Information Services standard],

[http://www.imsglobal.org/lis/alliance.html] which can is compatible with Cambridge's Oracle PeopleSoft SIS, CamSIS, via Oracle's Student Administration Integration Pack [http://www.oracle.com/applications/peoplesoft/campus_solutions/ent/module/student-administration-integration-pack.html].

Overview of KS Curriculum Management

Carol Bershad (University of Washington) took us through a course creation workflow for Kuali Reference University (KRU), giving an early peek at the KS user experience. KRU is a default configuration for Kuali Student, exercising many of the business requirements identified by the BA team and representing a fairly simple North American University. The course proposal itself is a simple web form at KRU, but in real deployments this could be as elaborate as necessary.

Much of KS LUM application development effort is focusing on the processes of curriculum management, drawing heavily on the services of the Kuali Rice middleware's Enterprise Workflow and Identity Management modules, KEW, and KIM. KIM looks after org charts, people and permissions, KEW assigns roles and shepherds things through processes, including conditional routing. KEW and KIM get involved mostly at

the proposal review and modification stages, less so at the submission and deployment stages which bracket them. At present, there is no 'overall' view built in within the system – something that would help both transparency and planning, and which Course Tools is interested in developing.

Within KIM is an organisation model, in which organisations contain 'positions', each of which is held by a person. In KEW, positions are assigned 'roles' in workflows.

Initial population of the course data in Kuali Student can be achieved via the orchestration layer's Java APIs, or directly into the underlying database.

Configuration in KS and the KS Data Dictionary

Leo Fernig, UBC; Scott Thorne, MIT; and Andy Bucior, Florida State University; Kuali Student service architects

The KS Data Dictionary (DD) is one of the main configuration points of KS, defining type-state pairs in the object data model. Learning unit types and their allowed states are defined in the DD. This allows institutions to create completely custom learning types, but it also means the DD is the natural place to start when localisisng for a particular institution. Other configuration points include workflow (KEW), the GWT-based UI framework itself, business rules (Business Rule Management Service), and the dynamic attributes which exist on each service block.

The DD specifically configures the data orchestration layer (DOL) and message structures. The DOL translates services' rather abstract concepts such as LU's to more concrete objects like Tripos. The first configuration step is to list types of durations, organisations, LUs and academic time periods (terms, etc.). Each of these may be constrained by allowed values, ranges etc. The next step is to edit the message structures (of which more details are available in the service contracts), e.g. renaming attributes and changing constraints.

KRU comes with five LU types, arranged in a hierarchy of scale: activities < formats < courses < general educational requirements < degree programmes. These differ in scale but not in kind (for KRU-based example, see fig.1), and the beauty of the LU concept is that this single structure can be used to model teaching 'objects' anywhere along this spectrum. It wouldn't be unreasonable to point at this as the key way in which pan-institutional configurability is achieved in KS.

Fig. 1

- **Key**: Course / Program
- Intensity: Credit Hours / Full Time
- Duration: Semester / Years
- Contents:Activities / Courses
- Enrollment Criteria: Prerequisites / Application Requirements

- Results: Credit / Grade GPA

KS User Interface Design

Kuali Student uses GWT to provide a rich UI that is modular and configurable, and also to take care of language internationalisation. The configurable UI framework consists of a set of default implementations for these UI components, as well as an API for developing custom ones. GWT deferred bindings are used to allow implementers to modify behaviours without changing KS Java code itself, an approach which also means upgrades won't destroy customisations. William Washington (University of Washington, KS UX Architect) is maintaining a UI reference framework[https://test.kuali.org/confluence/display/KULSTG/User+Interaction+Model] as a means of ensuring consistent look and feel and behaviour throughout KS.

Internationalising Kuali Student – free trade in learners?

Ian Boston and Amyas Phillips were panellists in two sessions that ran on this topic: "Taking Curriculum Management Outside of North America", and "Exchanging Student Records Globally". These sessions were intended as opportunities to share information about exchanging student records, both within and across different educational jurisdictions.

Kuali Student is strongly committed to the concept of internationalisation – the project charter makes a point of it, many of the founder institutions have a high proportion of foreign students, and as an open source project the bigger its base of users and contributors the more it will thrive.

Release 1 of Kuali Student is the curriculum management module. It was developed by a consortium of US and Canadian universities, but is meant to work in any educational jurisdiction. Our work in the Course Tools project is to some extent an opportunity to test this.

Release 2 of Kuali Student, scheduled for public release in mid 2011, covers enrollment and student records, for which designers have already started collecting requirements – including from 'discussion with institutions in Hong Kong, New Zealand, Australia, and Great Britain'. This means being able to consume transcripts that conform to North American and European standards as well as those of other regions.

The delegates attending these sessions mostly hailed from US institutions, although South Africa, Australia (and of course the UK) also featured. It was interesting therefore to talk about the Bologna process and the relative levels of standardisation and portability in the EU vs the US. Many US institutions make provisions for accepting credit transfers, but predominantly to accommodate students who find they have to move for non-academic reasons. The Bologna process' ambition to facilitate lifelong learning extends this idea further, and placement at another institution for educational reasons is understandably not a familiar idea.

The Kuali Reference University (KRU), a sort of default or example configuration of Kuali Student built from business process requirements of the founder institutions, gets implementers quite a long way towards a configuration that works to their own institution. All implementors have to do quite a bit more work to get to a production system, but non North-American implementors might be expected to have to go a bit further. How much further ios one of the hings Cambridge is discovering.

This was a question that also arose at our Kuali Workshop in May, where administrative computing officers were particularly concerned about the burden of keeping up to date with regulatory requirements, such as this year's visa monitoring rules. A lone institution would have to maintain all this itself, but a group of UK institutions (or their support contractor) could easily split this burden by making a 'KRU UK'. Given the Bologna processes' standardisation across Europe of a three-cycle HE system, it wouldn't be unreasonable to expect that a 'KRU-EU' could be used to share some localisation elements EU-wide.

At present there is no procedure for accepting local code contributions back into the project trunk, but that is certainly the intention once the project reaches that level of maturity.

At the student credit session I spoke briefly on the European Credit Transfer System (ECTS) and Diploma Supplement, and took the opportunity to plug XCRI [http://www.xcri.org/] as a likely European standard for sharing this information. I suggested that a portable student record should include a record of competencies, academic achievement (credits and levels, where credits might be alotted on completion of a nominal number of hours of study or on achieving learning objectives to at a minimum level), grades and grading scheme, awards and award system, and a description of what exactly was studied.

It seemed XCRI was news to the delegates but there was discussion of the PESC post-secondary transcript [http://www.pesc.org/interior.php?page_id=164] (which apparently achieves broad compatibility by enumerating all possible schemes), the RS3G [http://wiki.teria.no/confluence/display/RS3G/Home+of+RS3G] (also see PESC's commentary) [http://www.pesc.org/interior.php?page_id=176] and HR-XML [http://www.hr-xml.org/hr-xml/wms/hr-xml-1-org/index.php?language=2].

Finally, an interesting discussion took place around the question of whether a University's desire to maintin a relationship with its graduates trumps it's graduates' ownership of their transcript data.

Sakai and Kuali

Michael Korcuska, Executive Director of the Sakai Foundation presented a general overview of the Sakai project, community, and potential collaborations with Kuali projects.

Sakai and Kuali are about as different in style as it is possible for two community-source projects to be. Kuali is formal and highly managed, Sakai seems to work on a 'best efforts' principle and has a comparatively organic style. Kuali's design is reminiscent of Oracle or SAP, Saki's more of Google Docs or Flikr. These differences probably reflect what their respective customer bases are used to working with – academics and students on the one side, university administrators on the other – but there are more philosphical differences. Sakai's functional council decides what to merge in to the trunk from the community, Kuali's decides the development roadmap the community should follow. Sakai 3 has adopted a policy of incorporating open-source projects wherever they deliver functionality it needs – the majority being infrastructural. Kuali Foundation projects use very little external code, preferring to keep management in-house, and keep control.

Sakai 3 is being developed around 7 principles: learning space construction, workflows (no tool 'silos'), scholarly networking, breaking site boundaries, 'everything is content' (taggable, searchable, shareable), groups != sites. It explicitly isn't a CMS, as everyone is a contributor in Sakai, whereas most CMS assume few authors and many users. Scholarly networking is based around both content (like Flikr) and people (like LinkedIn).

Sakai-Kuali Student interfaces might include learning design elements, for curriculum, course or ctivity; presentation of the syllabus on campus and to the public, assessment and grading, and student portfolios, and they might be based on the aforementioned IMS Learning Information Services standard [http://www.imsglobal.org/lis.cfm].

General notes

I would have liked to attend some of the parallel sessions on Kuali Rice, which looks rather capable and interesting, especially Enterprise Workflow (KEW), [http://rice.kuali.org/kew] but KS sessions trumped all of them.

An interesting session for next year, or for our own KS conference, might be "**KS LUM and pedagogic practice**".

Posted in Kuali Student, University administration | No Comments »

Notes from Kuali Days VIII Tuesday, November 24th, 2009

Ian Boston, CARET's technical director shares his take-home points from the Kuali Days conference [http://www.kuali.org/kd8], held last week in San Antonio.

We went to Kuali Days to strengthen ties between Kuali Student and the Curriculum Design project at Cambridge as well as strengthen ties between Sakai 3 and Kuali. Although the conference ran over Tuesday 17th and Wednesday 18th we were present from Sunday 15th. On Monday we had a constructive meeting with Leo Fernig (UBC, lead technical architect) and Scott Thorne (MIT, service architect) to discuss how the curriculum design project would work in Kuali Student.

Curriculum Design

It became apparent as we went into the detail that the headline datamodel of KS did not match the details. The main issue for Cambridge is that a learning unit (LU) is associated with many organizations. In Kuali, at the headline model level, an LU is associated with a single organization. Once the detail was unpacked it became apparent that an LU is associated with many organizations, each association annotated with metadata. In the case of Cambridge we have many stakeholder organizations associated with each LU, each organization with a different concern. There is a single coordinating organization for the LU which is equivalent to the headline organization-LU relationship visible from outside the project. The task now is to analyze the Cambridge LU model and map it into KS to determine the gaps.

Kuali Student like many of the Kuali projects has been driven by a analysis stage performed by 30 or more Business Analysts at the participating institutions. The output of their work can be found on the KS wiki (you will need to register for a login) where there are descriptions of the entire entity model and service descriptions. The use of that wiki in itself is interesting since each page contains markup from which much of the service framework and entity model is generated at build time. Although there is an entity model and service definition in KS much of this is abstract, concreted by a data dictionary that defines each type of entity within the model. This similar to most of the other Kuali products which require local implementation effort to adjust the data dictionary to describe the local environment. For instance there is an abstract definition of a LU, which may represent a course in North America or a Part of a Tripos at Cambridge. Changing the system to represent both of these concepts is a matter of changing some data within database to represent the Cambridge model. KS comes with a standard configuration known as “Kuali Reference University” which is close to most North American models. We will have to modify sections of that “University” to form a localized model.

In addition to the datamodel within Kuali there is a workflow engine, Kuali Enterprise Workflow (KEW) [http://rice.kuali.org/kew] [part of the Kuali Rice [http://rice.kuali.org/rice] middleware] and a rules engine, Drools [http://www.jboss.org/drools/]. Both of these use the data dictionary to bind actions and rules to entities within the data model, consequently modifications to the data dictionary will require changes in the workflow and rules definitions. KEW is shared by other Kuali applications.

In conclusion it is felt that there is no reason why the Cambridge teaching model cannot be represented in KS, however we need to validate that assumption by attempting mapping.

Kuali Student UX

Many Kuali projects use Kuali Rice to create their UI. This is a highly data driven Forms creation environment that results in the rapid implementation of forms based systems which are closer to an administrative client server application than a web environment.

Kuali Student is unique within Kuali by not adopting this approach to user interface. It has taken a more design focused approach that is apparent from the resulting user experience, although it is not nearly as UX driven as Sakai3. KS uses Google Web Toolkit (GWT) [http://code.google.com/webtoolkit/] to deliver its UI. It is more interactive and responsive than the web 1.5 applications that typify the rest of Kuali, but the use of GWT and the development process restrains the UX process. They have a UX designer, Will Washington at University of Washington who is working within the constraints of the Java development team using GWT to deliver a UX. He is developing navigational components and detail components [https://test.kuali.org/confluence/display/KULSTG/UX+Team] that are implemented in GWT to provide a UI constructed from those components. Kuali Student is intended for a wider user base than other Kuali products, which are focused squarely on serving central administrators, and this UX approach is light years ahead of those products in terms of usability. It does not come close to the freedom and flexibility available within the Sakai 3 UX process, but this can be seen as a reflection of different project priorities.

Sakai 3 and Kuali

The model of Sakai 3, with its reduction in code base by extensive use of components adopted from 3rd-party open source projects, was held up as a model for Kuali which like Sakai 2 had adopted the "not invented here" approach to third party libraries. Oddly the result of this approach is a mish-mash of dependencies in the centrally provided systems that does not always result a smaller or efficient footprint. For instance, the Kuali Coeus release is a 200MB download with about 30 – 40 jars all packaged into a single webapp. Much of that code base including components like Kuali Enterprise Workflow, Kuali Service Bus (KSB) [http://rice.kuali.org/ksb] and Kuali Nervous System (KNS) [http://rice.kuali.org/kns] is maintained within the Kuali Foundation. The Sakai 3 model where the KSB would be represented by Apache Service Mix, [http://servicemix.apache.org/home.html] and KNS by Apache ActiveMQ [http://activemq.apache.org/] was of interest.

Sakai 3 and Kuali Student

There are obvious areas for integration between KS and S3. KS provides management of the structure of courses and a repository to store that structure and S3 provides an interface for the teaching and learning community to interact with that model. We had discussions surrounding how closer cooperation might work, including the concept of LU's being derived from the authoring process of a course template within S3, and that the structure of LU's being taken through an approval workflow within KS might result in a published course structure from which S3 would be able to generate course instances. Clearly this is very early stages but there are positive indications of both sides.

Kuali Coeus

Kuali Student addresses the administration of teaching, one half of a university's core activity. Kuali Coeus[http://www.kuali.org/kc] is intended to address the other half – research.

I sat in on a number of Kuali Coeus sessions to gather information. Kuali Coeus was until recently called Kuali Research Administration. Coeus is a MIT driven project that addresses the needs of Research Administration. Coeus [http://coeus.org/] has been going for 15 years and has a community of adopters within North America. Kuali is attempting to achieve convergence between KRA and Coeus in the form of Kuali Coeus.

The history of KRA is checkered. The initial roadmap staked delivery dates for various feature sets, all of which have been missed, however there is new hope converging with Coeus as it represents of body of knowledge in the area. The Coeus product is a Java GUI bound to a Oracle back end, although there is a light version with a web based interface. KC will follow the standard Kuali Rice model, an abstract data model represented as Oracle Tables, a data dictionary to concrete that model and a forms based UI backed up with Kuali Enterprise Workflow. Although this produces a user interface that is unusable outside administrative circles, the implementation cost is low allowing resources to be allocated to analysis phases.

Unfortunately KC has many integration points and features that bind it closely to Research Administration and Grant management in North America (eg Government Forms). It is likely that any other region would have to expend significant effort in re-defining the Workflow, data dictionary, perhaps data model and certainly integration code to make it suitable for another region. It is debatable and unknown if the cost of that re-definition would be more expensive that doing the work from scratch.

Posted in Kuali Student, Project development, University administration

Will bids farewell: how to build a course database Wednesday, September 16th, 2009

A guest post from our summer student William Brookes, who now returns to Part II Mathematics after doing some great work with Course Tools during his eight weeks at CARET.

I have come to the end of a very enjoyable 2 months working on the Course Tools project. I have learnt more than I am ever likely to need about the intricate structures of various degrees at Cambridge and how you can combine and convert between different Courses and Parts.

The simplistic structure I envisaged for degrees was that Lectures/Practicals make up Courses which make up Parts (what's a Part at Cambridge? [http://www.cam.ac.uk/admissions/undergraduate/courses/tripos.html]) which make up Degrees. Lectures seem to be in a single course, whereas courses and parts can be in

multiple parts and degrees. In addition, multiple departments can be involved in running a course or a part.

I wanted to be able to require certain information about different levels of the degree (Lectures need times and days of the week, degrees need information for prospective students) so decided to make a relational database with each level as a table. Where there was a need for multi-parenting, I added an additional join table which allows many joins to be made.

Upon collecting data, I found not all degrees fitted neatly into my original structure. Part *Engineering IIA* groups its courses into "Engineering Areas" which is important for specialisation (you must take 6 courses from your specialist area). Course *Biochemistry and Molecular Biology IB* divides the lectures in it into groupings. I felt my database needed to be able to store this information as it is important to the structure of the course.

To solve this, I could have added additional tables to the database for each of these groupings, with associated join tables (what's a join table?) [http://megocode3.wordpress.com/2008/01/04/understanding-a-sql-junction-table/]) . This would create multiple routes through the hierarchy and make it less clear where data should be stored. I aimed to use the fact that lectures and courses are only in a single grouping to avoid restructuring my database. Instead I added them as an additional field, which both allowed the structure to be stored without complicating the database. I also created a join table between Parts so I could record that a Part allowed you to do another Part. This is particularly useful as Cambridge aims to enable students to switch between courses simply, and this helps to see it.

My structure caused a few issues during implementation. In Natural Sciences, you specialise as you progress, choosing 4 courses in IA, 3 in IB and 1 in II. From this progression, one would expect something like *History and Philosophy of Science II* to be a course, however it comprises of things that are structurally courses and it is a whole year's work. I therefore made the Part II NST into parts in their own right. This fits structurally, and *Physical Sciences II* contains courses from NST IB so were it a course it would be a course within a course which would not fit within my system. This brings up issues as I am defining things by their structure rather than what they should naturally be.

In addition, lots of the groupings of courses are called things like "course" or "option". This created problems in identifying them, as groupings do not have a unique identifier. Part *Physical Sciences II*, for example, contains 2 groupings in it both called "option". The best way I could find to attempt to avoid this is to say groupings are the same if the courses are included in the same parts, else they are different. However this is not a satisfactory solution.

Kuali Student's nested hierarchy avoids the issues in difficult structures and is able to do any groupings necessary without needing a change to the database. It uses templates for different things and so creating a whole new structure is simply a case of creating new templates. A current issue that arrises from Kuali is that it does not permit courses to be

multi-parented by departments. This is highly unsatisfactory for modelling the Cambridge structure.

After much studying of the structure of courses at Cambridge, I have come to appreciate the level to which it is designed to aid the students. Much effort is put into allowing students options to change and flexibility in their courses. While this may complicate any models of structure, it is a valuable service.

My conclusion about a design for a way of storing the data is that a rigid, hardwired structure is not a good solution, as it forces things to comply and has to fudge awkward cases. It cannot adjust well to potential changes in structure in the future and some structures may arise that do not fit. A solution like Kuali will solve such structural problems so is likely to be a good solution, but due to the vagueness of the structure great care is required in implementation.

Posted in Kuali Student, Project development,

Key Kuali staff joining our Kuali Student workshop Tuesday, July 7th, 2009

An exciting update on our Kuali Student workshop [http://coursetools.caret.cam.ac.uk/post/kuali-student-workshop-register-your-interest] on the 23rd July.

Leo Fernig, chief technical architect and the main speaker of the day, has arranged for four of his colleagues on the project to join us via Skype for the afternoon session's discussion, on mapping Kuali Student's Learning Unit Management (KS LUM) service to Cambridge's tripos structure. This will be an opportunity to explore in depth KS LUM's ability to model a challenging UK HE course structure.

The Kuali team members joining us via Skype will be:

- **Scott Thorne** and **Andy Bucior**, the Kuali Student service architects.
- **Cathy Dew**, who will be leading the team implementing the Learning Unit service for 'experiential leaning and projects' at UC Berkeley. Like Cambridge's triposes, these configurations are expected to test the adaptability/flexibility of the LU service.
- **Cindy Nahm**, who will be leading the functional implementation team at UC Berkeley. Cindy is also on the Business Process UIG [https://test.kuali.org/confluence/display/KULSTP/IUG+-+Business+Process] (User Implementation Group) which is a focus group for common concerns around the the process of implementing Kuali Student.

We're really excited to have them join us and looking forward to an exciting day.

Anyone who hasn't registered for the workshop but would like to follow can do so via our Ustream feed.

Posted in Kuali Student