第 37 章　クリーンルーム開発

「クリーンルーム開発」とは何か

半導体を作るごく初期の作業に、シリコンなどの基板の上に半導体の回路を焼き付けるという作業がある。半導体の集積度が上がって回路を構成するトランジスタや抵抗／コンデンサ、配線などの要素がたいへん微細になり、回路を焼き付けるときにほこりが基板に付着していたりすると、それが不良品を作り出す原因になる。「クリーンルーム」とは、この不良品ができるのを避けるために部屋の空気をたいへんきれいに保ってある部屋をいう。

「クリーンルーム開発」とはこの発想をソフトウェア開発に適用し、開発中に誤りが入り込まないようにして高品質のソフトウェアを開発しようとする手法である[1][Hutatugi97]。

クリーンルーム開発の源流は、1970 年代のダイカストラ（Edger W. Dijkstra）などが提唱した構造化プログラミングや段階的詳細設計法などの構造化技法にある[2]。これらを元に IBM のフェローであったハーラン・ミルス（Harlan Mills）等によって、1987 年[3]に「クリーン・ルーム・ソフトウェア・エンジニアリング」としてまとめられた[Mills87]。

クリーンルーム開発を構成するもの

クリーンルーム開発は、次の 4 つの作業によって特徴付けられる。

- インクリメンタル開発プロセス
- ボックス構造化分析
- 関数等価性検査
- 統計的品質保証

以下で、これらについて個々に見てゆきたい[Hutatugi97]。

インクリメンタル開発プロセス

クリーンルーム開発を特徴付けるものの 1 つに、インクリメンタル開発プロセスがある。

当時主流だった開発手順は、ウォータフォール型だった[4]。しかしウォータフォール型の場合、要件の確定からそれが結果としてコンピュータ上で実現できるまで、長い時間を必要とする。さらに要件定義や設計などの初期工程の誤りが、ずっと後にならなければ誤りとして発見できない。これらを避けるためにクリーンルーム開発では、インクリメンタル型の開発手順を使用する。

ミルス等はこの開発手順を「インクリメンタル型」と記しているが、これは我々の今の感覚では「スパイラル型」に近い[Mills87]。

ボックス構造化分析

---

1 「開発が終わった後で欠陥を見つけてそれらを取りのぞくのではなく、最初から欠陥をソフトウェアに組み込まないようにする」ことが、クリーンルーム開発の考え方である。

2 構造化技法については、第 15 章で述べた。

3 今となっては、クリーンルーム開発の特徴のいくつかは我々の常識ともいえるものになっている。しかしこのクリーンルーム開発が提唱されたのは 1987 年だったということを、改めて留意しておきたい。

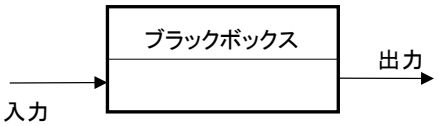
4 ウォータフォール型やインクリメンタル型、スパイラル型などの開発手順については、第 13 章で述べた。

クリーンルーム開発では、情報システムの構築は以下の手順で行う[Hutatugi97]。

- 問題定義：通常の開発でも最初のフェーズとして行うステークホールダの要求を調査し、自然言語（日本語、英語など）でそれを記述する。通常の開発での要求仕様書の作成に相当する[5]。
- 要件定義：問題定義をベースに、ステークホールダの要求を明確にする。記述は原則として、VDM や Z などを使用して行う。通常の開発での要件定義書の作成に相当する[6]。
- 分析と設計：以下で述べる 3 種類のボックスを使用して、前記要件をいかにしてソフトウェアとして実現するかを明らかにする。作業の位置づけとしてはプログラムの設計に相当するが、作業の内容と設計の結果は通常の設計とは大きく異なる[7]。
- 実装：詳細化されたシステムの設計結果を、運用可能なシステムとして実装する。プログラムの作成に相当する[8]。

この中で、たいへん特徴があるのが 3 種類のボックスを使用したシステム分析と設計である。これについて、以下で記述する。

ボックスを使用したシステム分析と設計



図表 37-1 ブラックボックス

クリーンルーム開発では、以下（図表 37-1 から図表 37-3）で図示する 3 つの種類のボックスを使用してソフトウェアを設計する。

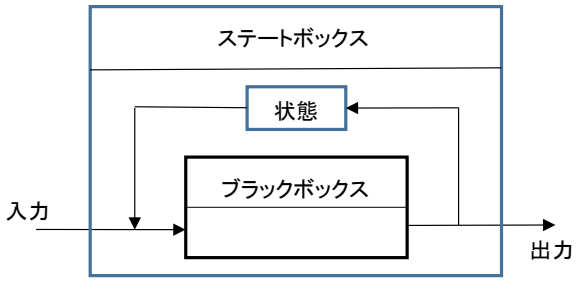
ブラックボックス（図表 37-1）は単に入力を出力に変換するだけで、その中に値や状態を保持することはできない。ブラックボックス内の処理は、構造化プログラミングの 3 種類の制御構造（順次、選択、繰り返し／図表 37-4 参照）とその組み合わせを取ることができる[9]。また、入力や出力が複数になることは差し支えない。銀行業務では、金額から利息を計算する過程などでブラックボックスが使用される。

2 つ目のステートボックス（図表 37-2）は、内部に 1 つのブラックボックスと 1 つの状態などを記憶する機能を持つ。1 つの例として銀行の預金口座を考えると、「状態」に保持されるのが残高であるとすれば、入力が「出金」であればブラックボックスへの入力は出金指示と出金前の残高になり、ブラックボックスからの出力は出金が可能かどうかの判断と、出金できなかった場合も含めて処理後の新しい残高になる。

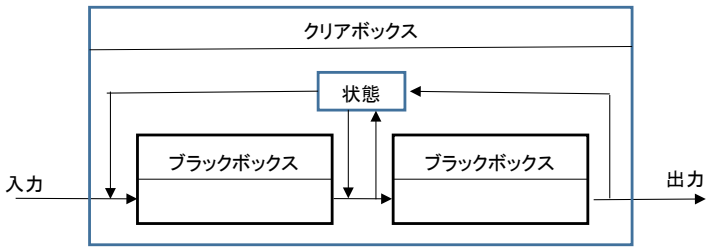
3 つ目のクリアボックス（図表 37-3）は、内部に 2 つのブラックボックスと 1 つの状態などの記憶機能を持つ。クリアボックスを再度銀行預金口座の例で示すと、最初のブラックボックスで出金

[5] 要求仕様書の作成については、第 20 章で記した。
[6] 要件定義書の作成については、第 21 章で記した。
[7] プログラムの設計については、第 26 章で記した。
[8] プログラムの作成については、第 27 章で記した。
[9] 構造化プログラミングの 3 種類の制御構造については、第 15 章で述べた。

が可能かどうかを判断し、可能であれば 2 つ目のブラックボックスで新しい残高を計算することになる。



図表 37-2　ステートボックス



図表 37-3　クリアボックス

クリーンルーム開発では設計段階で、全てコンピュータによって処理する内容をこの 3 種類のボックスの組み合わせで表す。そして実装では、実際のプログラム言語を使用してこれを記述することになる。

数等価性検査

クリーンルーム開発では、プログラムの検査（正しいことの検証）はレビューを通してのみ行い、プログラムの正しさを証明するためのテストは行われない[10]。

クリーンルーム開発では、プログラムは構造化定理と呼ばれている構造化プログラムの 3 つの制御構造（順次、選択、繰り返し／（図表 37-4)[11]）のみで表現されると考える。そしてこのそれぞれの制御構造ごとに、レビューするべき項目が以下のように明確にされている[12][Hutatugi97]。

「順次」では、「それぞれの処理をこの順次で実行することで、目的とする機能を達成できるか」がレビューするべき内容である。

同様に「選択」では、以下のようになる。「if 分の中の記述が真の時、then 以下の文を実行することで常に目的とする機能を達成できるか。同時に if 分の中の記述が偽の時、else 以下の文を実行することで常に目的とする機能を達成できるか[13]。」

---

[10] テストは、目標とする出荷可能レベルに達したことを確認するために行われる。

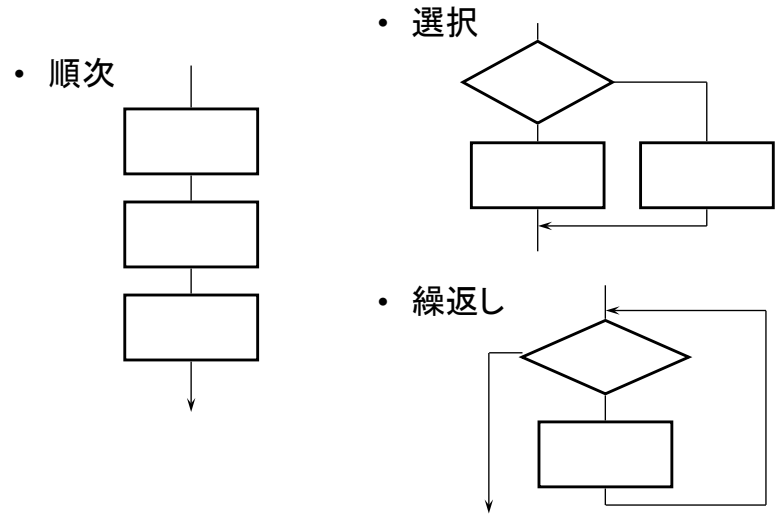
[11] 図表 37-4 は、図表 15-1 として示したものと同じである。

[12] 現実のプログラムはこの制御構造の組み合わせで実現されており、実際のレビューはここに書いたような単純なものではなく、現実のプログラムの記述に基づいたこれらの繰り返しになる。

[13] else 以下がない場合、case 文の場合では、レビューするべき内容が異なる。

”while B do S”の形の「繰り返し」では、以下のようになる[14]。

- 停止するか。
- B が真である間は、常に S を実行することで目的とする機能を達成できるか
- B が偽である時には、常に S を行わないことで目的とする機能を達成できるか



図表 37-4　3 つの制御構造

統計的品質保証

クリーンルーム開発ではソフトウェアの完成度を、現実の運用環境に準じた状態での MTBF（Mean Time Between Failure）[15]で表す。そのため開発の初期に記述する要件定義には、通常の開発と同様ソフトウェアが具備するべき機能と非機能の要件に加えて、そのソフトウェアが運用される時の運用要件の 2 種類が必要になる。

この運用要件に基づいてテスト計画が立てられ、テストが実施される。前述のようにテストは障害を発見してそれを修正するのが目的ではなく、そのソフトウェアの運用で実現できる MTBF を把握するのが目的である。つまりソフトウェアが出荷可能であるかどうかの判断は、当初目標とする MTBF を定めておき、その数値が実現できたかどうかで判定することになる。

現実のクリーンルーム開発の内容

クリーンルーム開発を特徴付ける 4 つの作業について、最初にその作業名をあげ、その後でそれぞれについて簡単に説明してきた。しかし現実は、「この 4 つ全部がそろっていないとクリーンルーム開発と呼ばない」という訳ではない。

クリーンルーム開発の考え方は、冒頭で述べた。この考え方に基づいた開発方法をとり、そのなかでこの 4 つの作業のいくつかを取り入れれば、それはクリーンルーム開発と呼んでもかまわない。

クリーンルーム開発を紹介した書籍では、現実的なアプローチの方法として以下のような段階的な方法を紹介している[Hutatugi97]。

---

[14] repeat until の場合などでは、「停止するか」の部分以外のレビュー内容が異なる。
[15] ある障害が起きてから、次の障害が起きるまでの平均時間。

1. 構造化プログラミングの徹底
2. 個人レベルでの検証（モジュールテストの廃止）
3. インクリメンタル開発の採用
4. チーム制開発と全体レビュー（統合テストの廃止）
5. 形式的仕様記述の実施
6. 仕様・実現の証明レビュー（システムテストの廃止）
7. 品質保証プロセスとの連携
8. MTTF 管理
9. 全体の開発プロセス制御

キィワード

クリーンルーム開発、インクリメンタル開発プロセス、ボックス構造化分析、関数等価性検査、統計的品質保証、ブラックボックス、構造化定理、ステートボックス、クリアボックス、MTBF、運用要件

略語

MTBF：Mean Time Between Failure

人名

ハーラン・ミルス（Harlan Mills）

参考文献とリンク先

[Hutatugi97] 二木厚吉監修、佐藤武久、大槻繁、金藤栄孝著、「ソフトウェアクリーンルーム手法 高品質ソフトウェア開発パラダイム４」、日科技連、1997 年.

[Mills87] Harlan D. Mills, Michael Dyer, Richard C. Linger, “Cleanroom Software Engineering“, IEEE Software, Sep. 1987, pp19-25.

（2016 年（平成 28 年））8 月 15 日　新規作成）