未来に向かって

自動プログラミング，自動バグ修正への夢

11

基般

肥後芳樹（大阪大学） 鵜林尚靖（九州大学）

自動プログラミング

ソフトウェア工学研究の長年の夢として，自動プログラミングがある．その歴史は古い．コンピュータソフトウェア研究の黎明期ではコンパイラが自動プログラミングと呼ばれていた．機械語によるプログラミングが当たり前の時代において，IBM による 1957 年の FORTRAN コンパイラのリリースは夢のような出来事だったと思う．人間にとって読みやすいプログラム記述から自動的に機械語に変換してくれるコンパイラ技術は，その後のコンピュータの歴史において重要な役割を担うことになる．その後，プログラムではなくソフトウェア仕様や設計モデル記述からのコード生成へとソフトウェア工学の技術が発展したのは周知の事実である．しかしながら，今ではコンパイラのことを自動プログラミングと呼ぶ人はいない．また，モデル駆動開発によるモデルからのコード生成を自動プログラミングと呼ぶ人もいないと思う．それでは，現在のプログラマが期待する自動プログラミングとはどのようなものであろうか？

昨今，機械翻訳ソフトあるいは Web 公開サービスを使用する機会が増えている．必ずしも期待した翻訳精度ではないものの，ちょっとした翻訳には便利である．Web から文章を入力すると，英語等に変換してくれる．もし，これをプログラミングに置き換えてみるとどうであろうか？

不完全なプログラムコード（あるいはバグ付きのコード）を入力すると，不足しているコード片を自動的に補ってくれたり，バグを自動的に修正してくれると，多くのプログラマは喜ぶであろう．また，すでに動作しているソフトウェアを止めずに，自動的にプログラムを修正してくれる仕組みがあると便利であろう．現在のソフトウェア工学の技術では，このようなことが可能になりつつある．実際，ICSE（International Conference on Software Engineering）などのトップカンファレンスではホットな話題の 1 つになっており，毎回，複数のセッションが設けられている．今まさに，世界中の研究者がこの夢に取り組んでいる．

遺伝的プログラミングを利用した自動バグ修正

発見したバグを人の手を介さずにプログラムコードから取り除くことを自動バグ修正と呼ぶ．自動バグ修正の研究でブレイクスルーをもたらしたのは遺伝的プログラミングを用いた自動バグ修正手法 GenProg[1] である．

遺伝的プログラミングとは，一言でいえば生物の進化を模すことにより解に到達するためのアルゴリズムである．ここでの解とは，与えられたすべてのテストケースを通過することである．GenProg ではプログラムコードを生物の個体とみなす．プログラムコードはテストケースを実行することによりそれがどの程度解に近い状態にあるかが評価される．より解に近い状態のプログラムコードがいくつか選別され，それらから次の世代のプログラムコードが生成される．すべてのテストケースを通過するプログラムコードが生成されるか制限時間に達するまでこの操作が繰り返し行われる（**図 -1**）．

自動バグ修正では入力されたバグのあるプログラムコードはテストケースを用いて評価され，どの部分がバグの原因であるかの当たりがつけられる．

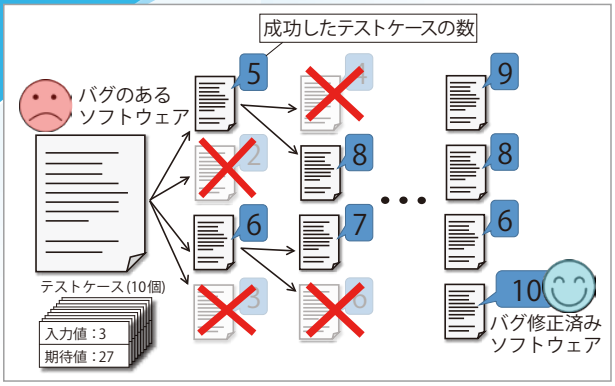

図-1　GenProg による自動バグ修正の様子

GenProg では次の世代のプログラムコードを生み出す操作は比較的単純な方法で行われる．当たりをつけたコードを削除する操作や，ランダムに選択されたほかのコード片を当たりをつけた個所に挿入する操作が行われる．

しかしながら，現時点では，自動バグ修正技術は修正可能なバグの数が少ない，生成されたプログラムコードが汚くその後の保守作業がしづらいなどの課題がある．実際のソフトウェア開発現場で用いるにはまだ未成熟な技術である．

現在多くの研究者が自動バグ修正の研究に取り組んでいる．バグを生みやすい条件分岐の条件式に特化した修正[2]，読みやすさ向上のための基本ブロック単位でのコードの移植[3]等が提案されている．また，プログラムコードを特定のルールに従って合成する自動バグ修正手法[4]も提案されている．

プログラムコードの変更はバグ修正のためだけではない．新しい機能を追加するための変更や，将来の保守作業に備えるためのリファクタリング，ソフトウェアを新しい環境で動作させるための移植作業もコードの変更を必要とする．各種類の変更を支援する研究が活発に行われており，また変更の種類によらずプログラムコードの次の変更後の状態を予測する研究も行われている[5]．この技術が実現すれば，次の変更でどのようにプログラムコードを変更すればよいのかをプログラマに自動的に提示できるようになる．プログラマがその変更を承認すれば自動的にコードが変更される．提示されたコードに良くない部分があればその部分のみを開発者自身で変更すればよい．

群衆知による自動プログラミング

自動プログラミングや自動バグ修正で新たな鍵となるのが，群衆知（Crowd Knowledge）である．現在，プログラマの多くは，コード作成やバグ修正に困ったとき，開発支援ツールが提供するプログラム解析情報を参照したり，デバッガでプログラムの振舞いを確認したりする．それでも解決方法が分からない場合は，インターネット上に遍在する群衆知に助けを求める．たとえば，Stack Overflow のようなソフトウェア開発者向けの Q&A サイトにアクセスしたり，類似するコードがオープンソースソフトウェア（OSS：Open Source Software）にないかを検索したりして問題解決を図る．Q&A サイトから類似コードを自動的に検出し，その解決策をプログラマに提示するような技術はすでに開発されている[6]．

群衆知をさらに高度に利用するためには，**図-2** に示すような「群衆知の循環による群衆ソフトウェア開発エコシステム」を形成する技術開発が挙げられる．このエコシステムでは，ソフトウェア開発支援ロボット（ネットワークエージェント）が「群衆知の発生」を捉え，リポジトリマイニングにより「群衆知の形式知化」を図り，Q&A 情報として蓄積された「群衆知の利用」を支援する．群衆知の利用支援では，深層学習（Deep Learning）を用いて開発者の思考プロセスを模擬（プログラム解析情報と群衆知を融合し総合的に解決方法を提示）することにより，自動プログラミングや自動バグ修正というかたちで，ソフトウェア開発者の作業をサポートする．

今まで夢であった自動プログラミングや自動バグ修正は，伝統的なプログラム解析技術（コードから導出可能な演繹的情報）と OSS や Q&A サイトなどが保有するソフトウェア開発ビッグデータに裏付けされた群集知（群衆から湧き上がる帰納的情報）を結びつけることにより，現実味が増してきている．世界中の開発者が OSS プロジェクトに参画し，ロボットがリポジトリの変化情報を群衆知化し，Q&A サイトに投稿することにより，世界中の開発ノウハウを世界中の開発者が共有することができる．そし

て，それらの情報を参照して，さらにOSSの開発が進むという循環的なエコシステムが実現される．もはやソフトウェアの開発環境はPC内の統合開発環境（IDE：Integrated Development Environment）には閉じておらず，開放的な群衆開発環境の世界に移行しつつある．これは，ソフトウェア工学における大きなパラダイムシフトと言える．

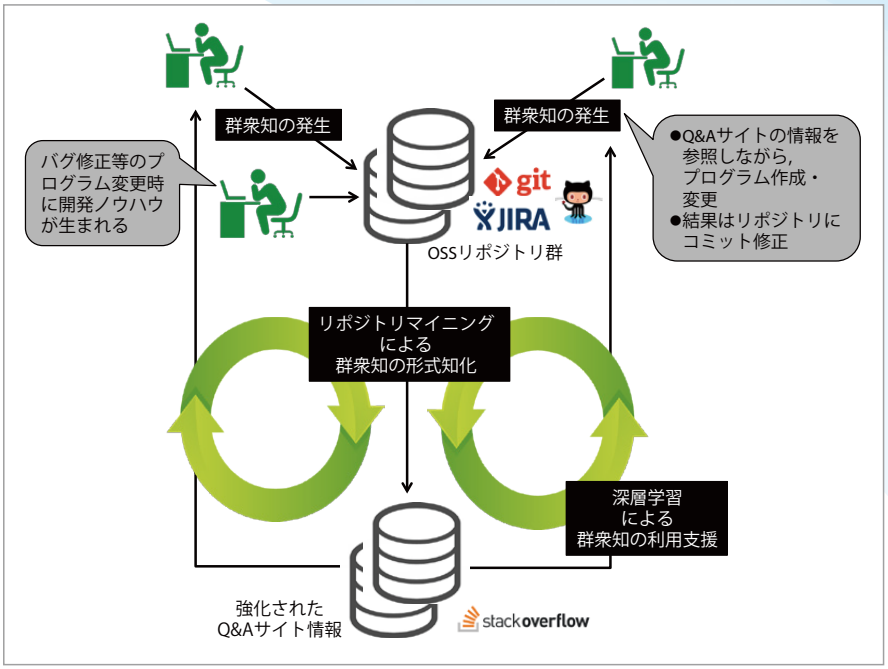

図-2　群衆知の循環による群衆ソフトウェア開発エコシステムの概要

究極の夢

自動プログラミングに関する究極の夢は，欲しい機能を伝えるだけで，大規模なプログラムが自動生成できることであろう．この段階に到達するにはいくつもの壁を越えなければならなく，現時点の技術水準では難しい．

たとえば，インターネット上にあるプログラムコードや実行ファイルを収集して，それらの中に含まれる部品を再利用することにより欲しいプログラムを自動合成するようなことが将来できるようになるかもしれない．そのような自動合成を意識したインタフェースやプロトコルが標準化されてくると現実味を帯びてくる技術であろう．

ソフトウェア工学は主にソフトウェアの開発者を支援するための学問分野である．しかし，ソフトウェアが自動で生成できるようになればもはや開発者は必要ではないかもしれない．ソフトウェアを使いたいユーザが，自分自身でソフトウェアを生成してそれを使う，そんな時代がいつか来るだろう．そのような時代にはソフトウェアという名はもはやその体を表しておらず，クイックウェアやインスタントウェアのような新しい名がついているかもしれない．

参考文献

1) Goues, C. L., Nguyen, T., Forrest, S. and Weimer, W. : GenProg : A Generic Method for Automatic Software Repair, IEEE Trans. Softw. Eng. 38, 1 (Jan. 2012).
2) Long, F. and Rinard, M. : Staged Program Repair with Condition Synthesis, In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015).
3) Ke, Y., Stolee, K. T., Goues, C. L. and Brun, Y. : Repairing Programs with Semantic Code Search. In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015).
4) Xuan, J., Martinez, M., Demarco, F., Clément, M., Lamelas, S., Durieux, T., Berre, D. L. and Monperrus, M. : Nopol : Automatic Repair of Conditional Statement Bugs in Java Programs, IEEE Trans. Softw. Eng. 43, 1 (Jan. 2016).
5) Murakami, M., Hotta, K., Higo, Y. and Kusumoto, S. : Predicting Next Changes at the Fine-Grained Level, In Proceedings of the 2014 21st Asia-Pacific Software Engineering Conference (APSEC 2014).
6) Chen, F. and Kim, S. : Crowd Debugging, In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015).

（2017年4月7日受付）

肥後芳樹（正会員）higo@ist.osaka-u.ac.jp

大阪大学大学院情報科学研究科准教授．本会長尾真記念特別賞や論文賞等を受賞．IPSJ-ONE2017にて講演．

鵜林尚靖（正会員）ubayashi@acm.org

九州大学大学院システム情報科学研究院教授．本会フェロー．2016年度までソフトウェア工学研究会主査．