Increasing Software Quality with Agile Experiences in a Non-Technically-Focused Organization

Aaron B. Hockley
aaron.b.hockley@multco.us

Abstract

Agile development methodologies are well-documented but most of the textbook examples and anecdotes found on the internet provide stories of the use of agile methodologies as part of a technically-focused organization such as a software development contractor, retail software company, or hardware/systems organization with software teams that support the company's technical products.

Multnomah County (Oregon) is not such an organization. The county's business consists of providing services such as health services, jails, taxation, licensing, animal control, and managing individuals on parole and probation. A few small software teams build and support tools to support the county's business, but the organization is decidedly focused on non-technical ventures. In a non-technically-focused organization, the role of the product owner (internal customer) becomes challenging. Software development participation competes with their other (usual) job activities and the product owners are often unfamiliar with software development experiences.

Over the past three years, a variety of agile practices have been introduced at Multnomah County; lessons learned by the county's Public Safety Development Team have resulted in software that better meets the customer's needs. Experience has shown that quality improves when the development team has frequent access to business personnel even though the ideal co-located customer scenario cannot be achieved. Communication with the business product owners is key; the county's software teams tried a variety of agile work tracking and communications systems before finding one that works well for all parties. No tool is perfect. The team concluded that given the challenges of customer time and participation, the tool which provides the best customer experience is probably the best tool.

Experience demonstrated that it wasn't feasible to use a textbook Scrum approach due to customer availability challenges. Project teams and customers eventually settled into a development process that's a hybrid between a Scrum approach and a Kanban-style system. This nimble development cycle is working well for all involved parties.

Better software means that the core business services are better met. Software that best supports the staff providing day-to-day county services is important; a nimble development approach enables applications to be created that best meet the staff and resident needs.

Biography

Aaron Hockley is a QA development analyst with the Public Safety software team at Multnomah County in Portland, Oregon. He is actively involved in both day-to-day software testing (both manual and automated) as well as process improvement for software development in the public safety business areas. Over the past ten years, he has worked for a variety of software and technology companies with a focus on software development and testing.

1 Introduction

When we hear or read stories and textbooks of agile software development, the information is often presented from the idealized position that involves a software development team working for a software company. A product manager usually plays the clearly defined role of product owner; the entire organization understands software development and things move forward with only the agile manifesto (Beck et al. 2001) and market forces as guidance. In such an environment it's easy to look at the bottom line and market share analysis to measure a company's position.

The Multnomah County government, based in Portland Oregon, is not such an organization.

1.1 About Multnomah County's Software Development Situation

The business goals of the county are to provide services for its citizens. Instead of an organization that focuses on delivering technology, the county is ultimately tasked with managing jails, public health, libraries, prosecuting criminals, and other miscellaneous services such as animal control. While software and technology solutions are used by county employees (and citizens) in the delivery of these services, the county's business is government as opposed to technology.

The work of the county is performed by well over 4,500 employees; only 40 of those (less than 1%) are software developers. With the development staff representing such a tiny portion of the overall workforce, effective software development is barely a blip on the overall radar of decision makers. A change in library operating hours will consume far more public attention and staff time than whether the county's software engineers are using Scrum, XP, waterfall, RUP, or any other development methodology or toolset.

Development personnel are organized into four teams based on business area. The Public Safety Development Team is the largest group and provides support for the Sheriff's Office (jail management) as well as the Department of Community Justice (parole, probation, and juvenile offenders). While a few off-the-shelf applications are used for some areas of the business, most of the software used to manage offender caseloads, adult and juvenile detention (jail), and treatment and counseling programs are custom application development performed by the county staff on the Public Safety Development Team. Over the past three years, the Public Safety group has become increasingly agile and has found a nimble solution for better software to help the county meet its public service goals.

1.2 Past Challenges with Waterfall Development

In 2008, development staff and management broached the idea of a change to a more agile development process. Existing applications were built using a traditional waterfall-style method. The last large project that was run in such a manner spanned over two years with heavy up-front requirements and design followed by a period of development. After development, customers were presented with the "finished" result. Because over a year had elapsed from the time of initial requirements gathering, business processes had changed. Decisions made by the software development team during construction weren't always correct due to differing interpretations of the requirements documents. After presenting the software to the business group, several more months of work were involved in making changes and updates that were needed due to changing requirements and processes. In short, the software was suffering from most of the usual failures of a waterfall approach. (Leffingwell 2007)

Using the Public Safety Development Team as an example, there are several lessons learned in how to best implement an agile development process in a non-technical organization such as a county government.

2 Software in a Non-Technical Organization

In a technically-focused organization such as a software development company or another technical company where software plays an integral role in the company's business model, the role of software is well-understood and well-supported as a factor in the business' success. At Multnomah County, software plays a background role to the county's services. For a software development team working in this environment, these differences are seen in the perception of software quality and difficulty of scheduling interaction between the business and the development staff even when such interaction is critical.

2.1 Defining Software Quality

When defining software quality, one can look at a couple broad categories. Some quality measures are absolute; these measures are often those of basic functionality. The question is "Is the software able to perform all of the functions it was designed to perform?" and the answer can be summed up as Yes or No. This level of quality is the one which is often obvious even to those outside of the industry. A second measure of quality is more relative. This form of quality includes things like the overall user experience, the ease of use, and how well the software assists the user with meeting business needs.

For external-facing software teams, this second indicator of quality can often be measured by market forces. If a team produces great applications, revenue will likely increase and customers will sing the company's praise. Buggy software or other missteps will be reflected in customer complaints across the internet and will likely result in slower sales. Customers generally have many options when choosing software and will generally select the software which provides the most functionality in a pleasant interface at a competitive price.

Internal customers at an organization such as a county government don't have such choices. At Multnomah County, if the juvenile counseling staff is unhappy with the software being produced by the Public Safety Development Team, they don't have easy options. Off-the-shelf software likely isn't an option (hence the decision to build a custom application in-house) and they can't simply find other developers to build something better. This environment requires a slightly different view to measure software quality.

Four indicators measure software quality at Multnomah County:

- Ease of Use
- Consistency with other applications
- Minimal defects post-implementation
- Perceived relationship between business staff and application development staff

Taken one at a time:

2.1.1 Ease of Use

This is simply the subjective measure of whether the business staff finds the software intuitive and is able to use the system to meet their business needs. County employees have a wide range of technical abilities; steps are taken to minimize the learning curve for users so that new employees can quickly come up to speed and training costs are minimized.

2.1.2 Consistency with Other Applications

In an ideal world, perhaps all of the needed information for a particular job would reside inside one application. The reality of working in the government is that for some function such as public safety that information is distributed amongst various organizations and departments. Depending on one's exact job duties, it's not uncommon for county staff to routinely use up to eight line of business applications and

databases to access information at the county, city, state, or national government levels. Because of the wide variety of systems used, development staff strives for consistency when building or maintaining systems so that business staff is able to easily jump between programs with a minimal amount of confusion around navigation, terminology, and data display formats.

2.1.3 Minimal Defects Post-Implementation

As is common with most development groups, the amount of defects discovered after implementation is used to gauge the quality of the development process. Many studies have shown that the cost of resolution for a post-implementation defect is significantly higher than a defect found earlier in the development cycle (Kaner, Bach, and Pettichord 2001). Defects found by customers after release also have the perception cost incurred when end users view that the software doesn't work as expected.

2.1.4 Perceived Relationship Between Business and Development Staff

This is an intangible measurement but one which has proven valuable. Because of the ongoing relationship between business users and development staff (and, as noted above, the model in which business staff is limited in application development options), there is a long-term benefit in maintaining a healthy and friendly relationship between the technical and business staff. County development staff found that perception often becomes reality; enjoying a positive working relationship with the business staff creates a positive working environment and leads to better communication (which in turn leads to better software).

2.2 Business Staff Availability for Software Development

In a technically-focused organization where software is a recognized product and/or source of revenue, organizational decisions will be made to support said software development. In an organization such as Multnomah County, organizational structure is focused around the services that the county delivers to its citizens, which isn't necessarily the ideal organizational structure to optimize internal services such as custom software development.

In the textbook version of an agile Scrum project, the software developers, customers, Scrum master, and all other stakeholders are colocated with staff sitting physically in the same location. The physical proximity makes ad hoc communication simple. One area of challenge is that the county's physical organization generally precludes having a customer (product owner) colocated with the software development team. The IT software development teams are located in a centralized county administration building whereas customers are spread out throughout the county in a location appropriate for the services provided. For the Public Safety Development team, customers are most often located at the county courthouse, jail, or juvenile services and detention facility.

Because the customer and developers are not physically colocated, it's vital that the development processes and tools facilitate easy and effective communication. For times when face to face discussion isn't feasible, the entire project team uses instant messaging for quick chats as well as the commenting and email features of Pivotal Tracker (a project management tool, discussed further below) to record decisions made about particular work items.

2.2.1 The Quest for Customer Involvement

For a member of the Public Safety development team, one's primary job function is to build and support software which helps the county conduct its business. For an internal customer of the Public Safety development team, one's primary job function has very little to do with software. These personnel are performing day to day activities that involve working with a parolee to develop job skills, assisting a juvenile offender with entry into drug counseling, or making a decision on whether a just-arrested defendant will be released on their own recognizance or held in jail until arraignment. Software supports their jobs, but software isn't their job.

The fact that software is not a primary job duty for the average county software user means that it can be challenging to develop engaged relationships between the software developers and the customers. As custom software projects begin, the reality is that the software development project will compete for the customer's time along with that individual's usual job duties. In an ideal world the customer would be able to shift usual duties onto other personnel in order to be able to fully participate with the software development team, but that isn't always practical or financially viable. With a software development project competing for a customer's time, there needs to be an explicit decision to obtain a customer's time commitment to the project.

Another challenge presented by the customer situation in a non-technically focused organization such as Multnomah County is that the customers are not necessarily savvy or skilled with current computer technologies. The county's workforce consists of individuals of all ages and skill sets, a majority of whom have working knowledge of how to use a computer for their basic job duties but with a small minority falling into the category of early adopters or those that will readily embrace new technologies. Outside of the software development teams, county staff is unfamiliar with software development practices and processes. In a software or technical organization, some familiarity with software development is part of the baseline set of knowledge. In a non-technical environment, a steeper learning curve is present for the business staff that participates in software development.

2.2.2 Customer Involvement as the Key Quality Indicator

Why point out the particular challenges with customers in a non-technically-focused organization? Because the development teams at Multnomah County have found **that the single largest indicator of the overall quality of software built in a custom development project is the involvement level of the business customer.** For projects with actively involved customers who have a decent level of technical understanding, the rate of overall project success and the quality level of the software have consistently been higher than for projects where customers have a limited level of involvement or where there are significant gaps in technical ability.

The involvement of the customer provides benefit in a few ways. With the perception of the process being a significant factor in the overall quality of the project, active customer involvement is important because the customer will feel as if they're involved in the project. Attending key project meetings (stand-ups, iteration kickoff meetings, and backlog prioritization) provides the customer with the ability for real-time input into the development process. Customers should feel that they can contact the software development staff at any point they feel it's necessary to provide feedback. From the software developers' perspective, having easy access to business staff is essential to remain on a nimble development cycle. In an agile process, working software is favored over extensive documentation. This lack of documentation occasionally results in the development staff having questions as the software is constructed. The ability for developers to access customers for rapid feedback is a key requirement for developers to move forward in a nimble fashion.

In an ideal world, development teams could pick and choose the customers of their choice. While it's not realistic to only work with customers with a high level of technical ability, the involvement and time commitment is a variable which can be controlled. On every project, the overall perception of product and process quality increased directly proportional to the amount of time involvement on the part of the business customers. When management allows for staff to spend larger quantities of time working with the software development team, the projects operate in a smoother fashion and users are more satisfied with the resulting software. Therefore it is of key importance for the IT and software development staff to convince the business' upper management of the value in dedicating business employees' time to software development projects.

3 Agile Tools: Lessons of Visibility

In moving to a more agile style of development, it quickly became obvious that new tools would be needed to facilitate agile project management and communication amongst the various project team members including the technical development staff as well as the business customers. There is no shortage of tools and processes on the market designed to make agile software development easier for an organization. Much like iterating through software features, the Public Safety Development Team at Multnomah County has used a variety of tools to support their software development, with three tools being used for extended periods of time.

No tool is perfect; each of these systems has benefits and drawbacks.

3.1 Post-It Notes on a Wall

The first tool used at Multnomah County was the most analog and had the lowest cost of entry. The large backlog of pending work was managed via a spreadsheet; at the iteration kickoff meeting these work items were transcribed onto Post-It notes. The notes were then stuck to a wall based on their position in the current development cycle. All notes would initially be in an "Unstarted Work" column, with notes being moved across to subsequent columns as the work items entered construction, then QA testing, then acceptance testing, with each work item eventually ending in an "Accepted" bucket.

Positive Results from Post-It Notes:

- The notes on a wall were very visible to the members of the development staff that sat nearby
- The act of moving a tangible item from column to column gave a sense of accomplishment
- The system required no software or other technical complexity; Post-It notes are cheap

Negative Results from Post-It Notes:

- The notes on a wall were invisible to the portion of development staff that did not sit nearby
- The notes on a wall were invisible to customers (which did not sit nearby)
- There is no easy way to archive and search the completed work items

3.2 Microsoft Team Foundation Server

With most of the group's development happening in a Microsoft .NET environment (both ASP.NET and Silverlight), Microsoft's current development tools server was a logical fit. Team Foundation Server (TFS) offers source control, a build server, work item tracking, reporting, and other tools designed to support a software development team. The TFS software runs on a local server and exposes various functions via services. Access to source control, work item tracking, build management, and reporting is obtained using client applications which communicate with the TFS services. For most developers, client access is provided using Microsoft's Visual Studio Team System Integrated Development Environment (IDE). The IDE allows for source code management and work item tracking within the same development environment used to write and test code.

For access without using Microsoft's development environment, limited functionality is provided via a web interface built using Microsoft's SharePoint functionality. Unfortunately the default views provided as part of the product did not provide meaningful "at a glance" views for business staff that allowed for sufficient visibility as to the state of the project, the work items, and a quick view as to which version of the software was ready for user acceptance testing.

This tool was in place at Multnomah County for about 18 months (two major versions of TFS). The source control and build server were used exclusively by the developers without any sort of customer interaction; from a process and quality assurance standpoint the work item tracking features were of

greatest importance. The work item features were very granular – the default forms used for recording features, bugs, and other work items featured dozens of fields. Although there were high hopes for the work item tracking abilities of TFS, the team found out that less is more; the number of fields on the screens created a barrier to entry which prevented customers from utilizing the tool. The work item tracking abilities of TFS proved to be a bit too heavy.

TFS offers the ability for customization by writing code to communicate with the server's services, but county management did not feel that spending extensive developer resources to maintain a custom toolset was a wise use of limited financial resources.

Positive Results from Team Foundation Server:

- Work item visibility via a web browser or within the IDE for developers
- Very granular, detailed work item records with many customizable fields
- Ability to record large quantities of project management data points

Negative Results from Team Foundation Server:

- Default work item views not customer-friendly; required extensive customization
- Relatively expensive server and client licensing requirements
- Requirement to maintain and upgrade web, database, and reporting servers
- Hard to evaluate and report on team strengths and weaknesses

3.3 Pivotal Tracker

Due mostly to the limitations in customer visibility of TFS work items (and ongoing licensing costs), the Pivotal Tracker tool was introduced as an alternative to TFS on an experimental basis for one project. It worked out well and is now being used across the entire development team (including customer and business analyst access).

Pivotal Tracker is a web-based, hosted work item tracking system. The vendor describes it as:

> *…a story-based project planning tool that allows teams to collaborate and react to real-world changes. It's based on agile software methods, but can be used on a wide range of projects. Tracker maintains a prioritized backlog of project deliverables, broken down into small, estimated pieces, called stories. It dynamically groups these stories into fixed segments of time, called iterations, and it predicts progress based on real historical performance (velocity).* (Pivotal Labs 2011)

Whereas TFS provided an overabundance of options for entering work items, Pivotal Tracker provides a very minimal set of fields and features. In contrast to TFS's multiple pages of dropdowns and text fields, Pivotal Tracker offers only eight fields for a work item (with half being optional). Although not as granular as TFS, the simplified interface makes it easy for developers and customers to view the important information about a work item. While the number of fields is limited, key information is captured and made visible to all. The signal-to-noise ratio is high which makes it easy to identify the current status of pending, in-progress, and completed work.

Positive Results from Pivotal Tracker:

- Good visibility to work items for customer, development staff, and others
- More affordable licensing than TFS
- Pre-built reporting for burndown and velocity is easy to use and understand
- Easy to understand "Velocity" metric which measures the team's speed at completing work

Negative Results from Pivotal Tracker:

- Custom reporting is difficult

One caution with agile development and choosing a toolset is that because agile favors communication and collaboration over documentation, the effectiveness of a given tool will often depend on the personalities and communication styles of the members of the team. In the Multnomah County scenario, Pivotal Tracker has proven to be an effective tool for tracking work items for the development staff while providing transparency for the customers. While customers and developers have differing needs from a work item tracking system, Pivotal Tracker is an ideal compromise.

One team member notes:

"TFS is a far superior project management tool because of the breadth of information it can capture. Pivotal Tracker is the best 'customer expectation management' tool." (Chapel 2011)

As noted previously, one of the quality measures is the perception of the relationship between business staff and the software development team. Managing customer expectations is a key factor in managing that customer relationship.

4 Our Nimble Solution

Given the challenges described previously and the lessons learned about tools and communications, the Public Safety Development group at Multnomah County has found a development methodology which is based on the agile manifesto but doesn't adhere strictly to the textbook definition of a particular method such as Scrum or XP. What follows is a description of this nimble development process that is working well for the technical development staff as well as the (mostly) non-technical business staff at Multnomah County.

4.1 A Prioritized Queue of Pending Work

All pending work (features, bugs, changes) is entered into Pivotal Tracker as new work items. Everyone in the organization (developers, business analysts, project managers, customer stakeholders) is empowered to add new work items. This queue of pending work is reviewed on a weekly or semi-monthly basis at a backlog review meeting. In attendance are the project manager, business analyst, business product owner, and (sometimes) the lead developer. The business product owner is the decision maker and is tasked with prioritizing the pending work.

Because the backlog of pending work items could be dozens of items, it isn't practical (or a good use of time) to review each work item at every meeting. Instead, these meetings focus on reviewing and prioritizing two groups of work.

The first set of work to be reviewed is those *items at the top of the priority list* which will become the next items to begin construction. These items are reviewed to ensure that they are well defined (the software developers have enough information to begin work) and that they are correctly prioritized.

The secondary group of items is those *recently added to the backlog* which hasn't yet been prioritized. This group will usually be a mix of bugs which have been discovered as well as new features that have come directly from the business staff or have become evident during construction and testing. Reviewing these items regularly ensures that high-priority work is properly placed at the front of the queue.

4.2 Every Other Week: Iteration Review & Kickoff

Although development doesn't wrap up in the same fashion as a Scrum sprint (more below), every two weeks the team gets together for an Iteration Review and Kickoff meeting. Everyone is in attendance: the

project manager, business analyst, business product owner, developers, and quality assurance staff. Sometimes the end user line staff is brought into the mix as well.

At this meeting, the team performs three activities. First is a retrospective of the previous iteration. Everyone is free to contribute notes about what went well along with any challenges encountered. The project manager may take away some tasks related to the challenges noted. After reviewing the previous work, the team reviews the work items at the top of the prioritized backlog. Each work item is reviewed to ensure that it is well-defined. The development and testing staff are making sure they have the information needed to build and test the work. If questions arise, the business staff and analyst may be able to help flesh out definition and resolve any issues so that the developers can begin construction. Acceptance criteria are identified. If discussion determines that an item isn't able to be quickly defined, the work item is moved lower on the priority list and will likely become part of a subsequent iteration of work.

The other activity at the kickoff meeting is noting the team strength for the coming iteration. Holidays, vacation time, or other activities that reduce staff availability must be included as a factor in order for Pivotal Tracker to establish a more accurate velocity for the team. Team strength can also be increased if there are additional resources available beyond the usual or baseline level for the project. Recording accurate team strength leads to a more accurate velocity, which in turns leads to a better ability for project managers to estimate completion of a set of work.

4.3 Allow Work to Extend Beyond Strict Two-Week Sprints

The usual definition for a sprint is that it's a fixed period of time in which a development team will commit to, build, deliver, test, and gain customer acceptance for a specific set of work items. While this scenario might work for a software company or other organization that has a focused and dedicated product owner, it hasn't proven practical for the situations described previously where a software development project is competing for a non-technical customer's time and attention. The beginning of the sprint starts as expected with everyone participating in a kickoff meeting followed by the engineering group beginning construction. As work items are completed, attempts were made to have customers test, provide feedback, and ultimately accept the work items throughout the iteration. This constant demand for customer attention was often met with failure because the business individuals did not have time to dedicate to the project on a daily basis. It proved to be near-impossible to have work items accepted on a continuous basis with all items accepted prior to the end of the iteration. Instead, work items flow into subsequent iterations if they are not completed.

4.4 Automated and Frequent Deployment Infrastructure

In order to gain rapid feedback both within the software development team as well as with business staff, easy and frequent deployment ability is crucial. The Public Safety Development Team has an automated, repeatable deployment strategy that makes it very easy to deploy the application to various development, testing, and production environments. While explanation of the entire deployment process is beyond the scope of this paper, a summary is provided for context since ease of frequent deployment has shown to be an important part of the software development strategy at Multnomah County.

4.4.1 Application Build Process

All source code is stored in a Subversion repository. TeamCity is used as the build server. A continuous integration (CI) build is automatically triggered on each source code commit in order to provide immediate feedback to the development staff if a problem exists. An integration build brings together all of the application bits, compiles them, and generates an installer (.exe) for the web application and any related services. The build also packages the database scripts to create or modify related database tables, procedures, views, and security configuration. The database scripts are zipped into another executable file.

4.4.2 Application Deployment Process

In the development and quality assurance environments, applications can be deployed with one click via the TeamCity build server. After packaging an integration build, the application is installed remotely onto a web server. The database is refreshed by starting either with an empty database (in the case of a new application) or a restored copy of the current production database (in the case of maintenance or other software updates). Database scripts are then applied via an automated job such that there is no opportunity for a flawed deployment due to human error during the deployment process. This automated deployment also helps ensure that as a set of code is promoted through the various environments (development, QA, user acceptance, and finally production) that the code is intact and is not changed as it moves from environment to environment. Basing the database refresh and deployment process on actual production data ensures that real-world scenarios are exercised and not merely dummy or test situations.

4.5 Daily Deployments to a QA Environment

Throughout the development cycle, code is deployed and tested in a QA environment on a daily basis. Rapid feedback to the developers allows for nimble development and ensures that potential problems are identified and resolved as soon as possible.

4.6 Regular Deployments to a User Acceptance Environment

Given that one of the identified keys to success for a project is the involvement of customer feedback throughout the process, regular deployments to an environment for customer acceptance are important. The exact deployment schedule is identified for a given project based on the availability and circumstances surrounding the particular customers involved, but a weekly deployment is seen as standard. After QA testing, a build is moved into the User Acceptance Testing (UAT) environment. The QA staff will notify the business staff that the software is ready for their hands-on acceptance. Users are expected to test the application, specifically looking at business functionality and the acceptance criteria which were identified at the iteration kickoff meeting.

4.6.1 User Feedback: Acceptance of Work

User feedback often occurs informally via conversations with the development team, but formal acceptance is recorded by the business staff marking a particular work item as *Accepted* in Pivotal Tracker. This indicates that the item meets the acceptance criteria that were identified and that the business believes the work item is behaving as designed.

4.6.2 User Feedback: Rejection of Changing of Work

If testing discovers problems, the business staff provides feedback to the development team. One of the challenges and training bits that happens with business customers that are new to software development surrounds identifying those issues which are bugs versus those which are changes or new features to be added.

If a work item is *broken*, that is, it isn't working as specified, then the item is marked as *Rejected* and the development staff makes changes to address the problem.

If a work item is *working as designed* but the business staff realizes that it needs changes (the classic "You built what I asked for, but not what I wanted" scenario), the item is *Accepted* and a new work item is created to reflect the changes (additional work) desired. This helps ensure that the velocity calculated by Pivotal Tracker includes credit for the original work that was requested as well as the changes.

4.7 Continuous Feedback

Although there are defined user acceptance cycles and methods, constant communication between the business staff and software developers has shown to be the most effective way for improving the software quality. This communication should flow both ways; while development staff will articulate the desire to obtain rapid feedback from the business customer, the development staff should also be providing information (and software) to the customers on a frequent basis. While tools such as Pivotal Tracker help facilitate said communication, the importance of ad hoc conversations (face to face, instant messaging, or email) cannot be discounted.

5 Nimble Development Experiences

5.1 An Estimation Solution

Software estimation is hard. Unlike assembly-line manufacturing processes, custom software development is just that: custom. Each project can often involve new business processes, new technologies, or new team dynamics. Waterfall estimation was generally a failure because in the drawn-out development process, changes of all varieties would introduce additional scope, risk, and unforeseen delays.

Despite the fact that estimation is hard and often unreliable, executive staff and those with budget concerns always desire accurate estimation.

Agile development doesn't change the difficulty of estimation. Stories are still given a size or point value. The difference is that instead of using the estimation to preemptively predict when a project will reach completion; those point values are used in a more retrospective fashion to identify the speed of the project.

Over the past three years, the development teams have settled into a project estimation and scheduling pattern that seems to both meet the desire to move in an agile fashion as well as being able to provide some rough estimates that can be used for big-picture budgeting and scheduling purposes. For a given project, the estimation and scheduling process now works as such:

1. A Business Systems Analyst (BSA) works with the business staff to gather some initial user stories and requirements. These are high-level functions desired for the new or enhanced application and would include things such as identifying the various actors, recording workflows that would become part of the application, noting any external application or data interfaces that would be needed, and identification of any potential data conversion efforts.
2. The Business Systems Analyst consults with a few members of the software development team. Based on the information gathered by the BSA and the knowledge of the software developers, a very high-level, low-confidence, low-precision estimate is made. Examples of such estimates might be "Based on other projects that seem to have similar scope, this appears to be a 6-9 month project" or "With the information we have and the unanswered questions, this seems like it would be a 12-18 month project."
3. This high level estimate is presented to the business and budgetary decision makers. Based on the information available, a decision is made whether to begin the project. There is an understanding that the initial broad estimate is just that: broad and low-confidence.
4. Development begins. Initial stories are broken down into more details so that they can be scored by the development team. Work is prioritized and the application is constructed.
5. After four or five iterations, the team's velocity (based upon story scoring and the number of points completed per iteration) becomes evident. In the meantime, the Business Systems Analyst is leading the effort to define remaining work items. These work items are prioritized by the product owner and scored by the technical staff.

6. Once a velocity develops and a bulk of the remaining work items have been scored, a tentative completion date is obtained. Although this date is not exact, it will have more precision than the previous high-level estimate.
7. Budgetary and planning decisions are made based on the new date, keeping in mind that new work will develop as the project progresses.

When resources are fixed, software development project management can use one of two targets: a date or a feature set. If one targets a feature set, development simply continues until that feature set is complete. Targeting a date, as is usually done after a reasonably accurate date has been identified, implies that project management becomes scope management. As work progresses and approaches the target date, decisions are made about which features will remain and which will be deferred to a future development project. The set of items which remains “in scope” is constantly shifting as the project team reacts to both planned work, bugs, and new work that is requested as the project progresses.

5.2 The Life of a Nimble Work Item

Software of the quality indicators, tools, and processes previously described, this describes the lifecycle of a work item in the nimble development process being used at Multnomah County.

1. Potential work items are prioritized and estimated as described in the previous section.
2. As a work item moves toward the top of the list, the business analyst ensures that background information is obtained and that the work item is ready for work.
3. At the iteration kickoff meeting, the team reviews the work item; any disagreements or uncertainty about the meaning of the work item is resolved. Everyone agrees they understand the acceptance criteria for the work item.
4. The development staff “Starts” the item in Pivotal Tracker and constructs what is needed for the work item. Developers work in communication with QA staff to ensure the design and construction meets the quality standards. When completed, the developer “Finishes” the item in Pivotal Tracker.
5. Seeing the work item is marked as Finished, QA staff deploys the relevant code to the QA Testing (QAT) environment and tests the unit of work. If rework is needed, QA staff will “Reject” the item in Pivotal Tracker and development staff will address the issues. If the work item meets the acceptance criteria, QA staff will “Deliver” the work item and promote the code to the User Acceptance Testing (UAT) environment.
6. Once a work item has been delivered, Pivotal Tracker shows it with a green “Accept” button and a Red “Reject” button. The Business Systems Analyst works with the business customer to perform user acceptance testing. Assuming things are satisfactory, the work item is marked as Accepted. If there are problems, the work item is Rejected and developers attend to the defect(s).
7. Accepted work shows in the “Done” category of Pivotal Tracker and factor’s into the team’s velocity for planning purposes based on completed work.

6 Conclusions

In a traditional software development company or other technically-focused organization, the systemic understanding of software development processes means that most of the widely-known agile development methodologies can be implemented as prescribed. A non-technical organization (such as Multnomah County) will be unable to implement the "textbook" versions of agile development, primarily due to the non-technical nature of the business customers and the challenges of obtaining active participation from customers who must juggle their software development role along with their usual business duties.

A nimble system based on an iterative development approach has proven to be successful by clearly defining the customer's role, expected meeting attendance, and time commitment for feature definition

and acceptance testing. This process has proven to increase software quality and result in a more favorable view of the software development team and process by the business staff.

References

Beck, Kent. Beedle, Mike. Bennekum, Arie van. Cockburn, Alistair. Cunningham, Ward. Fowler, Martin. Grenning, James. Highsmith, Jim. Hunt, Andrew. Jeffries, Ron. Kern, Jon. Marick, Brian. Martin, Robert C. Mellor, Steve. Schwaber, Ken. Sutherland, Jeff. Thomas, Dave. "Manifesto for Agile Software Development." http://agilemanifesto.org/

Chapel, Ed. 2011. "Why We Left Team Foundation Server" http://edchapel.tumblr.com/post/4180369020/why-we-left-team-foundation-server

Kaner, Cem. Bach, James. Pettichord, Bret. 2001. *Lessons Learned in Software Testing: A Context-Driven Approach*. New York: Wiley.

Leffingwell, Dean. 2007. *Scaling Software Agility: Best Practices for Large Enterprises*. New York: Addison-Wesley.

Pivotal Labs. 2011. "Frequently Asked Questions." https://www.pivotaltracker.com/help#whatispivotaltracker