Lesson 1 - Peer code review is a staple of the software industry. It is one way that practicing professionals continually increase their technical skills.[1]

Lesson 2 - The earlier a defect is found, the better. The longer a defect remains in an artifact, the more embedded it will become and the more it will cost to fix.[2]

Lesson 3 - Reviews should be conducted by invested experts and co developers.[6]

Lesson 4 - Let expert developers self-select changes they’re interested in and competent to review.[6]
Peer code review related lessons

Lesson 5 - Authors should annotate source code before the review begins.[3]

Lesson 6 - Use of checklists substantially improve results for both authors and reviewers.[3]

Lesson 7 - Reviewer should always aim for an inspection rate of less than 300-500 LOC/hour.[3]

Lesson 8 - Managers must foster a good code review culture in which finding defects is viewed positively.[3]

Lesson 9 - Lightweight-style code reviews are efficient, practical, and effective at finding bugs.[3]
Peer code review related lessons

Lesson 10 - Verify that defects are actually fixed after review.[3]

Lesson 11 - Reviewers may not be willing to do their best to review the written code by their peer developers, especially when they are assigned poor program writers.[4]
Question 1: Your team has been working on software development for quite a some time now. Few of the developers have even started submitting the complete code modules after self testing and forwarded it to you to decide on the further review process. Being the Project Manager, it is you who has to decide if these software modules need to be reviewed or sent directly for testing. What would your decision be?

Justification: In a software development process, once developers are finished writing their codes, they are first reviewed and then tested. The reason for this is that the yield of the Code Review phase is 50 to 80% better than that of the Test phase. [3] Which means that the defects which are uncovered and fixed via review phase would usually take hours of testing and fixing otherwise. [Refer to fig 1.]
Option 1 - Review must be started right away!
Justification - The earlier a defect is found, the better [Lesson 2].
Overall cost reduces, time is saved and quality improved.

Option 2 - Perform review only after development of all modules.
Justification - The longer a defect remains in an artifact, the more embedded it will become and the more it will cost to fix. [Lesson 2]

Option 3 - I would wait for some more modules to come before initiating review process.
Justification – Find bugs early and frequently[6]. This option is better than option 2 (and others) but not than option 1, in terms of cost, time and quality.

Option 4 - Review is very expensive, I need more time to think about it.
Justification – The sooner a review is done, the better it is. Delivering bugs to QA/testing costs money but delivering bugs to customers costs a lot of money and loss of goodwill[5].

Option 5 - Review is not required, testing will uncover the problems anyway.
Justification – The amount of defects found in testing are very less as compared to the ones located by review and also the time and cost of testing and fixing those bugs is higher too. A single testing stage is unlikely to remove more than 35 percent of the defects in the tested work product, whereas design and code inspections typically find 50 to 70 percent of the defects. [8]
QUESTION 2: Among the 8 team members, you have to select a person to carry out the review process. Who will you choose?

Justification: Most important predictor of the number of defects detected during review is reviewer expertise; the process has minimal impact[9]. Thus the more the experience and expertise, better it is.

Option 1- New developers with little or no experience
Option 2- Co-Developers with industry experience of 2-3 yrs
Option 3- Co-Developers with industry experience of 4-6 yrs
Option 4- Co-developer with industry experience of 8 yrs
Justification – Reviews should be conducted by invested experts and co developers . [Lesson 3]

Option 5- Hire a new developer just for carrying out reviews.
Justification – Review is carried out by a co developer. There is no designation as full time reviewer, it is the co developer with more experience and expertise who performs review.
QUESTION 3: What type of review process would you like to use in this project?

The review process can be broadly categorized into two types-
Heavy weight reviews like formal, over the shoulder, email pass around and pair programming.
Light weight review like tool assisted review.

Option1- Heavy weight reviews
Justification- Formal, or heavyweight, inspections have been around for 30 years – and they are no longer the most efficient way to review the code[3]. They involve more people and are time consuming.

Option 2- Tool Assisted Review
Justification- The results showed that lightweight code reviews take 1/5th the time (or less!) of formal reviews and they find just as many bugs! [3] Tools can help structure reviews and integrate them with other development Systems too.[6]
QUESTION 4: Being the project Manager, you need to set a policy to be followed while carrying out the review process. The policy to review LOCs should maximize the quality of software code and at the same time increase the productivity of reviewer involved. Which of the following policies would you implement? (irrespective of code's complexity)

Justification:

![Defect Density vs. Inspection Rate](image)

Figure 2: Inspection effectiveness falls off when greater than 500[3].
Option 1 - Review 100-200 LOC/hour.

Option 2 - Review less than 300-500 LOC/hour and continue with development if time allows.

Option 3 - Review at least 600-800 LOC/hour and continue with development if time allows.

Option 4 - Review 1000 or more LOC/hour and complete the whole stack of code quickly.

Justification – Figure 2 shows the answer: reviewing faster than 400-500 LOC/hour results in a severe drop-off in effectiveness. And at rates above 1000 LOC/hour, you can probably conclude that the reviewer isn’t actually looking at the code at all.

Option 5 - Don't worry about LOC and time. Proceed as you wish!

Justification – Reviewing optimal lines of code will only make review effective or else it is just wastage if time leading to nothing.
**Situation- 5:** Reviewer comes to you with a complaint of overwork. He is finding it hard to review the code modules of each and every team member. What will you do in such a situation?

**Option 1:** Implement a new policy where in only fresher's will submit their code for review and senior members will review the code on their own.

**Justification:** Let expert developers self-select changes they’re interested in and competent to review[6]. It is the new developers whose code need to be reviewed thoroughly.[Lesson 4 ]

**Option 2:** Select one more team member to perform code review.
**Option 3:** Recruit another reviewer.
**Justification** – These are not viable options considering the amount of cost and time involved.

**Option 4:** Ask him to review as much as he can and leave the rest.
**Justification** – Reviewing code is an important practice in industry and hence cannot be taken lightly.[Lesson 1]
QUESTION 6: It comes to your notice that reviewer has been threatening the fresher's that they will be given poor feedback in performance evaluation if they don't improve their coding skills. As a manager what steps would you take to foster a good code review culture in your team? (multiple options may be correct)

Justification:

The reviewers aren’t there to show that they’re smarter than the author, and the author isn’t there to justify every bit of the work product and rationalize away problems. Reviewers should thoughtfully select the words they use to raise an issue, making their observations about the product and not about the author. An author who walks out of a review meeting feeling embarrassed, personally attacked, or professionally insulted will not voluntarily submit work for review again. [7]
Option 1 - Exclude the review feedback from performance evaluation and promote the idea that finding defects is good as long as you learn from it.

Justification - To maintain a consistent message that finding bugs is good, management must promise that defect densities will never be used in performance reports [Lesson 8]

Option 2 - Clarify it to developers that reviews are meant to help them correct bad habits, learn new tricks and expand their capabilities.

Justification - Review is an opportunity to improve the code, and the goal of the bug review process is to make the code as high quality as possible.

Option 3 - Warn reviewer not to repeat such behavior in future and focus only on code review and quality.

Justification – Managers must support the idea that teams should maintain the attitude that finding defects means the author and reviewer have successfully worked as a team to jointly improve the product.

Option 4 - Include the review feedback in performance evaluation and promote the idea that finding defects is bad.

Justification – Using defect data from reviews to evaluate the performance individuals is a classic culture killer [7]

Option 5 - Warn the developers to improvise their coding skills or else there can be serious repercussions.

Justification – This misguided strategy could lead developers to not submit their work for review.
QUESTION 7: Since the fresher's are new to projects and software development, you decide to give them certain instructions or guidelines to carry out the code development and review process smoothly to save reviewer some time. Select the guidelines you would like to give to developers. (Multiple options may be correct)

Justification:
A strong argument can be made that overall software quality is affected far more profoundly by improvements to developer skills, which reduces future defect creation, than by simply removing defects from current individual documents\textsuperscript{10}.

Option1- Make a habit of adding annotations to the code.
Justification- Annotations guide the reviewer through the changes, indicating which files to look at first and defending the reason and methods behind each code modification.
Option 2- Maintain a personal checklist of mistakes that you often make, things you normally forget or those mentioned in the feedback you get after your code review. In fact reviewer should maintain a checklist too.

Justification - The “checklist review” gives the reviewers a specific list of things to check for at the class, method, and class hierarchy levels. The checklist method is the most successful one, uncovering more defects in less time than the other techniques, 30% better than the worst in the rate at which defects were uncovered.[3]

Option 3- To thoroughly scan through the code in the end and look for defects then. Maintaining a checklist is not mandatory and depends entirely on developer's choice.

Justification – Self testing before sending code for review and maintaining a checklist are a good practices for carrying out reviews smoothly[5].

Option 4- Reviewer is there to look out for the bugs in their code, they should only focus on the development part and leave the bug location and fixing entirely to reviewer.

Justification – Review is a team process, both reviewer and developers are responsible for it. Everything cannot be left at the reviewer as he himself is a co developer and need to carry out other tasks too.

Option 5- To not submit their code for review if they are 100% sure that it is bug free. Instead add it to the final software repository and commit for final product release

Justification- A novice developer cannot make this decision themselves as they are not aware with the industry trends. Doing this will have serious repercussions on the project in terms of quality, time and cost.
**QUESTION 8:** You get a call from your project’s sponsor. You're 2 months away from the project deadline and she wants the deadline bumped up by 3 weeks. She says it is due to an unforeseen market change and it is critical - so much so that the project will be worthless if it is not received by the new deadline. What would you do? (multiple choices may be correct)

**Option 1** - Ask for budget raise to give incentives to the existing team members  
**Justification** - Making best use of current employees and offering them incentives for their overwork is the best viable option as they are already acquainted with the work flow since beginning and therefore no extra time or resources need to be wasted.
Option 2- Hire new employees to achieve deadline.  
**Justification** - Hiring new employee will delay the work a bit, as they will need few days to adjust to the environment and learn work practices followed.

Option 3- Ask your entire team to focus only on project delivery and seek code reviews or bug fixing as something to be handled later on during maintenance.  
**Justification** – Achieving high levels of software quality will shorten schedules and lower costs at the same time, therefore leaving review process for later on does not help.[5]

Option 4- Scrap off the entire project and inform the sponsor that it's impossible to achieve new deadline.  
**Justification** – Just ending the project without even trying to work out all possibilities is not a good option as a lot of resources and time have already been invested in the project. Also it is not good for company’s credibility in market.[3]

Option 5- Stick to the original schedule and ignore the project sponsor's demand.  
**Justification** – Sticking to original schedule with pre poned deadline is poor management practice.
QUESTION 9: It has come to your notice that many a times even after finding bugs during review, the developers forget to fix them. It's hard for reviewer to keep track of these unsolved defects and these leftover issues keep piling up affecting code quality. So, what’s a good way to ensure that defects are fixed before the code is given the 'All Clear' sign?

Justification:
If you’re going to all of the trouble of reviewing code to find bugs, it certainly makes sense to fix them. Yet many teams who review code don’t have a good way of tracking defects found during review or ensuring that bugs are actually fixed before the review is complete. When the modified version of the code becomes available, peer reviewers is performs the task of verifying that all changes were correctly made (and no others). The code is also re-checked, by rerunning all analyzers, to make sure no new issues appeared. [Lesson 10]

Option 1- Ask reviewer to log the bugs and discuss them with the developer along with the deadline till which they will be fixed and submitted for review again.

Justification- Fixing mutually the deadline for fixing codes is the most viable option as it has consent of both reviewer and developer. Also discussing defects with developer is a high quality, efficient, and effective means for learning.[10]
Option 2- Ask reviewer to himself fix the deadline for the developer to resolve the issues uncovered during code review and send updated code for review again.  
**Justification**- Developers have other tasks and deadlines to handle too and therefore fixing deadline without seeking their opinion is a bad idea.

Option 3- Developers should be informed about the defects and given the freedom to resolve them as and when they can.  
**Justification** – Giving developers the freedom to fix the defects on their own whenever they want is more worse option as keeping defects for long can be problematic too.

Option 4- The reviewer should himself fix the bugs if it has been unresolved for a long time.  
**Justification** – The task of reviewer is just to find defects and not fix them. Raise issues, don’t resolve them.[10]

Option 5- Fixing codes is not as important as locating them, submit the code with defects in case they are un resolved  
**Justification** – Finding defects without intent of fixing them is a complete waste of time, money and resources.[3]
**QUESTION 10:** Your project is 1 month away from final release, work is at its peak and one of the developers decided to quit. You cannot afford to have staff shortage and at the same time hiring a new developer would have its consequences on the project. What would be your decision?

**Option1** - Divide the pending work amongst the remaining developers and offer them incentive for doing the same.

**Justification** - Considering the remaining time and work, dividing work amongst the current team members who have been involved with the project since starting is the most viable option.
Option 2- Ask the reviewer to stop reviewing codes and devote all of that time to code ex-developer's module.

**Justification**- Stopping reviews at end stage will only subsidize your code quality which you have been maintaining till now.

Option 3- Focus on current developments going on & leave his part for later.

**Justification** – Since the entire project will not be complete without each and every module specified, this option will definitely put all the efforts on project till now at stake.

Option 4- Hire a new developer anyways.

**Justification**- Any new developer will take few days to get acquainted with the project and considering the time remaining, it does not seem likely that this option will be of much help.

Option 5- Instead of hiring a new developer, you can opt for an intern which would save you money.

**Justification** – Hiring an intern is even worse than experienced developer as they have absolutely no prior development experience what so ever and will therefore be of absolutely no help in such situation.
References

[1] Deborah A. Trytten
   "A Design for Team Peer Code Review“, DA Trytten - ACM SIGCSE Bulletin’05

[2] Peter C. Rigby, Daniel M. German & Margaret-Anne Storey


[4] WANG Yan-qing, LI Yi-jun, Michael Collins and LIU Pei-jie

   ” Design and Code Inspections to Reduce Errors in Program Development.”


