January 2022

Preserving Manual Modifications to Auto-generated Code Across Versions

Patrick Blesi

Preserving Manual Modifications to Auto-generated Code Across Versions

ABSTRACT

When an automatic code generator updates code that has been manually modified, the manual edits get overwritten and effectively lost. This is a problem for test and development engineers. In such situations, the user (test/development engineer) must laboriously re-apply the changes they made manually for each auto-generated update. This disclosure describes techniques that preserve manual edits as auto-generated code undergoes automatic updates. The developer inserts sentinel values into the code, indicating lines that have been manually inserted or deleted. During automatic update, the sentinel values are used to recover the base file. An automatic code generator automatically generates a new revision of the file. The base file, the new revision, and the file with manual edits are merged using a three-way merge utility to obtain auto-updated code that includes the previously-made manual edits.

KEYWORDS

- Auto-generated code
- Code generation
- Code merging
- Manual edit
- Sentinel value
- Code annotation
- Test case
- Source control
- Version control

BACKGROUND

Software and hardware testing often relies on auto-generated source code, e.g., high-level language (C++, python, etc.) code that is automatically generated using code generation programs. Auto-generated code can be used, for example, to generate test cases, and can be automatically updated based on user-provided configurations. Sometimes, auto-generated test cases (in the form of code) require manual modifications, e.g., to support features unsupported by the automatic code generator.

When an automatic code generator is used to update a test case that has been manually modified, the manual edits get overwritten and are effectively lost. This is a problem for test and development engineers. In such situations, the user (test/development engineer) must laboriously re-apply the changes they made manually for each auto-generated update.

Source control solutions used for merging disparate changes into a unified source file are tangentially related to the problem under consideration. However, current source-code management techniques do not specifically address this problem.

DESCRIPTION

This disclosure describes techniques that preserve manual edits as auto-generated code undergoes automatic updates. When a user manually inserts lines into auto-generated source code, the manually added line is appended with a sentinel added by the user. An example sentinel can be a comment such as “# MANUAL_EDIT_ADDITION” included by the user on the same line as the manual insertion.

In order for a user to effectively delete a line from the auto-generated source code, the deleted line stays in the generated source code while being inactive. To prevent the line from being interpreted by the compiler or processor, the deleted line is commented out by the user and

is appended with an end-of-line sentinel such as “# MANUAL_EDIT_DELETION.” Manually modified lines in auto-generated source code are represented by a deleted line followed by an added line with sentinel annotations.

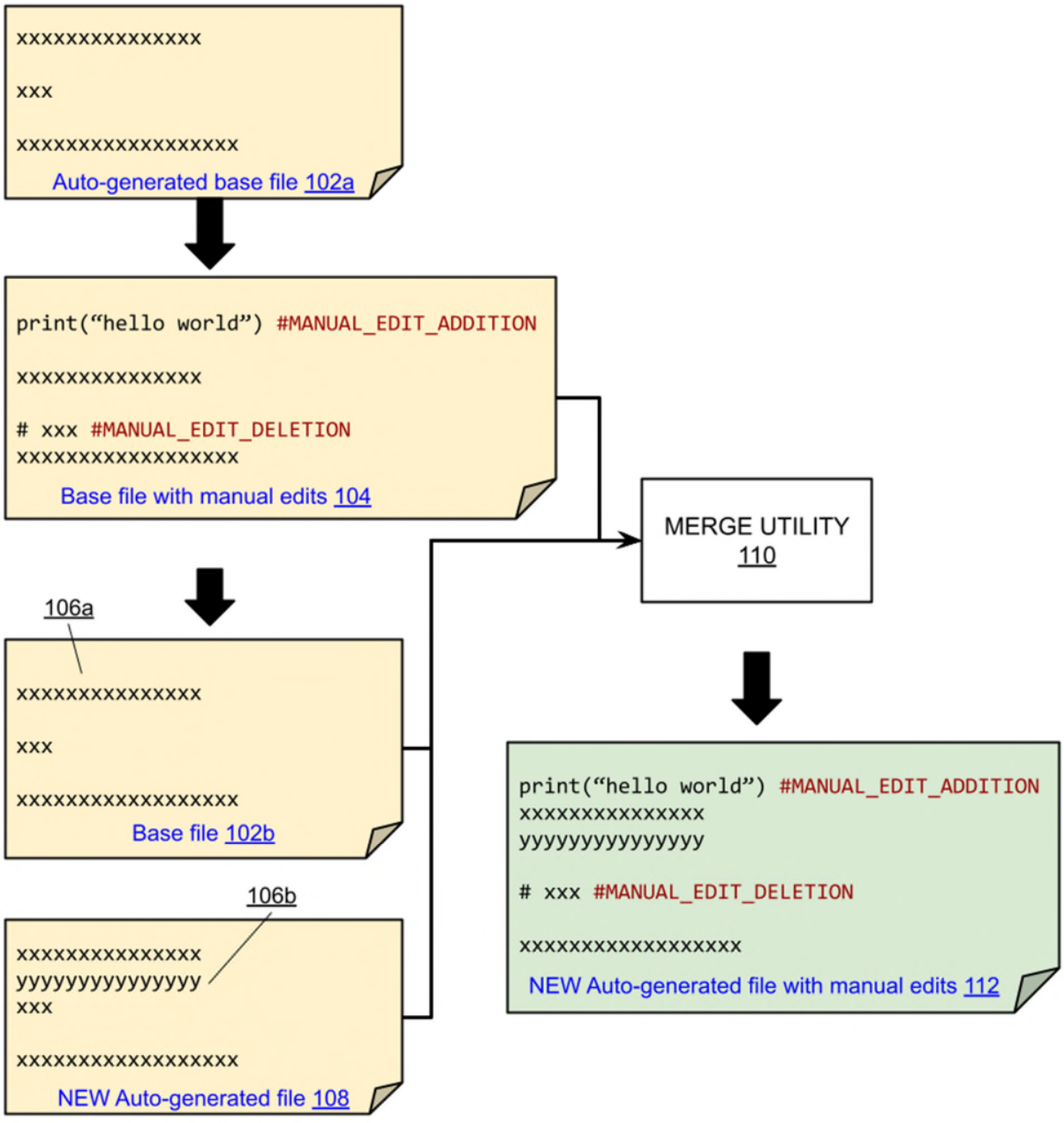

Fig. 1: Maintaining manual modifications to auto-generated code

With the sentinel annotations added by the user as described above, automatic regeneration of source code automatically incorporates manual modifications, as illustrated in Fig 1. An auto-generated base file (102a) comprises purely auto-generated code (indicated by lines with ‘x’s). The user introduces manual modifications, e.g., the line `print("hello world")`, that are annotated by #MANUAL_EDIT_ADDITION and #MANUAL_EDIT_DELETION to obtain a base file with manual edits (104).

To preserve the manual edits as the file is auto-updated, manually added lines (lines with the #MANUAL_EDIT_ADDITION sentinel) in the base file with manual edits (104) are automatically deleted. Manually deleted lines (lines with the #MANUAL_EDIT_DELETION sentinel) have their leading comment stripped off and trailing sentinel deleted. This results in restoration of the base file (102b). With manual edits removed, the resulting source code acts as a restored, purely auto-generated base file with no manual edits or updates from newly generated test cases.

The base file (102b) is auto-updated to generate a new auto-generated file (108) that reflects new configuration information or new test cases. In the example of Fig. 1, line (106a), comprising a string of ‘x’s, is automatically appended with line (106b), comprising a string of ‘y’s. The new auto-generated file (108) is derived purely from the updated configuration information (including potential new test cases).

The three versions of the source code file, namely, the base file without manual edits (102b), the version with manual edits (104), and the version with the auto-generated updates (108) are merged using a merge utility (110) to generate a merged file (112). The merged file is a new auto-generated file that retains the manual modifications made on the previous version (102a) of the auto-generated file.

Merging the three source versions can be done using a merge utility such as the GNU diff3 three-way merger. In places where only manual edits exist, they are automatically merged into the resulting output. In places where only auto-generated updates exist, they are automatically merged into the resulting output. For lines where both manual and automatic updates exist, conflict-resolution outputs are displayed showing the lines from all versions where conflicts exist, such that conflicts can be manually resolved by the user.

The described techniques can be applied to any generated files or data where manual changes are applied after auto-generation and there is a need to provide subsequent automated updates to the same files. The requirements are that the manual edits should include annotation sentinels indicating which lines were manually added or deleted and that a mechanism is available for the underlying processor (compiler, etc.) to ignore the manual edit sentinels and deleted lines.

While the foregoing description relates to automatic code generation where manual edits to individual lines of code - additions or deletions - are preserved between versions, it is also possible to include block-level sentinels that indicate if a set of contiguous lines of code have been manually added or deleted. The block-level sentinels function similar to the line-level sentinels and can be used to preserve manually edits across versions of automatically generated code.

In situations where auto-generated and manual code modifications occur together, it is often difficult to produce auto-generated solutions that produce the exact code (including manual insertions that persist across updates) necessary for execution. The described techniques provide a streamlined solution for incorporating manual edits. An example use case, in addition to the

test-and-development situation herein described, is auto-generated client libraries targeting different programming languages.

CONCLUSION

This disclosure describes techniques that preserve manual edits as auto-generated code undergoes automatic updates. The developer inserts sentinel values into the code, indicating lines that have been manually inserted or deleted. During automatic update, the sentinel values are used to recover the base file. An automatic code generator automatically generates a new revision of the file. The base file, the new revision, and the file with manual edits are merged using a three-way merge utility to obtain auto-updated code that includes the previously-made manual edits.