By using `[[deprecated]]` as shown in the previous example, existing clients of `RandomGenerator` are informed that a superior alternative, `BetterRandomGenerator`, is available, yet they are granted time to migrate their code to the new solution rather than having their code broken by the removal of the old solution. When clients are notified of the deprecation (thanks to a compiler diagnostic), they can schedule time to rewrite their applications to consume the new interface.

Continuous refactoring is an essential responsibility of a development organization, and deciding when to go back and fix what's suboptimal instead of writing new code that will please users and contribute more immediately to the bottom line will forever be a source of tension. Allowing disparate development teams to address such improvements in their own respective time frames, perhaps subject to some reasonable overall deadline date, is a proven real-world practical way of ameliorating this tension.

Potential Pitfalls

Interaction with treating warnings as errors

In some code bases, compiler warnings are promoted to errors using compiler flags, such as `-Werror` for GCC and Clang or `/WX` for MSVC, to ensure that their builds are warning-clean. For such code bases, use of the `[[deprecated]]` attribute by their dependencies as part of the API might introduce unexpected compilation failures.

Having the compilation process completely stopped due to use of a deprecated `entity` defeats the purpose of the attribute because users of such an `entity` are given no time to adapt their code to use a newer alternative. On GCC and Clang, users can selectively demote deprecation errors back to warnings by using the `-Wno-error=deprecated-declarations` compiler flag. On MSVC, however, such demotion of warnings is not possible, and the available workarounds, such as entirely disabling the effects of the `/WX` flag or the deprecation diagnostics using the `-wd4996` flag, are often unsuitable.

Furthermore, this interaction between `[[deprecated]]` and treating warnings as errors makes it impossible for owners of a low-level library to deprecate a function when releasing their code requires that they do not break the ability for *any* of their higher-level clients to compile; a single client using the to-be-deprecated function in a code base that treats warnings as errors prevents the release of the code that uses the `[[deprecated]]` attribute. With the frequent advice given in practice to aggressively treat warnings as errors, the use of `[[deprecated]]` might be completely unfeasible.