LIBRARY LEADERSHIP & MANAGEMENT

Managing Library Software Development: A Case Study In Developing An Inventory Management System For Off-Site Storage Using An Outside Contractor

Aaron B. Bales, Robert Fox, Miranda R. VanNevel

Introduction

The past thirty years have seen a major paradigm shift in academic libraries. In 1986, Harvard University constructed its first off-site storage facility. Since that time, academic libraries with substantial collections have transferred millions of print volumes off-site in order to repurpose space utilization.[1] Reasons for this include the increasing lack of shelf space for new acquisitions, the trend toward the creation of more user oriented physical spaces that allow for study and collaboration, the proliferation of born-digital media including books and journals, and the increasing expectations of accessing materials online. Hesburgh Libraries, at the University of Notre Dame, was approved for a complete internal renovation including all patron accessible areas of the fourteen floor main library building. As a part of this renovation, the library decided to follow this trend by dramatically increasing the footprint of patron-oriented collaboration, study, and teaching spaces throughout the library. This necessitated the transfer of a substantial amount of materials to an off-site storage annex.

During the initial planning stages for the off-site annex, it was agreed that a system would be needed for stocking, tracking, requesting and delivering volumes housed at the annex location. This case study documents the rationale, phases of the project, outcomes, and related administrative concerns that the Hesburgh Libraries faced over the course of implementing an inventory management system (IMS) that could be used to facilitate the aforementioned requirements. In bringing this project to a successful conclusion, the IMS implementation team chose to follow management processes that were new to the Libraries, but necessary in order to move the project forward. For example, the library had not previously engaged in a formal project management process prior to active planning for the annex. Another novel approach was the consideration to hire a contractor to work on custom software instead of developing a solution in-house or purchasing a vended solution. The confluence of these methods required the team to carefully consider how the problem needed to be approached.

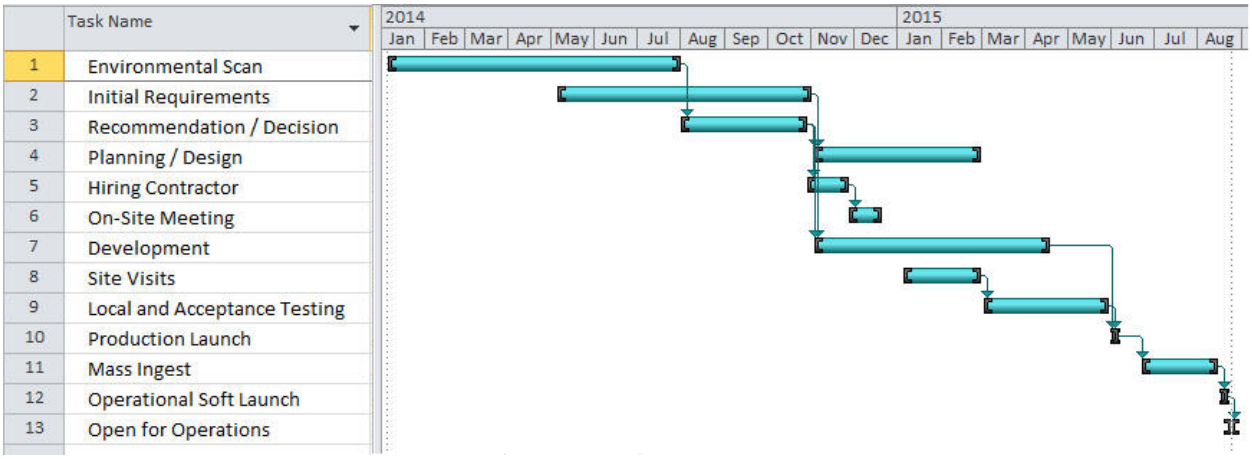
The various methods for how the team solved challenges over the course of the project is discussed in detail in this study. The nature of the problem, potential

solutions, and decisions are first discussed and analyzed. Following this, the method for establishing the appropriate roles for the project is discussed which includes an overview of the model that was used as well as the particular individuals that filled each role along with their duties and areas of accountability. The architecture of the application and the development process itself are then discussed in detail, outlining many of the challenges the team faced in working with a contract developer. The study concludes with a section discussing the lessons learned from this project, both the successful elements employed to bring the project to conclusion, as well as opportunities for improvement.

Problem Statement and Potential Solutions

In the spring of 2014 plans were set into action to begin a complete multi-phase renovation of the 14 floors of the Hesburgh Library. In order to accomplish this renovation, some collections needed to be moved from the building to create room for collaborative work environments and technology spaces. A committee was formed to establish criteria for items to be moved out of the main library building, with a goal to transfer approximately 1.5 million items to a new Annex facility. To accommodate immediate construction needs, the library needed to transfer a third of these by May 2015. For the overall timeline for the project see Figure 1.

Figure 1. Overview of Project Timeline



As an early part of the planning process, the Libraries conducted an environmental scan, contacting other academic libraries with off-site storage facilities. The conversations covered a variety of topics, such as materials, staffing and services, and included questions about Integrated Library Systems (ILS) and any IMS they might have. After discussions with seven academic libraries, three

options were considered for inventory management in the Annex: purchasing an IMS, tracking inventory in the ILS, or developing a custom IMS.

Four of the institutions consulted had opted to purchase an IMS. Three of these used the vended software product Library Archival System (LAS) developed by Generation Fifth Applications (GFA) for inventory management. The fourth used an IMS bundled with an Automated Storage and Retrieval System (a system using automated cranes to retrieve bins). That option was not applicable as Notre Dame did not plan that type of facility.

Although LAS was described as stable and reliable, the team felt the product did not sufficiently meet the project requirements. Because LAS does not interface directly with the ILS or Interlibrary Loan (ILL) Systems, library staff would need to re-enter patron requests. No identifying information other than the item barcode is provided at the point of retrieval for those requests. It also does not update the ILS in real-time, but instead relies on batch reports of stocked or shipped items produced at the end of the day. For these reasons, the team determined that LAS did not provide sufficient value over the second option, using the ILS, and it was not pursued further.

The Libraries also considered the possibility of software used for inventory management in other industries. Various options were reviewed, and Accellos Warehouse Management System was identified as a possibility. Accellos would ingest the barcodes and additional data (e.g., author, title, volume) about the items. The book trays would be treated as “cartons” and the shelves as “bins” in the inventory system. As items were scanned to the trays, and the trays to the shelves, the location information would be updated live in the system. Items removed for use would be temporarily associated with an in-use or out-of-annex bin. Size information could be recorded at point-of-retrieval in order to track available space.

Although this system was a closer match to the identified needs than the library-specific option, there were still some concerns. The system included a significant amount of irrelevant functionality. Also, it was not clear how difficult it would be to integrate with our ILS and ILL systems, which would still require local development. Additionally, purchased software required licensing and significant ongoing maintenance costs. License requirements included limits on concurrent users without paying additional fees. This would have been a barrier for large ingest projects, especially those using contractors.

Two libraries in the environmental scan tracked their inventory in the ILS instead of using a separate system. The primary advantage of tracking inventory within the ILS was the minimal cost. It would, however, still require some local development, primarily to support the ingest process. Staff would need to scan barcodes for books, trays, and shelves as items were stored, and a process would be needed to insert that data back into the ILS. Interoperability with the ILS would be not be an issue, as circulation requests for materials would be handled with call

slips or lists, which would be printed at the facility. Interoperability would, however, still be an issue for the ILL system.

Although this approach met the minimum requirements for storing and retrieving items, it otherwise provided limited functionality. It would not allow for tracking or other management of open space, or for recording additional information (e.g., size of item). There were also data integrity concerns as items are regularly updated for other purposes, which could lead to the accidental loss of the storage information.

One academic library had developed their own IMS in-house. In many respects this appeared to be the best option. For example, it would allow the scope of the project to be managed as well as the prioritization of required features. The information about their experience, including the time and resources to develop their system, was encouraging. The primary advantage of developing a system is that it could be designed to precisely meet the specific IMS needs, and could potentially be less challenging than adapting an existing system not specifically designed for library materials management. Their advantages include full ownership of the software, absence of licensing fees, unlimited users, and the rights to modify or share the software as desired. The disadvantage is that we would be responsible for future maintenance and enhancements, whether that work was done in-house or contracted out.

While considering which option to move forward with, it was also acknowledged that for any project there are 3 factors that must be controlled: time, budget, and scope. This is known as the Triple Constraint. “This triple constraint must be balanced during the initiation and planning processes if the cost and schedule of the project are to be controlled through the execution and closeout processes.”[2] There were some unique challenges for the IMS project that put strict controls on multiple aspects of the Triple Constraint and therefore played an important role in our final decision on how to proceed.

With respect to the Time aspect of the triple constraint, there were two pieces to consider with the IMS project: personnel resource availability and the schedules of dependent projects. Due to other predetermined strategic initiatives, internal development resources were not available for developing an IMS. This pushed us to consider vended products or the use of a contractor if a custom solution was selected. Additionally, the IMS project was a dependency for two separate, yet related, construction projects. The first phase of renovation of the library required moving 500,000 items to the off-site facility, however the off-site facility required renovations before it would be suitable to house collections. The IMS had to be ready as soon as the Annex facility was completed to reduce delays in library renovations and inaccessibility of collections. Both staff time and construction deadlines were non-negotiable and inflexible.

Budget was another factor in the triple constraint that was tightly controlled,

as a specified project budget had been allocated. Along with initial implementation costs, maintenance and upgrade expenses were also weighed. While vended solutions offered less upkeep by internal development staff, they did require recurring annual fees that would have to be accounted for in all future budgets. A custom solution, while potentially built by a paid contractor, could be supported going forward by internal developers that were already accounted for in annual budgets.

The final item in the triple constraint, Scope, was the one the IMS team had the most control over. Vended solutions offered a greater scope, however this was often in areas that would be leveraged only slightly, and in many instances, not at all. Moving forward with a custom solution would require that scope be tightly controlled and necessitate identifying priorities and a minimally viable product. By going with a custom solution, functionality within the IMS could be focused on as prioritized by the stakeholders, with necessary functions implemented first and enhancements added in a controlled way to manage the balance of the triple constraint.

Based on the assessment conducted of the three possible options, in conjunction with the unique challenges of the project, it was decided that contracting a consultant to develop a system was the best way to proceed.

Roles

Defining the roles of the individuals on the project team was one of the essential factors that led to the success of the project. The core team consisted of a Project Manager, Systems Librarian, Lead Developer, and Contracted Developer. It was also important to identify the major stakeholders the team would interact with throughout the duration of the project. These individuals included the Project Sponsor and Library Staff that would inevitably be using the services developed.

As found in a study by Drexel University, “a RAM is very helpful to understand the roles and responsibility of the team.”[3] A RAM or *Responsibility Assignment Matrix*, is a useful tool for organizing individuals related to a project and the expectations for those individuals. One form of a RAM is a RACI chart.

Responsible - This indicates the people that will DO the work. This can be one or more people per identified task.

Accountable - This is the single person that makes sure the work gets DONE. There can only be a single person assigned as Accountable for each identified task.

Consulted - This is the person or people that are asked to provide insight related to each identified task

Informed - This is the person or people that are kept in the loop regarding each identified task.

The use of this tool helps to ensure that a single individual is identified as accountable for each task. It also provides clear definition of each person's roles and expectations.[4] We identified the main tasks for completing the IMS and indicated who was Responsible, Accountable, Consulted, and/or Informed for each. In the end, we defined roles and assigned expectations as depicted in Figure 2.

Figure 2. Project RACI Matrix

	Project Manager	Development Lead	Systems Librarian	Contractor	Project Sponsor	End Users
Determine Necessary Functionality	C		A,R		I	C
Design Functionality	A,R	C	R	I		C
Build Mock-Ups	A,R	C	R	C		C
Design Architecture	I	A,R	C	C		
Determine Coding Process	C	A,R	C	R		
Develop Functionality	A	R	I	R		
Code Review Functionality	I	A,R	I	I		
QA Functionality	A,R	I	R	I	I	C
Load Testing	A,R	C	R	I	I	R
Production Launch	A	R	R	R	I	I
User Training	A,R	I	R			I

R-responsible, A-accountable, C-consulted, I-informed

Specifications and Design

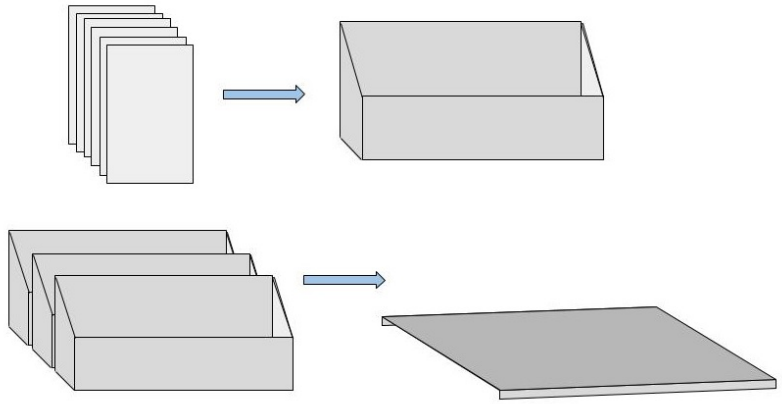
The Systems Librarian was responsible for compiling a list of required features for the IMS. Although this was actually done prior to the decision to develop an IMS locally, it also served as the first stage of design. The requirements comprised six areas:

- location tracking
- ingest

- bibliographic description
- de-accessioning
- request processing
- consolidating or rearranging

Location tracking was based on the physical arrangement of the (planned) storage facility. Items would be grouped by size and stored in trays. Trays of the same size would be stored together on high-density mobile shelving (Figure 3). The items, trays, and shelves would each be identified by appropriate barcodes. Tracking would be done by associating the barcode of the item to the tray, and associating the barcode of the tray to the shelf.

Figure 3. Items, Trays, and Shelves



An ingest process would be necessary to transfer materials to the facility, and record the data for location tracking.

In addition to the barcode, the Libraries wanted to record brief bibliographic data for each item. This would include author and title, ILS bib number, standard numbers (ISSN or ISBN), volume designation, and call number. Although the IMS would not be the system-of-record for any of this data, including it would have two benefits. It would allow the user to identify items directly in the IMS without first looking up the barcode in the ILS, and would ensure the capability of the system to provide identifying information during processing and at the point of retrieval.

Aside from ingesting and tracking items, the next area of critical functionality was the ability to retrieve items for use. The IMS needed to ingest requests from both ILS and ILL, match the request to the relevant item, and supply the needed information to retrieve the item. At this stage in planning, it was assumed that the IMS would provide this in a paper format, either as individual call slips or as a

retrieval list. Once an item was retrieved for use it would be necessary to track its status, particularly if it was sent out of the facility. Following use (whether an article was scanned at the facility or after a patron returned the item), it would be restocked.

Features to allow for consolidating or re-arranging materials in the facility, and for de-accessioning materials (either to transfer back to the main library or to permanently withdraw from the library collection) were also planned, but were not necessary as part of a minimally viable product for the first phase transfers. As such, these features were deferred for future development.

Once the decision had been made to develop rather than purchase an IMS, the next task was to design functionality that would meet the specified requirements. The Systems Librarian, Project Manager and Lead Developer worked together on this. This involved a few concurrent activities: strategic planning, making technical decisions, developing a database ontology, organizing the requirements into functional areas, and setting priorities for those functions.

The team began to develop general strategies and goals for the IMS. First, it was agreed that the IMS should be a web-based system, which would not require developing and installing client software on workstations. This would be simpler both for development and maintenance.

Second, the team wanted to develop a system that could be used on a hand-held device over Wi-Fi. This would allow the user to update data, e.g., stocking a tray to a shelf, in real time, rather than recording information to be uploaded to the system later. This also allowed a change of direction from the use of paper call slips or retrieval lists (as specified in the original requirements) to presenting retrieval information interactively via the hand-held device.

Third, the team adopted an API oriented design. That is, a design that utilizes http-based interfaces between the various systems (IMS, ILS and ILL) to provide real-time interactions instead of relying on exporting data from one system and importing it into another in batches. For example, the original specifications had indicated that items and related bibliographic details would be pre-populated by exporting the data from the ILS and loading it in the IMS prior to ingest. With the API approach, this information is queried as individual items are ingested. This saved time and effort that would have been involved in the export/import process, allowed the system to operate in real-time, and avoided loading data for items that might not actually be ingested. The same API strategy was also used for retrieving requests and for updating the ILS whenever an item was stocked in or sent out from the facility.

In terms of basic technical decisions, the team decided to use Ruby as the development language, PostgreSQL as the database and Amazon Web Services (AWS) as the hosting platform. Ruby was already one of the primary languages

used for local development and seemed well suited to this project. PostgreSQL was chosen due to its extensive feature set in comparison to other open-source relational database systems. AWS presented some challenges for the team, but was adopted for the project in response to a university-wide initiative to utilize cloud-based platforms.

The database ontology (objects, properties and their relationships to each other) largely flowed directly from the requirements: *items*, *trays* and *shelves*. As discussed above, items have bibliographic properties such as title and identifying numbers. They also have conditions (e.g., damaged covers), size information (used for determining capacity) and current status (e.g., stocked). In addition, the team decided to treat *requests* as objects in the system, to organize requests into *batches* and to place retrieved items in *bins* until the request was filled. With this in mind, the team began to organize the features into functional areas, such as *stocking* and *batch processing.* Stocking addressed the first three required features from the specifications, and was given the highest priority for development. Stocking was further divided into smaller functional areas such as 'item to tray' and 'tray to shelf.' The team created sketches of the user interface (UI) for these areas, although some changes were made as development progressed (Figures 4 and 5).

Figure 4. Initial UI Sketch for Tray Stocking Function

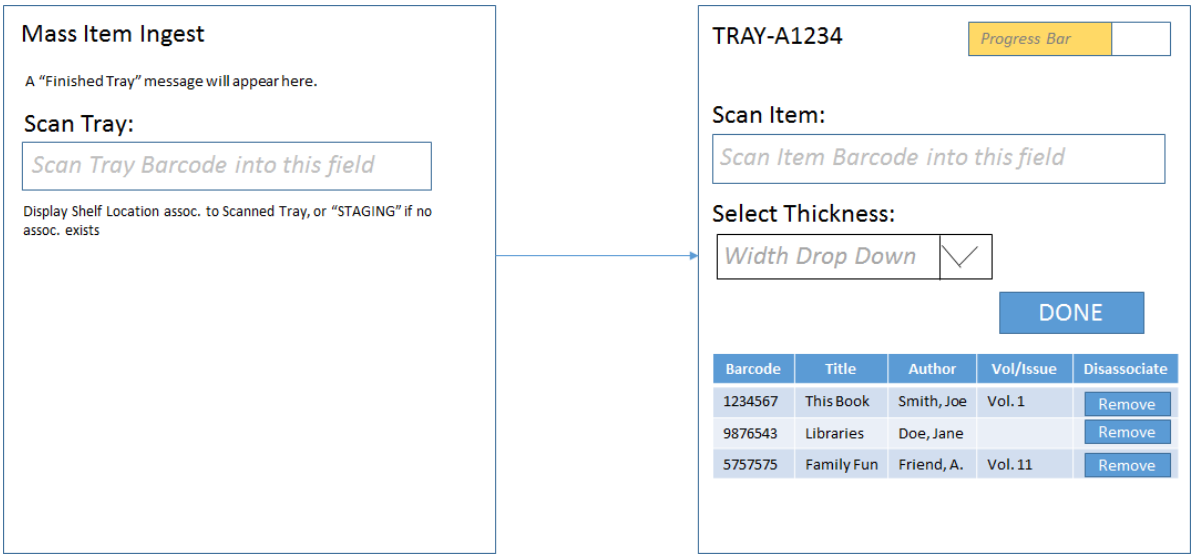
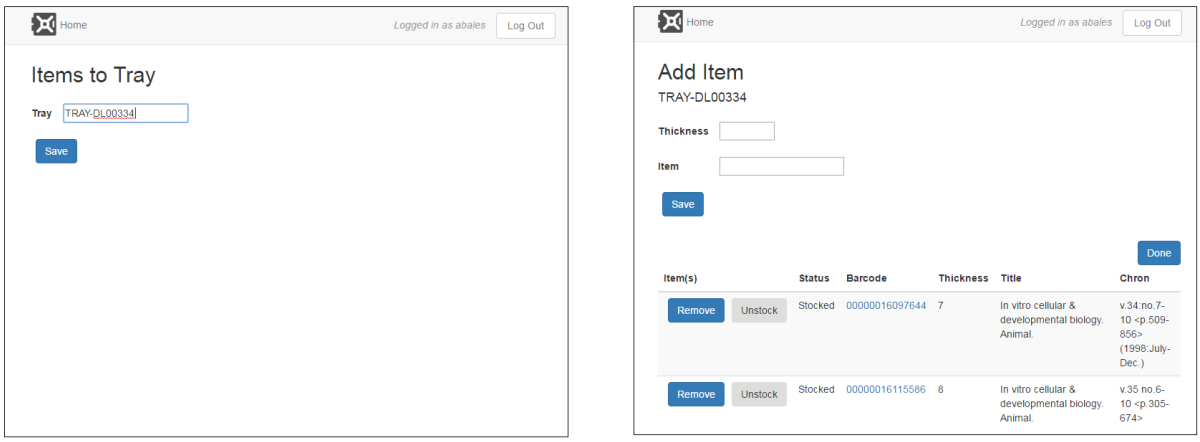


Figure 5. Actual UI for Tray Stocking Function as Developed



Batches were given the second priority. Although the concept of a “batch” was not specified in the original requirements, the team and stakeholders adopted a batch approach to processing requests as they discussed workflow. Each request would be matched to the appropriate item and assigned to an active batch. The requests and their matching items would remain in the batch throughout the retrieval and fulfilment process.

Along with developing specifications, the Libraries moved forward with engaging a contractor for the project by working with TEKsystems, an IT staffing company. Although working with a contracted developer was a new experience for the library, the central Office of Information Technology (OIT) had used this approach for previous projects. Talking with members of the OIT, especially about their good and bad experiences using contractors, proved to be invaluable. Some of the key takeaways were the importance of interacting frequently with the developer, staying closely involved with the process, and having short term tasks and well defined deliverables. This would help to insure that the developer understood what the Libraries needed, and allow for early redirection when necessary.

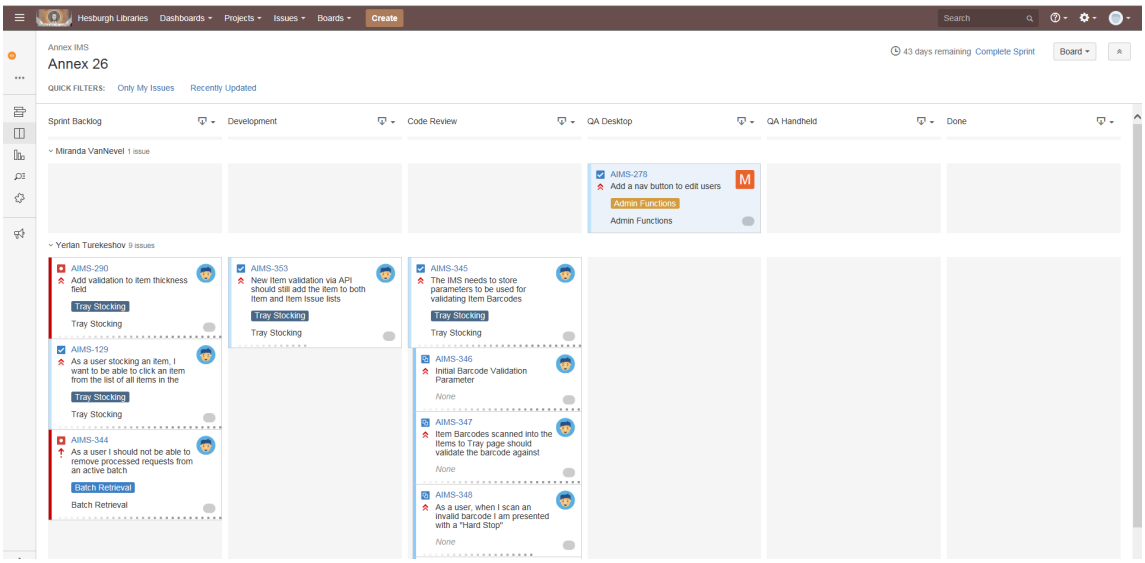
The team met with representatives of the company to outline the project and discuss the skills and experience that a contractor would need. TEKsystems identified two candidates for the position, who were interviewed via Skype by a group of six library staff, including the Project Manager, System Librarian, Lead Developer and other Developers. Following the interviews, both candidates were asked to complete a sample programming task, and one was selected.

Following the selection process, TEKsystems was involved with record keeping and payments to the contractor, but the Libraries were responsible for assigning and overseeing all work. Prior to beginning active development work, the contractor was invited to a site visit on campus for an overview of the project, including the development cycle and expectations.

Software Development

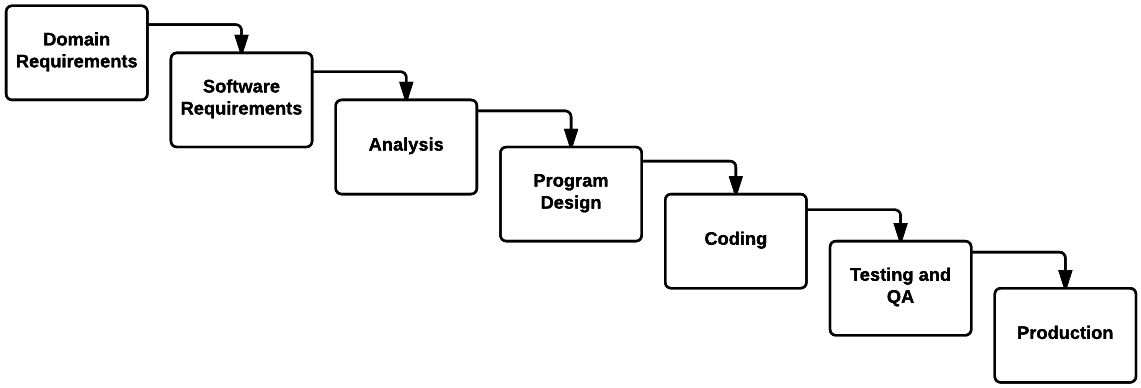
After having chosen to work with a contractor, it was agreed that there needed to be a consistent method to assign and track work, and that the method needed to satisfy several concerns. One of the primary concerns was monitoring the number of billable hours consumed by the contract developer due to the limited budget of the project as a whole. Also features and tasks needed to be prioritized and moved through a specific workflow that involved the backlog, active development, code review, two phases of quality assurance (QA), and done. The method used therefore needed to capture the current state of work, the priorities for the project, as well as a method to predict when features would be completed. Finally, because the primary developer on the project would be working remotely, the system used needed to quickly facilitate communication. To this end, Jira (Figure 6) was chosen as a tool because it incorporates functionality that supports the tracking of tasks in discrete phases of work, and allows the developer to manage tasks via an easy to use click and drag interface.

Figure 6. Jira Board used to track tasks.



Because the requirements of the IMS continued to evolve over time, it would have been very difficult for the team to adopt a method similar to the waterfall method that had been traditionally used by teams in the 1960s, 70s and 80s. This method was originally described by a software engineer from the Lockheed Corporation by the name of Winston Royce.[5] A simplified diagram below (Figure 7) is taken from Royce's 1970 article, which demonstrates the workflow of the waterfall method.

Figure 7. Waterfall method of project planning and execution

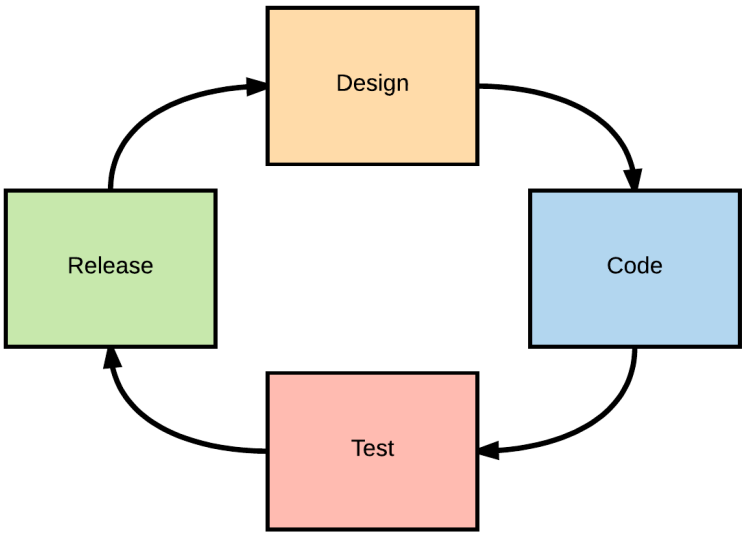


This method is tenable under the conditions in which Royce was working at Lockheed, where planning involved a significant amount of mathematical and engineering analysis prior to the development of software that would regulate complex aeronautical systems. In the case of the IMS, requirements could evolve as the software was tested. Therefore, the team chose to use a modified Agile Kanban method for assigning work to the contract developer. Agile Scrum was not feasible because it requires the development team to have daily standups with a Scrum Master and the contract developer did not have that level of availability. Agile methodology treats aspects of development work as a part of a larger "tale" that is being told. Major components of the IMS were described and labeled as *epics*. Epics, in agile parlance, are major feature sets that require a larger time and effort investment in order to complete. The smaller tasks that are assigned to individual developers are labeled *stories*. Agile is flexible enough to allow mixing and matching of tasks at any point such that they can be broken down into other stories, or collected into a larger epic. This makes it much easier to conceptualize larger projects, as well as measure progress.

The Agile Kanban method is modeled after various forms of Japanese manufacturing, and follows a production schedule similar to auto manufacturing where the product (the car) is modified at a component level as it passes through the manufacturing line until it is complete. Kanban tends to be more linear as features are worked on in a more strictly serial fashion. This differs from Scrum, which is more flexible and takes into account the potential mutability of outcomes given the input of the product owner.

Agile favors a more iterative process that entails continuous involvement of a *product owner,* who represents the user population as well as the functional requirements. In this case, the project manager in collaboration with the systems librarian collectively held this role. A representation of the Agile method appears in Figure 8.

Figure 8. Agile rapid release cycle



User stories are the central component of any Agile method, because they represent the requirements of the application and the atomic tasks that developers engage in. In the case of the IMS, time was used as an estimation metric primarily because the hours worked had to be closely monitored. The stories were written from the user's point of view and encapsulate the functional requirement in the way they are written. For example:

> *As a user I should not be able to remove processed requests from an active batch*
>
> *As a user using the scanner when I select current batch I should see the first item on the list*

As each functional requirement was worked on and completed, the developer was required to submit the code for review so that the internal development team could inspect it for quality assurance purposes. This was the first phase of the review process prior to release. The process also included two QA phases based on the requirements. The first QA phase was used to test functionality on a desktop because almost 50% of the work would be done on a desktop computer. The second phase of QA tested the functionality in a mobile or "hand held" environment. This was necessary because much of the work of stocking and retrieving items would be completed in the compact shelving on a mechanical lift, and the technician

would be using a handheld scanner with a screen size roughly equal to a mobile phone.

The final phase of testing involved “user acceptance” testing, which was done on an infrequent basis – usually when major features were released that would affect the workflow of the annex technician. Following the two QA phases and user acceptance testing, the software would be released in production and made available to users of the IMS.

As previously stated, the agile method facilitates breaking down more complicated tasks into smaller, actionable user stories that can be easily tracked over the course of a development cycle. Flexibility is built into the cycles through the focus on a *minimum viable product* for any given feature. Once the minimum functionality is achieved, the team can proceed with feature enhancement. The volume of functionality required to satisfy the basic needs of an IMS required the team to work serially through the user stories as quickly as possible in order to achieve a minimally viable product. When a revision was required on a particular feature, a new user story was written that reflected the required change and put into the backlog.

The turnaround time for each feature set was approximately two weeks. In Agile parlance, these were two week *sprints*. At the beginning of each sprint, the team would review what had been accomplished previously by the developer, discuss roadblocks, and then add stories to the new sprint – keeping in mind that there was a minimal set of functionality required to be in place by the deadline. During each sprint, the team would iterate through the entire process of design, coding, QA and release. This promoted an atmosphere of “continuous delivery” for the product. As functionality was completed, having gone through testing and QA, it was released into production and could be used by the annex technician.

For the purposes of planning and sprint review cycles, the team chose Google Hangouts to have “face to face” contact with the software developer. The participants in those meetings usually included the project manager, the systems librarian, and the manager of software engineering. The meetings lasted roughly an hour, and gave the four participants time to go over the critical aspects of the project on a sprint by sprint basis. Realizing that the developer would need more frequent access to the product owner, the team was set up with a collaboration product, HipChat, which allowed the team to communicate instantly with one another during normal business hours. If anyone had a question or needed to clarify something, that could be done via HipChat. Lengthier conversations happened during the sprint review on Hangouts. This proved to be a successful communication method for the team, and satisfied the need for frequent brief conversations and questions as well as extensive planning sessions that required more time.

It should be noted that even with precautionary measures in place, and communication channels established, expectations needed to be managed closely to keep the project on schedule. The contractor was wrapping up obligations associated with a previous project when he began work on the IMS. This isn't unusual from a contractor perspective, but it initially created a roadblock for the IMS team because the contractor had overestimated his capacity during that period. The actual development work began at a slow pace, and after 3-4 weeks of negotiations and reminders it became apparent that the project was beginning to head down an unsatisfactory path. The project manager was able to set more rigid expectations of biweekly output at that time, though, and the contractor was able to dedicate focused effort to our development schedule. This hiccup didn't ultimately impact the overall delivery schedule, but the contractor did need to put in extra hours near the end of the project in order to satisfy the minimum viable product standards.

The team also knew going into the project that a rigorous code review process was needed to assure quality control and maintainability of the code going into the future. At the outset of the project, code standards and style issues were negotiated with the contractor and acceptable compromises were made. While every programmer will solve application programming issues in their own way, it was important to establish norms that would allow the in-house development team to parse the code and understand the overall organization of the application. The local developers also negotiated certain infrastructure concerns such as which relational database to use, how indexing would be accomplished, which CSS and JavaScript libraries would be used, etc. Allowances were made on both sides of the equation and the team was able to come to an acceptable understanding with the contractor.

Quality Assurance and Testing

As the product matured and basic functionality was implemented, the team met with some predictable challenges for a project of this scope. The challenges could be roughly aggregated into two groups: performance tuning and logging / reports. Each of these areas had to be dealt with separately and were primarily solved by the Libraries' in-house software engineers instead of the contract developer. This decision was made partially because of budget and time constraints, and partially because of the complexity of the problems.

Having identified application performance as a potential barrier to the success of the project, the Libraries' software engineering team engaged in smaller projects to make sure the application would be scalable during periods of high throughput. Performance was an implicit requirement that the product team had as an expectation from the beginning of the project so it was necessary to address this even after the bulk of development had taken place and the contract developer was no longer on the project. During the first round of item ingesting, the quantity of

items being processed (~500k items) necessitated that the system perform reliably under heavy load. Another potential hazard would be lack of network availability and throughput bottlenecks during those peak periods. Ingest would need to continue even if the network between the IMS and the campus based API failed. Therefore, it was decided to implement background processing as a core infrastructure component, which was done about a month prior to the beginning of the initial ingest. This guaranteed that the initial bulk stocking work could continue in the event that there was a network glitch between Amazon Web Services and the campus network.

Another expectation that the product team had, which was based on an understanding of how the system would be used, was that transaction logging and basic reporting would be available as a part of the initial rollout. This feature set, in addition to the performance enhancements, was developed by the in-house development team. It turned out that this was beneficial to the transition from relying on the contract developer to in-house maintenance and support. The local development team was able to become more familiar with the architecture of the IMS, and that familiarity fueled the ability to implement enhancements and bug fixes. Other features that had been talked about during the initial planning sessions were determined to be lower priority and put into a backlog. However, they were described with sufficient detail such that the local development team could implement those features over time and as the need arose.

Once the project neared the stage where the team would transition away from the use of the contractor, the release cycles were planned out and happened on a scheduled basis. Once the IMS was in production, the team needed to have a mechanism to communicate when the maintenance windows would occur, and when people would need to refrain from using the system. The team ended up establishing a set maintenance window that would be least disruptive to the team working in the annex ingesting items on a daily basis. A changelog was also created that could be published to let the annex staff know what was changing and to set expectations.

Conclusion

All software projects have inherent challenges such as time constraints, personnel limitations, communication gaps, and budgetary concerns. While a software project of this complexity is not unheard of in the library community, neither is it commonplace. The managerial hurdles that were faced during the development of an IMS for the Hesburgh Libraries required a high level of coordination and decision making with very little room for mishaps. The success of this project needs to be attributed to the efforts of many individuals working closely together with a high degree of communication and investment. In addition, the

success hinged on having a multidisciplinary team with complementary areas of expertise who were willing to devote a significant amount of time towards the goals of the project.

From the managerial perspective, it is always important to capitalize on individual strengths, and compensate for weaknesses in team composition. In this case, as time was a critical factor, and due to capacity limitations it was necessary to hire a contract developer. The advantage of doing this is to increase capacity, and that was certainly the case here. It should be noted, however, that the overhead required to manage an off-site contractor can be significant. Expectations need to be clearly outlined, and immediate goals defined or, as with any employee, misunderstandings arise. In this case, it had to be made clear about a month into the process that the project required more devoted time from the contractor to meet planned goals. There was also a need for more frequent communication with the project manager and in house development team to make sure sprint goals were achieved. While the project did go a little over budget, the team was able to do a course correction regarding time management, which helped considerably.

In order to compensate for the overhead required to manage a contractor, there are some important considerations to keep in mind. When entering into the contract, it is important to understand the prior obligations that may be in place for the contractor. As a part of setting expectations, management should be clear about expectations regarding time. Having both a verbal and written agreement regarding the minimum time commitment per week may be necessary in order to maintain momentum. In general, it is difficult to estimate the number of hours required to complete tasks. This has a direct impact on the overall cost of a project when tasks are chronically underestimated. In order to compensate, library management should aim for modest goals to begin with and make sure that the essential feature set is outlined prior to engaging a contractor. One of the best antidotes to the malady of underestimating time is to have engaged product owners who are actively prioritizing tasks when needs change as well as at the beginning of the project. Finally, with regard to employing a contractor, it's essential to make sure that the individual is suited for the technical aspects of the project. One useful strategy to gauge this factor would be to give the contract programmer sample problems to solve, and have them submit their work to a local technical team for evaluation. This was used during the vetting process for this project and proved to be a successful strategy.

One of the most valuable assets that brought the project to a successful conclusion was the use of project management. Given the number of stakeholders involved as well as the necessity for accuracy and usability, it would have been

impossible to develop an effective product without the oversight provided by project management. More specifically, the project manager in this case was heavily involved in the daily operations of preparing the Annex and library staff for handling the demands of a radically new process. The level of insider knowledge provided by the project manager allowed the contractor and local development team, as well as the stakeholders and product owners, to do their work efficiently. Important knowledge was concentrated in a single individual who acted as a reference point for everyone involved.

Finally, another benefit as well as challenge for the team was the use of cloud technology to host the application and data for the IMS. There was a certain degree of overhead that was required in order to solve problems that are rather cutting edge within IT let alone library IT. In the long run as a part of a larger strategy, this was a wise move. With the momentum behind the campus cloud initiative, it was a relatively easy choice to make but it still presented the team with additional challenges. Fortunately, there were motivated systems staff on the team who worked diligently to not only learn brand new processes but also efficiently prepare a production environment within which the IMS could be hosted. Having skilled technical personnel on staff is another essential ingredient that contributes to the success of a project of this nature.

The Inventory Management System was rolled out to production in the summer of 2015 for initial ingest to begin. The process ran smoothly in large part due to the attention that went into the design, code review, quality assurance, and functional and performance testing. In August of 2015 the Annex facility officially opened for taking requests for the approximately 450,000 items that had been ingested into the IMS. Patrons were able to request the items and receive them within the time expectations set by administration. The Hesburgh Libraries is currently exploring ways that the software could be made into an open source project, maintained and shepherded by the larger library community in response to interest indicated by other institutions. The internal success of, and external interest in, the IMS are indicative of the success of the project. This is due in no small degree to the positive elements of the management process as described here, and it is hoped that this experience is useful to other libraries considering similar endeavors.

Aaron B. Bales (Aaron.B.Bales.2@nd.edu) is Head, Discovery and Access at Hesburgh Libraries, University of Notre Dame

Robert J. Fox (rfox2@nd.edu) is Manager, Web and Software Engineering Unit at Hesburgh Libraries, Univeristy of Notre Dame

Miranda R. VanNevel (Miranda.R.Vannevel.7@nd.edu) is a Digital Project Manager at Hesburgh Libraries, University of Notre Dame

Published: August 2017

---

[1] Ron Chepesiuk, "Reaching Critical Mass: Off-Site Storage in the Digital Age. (Cover Story)." *American Libraries* 30, no. 4 (1999): 40.

[2] Kuehn, Ursula. *Integrated cost and schedule control in project management*. 2nd ed. Management Concepts Inc., 2011.

[3] Mark Allen, Del Alleyne, Crystal Farmer, Angela McRae, and Charles Turner. "A Framework for Project Success." *Journal of Information Technology and Economic Development* 5, no. 2 (2014): 1-17.

[4] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* Fifth ed. Newtown Square, Pennsylvania: Project Management Institute, Inc, 2013.

[5] Winston W. Royce, "Managing the development of large software systems." In *proceedings of IEEE WESCON*, vol. 26, no. 8, 328-338. 1970.