第 5 章　ピタゴラ装置をデバッグしてみよう

バグはこう洗い出す（デバッグの基本）

森 孝夫

ここでは，ソフトウェアの挙動がおかしい場合にいかに原因究明を進めていけばよいのかを解説する．基本は，まず現象を正しく把握し，考えられる原因の中から最も状況に合致しているものを特定していくことである．このような原因究明の具体例として，ピタゴラ装置とソート・プログラムのデバッグを比較しながら見ていく．（編集部）

1．ピタゴラ装置がバグった！

ピタゴラ装置というのをご存じでしょうか．この装置は，ブロックや木材を組み合わせて，連続して動作するように仕掛けてある「からくり装置」です（**図1**）．ドミノ倒し風に，次々と装置が倒れて最後にスイッチが押されるようなものもあれば，流しそうめんで使うような竹筒やシーソーやブロックの上を，玉が転がっていくものもあります．

すべてのピタゴラ装置に共通する特徴は，一度起動すると，その後は全自動で流れるように最初から最後まで動くことです．なかなか見事な仕掛けです．中には，最後に卵が割れて目玉焼きを焼き始めたり，トースターからパンが飛び出す装置もあるようです．

さて，われわれは卵焼きができるピタゴラ装置を開発しました．これを家庭に持っていくときには，竹筒やシーソーが見えていなくてもよいわけですから，そこは箱に包んでおくことにしました．

そうしてできたピタゴラ装置を自宅に配置しました．これで，玉さえ配置すれば，自動的に卵焼きが焼けるようになります．毎朝ルンルンです．

ところが…．ある日の朝のことです．

コロコロコロコロ……ゴトッ!!

なんだか箱の中で変な音がしています．途中でコースを外れた玉がこぼれてきています．ああっ，装置の部品まで… Ooops!

こんなとき，あなたはどうしますか？

● デバッグの流れ

ピタゴラ装置が動かなくなったら，多分ほとんどの人は，箱を開けて中身を見ながら玉を転がしてみることでしょう．そして，玉がどこでこぼれ落ちたかを確認するのではないでしょうか．ピタゴラ装置の場合，「中がどう動けば正しいのか？」がはっきりイメージできるので，1回動かしてみれば「あ，ここがおかしい」と分かることでしょう．

おかしい個所が特定できたら，「う〜ん，何でここで玉がこぼれてしまうんだろう」と考えて，修正案を設計します．修正案を設計できたら，装置を修理します．修理できたらもう一度玉を転がしてみて，うまくいけばデバッグ完了です．

この流れは，プログラムの場合も同じですね．プログラムを動かしてみておかしい場合は，デバッガで中の動作を確認したり，場合によってはソース・コードをウォーク・

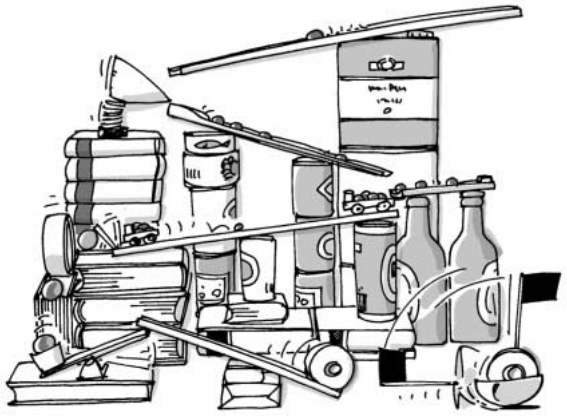

図1　ピタゴラ装置

KEYWORD ── デバッグ，欠陥，故障，構造，機能，振る舞い，要求，ソート・プログラム

図2　内部を確認しながらおかしい場所を特定する

スルーして動作をイメージしたりして，おかしい場所を特定します(**図2**)．特定できたら修正案を設計します．修正案を設計できたら，再プログラミングします．そしてテストしてみて，うまくいけばデバッグ完了です．

すなわちデバッグとは，「バグ発見」→「バグ原因究明」→「修正案の設計」→「修正」→「テスト」という流れで進みます．

● 原因と結果

ところで，「バグ原因究明」という言い方は，ソフトウェア開発を行っているどこの現場でも耳にする言葉ですが，そもそもバグの直接の原因は何でしょうか？ 日ごろの行いや運ではないことは確かです(因果関係がないとは言わないが，直接の原因というわけではないだろう)．

ピタゴラ装置の場合，装置と装置の間にすき間があり過ぎると，玉はこぼれてしまいます．シーソー形式の機械を使っている場合は，シーソーの軸の滑りが悪いと，玉はそこで止まってしまいます．このように，原因としてはいろいろあり得るのですが，すべてに共通するのは「構造上の問題」であるということです．すなわち，構造上の問題がバグを引き起こすのです．

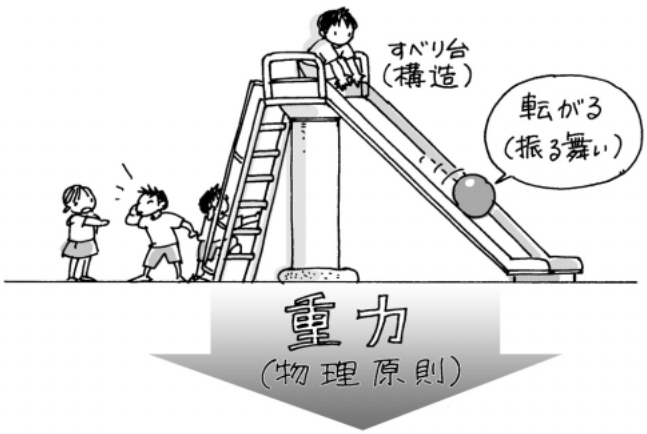

図3　振る舞い＝構造＋物理原則

表1　不具合の原因究明における原因と結果の関係

原　因	構造	欠陥	プログラム
結　果	機能	故障	内部の動作/外部の動作

構造上の問題のような，不具合につながる内部の問題のことを，「欠陥(Fault)」と呼びます．欠陥が原因となってもたらされる，意図とは異なる動作や影響のことを「故障(Failure)」と呼びます．通常，バグとは「故障」のことを示すようです．そして，バグの直接の原因は，「欠陥」なのです．

これは，実はプログラムにも同じことが言えます．プログラムは，皆さんの設計意思や「思い」の通りに動くわけではありません．あくまで，メモリ上に配置されたプログラムに書いてある通りに動作するだけです．プログラムに「欠陥(Fault)」があれば，設計者の意図とは異なる動作を行い，結果として「故障(Failure)」を引き起こします．

機械の故障原因究明にせよ，プログラムのデバッグにせよ，**表1**のような「原因と結果の関係」に着目することは非常に重要です．

● 「振る舞い」の概念をデバッグに利用する

ピタゴラ装置のように，動きを伴う装置の場合には，「振る舞い」という概念を導入すると，「構造」と「機能」の因果関係がより理解しやすくなります．

ピタゴラ装置は，玉を設置してスタートすると，時間の経過に伴って，玉の位置が変わっていきます．装置の形状も変わっていきます．つまり，「時間の経過に伴って，装置や装置を流れる物の状態が変わる」ということです．時間経過に伴う装置や物の状態変化のことを「振る舞い」と呼びます．

振る舞いは，構造が原因となって，物理原則に従って発現します．例えばすべり台の上に玉を置くと転がっていきますが，これはすべり台の構造をした機械が，万有引力が存在する地球上に配置されているからこそ，そのように振る舞うのです(**図3**)．ピタゴラ装置も，構造を組んでおくと，それが物理原則に従って振る舞いを示します．

実はプログラムについても同じことが言えます．メモリ上にプログラムを配置し，コンピュータの電源を入れると，