A Case Study on Using the Mendix Low Code Platform to support a Project Management Course

Lionel Mew
lmew@richmond.edu
University of Richmond
Richmond VA

Daniela Field
daniela.field@mendix.com
Mendix Corporation
Boston MA 02210

Abstract

This paper discusses the experience of using Mendix, a low code platform, to support the goals and objectives of an undergraduate project management course in information systems curriculum for students who are continuing studies in a bachelor's program. Model Driven Development (MDD) tools such as Mendix have evolved from the Computer Assisted Systems Engineering (CASE) tools of the eighties and nineties. They now yield capabilities that push these tools into mainstream development. A tool such as Mendix is designed to allow system development at a higher level of abstraction and generates fully functional applications from a model driven environment. The legacy CASE tools never reached the anticipated wide use because the technology of the time was not able to fully support the hype. The CASE tools required intensive resources, maintenance, support, training, knowledge of use, and rarely performed fully as advertised. As the new tools continue to emerge into the mainstream, students and practitioners must develop competencies in their use. This paper provides insight into using Mendix to support a project management course and discusses the successes and the challenges.

Keywords: Mendix, low code, project management, model driven development, CASE, Information Systems.

1. INTRODUCTION

This paper reports on the use of Mendix, a low-code application development platform. It showcases and provides details on how Mendix supports the learning objectives of a project management course in an information systems bachelor's degree program. The use of this tool in support of the course is a result of the emergence of Model Driven Design tools such as Mendix in a new wave of Computer Aided Software Engineering (CASE) tools with expanded capabilities.

From the advent of CASE tools about 1970, the ability of these tools to easily and reliably generate fully functional applications did not emerge until decades later. CASE tools are software tools used to design and generate applications. This occurs at a higher level of abstraction than application development using linear programming methods (Halpern & Tarr, 2006).

The CASE tools of the 1980's facilitated higher levels of abstraction by automating and simplifying application development using the context of domain models, with the tool generating development documentation, code, and in some cases, fully functional applications.

While many professionals used case tools as aids to the development process, very few of them used the tools for full-blown database application generation engines. The cost and complexity of implementing and maintaining the tools, training and retaining their users, coupled with the tool's abilities and failure to perform as predicted led to the tools having very little commercial impact during the 1980's and 1990's (Schmidt, 2006). Jones (2002) notes that as much as 70 percent of CASE tools were not being used by the end of the first year.

It is widely accepted that reasons for the first generation CASE tools not gaining widespread acceptance include unrealistic performance expectations and inadequate training. (The FFIEC IT Examination Handbook Infobase) The emergence of better tools for Model Driven Development (MDD), evolving from first generation CASE tools, has improved performance to the extent that they are now being used for large-scale development projects according to Gartner 2018 Magic Quadrant report and assessment of such tools.

Efforts are being made to address the training issues. It is imperative for students about to enter the workforce to have some exposure to these new and innovative tools. Incorporating Mendix into this project management course is an effort to provide students with exposure to a development project using an MDD tool, in addition to fostering student understanding of working in teams. Students work in group projects and have diverse roles and the Mendix tool fosters their collaboration throughout the development process.

2. STUDENT DEMOGRAPHIC

The student population at the School of Professional and Continuing Studies (SPCS) consists of mostly nontraditional students. Although nontraditional defies definition, demographic information gives perspective to the typical student. The average student age is 37, although information systems students tend to be closer to 30. Although the majority of students have traditionally been men, the number of women in the program is slowly increasing, with women accounting for more than half of enrollments in the current semester. Experience levels and goals of female students are similar to those of the male students.

Eighty-one percent of students are part-time. Both part-time and full-time students are working on either Bachelor of Science in Professional Studies degree with a major in Information Technology Management or Information Security, or a post-bachelor Certificate in Applied Studies in Information Systems or Information Security. Student experience varies, with some students who have associate's degrees or at least some community college work, and students who have immediately transferred to SPCS with a desire to complete their bachelor's degree. Some students have been in the workforce for some time and need a degree for promotion. Whereas others are trying to break into the information systems field, often with significant work experience and success in other fields.

The factors which make SPCS students unique leads to a wide variance in student understanding, experience and ability. All major core courses are classroom courses, although some are offered in the hybrid format of in person and online courses. There are currently no completely online information specific systems courses. Most of the students live in the local metropolitan area and most of the students decide to stay in the area after graduation. Courses are generally capped at 15 students, allowing significant individual attention and interaction with instructors. The project management course is a four-credit hour course meeting weekly in the traditional format.

This demographically diverse student population presents numerous opportunities as well as challenges. Many of these opportunities and challenges are well known and well documented elsewhere. An example of opportunity is the benefits of experience and maturity of the student population. A challenge is their family and work obligations. The applied aspects of the project management course arguably add more value to this student population – whereas the traditional students continue to mature and learn to think critically during their degree programs, it is expected that the continuing students in our program focus more on professional competencies. Thus, the primary focus for instructors is to help students grow professionally and gain industry-relevant skills.

To facilitate learning amongst this target population, the high-level philosophy is to provide an applied aspect for each course. The project management course, offered for over a decade in this program, has long included a course project. It was hoped that by using Mendix, the students would add additional realism and domain skills to student repertoires.

3. MODEL DRIVEN DEVELOPMENT

Background
Model Driven Development (MDD) is a software development methodology that uses model architectures to raise the level of abstraction so that complexity can be decreased, and productivity increased. The use of CASE tools began in the early 1970's. Teichroew & Hershey (1976) discuss how a tool was used to analyze, define and document requirements and specifications for information systems. During the 1980's and 1990's, first generation CASE tools were most prolific and reached their apogee. In the 1980's CASE tools were used to collect requirements, and by the 1990's the tools were generating partial application code. However, these tools never lived up to expectations, and never had the expected effect on development paradigms (Yourdon, 2001; Henkel & Stirna, 2010; Schmidt, 2006).

Current State
Evolving from the CASE tools of the eighties and nineties, the current crop of MDD tools have improved functionality to the extent that they can reliably generate applications and are easier to use and maintain than ever (Selic, 2003). Henkel and Stirna (2010) provide a structure for evaluating MDD tools. They list metrics for the desired functionality of the tools. Areas of desired functionality include modeling support and development support. According to Henkel and Stirna, modeling support consists of language(s) for modeling constructs in the specific domain. This is supported by key underlying areas of abstraction, understandability, executability and model refinement. Regarding development support or support for development processes, Henkel and Stirna found the literature suggests six areas: observability, turnaround time, collaborative development support, integration, developer competence support and reusability.

When evaluating Mendix against these criteria, Henkel and Stirna found Mendix suited to the web-based development of small to medium complexity, for small projects with short delivery times. These qualities were determined to make Mendix a good fit for supporting this course.

The FFIEC IT Examination Handbook Infobase (n.d.) proposes four areas of risk when considering the implementation of CASE tools, including inadequate standardization, unrealistic expectations, inability to implement quickly and weak repository controls. Mendix was chosen based on their ability to manage these risks.

4. MENDIX

Mendix was used for the course due to its functionality, simplicity and the support offered by Mendix. The platform is the leading low-code solution recognized by analyst reports like Gartner and large enterprise companies such as SAP and IBM. It is a full stack platform designed to build applications rapidly. The platform abstracts and automates the various application development layers from the front end to back end. For example, the data structure is build using unified modeling language (UML). The business logic uses Business Process Modeling Notation (BPMN) and the user interface is built with widgets following a what you see is what you get (WYSIWYG) model.

The platform allows for business and IT to collaborate and build applications that add business value. It handles 6 main functions:

- Collaboration
- Data Structure and Domain Model
- Business Logic
- User Interface and Experience
- Security and user authentication
- Deployment

Collaboration
Collaboration is the key to building successful applications that solve business problems. The business understands the critical business problems and needs digital solutions to fix those problems. Whereas, the IT department needs to support the business by providing the solutions that work. Communication across these two departments has always been challenging. For both business and IT to collaborate, speaking the same language is vital to delivering business solutions on time and under budget. In addition, it is unreasonable for solutions to be delivered months and years after the original business problem has been identified.

To deliver applications as quickly as 2-3 weeks, there needs to be a process change. The agile methodology follows the iterative process and business can have a minimal viable product within weeks and can iterate as needed (Frydenberg, Yates & Kukesh, 2017). The agile methodology allows for iterative development and for the business to provide input and shape the product before it is delivered.

When students create a Mendix project, the collaboration workspace is automatically created with all the built-in agile process features. For example, students can capture their user

requirements and add user stories and sprints. They can create sprints that run for a fixed set of time and manage the backlog of stories and work to be done. In addition, using the feedback widget, they can gather feedback from the business or professor and implement additional features and functionalities. The application they build in the first sprint will be vastly different than the application they deliver at the end of the semester. Students can see the process and workflow as they build the solutions out. The project manages the code repository and code check-in and check-out process which allows for multiple students to work on a diverse set of user stories.

Illustration 1 in Appendix A provides an illustrative overview of the agile process.

Database Structures
The database structure is the foundation of the business solution and where data is stored. In the Mendix, students do not have to worry about what kind of database to create and manage, they don't have to create SQL queries and other code functions central to the architecture of an application.

The database structure follows the Unified Modeling Language (UML) principles and is composed of 3 main components: entities, attributes, and associations. The entities are translated into database tables. The attributes are the table columns with a type and size. The associations store the primary keys of the related entities in the database automatically underneath the hood. Associations are the relationships between entities based on cardinality: one to many, one to one, or many to many.

Business Logic
The business logic is designed using the principles of Business Process Modeling Notation (BPMN). The business logic is the logic layer in the application and students will build microflows. You can use microflows to add logic such as validations, calculations, integration pieces and other activities. Microflows are visual representations of all the logic and steps. The main building blocks are start and end points, activities, exclusive splits, loops and much more.

For a microflow example see Illustration 2 in Appendix A.

The example microflow shows the business logic applied to classroom registrations. It checks if the class is full before allowing a student to register.

User interface and user experience
The user interface and user experience are what the end users of the application will see. It is essential to build an application that not only looks good but the end users can navigate with ease and fit their needs. The users shouldn't have to think about where to go and what to do. You want user adoption and the application should be easy to use and delight the users.

If the application has too many buttons and shows too many data, the end user will be exhausted and not know where to go. Thus, applications need to delight end users and provide an experience that flows and feels natural to the end users. In Mendix, the user interface is designed with a what you see is what you get (WYSIWYG).

Students can drag and drop user interface components and arrange them as they see fit. The can design and iterate over the pages as needed based on feedback and end-threuser behavior. Mendix follows 3 principles for design: harmony, simplicity, and flexibility. Additional resources may be found on the Mendix website (https://atlas.mendix.com/).

For a UI/UX dashboard example see Illustration 3 found in Appendix A.

Students can build responsive pages that will render on any device type from desktop to tablet to mobile. In addition, they can build mobile applications with fully native functionalities.

Security
In today's world, security and privacy are paramount. Thus, an application has to have user authentication and allow users enough permissions based on their user roles.

As a developer, user roles are defined in the user stories and implemented based on what the users should be able to see and do. Most applications have some form of user authentication and limited permissions based on user roles. As a developer, user roles are created from the user stories. The developer can limit the access of forms, microflows, and specify the create, read, update, and delete (CRUD) permissions on the entity level. Through clicks and easy to navigate dashboards students can see the access rules and give user roles permissions based on the user stories and functionality.

Deployment
In addition to building the application, Mendix is partnered with cloud providers to handle the infrastructure of deploying applications. The students can build their applications and deploy them into a free sandbox environment. They can share their applications with anyone they want.

3. COURSE DETAILS AND DELIVERY

The course discussed in this paper is a three-credit undergraduate project management course. It has been offered for over 10 years and has always had an applied component using a group project. In the current iteration of the course, the group project used Mendix instead of the traditional group processes such as brainstorming to develop a scope statement, work breakdown structure, charter, etc.

Learning outcomes for the course included three high-level goals with supporting objectives. These goals included:

Goal One: Examine and understand project components and phases.
The goal helps students understand the basic concepts of project management from a holistic view.

Goal Two: Understand the basic functions required to create a project such as scoping, chartering, work breakdown structure development, task identification, scheduling, resource assignment, status tracking, contracting, earned value analysis, and risk management.
The goal helps students learn to develop a notional IT Project Plan.

Goal Three: Understand Uses and Limitations of Project Management Software Tools
The goal helps students understand how the triple constraint affects project implementation and control using software tools.

Mendix Goals and Objectives
In using Mendix for the course project, an additional goal and supporting objectives were added:

Goal Four: Work in teams on an App development project Mendix.
The goal helps students learn how to work in small teams and to develop real-world apps in a low code environment

Objectives

- Students will develop a basic understanding of Mendix
- Students will become certified in Mendix Rapid Development
- Students will use Mendix to develop an App

To achieve this goal and objectives, the following activities were included in the syllabus:

- Completion of Mendix online rapid developer training modules.
- The Requirement that students attain rapid developer certification.
- Classroom lectures and discussion on Mendix.
- Group project with a deployed app using Mendix.

Pedagogical Delivery
There is a significant body of literature discussing pedagogical delivery methods for project related courses. For example, a Lynch, Goold & Blain study (2004) discusses student preferences for how project courses are delivered. The authors suggest that the delivery method affects the control amount instructors have over course conduct, based on the pedagogy of the teaching model. They discuss four models: industry-sponsored, studio, traditional and directed.

The industry-sponsored model uses a scenario where students play the role of junior enterprise employees, with course tasks assigned dependent on enterprise needs. In the studio model, students collaborate with experts and mentors to gain insight into best practices. In the traditional model, students collaborate in teams on projects, with a low level of interaction with faculty. Finally, in the directed model, students work with a technical and a managerial faculty member, on a clearly defined set of deliverables. In Lynch, Goold and Blain's 2004 study of 196 students from three institutions, studio, traditional and directed models were examined. The most significant finding is that students much prefer well-defined deliverables.

It was determined that for this course, the most appropriate delivery method would be the directed model, with tenets of the traditional model interjected.

The Course Structure and Experience
The first seven weeks of the course consisted of introductory work on basic aspects of project management. To prepare for the applied portion

of the course, students were assigned two modules a week of the online Mendix Rapid Developer certification self-study course.

An immediate challenge was that of verifying that students were keeping up with their modules. The fear was that students would procrastinate and not complete the modules until the last minute when certification was due. The course policies on late work (one letter grade a day, with a minimum of 50 percent for any completed assignment) served to motivate students to complete the assignments, but validating completion proved to be a challenge. It was decided to have students upload a screenshot of their module completion screen to a Blackboard assignment, but the solution was never fully implemented due to time constraints. It was discovered after course completion that Mendix provides a way for instructors to check student work and ensure completion of assignments. This functionality will be used when the course is taught again.

As students began to complete their training modules, the process of testing began. The first student to take the exam was a hard-charging, highly motivated student. She already held a bachelor's and master's degree in a technical field, yet, surprisingly failed the examination. She and other students opined that the test was difficult to understand, with a poor translation from the original language. This feedback was passed to the Mendix team working on certifications. The team was responsive to the feedback, and they anticipate a new exam release within months.

Fortunately, Mendix provided superb support in resetting tests for students who failed on their initial try. This was significant, as Mendix provided the certification tests to students at no cost, whereas a retake for a retail customer would incur a significant test fee. Most students passed the test on the first or second try, with only one of the six students requiring three tries, and only one student failing to become certified. Achieving Mendix certification constituted 10 percent of the semester grade.

Following completion of the online training and certification process, students began work on their course projects. From the beginning of the semester, participants were placed in groups of three for the entirety of the semester. Early placement of students in groups was deliberate, in hopes that this would jumpstart Tuckman's (1965) model for the development of small groups. Tuckman proposed that all groups go through stages of forming, storming, norming, performing and adjourning. Introducing this concept to students was a critical part of the course, as these factors are critical to project management. Students were therefore aware of these processes and were able to experience group development as part of their course project. They started by forming their groups and creating a Mendix group project. The students starting the storming and norming process by creating the requirements and diving the work to be done. Within Mendix, the professor can see who is doing most of the work and how the students are tackling the requirements.

At the start of the course, participants developed a project charter and work breakdown structure to kick off their projects. In addition to providing insight on how the use of these deliverables applied to real-world projects, these exercises helped students scope and frame their course projects.

There were no prerequisites for the course, although most participants had previously completed systems analysis or database courses. These courses helped with the Mendix project. This an example of the kind of synergy that program developers strive for – courses informing other courses, with students having opportunities to apply concepts learned in previous courses in real-world settings.

Students were required to provide intermediate deliverables or milestones. This was accomplished by having an assignment (with requirements listed) on Blackboard. When they completed the requirements, students would commit and save their changes in Mendix, then notify the instructor that there were items requiring grading via Blackboard.

After creating their Mendix projects and familiarizing themselves with the development environment, the first deliverable for students was to develop their use cases, referred to in the Mendix processes as *user stories*. This was accomplished by having participants first develop requirements for application functionality. These requirements were translated into use cases by operationalizing them and incorporated into an excel spreadsheet as user stories. The spreadsheets were then uploaded into the Mendix development environment, although one group chose to bypass the spreadsheet phase, and type in the user stories directly into Mendix.

After all user stories were uploaded into Mendix, the changes were committed and saved.

Students notified the instructor via Blackboard, and the assignments were reviewed in Mendix. Grades were entered using the rubric on Blackboard.

The next milestone involved developing a domain model. As the course used a directive pedagogy, the instructor chose the wedding event planning application to be developed and the suggested use of the event app template in Mendix. This proved to be a mistake because the domain modeling used to construct the template was found to be developed primarily to make the template functional, rather than to provide an example of a proper domain model. Students were not able to reconcile this model with their models developed for the project. To preclude this outcome, future iterations of the course will not use a template and will require students to use their own models in their entirety. After completion of this step, students again committed and saved their changes, the instructor reviewed the work, and grading was conducted through Blackboard rubric.

Other reviews included the interim deliverables of interface design and business logic. These reviews were reviewed and graded the same manner as the previous, using Blackboard to grade and reviewing in the Mendix environment.

The typical class agenda included a lecture/discussion, in-class exercise and group project time. During the week before the project presentations, the class did not meet so students could polish their apps and prepare for presentations.

The presentations were professionally delivered, well-structured and concise. Presenters were well versed in the application and the development process and could provide nicely articulated lessons learned. They all agreed that use of Mendix contributed positively to their learning experience and was unanimous in agreeing that Mendix should continue to be used in future offerings.

Following presentations, the students discussed lessons learned. Areas, where they felt the Mendix project contributed to their course experience included the definition of scope, development of work breakdown structures, user stories and database models, developing team communications, and the shared experience of team investment in the project.

Areas that students felt need improvement included application errors, synchronization, communications, and inability to roll back to a historical version of their development model.

There is no question that Mendix added to student repertoires of useful skills. Following certification of the students, the instructor received a communication from an out-of-state company offering certified students paid internships including housing. For those graduating, or for certificate students, they were offering job interviews.

Improvements

As the course progressed, several areas where the student experience could be improved were noted. These areas include both pedagogical and technical areas for improvement.

Mendix has a list of learning objective areas and how Mendix supports a systems analysis course. It is recommended that a similar list should be developed for a project management course. Ensuring that course goals and objectives which can be supported by Mendix would yield better course organization.

Students used an event template to start their projects, with the intent of modifying it to meet their requirements. This did not work as intended. The architecture of the template was designed to make the template functional, which led to integration problems. For example, student developed domain models were not compatible with the domain model in the template. This caused application errors, with an administrative underlying database causing consistency errors, and not allowing students to update their applications. This problem could be mitigated by using a blank project instead of a template.

Using the web modeler and free tier access caused students not to be able to roll back to previously saved versions of their work, which caused them to have to reenter work numerous times. Students who used the desktop modeler did not experience these problems, so these may be associated only with the web modeler. Students also found that synchronization between web modeler and desktop modeler was not always fast or accurate. More investigation into these occurrences must be made to determine whether these issues are related to student inexperience or application error. It is possible that these problems might be avoided by having students use only the desktop modeler.

Students used a variety of options for help with their applications, including Mendix helpdesk,

dedicated support from Mendix academic support department, online forums, and course instructor with mixed results. The process will be standardized as Mendix continues to be used by the program.

5. FUTURE RESEARCH

Mendix is anticipated to be used in an increasing number of SPCS courses in the future. A dedicated course on Mendix development is currently being offered during a six week summer session, as well as a systems analysis course offered during the fall semester. Future research opportunities include using a survey instrument to determine how students feel about the Mendix experience, qualitatively determining how well Mendix supports a systems analysis course and conducting a meta-analysis to provide insight on how Mendix supports projects.

6. SUMMARY

In summary, Mendix was used to support a project management course. The intent was to leverage Mendix to meet goals of the legacy course while adding some new goals and objectives to give students additional competencies using Mendix.

The Mendix project helped students meet course goals and objectives by:

- Exposing them to Mendix, Course Goal 4
- Helping them obtain Mendix certification which is an industry recognized certification allowing students to showcase industry experience and knowledge
- Demonstrating small group development processes
- Facilitating real-world use of project charter, work breakdown structure, etc.
- Supported Course Goals 1, 2 and 3 through participation in a real-world project
- Introducing students to Agile methodologies

Areas for improvement:

- Attaining Mendix certification
- Application of Mendix to achieve real-world objectives in an applied setting
- Inability to reset crashes when using the web modeler
- Unwieldy processes for milestone reviews
- Inconsistency in finding help
- Mapping of course objectives to the tool processes.

7. CONCLUSIONS

Students, instructors, and administrators agree that using Mendix in this course added value in developing student knowledge, skills and abilities. The experience added significant tools to student repertoires but was not without its challenges. However, Mendix has been responsive to problems, both real and perceived. There is no question that the experience added tremendous value, and it is anticipated that with appropriate refinements, Mendix will again be used in succeeding iterations of the course.

8. REFERENCES

FFIEC (n.d.). Computer-Aided Software Engineering. *FFIEC IT Examination Handbook Infobase*. Retrieved from https://ithandbook.ffiec.gov/it-booklets/development-and-acquisition/development-procedures/software-development-techniques/computer-aided-software-engineering.aspx

Frydenberg, M., Yates, D., & Kukesh, J. (2018). Sprint, then Fly: Teaching Agile Methodologies with Paper Airplanes. *Information Systems Education Journal*, *16*(5), 22.

Hailpern, B., & Tarr, P. (2006). Model-driven development: The good, the bad, and the ugly. *IBM systems journal*, *45*(3), 451-461.

Henkel, M., & Stirna, J. (2010, September). Pondering on the key functionality of model driven development tools: the case of mendix. In *International Conference on Business Informatics Research* (pp. 146-160). Springer, Berlin, Heidelberg.

Schmidt, D. C. (2006). Model-driven engineering. Computer-IEEE Computer Society, 39(2), 25.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE software*, *20*(5), 19-25.

Teichroew, D., & Hershey, E. A. (1977). PSL/PSA: A computer-aided technique for structured documentation and analysis of information processing systems. IEEE transactions on software engineering, (1), 41-48.

ISSN: 2473-3857
v4 n4721

Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological bulletin*, *63*(6), 384.

Vincent, P. et al (April 26 2018). Industry Report: Magic Quadrant for Enterprise High-Productivity Application Platform as a Service *ID G00331975*

Yourdon, Ed (Jul 23, 2001). Can XP Projects Grow? *Computerworld,* 35(30), 28.

ISSN: 2473-3857
v4 n4721

Appendix A – Illustrations

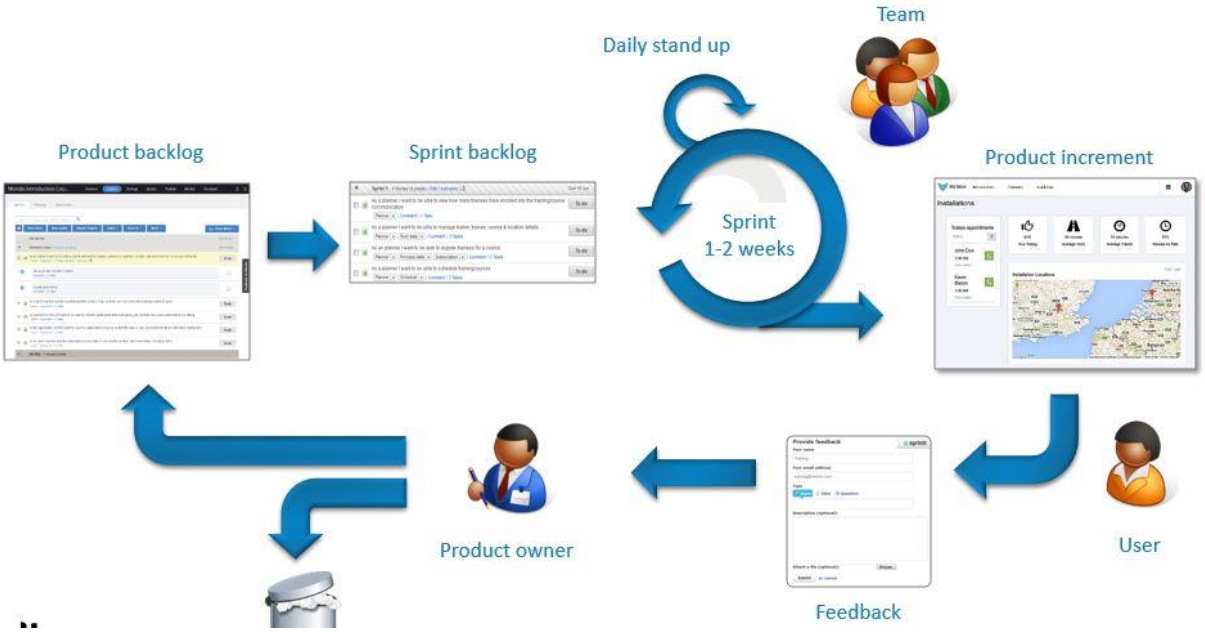

Illustration 1. Agile process overview.

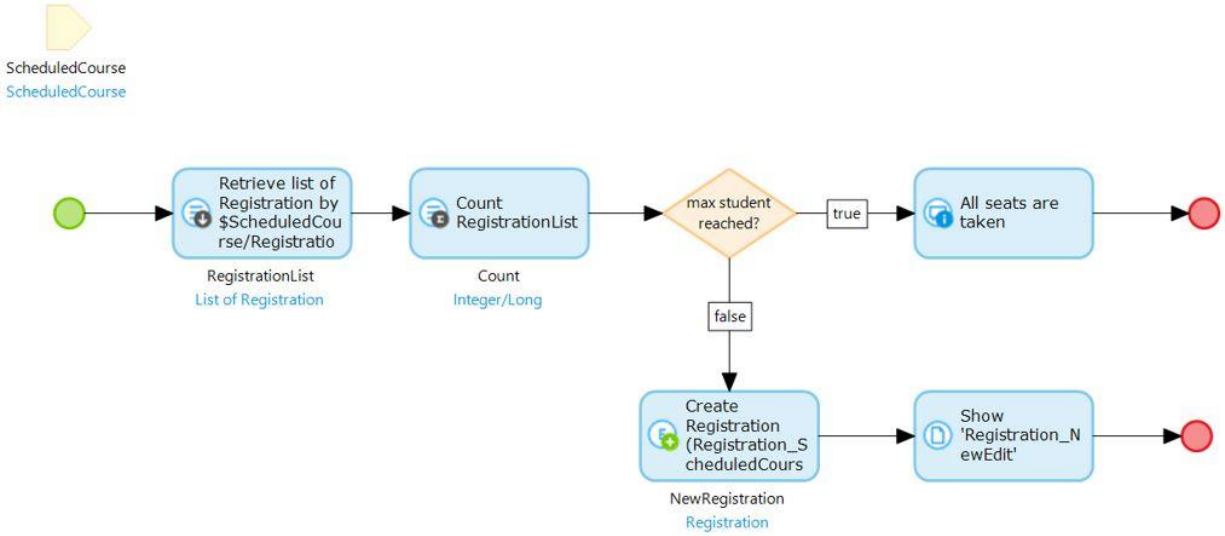

Illustration 2. Microflow example.

ISSN: 2473-3857
v4 n4721

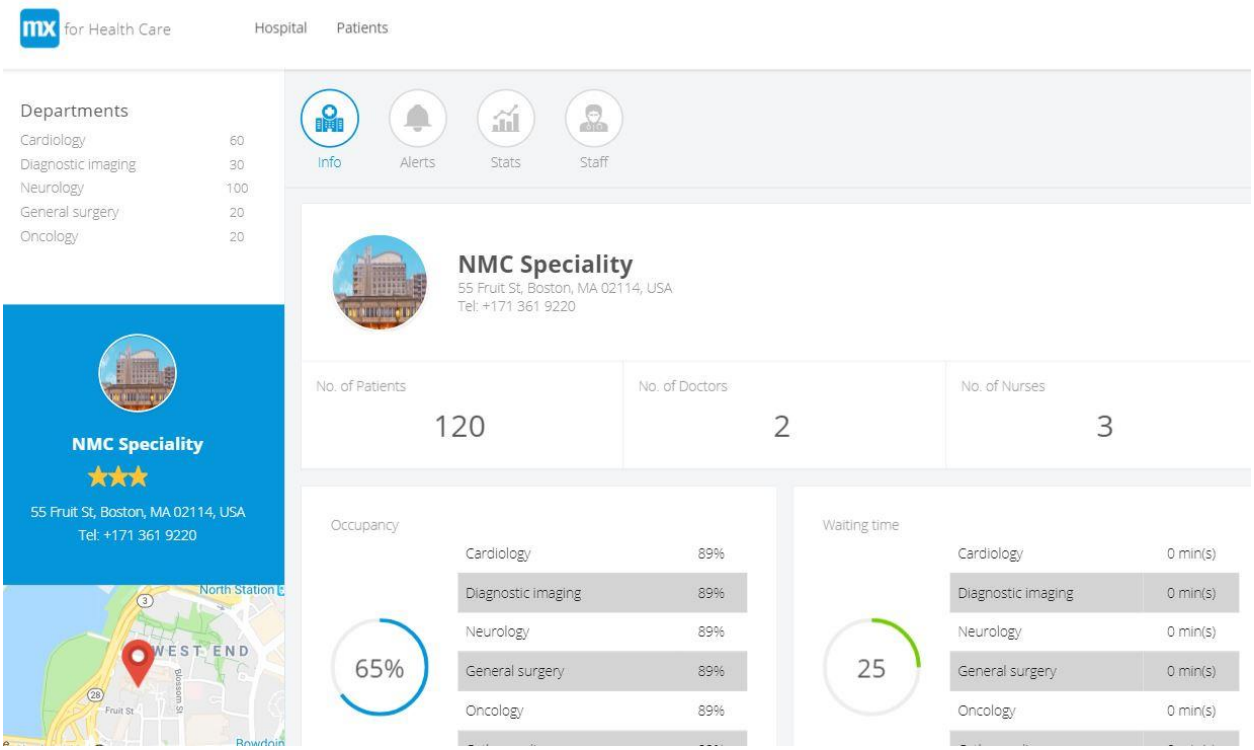

Illustration 3. UI/UX dashboard example image.