Systematic Implementation of ASM (Asset Management System)

Dwi Sari and Alex Elentukh

August 17, 2020

Systematic Implementation of ASM (Asset Management System)

Dwi Puspita Sari, Alex Elentukh
Metropolitan College, Boston University
755 Commonwealth Avenue
Boston, Massachusetts, 02215, United States of America
dsardiyo@bu.edu, elentukh@bu.edu

***Abstract* — Maintaining organizational assets is crucially important for any business. Evaluation of a public company is driven by the value of its assets. Additionally, effectiveness of a company's operation depends on how well its assets are managed. This paper reflects on the semester-long term project completed as part of the graduate course [1] MET CS 633 of Boston University. ASM (Asset Management System) has been implemented, while following the prudent software development process, using Agile / Scrum methodology.**

Keywords—asset management system, agile methodology, scrum software development, pedagogy

I. Introduction

The world of support management systems (such as customer relationship management and IT management) is focused, above everything else, on optimizing client interactions. This observation made by our team concludes that such lopsided and intense focus creates a potential market opportunity. Namely, we could offer a comprehensive asset management over optimized customer relations management. A fitting example could be a company operating within the scope of HVAC industry. In this scenario, it is far more important to utilize a powerful asset management infrastructure, which empowers technicians, over a relationship management system designed to facilitate customer interactions.

This project is not about making a support management system for a niche industry such as HVAC. Instead, our aim is to empower companies to facilitate their very own asset management. The scope of this project is to develop a web-based IT Service Management infrastructure allowing customers to create systems that will record, update, and share assets across their respective organization.

The first word in the title of this paper is "Systematic". Hence, it worth dwelling on the "system" by which this project, as well as other projects, have been undertaken in the context of a graduate course BU CS633. During past five years the course is offered in its current format. Some fifty-six projects were completed by students with a variety of professional interests and backgrounds. The lecture track with theories and methodologies - parallels the practical track with implementation of an actual software system. After a completion of each project, lessons-learned are fed back into theoretical track to fuel changes to methodology. Each next project is able to benefit from experience of previous projects. In fact, the focus of the whole course has shifted. During initial year of running the course, ninety percent of learning attributed to theoretical track. This is contrary to the report from current students, who confirmed that most significant part of their learning came from a term project. Fifty-six completed projects (along with their best and worst parts) represents an extensive body of knowledge from which any student is able to learn. As an example of learning from past misguidance, several projects spent far too much time on definition of personas, while delaying the UI and coding effort. Other projects did not allocate enough time for final verification. In summary, software methodology cannot be copied from somewhere just to be applied immediately. Tailoring to project specifics and nurturing process is a must.

II. Requirements and Planning

A. System Development

All project activities had to be completed within the timeframe of a single semester. Standard deliverables were defined in a Configuration Items List, including the following, scope definition, personas, RASCI, requirements in Pivotal, estimation record, UML diagrams, ERD, use cases, wireframes, and test case. Code was maintained in GitHub. During project, estimation has always been considered a crucial activity. The size of each deliverable, as well as the corresponding effort, were carefully estimated, [2]. Estimation Record, see below, shows the progress from Size to Effort for each project artifact. It is natural for the 'implementation' to consume the largest part of overall effort. At the end of the project, the column with Actual Effort was added. This enabled the team to have a meaningful conversation focusing on gaps between Estimated versus Actual.

TABLE I. Estimation Record

Details		Size Measurement	Size	Effort (Hour)
Requirements	Definition of Users (Personas)	#Roles	10	1
	Definition of Scope	#Requirements	10	1
	Definition of Requirements	#Requirements	20	7
Configuration Management	Configuration Items List	#CL Items	10	2
Estimation	Estimation Record	#Activities	10	2
Design	Use Cases	#Use Cases	25	4
	User Interface	#Wire Frames	25	5
	Database	#ERD	10	4
Peer Review	Issues from Peer Reviews	#Activities	20	15
Implementation Coding	Create Database	#DevOpt	200	8
	Create Database Queries	#DevOpt	50	5
	Create Front-End	#DevOpt	10000	5
	Create Back-End	#DevOpt	20000	5
Test Design	Test Database	#Use Cases	5	1
	Test Front-End	#Use Cases	10	2
	Test Back-End	#Use Cases	10	3
Test Execution	Bug Recording	#Defects	5	3

B. Asset Management System

An "Asset" is commonly defined as a physical item owned by an individual or an organization and used to run business on a daily basis. Asset Management is a formalized approach to maintain assets for the benefit of an organization or a stakeholder [3]. Asset tracking includes monitoring activities to enable a user capturing asset's parameters. Most importantly, asset management allows for various metrics tracking of assets' and hence creates a foundation for logical decision-making [4][5].

Fig. 1. AidVanTech (AVT) Logo

The AidVanTech is an asset management system developed by this team. It maintains the asset parameters for an organization. One has to track physical assets for various purposes, e.g. provisioning, maintenance, financial reporting, etc.

III. Methodology

The traditional Systems Development Life Cycles (SDLC) framework, such as waterfall model - is a plan-based software development with several distinct phases, e.g. planning, analysis, design, and implementation. Agile is a fundamentally different methodology, as it ships incremental chunks of functionality within short iterations. [6][7].

In this project, we thoroughly reviewed and agreed to adopt the four main values of agile manifesto: individuals and interaction over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [8]. Our team had two regular meetings. First, online meeting during a weekend to review the project progress, and second weekly meeting to clear any scope issues regarding detailed deliverables.

This project used Scrum, an agile methodology that emphasizes a team-based, collaborative approach and organizes the work onto series of sprints for incremental delivery [9]. This methodology covers three roles, three artifacts, and five key activities. These three roles are: product owner, scrum master, and development team. The three artifacts: product backlog, sprint backlog, and potentially shippable product increment (PSPI). The five activities include sprint planning, sprint execution, daily scrum, sprint review, and sprint retrospective [10]. During project, Pivotal Tracker was used as an agile project management tool of choice supporting real-time, team-specific collaboration. In addition, Slack was used as an effective communication tool for meetings and individual asynchronous chats.

IV. Result and Discussion

A. Analysis

In the very beginning, the team defined the product goal for each persona (an archetype of a system's user). Table II lists requirements in canonical form for each persona.

TABLE II. Requirements and Personas (User Archetypes)

Requirements	Persona
As a Customer at AidVanTech, I am responsible for registering for service, purchasing a subscription for service, and creating a "System Owner" account which can be used to initiate the process of creating an AMS System.	Customer
As a System Owner at AidVanTech, I am responsible for creating an AMS-User, initiating the process of creating an AMS System, creating, updating and deleting tables and columns from the management system data, adding or removing as well as managing and maintaining all AMS-Users who have access to the AMS System, and determining data categories that defube data on the AMS System.	System Owner
As AMS-User at AidVanTech, I am responsible for updating and maintaining Asset Management System Data, and running queries on Asset Management System data.	AMS-User
As a Database Administartor at AidVanTech, I am responsible for updating and maintaining the database that houses the Asset Management System, resolving the customer database issue and updating the Support Technician with the progress, and updating and maintaining the database that houses the AMS.	Database Admin
As a Supprt Technician at AidVanTech, I am responsible for receiving any technical issues of an asset tracker management system from customer, fixing the customer technical issue and updating the customer with the progress, and passing customer's database issues to Database Administrator.	Support Technician

Based on the analysis, there are 5 different personas for AidVanTech divided into two categories: internal members (e.g. Database Admin) and external members (e.g. Customer, System Owner, AMS-User) .

Each persona is detailed in Table III. Definition of Personas for AVT, see below. Note that a persona could be both human and non-human, e.g. AMS - Asset Management System (API).

TABLE III. Definition of Personas

Persona	Description
Customer	Any person interested in purchasing a subscription to our service.
System Owner	A paying customer with the ability to creation an Asset Management System (AMS) as well as the ability to manage and maintain the subscription, the AMS System itself and any users associated with that AMS system.
AMS-User	A customer created user account designed for the purpose of managing and maintaing the AMS Data.
Database Administrator	An internal member who is responsible for managing and maintaining the health of the Database that houses all AMS Customers.
Support Technician	An internal member of our team, who is responsible for providing support to Customers, System Owners and Database Administrators.

RASCI (Responsible, Accountable, Supporting, Consulted, Informed) Matrix was used to further define the responsibilities and assignments for each user engaged with ASM. Table IV. RASCI Chart below has a summary of all high-level tasks. Note that tasks in RASCI are much more broad than stated requirements.

TABLE IV. Rasci Chart

Task	C	SO	AU	DA	ST	AMS
Register for service	R	-	-	I	I	I
Purchase subscription	R	-	-	I	I	I
Create system owner account	R	-	-	I	I	I
Create AMS via AMS creating tool	I	R	-	C	C	S
Create AMS-User	I	R	I	C	C	S
Adds and/or remove tables and columns from management system data	I	R	I	C	C	S
Determine data types the system will manage	I	R	I	C	C	S
Determine data categories that define data	I	R	I	C	C	S
Update and maintain asset management system data	I	C	R	C	C	S
Update and maintain the database that houses the asset management systems	I	I	I	R	S	I
Run query on asset management systems data	I	A	R	C	C	S

C = Customer
SO = System Owner
AU = AMS-User
DA = Database Administrator
ST = Support Technician
AMS = Asset Management Systems (API)
R = Responsible: Owns the task
A = Accountable: Whoever responsible accounts to
S = Supporting: supports the task
C = Consulted: Has the capability to complete the task
I = Informed: Must be notified of the results

B. Design

An Entity Relationship Diagram (ERD) below describes the AVT's data model and the detailed relation between one entity and another. The AVT's ERD diagram is shown below in Figure 2. ERD Interface.

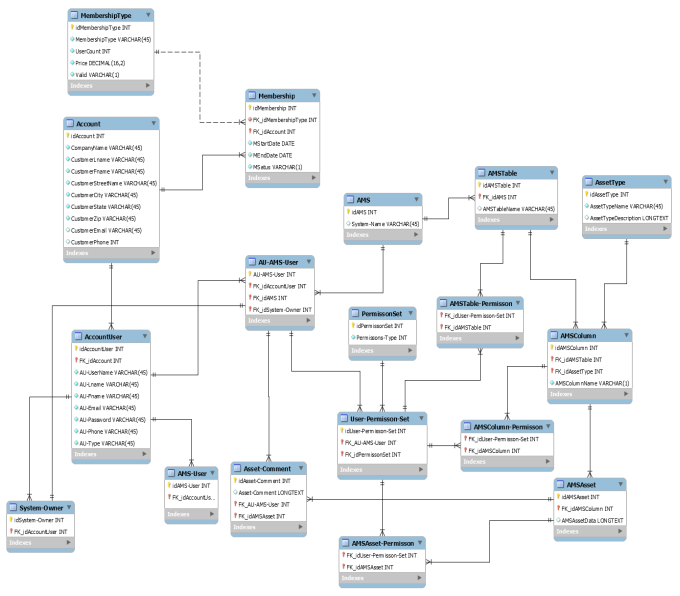


Fig. 2. ERD interface

Design process utilized UML to prototype the system. Each Use Case is based primarily on RASCI role definition, with references to ERD database schema, as well original requirements. These Use Case diagrams summarize the logic of the application in relation to the underlying database for specific usage scenarios. To help with understanding the logic each Use Case is accompanied by a description.

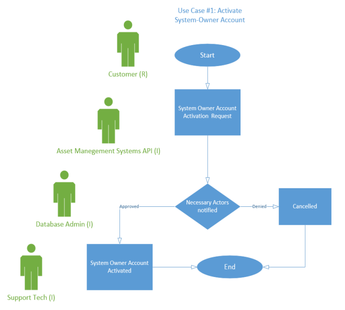

Use Case #1: Activate System-Owner Account
Customer (R)
Start
System Owner Account Activation Request
Asset Manegement Systems API (I)
Database Admin (I)
Approved
Necessary Actors notified
Denied
Cancelled
Support Tech (I)
System Owner Account Activated
End

Fig. 3. Use Case #1 Activation Process

Use Case #1 addresses activating the System-Owner Account and mirrors the users involved with the corresponding RASCI task. This use case diagram references the roles each persona will have in this particular use case and how their involvement aligns with the state transitions of the application's logic. In this case the customer interacts the app to submit a request to activate their account via the AMS-API. The Database admin either will approve or deny this request leading to either System Owner account activation or request cancellation.

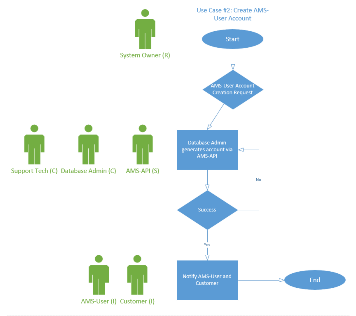

Use Case #2: Create AMS-User Account
System Owner (R)
Start
AMS-User Account Creation Request
Support Tech (C)
Database Admin (C)
AMS-API (S)
Database Admin generates account via AMS-API
No
Success
Yes
AMS-User (I)
Customer (I)
Notify AMS-User and Customer
End

Fig. 4. Use Case #2 AMS-User Creation

Use Case #2 addresses the creation of an AMS-User account, where the System Owner submits a creation request via the AMS-API, which is either approved or denied by the Database Admin. Upon approval, the AMS-User and Customers associated with the request will receive a notice from the API system.

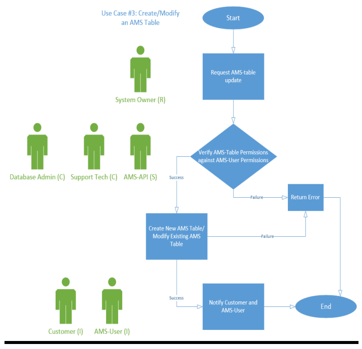

Use Case #3: Create/Modify an AMS Table
Start
System Owner (R)
Request AMS-table update
Database Admin (C)
Support Tech (C)
AMS-API (S)
Verify AMS-Table Permissions against AMS-User Permissions
Success
Failure
Return Error
Create New AMS Table/ Modify Existing AMS Table
Failure
Success
Customer (I)
AMS-User (I)
Notify Customer and AMS-User
End

Fig. 5. Use Case #3 Creation and Modification of AMS Tables

Use Case #3 addresses the creation and modification (modification includes deletion) of AMS tables within the database. The System Owner submits a table update, which would then be verified against the AMS-Table Permissions and AMS-User Permissions database fields. If permissions match, then the table creation/update will be completed with the notification to relevant users; if not the system will return an invalid request error.

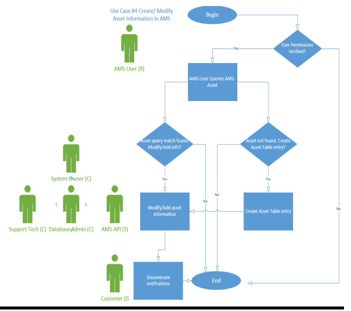

Use Case #4 Create/ Modify Asset Information in AMS
Begin
AMS-User (R)
User Permissions Verified?
Yes
AMS-User Queries AMS-Asset
Asset query match found. Modify/Add info?
Asset not found. Create Asset Table entry?
System Owner (C)
Yes
Yes
No
No
No
Support Tech (C)
Database Admin (C)
AMS-API (S)
Modify/Add asset information
Create Asset Table entry
Customer (I)
Disseminate notifications
End

Fig. 6. Use Case #4 Verification Permissions Process

In Use Case #4 we see the modification/creation of asset information within the AMS. The AMS-User begins by verifying their permissions against the AD database tables. Once verified the AMS-User submits a query for a particular AMS-Asset; and this asset is either found or not found. If found, the AMS-User will be asked to modify data; if asset is not found, the AMS-User will be asked to create an asset table entry. Once one of these is selected, the modification/creation will be completed triggering notifications to be sent to the Database Admin, System Owner, and Support Tech through the AMS-API.

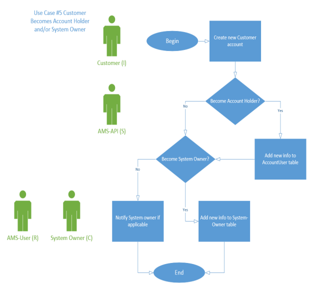

Use Case #5 Customer Becomes Account Holder and/or System Owner
Customer (I)
AMS-API (S)
AMS-User (R)
System Owner (C)
Begin
Create new Customer account
Become Account Holder?
No
Yes
Add new info to AccountUser table
Become System Owner?
No
Yes
Notify System owner if applicable
Add new info to System-Owner table
End

Fig. 7. Use Case #5 Account Holder Assigning Process

Use Case #5, the scenario is that of the Customer becoming the Account Holder and/or the System Owner. The Customer initiates new account creation and the AMS-API prompts the Customer to become an Account Holder. If confirmed, the new data is added to the Account User table. Once this is complete, or if the Customer elects not to become an Account Holder, the Customer will be prompted to become a System Owner. If confirmed the System-Owner database table will be updated with the new information. If not, the proper System Owner will receive a notification that there is a new Account Holder.

C. *Human Interface Design*

Key wireframes focusing on basic functionality and user process-flow are shown below. They demonstrate the process for signing up a new customer, AMS-User, or System Owner. They also show the email invitations received by a user and the location of the wizard to enter their specific asset information. The system can be customized for whatever information a user needs to track. In this case, the system owner has full control of the data asset management activities. He/she is able to add assets, users, and permissions, as well as, customize the system for a specific user needs. See below, Figure 8 called Dashboard Interface Design for AMS-System. However, system owner has no authorization to set permissions or add other users to the system.

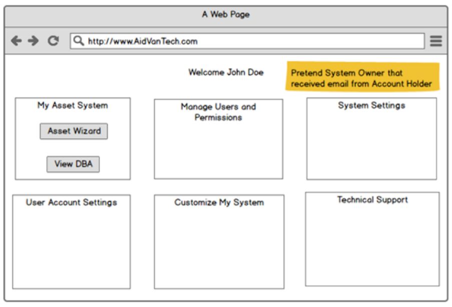


Fig. 8. Dashboard Interface Design for AMS-System

Figure 9. Input and Update Directory for AMS-System shows that both AMS-User and System Owner are able to examine their asset system and edit directly the asset, any specific way each user needs.

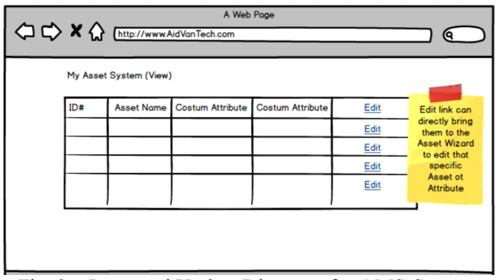
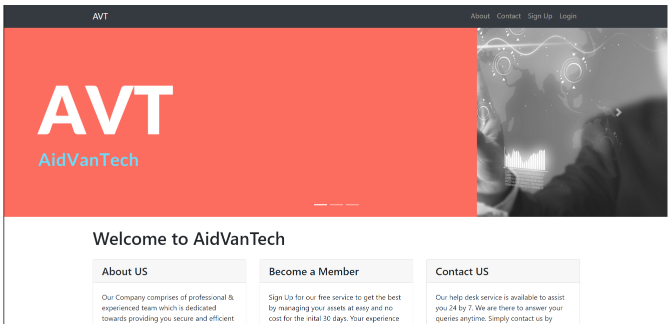

A Web Page
http://www.AidVanTech.com
My Asset System (View)
ID#
Asset Name
Costum Attribute
Costum Attribute
Edit
Edit
Edit
Edit
Edit
Edit link can directly bring them to the Asset Wizard to edit that specific Asset ot Attribute

Fig. 9. Input and Update Directory for AMS-System

During project retrospective, it was important to examine how wireframes were morphed into actual system screenshots. Wireframes in BalsamIQ use no color and have yellow sticky notes to explain functionality. The layout for AVT System homepage is shown in Figure 10.



Welcome to AidVanTech

About US

Our Company comprises of professional & experienced team which is dedicated towards providing you secure and efficient

Become a Member

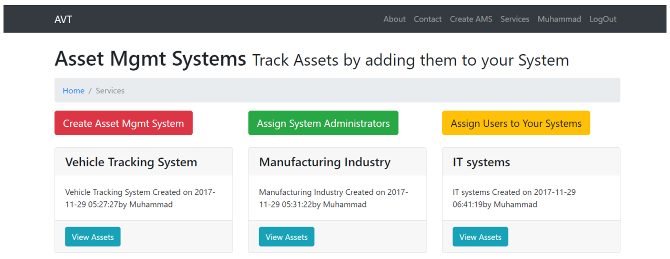
Sign Up for our free service to get the best by managing your assets at easy and no cost for the inital 30 days. Your experience

Contact US

Our help desk service is available to assist you 24 by 7. We are there to answer your queries anytime. Simply contact us by

Fig. 10. The Layout Design for AVT System Homepage

The layout design for AVT System Dashboard Menu is shown in Figure 11.



Asset Mgmt Systems Track Assets by adding them to your System

Vehicle Tracking System

Vehicle Tracking System Created on 2017-11-29 05:27:27by Muhammad

Manufacturing Industry

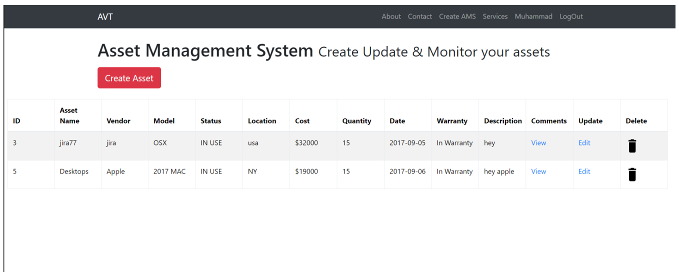
Manufacturing Industry Created on 2017-11-29 05:31:22by Muhammad

IT systems

IT systems Created on 2017-11-29 06:41:19by Muhammad

Fig. 11. The AVT System Dashboard Menu

The layout design for AVT System Page Directory for AMS-System is shown in Figure 12.



Asset Management System Create Update & Monitor your assets

ID	Asset Name	Vendor	Model	Status	Location	Cost	Quantity	Date	Warranty	Description	Comments	Update	Delete
3	jira77	jira	OSX	IN USE	usa	$32000	15	2017-09-05	In Warranty	hey	View	Edit	🗑
5	Desktops	Apple	2017 MAC	IN USE	NY	$19000	15	2017-09-06	In Warranty	hey apple	View	Edit	🗑

Fig. 12. Monitoring Page Directory for AMS-System

D. *Implementation (Coding)*

The design of AVT System (elaborated in previous sections) was implemented as a set of programs using PHP programming language. In addition, there are several tools were used to develop the system including: MySQL.

E. *System Testing and Verification*

Table V. Test Cases, see below, lists several key test case that have been documented and executed.

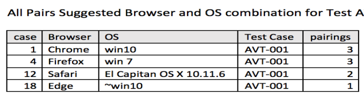
TABLE II. TEST CASES

Test Case ID	Test Description
AVT-001	Verify customer can signup
AVT-002	Verify customer can invite use to create a system owner account
AVT-003	Verify system owner can create an AMS user account
AVT-004	Verify system owner can create an asset table
AVT-005	Verify AMS-User can create an asset

During the testing phase, this group followed the black-box testing method of software verification, where the testing was fully focused on the functionality of the AVT System.

Before shipping the product for each sprint, the test was conducted. Testing was performed to improve reliability of the program by finding and removing errors. This phase started with the fundamental assumption that program contained errors and the main purpose of this phase was to find as many errors as possible [10]. The AVT-001 test case is elaborated in the Table VI. Test Cases Details below.

TABLE III. TEST CASES DETAILS

AVT-001	
General Information	
Test Type: Functional - Positive	
Test Type: AVT-001	*Test Date:* 12/06/2017
Test Case Description: Verify customer can signup.	
Expected Results: User successfully registers.	
Requirements to be Tested: As a customer at AidVanTech, I am responsible for registering for service.	
Setup and Constraints: Computer with internet access is available to the user, user is not registered.	
Input: User information	
Procedural Steps: 1. Navigate to AidVanTech website 2. Click "Signup" 3. Input "Full Name, Email Address, Username, Password", then Click Submit	
Expected Results: 1. AidVanTech website loads 2. User signup form should display 3. User is registered and subscription selection page is displayed	
Actual Results: 1. AidVanTech website loads 2. User sigup form should display 3. User is registered and subscription selection page is displayed	
 All Pairs Suggested Browser and OS combination for Test AVT-001	

case	Browser	OS	Test Case	pairings
1	Chrome	win10	AVT-001	3
4	Firefox	win 7	AVT-001	3
12	Safari	El Capitan OS X 10.11.6	AVT-001	2
18	Edge	~win10	AVT-001	1

V. CONCLUSION

The capability for a business organization to keep track of their assets is vital for resources provisioning and planning. During the project, the team learned the importance of systematic development of any asset management system.

An important aspect of this project is collaboration among team members. As a point of fact, this paper is being presented in Inna Prat, while reflecting the effort undertaken in Boston, which is a fifteen thousand kilometers away. Software development has become a global affair. One has to prepare for a collaboration, as it takes a certain learning to appreciate and improve effectiveness of team dynamics.

Much like the software methodology has improved dramatically over five years of running this university course. Any software organization has to strive toward its razor-sharp goals. Agile methodology cannot be copied from somewhere just to be applied immediately. Tailoring to project specifics and nurturing of software process is a must.

ACKNOWLEDGMENT

We would like to share our gratitude to LPDP (Indonesian Endowment Fund for Education) Scholarship funding, Indonesia Ministry of Finance.

We would like to acknowledge the contribution of all team members of MET CS633 (Software Quality and Security Management Course Fall 2017).

Should note that seven members of this project team have had a variety of backgrounds. Although the Scrum Guide [12] defines three roles only for a team. As a starting point, each team member naturally gravitated to those tasks that he / she is most familiar with. Such assignment of tasks, according to a personal preference - is a part of the process tailoring that is unavoidable. Jeff Sutherlad, the author of Scrum Guide, calls it "situational adaptation". We treated Scrum Guide as a common vocabulary to map our specific activities, as follows.

– Dwi Puspita Sari and Isaque Rezende - (Product Owner) maintaining requirements in Pivotal Tracker, icebox / backlog
– David Fortin - (Scrum Master) bringing the team together, alleviating impediments, tracking actions
– Andrew Guinther - UML prototyping and analysis
– Andreal Leak - UI Design, wireframing, BalsamIQ
– Muhammad Kamran - coding, PHP, MySQL, GitHub
– Alok Asoor - test cases development and execution, system verification
– Alex Elentukh - (Agile Coach) university professor teaching the class, responding to questions, clarifying deliverables, providing feedback

REFERENCES

[1] BU MET CS 633 "Software Quality and Security Management", Alex Elentukh, https://www.bu.edu/csmet/cs633/

[2] Steve McConnell, Software Estimation, Washington, United States of America: Microsoft Press, 2006.

[3] R. Davis, An Introduction to Asset Management, United Kingdom: The Institute of Asset Management, 2008.

[4] D.P. Sari, S.J. Putra, and E. Rustamaji, "The development of project monitoring information system (Case study: PT Tetapundi Prima Kelola)," IEEE Conference, pp. 39-43, 16 February 2015 [The 3rd International Conference on Information Technology for Cyber and IT Service Management (CITSM 2014)].

[5] World Shelter People, Monitoring Information System, retrieved 15 March 2018, http://www.fukuoka.unhabitat.org/docs/publications/pdf/peoples_process/ChapterVII-Monitoring_Information_System.pdf.

[6] A. Dennis, B.H. Wixom, and R.M. Roth, System Analysis and Design (Fourth Edition), United States of America: John Wiley & Sons, Inc, 2012.

[7] Cockburn, A, Agile Software Development, Boston: Addison-Wesley, 2001.

[8] Agile Alliance, Manifesto for Agile Software Development, retrieved 15 March 2018, https://www.agilealliance.org/agile101/the-agile-manifesto/.

[9] Tycho Press, Scrum Basics, Berkeley, California: Tycho Press, 2015.

[10] K. S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, United States of America: Pearson Education, Inc., 2013.

[11] G. J. Myers, The Art of Software Testing, United States of America: John Wiley & Sons, Inc., 1979

[12] Jeff Sutherland and Ken Schweber, Scrum Guide, November 2017