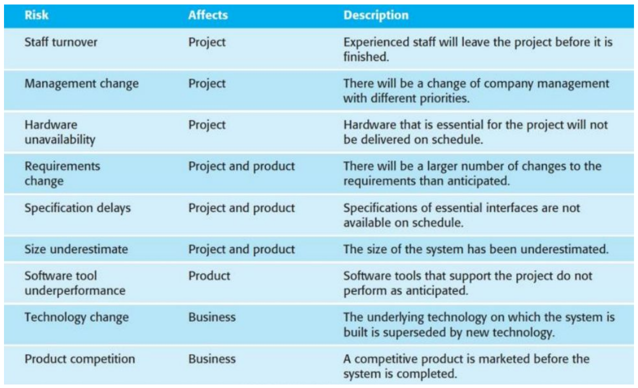
CS 362 Homework 2, Lyell Read

Q.1: In addition to the risks shown in the figure below, identify at least six other possible risks that could arise in software projects.



Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of company management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
Software tool underperformance	Product	Software tools that support the project do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

Budgeting Errors
At the beginning of the project, the financial team and the project leader sat down and formulated the budget for the project. Accidentally (or possibly purposefully, if malicious intent exists), some changes were not saved to the spreadsheet (this is a large project, so this could go unnoticed), or both finance and PL forgot to budget something large (i.e. research and development space rent, compute resources, test machines, visualization software), and this forces the team to spend money budgeted for other areas on the forgotten areas, spreading the funding extremely thin. This poses a risk to the team as they could end up with too little money to complete essential aspects of the project, or maybe quality would suffer. Further, some members could leave as a result of the lacking compensation that is likely to come out of this.

Funding Cutbacks
As a result of hard times, the parent company can no longer fund the complete sum required to keep the project running. In this case, the project would suffer scarcity of funds and the associated issues, as listed above.

Customer or PM Change / Contract Renegotiation
Customers or PM's can change, and they sometimes do in the middle of projects. In this case, the new customer might have very different ideas of what they want accomplished in the project, and this can severely impact the project's progress: if the team has been working on a feature for months and now the new director does not want that feature, the work, time, effort, money will be in vain. Depending on the grace with which the new manager takes over, there could be some resentment in the team with respect to the new manager, especially if a lot of the work done previously is now void of utility.

Illness

Very obviously, if anyone on the team gets ill (short term or long term), this will put extra strain on the other members of that team. If this person is in a capacity that is essential for work, they might need to be replaced immediately which can cause compatibility issues, as well as financial strain. If they are crucial and irreplaceable to the team, work can completely stop. None of these are good for the project.

Cyberattacks / Hacking
Especially if the software being developed is high tech, government work, luritive, bank related, or otherwise desirable, hackers may want to get their hands on it. These bad actors can range from an individual with ulterior motives to a state actor seeking to enact nation-state-calibre cyberwarfare against your government. Either way, this poses a great risk financially, as well as a reputation risk: if your company / project is hacked and sensitive data is exfiltrated, you will likely not get another contract with whoever gave you that data. You could also have the current contract suspended, depending on how the hacker got in.

Requirement Change
Changing times, changing leaders and other factors can make the requirements on a project change. Take the example where you are running a protein simulation program development project in a high tech biotech firm. As the COVID pandemic appears, you (or your manager) might opt to redirect efforts to fight COVID rather than what you were previously fighting. This is great in the short term, but can set back your overall goals in the long term, and cause mismatch between what you promised to perform and what you performed.

Data Loss
Typically in software, a “branch and merge” development process can be used. If this is the case, and one single person is working on a branch to add a feature, they might not push to the online repo, and leave it on their computer until they are done. If this is the case, and they lose / break their computer, they could lose all that work. Similarly, if the online repo is hosted onsite without backups(or if backups are onsite) (nowadays, unless the project is sensitive in terms of data, repos are cloud-based, so data loss is not an issue), an unplanned event (fire, earthquake, ...) could result in the loss of all work - an immense setback for the project and team.

Q.2: Write a case study to illustrate the importance of communications in a project team. Assume that some team members work remotely, and that it is not possible to get the whole team together with short notice.

I will present two distinct threat models / case studies that illustrate why it is hard but necessary to maintain team contact and communication, especially when distributed.

Redundant Work
In a fast moving project, it can be the case that two or more people are under the understanding that they should work on something. Especially if ticketing is not kept up to date (if using tickets or issues), or if features were accidentally confused when they were assigned, two people can

end up working on the same code or feature. In this event, one of them is completely wasting their time, as the other is already in charge of forming the feature. This creates a situation where time, effort and capital have all been wasted because one person misunderstood or was misassigned to work on a feature someone else was already assigned to and working on.

Specification Mismatch
In a large project, there are many teams or individuals each writing software for a common goal. These teams do communicate, but given geographic distance between them, they have to communicate over voice chat. Unlike working together physically, the effort required to arrange a time (and work with time differentials) makes meeting extremely labor-intensive. For that reason, each team is interested in assuming the most they can without having to call and confirm, or message and wait for a response from other teams. These teams are producing a microcontroller, and one team, Team U is in charge of writing the UART driver. Team A is designing the processor, and Team B is designing the programs to test it. In this case, B has completed their operating system, and it passes all checks - it works. U has also completed their UART software - it too passes all checks and works great. When merged, though, the MCU refuses to boot or even output text - what happened? As it turns out, Team U assumed the baud rate for their UART to be 115200, as that is what all the previous drivers U has written run at. However, the B Team's software provides UART Baud of 38400, because that was the only multiplier their architecture could get from the builtin clock (pretty rustic architecture). Therefore, the processor and code does not work until someone catches this simple mistake. The same thing could happen with different assumptions of calling convention, different instruction set architecture (ISA) versions that added / removed different opcodes, so the processor will no longer run certain commands.

Q.3: Examine an IDE that you have used before and determine if it is possible for two or three people to work on the same project concurrently. If no capability exists within the IDE, describe how you would organize communication between the team members.

I have never used an IDE that features multi-user support. That is, unless you count Google Docs as an IDE, which I do not. Therefore, I have had to refine my team work processes for making sure that everyone can work together on a project without walking all over each other, or missing sections. In the past I have used a Windows development environment (running code in VSCode, as well as running code in a Linux VM hosted locally), using IDE's VSCode and (if it is considered an IDE) Notepad ++ and SublimeText3. I later switched to native Linux, where I use vim or SublimeText3, though neither of these are dedicated IDE's. I am not familiar enough with VSCode to discuss how it could be used as a collaborative IDE, so I will present how a team would work on the same project using a text editor and shell setup like mine.

First, when the project begins, you establish ground guidelines. For example, you explain the branch-and-merge technique of collaborative development, branch naming conventions, cheat sheets on how to perform all the steps in the branch-and-merge process. Also, you make sure that everyone in the team can run and clone your code, and has appropriate access. Once these have been ascertained, the task is divided up into subtasks, which I have historically stored in GitHub or GitLab issues. These tasks include full descriptions, team member assignments and deliverables, as well as linked merge requests to keep track of status. Further, an environment is set up where programmers can chat about their progress (Discord, Mattermost, Slack ...) and ideally voice and video chat too. This is the crux of communication between team members - the ability to ask questions, talk and message others is essential.

Each team member chooses tasks until all tasks corrupt the safety of the country that you are working for if that is the caseare assigned to a team member. Then each person makes a branch as they tackle each task. These branches are merged into master when complete. Unit testing and CI are performed to ensure function remains *before* merges are made. This ideally eliminates the need to perform manual merges, and all merges can be done manually.