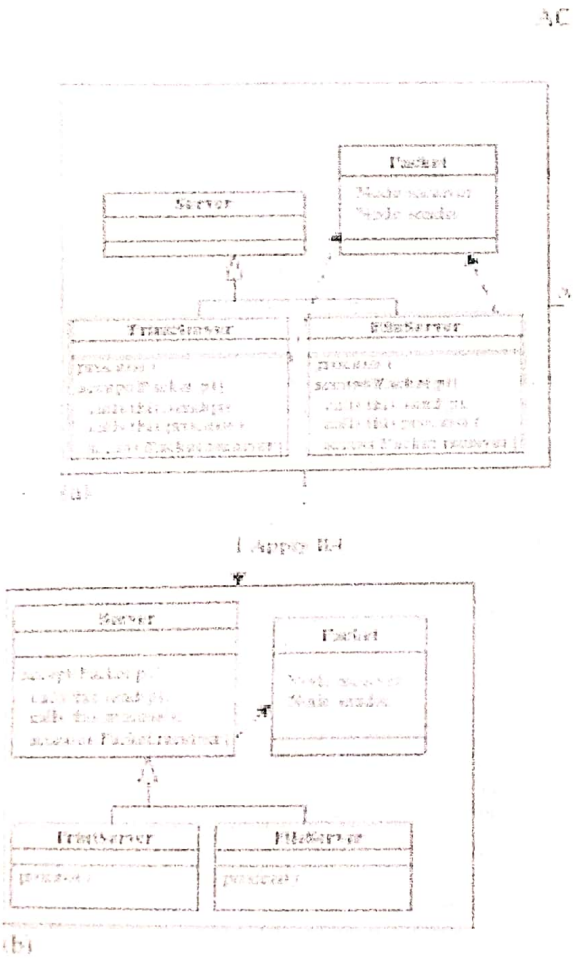
Software Re- Engineering Mid Solution Fall 2022

Q1. Since system A is mission critical, we wish to reduce the impact of the new extraction task as much as we can, while still achieving the aims of the new application. System B is not mission critical, and is to be used for decision support; i.e. for longer term decision making rather than for day-to-day operational matters. We can therefore rule out options a and b. The data to be extracted must aggregate a weeks sales into one set of values, therefore a weekly extraction (answer c) seems sensible, especially if this can be done without impacting on the Monday morning sales processing. If the decision support application will be used only very infrequently, then a monthly migration (answer d) might be considered, but this may cause the extraction period to take too long to fit conveniently into the available extraction windows on system A.

Q2.



Q3.

- Development issues. Highly-skilled developers are required for this task. Initial developers have already switched to other projects or even left the company. However, the company typically has no money to hire new professionals and no time to teach the low or medium-skilled employees.
- Management issues. Legacy software maintenance can stop being a top priority for businesses. In such a case, the top management loses interest in the software support, and it can be hard to justify the need to spend on updating the working code. Possible advantages are remote and not clear to stakėholders.
- Cost risks. It can be hard to make exact cost predictions for the legacy re-engineering project due to a lack of expertise and documentation. It can become even more complicated because there is typically a need to expand the existing functionality, and at the same time, maintain compatibility with previous product versions.
- Potential threats of changing environments. It is always risky to perform "live" changes in the working functionality – in real-time in production environments.

Q4. Reengineering = Reverse engineering +Δ+Forward engineering.

Let us analyze the right-hand portion of the above equation. The first element "reverse engineering" is the activity of defining a more abstract and easier to understand representation of the system. For example, the input to the reverse engineering process is the source code of the system, and the output is the system architecture. The core of reverse engineering is the process of examination of the system, and it is not a process of change. Therefore it does not involve changing the software under examination. The third element "forward engineering" is the traditional process of moving from a high-level abstraction and logical, implementation-independent design to the physical implementation of the system. The second element "Δ" captures alterations performed to the original system.
While performing reverse engineering on a large system, tools and methodologies are generally not stable. Therefore, a high-level organizational paradigm enables repetitions of processes so that maintenance engineers learn about the system.