Agile and Crystal Clear with Library IT Innovations

May Chang
Head, Library IT Services
University of Maryland, Baltimore County
maychang@umbc.edu

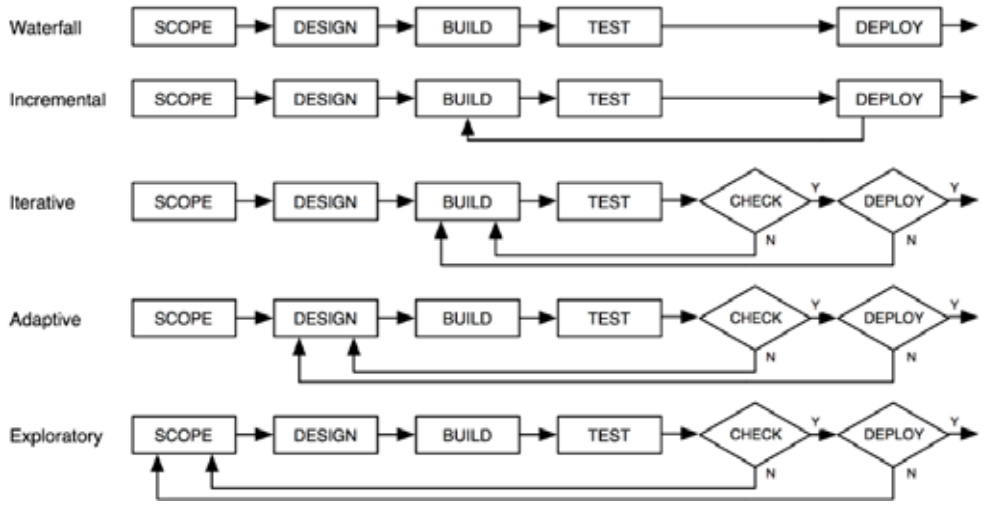
Abstract:
The Agile project management approach with the Crystal Clear method was used to rapidly develop and deploy a variety of innovative IT applications. Our A-Team of staff and students thrived on the flexible and iterative process, and helped fine-tune our adaptation of the Agile approach for future IT projects.

Introduction

Information Technology (IT) applications and innovations come in all shapes, sizes and complexity, and often faster than many libraries can keep up with. While we are keen to investigate and implement them, many of us are faced with limited staff and financial resources, especially in the current economic situation. There is also the awareness that the IT environment is very fluid, where change is the new norm, and that some traditional organisational and software development practices may not be flexible enough to keep pace. Faced with these challenges, this author chose to focus on better software development and project management practices for new IT projects.

There are various software and IT development methodologies. Figure 1 shows these arranged by decreasing degree of linearity, i.e. from little stakeholder feedback to increasing reliance on feedback as an integral part of the development process (Ward & Legorreta 2009, p3).

Figure 1: A taxonomy of software engineering models



Waterfall models are characterised by a one-shot approach and each stage of the development is fully completed before starting the next stage. The development process is sequential and accepts little or no feedback to minimise "scope creep". Incremental models are often chosen when the project is too large to implement in a single development cycle. They are conceptually identical to waterfall models except that the project is divided into smaller implementations and the development team asks for user feedback. Iterative models are hybrids of waterfall models and exploratory models. They plan, build, test, and implement small units of the project at a time, add new functionality where necessary, and often return to existing units to expand or improve these. Adaptive or Agile models incorporate a good deal of research and exploration while developing the IT solution. Requirements elicitation is an ongoing activity, because needs are constantly changing, and user feedback is collected at the end of the development cycle. The development team is often willing to go back to the design phase to redesign the system around user feedback. At the

other end, exploratory models are pure research and development models that focus on searching for the optimal solution to vague or unknown problems. Significant stakeholder involvement and feedback is required and the development team is willing to revisit and redevelop from the beginning phases of the project.

Of these, the adaptive or Agile model was the best fit and most appropriate for our environment and needs. With this lightweight and adaptable approach, a small team of selected staff and student assistants was able to fast track the implementation of innovative applications, without sacrificing quality. Within ten months we had completed a variety of innovative projects, and team members were energised by the non-traditional active development and learning environment. This was a new and untried process for us, but the overall success gave us sufficient reason and confidence to continue with our adoption of the Agile approach.

Agile methodology

Agile is an approach to system design and development based on incremental and iterative processes. It also refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Although the term was coined in 2001, the modern definition of Agile software development evolved in the mid-1990s as part of a reaction against "heavyweight" methods, perceived to be typified by a heavily regulated, regimented, micro-managed use of the waterfall model of development (Wikipedia 2009a). In 2001, a group of representatives of various new methodologies met to discuss the need for an alternative to documentation driven, heavyweight software development processes. This resulted in the Manifesto for Agile Software Development that laid out the values that underpin the approach. Basically, proponents and practitioners (sometimes referred to as Agilists) value:

- Individuals and interactions *over* processes and tools.
- Working software *over* comprehensive documentation.
- Customer collaboration *over* contract negotiation.
- Responding to change *over* following a plan.

Behind the Manifesto is a collection of twelve principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The developers, sponsors, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximising the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Agile methods generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organisation and accountability, a set of engineering best practices that allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals (Wikipedia 2009a). In Agile project management, Highsmith (2004, p27) laid out two categories of guiding principles:

Customer Value through Innovative Products

- Deliver Customer Value
- Employ Iterative, Feature-Based Delivery
- Champion Technical Excellence

Leadership-Collaboration Management Style

- Encourage Exploration
- Build Adaptive (Self-Organising, Self-Disciplined) Teams
- Simplify

These principles resonated with this author's approach to software development and project management, and were fully incorporated into our practice. Team members are also often reminded of these as they develop applications.

Several methodologies have evolved with Agile at their core, including Extreme Programming (XP), Scrum, Lean Software, Features Driven Development (FDD), Agile Unified Process (AUP), Dynamic Systems Development Method (DSDM), and the Crystal Method. While all these methodologies subscribe to the Agile principles, each has different core values, processes and requirements.

Coffin and Lane's (2007) two-part article provides a comprehensive description and table of the strengths and weaknesses and the appropriate use of these Agile methodologies. For example, XP is a programmer-centric methodology that emphasises technical practices to promote skilful development through frequent delivery of working software. Scrum's primary goal is to deliver software that, after every iteration, provides the highest business value. It is based on a generally accepted four-week iteration called a "Sprint." Lean's goal is to meet the challenge of defining, building, and delivering complex software systems that are exactly what a business really needs to stay competitive. A lot of attention is paid to gathering the "right" requirements which must be defined in clear, complete, and verifiable ways. FDD is features-centric, and has a specific order of process: develop an overall

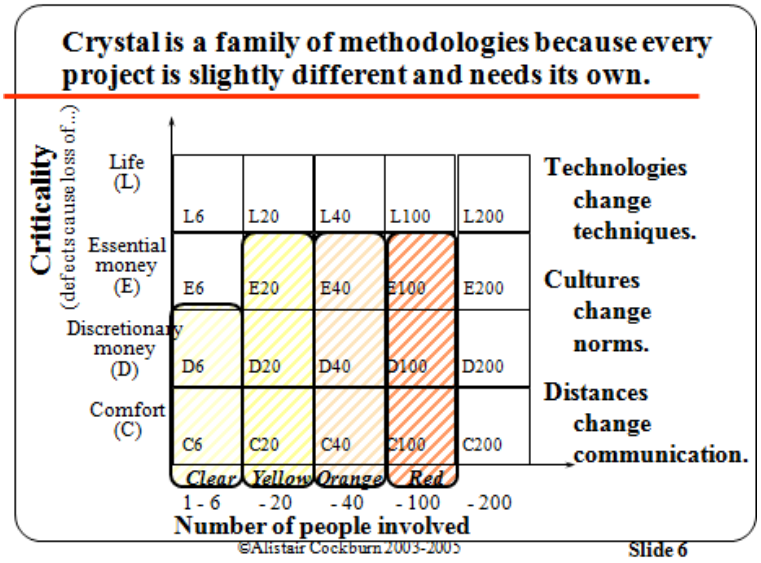
model, build a list of features, plan by feature, design by feature, and build by feature.

After some research, this author selected the Crystal method as the one best suited to our needs, particularly because of the small staff size and low criticality of proposed projects.

Crystal Clear

Originated by Alistair Cockburn, Crystal is a family of methodologies (Clear, Yellow, Orange, and Red) and is based on project size (where the number of people involved are about 1-6, 1-20, 1-40, 1-100) and criticality (where defects could cause loss of comfort, discretionary money, essential money or life). As the team size grows, Crystal implementations change to add more formality to the structure and management of the project. Project criticality also increases the rigidity of the project needs to ensure the expected demands can be delivered. Crystal thus acknowledges that each project may require a slightly tailored set of policies, practices, and processes to meet the project's unique characteristics. The Crystal implementation most appropriate for our needs is Clear based on size (1-6 people) and low criticality.

Figure 2: Crystal: A family of methodologies (Cockburn 2005b)



The characteristics of Crystal include:

- *human-powered* where the focus is on achieving project success through enhancing the work of the people involved (other methodologies might be process-centric, architecture-centric, or tool-centric, but Crystal is people-centric).
- *ultralight* where for whatever the project size and priorities, a Crystal-family methodology for the project will work to reduce the paperwork, overhead and bureaucracy to the least that is practical for the parameters of that project.
- *stretch-to-fit* where you start with something just smaller than you think you need, and grow it just enough to get it the right size for you.

(Cockburn 2008)

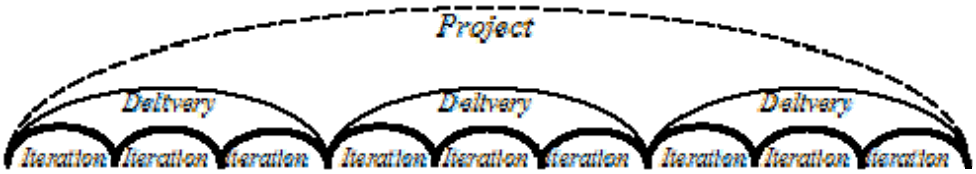
Cockburn stressed the importance of communication and conversation: "Focusing on skills, communications, and community allows the project to be more effective and more agile than focusing on processes and plans" (Highsmith 2002, p261). He also listed the properties of successful projects in Crystal regardless of which implementation; the first three are required, while the next four get the team further into the safety zone where team members are most comfortable and working at their best:

1. **Frequent delivery**. Stakeholders can expect deliverables every couple of months and/or see intermediate versions, and be able to provide feedback.
2. **Continuous feedback** and reflective improvement. Team members discuss activities to ensure the project is headed in the expected direction and to communicate any new discoveries that could impact the project.
3. **Constant communication** with co-location. Small projects expect the entire team to be in the same room, while larger projects are expected to be co-located in the same facility.
4. Personal safety. This comes in two forms – one where team members can communicate and be effective without fear of reprisal, and the other recognises that some projects affect the safety of their end-users.
5. Focus. Team members are expected to know the top two or three priority items each member should be working on and should be given time to complete them without interruption.
6. Easy access to users, stakeholders and experts.
7. A technical environment that supports versioning, automated testing, and frequent integration of system components.

Crystal can thus be seen as a highly optimised way to use a small, co-located team, prioritising for efficiency, habitability, and safety in delivering a satisfactory outcome (Cockburn 2005a, p303).

Crystal uses nested cyclic processes of various lengths: the development episode, the iteration (a unit of estimation, development and celebration, for example one week to three months), the delivery period and the full project. What people do at any moment depends on where they are in each of the cycles.

Figure 3: Iteration and delivery cycles within a project (Cockburn 2005a, p113)



In the Agile approach, the only true measure of progress on a software development project is the delivery of working software. By working in short iterations, and by delivering working software each iteration, there is evidence that the project team is progressing. As described by Ambler (n.d.), this provides greater visibility into the true status of a project than does documentation-based earned value measures, and also improves the ability to effectively govern IT projects.

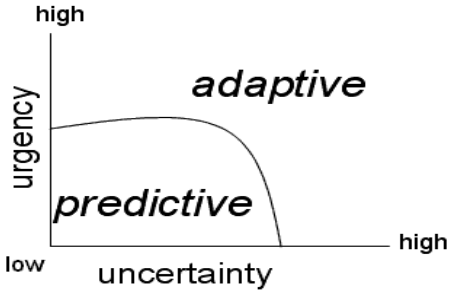
While predictive methods do lead to predictable outcomes, Nicolette (2005) noted that more commonly, they led to cost overruns, long bug lists, excessive overtime, dissatisfied customers, and poor business value. In the adaptive or Agile model, the development process is a series of iterations, and there is continuous adaptation and innovation through close collaboration between developers and stakeholders. This often results in better and more reliable features and on a shorter delivery schedule.

Although an Agile approach can better address business and customer needs, it must be noted that it is not appropriate for all environments or projects. For example, Agile methods work well in an organisational culture where uncertainty and change is welcomed and creativity and innovation can thrive. Nicolette (2005) highlighted the general elements of when to use the predictive or adaptive approach in Table 1 and Figure 4 below.

Table 1: When to use Predictive and Adaptive models

Predictive (Waterfall)	**Adaptive (Agile)**
Low uncertainty and low urgency	High uncertainty or high urgency
Repeating process	Non-repeating process
Not new product development	New product development
Traditional, process-oriented staff	Right sort of people on staff (innovative culture)
Low level of direct customer participation	High level of direct customer participation

Figure 4: Predictive and Adaptive matrix



Putting into practice

The Library IT Services unit consists of three full-time staff and five student assistants, and manages all technology and computing resources as well as keep abreast of emerging technologies and develop digital initiatives. When this author first joined the library, one of the IT staff doubled up as a programmer and handled most programming needs. Given the expanded scope of the unit to develop and implement innovative applications to improve library service, that software development process was not a sustainable model. The situation was a major impetus to use a different software development and project management approach especially for innovative IT applications.

Several weeks were spent in research and developing an appropriate action plan for our environment. Although the Agile approach and Crystal Clear method allows flexibility and agility based on “one size does not fit all”, we kept mainly to their principles. However, we did apply our experience of local conditions and best practices that would complement Agile, and we used project management tools that would work better for us.

We also found that these principles were very much in line with our view of the applications development process, and had little difficulty or hesitation about putting them into practice. We departed from the less interactive predictive models as well as reduced the need for numerous committee meetings. With the seven properties of Crystal described earlier, we found the first three – frequent delivery, continuous feedback, and constant communication - were key success factors, and the last four helped to create a good working environment.

Co-location, where team members are situated in the same space or facility, is a key element that allows faster and direct communication. This author’s original office often felt like Grand Central Station, with students working on projects as well as staff needing to discuss work matters. With the initial successful project, we managed to convince the Library Director to move us to a slightly larger office that would accommodate two more workstations for the team. Co-location also promoted the sense of community and helped shape the safety zone where the team could feel comfortable and confident to participate openly in the project planning and implementation process.

The people factor

Highsmith (2004, p16) defined agility as "the ability to both create and respond to change in order to profit in a turbulent business environment ... the ability to balance flexibility and stability." In the development of innovative applications, too much structure stifles creativity, while too little breeds inefficiency. This agility requires people who are creative and nimble improvisers who can deal with ambiguity – the ability to think outside the box and work within it.

As this was a new process for us, this author had to wear a variety of hats. The projects were mainly organised with this author as observer and main stakeholder, to implement and observe the use of the Agile approach as well as being directly involved in project development. As project lead and manager, this author had to provide the vision and technical skills and plan and coordinate the implementation. A large part of this role is to be an enabler and facilitator, so that team members can perform at their best. This author also tried to develop and foster an adaptive culture that is comfortable with change as well as highly collaborative and interactive.

Faced with limited staff and recognising the mutual benefit of working with students, this author recruited student assistants to be part of the prototype Agile-Team (A-Team). This resulted in a diverse group with different skill sets and levels of competence but they also brought in much enthusiasm and a “can do” spirit. Some library staff were stakeholders in different projects and they obtained some exposure to the Agile process as active developer and stakeholder collaboration and communication is encouraged. Stakeholders and users should also be readily available for feedback and clarification in the iterative development process, but this was not often possible and had mixed staff reaction.

The A-Team had a variable number of members depending on the project. Each would work on a different aspect of the application, and when they were in the same place in the iterative process they were able to collaborate and check that different elements of the resulting feature and application worked well together. We were thus also able to ensure that the application’s programming code did not limit the interface design or that design dictated the programming. With direct communication with stakeholders, the developers were also able to quickly rethink and fine tune features.

Following the principle of reflective analysis and improvement, the team would take time after each major iteration and project to debrief and discuss how to improve team effectiveness. As the approach became more widely understood and accepted, team spirit and confidence grew and the students also began to demonstrate leadership qualities.

Project management tools

A wide variety of project management applications are currently available, ranging from Microsoft Project to in-house developed spreadsheets for project planning and tracking. The Gantt chart is also one of the most familiar project management tools. In our case, we did not adopt any of these as we felt it would be of little or no value

especially with our type of innovative and experimental projects. We have milestones and deadlines but the comparatively small size of our projects and speed of development would make it difficult to keep the chart current. The 2007 Agile Adoption Survey (Ambler n.d.) also showed that Gantt charts are the least valuable work product on Agile projects whereas iteration task lists were one of the most valuable.

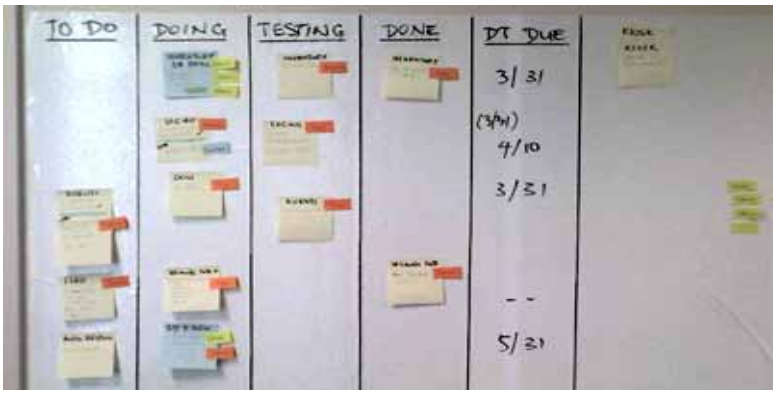
This author is a proponent of visual communication tools which often presents data more effectively than lists and bullet points. Our tools included a task board to display project requirements and status, dry erase sheets for notes and drawings, and a mind map to broadly visualise projects and scope. With these, the developers and stakeholders could see at a glance all the necessary information and added to the transparency of the development process.

Task boards

This tool is an important part of Agile practices and an indispensable reference point for the team. The board displays the progress and status of the project and can be as simple or as complex as the team and project requires. They can be "home-made" (whiteboard or wall with Post-it notes or corkboard with note cards and pins), and task board applications, both commercial and free, are also available online.

Our home-made version had columns headed To Do, Doing, Testing, and Date Due. Post-it notes are used for writing down project features and core requirements. Smaller notes are used for the names of team members and these are attached to the feature note(s) they are responsible for. The board thus provides a snapshot of the project, and is highly visible and accessible by all. Over time, the area around the board also became a focal point for informal stand up meetings and review sessions.

Figure 5: Task board of project requirements and progress



Dry erase sheets

A dry erase board or whiteboard is useful for drawing ideas and making notes. Due to the size limitations of a regular whiteboard, this author used static-cling dry erase sheets instead. The length of a wall in this author's office was turned into a large whiteboard by arranging two long rows of sheets. Each workstation also had its own sheet on the facing wall for making notes specific to the work on that workstation. Dry erase marker pens are used for writing, and "writing on the wall" is a popular and necessary activity for the team.

Figure 6: Dry erase cling-on sheets on office wall

Mind maps

This author used mind maps (also known as content maps) to supplement the task board for displaying broad project level information. A mind map is a diagram used to represent words, ideas, tasks, or other items linked to and arranged around a central key word or idea (Wikipedia 2009b). Relationships with the central topic are denoted by radial hierarchies and tree structures. Each project is a node in the map that can be expanded to show details, and displayed or printed out separately for reference.

The mind map allowed us to visualise our projects and their status, and organise and plan activities. We could also brainstorm ideas and features with no particular order of importance as the relationships are non-linear. One of the students referred to it as our "visual brain dump".

A comprehensive list of mind mapping software is available at http://www.mind-mapping.org/ including some that are free or in open source, for example FreeMind (http://freemind.sourceforge.net/), MindMeister (http://www.mindmeister.com/), and VUE (http://vue.tufts.edu/).

Figure 7: Screenshot of an A-Team mind map

Projects

As this author began planning the implementation of the Agile approach, we also drew up a list of potential projects and prioritised them. The project scope, features and requirements formed the iterations in the development process. These iterations were kept short, about one to two weeks, and were a measure of our progress in developing the applications.

The first two to three projects enabled us to settle into the approach and fine-tune where necessary. From them, we developed an awareness of the Agile development process and team dynamics, and the reflective analysis sessions provided insights and suggestions for improvement. This author was subsequently able to have different students and staff working on new projects and following and adapting the Agile approach.

Figure 8: List of projects (Sep 2008 – Jun 2009)

Project	A-Team	Start	Approx*
PR materials (brochure, flyer)	MC, SS, *SK*	Sep	2 wks
LS website (+ 3 Flash apps)	MC, *KM, SK*	Oct	6 wks
Dashboard (applets #1, #2)	MC, KH, *KM, TS*	Nov	5 wks
QRCodes	MC, *PT*	Nov	2 wks
Wiimote IWB (phase 1)	MC, *KM*	Dec	2 wk
Dashboard (applet #3)	MC, KH, *KM, CS*	Jan	2 wks
Screencasting (tutorial)	MC, SG, *DJ, NS*	Jan	5 wks
PR materials (bookmarks, images, icons, merchandise branding, etc)	MC, LW, SS, *SK, TS*	Jan	3 wks
Dashboard (applet #4, skin)	MC, KH, *KM, TS*	Feb	6 wks
Immersive Environments	MC, *KM, CG, RD*	Apr	5 wks
Wiimote IWB (phase 2)	MC, LITS, *KM*	May	3 wks
Concept Mapping (VUE)	MC, LA, *CG*	Jun	2 wks

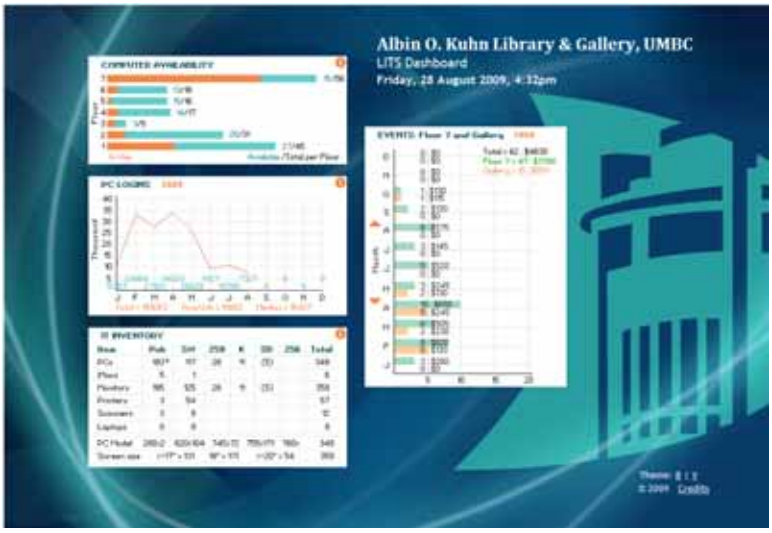
* ≈10-15 hours/week/student (*initials in italics*)

The first major project was the development of the Learning Spaces microsite to showcase the variety of learning spaces available in the library and to promote support for a proposed Student Learning Centre. This author was the main stakeholder and charged the team to design an interactive multimedia experience that would engage our audience, and avoid the ubiquitous thumbnail-and-text style found on most websites. Adobe Flash was selected and although we had only basic knowledge of the software, the awareness of its potential and the team's confidence and eagerness to learn carried us through (http://aok.lib.umbc.edu/spaces/). The site has three Flash-based applications, and programming and interface design played equally important roles in the outcome. This was a parallel development process within the larger microsite development.

The team had three members – this author was the main stakeholder and did web development, a student programmer worked on the Flash Action Scripts and a graphic design student on the interface. Our iterative process was like a three-step dance, we would work on our own tasks and then took turns to work with each other on them to ensure consistency in both programming and interface design. All through the process, we were learning, applying, and building up the features, application and site.

Armed with our expanded knowledge of Adobe Flash, we embarked on a digital dashboard application. This author wanted a simple visual way to review key management statistics, instead of referring to database tables and stacks of printouts. The Agile approach was also used but the team composition changed with a different graphic design student and included the staff programmer that developed the databases on which this application would draw data. Here, the application was more programming intensive and ahead of interface design in the iteration process. This author was thus able to observe the Agile approach in action in a different context.

Figure 9: Screenshot of digital dashboard for Library IT management



As we worked through the list of projects over a period of about ten months, we had different combinations of staff and students depending on skills required. Sometimes we had two to three projects going on at the same time due to need or circumstance, and this author would then be the "meta project manager" and juggled them to ensure steady progress. The continuous feedback, communication and co-location advocated in the Crystal Clear method were a tremendous help in doing this. We gained confidence and experience and could easily adapt the approach to different project needs.

The delivery of working software is the true measure of success in the Agile approach, and we were pleased that several of the projects also attracted much interest from beyond the library, with some having been implemented in other units and organisations. These include the Wiimote interactive whiteboard (both the portable version and ceiling-mounted in the active learning room – http://aok.lib.umbc.edu/ spaces/activelearningrm.php), Flash-based applications, immersive environments to model library and gallery spaces, and concept mapping.

Challenges

An early challenge was trying to find the right students for the projects and to participate in the Agile approach. Through the period of this study, five out of eight students thrived in the environment while the others wanted more structure and direction and did not stay on after their project was completed. Those who stayed were eager to learn, and enjoyed the gratification of successful and better features after every iteration in the development process.

As the students are based in this author's office, almost daily interactions were possible, unlike with other staff members who were also stake-holders on the team. They were keen and cooperative, but time and ready availability were issues for some. One of them preferred scheduled meeting times as a group rather than informal face-to-face conversation with the developers.

Initially, the principle of welcoming changing requirements even late in the development process, led to the perception of a lack of an orderly design process as features were sometimes modified in mid-stream. We also had to draw the line with changing requirements based on available resources and time. We would then propose a beta rollout or implement a phase two later.

Co-location is highly recommended for small teams, but with the sharing of office space, this author had to give up some privacy and personal space. However, the students preferred the open arrangement to working in cubicles as they could sit in their chair and roll over to the person at the next workstation or over to this author's desk for discussions. To create space and enable a better use of time, schedules were set so that the students reported for work on the same three to four days a week and at about the same time, so that this author had clear open times for other work and administrative duties.

Summary

The Agile approach was a new and untested venture for us but the team kept an open mind and remained patient and flexible throughout the process. Despite the challenges, our overall experience has been positive and we plan to continue using the approach for new projects. With the speed of innovation and availability of potentially useful applications, we found that this lightweight and adaptive software development and project management approach enabled us to undertake rapid development and deployment of quality applications with a small team.

The Crystal method, which is based on project size and criticality, helped us manage the variety of IT innovations and their implementation. As every IT application and project has unique characteristics, the method provided a framework of policies, practices, and processes that we could apply. Like horses for courses, we adopted the core principles and adapted the process to our projects and local needs.

We also fully subscribed to Highsmith's principles of Agile project management: customer value through innovative products and a leadership-collaboration management style. Our success can be attributed to the developers and stakeholders on our A-Team and the range of innovative projects that held strong interest and learning value. Unlike traditional project management approaches, the iterative development and feedback loop also allowed us to develop the applications and features incrementally, and resulted in a dynamic working environment with good collaboration and communication.

In moving forward, our outlook can be summed up in the words of Highsmith (2004, p253), "Agilists want to build innovative products – products that test the limits of our abilities as individuals and teams – and create a work environment in which people, as individuals and in teams, can thrive."

References

Ambler, S n.d., *Agile project planning tips*. Available from: <http://www.ambysoft.com/essays/agileProjectPlanning.html>. [2 August, 2009].

Beck, K, et al 2001, *Manifesto for Agile Software Development*. Available from: <http://agilemanifesto.org/>. [2 August, 2009].

Cockburn, A 2005a, *Crystal Clear: a human-powered methodology for small teams*, Addison-Wesley, Boston, MA.

Cockburn, A 2005b, *The Crystal Family of Methodologies for Software Development*. Available from: <http://alistair.cockburn.us/Media%3acrystalfamily2005.06-060.ppt>. [2 August, 2009].

Cockburn, A 2008, *Crystal methodologies*. Available from: <http://alistair.cockburn.us/Crystal+methodologies>. [2 August, 2009].

Coffin, R & Lane, D 2007, *A Practical Guide To 7 Agile Methodologies, Part 1*. Available from: <http://www.devx.com/architect/Article/32761/1954>. [2 August, 2009].

Coffin, R & Lane, D 2007, *A Practical Guide To 7 Agile Methodologies, Part 2*. Available from: <http://www.devx.com/architect/Article/32836/1954?pf=true>. [2 August, 2009].

Highsmith, J 2002, *Agile software development ecosystems*, Addison-Wesley, Boston, MA.

Highsmith, J 2004, *Agile project management: creating innovative products*, Addison-Wesley, Boston, MA.

Wikipedia 2009a, *Agile software development*. Available from: <http://en.wikipedia.org/wiki/Agile_software_development >. [2 August, 2009].

Wikipedia 2009b, *Mind map*. Available from: <http://en.wikipedia.org/wiki/Mind_map>. [2 August, 2009].

Nicolette, D 2005, *When to be Agile*. Available from: <http://www.davenicolette.net/articles/when_2b_agile.html >. [2 August, 2009].

Ward, C & Legorreta, L 2009. *Beyond waterfall and agile methods: towards a new contingency model for IT project management*. Available from: <http://ssrn.com/abstract=1400254>. [2 August, 2009].