Journal of Information Systems Engineering and Management
2020, 5(1), em0110
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

MODESTUM
OPEN ACCESS

Development of Business Activity Monitoring Application to Increase Competitiveness: A Case Study

Charles Martino [1*], Johanes Fernandes Andry [1]

[1] Faculty of Technology and Design, Universitas Bunda Mulia, Jakarta, INDONESIA
***Corresponding Author:** charlesmartino451999@gmail.com

Citation: Martino, C., & Andry, J. F. (2020). Development of Business Activity Monitoring Application to Increase Competitiveness: A Case Study. *Journal of Information Systems Engineering and Management, 5*(1), em0110. https://doi.org/10.29333/jisem/7823

ARTICLE INFO

Published: 6 Mar. 2020

ABSTRACT

As the number of internet users goes up, Internet Service Provider (ISP) companies will compete to gain new customers. An ISP company has difficulty in tracking their client request's progress, that result in poor customer service, which in turn lead to bad review about the company. This along with their internal communication problem bring a negative impact on their competitiveness. A Business Application Monitoring (BAM) application was proposed to solve their problems. This BAM will track all of their business activities including clients' requests and allow communication per activity. The development of the BAM uses Extreme Programming (XP) methodology to reduce development time while keeping the software quality. The development process consists of five parts, namely planning, design, coding, testing, and software increment. The result of this research shows that XP greatly reduces the development time while still able to maintain code quality. The company also able to track all of their business activities and allow a more controlled communication within an activity; which resulting in reduced bottlenecks on service delivery time, improved customer service quality by allowing activities status continuously informed to the customer, and increased competitiveness to compete against a stronger competitor.

Keywords: business activity monitoring, development, extreme programming

INTRODUCTION

As technology advances, the internet became more of a need rather than a desire (Apăvăloaie, 2014), more and more people need access to the internet. World Bank shows 48.5% of the world population and 32.3% of the Indonesian population already using the internet in 2017 (The World Bank, 2017). This number has grown over the years and is expected to keep increasing. The increasing need for the internet means more customers for Internet Service Provider (ISP) companies. To take advantage of this, ISP companies must compete with each other to gain new customer loyalty (Yeboah & Ewur, 2014).

An ISP company in Jakarta has some problems in its business process that affect their competitiveness in the market. When a customer requests a service (e.g., installation, upgrade, or termination), most divisions do not know about the actual progress of the request once they complete their task and send the request to the next division. If there is any delay happen, it is really hard to track the cause for the delay. The customer cannot know their request progress, resulting in bad customer service, especially when their request is delayed. If this happens, the customer may have lower satisfaction towards the company and may spread bad reviews about the company brand, thus becoming a competitive disadvantage (Adewale, 2016).

Another problem is there are a lot of times where a division request another division (e.g., design or purchasing) for some items. But after a few days, the requested division still has not notified the requesting division about the status of the requested item, slowing down and making it harder for the requesting division to make decisions or take actions (e.g. wait for the request to complete or use an outsource service to make the design). This often happens when the requested division has too many tasks assigned to them and forgot to notify the requester.

The communications between divisions about requests or any business activities in the company mostly done using email and instant messaging service. This method results in another issue, where when someone wants to find a certain message, it may take some time, and the chat is also often cluttered with other business activities or even other topics.

The first problem revolves around the tracking of tasks in business activity, while the second problem mostly caused by the lack of communication and information about other divisions' tasks. Both of these problems can be solved using a Business Activity Monitoring (BAM) application (software) that tracks all business activities and allow communication between divisions. Information about an activity can be shown to all people related to the activity as well as to the management, admin, and customer service, while the number of activities can be shown to all divisions. This way, customer can know precisely how far their request

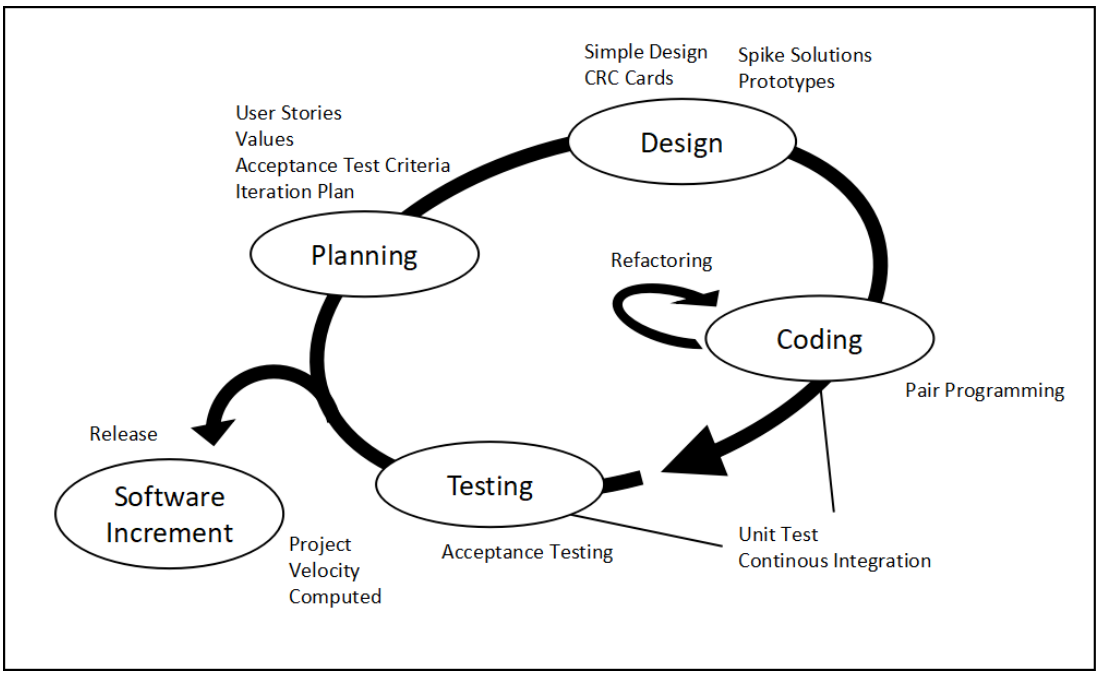

Figure 1. XP Methodology (Suryantara, et al., 2018)

is processed and allow management to know where a delay happens and can take action accordingly, such as hire more staff to the division or do an employee discipline training, and allow a division to determine what to do based on the number of activities currently assigned to another division. As for the communication, a chat feature separated for each request will be implemented. This feature will only accessible for the people related to the request. This will also address the issue of using email and instant messaging service as communication platforms.

For the BAM development, Extreme Programming (XP) methodology is used to develop the software. This method is chosen so researchers can develop the software in a shorter time frame (Taibi, et al., 2017) while keeping the quality of the software (Andry, et al., 2018). Other reasons for choosing this methodology are the size of the researchers' team and the ability to adapt to new requirements.

THEORETICAL BASIS

A BAM is aimed to provide real-time information about the status and results of various business operations, processes, and transactions (Nafie & Eltahir, 2016). So that managers can know what is happening in the company and make decisions rapidly and effectively (Ouassarah, 2015).

XP is a set of software engineering practice that was developed by Kent Beck back in 1999 (Özcan-Top et al., 2018). XP emphasizes teamwork between managers, customers, and developers. The main values of XP are communication, simplicity, feedback, respect, and courage (Tonin et al., 2017).

METHODOLOGY

The methodology used for developing the software is XP. XP is divided into five steps which are shown in **Figure 1**, namely (Suryantara & Andry, 2018):

1. Planning. In this step, researchers learn the business processes, listen to users' stories, define software function and output, and estimate development time and cost.
2. Design. In this step, researchers design the database, user interface, and the software class structure.
3. Coding. In this step, researchers create a database, software, and unit test for each class; along with software documentation and user guide.
4. Testing. In this step, researchers do tests to make sure the software matches the user requirement.
5. Software Increment. In this step, the software is released to the user.

The BAM will split each process into smaller sub-processes with a division responsible for each sub-process. The process will be referred to as activity and the sub-process will be referred to as task from now.

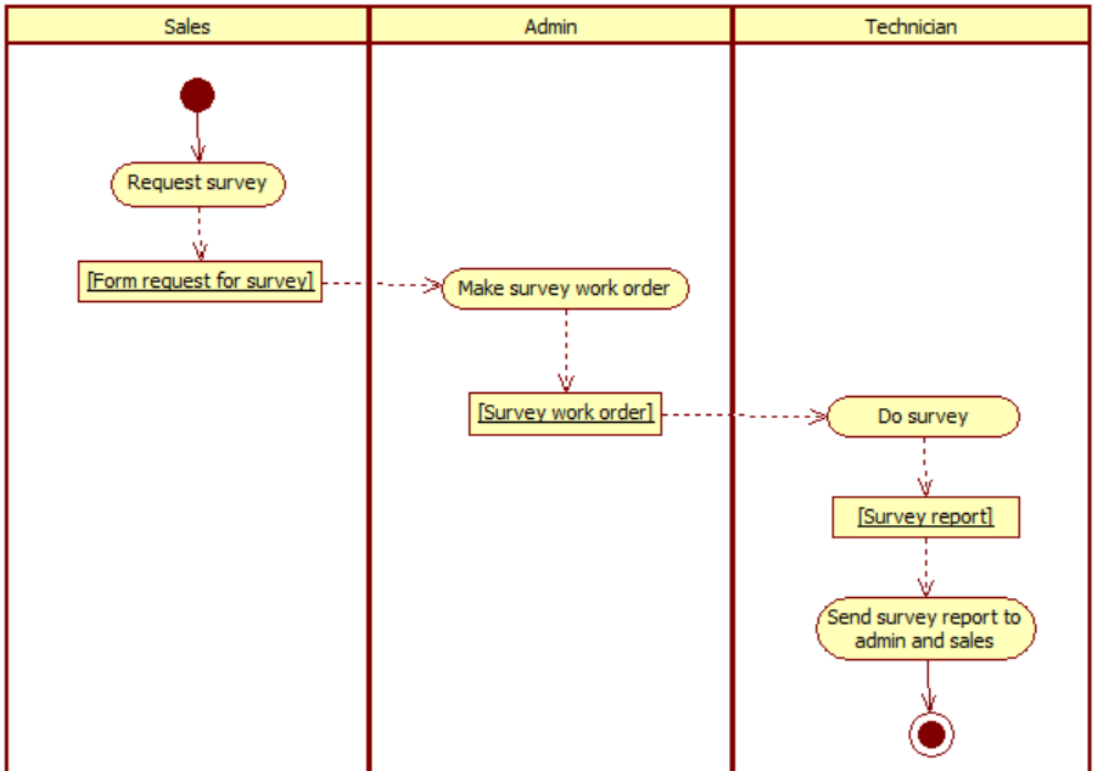

Figure 2. Request for Survey Process

RESULT AND DISCUSSION

Planning

Learn the business process

Researchers learn the company business process by interviewing with the company admins and observation. The reason for choosing admins as the interviewees are because all of the company processes will require their admin validation making them know all of the processes in the company. One of their business processes is a request for a survey by a client that may want to install internet service. This process can be seen in **Figure 2**.

Listen to users stories

Researchers gather stories from multiple users, such as admins, a director, and some staff. In this step, researchers also analyze the stories gathered to define the software requirements and acceptance criteria. One of the user stories from an admin is:

"As a user, I want the form does not reset when a submission failed, like when the form does not pass the form validation; it is annoying to keep filling the form over and over, especially when the form has many fields to fill."

The acceptance criteria for the story above is the software must keep the field value even when the submission failed for any kind of cause.

The requirements for the software are defined during each user story phase over several iterations. The feature for each iteration will be discussed later in point E of this discussion.

Define software function and output

Based on the stories gathered in the previous step, researchers able to define the required software functions and outputs.

Some of the important functions are as follow:

- Clients need to be validated by admin before they can be assigned to any activity.
- All users should be able to see all divisions' numbers of the currently assigned tasks.
- A user can only create a new activity if his division and job level match with the activity's first task.
- Users must be able to upload files and images in chat.
- Users must be able to reject a task if the previous task does not provide a required document correctly, and the previous user must be able to update the uploaded document if it happens.

Figure 3. Development Iteration

Table 1. Mapping of Views

No	View Name	Description
1	login	UI login
2	dashboard	UI dashboard
3	notification	UI notification page
4	master/user	UI user master
5	master/job_level	UI job level master
6	master/job_title	UI division master
7	master/role	UI role master
8	master/company	UI company master
9	master/client	UI client master
10	master/client_title	UI client title master
11	master/city	UI city master
12	master/activity	UI activity master
13	master/task	UI task master
14	master/activity_task	UI activity task master
15	master/activity_doc	UI activity document master
16	master/priority	UI priority master
17	master/email_template	UI email template master
18	transaction/progress	UI activity transaction
19	transaction/progress_detail	UI activity task transaction
20	transaction/progress_doc	UI activity document transaction
21	transaction/task_activity	UI activity related to a user
22	report/activity	UI activity report

- Users can see their current assigned tasks, tasks that may be assigned to them, completed tasks, and completed activities.

The software outputs are as follow:

- Report on the number of tasks and time taken to complete all tasks for all divisions.
- Report of currently assigned tasks for each user.
- Report on tasks in a user's waiting list.
- Report of tasks that have been completed by a user but the activity itself has not completed.
- Report on all activities.

Estimate development time and cost

Researchers estimate the time needed to complete the project based on the development difficulty. The estimated time for the project is 3 months and the development planned to be divided into 5 iterations as in **Figure 3**. The process of listening to users' stories, design, coding, and testing is done in each iteration. The cost was mostly estimated from the hardware and services needed for the software development and implementation, such as internet service, server, and electricity upkeep.

Design

The software is designed with the Object-Oriented Programming (OOP) paradigm and Model-View-Controller (MVC) architecture in mind. This design is chosen to separate concerns and ease the coding and testing phase. The software is built as web-based software, so it can have high portability and does not heavily depend on the user's device.

Class-Responsibility-Collaborator (CRC) card is used to design the software. CRC card is chosen because it makes the changing of the software design easy (Tabassum et al., 2017) and is suitable with OOP (Suryantara, et al. 2018). Each card represents a class in the software.

The following table shows the views of the software, see **Table 1**. The views are not in a class form, instead, it is just HTML files that function as a user interface.

The mapping of CRC cards and model classes can be seen in **Table 2**. The model classes do not have any collaborators since models do not 'care' about any other classes and their primary function is to alter and fetch data from the database.

The mapping of CRC cards and controller classes can be seen in **Table 3**. The controller classes have views and models as their collaborators since controllers function as a bridge and chose what views and models to use based on user input.

Table 2. Design CRC and Mapping of Models

No	CRC Design	Class Mapping
1	Account Model	Account_Model
2	Dashboard Model	Dashboard_Model
3	Notification Model	Notification_Model
4	User Model	master/User_Model
5	Job Level Model	master/Job_Level_Model
6	Division Model	master/Job_Title_Model
7	Role Model	master/Role_Model
8	Company Model	master/Company_Model
9	Client Model	master/Client_Model
10	Client Title Model	master/Client_Title_Model
11	City Model	master/City_Model
12	Activity Model	master/Activity_Model
13	Task Model	master/Task_Model
14	Activity Task Model	master/Activity_Task_Model
15	Activity Doc Model	master/Activity_Doc_Model
16	Priority Model	master/Priority_Model
17	Email Template Model	master/Email_Template_Model
18	Activity Transaction Model	transaction/Progress_Model
19	Activity Task Transaction Model	transaction/Progress_Detail_Model
20	Activity Document Transaction Model	transaction/Progress_Doc_Model
21	Assigned To Me Model	transaction/task_activity_sub/Assigned_To_Me_Model
22	Waiting List Model	transaction/task_activity_sub/Waiting_List_Model
23	On Progress Model	transaction/task_activity_sub/On_Progress_Model
24	Resolved Activity Model	transaction/task_activity_sub/Resolved_Activity_Model
25	Activity History Model	transaction/task_activity_sub/History_Model
26	Activity Chat Model	transaction/task_activity_sub/Chat_Model
27	Activity Report Model	report/Activity_Model

Table 3. Design CRC and Mapping of Controllers

No	CRC Design	Class Mapping
1	Account	Account
2	Dashboard	Dashboard
3	Notification	Notification
4	User	master/User
5	Job Level	master/Job_Level
6	Division	master/Job_Title
7	Role	master/Role
8	Company	master/Company
9	Client	master/Client
10	Client Title	master/Client_Title
11	City	master/City
12	Activity	master/Activity
13	Task	master/Task
14	Activity Task	master/Activity_Task
15	Activity Doc	master/Activity_Doc
16	Priority	master/Priority
17	Email Template	master/Email_Template
18	Activity Transaction	transaction/Progress
19	Activity Task Transaction	transaction/Progress_Detail
20	Activity Document Transaction	transaction/Progress_Doc
21	Task Activity	transaction/Task_Activity
22	Assigned To Me	transaction/task_activity_sub/Assigned_To_Me
23	Waiting List	transaction/task_activity_sub/Waiting_List
24	On Progress	transaction/task_activity_sub/On_Progress
25	Resolved Activity	transaction/task_activity_sub/Resolved_Activity
26	Activity History	transaction/task_activity_sub/History
27	Activity Chat	transaction/task_activity_sub/Chat
28	Activity Report	report/Activity

An example of controller CRC can be seen in **Figure 4**, which shows the User controller. This controller has a view and some models as its collaborators. The models are used to provide data list in the view.

Document Name	CRC Design	
Application Name	Business Activity Monitoring	
Activity	CRC Design	
During	28/05/2019 - 21/05/2019	
No	**CRC**	**Description**
4	**User** **Superclass** : - **Subclass** : - **Responsibilities** / **Collaborators** Intepret user input in master user view / master/user Display master user view / User Model Send changes to User Model / Job Level Model Division Model Company Model Role Model Notification Model	Controller class to handle master user

Figure 4. User Controller CRC

Table 4. Database

No	Table Name	Description
1	msuser	Store users
2	msjoblevel	Store job levels
3	msjobtitle	Store divisions
4	mscompany	Store company branches
5	msclienthd	Store clients
6	msclienttitle	Store client titles
7	mscity	Store city names
8	msactivity	Store business activities
9	mstask	Store tasks / business activity sub-process
10	msactivitytask	Store activity and task relations
11	msactivitytaskdoc	Store task document names
12	mspriority	Store activity priorities
13	msemailtemplate	Store email templates to send to customer
14	trclientprogresshd	Store activities that have started
15	trclientprogressdt	Store tasks from activities that have started
16	trnotificationlog	Store notifications about tasks
17	trclientprogressdtuser	Store assigned user of each task
18	trclientprogressdtdoc	Store task's uploaded document paths
19	trclientprogressinfo	Store chat message
20	trclientprogressinfodoc	Store chat uploaded file paths
21	msrole	Store roles for access control
22	msmenu	Store menus for access control
23	msrolemenu	Store role and menu relations

Coding

In this step, unit tests, database, and software are created. The unit tests are created before the software code. Every time a new class made or an existing class re-factored, the unit test for the class runs to verify the class still has the same functionality.

The following table shows the database tables used for the software, see **Table 4**.

Below is the structure of some database tables, shown in **Figure 5**. Most of the tables in the database have created by, created date, modified by, and modified date field. This is to record the person that create and the last person that modifies the entry, so if there is any anomaly happen or wrong data inserted, the company can know who did it.

Document Name	Table Design
Application Name	Business Activity Monitoring
Activity	Table Design
During	01/06/2019 - 10/06/2019

No	Table	Description
11	Field / Type / Comment ActivityID varchar(50) ActivityName varchar(100) Duration int(10) Priority varchar(50) CompanyId varchar(50) Remark varchar(500) FgActive varchar(1) CreatedBy varchar(50) CreatedDate datetime ModifiedBy varchar(50) ModifiedDate datetime	Structure of msactivity table
17	Field / Type / Comment ClientProgressId varchar(50) ClientId varchar(50) ActivityId varchar(50) PriorityId varchar(50) Description varchar(8000) Attachment varchar(255) JobTitleId varchar(50) FgValidate varchar(1) FgActive varchar(1) CreatedBy varchar(50) CreatedDate datetime ModifiedBy varchar(50) ModifiedDate datetime ValidatedBy varchar(50) ValidatedDate datetime	Structure of clientprogresshd table

Figure 5. Database Structure

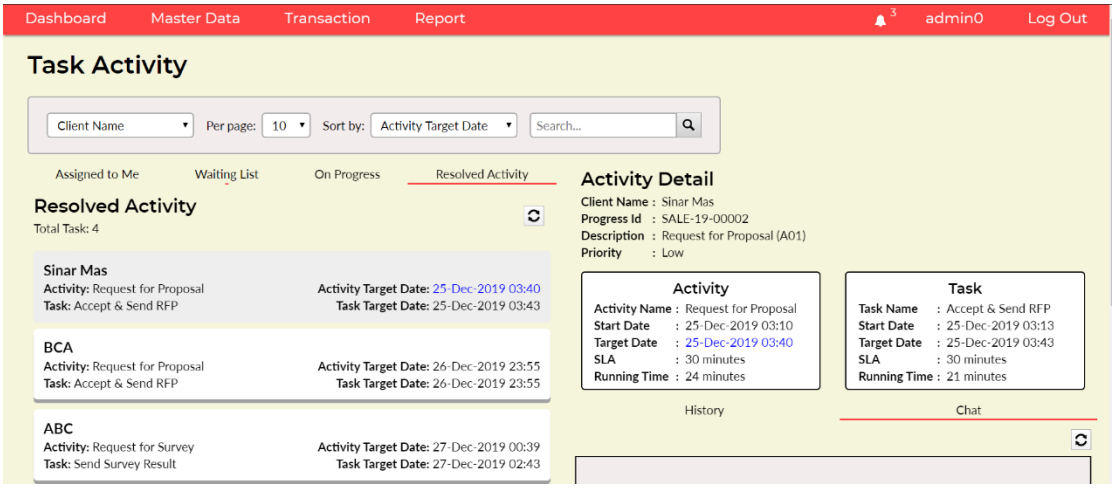

Figure 6. User Interface

One of the user interfaces is shown in **Figure 6**. The user interfaces are built based on users' provided mock-ups during the listening to users' stories phase.

Testing

The software tested using the unit tests developed on the coding phase and User Acceptance Test (UAT). Each class is tested with its unit test. Each unit test tests all methods of the class it is testing. The result of the unit tests can be seen in **Table 5**.

Table 5. Result of Unit Test

Document Name	Unit Testing	
Application Name	Business Activity Monitoring	
Activity	Unit Testing	
During	10/06/2019 - 31/07/2019	
No	**Class**	**Pass**
	Controller	
1	Account	✓
2	Dashboard	✓
3	Notification	✓
4	User	✓
5	Job Level	✓
6	Division	✓
7	Role	✓
8	Company	✓
9	Client	✓
10	Client Title	✓
11	City	✓
12	Activity	✓
13	Task	✓
14	Activity Task	✓
15	Activity Doc	✓
16	Priority	✓
17	Email Template	✓
18	Activity Transaction	✓
19	Activity Task Transaction	✓
20	Activity Document Transaction	✓
21	Task Activity	✓
22	Assigned To Me	✓
23	Waiting List	✓
24	On Progress	✓
25	Resolved Activity	✓
26	Activity History	✓
27	Activity Chat	✓
28	Activity Report	✓
	Model	
1	Account Model	✓
2	Dashboard Model	✓
3	Notification Model	✓
4	User Model	✓
5	Job Level Model	✓
6	Division Model	✓
7	Role Model	✓
8	Company Model	✓
9	Client Model	✓
10	Client Title Model	✓
11	City Model	✓
12	Activity Model	✓
13	Task Model	✓
14	Activity Task Model	✓
15	Activity Doc Model	✓
16	Priority Model	✓
17	Email Template Model	✓
18	Activity Transaction Model	✓
19	Activity Task Transaction Model	✓
20	Activity Document Transaction Model	✓
21	Assigned To Me Model	✓
22	Waiting List Model	✓
23	On Progress Model	✓
24	Resolved Activity Model	✓
25	Activity History Model	✓
26	Activity Chat Model	✓
27	Activity Report Model	✓

UAT is done by the end-user of the software, guided by the researchers. The UAT result for some of the task assignment features shown in **Table 6**.

Table 6. Result of UAT

Test Case	Precondition	Test Step	Expected Result	Actual Result	Pass
Pass a task	- User already in assigned to me tab - There are at least a card in assigned to me tab - User has a subordinate or there are another staff in the same division and has the same job level as the current user - Card detail already opened	1. Click Assign button 2. Choose a staff that will receive the task 3. Put in additional note 4. Choose On Progress as status 5. Click Send button	Task passed to the chosen staff and moved from assigned to me to on progress tab	A "Success Update." message appear and the task card moved from assigned to me to on progress tab	Yes
Mark a task as completed	- User already in assigned to me tab - There are at least a card in assigned to me tab - Card detail already opened - All mandatory documents already attached	1. Click Assign button 2. Choose the staff that completed the task 3. Put in additional note 4. Choose Done as status 5. Click Send button	Task passed to the next division and moved from assigned to me to on progress or resolved tab	A "Success Update." message appear and the task card moved from assigned to me to on progress tab	Yes
Reject a task and return it to previous division	- User already in assigned to me tab - There are at least a card in assigned to me tab - Card detail already opened	1. Click Assign button 2. Choose the staff that reject the task 3. Put in additional note 4. Choose Reject as status 5. Click Send button	Task returned to the previous division and removed from assigned to me tab	A "Success Update." message appear and the task card removed from assigned to me tab	Yes
Mark a task as completed without attaching mandatory documents	- User already in assigned to me tab - There are at least a card in assigned to me tab - Card detail already opened - At least one mandatory document is not attached.	1. Click Assign button 2. Choose the staff that completed the task 3. Put in additional note 4. Choose Done as status 5. Click Send button	Task stay in assigned to me and an error message appear	A "Failed to update, please attach mandatory document." message appear	Yes

Software Increment

The completed iteration released to the user and implemented. The main requirements of each iteration listed in **Figure 3** are as follow:

Master data

Develop login and data master pages for staff, client, activity, task, and any related data e.g., division and job level.

Activity transaction

Allow activity to be started, send emails to clients about activities status, and to add clients and activities validation. The validation is done outside the application and only recorded to the application by admins.

Task activity

Show list of currently assigned tasks, incoming tasks, tasks in progress, and completed tasks. Add chat and activity history functionality, as well as task list exportation. Users also should be able to pass tasks that are assigned to them; to a subordinate or someone in the same division and has the same job level as them, to the next division, and the previous division. Lastly, the user must receive a notification about their activities' status.

Activity report and dashboard

Generate an activities report containing user filtered activities, and add a dashboard showing a summary of activities and SLA overdue in the last 30 days grouped by users' division. The dashboard should be exportable.

User access control

Change login to be based on email rather than a username, and allow admin to add and configure role for each user.

Some more minor requirements are not listed and some of the master pages are created in iteration 2 to 5.

CONCLUSION

The BAM software can help the company to track its business activity, so they can report it to their customer, and to provide a platform to communicate about business activities between divisions. This way, the company will be able to complete their business processes faster and controlled, reducing delays and give managers the ability to take the right action, leading to an increase in competitiveness. Without this software, the company will continue to serve a slow service and reducing its customer count.

In its current state, the software does not provide a way to pass a task to subordinates other than by assigning it to them directly. This may cause a task management problem and can limit staff who have the initiative to take a task. In the future, this may be solved by allowing staff to take a task on their own accord.

XP methodology helped the researchers' small team by splitting the development process into many iterations. This helped researchers to finish the software earlier than the planned development time while keeping the quality of the software; since the software is also tested throughout the coding phase, greatly reducing the number of errors.

Suggestion for future researches on BAM is to include key performance indicators matrix to improve performance measurement based on already existing SLA or decision support system to help management remedy the immediate problem.

REFERENCES

Adewale, A. A. (2016). Change, Customer Satisfaction and Competition: Issues from the Strategic Management Context. *International Journal of Economics, Business and Management Studies, 3*(2), 55-66.

Andry, J. F., Juliawan G., Christian Y., Leonardo J., & Nicholas. (2018). Parking System Development Using Extreme Programming Method. *Journal of Digital Information Management, 16*(6), 279-288. https://doi.org/10.6025/jdim/2018/16/6/279-288

Apăvăloaie, E. I. (2014). The Impact of the Internet on the Business Environment. *Procedia Economics and Finance, 15*, 951-958. https://doi.org/10.1016/S2212-5671(14)00654-6

Nafie, F. M., & Eltahir, M. A. (2016). Real-Time Monitoring and Analyzing Business Process. *Research Inventy: International Journal of Engineering and Science, 6*(7), 31-35.

Ouassarah, A. A., Averseng, N., Fournet X., Petit, J. M., Revol, R., & Scuturici, V. M. (2015). Understanding Business Trends from Data Evolution with Tornado. *International Conference on Data Engineering, Apr 2015, Seoul, South Korea*. pp.1456-1459. https://doi.org/10.1109/ICDE.2015.7113400

Özcan-Top, O., & McCaffery, F. (2018). Conformance to Medical Device Software Development Requirements with XP and Scrum Implementation. International Conference on Software Engineering Research and Practice, July-Aug 2018, Nevada, United States of America. pp. 99-105.

Suryantara, I. G. N., & Andry, J. F. (2018). Development of Medical Record with Extreme Programming SDLC. *International Journal of New Media Technology*, *5*(1), 47-53. https://doi.org/10.31937/ijnmt.v5i1.706

Suryantara, I. G. N., Bernanda, D. Y., & Andry, J. F. (2018). *Pengembangan Aplikasi Akuntansi Dengan Kerangka Kerja Extreme Programming.* s.l., SEMNAS RISTEK 2018, pp. 117-122.

Tabassum, A., Bhatti, S. N., Asghar, A. R., Manzoor, I., & Alam, I. (2017). Optimized Quality Model for Agile Development: Extreme Programming (XP) as a Case Scenario. *International Journal of Advanced Computer Science and Applications, 8*(4), 392-400. https://doi.org/10.14569/IJACSA.2017.080453

Taibi, D., Lenarduzzi, V., Ahmad, M. O., & Liukkunen K. (2017). Comparing Communication Effort within the Scrum, Scrum with Kanban, XP, and Banana Development Processes. *Evaluation and Assessment in Software Engineering, June 2017*, Karlskrona, Sweden. https://doi.org/10.1145/3084226.3084270

The World Bank. (1997-2017). *Individuals using the Internet (% of population).* Retrieved on 28 August 2019 from www.data.worldbank.org

Tonin, G. S., Goldman, A., Seaman, C., & Pina, D. (2017). Effects of Technical Debt Awareness: A Classroom Study. *Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, May 2017*, Cologne, Germany. pp. 84-100. https://doi.org/10.1007/978-3-319-57633-6_6

Yeboah, J., & Ewur, G. D. (2014). Quality Customer Service as a Competitive Advantage in the Telecommunication Industry in the Western Region of Ghana. *Journal of Education and Practice, 5*(5), 20-30.