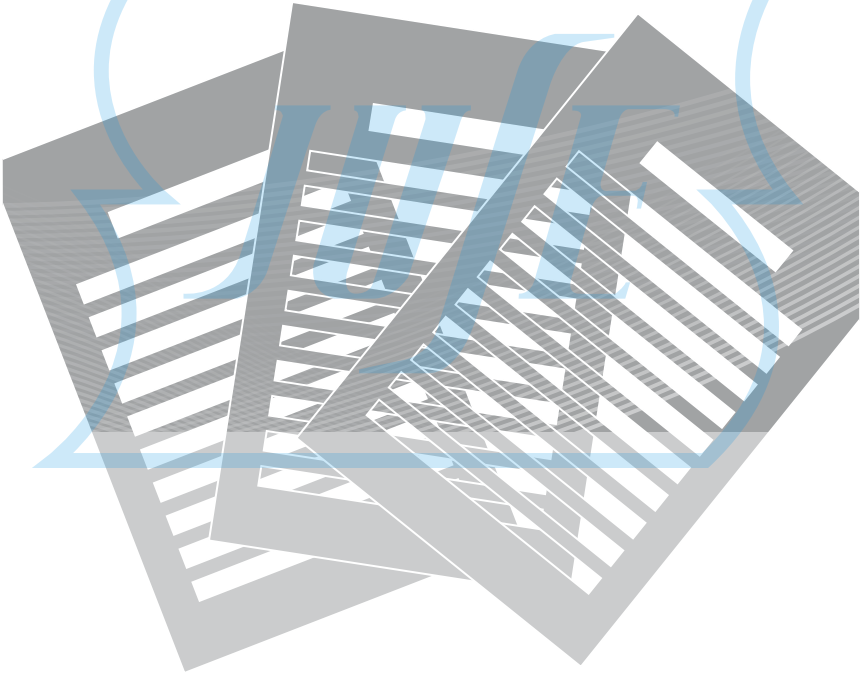
ソフトウェア技術者のための バグ検出テキスト

山浦恒央・田中智康 著



日科技連

はじめに

本書は、アイティメディア㈱の Web サイト、MONOist に連載した「山浦恒央の”くみこみ”な話」の内容を大幅に改稿したものである。本書は、開発技術者だけでなく、品質保証エンジニアも対象にしている。

近年、電気電子関連の製品におけるハードウェアとソフトウェアのコスト比は、ソフトウェアが上回り、大規模化、高価格化している。

例えば、一般に普及している自動車に搭載しているマイクロプロセッサの個数は 50 を超え、プログラムの行数は 10MLOC(1,000 万行)以上となっている。また、日本も採用している最先端のステルス戦闘機、F-35 は、ベーシックな機能を制御するソフトウェアだけでも、C++ で 18MLOC(1,800 万行)にものぼる。18MLOC のソースコードは、300 ページの文庫本 3,000 冊に相当する量で、3,000 冊のすべての書籍を誤字、脱字、誤記がないように作るのはきわめて難しい。

さらに、F-35 のソフトウェアは、段階的に機能を向上させるインクリメンタルな開発方式を採用しており、規模は増え続ける。超大規模のソフトウェアは、開発が困難だが、検証、品質保証は開発よりさらに難度が上がる。

近年、ソフトウェアのバグにより、人間の身体や財産に大きな損害を与えるケースが増えている。死亡事故に至った初期の事例として有名なのが、1985 年から 1987 年に発生したセラック 25 の医療事故であろう。フランスの CGR 社とカナダの AECL 社が共同開発した放射線医療機器、セラック 25 は、セラック 6 とセラック 20 の発展型で、両方のソフトウェアのバグも引き継いでしまった。セラック 20 の場合、本体とは独立した制御機器の安全装置により、電子線がターゲット外に照射する事故を防いでいたが、安全装置を含め、すべてを一体化してソフトウェアで制御するセラック 25 では、バグが表面化し、5

人が死亡したといわれている。

セラック 25 は、開発段階で安全性を検証したものの、バグを見つけられなかった。「ソフトウェア工学」という言葉が初めて公的な文献に登場したのが 1968 年であり、それから 20 年も経たないソフトウェア開発の黎明期とはいえ、痛ましい事故である。体系的にテストしていれば防げたであろう。現代のソフトウェアのバグは、当時に比べ、人体や財産に対し、はるかに大きな影響を与えることを忘れてはならず、プログラムの品質は、ソフトウェアを内蔵する製品の根幹を形成している。

ソフトウェア開発エンジニアの最大の悩みが、「プログラミングが完成し、デバッグに入ってプログラムを走らせたところ、仕様どおりに動作しない。いろいろ試したが、バグの原因が見つからない」である。

バグの原因は、わかってしまえば非常に単純だが、わからないとプログラムの不可解な動作に悩み、時間と労力を投入し、身を捩ってバグの規則性を見つけようとする。

「楽しいコーディングの時間」の後に待っている「苦悩のデバッグ」でプログラマは頭を悩ませる。筆者も、入社 1 年目に担当したソートのプログラムでいろいろなバグを作り込み、基本機能が疎通していることを確認するため「5 件のデータを正しく並び替える」というテストに 2 週間もかかった経験がある。

当時、使用したマシンは 16 ビットのレジスタが 16 個あるハードウェア構成で、プログラム全体を常駐させるほどメモリの大きさが十分ではなく、プログラマは、オーバーレイの手法を使って、自分で仮想記憶を実現していた。プログラミング言語はアセンブラであり、また、ソート方式のアルゴリズムが複雑だったことも、テストに 2 週間もかかった要因だったと思う。

バグが発生すると、仕様書、標準出力やログを確認し、規則性を探り、原因を見つける必要がある。簡単に見つからない場合、身体的に精神的に、非常に苦しい時間となる。これが数日、数週間も続くと、一生見つからないかもしれないと悲観的になり、精神的に追い詰められることも少なくない。

学生のプログラマであっても、ビジネスとしてソフトウェアを開発するプロでも、自分が困っていることは、他人も困っている。デバッグで動作不良を検

出した場合、種類別にバグの動作、傾向、未然防止策を網羅的にまとめた書籍があると高い効果を期待できる。また、品質保証エンジニアは、バグの全体を俯瞰できるため、漏れのないテストが可能となる。これが、本書を執筆した理由である。

前著『ソフトウェア技術者のためのバグ検出ドリル』(日科技連出版社)においては、実際にバグを含む要求仕様書、設計書、コーディング、デバッグ、保守を具体的に取り上げ、練習問題として31問を出題した。これは、囲碁や将棋における「次の一手」問題であり、ピンポイントの場面を取り上げ、実践力を鍛えることを目的とした。

この『バグ検出テキスト』では、「要求仕様のバグ」「テスト業務のバグ」など、バグの種類別に網羅的に取り上げ、詳しく解説している。将棋なら「角換わり」「四間飛車」「藤井システム」などの戦法、チェスなら「クイーンズ・ギャンビット」「キングス・ギャンビット」「ルイ・ロペス」などのオープニング、囲碁なら「死活」「しのぎ」「模様」「寄せ」などのテクニックのように、バグ全体を俯瞰して解説するものである。

本書では、代表的なバグを網羅的に36個取り上げた。この36個のバグは、ソフトウェア開発で体験する大量のバグを分類したもので、現実のソフトウェア開発で経験するバグの大部分はこの分類の中に入ると思われる。多数のバグを分類し、総合的に網羅したものであり、体系的にデバッグを実施したり、品質保証をする場合のベースになると考える。ソフトウェアの品質を確保するうえで、これが必要条件になる。このベーシックな必要条件に、各ソフトウェア固有のバグ、例えば、リアルタイム系での「プロセスの優先順位の設定誤りにより、ハードリアルタイム性を実装できない」などを補足すれば、漏れのない品質制御が可能となる。36個のバグは、いずれも、多くのプログラマや品質保証エンジニアが経験するバグである。それぞれのバグで、どのような不具合が外部現象として起きるか、どこに注目すれば当該バグが原因であるとわかるか、そのバグを未然に防ぐ方法などを記した。これにより、品質保証エンジニアは、網羅的なバグの要因をもとに、効果的なテストを設計できる。

各バグでは、筆者が経験したバグ(作り込んだバグ)を具体的な例としてあ

げ、不可解な現象が発生するメカニズムの詳細をした。また、「うるう年バグは、2月29日、12月31日に発生することが多い」など、各バグの傾向を取り上げた。これにより、バグを短時間で特定すると同時に、テスト項目に追加することでデバッグを効率化できる。

「次の一手」問題であるピンポイント的な『ソフトウェア技術者のためのバグ検出ドリル』と、戦法全体の解説書である本書を併用し、効果的にデバッグすると同時に、デバッグ能力を高め、バグを未然に防ぐ高い能力をもった開発エンジニアになり、また、バグ摘出能力の高い品質保証エンジニアになっていただければ幸いである。

最後に本書の執筆にあたり、ご協力や貴重なコメントをいただいた秋山大樹氏(システムコンサルタント)、玉城良氏、浜光彦氏(有限会社スペーステクノロジー)、福山祐哉氏、山崎知康氏(有限会社スペーステクノロジー)に感謝いたします(五十音順)。

2020年11月

山浦 恒央

ソフトウェア技術者のための バグ検出テキスト

目　次

装丁・本文デザイン＝さおとめの事務所

2.10 【バグ10】
なぜか動かない：作業手順ミスによるバグ

2.10.1　意図したとおりに動かない

プログラムは作ったようにしか動作しないが、正しくプログラムを記述しても意図した結果とならないことがある。大半は、記述ミスだが、別の要因もある。

よくあるのが、「エディタの保存忘れ」や、「コンパイルせずに実行していた」などがある。非常に初歩的なバグだが、目視では正しいはずなのに事象が特定できずに慌てることになる。

2.10.2　筆者の恥ずかしい経験

筆者の恥ずかしい経験を以下に示す。

(1)　正しいプログラムのはずなのに

自宅でプログラムを作成した際、ちょっとしたプログラムの記述ミスをした。何度も読み直し、修正し、実行したが意図した結果とならない(図 2.10.1)。

図 2.10.1 は、当時の作業イメージを示す。左のエディタを見ると、3 行目のメッセージが「halo world」と間違っていることがわかる。メッセージの内容を「hello world」に変更し、右側の状態でコンパイル・実行をしたが、

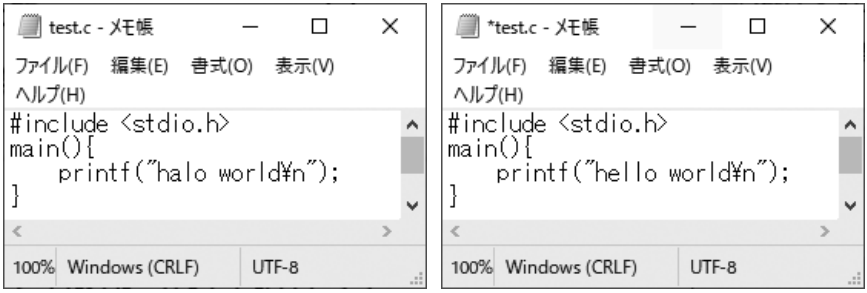


図 2.10.1　当時のイメージ(左：修正前、右：修正後)

それでも「halo world」のままだった。

「パソコンが壊れたのでは」とも思ったが、数分考えたのち、テキストエディタで修正した改訂版を保存し忘れており、修正前の状態で動作していることがわかった。修正後のファイル名には、「*test.c」となっていることがわかる。

(2) プログラムのバージョン間違い

バグがあったプログラムを修正した段階で、その日の作業を終えた。週明けに出勤し、作成したプログラムを開いたが、修正前と同様の問題が発生した(図 2.10.2)。

図 2.10.2 には当時のディレクトリ構造のイメージを示した。前に解決したはずのバグが発生したのは、work ディレクトリに作成していた同名のプログラムを実行していたためである。正しいモジュールは、project フォルダにあるプログラムだった。

上記は、重大なバグではないが、正しく動くはずのものが動かない事象が発生し、混乱する。

2.10.3 よくある意図したとおりに動かない例

プログラムを修正したつもりでも、何らかの原因で意図した結果とならない場合がある。よくある間違いを以下に示す。

(1) エディタでの修正を保存し忘れる

エディタで修正したソースコードを保存し忘れると、いくらプログラムを修

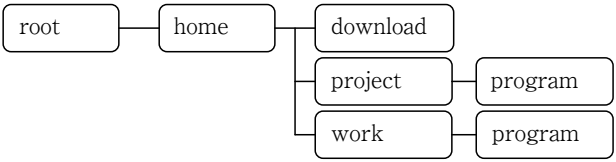


図 2.10.2 当時のディレクトリ構造イメージ

正しても正しく動作しない。そのため、例えば、「コンパイルは通るけれど前の状態のまま」「修正したのにコンパイルエラーが発生する」となる。

(2)　ビルド忘れ

エディタ同様にコンパイルを忘れて実行する場合も意図した動作とならず、前に使用した実行可能形式のファイルと同じ結果となる。

(3)　焼き忘れ・置き忘れ

マイコンにプログラムを入れ込む際、ターゲットの ROM にプログラムを入れる必要がある。入れ忘れると、過去の状態で実行することになる。

また、Web 系開発では、サーバの適切な位置にコードを置く必要がある。環境によっては、サーバにデータを置き忘れることもある。

(4)　プログラムのバージョン間違い

意外と多いのが、間違えたバージョンを使用して実行することである。例えば、自分の環境で実行を確かめ、テスト担当者に渡すファイルを間違えるなどである。バージョンが異なると、意図した動作とならないし、場合によっては再テストとなるため注意が必要である。

2.10.4　バグの発生傾向

(1)　エディタを保存し忘れる・ビルド忘れ・焼き忘れ

修正内容に強い自信がある場合は、そもそものエディタの保存忘れやビルド忘れを疑おう。軽微なミスだが、盲点に入ると気がつかない。とにかく冷静になろう。

(2)　バージョンを確認しよう

古いバージョンで確認していないかもう一度確認しよう。過去のバージョンのまま実行し、変更差分が反映されていない可能性がある。

(3) バグが自然治癒する

プログラムを見直している最中に、エディタを保存するなどして正しい状態になる。この場合、バグが自然治癒したように見える。

2.10.5 対策

(1) エディタでの修正版の保存し忘れ・ビルド忘れ・焼き忘れ

エディタでの修正版の保存忘れやビルド忘れを防ぐため、統合開発環境かスクリプトで自動化するとよい。例えば、Visual Studio のエディタ上から実行する場合は、1 回、エディタを保存する仕組みになっている。エディタの機能で自動保存から実行までを可能な限り一括でできるように工夫しよう。

(2) バージョンを確認しよう

誰しも整理整頓が得意でないように、日々のテンポラリファイルを整える習慣を付けよう。

2.11 【バグ11】
バグを再現できない：再現条件がわかればバグの 90% は解決

2.11.1 バグの再現性

バグが発生して悩むことは、バグの再現性である。再現性とは、同じ入力をすると同じ出力結果が現れることで、規則性がわかるとバグの原因特定に非常に役立つ。バグ修正の第一歩は、この再現性を探すことにある。

エンジニアは、ログなどから原因を探る。例えば、入力値、タイミングが関係していそうだなどである。再現性がわからないと、エンジニアには苦しい時間となる。机上テストやデバッガを使用し、再現しないか確認していく。それでも見つからない場合は、不良事象の発生回数や調査工数を顧客に報告し、影響がないことを示さねばならない。

再現条件がわかると、実行パスがわかり、事象を正確に把握できるようになる。再現条件がわかれば、バグの 90% は解決する。ただし、デバッグやテス

著者紹介

山浦恒央(やまうら　つねお)

1977 年、日立ソフトウェアエンジニアリング株式会社に入社、1984 年から 1986 年、カリフォルニア大学バークレイ校客員研究員。2006 年より、東海大学情報理工学部ソフトウェア開発工学科准教授、同大学院組込み技術研究科准教授。現在、同大学非常勤講師。

大阪大学基礎工学研究科情報数理系博士課程単位取得退学。博士(工学)。

主な著書に『ソフトウェア技術者のためのバグ検出ドリル』(共著、日科技連出版社、2019 年)、『ビューティフルテスティング』(共著、オライリー・ジャパン、2010 年)がある。

また、翻訳書には以下のようなものがある。『デスマーチ』(エドワード・ヨードン著、2006 年)、『初めて学ぶソフトウェアメトリクス』(ローレンス・パトナム著、2005 年)、『ソフトウェア開発 55 の真実と 5 つのウソ』(ロバート・グラス著、2004 年) いずれも日経 BP 社、単訳。『ピープルウェア』(トム・デマルコ著、2001 年)、『ソフトウェアテスト技法』(ボーリス・バイザー著、1994 年)、いずれも日経 BP 社、共訳。

ウェブサイト「IT MONOist」と「TechFactory」でソフトウェア工学のコラムを連載中。

田中智康(たなか　ともやす)

2016 年、有限会社スペーステクノロジーに入社。情報通信学修士。主な著書に『ソフトウェア技術者のためのバグ検出ドリル』(共著、ペンネーム＝大森祐仁、日科技連出版社、2019 年)がある。

ソフトウェア技術者のための
バグ検出テキスト

2020年12月27日　第1刷発行

著　者　山　浦　恒　央
　　田　中　智　康

発行人　戸　羽　節　文

検　印
省　略

発行所　株式会社 日科技連出版社
〒151－0051　東京都渋谷区千駄ヶ谷5-15-5
DSビル
電　話　出版 03-5379-1244
　　営業 03-5379-1238

Printed in Japan　　印刷・製本　株式会社三秀舎

ISBN 978-4-8171-9725-2
URL https://www.juse-p.co.jp/