ソフトウェア工学 72－3
（１９９０．５．２９）

ライフサイクル・モデルへの一考察

－ウォーターフォール型ライフサイクル・モデルのフェーズ分けについて－

加藤　重信
凸版印刷株式会社　総合研究所

システム開発の過程をライフサイクル・モデルとして議論することが多い。いろいろなモデルが提案される中で，現実のシステム開発においては意識するかしないかにかかわらず伝統的なウォーターフォール型のライフサイクル・モデルによっていることが少なくない。システムの品質を開発過程における品質保証という観点から見ると，ウォーターフォール・モデルでいわれるテスト工程は，その検証方法に説明しづらい側面を持つ。この理由を考察するとともに，フェーズ分けの考え方について提案する。

A CONSIDERATION ON SOFTWARE LIFE CYCLE MODEL

Shigenobu Katoh

Image Technology Laboratory　Technical Research Institute
TOPPAN PRINTING Co., Ltd.

3-3 Suido 1-Chome, Bunkyo-Ku, Tokyo, Japan

We often consider a proccess of software development as a software lifecycle. There are many kind of lifecycle model offered in the world, however most of the systems follow so called Water-Fall-Lifecycle model. From the viewpoint of quality assuarance of software development proccess, the test phase of Water-Fall-Lifecycle model does not conform to the verification proccess. Then I would like to point out a reason of non conformity and to propose an idea for the definition of test phase.

1．まえがき

まず最初に，ここに発表する話題は現在の所属とはまったく無関係であることをお断りしておく。ここ数年の間に，ＩＳＯ規格作成や，システム開発のコンサルテーションなどを通じて多くの人々と議論をする中で，ウォーターフォール型のライフサイクル・モデルにおけるテスト工程の位置付けに何か不自然さを感じてきた。昨年ある企業において「テスト作業の効率向上」のためのプロジェクトに参加する機会があり，この議論の中でこれまでに感じてきた不自然さの原因を見つけたような気がするので，識者の皆様と議論するきっかけとすべく話題を提供することにした。

2．テスト工程と検証

ウォーターフォール型のライフサイクル・モデルは，その名が示すように水が川上から川下に流れゆく様に似ていることから連想されたもので，一般に開発作業が順次に展開されていく状況と合致するのでシステム開発を説明する際に非常に良く引き合いに出される。現実のシステム開発の工程に合わせて要求定義，設計，開発，テスト及び運用の段階に分けて説明されるのが通常であるが，観点を変えて見ると，テスト工程の位置付けに不自然さを感じる。特に作業の正当性を検証・確認するという観点に立って見るとテスト工程の検証という厄介な問題に遭遇する。テストという作業そのものが検証工程であるため，その妥当性の確認を何に求めるのかがライフサイクル・モデルを議論するときにいつも問題になってきた。この問題を実際の議論の中で取り挙げた経緯があるので紹介しておきたい。

3年前から，ＩＳＯ／ＴＣ１７６（品質管理及び品質保証）の中のＳＣ２／ＷＧ５に関係して規格原案作成に携わってきた。この作業部会は「ソフトウェアの品質保証」についての標準を作成するものであり，日本規格協会の中に組織された対応する日本の国内委員会ＪＮＣ／ＴＣ１７６／ＷＧ１５に参加して，規格原案の検討に加わってきている。現在は日本の国内委員会に加えて，上記のＷＧ５にもメンバーとして登録されている。このＷＧ５では，二者間の契約による設計から保守までを含むソフトウェア開発における品質保証のガイドラインを設定すべく検討を重ね，原案を作成してきた。昨年の会議においてＤＩＳ（Draft International Standard）として，ＴＣ１７６の加盟各国の郵便投票に掛けることが議決され，現在は投票結果を待っている状況である。この規格は，すでに制定されている一般的な産業製品，特に組立て形式の製品に適用する規格ＩＳＯ　９００１　品質システム－設計／開発，製造，据付け及びアフターサービスの品質保証－のソフトウェア用の版を狙っているものである。この規格では，ソフトウェア（システム）の開発過程について特定のライフサイクル・モデルに依存しないことにして議論を重ねてきたが，議論に参加する人々の知識の範囲がウォーターフォール型のライフサイクル・モデルに強く依存している関係で，陽には明示していないもののウォーターフォール型のライフサイクル・モデルを暗に想定している。品質保証の議論であるので何に対して何が保証されるのかが当然議論の中心である。最近，欧米において日本流の品質作り込みという概念に対抗するように，Ｖ&Ｖ（Verification and Validation）という概念が強く打ち出されている。言い替えれば欧米流の品質作り込み技術ということになると思われる。このＶ&Ｖの考え方が上記のＤＩＳ規格はＤＩＳ９０００／３と呼ばれるものであるが，この規格の原案作成の段階の議論で

ライフサイクル・モデルの各工程のV&Vを何によって行うかが議論された。Verification（検証）とValidation（妥当性確認）は別の事象であるとする日本とVerificationとValidationには区別が付けられなくなっているとする欧米各国との議論はあったが，要求定義から順にV&Vについて確認していくと，テスト工程に至って検証並びに妥当性確認をする基本となる対象がないことが分かってきた。

また，昨年たまたまある企業のシステムコンサルテーションを手掛ける中でテスト作業の効率向上が主題となり，ＩＳＯ規格の作成において疑問になっていた話題でもあるので，テスト工程のV&Vの観点から検討し直してみることにした。その結果，ウォーターフォール型のライフサイクル・モデルにとってテスト工程という切り出し方がまずいのではないかと感じるようになった。確かにテストを実施している段階は存在しているのではあるが，テストをすることが目的なのではなく，テストをを手段としている工程があると考えることの方が妥当であると結論付けられるようになった。

３．ライフサイクル・モデルにおけるテスト

一般に事務処理等のシステム開発においては，ウォーターフォール型のライフサイクル・モデルに従うことが多い。これは単にこのモデルが良く知られているだけではなく，その適用性が高いからだと思われる。しかし，システム開発の過程を見直して見ると，大きく分けて，要求の定義・詳細化の段階，要求仕様をコンピュータで実行可能なもの（すなわちプログラム）に変換する段階及び運用可能なシステムに組み上げる段階の３つに分けられる。第１の段階はユーザの曖昧な要求を明確な要求にまとめ，その要求を段階的に詳細化する過程であると定義することができる。この段階の最終出力は作成すべきプログラム仕様である。第２段階は，この詳細化されたそれぞれのプログラム仕様をコンピュータで実行可能なプログラムに変換する過程となる。最後の第３段階は，したがって詳細化された要求を実行可能な形にしたもの，すなわちプログラムを組み上げて，要求仕様に対応するシステムを構成する過程となる。

第１段階は，通常外部仕様記述，内部仕様記述，サブシステム仕様記述と段階的に詳細化されていき，最終的にプログラム仕様を出力する（図１参照）。第２段階は，図３に示すように，コンピュータで実行可能なプログラムを作成する過程であり，一般的なプログラム作成の工程と対応すると考えればよい。第３段階以降がウォーターフォール型ライフサイクル・モデルでいうテスト工程ということになるが，図２に示すようにテスト工程と見るよりは詳細化されたプログラムを要求されたシステムとして実現するための組上げの工程と見る方が実体に近い。

４．検証と妥当性確認

前出のV&Vの観点から，第１段階の各工程を見直してみる。

①曖昧なユーザ要求は，フォーマルに書き直されたユーザ要求仕様によって妥当性を検証できる。
②フォーマルに表現された要求仕様は外部仕様によって検証できる。
③外部仕様は対応する内部仕様によって検証できる。
④内部仕様は細部に分割したサブシステム仕様によって検証できる。
⑤サブシステム仕様は，それを構成するプログラム仕様またはブロック仕様によって検証できる。

以上のように確かに各段階は順を追って，その妥当性を検証する手段を持っている。

第２段階の検証も従来からのプログラミング工程における検証の方法と同様に考えればよい。すなわち，プログラム仕様に対応するプログラムであることを検証するためのブラックボックス・テストや，いわゆるホワイトボックス・テスト（グラスボックス・テスト）によって妥当性が確認できる。第３段階については，ウォーターフォール型ライフサイクル・モデルでは単体テスト，結合テスト，さらには統合テストを実施するとしてきたが，何によってそのテストの妥当性を検証するのかが非常に曖昧であった。上流から流れてくるという想定では，そのテストにおける責任体制，結果の検証について明確な記述をすることは難しい。

５．フェーズ定義の見直し

システム開発がユーザの要求をコンピュータで実行するシステムに変換する工程であるとするならば図２に示した工程は，図１に示す工程と鏡面対称となっていなければならないはずである。図１に示した工程が要求仕様の段階的詳細化であるならば，図２に示す工程は段階的統合化として捉えなければならない。すなわち，従来からテスト工程とみなされていたものは，実はテスト工程と見るべきではなく，統合工程と見なければならない。テストはあくまでも統合の妥当性を検証する手段であって，目的ではない。こう考えると，図２に示す各工程のＶ＆Ｖは容易に説明することができる。すなわち，プログラム，ブロック，サブシステム，実行可能なコンピュータ・システム，運用可能なシステムのいずれの段階においても，それぞれの妥当性を検証するためのテスト仕様書が存在しており，そのテスト仕様書に対して妥当性確認（Validation）が行われる。そのためには図１に示す各仕様が作成されたときに同時にそのテスト仕様が作成されていなければならない。

システム開発は水が流れる様子に喩えられるべきではなく，あたかもＶ字谷を越えて対岸に辿り着く様子に対応させなければならない。すべての工程，ドキュメントについて対岸に対応する工程またはドキュメントが存在しており，そのそれぞれについて検証手段が確立していることが明確に説明できる。プログラム仕様，サブシステム仕様については，それぞれそのテスト仕様が記述されていて，当然その記述の中に受け入れ基準が明示される。いわゆるシステム・テストについては内部仕様の作成と同時にシステム・テスト仕様が作成され，外部仕様の作成時に総合テスト仕様が確定される。最終的な受け入れ検査（Acceptance Test）については，ユーザの要求仕様をフォーマルに記述したときにそのテスト仕様が作成されると考えればよい。それぞれのテスト仕様については，一段階詳細化した理由があるはずであり，その理由が確認できる内容であることが要求される。

このように考えてくると，テストという作業において何が確認されなければならないかが自ずから明確になってくる。システム構築の段階を構造的に整理することができるだけでなく，各段階においてなすべきこと，その責任体制，検証・妥当性確認の方法などを明示的に表すことがてきる。

６．まとめ

システム開発過程が一方的な流れではなく，Ｖ字谷を越える軌跡のように，ある工程は必ず対応する対岸に対応する工程が鏡面対称のように存在していると考えるべきであることを主張した。その結果として，テスト作業は分割された要求がじつさいのシステムに統合化される過程での１つの手段でしかないことが明かになり，その責任体制，テストすべき内容の想定も容易にできることが分かってきた。

ウォーターフォール型ライフサイクル・モデルの存在を否定するものではないが，品質保証という観点からすると，テスト工程において何に対して妥当性確認が行われるのかという点でウォーターフ

ォール型ライフサイクル・モデルにははっきりしない面があった。本提案のように考えると，その不明確な点は解決され，さらにその工程の実施に明確な指針を示すことができた。この提案を現実のシステム開発に当てはめて見たわけではないので，今後の実証が必要ではある。ただし，何人かのシステム開発に携わっている人々との接触では，その適用性が良いとの感触を得ている。

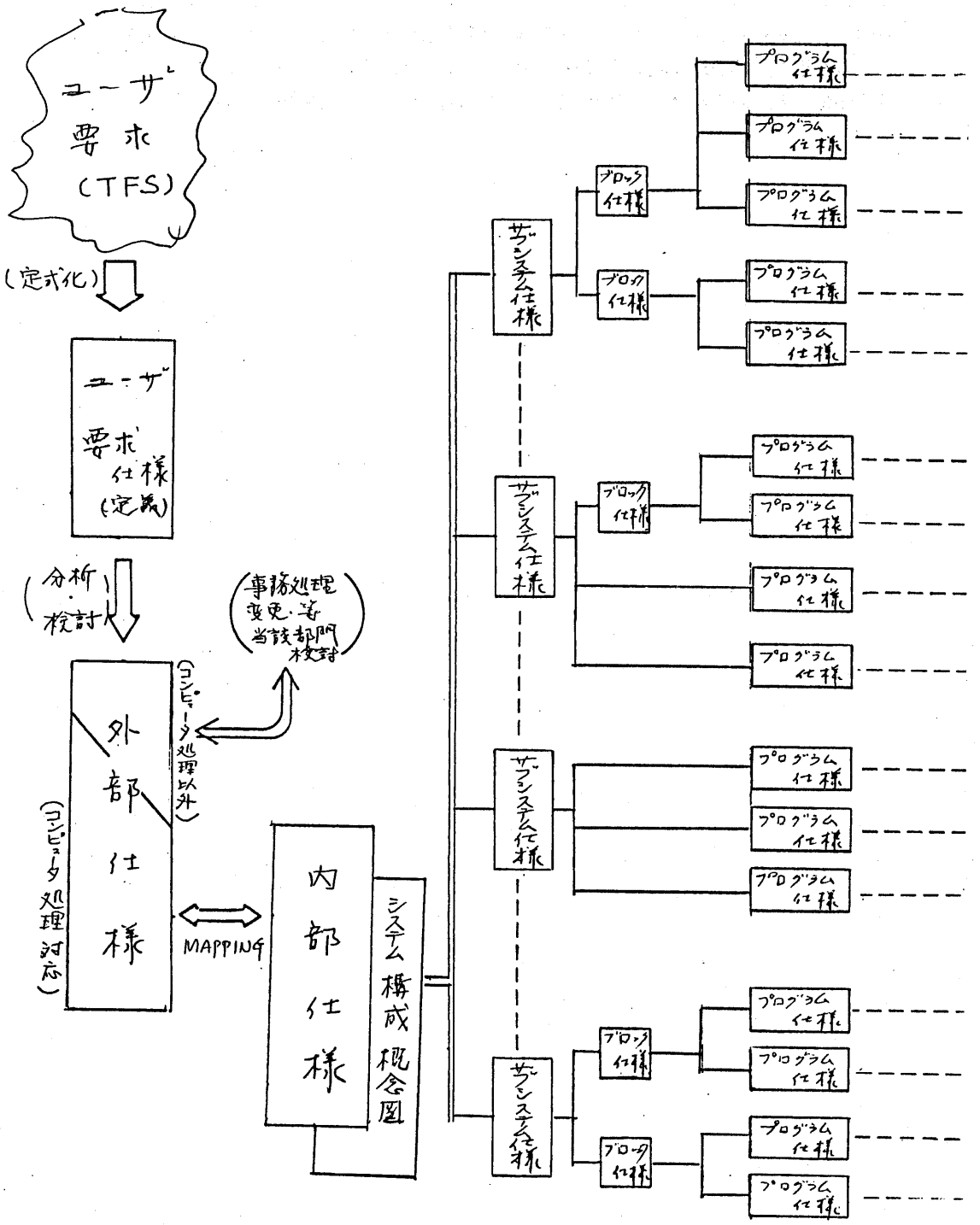


図1　第1段階　システムの要求の詳細化

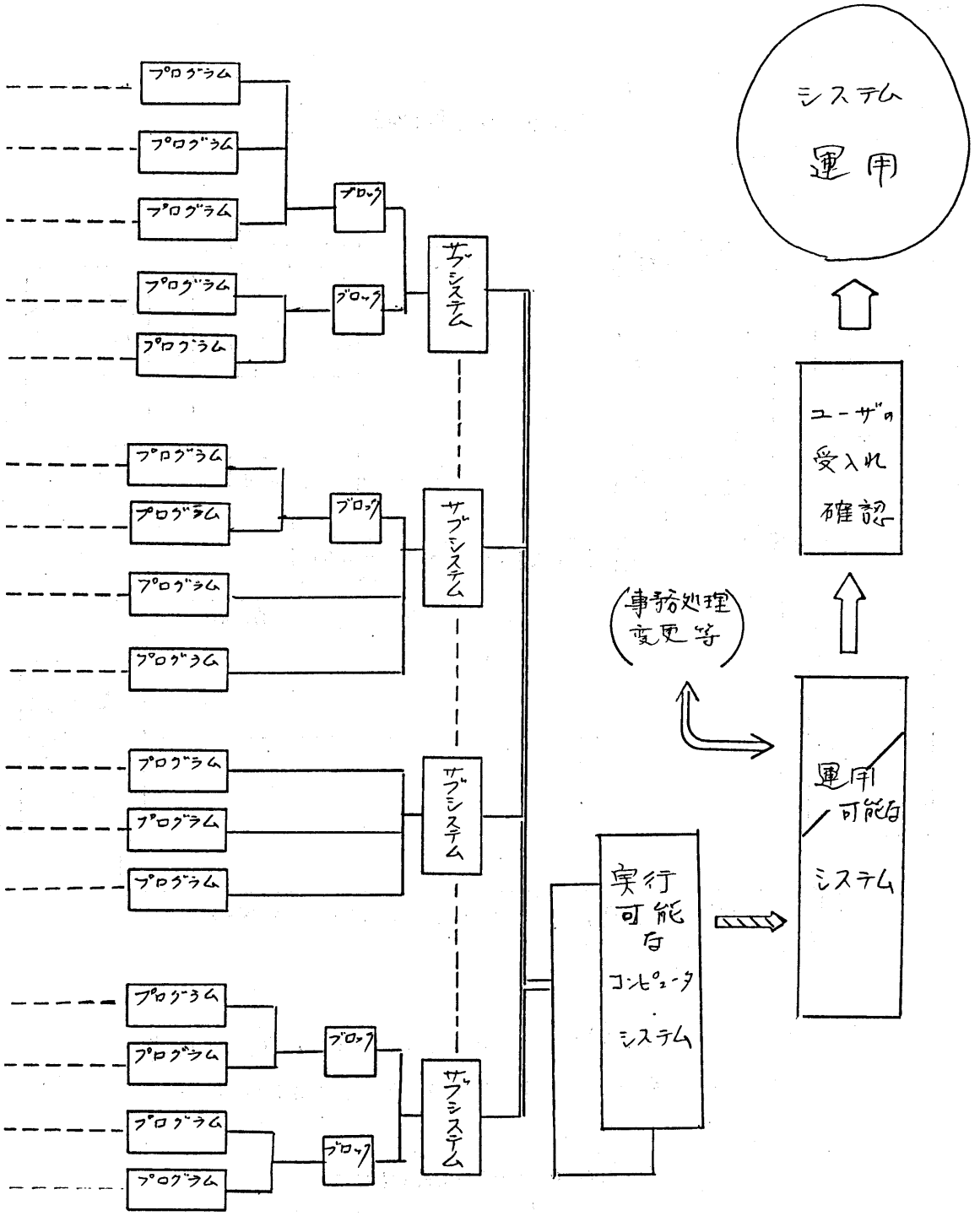
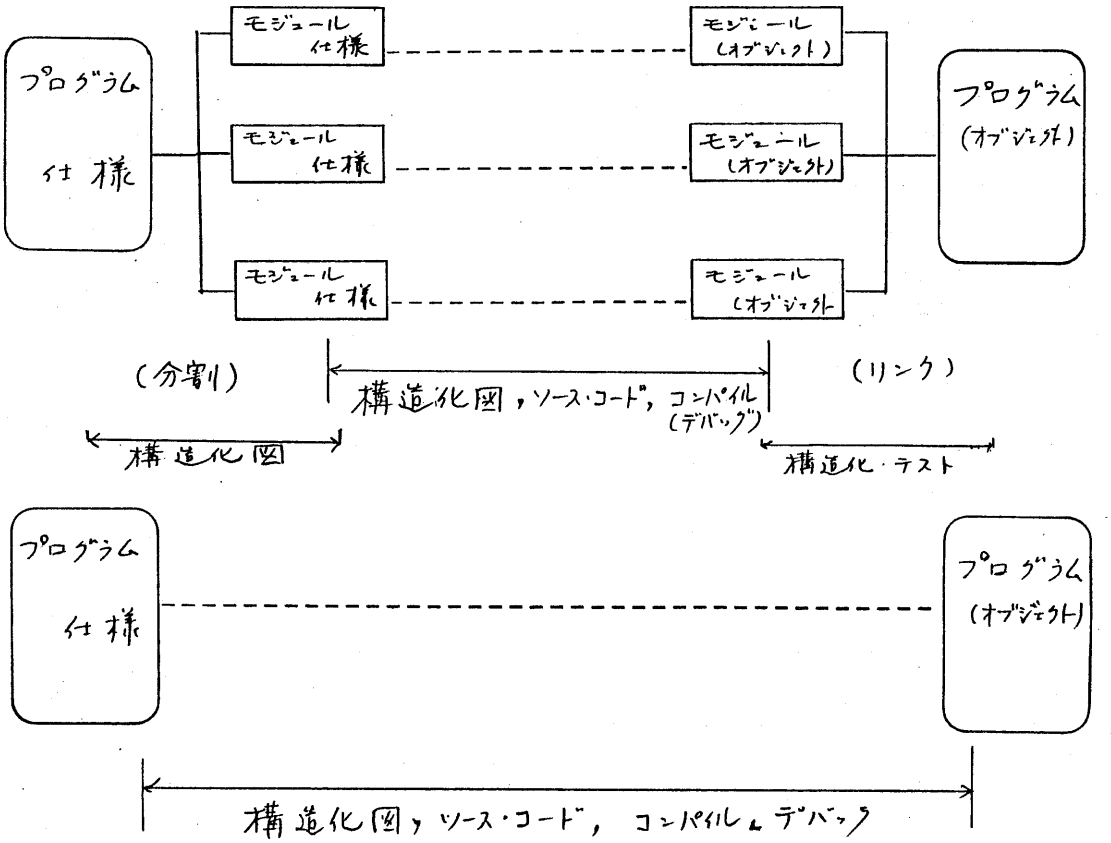


図2　第3段階　運用すべきシステムへの統合化

製　　造
（プログラム開発）



モジュール仕様にもモジュール・テスト仕様が必要
⇒ モジュールのブラック・ボックス・テストに相当

図3　第2段階　プログラム作成（製造工程）