

DRY原則

作者：スティーブ・スミス (Steve Smith)

訳者：夏目大

Kevlin Henney(編)、和田卓人(監修)『プログラマが知るべき97のこと』(オライリー・ジャパン、2010年)を出典とする。各エッセイはCC-by-3.0-USによってライセンスされている。

DRY(Don't Repeat Your Self：繰り返しを避けること)原則は、プログラミングに関して守るべきとされている原則の中でも特に重要なものと言っていいでしょう。これは、Andy HuntとDave Thomasが、著書「達人プログラマ」の中で提唱した原則です。よく知られたソフトウェア開発のベストプラクティスやデザインパターンの中にも、基本的には考え方がこの原則と同じものがたくさんあります。開発者は、アプリケーションの中に何らかの「重複」があれば、また、重複が起きそうであれば、それを察知する必要があります。そして、適切なプラクティスや抽象化によってそれを排除する必要があるのです。そのための方法を学べば、よりきれいなコードを書けるようになるはずです。

重複は無駄である

アプリケーションを構成するコードはすべて、保守を必要とします。どのコードも将来バグになる危険性を秘めています。重複があると、コードベースは不必要に大きくなり、それにつれて、バグが生じる危険性も高まります。また、システムの構造は、そう意図していないにもかかわらず複雑になってしまいます。重複によりコードベースが大きくなれば、開発に携わる人間がシステム全体を完全に理解することも難しくなります。特に困るのはコードに変更を加える時です。どこかに変更を加えた場合、それとロジック等が重複している箇所にも同様の変更が必要かどうか確認しなければなりません。DRY原則を守れば、そういう事態に陥らずに済みます。DRY原則を守るとは、言い換えれば「すべての知識はシステム内において、单一、かつ明確な、そして信頼できる表現になっていなければならない」という条件を満たすことです。

作業の重複は自動で防ぐ

ソフトウェア開発に関わる作業の多くは「同じことの繰り返し」です。つまり作業が何度も重複します。この重複は、自動化によって容易に解消できます。DRY原則は、アプリケーションのソースコードだけでなく、開発作業にも適用すべき原則ですが、そのためには自動化が役立つというわけです。たとえばテストは同じ作業の繰り返しになることが多いですが、手作業でそれをやっていては手間も時間もかかり、しかも誤りも起きやすくなります。したがって、可能な限り、テストは自動化すべきなのです。同様に、ソフトウェアの結合(ビルド)も同じ作業の繰り返しになることが多いですが、手作業だと時間がかかり、誤りが起きやすくなります。従ってビルドプロセスも自動化し、しかも頻繁に走らせるべきです。コミットの度に走らせるというのが理想でしょう。手作業だと負担が大きすぎるようなものは、すべて自動化の対象となります。自動化し、かつ標準化するのが望ましいのです。大事なのは、作業を遂行するための方法を1つだけにするということです。そうすれば、手間が省ける上、問題も起きにくくと言えます。

ロジックの重複は抽象化で防ぐ

ロジックの重複には多数の種類があります。たとえば、if-thenやswitchの部分が単純にコピー&ペーストされている場合は、発見するのも解消するのも非常に容易でしょう。デザインパターンの多くは、明らかにアプリケーションの中の重複を減らす、あるいは排除することを目的としています。あるオブジェクトを使用するために整えるべき条件がほぼいつも同じであれば、その場合は Abstract FactoryパターンやFactory Methodパターンを使用すればいいでしょう。オブジェクトのふるまいに様々な種類があり得る、という場合には、長いif-thenを書くよりは、Strategyパターンを使用します。実際、デザインパターンは、同じような問題について何度も繰り返し解決策を考えるという重複が起きないように作られたとも言えます。DRY原則は、データベーススキーマなどの構造にも適用されます。これは、いわゆる「正規化」につながります。

関連する原則

ソフトウェア開発に関する原則には、他にもDRY原則に関連するものがいくつかあります。 OAOO(Once and Only Once:1度、ただ1度)原則はその1つです。これは、コードの機能、ふるまいについてのみ適用される原則で、DRY原則を特殊化したものと考えることもできます。 OCP(Open/Closed Principle : 開放/閉鎖原則)というのもあります。これは、クラスなどのプログラム単位は、拡張に対して「開いて(open)」いなくてはならず、反対に、修正に対しては「閉じて(close)」いなくてはならない、という原則です。この原則は、DRY原則が守られている場合にのみ有効です。他には、SRP(Single Responsibility Principle : 単一責任原則)という有名な還俗もあります。これは、「クラスに変更を加える理由は2つ以上存在してはならない(1つの変更の理由は常に1つでなければならない)」という原則で、やはりDRY原則が守られている場合のみに有効です。

DRY原則は、構造、ロジック、プロセス、機能などあらゆる面で、開発者がシンプルなアプリケーションを作る上での基本的な指針です。この原則を守ることで、シンプルで、品質が高く、保守もしやすいアプリケーションが作りやすくなるのです。状況によっては、パフォーマンスを上げるために、あるいは何らかの要件(データベースを非正規化しなくてはならないなど)を満たすために、重複がどうしても必要ということはあります。ただし、現実にすでに目の前にある問題に対処する以外の目的で、DRY原則を破ってはなりません。「こういう問題が起きそうだから原則をあらかじめ破っておこう」ということをしてはいけないのでした。

出典：プログラマが知るべき97のこと/DRY原則 - Wikisource - <https://goo.gl/eRdWzn>