第2章 設計書とコードが一致したプログラムを対話式修正で作り上げる

Copilot×Doxygenで始める現代版文芸的プログラミング

村井 和夫

Copilotで所望のソースコードを生成するためには，入力として適切なコメント（＝ドキュメント）を与えることが必要です．また逆に，ソースコードからコメントを生成し，ドキュメントを生成することもできます．これにより，Copilotではドキュメントとソースコードが同時に開発されます．

これはドキュメントとソースコードを同時に開発し，保守する手法であるDonald. E. Knuth博士の「文芸的プログラミング」に通ずるものがあります．文芸的プログラミングではドキュメントからソースコードを手動で記述していましたが，Copilotを使えば，この部分を自動化できます．

そこで，より良いソフトウェアを書くために，文芸的プログラミングの考えを利用してCopilotを使う方法について考察します．（編集部）

GitHub Copilot（以下Copilot）は今までの統合開発環境が提供するような単純な関数や変数のガイド，補完にとどまらず，LLM（Large Language Models）ならではの機能として実際のプログラミングやドキュメント作成まで行ってくれる機能を持っています．

提示されたプログラムの内容を理解，検証しないままこの機能を使うと，プログラムを粗製乱造する危険性もあります．実際，基本的に「わかりません」という答えはなく，間違ったプログラムを提示してきます．そのため，提示されたプログラムを理解しないで使うと非常に危険です．

一方，うまく利用することによって，従来のプログラム開発が抱えている問題を根本的に解決してくれる可能性を秘めています．

そのため，現在のプログラム開発の抱えている問題点をまず検証して，その上でどのように解決していけるかを見ていきます．

現在のプログラミング環境の問題

現在のプログラム開発環境は，プログラム開発にかかわる全ての機能を，1つのGUIから提供する統合開発環境が主流です．統合開発環境によって，初心者でもプログラム開発が非常に楽になったのも事実ですが，今の統合開発環境の仕組みでは解決されない多くの問題も残っています．

● 安易なプログラム開発が行われている

統合開発環境では，プログラムの編集作業とコンパイル，実行，デバッグがほぼ同じ画面でできます．プログラム言語やライブラリに依存した予約語や関数，また，その引数だけでなく，その定義場所や内容まで事細かに参照できるようになっています．

このため全体の設計をろくにせず，言語仕様やライブラリの機能もよく確認せずに，いきなりプログラムを書き出して，少し書いてはコンパイル，実行，デバッグを行うような品質の低い行き当たりばったりの開発が横行しています．

Copilotも使い方によっては，次から次へと正しいかどうか分からないソースコードを提案するので，提案してくるソースコードの良し悪しを適切に判断できないと，誤ったプログラム開発をさらに助長しかねません．

しかし正しい使い方さえすれば，プログラムの品質を高くするのに非常に有効です．

● プラットフォームごとに開発環境が異なる

WindowsではVisual Studio，macOSやiOSではXcode，組み込みソフトウェア開発には基本的に各社の提供する統合開発環境が必要となります．汎用的な統合開発環境としてEclipseなどもあり，さまざまな統合開発環境のプラットフォームが存在しています．それぞれ特徴があり，複数の環境を常時使い分けるのは大変です．

最近では，マイクロソフトが提供しているオープンソースのエディタVisual Studio Codeが，各種プラットフォームで使え，JavaScriptによる拡張機能により統合開発環境並みの機能を持たせることができるため，普及してきています．しかし，この拡張機能開発が容易なことが原因で，さまざまな機能拡張が出てきて機能の競合が起きるという問題も起こしています．