Automatically Enforcing Quality in Ada Software

Mark Dowson, Marlstone Software Technology Inc

Ada Quality and Style (AQS), and any similar comprehensive set of Ada programming style guidelines, includes few guidelines that can be automatically enforced, i.e., guidelines whose violation can be reliably detected and corrected without user intervention. Nonetheless, effective and useful support can be provided to help Ada developers conform to (or monitor conformance to) the majority of guidelines. Over the last eighteen months, Marlstone Software Technology has been developing an Ada Quality Toolset (AQT) to provide such support.

Support for a style guideline involves two dimensions: detection of guideline violations and correction of the code to eliminate the violations. Different degrees of support for both detection and correction are possible for different kinds of guidelines. Depending on the guideline, an AQT tool may be able to detect *actual* violations or only *potential* violations. For example:

> Detection of actual violations possible: "Avoid exit statements in while and for loops."
>
> Detection of potential violations possible: "Provide a way to avoid raising an exception."

In the former case, all occurrences of exit statements in while and for loops can be detected and presented to the developer either in the form of a report or interactively. In the latter case, AQT can only present information about each exception and the context in which it can be raised and handled, to facilitate human judgement as to whether the guideline has been violated.

AQT can support the correction of code to eliminate guideline violations in a number of ways. Correction can be *automatic*, *semi-automatic*, or *manual*. For example:

> Automatic correction possible: "Minimize the context (with) clauses in a package specification."
>
> Semi-automatic correction possible: "Do not use anonymous types."
>
> Only manual correction possible: "Ensure elaboration of an entity before using it. Use function calls in declarations cautiously."

In the first case, AQT can automatically eliminate unnecessary context clauses or move them to the package body. Automatic corrections of this kind are likely to be performed incrementally, with the developer reviewing each change. Other kinds of automatic corrections, e.g., those involving source code presentation and layout, can be performed in "batch" mode without user intervention.

In the second case, a developer must supply type names for the detected anonymous types, but a tool can then insert the names in the appropriate places in the source. In the third case, while AQT can report potential violations of the guideline for human review, automated correction is not practical (or necessarily desirable).

To allow AQT to work with a variety of compilers and development environments, AQT is based upon ASIS, the emerging standard for representing Ada compilation information. A set of information extraction, report generation, and source transformation utilities reads the contents of an ASIS library. These utilities provide the basis for guideline violation detection, violation reporting, and both automatic and interactive transformations of source code to conform to selected guidelines.

AQT is intended to be used as an integral part of the development process, supporting the needs of a variety of different kinds of user including: programmers; software maintainers and developers involved in reverse engineering; inspection teams; software development managers; and quality assurance engineers.

No tool can replace the need for careful design and implementation. However, automated support can help ensure quality and consistency, automate tedious manual tasks, and help engineers to check and review large volumes of source code methodically, to search for information and certain classes of errors, and to improve the efficiency and impact of human review.