Abraham Maslow says:

“When your only tool is a hammer, every problem looks like a nail!”

but Merlin Mann says:

“When your only tool is a hammer, every problem looks like a reason to buy a way nicer hammer!”

Relevantly, in this assignment, your only tool is your knowledge of software processes and development methodologies, and the problem is the situation described below. Either try your nicest hammer, or devise a new one which is the most sharpest to bang this nail down!!!
Case Study

Here goes our story: Once upon a time ...

There was a legendary IT division (we call it “BELL”), which was in charge of developing and maintaining the software systems of a private organization (we call it “FORGE”). For ten years, BELL had developed software systems (varying from small-scale to large ones) for FORGE. Due to the prevalence of its IT services, FORGE had a substantial number of service points all over the country and offered diverse on-site and online services to its customers. Since FORGE was not the only organization and provider in its type, the quality and novelty of on-demand IT services was the only competitive edge, and the golden key to survival in the turbulent market environment. In addition to this, the criticality and vitality of the offered IT services had posed BELL as a decisive factor in the success of FORGE.

On the other hand, ten years of working for FORGE, had given BELL the overall control of its software systems. Imagine the considerable benefits that were derived by this control! FORGE had to keep an ongoing relationship with BELL due to several reasons, such as:

- The operation and support of the existing software systems was a critical factor to FORGE which could only be provided by BELL;
- Offering new services by FORGE (in order to survive in the market) required the development of new software systems which were highly dependent on the existing systems (Therefore leaving BELL as the only solution).

However, BELL had deteriorated into a dilapidated ruin over the years:

- BELL managers had turned into nagging corncobs that could do nothing but negating, complaining and obstructing. Hence, no proper management activity such as project management, risk management and quality assurance was performed on the development activities (an antipattern called “project mismanagement”). This put a great deal of pressure and stress on all the poor programmers in charge of developing the software systems. No new idea could convince these corncobs other than tangible results which were delivered swiftly.
- There was no knowledge sharing mechanism between the developers and managers. Developers were kept isolated either from the users and contractors of the software systems (an antipattern called “mushroom management”), or the managers who were in charge of contracting with FORGE for the software projects.
- The burden of developing software systems were just upon the programmers. Hence, the success or failure of a project mainly depended on the skills and knowledge of the developers. This was while the developers of BELL just possessed expertise in some programming languages such “C++” or “Java”, and having heard a little about “RUP” and “UML”. This had resulted into disparate development of software systems with no architecture, design, documentation, configuration management,...; having one system implemented in “X”, while the other in “Y”. Only a few seasoned programmers had developed some ad-hoc architecture of which no
documentation had remained (an antipattern called “architecture by implication”). Occasionally, some UML models were also produced for some systems.

- The maximum number of developers working in a team on a software system was three. This was while due to the amount of stress and pressure which were upon these poor programmers, they quit their job quite often. Quitting the job was a frequent pattern in BELL, posing developers and their knowledge of the systems as a bottleneck. Just imagine the chaotic situation created after the departure of each developer: on one hand a series of critical and vital systems in desperate need of operation and support, one the other hand no documents, no test and the developers which were quitting one after the other.

- Maintenance was a nightmare for BELL. Only a few programming heroines (heroes) could support the software pieces developed by their predecessors. However, the coming and going of these heroines (heroes) in addition to the ever-changing needs of the developed software systems had resulted into a master piece of dead codes (an antipattern called “lava flow”), and software systems with ad-hoc architectures (an antipattern called “Spaghetti Code”). Just imagine what stove-pipe services were offered by FORGE!!! Manipulating these systems and services was just like “Walking through a mine-field”.

This sad story continued until a heroine (or probably hero) of “Software Development Methodologies” appeared 😊. She (or probably he) turned BELL from a bankrupt division into a flourishing IT company throughout the country (As you know this is a myth, therefore a little bit exaggerating is quite natural 😃).

**Problem Statement**

*Can you explain in details what this heroine (or probably hero) did?*