Notes on the Development of Learning Systems: A retrospective of experiences

This is a summary of posts by Mark J. Norton, Zach A. Thomas, David Adams, Antranig Basman, Sean Keesler, and Chuck Severance and observations.

1. Experience and knowledge of learning systems

   Long term projects such as the one discussed creates understanding of the learning processes. This experience can provide a list of "requirements" that guides short-term and long-term development and criteria for those procuring learning systems. A "history"—in the form of documents, video or audio—could be beneficial to others developing complimentary and competing products or procuring a learning system.

   Sean Keesler observed: "The [open source learning system project] community seemed to move from a technology-led group towards a more equal representation of pedagogy-minded and technical stakeholders. I saw this in the work of a number of really visionary non-programmers who were laying out some important principles of what they thought the future of academic computing needed to be." He points out this knowledge "should be applied towards any technical direction that the community takes...regardless of HOW [his emphasis] it is implemented."

2. Choosing technology for development should begin with engineering studies of performance.

   David Adams observed "After four years, the system [in development] can't support more than a few dozen users at a time, or search through more than a few thousand objects. Meanwhile, the potential customers need to operate at hundreds or thousands of times that scale. The architecture has been flawed from the start; built on false assumptions, buzzwords, and crossed fingers."

   To avoid this problem, Antranig Basman suggests quantifying a typical workload in terms of "transactionality and related throughput, perhaps expressed in terms of requests per second, related to an expected deployment size and number of active users." He also cites the "lack of appetite for steady, detailed work" as a source of failure.

3. The development of any complex software should begin with the development of a simple function.

   David Adams quotes John Gall: "A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system." Adams comments: "To write a successful complex LMS, we would need to start from a successful simple LMS."

4. There is a difference between a "platform" and an application that itself delivers useful functionality.

Referring to learning systems, David Adams also observes: “If anything, the story of [the software] is that it long ago moved from being an attempt to write an LMS to being an attempt to write a *platform* for someone else to write an LMS. That it's attempting to build a generic platform on top of other generic platforms (Sling, Jackrabbit, etc) may be part of the problem.”

This is an important difference since many learning systems start-ups are claiming to be “platforms.”

5. The pattern of activity and results of a complex software project on the path to becoming failure are known, but typically neither recognized or, if recognized, initiate corrective management.

   Zach Thomas cited “The Productivity Trap” and “Mythical Man Month” as describing projects headed toward failure. “Uncle Bob Martin” describes software that is built without clear design as a “mess.” He points out “That the high productivity you enjoyed at the beginning had now plummeted. Things that took you hours before took your days. Things that took you days before now took you weeks, months, or can’t even be done at all.” One of the causes was lack of a clear and complete requirements document. “You think there was a requirements document? Even if there was, do you think it was accurate [and current]? No, there were too many last-minute fixes and midnight modifications.” However, “The real requirements for the new system were in the old system, in the old system’s code.” His observation: “Then developers on the team started demanding a redesign.”

   Many higher education open source software products that offer both new technology and additional functions are severely behind the original schedule. The pattern Thomas described appear in several open source projects important to higher education. His insight may help project managers identify, understand, and remedy these delays.

6. The difference between now and the future

   Chuck Severance comments: “I just don't want to make the mistake again where we start talking too much about a grand and glorious future and it ends up diffusing the focus on the present. If we don't tend to the present - getting excited about a future is pretty pointless.” Describing a working system and its redesign Mark Norton committed: “The [current system] had a lot of momentum - until [a proposed replacement] appeared as a distraction.”

   This may be very timely advice when the public is daily offered a new technical solutions that in the future will more effectively instruct students and facilitate learning at sharply lower costs.

These comments were taken from posts made September 7 through September 9, 2012 of the Sakai Foundation’s Open Forum, Announcements, Sakai OAE, http://collab.sakaiproject.org/pipermail/openforum/.