第 8 章　ソフトウェアの構成管理

本来の「構成管理」の意味

本来の「構成管理」と呼ばれる作業は、1960 年代に既に米国の国防総省で行われていた。例えばジェット戦闘機では、“F15”という同じ名前の飛行機が毎年少しずつ、しかしかなり長期間にわたって作り続ける。本来の「構成管理」は、このような製品に適用されるものである。

ジェット戦闘機について私は強くないので、これを民間機に置き換えて話を進めたい。ボーイング社が作っている航空機の 1 つに、有名な 747 型機がある。「ジャンボ・ジェット機」である。

この航空機の一番機は、1972 年秋に就航した。この年私は 9 月から 11 月までニューヨークに長期滞在し、IBM の学校に通って勉強していた。その期間中にデルタ航空がニューヨーク・タイムズ紙に全面広告を出して、ジャンボ・ジェット機 2 機がニューヨークとロサンジェルスの間を定期的に毎日 2 往復することを高らかに宣言した。このジャンボ機には、女性の名前が付いていた。たいへん印象的な広告だった。この原稿を書いている時点（2004 年 6 月）からすると、30 年以上も前の話である。

ボーイング社はそのジャンボ機を、最近まで作っていた。しかしこの 30 年を超える期間の間に、この飛行機の多くのものが変わっただろう。当然多くの技術革新があった。不幸な航空機事故があったかもしれない。航空機の場合事故があると徹底してその事故の原因究明がなされ、再び同じ状況が起きても同じ事故を起こさないようにするという一大方針の下で、設計変更などの改善がなされ続けている[1]。

これらの変更の結果、飛行機を構成している部品が変わり、その部品の組み合わせ方を表している設計図も変わった。その結果、最終製品である飛行機も変わり続けた。大げさにいえば、初めから全く変わらなかったものは「ボーイング 747」という型式名と、ワイパーぐらいだったかもしれない。

世界の民間航空会社の中には今でも多くのジャンボ・ジェット機を所有して、世界中で運行しているところがある[2]。航空機の場合は民間機でも定期的にドック入りしてオーバーホールを行い、一定の飛行時間ごとに一部の部品などを交換しなければならないと決められている。仮に 1 つの航空会社がいくつもの種類のジャンボ・ジェット機を所有しているとすると、次にドックに入ってくる飛行機にはどの部品や設計図が使われているかを事前に知っていなければ、このオーバーホールや部品交換が円滑には進まない。別の言い方をすると、どの飛行機にはどのタイプの部品が使われ、どの設計図に基づいて作られたか、などをしっかりと管理しておく必要がある。

このように最終製品が時間とともに変化する場合、そのそれぞれの最終製品とそれを構成している部品や図面の関係を管理する必要がある。これが本来の意味での「構成管理」である。つまり構成管理で管理の対象とするものは、基本的に「製品」である。この管理は、最終製品にも各部品や図面にも、その中間段階の半製品にも、それぞれバージョンナンバー（版番号）を付け、何かが変わるごとにそのバージョンナンバーを更新し、どのバージョンの最終製品は

---

[1] 余談だが、多発するトラブルを回避するために、ソフトウェアについてもこれぐらいの真剣な取り組みが必要であると私は考える。

[2] 日本の航空会社は日本航空も全日空も、所有していたジャンボ・ジェット機を 2007 年頃に全て売り払った。

どのバージョンの図面を使って、どのバージョンの部品の組み合わせで作られているのかを管理することで行われる。

ソフトウェアの構成管理とその目的

ソフトウェアが、飛行機などハードウェア製品と大きく違うところが 1 つある。飛行機などでは、一つ最終製品を製造するとそのために使う部品を消費してしまい、次の最終製品を作るためには改めて必要な部品を供給しなければならない。

しかしソフトウェアは、そうではない。今あるソフトウェアの最終製品を作っても、部品はまだそのまま存在し続けている。だから次の製品を作るときでも、新たな部品の供給の必要がない。これはソフトウェアの、他のものとは異なるたいへん大きな特徴である。

この特徴もあって、ソフトウェア会社の製品の作り方／出荷の仕方は、ハードウェア製品を作っている企業とは大きく違っている。世界最大のソフトウェア会社は多分マイクロソフトだが、マイクロソフトの製品の作り方／出荷の仕方は、ボーイング社のそれとは違っている。このことなどからマイクロソフトの場合部品の変更を、最終製品にそれほどは頻繁に反映していなかった[3]。したがって、ソフトウェア会社の場合はハードウェア会社の場合ほど、「本来の」構成管理を必要としていなかった。

しかしソフトウェアには、変更がつきものである。本来の構成管理には、この変更を管理する機能も含んでいる。この変更管理の部分に注目して、ソフトウェアの構成管理（Software Configuration Management：SCM）が考案された。

ソフトウェアを作り、運用している組織には、決して好ましいことではないが、以下のようなことが起きることがある。

1. プログラムの一部である特定のモジュール[4]の資料やソース・プログラムのリストが、見当たらなくなってしまった。
2. 順調に動いていたあるプログラムが、ある日突然動かなくなった。原因を調査したら、そのプログラムの中のあるモジュールを作った若いプログラマが、誰の許可も得ずに、良かれと思ってそのモジュールに手を入れた。その際にミスがあって、トラブルが発生した。
3. 開発中のソフトウェアに仕様変更が入って、プログラムを修正することにした。その際に連絡の不徹底があって、一部のプログラムは修正したが、修正されなかったものもあって、結果としてトラブルが発生した。
4. ある仕様変更があってプログラムを修正し、そのプログラムは修正後に順調に稼働していた。その頃に別の仕様変更があって、それへの対処を行った結果、以前に対処してそれ以降順調に動いていた修正が元に戻ってしまった。これでやはり、トラブルが

[3] 最近は、少し様子が変わってきている。つまりマイクロソフト製の OS の弱点を突いて、多くのコンピュータ・ウィルスが作られるようになった。このウィルスの被害を最小限にとどめるため、マイクロソフトは OS の中に「Windows Update」の機能を用意し、OS の一部に変更がある都度ユーザに、この機能を使って OS を修正することを推奨している。しかしこれらへの対応はユーザの責任で行うことになっており、マイクロソフトが個々のユーザが使っている OS の構成に責任を持っている訳ではない。その意味でやはりマイクロソフトの方法は、ボーイング社とは異なる。

[4] モジュールとは、大きなプログラムを構成する小さな単一の機能を持ったプログラムのことである。オブジェクト指向プログラミングでいう、「クラス」に対応する。

発生した。

ソフトウェアへの変更を管理して上記のようなことを起こさないようにすることが、ソフトウェアの構成管理の目的の 1 つである。これらのことに苦い経験を持つ組織は、このソフトウェアの構成管理を取り入れて、再発防止に努めることが必要である。

ソフトウェアの構成管理は必要か

ソフトウェアの構成管理の目的や、それを実施することによる効果については、既に述べた[5]。このようなことを行うために、「ソフトウェアの構成管理」とわざわざ呼ぶような、ある意味で大げさな作業が必要なのだろうか。もっと単純な、ポイントを突いた文書管理やライブラリ管理でこれと同じ目的を達成することはできないのだろうか。

答は「否」である。その理由は、ソフトウェアの複雑さにある。具体的には、以下のような事項をあげることができる。

1. ソフトウェアを構成するものの種類が多い。具体的には、要件定義書、各種設計書、ソース・プログラム、オブジェクト・プログラム、ロード・モジュール、さらには各種の計画書やテスト用データ、ユーザ用のマニュアルなどがある。しかもこれらがお互いに、しっかりと整合性を保持していなければならない。
2. ソフトウェアの形態が、開発作業の進展とともに変化する。つまり、最初の要件定義の段階で要件定義書が作られ、その後の設計段階でそれに各種の設計書が加わる。その後ソース・プログラムが作られ、コンパイルされ、リンクされて、プログラムだけで見ても形態が大きく変わる。
3. 時間とともに、特性も変わる。開発途中だけでも環境の変化や要求自身の陳腐化などで仕様の変更が生じる。仕様の変更が入るとそれに応じて、ソフトウェアを構成するものの一部を変更しなければならない。その上でそれら全部が、しっかりと整合性を保持した状態になっていなければなければならない。ソフトウェアのライフサイクル全体を見れば、この特性の変化はもっと激しい。

以上のようなことからソフトウェアでは、単純な文書管理やライブラリ管理だけでは充分な管理を行うことが難しい。これが「ソフトウェアの構成管理」と呼ぶ作業を必要とする理由である。

ソフトウェアの構成管理の内容

ソフトウェアの構成管理については、多くの規格やルールが用意されている。当然「開発のための CMMI（Capability Maturity Model Integration-DEV：能力成熟度モデル統合）[6]」にも「共通フレーム 2013[7]」にも、ソフトウェアの構成管理についての記述がある。いろんな記述が用意されていることは、この分野がそれだけ重要であることの証拠である。しかし残念ながらこのソフトウェアの構成管理は、特にソフトウェア技術者たちから評判が悪く、多くのソフトウェアの開発組織が積極的に採用に踏み切り、導入しようとしているような仕組みでは必

---

[5] 今の ISO／IEC 12207：2008 をはじめ、共通フレーム委 2013 などには、「構成管理」と「ソフトウェアの構成管理」に 2 つが用意されている。

[6] CMMI については、第 40 章で述べる。

[7] 共通フレーム 2013 とその基の規格である ISO／IEC 12207：2008 については、第 12 章で述べる。

ずしもない。このスタンスの良否などについての問題は、また別のところで論じたい。

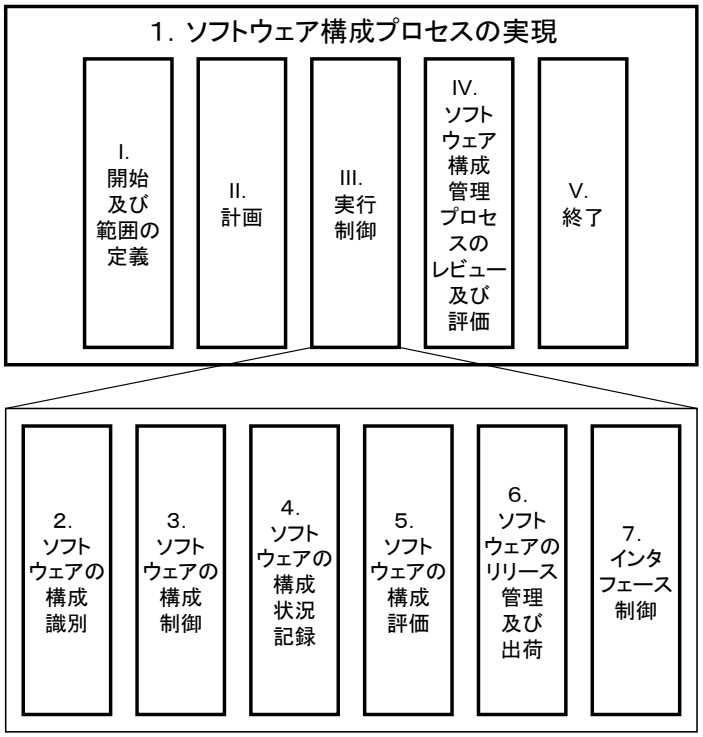
国際規格群の中の ISO/IEC TR 15846：1998 という技術報告書があった[8][ISO98]。この技術報告書によると、ソフトウェアの構成管理は以下の 7 つのプロセスから構成される[9]。

1. ソフトウェア構成管理プロセスの実現
2. ソフトウェアの構成識別
3. ソフトウェアの構成制御
4. ソフトウェアの構成状況記録
5. ソフトウェア構成評価
6. ソフトウェアリリース管理および出荷
7. インタフェース制御

以下で、この技術報告書を中心に、ソフトウェア構成管理のプロセスの内容について述べてみたい。

ソフトウェア構成管理プロセスの実現

上記の 7 つのプロセスのうち、最初の「ソフトウェア構成管理プロセスの実現」のプロセスは、さらに次の 5 つのサブプロセスに分けられる。



図表 8-1　ソフトウェア構成管理の全体像

[8] この規格（技術報告書）は、たぶん後続の規格が発行されないまま廃止されてしまった。

[9] 私はこの内容を、この技術報告書が JIS 化された TR X 0018：1999 から取った[JIS99]。構成管理についてのいろんな記述のうち、この技術報告書の記述が一番的確であると私は考えている。

I.　開始および範囲の定義
II.　計画
III.　実行制御
IV.　ソフトウェア構成管理プロセスのレビューおよび評価
V.　終了

この「ソフトウェア構成管理プロセスの実現」のプロセスは、他の「ソフトウェア構成識別」から「インタフェース制御」までの 6 つのプロセスより一段高く位置づけされていて、この中に他の 6 つのプロセスが全て包含されている。つまり、このプロセスの「Ⅲ 実行制御」の中に、「ソフトウェア構成識別」から「インタフェース制御」までの 6 つのプロセスが全て置かれることになる。この関係を図示すると、図表 8-1 のようになる。

このように見てみるとソフトウェアの構成管理全体のプロセスも、最初に準備の段階があり、それに基づいて計画を立て、計画に従って実行し、レビューと評価を行い、最後に終了するという段階を踏む。つまりここで 1 つの PDCA サイクル[10]を回すことになり、ソフトウェアの構成管理の仕事に進め方はソフトウェア開発の他の仕事の進め方と何ら変わることはないということになる。

ソフトウェア構成管理の開始および範囲の定義

この最初に位置づけされている「開始および範囲の定義」の作業は、計画作成前に行うものである。別の言い方をすると、ここで検討した結果などがソフトウェアの構成管理作業の大枠を決め、さらにその計画に記述されることになる。共通フレーム 2013 によれば、次のような事項を決定する作業がここで実行されることになっている[IPA13a]。

1. 構成管理におけるアクティビティ
2. そのアクティビティを遂行するための手続きと予定
3. そのアクティビティを遂行する責任を負う組織
4. 関係する他の組織

つまり共通フレーム 2013 によれば、この段階で明らかにしなければならないことは、ソフトウェアの構成管理を遂行するために、何を、どのように、誰（どの部門）が、行うのかを明確にすることである、としている。しかしこれだけでは必ずしも充分ではなく、これらに加えて、いつその作業を行うのか、その作業を行うに当たってどのような資源が必要か、などについても同時に明らかにしておかなければならない。つまり、構成管理の作業の実施に関する 5W1H を明確にする必要があるということになる。そしてこれらの内容は、後で述べるソフトウェア構成管理の計画に記載されなければならない。

ソフトウェアの構成管理は単に共通フレーム 2013 でいう開発プロセスをカバーするだけでなく、保守や運用のプロセスを含むソフトウェア・ライフサイクル全体で実施することができる。むしろ本来的には、ソフトウェアの構成管理がカバーするべき領域は、保守などを含めたこの広い範囲であるべきだろう。その前提で、前述の技術報告書にはここで行う作業として、具体的に以下のものなどを決めることをあげている[JIS99a]。

1. ソフトウェアの構成管理の対象にするソフトウェア製品の決定、構成品目の決定と構成品目識別の仕組み
2. ソフトウェアの構成管理の計画書に記述された通りに作業が行われている証拠、また

[10] PDCA サイクルについては、ISO9000 シリーズに関する議論をする第 39 章で論じる。

は保証の識別
3. ソフトウェアの構成管理の各作業を行う、あるいは責任を持っている組織の識別と、これらの組織体のソフトウェアの構成管理での役割と責任の明確化
4. 構成品目を構成制御の下に移す方法、およびその変更の承認／拒否の基準
5. 版管理の仕組み
6. 構成項目の構造を説明する文書の明確化
7. 構成品目の状態の意味

前述の通りここでの検討結果を基に、次に述べる計画を立案する。

ソフトウェアの構成管理の計画

ソフトウェアの構成管理の範囲が定義されると、それを前提にソフトウェアの構成管理を実施する計画を立案することになる。この計画に盛り込むべき内容ついて、IEEE にその規格がある。IEEE Std 828-2012 と呼ばれるものがそれである[IEEE12a]。

これによると、ソフトウェアの構成管理の計画については、以下の内容を以下の順序で記述するのがよいとしている。

1. はじめに
2. ソフトウェアの構成管理のマネジメント
3. ソフトウェアの構成管理の活動
4. ソフトウェアの構成管理のスケジュール
5. ソフトウェアの構成管理実施のためのリソース
6. この計画のメンテナンス

この規格には、計画についてもっと詳細な内容についての記述がある。しかし基本的には、既に記した検討の結果を記述することでよい。

ソフトウェアの構成識別

計画が立案されると、それに基づいて実際の作業が開始されることになる。それをここでは、「実行制御」と呼んでいる。前述の通りこの「ソフトウェアの構成識別」から「インタフェース制御」までの 6 つの作業が、「実行制御」の中に含まれる作業である。この 6 つの作業で、小さなもう 1 つの PDCA サイクルを構成している。

ソフトウェアの構成管理の目的は、ソフトウェアへの変更要求を管理し、それにうまく対応することにあると述べた。そのソフトウェアの構成管理での具体的な作業としての最初のものである構成識別の作業は、以下の 2 つから構成されている。

1. 構成品目を明確にすること
2. 構成品目の管理の方法を明確にすること

構成管理の対象物として、その変更の管理と対応を行うものをソフトウェア構成品目（Software Configuration Item：SCI）と呼ぶ[11]。構成識別での最初の作業は、まずこの構成品目を明確にし、それらがそれぞれ別のものとして識別できるようにするすることである。

スティーヴ・マコネル（Steve McConnell）は、ソフトウェア開発プロジェクトの成果物で

11 ソフトウェアの構成管理の考え方や手順／方法は決して難しいものではない。しかしここで使われる言葉はやや特殊なので、あるいは注意が必要かもしれない。「構成品目」とは一言でいえば、「構成管理で管理の対象とするもの」である。

は、次のようなものが構成品目になりうるとしている[12][MCC98]。

- 変更管理計画書
- 変更提案書
- ビジョンステートメント
- トップ 10 リスク一覧
- ソフトウェア開発計画書（プロジェクトのコストとスケジュールの見積もりを含む）
- ユーザインタフェースのプロトタイプ
- ユーザインタフェースのスタイルガイド
- ユーザーマニュアル兼要求仕様書
- 品質保証計画書
- ソフトウェアアーキテクチャ設計書
- ソフトウェア統合手順書
- ステージ別納品計画
- 個別ステージ計画書（小規模マイルストーンの達成スケジュールを含む）
- コーディング規約書
- ソフトウェアテストケース定義書
- ソースコード
- 製品に組み込んだメデイア（グラフィック、サウンド、ビデオなどのファイルを含む）
- ソフトウェアビルド指示書（make ファイル）
- 各ステージの詳細設計文書
- 各ステージ用のソフトウェア構築計画書
- インストールプログラム
- 導入マニュアル（カットオーバーハンドブック）
- リリースチェックリスト
- リリース承認用紙
- プロジェクトログ
- プロジェクト履歴

これらの構成品目ごとに適切な識別子をつけて、それぞれが別のものとして区別できるようにする必要がある。識別子として、例えば次のようなものを考えることができる。ここで、「版番号」が常に最後についていることに留意して欲しい。

- 文書
  - 文書番号、文書名、作成日付、版番号
- プログラム
  - プログラム ID、作成日付、版番号

この構成品目群を管理するためのライブラリを、ソフトウェア・ライブラリと呼ぶ。

次に必要な作業は、どのような場合にこの構成品目を構成管理の対象にするのかという基準などを明確にすることである。ソフトウェアの構成管理の方法を、図表 8-2 に示す。

---

[12] ここに挙げられている文書名等はマコネルがその著書の中で説明しているもので、この原稿ではほとんど説明していない。詳細はマコネルの著書[MCC98]を参照していただきたい。ここでこれを示した理由は、プロジェクトで作成する成果物（ドキュメント類だけでなくプログラムも含む）が全部構成品目になりうることを示したかったということである。

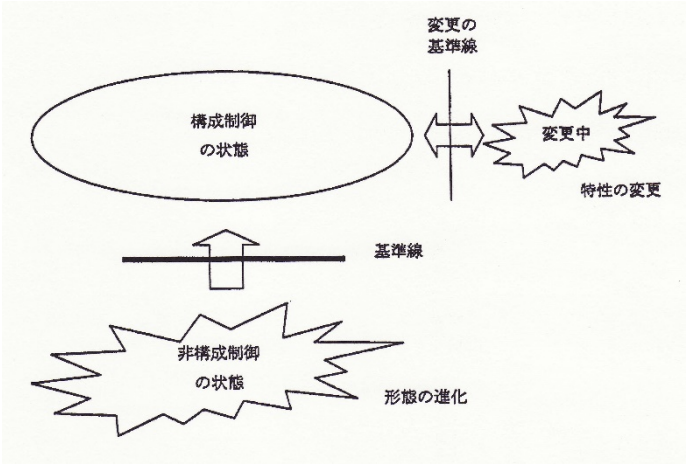
新しく作られる構成品目（文書やプログラムなど）は、あるレベルになると「基準線」と記された線を超えて、構成制御の下に置かれる。この状態で、構成制御の下に置かれた文書やプログラムへの変更は非常に厳しく管理されている。

ここで管理されている構成品目に内容の変更が必要になれば、その構成制御の仕組みに従って変更が許可され、変更作業が実施される。構成制御の仕組みについては、次項で述べる。変更作業中の構成品目は、一度この構成制御の下からはずされる。そして変更が終了すれば、改めて構成制御の下に置かれる。この場合に、版番号が更新されることになる。

構成識別の 2 つ目の作業として行われるものは、構成品目を基準線を超えて構成制御の下に移す条件と手続、版番号更新の基準と手続などを明確にすることである。一般にプログラムは単体テスト終了時点、文書類はレビュー終了時点に、構成制御の下に移される。

構成品目は、我々が作成したソフトウェア（最終製品と部品）だけである必要はない。外部から購入したり、外部に委託して開発したり、ベンダーから提供されたりしたもの、あるいは個々の部品を組み合わせた中間製品など全てを、必要なら構成品目に含めることができる。当然 OS も、対象にして良い。

なおソフトウェアの構成品目は、時間の経過とともに状態を変えることになる。具体的には、当初これらは「開発段階」にあることになるが、カットオーバーや製品出荷の時期になると「リリースおよび出荷段階」にその状態が変わる。このような構成品目の状態の変更を「昇格」と呼んでいる。



図表 8-2　ソフトウェアの構成管理における管理方法
（[JIS99a]より）

ソフトウェアの構成制御

ソフトウェアの構成制御とは、構成管理の対象になっている構成品目に変更が必要になった場合、それに対応するための全ての必要な活動を指す。この一連の作業の流れを、図表 8-3 に示す。

ソフトウェアの構成制御の下に置かれている構成品目に変更が必要になったと認識した人は、まずその事実を所定の書式に記載し、変更を要求する。

構成管理チームはこの変更要求を受け取ると、必要ならチーム外の人の協力も得て、この要求によって変更しなければならない構成品目は何で、その変更にどれくらいの工数を必要とす

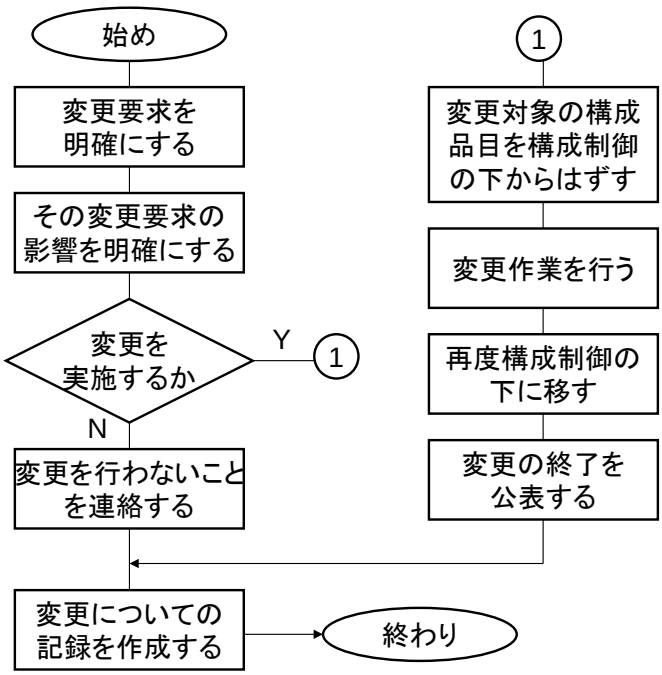
るのか、変更は不可欠か、不可欠でない場合この変更でどのような効果が期待できるか、などを明らかにする。

ソフトウェアの構成管理を行う場合に、この変更要請を受けるか拒否するかを決めるための委員会をあらかじめ設置しておく。この委員会を、ソフトウェア構成委員会（SCB：Software Configuration Board）と呼んでいる。小さなシステムではこの委員会の構成員は 1 人でもよいとされている。しかし私は個人的な経験から、最低でも 2 名以上が必要と考えている。

原則として定期的にこの委員会を開いて、それぞれの変更要求に対して、変更は不可欠か、不可欠でない場合にはコスト対効果などを基に、その変更を実施するかどうかを決める。

変更しないと決まった場合は、理由とともにその決定を提案者に連絡し、記録を取って一連の手続を終える。

変更する場合は変更を行う担当者を決め、変更するべき構成品目を構成制御の対象から一度外し、変更を加え、必要なレビューやテストを実施して、構成制御の下に移すための条件を充足していれば、改めて構成制御の下に移す。その際に、変更についての記録を取り、管理する。併せて、構成品目に変更が実施されたことを周知徹底する。同時に、変更された構成品目の版番号を更新する。



図表 8-3　構成制御の流れ

変更についての記録などについては、次の「ソフトウェアの構成状況記録」で述べる。

以上が、ソフトウェアの構成制御についての作業の流れである。

ソフトウェアの構成状況記録

ソフトウェアの構成管理に関して、その状況を克明に記録する必要がある。この記録は、構成品目の状況を把握したり、次に述べる「ソフトウェア構成評価」で構成管理が適切に実施されているかを判断したりするためなどに使用する。

構成状況の記録は、まず構成制御で変更要求が起票されるところから始まる。この変更要求のあるものは承認されて実際の変更に結びつき、あるものは拒否されて変更要求としてはそれで終わりになる。いずれの場合もこの過程、承認／拒否の理由、および変更が実施された場合にはその変更の内容、担当者、変更に要した期間や工数など、変更についての情報を記録する。

さらにソフトウェアの構成管理全体として、定期的に次のような内容の報告書を作成して、関係者に報告する。この報告書作成作業も、ここに位置づけされる。

1. ソフトウェア製品の構造
2. 各構成品目の状況
3. 全ての変更要求の状況
4. 承認された変更、およびその修正によって変更された内容と結果
5. 拒否された変更要求には、その拒否の理由など
6. 未解決の修正など
7. ソフトウェアの構成管理の作業全体で何らかの問題があれば
   ① その問題の内容
   ② 問題点を含んだ構成品目
   ③ 問題が引き起こす結果
   ④ 問題の解決方法
   ⑤ 必要に応じて、一時的な解決策

ソフトウェア構成評価

ソフトウェア構成評価では次のような事項を調査して、ソフトウェアの構成管理作業が全体として適切に運営されており、有効に効果を発揮していることを確認し、報告する。この基になる情報は、前記「ソフトウェアの構成状況記録」で記録されたものである。

1. ライブラリに保管されている構成品目が、それについての記録と一致していること。
2. ソフトウェアの構成品目が、承認された変更要求にしたがって適切に変更され、管理されていること。
3. ソフトウェア製品が、その製品を構成している構成品目について完全であり、利用可能であること。

ソフトウェアのリリース管理および出荷

ソフトウェアは最終的に、本番稼働されなければならない。そのために、リリース、あるいは出荷という手続が必要になる。したがって計画立案時にはリリース、および出荷についても計画を立て、この作業の実施に当たってもその計画に基づいて、作業を行う必要がある。

言うまでもないことだが、リリース、あるいは出荷の対象になるものはプログラムだけでなく、文書類もその対象になる。ここでリリース、あるいは出荷される構成品目は、そのソフトウェア製品の寿命がある限り保守作業の対象になる。

リリース、あるいは出荷されるものは、直接的には個々のモジュールや文書類であるが、しかし考え方として、リリース、あるいは出荷されるものは「製品」としてのソフトウェアである。したがってここではその「製品」を構成しているプログラムや文書などの構成品目を明確にするため、本来の意味での「構成管理」が必要になる。

開発の段階からリリース、及び出荷の段階に進むためには、構成品目の「昇格」が必要になる。

インタフェース制御

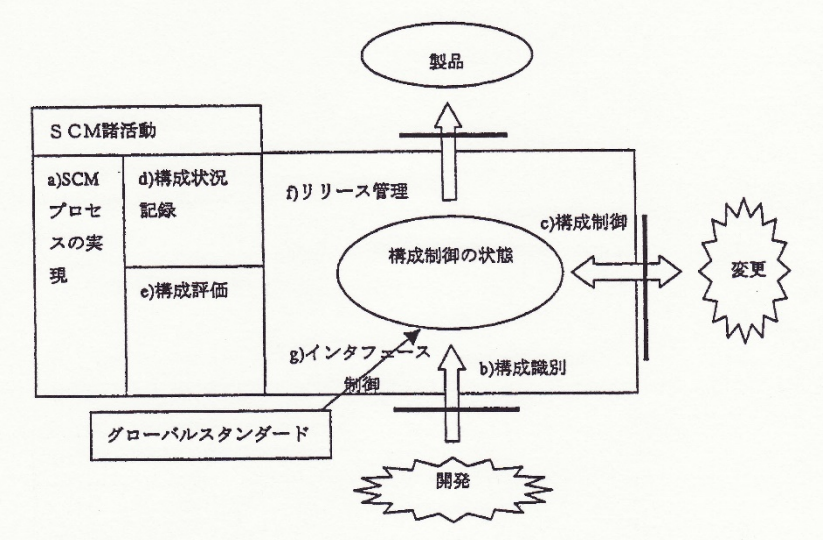
OS などのシステム・ソフトウェアやミドルウェアなどを開発する場合、ハードウェアや他のシステム・ソフトウェア、支援ソフトウェア、並列／並行開発によって製品化されるソフトウェアなどとのインタフェースが重要になる。そのような場合ソフトウェアの構成管理では、インタフェースについての文書を作成し、管理し、必要に応じて適宜変更しなければならない。

インタフェースについての文書を作成する場合、それに記載する内容には以下のようなものがある。

1. インタフェースの目的
2. インタフェースにおける要求事項
3. 影響を受ける組織
4. 管理するべきインタフェースについての文書
5. インタフェースに影響を与える変更要求を他者に伝える手順、およびインタフェースへの影響の評価を共同または個別に行う手順
6. インタフェースの変更管理責任者を含む、インタフェース文書の承認、変更および発行の手順
7. インタフェース文書の変更を、他の構成品目に対する変更として解釈する手順
8. 役割と責任

ソフトウェア構成管理の作業の全体像

「ソフトウェア構成プロセスの実現」の中に位置づけされる「ソフトウェアの構成識別」から「インタフェース制御」までのこれまで述べてきたソフトウェアの構成管理に関わる作業の全体像を、図表 8-4 に示す。



図表 8-4　ソフトウェア構成管理作業の全体像（[JIS99a]より）

この一連の作業それ自体、前述の通り 1 つの小さな PDCA サイクルを構成していると考えることができる。この一連の作業は、ソフトウェアの構成管理全体で構成される一段大きな

PDCA サイクルの「実行（D)」の部分に相当することになる[13]。

ソフトウェア構成管理プロセスのレビューおよび評価

このソフトウェアの構成管理の作業の中で、ソフトウェアの構成管理の作業全体が当初の計画に基づいて適切に実行されていることを保証しなければならない。仮に両者の間に不一致がある場合は、共通フレーム 2013 でいう「問題解決プロセス」や「プロセス修正プロセス」を実施して、両者の間に不整合がないようにしなければならない。

このための作業が「ソフトウェア構成管理プロセスのレビューおよび評価」である。これは、大きな PDCA サイクルの「チェック（C)」に相当するものでる。この結果に基づいて適切な修正の措置が執られると、それが「行動（A)」に相当することになる。

ソフトウェア構成管理作業の終了

ソフトウェアの構成管理が単に開発プロセスを支援の対象にしているだけでなく、運用や保守プロセスまでも支援している場合には、この構成管理作業には、あるいは終わりはないかもしれない。しかし開発作業だけを支援の対象にしている場合などでは、構成管理の作業には「終わり」がある。

構成管理作業に終わりがある場合には、当初に立てた計画に従って、この作業を適切に終了させることが必要である。

ソフトウェア構成管理から得られる情報

ソフトウェアの構成管理の作業全体を通して、以下のような情報を得ることができる。

1. 構成品目識別の仕組み
2. 構成品目の版管理の仕組み
3. 構成項目の構成
4. 構成品目の状態の意味
5. それぞれの構成品目の状況
6. 構成品目の状況の完全性

トレーサビリティへの対応

今開発中の情報システムに仕様変更が入って要求仕様書が変更になれば、それに応じて対応する機能を記した設計書が変更され、該当するソース・プログラムに変更が入り、場合によればテストシナリオやテストデータも変更されるかもしれない。ソフトウェア全体を的確に保守しようとすると、この多くの文書類を適切に変更することが必要である。

しかし現在のソフトウェアの構成管理では、構成品目間にトレーサビリティについての紐付けを行うことについて、何も述べていない。つまり構成品目同士は、常に独立したものとして取り扱われている。上で述べたようなことを的確に実行するためには、ソース・プログラムを含む文書類の各部分に記載されているものが他の文書類のどこと関連しているかを把握しておきたい。これが「トレーサビリティへの対応」と呼ばれるものである。

現在のソフトウェアの構成管理でこれを行うためには、構成制御の過程で変更対象を的確に洗い出して対応するしか方法がない。この作業を容易にし、より的確に行うために構成品目間

[13] よくあることだが、ここでも PDCA サイクルは二重になっている。

にトレーサビリティへの対応のための紐付けを行っておくのがよい。これによって、例えば要件定義書のある部分が変更されなければならなくなると、該当する設計書とソース・プログラムなどの変更必要箇所も容易に把握することができる。

あるユーザは社内に「ソフトウェア生産管理システム」と名付ける情報システムを用意し、その中にソフトウェアの構成管理の機能とこの紐付けの両方を取り込んで、効果を挙げている。

ソフトウェアの変更作業の支援

正規のソフトウェアの構成管理の規格には入っていないが、この作業を支援するツールの中には、変更作業を支援する機能を持ったものがある。

その機能には、以下のようなものが含まれている。

1. 同じ構成品目に、同時に複数の変更が入らないようにコントロールする。つまりある変更が終了して再び構成制御の下に戻ってくるまで、次の変更のためにその構成品目を構成制御の下から他のところに移すことができないようにする。
2. あるいは、同時に並行した複数の変更を許す。つまり変更された構成品目を構成制御の下に戻す時、並行してなされていて既に作業を終了し、構成制御の下に戻っている変更の内容が改めて正しく反映されるように、注意を促す。
3. 任意の構成品目を、ある特定の時点の古い版に戻す。

国際規格の中でのソフトウェアの構成管理の位置づけ

ISO の規格の 1 つに ISO 10007：2003 があり、「品質マネジメントシステムー構成管理の指針」という標題を持っている[ISO03c]。この規格はハードウェアなどを含む一般的な製品についての構成管理の規格であって、ソフトウェアの構成管理はこの規格と整合性を保っている。

ソフトウェアの構成管理の作業は、ISO／IEC 12207：2008 の国際規格の中では「支援プロセス」の 1 つとして位置づけされている。またそれを受けた共通フレーム 2013 では、構成管理は「2. 支援ライフサイクルプロセス」の中の「2. 2 構成管理プロセス」として位置づけされている。

またカーネギー・メロン大学ソフトウェア工学研究所（SEI：Software Engineering Institute）が設定した「開発のための CMMI」の段階表現では、ソフトウェア構成管理はレベル 2（「管理された」）でのプロセスに位置づけされている[CMM10a]。

つまりこれらのことは、ソフトウェアの構成管理の作業がソフトウェア工学の中で、極めて基本的なところに位置づけされていることを示している。

キィワード

構成管理、本来の構成管理、ソフトウェアの構成管理、PDCA サイクル、構成識別、構成制御、構成状況記録、構成評価、リリース管理および出荷、インタフェース制御、構成品目、ソフトウェア・ライブラリ、基準線、版管理、ソフトウェア構成委員会、SCB、昇格、トレーサビリティ、ソフトウェア生産管理システム、開発のための CMMI

略語

SCB：Software Configuration Board

CMMI：Capability Maturity Model Integration

規格

IEEE Std 828-2012、ISO／IEC 12207：2008、ISO 10007：2003、CMMI-DEV v1.3、共通フレーム 2013、

人名

スティーヴ・マコネル（Steve McConnell）

参考文献とリンク先

[CMM10a] CMMI 成果物チーム、「開発のためのCMMI® 1.3 版 CMMI-DEV, V1.23 CMU/SEI-2010-TR-033 ESC-TR-2010-033 より良い成果物のためのプロセス改善」、カーネギー・メロン大学ソフトウェア工学研究所、2010年

この資料は、次の URL からダウンロードできる（確認日：2017 年（平成 29 年）1 月 25 日）。

http://cmmiinstitute.com/resource/japanese-language-translation-of-cmmi-for-development-v1-3/

[IEEE12a] Software Engineering Standards Committee of IEEE Computer Society, “IEEE Std. 828 Software Configuration Management,” IEEE, 2012.

[IPA13a 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター編、「共通フレーム 2013～経営者、業務部門が参画するシステム開発および取引のために～ ソフトウェアライフサイクルプロセス 共通フレーム 2013」、（株）オーム社、平成 25 年.

[ISO98] ISO／IEC, “Information technology -- Software life cycle processes -- Configuration Management ISO/IEC TR 15846:1998,” ISO／IEC,1998.

[ISO03c] ISO, “Quality Management System – Guideline for Configuration Management ISO 10007:2003,” ISO, 2003.

[JIS99a] 日本工業標準調査会 情報部会 審議、「ソフトウェアライフサイクルプロセスー構成管理 TR X 0018：1999 （ISO/IEC TR 15846：1998）」、日本規格協会、平成 11 年.

[JIS12a] 日本工業標準調査会審議、「ソフトウェアライフサイクルプロセス JIS X 0160-2012（ISO／IEC 12207：2008）」、日本規格協会、平成 24 年.

[MCC98] Steve McConnell 著、（株）アルテア・ジャパン訳、久手堅憲之監修、「新訳ソフトウェアプロジェクトサバイバルガイド」、日経 BP ソフトプレス、2005 年.

この本の原書は、以下のものである。

Steve McConnell, “Software Project Survival Guide,” Microsoft, 1998.

（2004 年（平成 16 年）7 月 20 日　初稿作成）
（2006 年（平成 18 年）8 月 17 日　一部修正）
（2007 年（平成 19 年）5 月 20 日　一部追加）
（2007 年（平成 19 年）8 月 30 日　一部修正）
（2014 年（平成 26 年）3 月 10 日　一部修正）