CONCORDIA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

SOEN 6431: SOFTWARE COMPREHENSION AND MAINTENANCE: SECTION BB
SUMMER 2022

DÉJÀ VU

1. INTRODUCTION

The purpose of this project is **learning to make critical decisions** through theory and practice of reengineering the source code of a selected system. In doing so, the project studies the maintainability (as defined by the ISO/IEC 25010 Standard) of the selected system.

The source code of systems[1] that have proven to be **useful** also tend to be **modified relatively frequently** for some purpose, including enhancements. These modifications can lead to **deterioration of quality**[2], specifically, **deterioration of maintainability** and/or **deterioration of (space and/or time) efficiency**. It could be noted that, in general, maintainability is of interest to a software engineer and efficiency is of interest to a user (due to its relationship to performance, and thereby to usability).

The deterioration of quality of the source code of systems usually manifests itself as **'undesirables'**. These undesirables include, but are **not** limited to, 'smells'[3].

The deterioration of quality of the source code of systems can be **ceased, even reversed**, by reengineering. There are several techniques for **reengineering**, one of which is **refactoring**. The notion of refactoring is **independent** of any specific programming language or programming paradigm. In the past decade or so, there has been much interest in refactoring the source code of systems (especially those based on object-oriented programming languages) to **patterns**.

2. PROJECT-LEVEL CONCERNS

A team must make use of **project management body of experiential knowledge**, such as **collaboration patterns and thinklets**. For example, this can be done through practice and documenting such practice.

[1] In this document, a 'system' could either be a 'computer program' or a 'software system'.
[2] This phenomenon has come to be known by several metaphors including software aging, software bloat, software brittleness, software entropy, software erosion, software rot, (accrual of an unacceptable amount of) technical debt and so on.
[3] In this document, 'smell' and 'bad smell' are synonymous and therefore interchangeable.

DELIBERATION

The work towards Déjà Vu must be **visible**. It is imperative that each team **meet** regularly (at least weekly), as well as use **Social Software**. For example, a **Wiki Hosting Service** is such social software. The selected social software should be used by a team for **communication as well as collaboration**. The **roles and responsibilities** to be carried out by team members pertaining to each deliverable should also be posted on the selected social software. Let S(X) be social software being used by Team X. Then, the roles and responsibilities to be carried out by team members pertaining to each deliverable must be made available publicly, as early as possible (and no later than 72 hours before the submission of that deliverable).

CONFIGURATION

The source code of **R** must be placed in a **source code repository** that has support for **version control**. The versions should be available to and shareable by the entire team. For example, **GitHub** is a service that supports **Git**, a version control system, and provides **GitHub Gist**, a shareable pastebin.

ATTRIBUTION

It is necessary for a team to strive for the highest standard of academic ethics. To do that, a team must **cite and reference appropriately** any and all non-original work (that is, any work external to that team). A comprehensive collection of resources on citing and referencing is available[4]. It is important not to make claims that cannot be substantiated, and not to copy others' work verbatim regardless of whether it is cited. A copied work does not earn any credit.

The work on Déjà Vu has been divided into following deliverables:

- Reengineering Opportunity
- Reengineering Operationalization
- Reengineering Outreach

The details of these deliverables are discussed in the sections that follow.

3. PEOPLE-LEVEL CONCERNS

Déjà Vu is a team project involving individual as well as communal work.

[4] URL: http://library.concordia.ca/help/howto/citations.html .

4. DELIVERABLE 1 (D1): REENGINEERING OPPORTUNITY

DESCRIPTION

The team must explore and select one system to be a candidate for reengineering. (Let N be the **team size**. Then, N systems must be sought, one by each member of the team, before a system is selected.) Let this candidate be **R**. **R** is subject to a number of constraints, given in the following.

R must have been pursued as a non-commercial endeavor, in a setting that is unrelated to Concordia University. In particular, **R** must **not** be from any course work from within Concordia University.

The **domain** in itself of **R** is not significant. For example, **R** could be a game, a data science program, a graphics program, a machine learning program, a parser, a scientific computing program, or a system utility. However, **R**, specifically its **problem and solution domains**, must be **understandable and communicable by all members** of the team.

R must be **unique** across teams.

The source code for **R** must be **available publicly and openly** on the Web. There are several publicly hosted **source code repositories**, including, but not limited to, **Assembla, Bitbucket, GitHub, Google Developers, Rosetta Code, SourceForge**, and so on.

It is also possible to use **curated repositories**, such as those currently available for Java and Python.

The source code for **R** must be in one of the following programming languages, say, **L**:

- C
- C++
- Java
- JavaScript
- PHP
- Python

The size of the source code for **R** must be between **1 to 2 KSLOC**. (It is better to choose an **R** of a **non-trivial but manageable** size. Therefore, a size of **R** beyond the specified limits is not recommended).

In case of multiple versions of **R**, the **latest version** must be selected.

The source code for **R** must have at least **5N** unique undesirables. (The number of undesirables is negotiable, as the number itself is less important and the **type of undesirables** is more important. However, the burden of proof of suggesting a convincing number is on each member of the team.)

SUBMISSION

- A brief description of **each** of the N systems selected initially (except **R**), along with the URLs of their source code and a brief rationale for rejecting each.
- A detailed description of **R**, including its purpose and the programming language its source code is expressed in.
- A rationale for selecting **R**.
- The URL of the source code for **R**.

5. DELIVERABLE 2 (D2): REENGINEERING OPERATIONALIZATION

DESCRIPTION

There must be **manual and/or automatic identification** of undesirables in the source code for **R**. The two approaches are **complementary** rather than competing, as humans and machines each have their own strengths and weaknesses.

For example, undesirables could include, but are not limited to, presence of anti-patterns, smells, and/or violations of programming principles, programming style, programming conventions, or coding standard, as applicable, pertaining to **L**.

In case of the use of **tools**, it is important to be aware of the limitations of the tools, including the potential for **false positives and false negatives**.

In case the undesirables are **smells**, there must be application of appropriate refactoring methods to eradicate those smells in the source code of **R**.

The source code of **R** must be **debugged, reviewed, and tested** after the application of **each** change (such as, application of a refactoring method). (This is in agreement with the **Lehman's Law of Conservation of Familiarity** as well as **iterative and incremental change**.)

SUBMISSION

- An **organized list of source code undesirables and their definitions**. (An undesirable type can repeat across a team, but **not** across a team member. There must be at least one example of each undesirable type in the source code for **R**. The source, such as an article, book, or a Web Site, of the definitions of source code undesirables must be given.)
- An **organized list of reengineering methods and their definitions**. These reengineering methods could include, but may not be limited to, refactoring methods corresponding only to the source code smells. (The source, such as an article, book, or a Web Site, of the definitions of refactoring methods must be given.)
- A table showing the **mapping** between source code undesirables and reengineering methods.
- A report of **exact locations of the source code undesirables** in the source code for **R**, and the results after applying the reengineering methods. This report could also include source code smells that were discovered, but for which currently **no known refactoring methods** exist.
- A report of a **change in any relevant software metric**, before and after reengineering.
- A report on the specifics of **debugging, reviewing, and testing**.
- A **list of any tools used**, such as for the identification of undesirables, along with the details of their usage, (integrated) development environments, source code visualizers, quality management systems, and/or special-purpose utilities.

- A copy of **R** before reengineering.
- A debugged, reviewed, and tested copy of **R** after reengineering.

6. DELIVERABLE 3 (D3): REENGINEERING OUTREACH

DESCRIPTION

This could be **either** a **screencast** or a **presentation**.

SCREENCAST

The purpose of the screencast is to demonstrate a **representative use case** of **R** before and after reengineering. The screencast should be uploadable on a **Video Hosting Service**, such as **YouTube**.

The screencast should aim to be **educational**.

PRESENTATION

This could be **either** a collection of **slides** or a **poster**.

The purpose of the presentation is to defend the work done towards the project. In particular, the presentation must include **critical decisions** made throughout the project. (For example, such decisions could be about both doing as well as not doing something, challenges in selecting a tool, and so on.) It must also comment on the maintainability (as defined by the ISO/IEC 25010 Standard) of the selected system. It may also comment on **(internal or external) documentation** available on **R**, and **team retrospective**, if any. (For example, retrospective could include lessons learned and highlights of things that would be done differently if the project were to be pursued again.)

NOTES

The **teams in the audience** must **prepare (in real-time or otherwise) one question per team member and pose those questions** to the team that is presenting.