1. Introduction

The term **post-mortem** is latin for "after death", and originally referred to a medical examination of a corpse to determine the cause of death. The term has, more colloquially come to refer to any "after the fact" analysis and discussion of a recently completed process or event, to see what lessons we can learn from it.

Such analyses are have been going on for a very long time. Five thousand years ago Egyptian doctors recorded wounds, treatments and results to build up a body of knowledge about what did and did not work. During the middle ages, engineers from all over Europe would flock to the site of a collapsed cathedral to attempt to learn lessons that would prevent similar fates from befalling their own cathedrals. Military strategists have long studied every battle ever recorded so that they could learn lessons without having to suffer defeats.

The above examples all seem focused on understanding *what we did wrong* ... and historically (and perhaps psychologically), failure has proven to be one of our best teachers. But we also want to learn from our successes. Every task we do can be an experiment where we try new things. Some experiments will not turn out the way we had hoped, but some will turn out better than we could have imagined. In organizations that are serious about getting better at their jobs, retrospective reviews are an important part of every project.

2. Goals of a Post Mortem

The goals of a post mortem review are very simple:

- To identify the things we did right, so that we can remember to try them again in similar situations.
- To note the things that should have been done differently, so that we can refine our techniques in the future.
- To note the things that we did wrong, and to suggest alternative approaches or safety measures that we should employ the next time we face a similar problem.

Exploring what we did wrong is frightening ... and in some organizations it is dangerous. If admitting having made mistakes opens us to criticism or discipline, we are unlikely to make such admissions. This strategy is ultimately self-defeating, since failing to understand a past mistake usually condemns us to repeating it again in the future. Organizations that are serious about improvement understand this, and take trouble to create a process and culture wherein it is safe to explore mistakes.

When we enter into a post-mortem process, we must all accept a few basic premises:

- Everybody tries to do their best, as best they understand it.
- We make our decisions in stressed situations, with imperfect information.
- We are often called upon to carry out tasks for which we have not been trained, with what-ever tools and resources happened to be at hand.
- Mistakes are inevitable in such situations.
- The goal of this process is not to find fault with any individual or their actions. Rather it is to look at what happened and see what lessons we can learn from it.
- The output of this process will not be an assessment of any person or group of people, but rather an assessment of our processes, and how they can be improved.

It is absolutely essential that everyone involved completely accept this "no blame, we are here to learn model". Many organizations go to great trouble to create such safe environments. The FAA, for instance, has an Aviation Safety Reporting System, whereby pilots who make "mistakes" can gain immunity from regulatory discipline if they report those incidents to the ASRS.

3. Process of a Post Mortem

In highly stressed environments, built up fear and anger may make it very difficult to get people to discuss (or even to objectively look at) a recent project. In situations like this, the organization may be in serious need of healing, and it may be necessary to have the process managed by a professional facilitator. In healthier settings, it may be "just another meeting" where the goal is simply to gather opinions and formulate recommendations. The types of questions that should ultimately be explored are:

1. What things turned out well, and why?
2. What tools, techniques, and processes worked well, and why?
3. What improvements would we make to them?
4. What things turned out poorly, and why?
5. What tools, techniques, and processes worked poorly, and why?
6. What improvements would we make to them?
7. What alternatives should we develop?
8. What unexpected problems came up?
9. What warning signs should we have noticed?
10. What decisions could we have reasonably made differently?
11. What still confuses or concerns us?
12. How would we assess the success of this project?
13. How would we assess the success of our tools, techniques, and processes?
14. What are the few key recommendations we would make to management?

Even in non-threatening environments, gathering this sort of input can be a delicate process. It is important that everyone be given the opportunity to speak, and that the meeting not be monopolized by the people with the strongest opinions and loudest voices. As in brain-storming, the early phases should be non-critical:

Let everyone contribute what-ever they think interesting.

You can talk about your own experiences, but you are not allowed to comment on somebody else's experience.

After we have given everybody the opportunity to express all of their thoughts, we can then enter a second phase where we compare, combine, sift, sort, and synthesize punch-lines out of the raw material developed in the first phase.

People are often hesitant to complain about problems or concede errors when their managers are present. Thus, it is often a good idea to conduct separate post-mortem for people at different levels in the management hierarchy. Higher level managers need the output of the post-mortem process to make sure that the needed lessons are learned. It may actually be counter productive for the managers to be present as people are exploring the events of the project.

3.1 Post Mortem Report

The process of conducting a post-mortem is, in many respects, a brain-storming process. Many ideas and issues will be explored, and most of them will not prove worthy of capturing. Explorations of how things went wrong may be heated and/or cathartic ... but publishing the details of those stories would be antithetical to the goals of the process.

A Post Mortem report is not a compendium of stories that came out in post mortem meetings. All of those stories need to be examined, reconciled, filtered, distilled, prioritized, and synthesized into cogent punch-lines. They then need to be organized into a few key points, with examples of problems that occurred and recommendations for how to deal with them in the future.

A good post-mortem report clearly describes problems, and opportunities for improvement, and it does so without blaming people or rehashing heart-breaking stories. In many organizations, it is common for post-mortem reports to be widely distributed, so that many people have the opportunity to learn from every project.

4. Summary

There is no one best format or technique for post-mortem reviews, and there is no authoritative list of questions to be explored. Every organization has to evolve a post-mortem style that fits in with their culture and mission. Some may involve a weekend of drum circles and trust exercises at a forest retreat, while others are just 30 minutes of prepared power-point presentations in the main conference room.

A good post-mortem meeting is any meeting where people honestly share and explore their experiences with the intent of learning from them and doing better next time. Everything else is just a matter of style.

Post-Mortem Report TOC

1. **Project**
   A. Description
      i. Project Name:
      ii. Client:
      iii. Project Manager:
      iv. Solutions Architect:
      v. Start Date:
      vi. Completion Date:
   B. Project Overview [Describe the project in detail.
      i. Discuss the project charter
      ii. What was the project success criterion?
      iii. etc.
2. **Performance**
   A. Key Accomplishments *[List and describe key project accomplishments. Explain elements that worked well and why. Consider listing them in order of importance. Be specific.]*
      i. What went right?
      ii. What worked well?
      iii. What was found to be particularly useful?
      iv. Project highlights
   B. Key Problem Areas *[List problem areas experienced throughout the project. Be specific.]*
      i. What went wrong?
      ii. What project processes didn't work well?
      iii. What specific processes caused problems?
      iv. What were the effects of key problems areas (i.e. on budget, schedule, etc.)?
      v. Technical challenges
   C. Risk Management *[List project risks that have been mitigated and those that are still outstanding and need to be managed.]*
      i. Project risks that have been mitigated:
      ii. Outstanding project risks that need to be managed:
   D. Overall Project Assessment *[Score/rank the overall project assessment according to the measures provided. A 10 indicates excellent, whereas a 1 indicates very poor.]*
   E. Additional Comments:
      i. Other general comments about the project, project progress, etc.
3. **Key Lessons Learned**
   A. Lessons Learned *[Summarize and describe the key lessons and takeaways from the project. Be sure to include new processes or best practices that may have been developed as a result of this project and to discuss areas that could have been improved, as well as how (i.e. describe the problem and suggested solution for improvement).]*
   B. Post Project Tasks/Future Considerations *[List and describe, in detail, all future considerations and work that needs to be done with respect to the project.]*
      i. Ongoing development and maintenance considerations
      ii. What actions have yet to be completed and who is responsible for them?

iii. Is there anything still outstanding or that will take time to realize? (i.e. in some instances the full project deliverables will not be realized immediately)

Project

Description:

Project Name: SEPT Appointment Booking System
Client: Tutor - Mohammad Haqqani
Project Manager: Kyle Jackson (s3600341) and Tommy Chua (s3503207)
Solutions Architect: Kyle Jackson (s3600341) and Tommy Chua (s3503207)
Start Date: 7th of March
Completion Date: 21st of May

Project Overview:

This project was given to us from the product owner Mr Mohammad Haqqani on 27 February 2017. The duration given for this project is 4 months. Mr Haqqani wants a "Appointment Booking System". This is basically a booking system that can be used by any business, allowing a customer to book a time slot for an appointment. The goals of this project are

1. Customer booking will be displayed once they make a booking.
2. Customer information(Name ,Address, phone number and etc) are store in the system
3. Business owner can activity to a business
4. Business owner can add employee to the system and assign employee to particular activity.
5. Business owner can create new business.
6. Business owner can make a booking for customer.
7. Business owner able to view a summaries of customer booking.

This project uses agile method logic. Each task of for the project has been properly broken down using a work break down sheen and Gantt Chart.

The criteria for the project to be success are all the functionality that have been given by the product owner should be implemented, and has been progressively completed through cooperative teamwork through use of agile methods. The project should only take 4 months or less in order to be success. Upon delivering the product, product owner must be satisfy with the delivery.

Performance

Key Accomplishments:

Key accomplishments of the project would be the completion of each checkpoint for functionality on time and achieving all required functionality for 2 of the 3 checkpoints of the project.

Communication via Slack was very effective as it allowed for easy communication between team members and for an easy method of contact to discuss and delegate tasks between members of the group.

Use of Trello allowed to make it easier to organize and gather the requirements for each checkpoint of the project.

Project development through the use of GitHub allowed for an easy way to maintain code, version control and ensure that all team members developing were working on the latest version and continuously building up the project.

Key Problem Areas:

A key problems encountered would be wasted time spent on recoding functionality which occurred due to not enough communication regarding the system itself, more discussion on the system development would of allowed for less re-coding of the whole program therefore taking up less time and could be easily avoided through more thorough group communication regarding code development.

Another key problem encountered would be not thorough enough clarification, communication or understanding of some requirements specific details leading to time spent on having to make minor changes to cater for specific details of requirements. Potentially having an effect on the schedule for the next checkpoint.

Technical challenges included converting the project to maven and getting the executable to run the program from console. Also, learning and inclusion of logging framework called log4j. Another challenge would be learning and adapting the initial part A checkpoint to develop and include a Graphical User Interface through the use of the java library Swing.

Risk Management:

Project risks mitigated through the use of version control platform GitHub, preventing time wasted on managing version of the project and minimizing conflict occurring during development allowing for more productive coding to be done.

Overall Project Assessment:

Overall project assessment score of an 8.

Performance against project goals/objectives	*1 2 3 4 5 6 7 8* **9** *10*
Performance against planned schedule	*1 2 3 4 5 6 7* **8** *9 10*
Performance against quality goals	*1 2 3 4 5 6 7* **8** *9 10*
Adherence to scope	*1 2 3 4 5 6 7 8* **9** *10*
Project planning	*1 2 3 4 5 6 7* **8** *9 10*
Resource management	*1 2 3 4 5 6 7* **8** *9 10*
Project management	*1 2 3 4 5 6 7* **8** *9 10*
Development	*1 2 3 4 5 6 7 8* **9** *10*
Communication	*1 2 3 4 5 6* **7** *8 9 10*
Team cooperation	*1 2 3 4 5 6 7* **8** *9 10*
Project deliverable(s)	*1 2 3 4 5 6 7 8* **9** *10*

Additional Comments:

Overall very effective communication and organization between team members allowing for great productivity and progress to be made very quickly and smoothly when working together on the project simultaneously.

Key Lessons Learned:

Lessons Learned:

Communication is a huge key to success, followed by organization and finally completion of tasks on time. Communication practically flowing on and needed for effective organization and task completion.

Some best practices to take away from this project would be the use of Slack, and continuous improvement of the use of this communication of this tool as it is possible to use much more functionality of this tool to further improve communication between team members. An example of extended functionality to potentially use would be the use of more channels, creating a system of using reactions as well as creating specific threads when certain circumstances are encountered e.g. bug fixing

Use of Trello is very effective and useful for organizing project goals and showing progress of completion these goals.

GitHub also is a greatly effective and useful tool to use for version control more potential improvement to increase the effective use of this tool. Also was very easy to start to use due to it being very compatible with Eclipse which was the IDE used for development by team members making it easier to commits, push, pull and create branches.

Another great practice to use would be pair programming which allowed for the best communication whilst development was occurring, and would further minimize time wasted on recoding functionality which may have been misunderstood by team members therefore pair programming allows for more effective and efficient programming.

Post Project Tasks/Future Considerations:

Potential room for improvement of the system by refactoring code and arranging code more effectively to allow for a more easily maintainable and adaptable system for future changes, this can be done by improving arrangement of code by creating more methods to hold repeated code and functionality.