

Agile Project Management at Penn



Delphine Khanna
University of Pennsylvania
DLF Forum -- Nov 1, 2010



How we got there

- Grew organically over the last 4 years through trial and error
- Goals
 - Create a project management structure that:
 - Keeps overhead to a minimum
 - Allows for maximum flexibility and nimbleness
 - Gives us a sense that things are under control
 - Deadlines met, software development not seen as the bottleneck, ability to complete each project more quickly, and handle more projects concurrently and efficiently



How we got there (2)

- Took quite some inspiration from the agile/scrum approach
 - Agile Manifesto
 - Formal agile/scrum training about a year ago
 - With Kristine Shannon
 - Not trying to be systematic about following scrum
 - No claim at all that we are a Scrum shop
 - We don't use most of the jargon! ;-)



Note

- To give the context of our approach
 - Describe a bit our software system and our team set up



Key concepts in our project management approach

- Software development = series of distinct functionality pieces (Scrum: “user stories”)
- Transparency
 - Everybody in the organization can see exactly what we are doing and where we are in our development
- Clear and reliable milestones
 - (Internal) customers trust that we will meet our milestones
- Go into production only once a month
 - Move new code from development server to production server
 - Creates a regular rhythm around which our work is organized (Scrum: “sprints”)



Software system: the DLA

- PM approach went hand in hand with the development of a “generalized” software system
 - A single system that handles all our delivery needs
 - Collections of images, book facsimiles, EAD finding aids, “netflix-style” video catalog, staff directory, and much more...
 - Based on Solr/Lucene
 - With generalized ingestion tools and generalized web delivery
 - Both customizable through configuration files



Software system: the DLA (2)

- Some new projects require 0 core development
 - E.g., a new image collection, when we have already the features needed to handle image collections
 - Just a matter of ingesting the collection and configuring it
 - XSLT/CSS-based customization
 - + Cataloging / metadata clean up / scanning / QA, etc.



Software system: the DLA (3)

- Some new projects require some new pieces of functionality
 - E.g., a collection of arabic book facsimiles requires us to add the functionality “right-to-left page browsing”



Consequences on project management

- Clear distinction between “ingestion of a new collection” and “software development”
- Ingestion is done by “DLA Ingesters”
 - They do not need to be expert programmers (mostly XML and XSLT)



Consequences on project management (2)

- Software development
 - Done by Core Programmer(s)
 - Seen as a list of functionality pieces
 - Independent from each other
 - Small to medium in size
 - Advantages
 - Easier to establish priorities
 - Easier to control the timeline
 - Come up with clear milestones
 - Quicker results
 - Each time a piece is ready, it can go live, without waiting for a big release at the end of several months



Team structure

- 1 DLA Software Team
 - Develops the DLA software (all new DLA functionality pieces)
- Composed of:
 - 1 Team Lead / Project Manager
 - Core Programmer(s): 0.5 to 1 FTE



Staff structure (2)

- 4 DLA Content Teams
 - Ingest new collections into the DLA system
 - Each team focuses on one format
 - Images, Book Facsimiles, OPAC Subsets, Non-Marc (EAD, OAI, etc.)
 - Core members on each team
 - 1 DLA Ingestor, 1 Cataloger/Metadata Librarian, 1 Public Services Librarian, Web Designer
 - Note: not a cast of thousands
 - E.g., I am the Team Lead for DLA Software Team and a DLA Ingestor



Staff structure (3)

- Guest members on each DLA Content Team
 - Onboard only for the duration of one project
 - Collection-specific experts
 - Curators, bibliographers, catalogers, HR person, etc.
 - One of them is always the “project owner”
 - Provides ongoing advocacy for the project (even after the project is completed), takes care of it, notices problems in the long term, etc. (+/- Scrum: “Product Owner”)
- Advantage of having core members
 - Develop very strong DLA expertise



Heavy use of Google Docs spreadsheets

- Especially to manage each team's to-do list (Scrum: "backlog")
- Very low overhead to enter a new to-do item
- Can easily sort list based on various criteria
- Edit the spreadsheet during the meeting
 - By the end of the meeting your to-do list is essentially up to date
- Each to-do item is assigned complexity points and priority points (another Scrum thing)
 - Helps prioritization effort



Heavy use of Google Docs spreadsheets (2)

- Every to-do list is viewable by all staff
 - Total transparency
- Helps communicate on “what keeps you busy all day” question
- Helps working out prioritization issues across the organization
 - People can see where their desired functionality stands, and which other functionalities have higher priority
 - Really helps them understand “why” we are not working on their functionality right now
 - DLA Oversight Group can easily see our priorities and decide to reorganize them if needed



How many spreadsheets?

- 1 spreadsheet for the DLA Software Team
- 1 spreadsheet per collection
 - A DLA Content Team works on one or two collections at a time



Grooming the to-do list

- Very important
 - (Another Scrum concept)
- Done by the Team Lead
 - (Scrum: “Scrum Master”)
- Keep updating the to-do list
 - Make sure it gives an exact picture of the current reality (no tasks missing, etc.)



Grooming the to-do list (2)

- Look down the list to prepare the tasks
 - Identify road blocks (Scrum: “impediments”)
 - Remove them by talking to the relevant people
 - E.g., Sys Admin for new storage



DLA Software Team's to-do list

- During meetings: mostly look at the to-do list and update it live
- Work only on the first 4 or 5 to-do items at the top of the list
 - Clearly marked as “active”
 - All the other to-do items are officially inactive
= Waiting in line for their turn
 - Loose adaptation of Scrum’s “Sprint backlog”
- Forces clear prioritization
 - Can’t vaguely claim that you are working on “everything”



Milestones

- For every functionality piece that someone is actively “waiting for”
- Fake meeting in Meeting Maker (our web-based calendaring application)
- Works amazingly well because the milestones are right under people’s nose all day
 - Programmer cannot “forget” about it, and sees it coming
 - “Customer” is reassured, and does not ask you about their new functionality every 2 days



Milestones (2)

- “Move to production” milestones
 - Once a month at a predictable time (end of the month)
 - Also in Meeting Maker
 - Because moving small changes from the development server to the production server was becoming a full-time job for our programmer
 - People got used to this surprising quickly
 - Creates a regular rhythm around which our work is organized (Scrum: “sprints”)



Meetings

- No daily meetings like in Scrum
- DLA Software Team meets once a week
- Each DLA Content Team meets about every 2 weeks
- One overall DLA meeting a month with all the core DLA members
- Plenty of informal communication on a need-be basis



Process review

- We review our process regularly to see what can be improved
 - esp. at the end of each project



“Finishing” a project

- No more never-ending projects
- Push hard to go live early
 - Share development version of the project right from the beginning
 - Everybody sees the site evolve as we go
 - Ongoing testing
 - Put into production as soon as the site is minimally functional



“Finishing” a project (2)

- Control functionality creep
 - New functionality pieces
 - Waiting for their turn in the big DLA Software to-do list
 - “Competing” against all the other pieces in terms of priority
 - Each DLA Content Team knows that
 - Chooses the smallest possible subset of functionality to be implemented by go-live date
 - Functionalities on “Wish list” developed after go-live date (e.g., image rotation)
 - Negotiate reachable milestones for most important functionalities



“Finishing” a project (3)

- After a project goes live
 - The DLA Content Team completely stops working on it
 - Except bug fixing
 - No more meetings
 - The Project Owner collects feedback and creates a wish list



“Finishing” a project (4)

- DLA Content Team briefly reopens the project about 4 months after it goes live
 - Reviews the wish list
 - Decides if new pieces of functionality should be put on the DLA Software Team’s to-do list
 - Decides if the Ingester should be doing a few small tweaks
- If a project needs a new round of development
 - Handled as a completely separate project
 - Added to the DLA Content Team’s list of future projects



Conclusion

- We are very happy with this model. It really works for us!
- My recommendations
 - The Agile/Scrum approach is very powerful
 - Use it as a source of inspiration
 - But don't be afraid to pick and choose
 - Try pieces of it and keep what works for you
- Questions?