Ⅲ　ソフトウェアライフサイクル

到達目標

ウォータフォールモデル、プロトタイプモデルおよびスパイラルモデルなど、システムのライフサイクルモデルについて、それぞれの開発手順、適用および長所・短所を理解させる。

Ⅱで述べた基本計画、外部設計、内部設計、プログラム設計、プログラミング、テスト、運用・保守の一連の流れをシステムのライフサイクル、またはソフトウェアライフサイクルという。

ライフサイクルとは、情報システムの一生を段階的にモデル化したものであり、システムが開発され、運用された後、使用されなくなるまでを示したものである。どのようなシステムを開発したとしても、産業社会やユーザの要求は時代とともに様々な形で変化していく。したがって、常時その全てを満足させることは不可能であり、結果として費用対効果が時間とともに減少していき、最終的にシステムは使用されなくなる。そこで、再びシステム開発の最初の工程である基本計画から、システムを構築し直す必要性が生じることとなる。すなわち、費用対効果を考えながらシステムを運用することが、システムのライフサイクルを決定することとなる。

先に述べたように、システムのライフサイクルは、基本計画から始まり、所定の工程を経て、運用・保守となるが、運用中の経験がその上流の工程にフィードバックされ、修正や改良（ソフトウェアでいうバージョンアップ）が施されることにより、よりよいシステムへと成長していく。そして、それが時代のニーズに合わなくなったときや時代の流れに追いつけなくなったとき、システムは死滅するこことなる。すなわち、そこでライフサイクルは終了する。しかしながら、完全にシステムが死滅してしまうというのは過言であり、開発段階や運用段階で得た知識・経験が、次の新しいシステム開発に生かされることとなる。システム開発においては、開発に携わっている人間だけでなく、システム自体にも、いわゆる伝承や継承の概念が存在しているのである。

なお、システムのライフサイクルは一種類でなく、以下に示す数種の代表的なものがある。

・　コードアンドフィックスモデル

文字通り、まずコーディングしてみてから、ユーザの要求を確認し、また、手直しのためのコーディングを行うという手法である。コードアンドフィックスモデルでは、ユーザの要求が明確にならないまま開発が進められ、また、厳密な工程分けが行われていないため、システムの規模が大きくなればなるほど開発に膨大な手間と時間を必要とし、手直しが多い場合は、基本的なシステムの構造自体を見直さなければならないという問題も多く生じた。

厳密にいうと、このモデルには、現在のライフサイクルの概念は存在していない。

・　ステップワイズモデル

コードアンドフィックスモデルの欠点を補うために、現在のライフサイクルの概念、すなわち設計、開発、運用などの工程分けを行い、各工程ごとに管理するという方法を取り入れた手法である。しかしながら、ステップワイズモデルでは、ある工程の見直しの結果をその上の工程にフィードバックすることが認められていなかったので、上流工程の小さなミスが、下流工程および開発されたシステムに大きな悪影響を与えることとなった。

・　ウォータフォールモデル

ステップワイズモデルの改良型がウォータフォールモデルであり、このモデルが現在のシステム開発手法の基礎となっている。ウォータフォールモデルでは、ある工程の見直しの結果を、その上の工程にフィードバックすることが認められている。

・　プロトタイプモデル

ウォータフォールモデルでは、システム開発の後半の工程になってからでないと、そのシステムの全容は分からない。そこで、開発の早い段階から試作品（プロトタイプ：prototype）を作成してユーザに試用してもらい、その結果から要求を確定させるという方法である。試作品を作成するので、当然のことながら、ユーザの要求確認はウォータフォールモデルより優れているが、ユーザが納得する試作品ができない限り、実際のシステム開発が行えないという欠点がある。

・　スパイラルモデル

ウォータフォールモデルとプロトタイプモデルの両方の手法を取り入れたのがスパイラルモデルである。開発するシステムが独立性の高い部分・部品に分解可能なときに、このモデルが使用できる。

以上のモデルは、開発対象となるシステムの規模や業務の種類、処理の形態などにより選択されることになる。なお、その他にも様々なシステムのライフサイクルが存在するが、本章では、ライフサイクルの基礎となるウォータフォールモデルおよび現在よく用いられるようになってきたプロトタイプモデルとスパイラルモデルを中心に説明する。

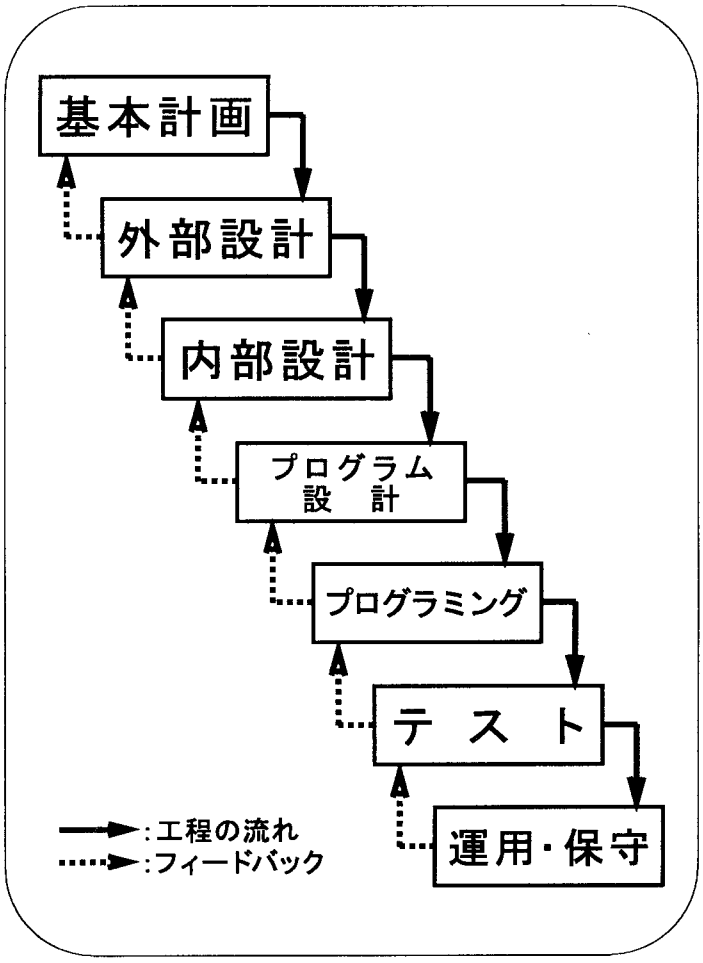
1　ウォータフォールモデル

現在のシステム開発手法の基礎となっているのがウォータフォールモデルである（図Ⅲ-1）。

ウォータフォールモデルによるシステムのライフサイクルは、基本計画、外部設計、内部設計、プログラム設計、プログラミング、テスト、運用・保守という順で進められており、この様子が、水が滝を流れ落ちるようであることからこの名が付けられた。もちろん、水が滝を流れ落ちる過程と同様に、開発工程が逆流することや所定の工程を飛び越すことはあり得ない。

従来から多くのシステムがこのサイクルに従って開発されており、現在でも多くのプロジェ

クトで採用されている。その大きな理由としては、各工程が明確に分かれており、それぞれの



図Ⅲ-１　ウォータフォールモデルによる開発工程

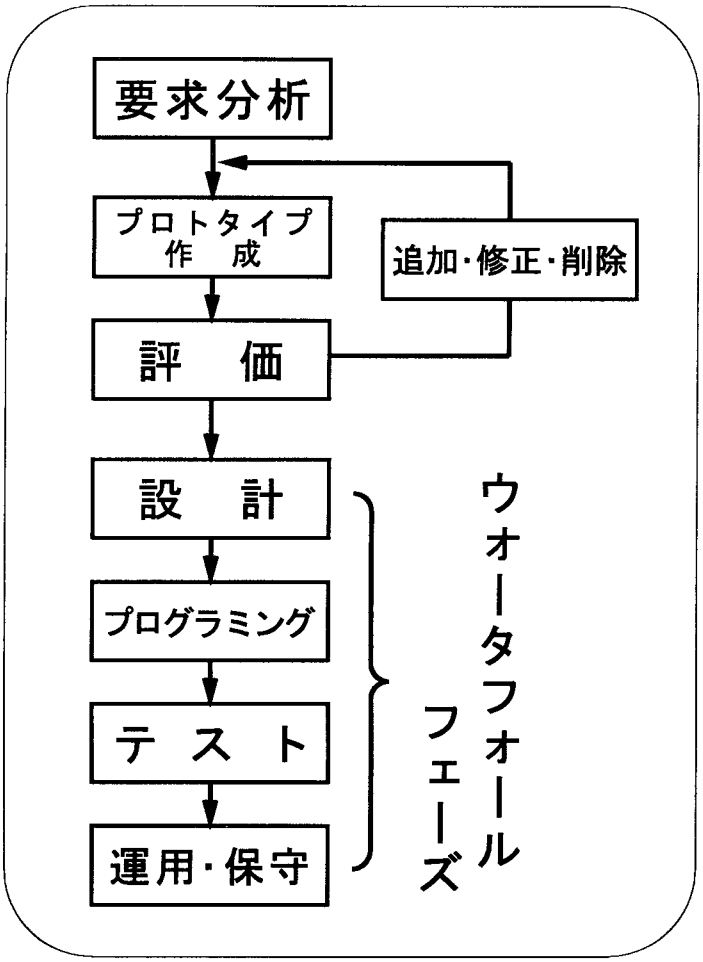
工程で作業内容のチェック・評価が可能で、しかも、ある工程での気付かなかった問題や小さなミスなどが次の工程で発見された場合は、フィードバックが許されるという点である。

ウォータフォールモデルでは、基本計画からプログラミングまでの工程では段階的詳細化技法（トップダウン開発）が用いられ、テスト工程では段階的統合化技法（ボトムアップ開発）が用いられる（図Ⅱ-２参照）。

２　プロトタイプモデル

開発の後半にならないとシステムの全容が分からないウォータフォールモデルの欠点を改善するために提案されたのが、プロトタイプモデルである（図Ⅲ-２）。このモデルでは、システム開発の初期の段階から試作品を作成することにより、ユーザ側と開発者側との意見のくい違いを吸収し、設計工程以降に大幅な仕様変更が発生するのを未然に防ぐことができる。したがって、ウォータフォールモデルよりも、よりユーザの要求しているシステムを実現すること

ができるという長所をもっている。また、試作品を作成しているという利点から、システム開



図Ⅲ-2　プロトタイプモデルによる開発工程

発の初期段階でのミスの発見確率も向上し、結果として開発期間の短縮、コストの低減などが図れる。

以上のように、特に大きな問題が発生しない限り、プロトタイプモデルはウォータフォールモデルよりも、システム開発全般にわたって優れた点を多く備えている。しかしながら、プロトタイプモデルでは、ユーザが納得する試作品が完成しない限り、次の工程に進めないという重大な欠点がある。すなわち、ユーザ側と開発者側の意見がくい違った場合は、初期段階から開発がストップしてしまうのである。グラフィカルなＯＳに代表されるように、最近は人間とコンピュータとのインタフェースが飛躍的に向上し、ユーザからの要求もますます厳しくなってきている。問題を回避しようと、試作品の作成に多大な労力を注ぎ込むことは、本来プロトタイプモデルがもっている長所が、逆に欠点となってしまう。全てのシステムがプロトタイプモデルで効率よく開発できるのではなく、対象となるシステムがプロトタイプモデルに適しているかどうかを見極めることが重要となる。

なお、プロトタイプモデルでは、試作品が作成された後はそれをもとに要求仕様書やインタ

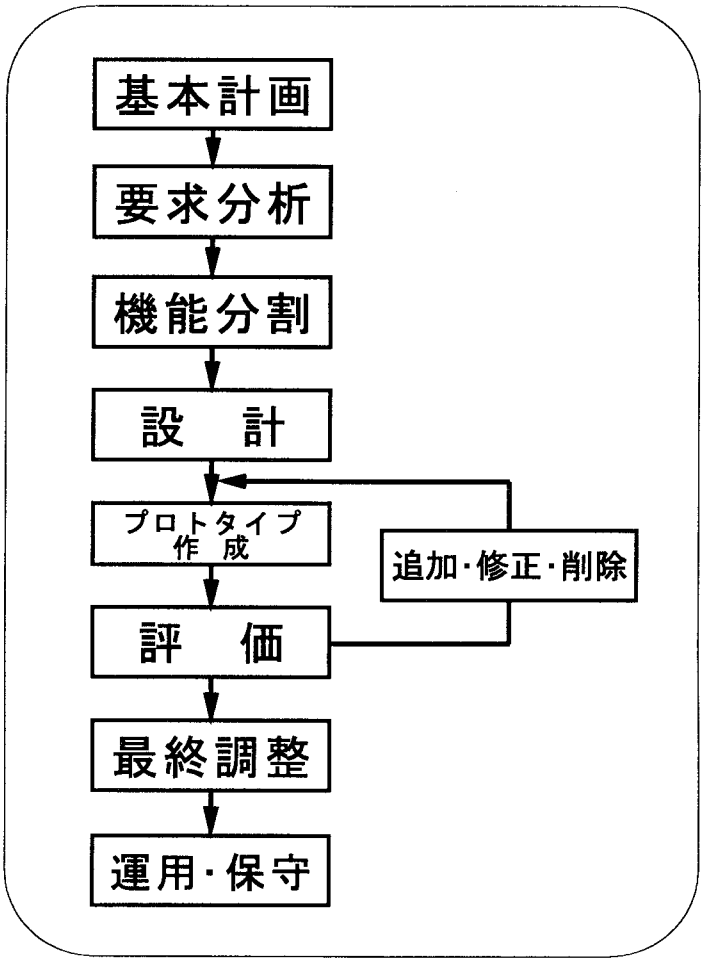
フェース仕様書などが作成され、以降の工程はウォータフォールモデルと同様に進められる。

3　スパイラルモデル

スパイラルモデルは、ウォータフォールモデルとプロトタイプモデルの利点をうまく融合したモデルである（図Ⅲ-３）。開発するシステムが独立性の高い部分・部品に分割可能なとき、すなわち機能分割が比較的容易に行える場合に、このモデルが使用できる。このモデルでは、機能分割された各ユニットごとに、設計、プログラミング、テストというウォータフォールモデルで採用した工程を順次進めていく。また、画面設計や帳票設計など、ユーザ側に大きく関わる部分は、プロトタイプモデルで採用した試作品を作成することとなる。

スパイラルモデルでは、システムが機能分割されることにより、各ユニットごとの同時開発が可能となり、その結果、そこで必要となる人員をより明確にすることができる。

システムの機能分割が比較的容易に行える場合、このモデルの信頼性は非常に高いといえ、逆に、機能分割が困難な場合は、他のモデルによるシステム開発が妥当となる。



図Ⅲ-３　スパイラルモデルによる開発工程

4　その他のモデル

先に述べたモデル以外にも、各企業や企業内の各グループ単位で独自のモデルを採用している場合が多くある。表Ⅲはそれらのモデルを示したものであるが、基本的にはウォータフォールモデルに若干の改良を加えたものであることが多い。なお、表Ⅲは、1990 年前後の資料であり、現在では、さらにプロトタイプモデルやスパイラルモデルなどの改良型が多く採用されていると考えられる。

表Ⅲ　その他のモデル

計　画		設　計			プログラミング	テ　ス　ト		
ｼｽﾃﾑ企画	ｼｽﾃﾑ分析	ﾕｰｻﾞｲﾝﾀﾌｪｰｽ設計	システム構造設計	ﾌﾟﾛｸﾞﾗﾑ構造設計		ﾌﾟﾛｸﾞﾗﾑテスト	結合テスト	ｼｽﾃﾑテスト

計　画		設　計				プログラミング	テ　ス　ト		
調査・立案	ﾌﾟﾛｼﾞｪｸﾄ計画	システム設計		プログラム設計			単体テスト	結合テスト	システムテスト
		初期設計	論理設計	ﾌﾟﾛｸﾞﾗﾑ構造設計	ﾓｼﾞｭｰﾙ設計				

調査・分析	基本計画	概要設計	詳細設計	プログラミング	テ　ス　ト

調査・分析	基本計画	DCD作成	DFD作成	HCP作成	プログラミング	テ　ス　ト