Entropy Guidelines

Document #:	Draft 0.1
Date:	2019-04-07
Project:	Programming Language C++

Reply-to:	Jérémy Morosi <jeremymorosi@hotmail.fr>

There must exist universal coding guidelines reflecting how nature works.

1 Introduction

2 Motivation and Scope

These guidelines emerged from the fact that we often tend to write big monolithic codebase that we can hardly maintain and we can't remember/explain how it works days later. The goal is to prevent such things to happen by providing coding guidelines that - when applied correctly - help us to write code that mimic how nature works; thus producing a high amount of entropy.

But why entropy ? Think about the efforts required to keep a house clean. It naturally tend to be in disorder because this is simply how entropy works. Cleaning a house is like lowering the amount of entropy by reorganizing everything. Such process requires a lot of energy and it goes the same for your code.

If you are afraid of your code becoming obsolete or afraid of having someone else working on it because you know it's too complex for him to understand, it's because you are trying to maintain a codebase with too much functionalities, responsibilities or organization; which is equivalent to trying to keep it's amount of entropy low by spending a lot of efforts and energy.

Now, what would a codebase with a hight amount of entropy be like ? Functionalities would be written as separate and independant subprojects. Understanding how a functionality works and is written would require little effort and it would be easy to write unit tests covering as much code as possible. The best is that you wouldn't be afraid of having to rewrite your code if it became obsolete or broken.

This is what these guidelines are about. Not how writing high performance code in a specific language, but how organizing it to result into a high amount of entropy, requiring less efforts to maintain.

3 References