A P E G S

Association of Professional Engineers & Geoscientists of Saskatchewan

Software Engineering Experience Review Guideline

Approved April 11, 2014

This guideline is used in combination with the following:

- *Experience Guideline 1 – Experience Reporting for Members-in-Training, Supervisors and Mentors;* and
- *Experience Guideline 2 – Components of Acceptable Engineering Work Experience.*

It is not meant to fetter the discretion of the Experience Review Committee (ERC) when considering any individual report. Its purpose is to improve transparency and consistency in decisions, not to duplicate past decisions. A recommendation to the Registrar as to experience credit toward professional registration granted or withheld is at the discretion of the ERC.

Software engineering work is considered acceptable engineering work for the purpose of experience credit toward professional registration where the approach taken is guided by engineering training, analysis, design and engineering supervision or mentorship. It involves "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software."[1]

Software engineering work considers the **process** used rather than strictly the ends. The software engineering process (also called the software design life cycle) includes software requirements analysis, software design and construction, software process engineering, software quality assurance, software assets management and maintenance, and software project management.

The software development may occur in a wide variety of fields, including but not limited to telecommunications, entertainment, health, agribusiness, manufacturing and defense. The subject for which the software is being designed is irrelevant and it is not necessary for a tangible object to be the subject of the design. For instance, development of software to manage financial systems may be acceptable engineering work experience.

Software engineering work may involve but is not limited to the following application areas:

a. business systems
b. internet systems
c. transaction systems
d. interactive multimedia systems
e. management systems
f. information systems
g. industrial applications
h. corporate software process
i. embedded systems construction
j. application of computational theories to real-life programming techniques

The Institute for Electrical and Electronics Engineer' (IEEE's) *Guide to the Software Engineering Body of Knowledge*[2] has elaborated those activities which are part of a professional software development process. They include the following:

- Definition of software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management (i.e. software project management)
- Use of software engineering processes (i.e. software lifecycle, process and product assessment and measurement)
- Use of software engineering tools and methods
- Software quality management

Engineering Principles[3]

Determining if software engineering principles are required involves an assessment of the development process and of the characteristics of the software product itself.

When looking at the software development process, consider what should have been undertaken over the life cycle of the software product. Look for both general engineering principles and software engineering principles. The software engineering principles are what make the work more than just programming. They include the following:

- Software design patterns / process analysis and program definition
  The software or system design and the various practices used during the lifecycle are defined and selected with care, as appropriate for the risk and the other aspects of the engineering problem.

- Management of risk
  Risks of various kinds are analyzed, and managed according to the engineering problem. Risks impacting public health safety and welfare are managed as a top priority, but technical and financial risks are also considered.

- Validation and verification of work
  At each stage of the lifecycle the scope of professional software engineering includes validation and verification of the software or system, including its components and intermediate work products.

- Reliability and repeatability of the process used
  Specific processes are defined / selected for use during the lifecycle that are appropriate for the engineering problem and level of risk.

- Assumption of responsibility / traceability / liability
  The software or system may be composed of many components and have interfaces to other devices or systems, all of which must function reliably and appropriately. The assumption of responsibility for a system (engineering work) includes the assumption of responsibility for components and associated systems

Since the actual development process that was used is not always known, we can also consider characteristics of the software product. If the software exhibits certain factors, then engineering principles were **likely** required to produce it. Any one of the following factors can require the application of software engineering principles:

- Uniqueness
  Is the product particularly novel? Products for which the software design follows familiar patterns, and can likely be implemented using well known, high level tools, will likely not constitute professional software engineering. Development of products where the design is not so obvious, or requires integration of technologies in novel ways, would be more likely to constitute professional software engineering.

- Complexity of project
  The overall complexity of the project, which could be indicated by the number of people involved in the development effort or the size of the design artifacts (e.g., source code), or the number of connected elements (both physical and virtual) gives an indication of the need for engineering principles.

- Appearance to end user
  Applications where software forms part of a larger system (*embedded* software), such that the user of the system is not normally aware of the software, instill a stronger expectation of correct behaviour and hence a higher standard of care is required in the development process. For example, we all expect that the accelerator and brake pedals on our cars function as expected all of the time, and probably don't often consider that there is a significant amount of software involved in these systems and that a significant amount of engineering knowledge of the electromechanical characteristics of those components is required to write the software.

- Interdisciplinary nature
  Software development where understanding the problem domain requires knowledge of other disciplines may also be professional software engineering due to the broad foundations in the physical sciences that are essential to successful development of such systems. This also applies to what is often called *engineering software*—software that incorporates some knowledge of other engineering disciplines in its internal algorithms, such as may be used in performing design calculations or simulations. Although it is clear in this case that the engineer taking responsibility for the final product (i.e. that the software is being used to design) is responsible for the outputs of the software, there is a clear application of engineering principles so development of such software can be deemed to be within the practice of engineering.

- Regulatory environment
  If the software was developed for and is operated in a highly-regulated industry (e.g. aviation, nuclear, etc.) then it is quite likely that software engineering was involved and that engineering principles were required.

The fact that a product exhibits one of these characteristics does not mean that it *must* be software engineering, but exhibiting one or more is certainly a sign that it *could* be.

References

[1] Institute of Electrical and Electronics Engineers, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990, New York, NY. http://standards.ieee.org/findstds/stadnard/610.12-1990.html
[2] Institute of Electrical and Electronics Engineers, *Guide to the Software Engineering Body of Knowledge* (SWEBOK), 2004. http://www.computer.org/portal/web/swebok/2004guide
[3] Adapted from Canadian Engineering Qualifications Board, Software Engineering Task Force, *Professional Practice in Software Engineering*, Draft July 16, 2013.