ウォーターフォールモデルの起源に関する考察

ウォーターフォールに関する誤解を解く

小　椋　俊　秀

要　旨

現在ウォーターフォールモデルについて批判的な意見が多いが，その論議の際，その出所はRoyceであるという誤解のほか，そもそもRoyceの主張は現在のウォーターフォールとは異なり，しかもウォーターフォールという用語自体が使われていないということがよく指摘されている。では誰がウォーターフォールという用語を初めて使ったのか，これが現在に至るまで明らかにされていない。40年を超えるソフトウェア工学の歴史において，主要な開発モデルであるウォーターフォールモデルの出所を見出し，検証可能なうちに事実を事実として記録に残すことは価値のあることである。

今回，Royceの主張，現在のウォーターフォールモデルの定義を整理した上で，1970年代80年代のRoyceに関する著作を中心に調査した。その結果，現在のウォーターフォールモデルの起源はTRW社でのサイト防衛アプローチにあること，またウォーターフォールという用語が使われたのがBellとThayerの1976年の論文であること，そして，彼ら及びBoehmの記述が現在の誤解を招いたことが明らかになった。

1．はじめに

従来からそして今でも主要なソフトウェア開発モデルの一つであるウォーターフォールモデル[1)]に対する批判が近年高まっており，しばしばその際にウォーターフォールモデルの提唱者はRoyceであると誤って指摘されている。また，ウォーターフォールモデルの提唱者はRoyceではなく，Royce［1970］の主張はウォーターフォールモデルとは別物だと指摘している場合でも，では誰が何を指してウォーターフォールモデルと名づけたのかということを明らかにはしていない。（Larman［2003］）

ソフトウェア工学の歴史は1968年のNATOの会議から始まるといわれているが（玉井［2008］），以来既に40年を超えている。一般に学問を進めるためにはその学問領域の歴史を知ることは大事なことである。主要な開発モデルであるウォーターフォールモデルの出所が不明であり，その出所を見出し，検証可能なうちに事実を事実として記録に残すことは価値のあることである。

本節ではウォーターフォールモデルに関する先行研究を調査したうえで，一般に考えられている「Royceがウォーターフォールモデルの提唱者である」という誤解を解くための調査研究対象を設定する。

先行研究の紹介としてまずLarman［2003］を取り上げる。Larman［2003］はウォーターフォールを次のような開発ステップを意味するといっている。

① 要求を事前に詳細に定義する。

② 「設計」を定義する（ソフトウェアおよびハードウェアの要素をテキスト及び図で記述する）。

③ システムを実装する。

④ コンポーネントを結合し，テストする。

（Larman［2003］：訳書p.70）

1）文献ではときおりウォータフォールという記述があるが，本稿ではウォーターフォールを使う。ただし，引用元がそうでない場合はそれに従う。

そして，これらのことを一度で終わらすものだといっている。

ウォーターフォールモデルでの開発は，工程の進捗管理がしやすいなどの利点はあるが，基本計画時点で全てのユーザニーズを見通すことは困難であり，納期間近の下流工程にならないとユーザの確認が取れなく，ユーザからの修正要望がでると結局上流工程に戻らざるをえなくなる。

そのような観点からウォーターフォールモデルは使えないと，Larmanをはじめとしてソフトウェア開発の世界の著名人も非難している。Brooksは，「ウォーターフォールモデルは間違っており有害である。私たちはこのモデルから脱却しなければならない」と書き（Brooks［2010］：訳書，p.34），McBreenは，「ウォーターフォール・アプローチは，危険かつ問題をはらんだ，企業における風土病」といっている。（McBreen［2002］：訳書，p.125）

このウォーターフォールモデルを提唱した元祖の論文はRoyce［1970］であるというのが定説なっていると玉井［2008］にあるが，事実いろいろなインターネットのサイトや書籍にそのように紹介されている。Andriole and Freeman［1993］には，ウォーターフォールモデルはRoyce［1970］で最初に紹介されBoehmがそれを有名したとある。

その玉井［2008］やLarman［2003］でも，Royce［1970］の主張は上述のウォーターフォールモデルとは違うものだと指摘しているし，Brooksは，Brooks自身が若い頃Royce［1970］の主張を古典的なウォーターフォールモデルを指しているものと思い，後にそれが間違いだったと気づいたと述べている。（Brooks［1995］：訳書，p.258）

しかしながら，彼らにしても，では誰が何を指してウォーターフォールと名づけたのかということには触れていない。

なお，Royceを提唱者としていない文献として，菅野による次の記述があることを確認した。（菅野［1996］，p.34）

ウォータフォール・モデルの歴史は古く，1968年にさかのぼる。この年，西ドイツのガルミッシュにおいて，NATO後援の国際会議が開かれ，ソフト

ウェア開発をそれまでの職人芸的な作成法から，近代的な工業製品としての作成方法に変えることが検討された。その結果，ソフトウェアを開発する段階をいくつかの工程（フェーズ）に分け，各工程の終了を意味するドキュメントを作成して，進捗を管理するとともに作成したドキュメントを検査することによって早いうちから品質の作り込みをしようとする考えが提言された。これがウォータフォール・モデルである。このときのモデルはベームによって修正され，今日世界中で使われるものになった。[2)]

以上，見てきたようにウォーターフォールモデルの提唱者はRoyceではないとの指摘があるにもかかわらず，「Royceがウォーターフォールモデルの提唱者である」という誤解が根強くある。

そこで，この誤解を解くためにRoyceではなく誰がウォーターフォールモデルという用語を最初に使ったのかを調査した。本稿はその調査内容と結果を示すものである。[3)]

今回の調査研究の対象は，

1）Royceの主張はどのようなものか

2）現在のウォーターフォールモデルの定義とそれに対する批判はどのようなものか

3）誰が最初にウォーターフォールモデルという用語を使ったのか

以上の３点である。

本稿では，次節より，２．Royceの主張，３．ウォーターフォールモデルの実態，４．ウォーターフォールモデルの起源，という構成で調査内容を整理し，それを踏まえ，５．考察，６．結論で調査結果を発表する。

2）原文のまま引用

3）本調査研究は2012年の小樽商科大学大学院博士課程の講義で，担当教員である持田泰昭教授から頂いたアイデアを基に進めたものである。本稿執筆の機会とご指導をいただいた持田教授に深甚な感謝の意を表し上げます。もちろん，内容に関する誤りは筆者に帰すものである。

2．Royceの主張

前節で指摘したようにRoyce［1970］は，ウォーターフォールモデル論議では頻繁に引用されている。

Royceの息子，Walker・Royceも父の論文内容をある程度詳しく紹介している（Royce, Walker［1998］：訳書，pp.6-18）。しかしながら，今後の「誰がウォーターフォールという用語を最初に使ったのか」などの検討のために必要な記述は充分ではない。また，今後の検討にはRoyceが論文で使っている図との比較が役に立つ。本節では，今後の検討に必要な情報を提供するという観点から，Royce［1970］の内容を図を中心に紹介し，Royceの主張を整理する。

以下，Royce［1970］を翻訳し，要約したものである。

タイトル「大規模ソフトウェアシステム開発の管理」

本論文は，大規模ソフトウェア開発についての個人的な見解である。

内部使用のための小規模なプログラムには分析とコーディングの２つのステップが有用である。

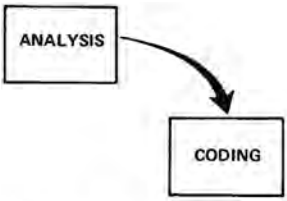

Figure 1. Implementation steps to deliver a small computer program for internal operations.

図１　内部使用のための小規模コンピュータプログラム提供のための実行ステップ。　出所 Royce［1970］，p.328

より大規模なソフトウェアシステムには，図２のように分析とコーディング以外のステップが必要になる。

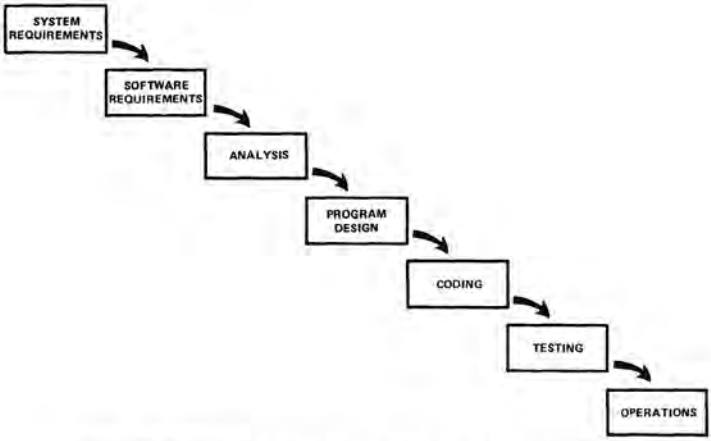

Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

図２　顧客向けの大規模コンピュータプログラム開発のための実行ステップ。　出所 Royce［1970］, p.329

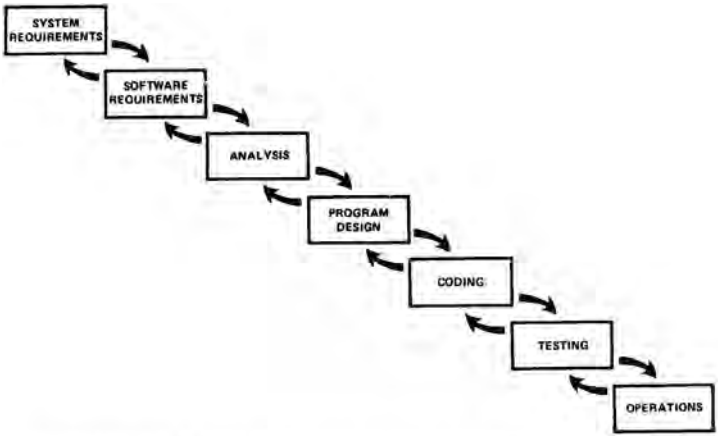

Figure 3. Hopefully, the iterative interaction between the various phases is confined to successive steps.

図３　うまくいけば，個々のフェーズ間の反復相互関係は連続した段階に限られる。　出所 Royce［1970］, p.330

図３はこのスキームのための連続した開発フェーズ間の反復関係を示す。

我々の持っているものは，救出・保護可能な前段階の作業の範囲を最大化するのに役立つ効果的な後戻りである。

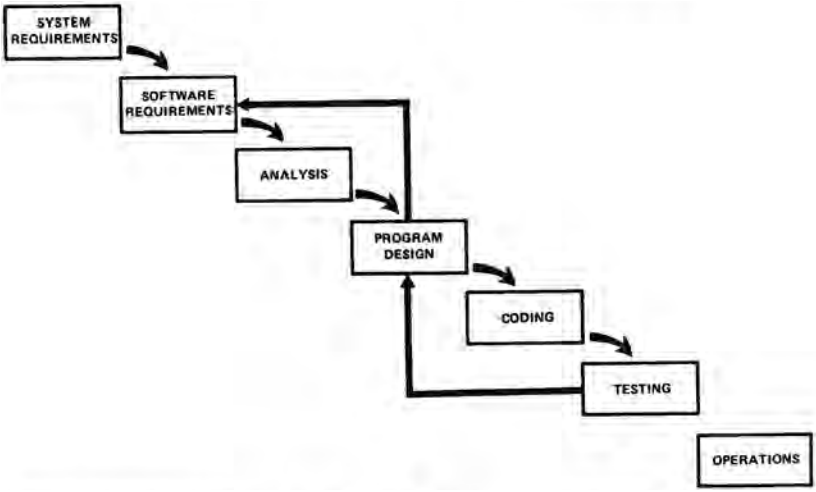

Figure 4. Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.

図４　残念ながら示された工程では，設計の後戻りは連続したステップにけっして留められていない。　出所Royce［1970］，p.330

図４で次の問題が示される。開発サイクルの最後に行われるテストにおいて初めて，同期，記憶装置，入出力変換等の分析の結果として識別されたものが経験される。これらは要求の修正または設計の修正のどちらかが必要となり，それは予算やスケジュールの超過につながる。

この基本的なアプローチに追加されるべきほとんどの開発リスクを排除する５つの追加項目を発表する。

STEP １　プログラム設計が最初にくる　図５

事前のプログラム設計をソフトウェア要求フェーズと分析フェーズの間に挿入する。

分析者やプログラマではなく，プログラム設計者により設計工程を開始する。

間違う危険を冒してでも，データ処理モードを設計・定義・配置する。

理解でき，有益で，最新の概要ドキュメントを書く。

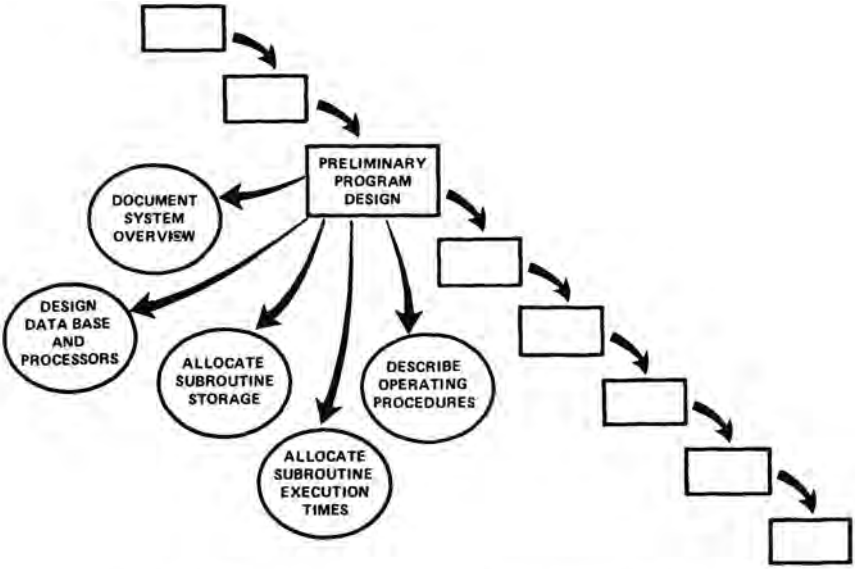

Figure 5. Step 1: Insure that a preliminary program design is complete before analysis begins.

図５　STEP １：分析が始まる前に事前のプログラム設計が完成されることを保証せよ。　　出所 Royce［1970］，p.331

STEP ２　設計の文書化　図６

「どれくらいのドキュメント？」という問には「とても多くの」と答える。

何故そんなに沢山のドキュメントなのか？

1）各設計者は，インターフェース設計者や，彼の管理者，そしてたぶん顧客ともやりとりを行わなければならない。

2）ソフトウェア開発の初期段階ではドキュメントは仕様書であり設計である。コーディングが始まるまで，ドキュメント，仕様書，設計は一つのものを意味する。

3）良いドキュメントの実際的な金銭的価値は開発工程の下流で始まる。

・図６では６つのドキュメントが作成されることを示す。

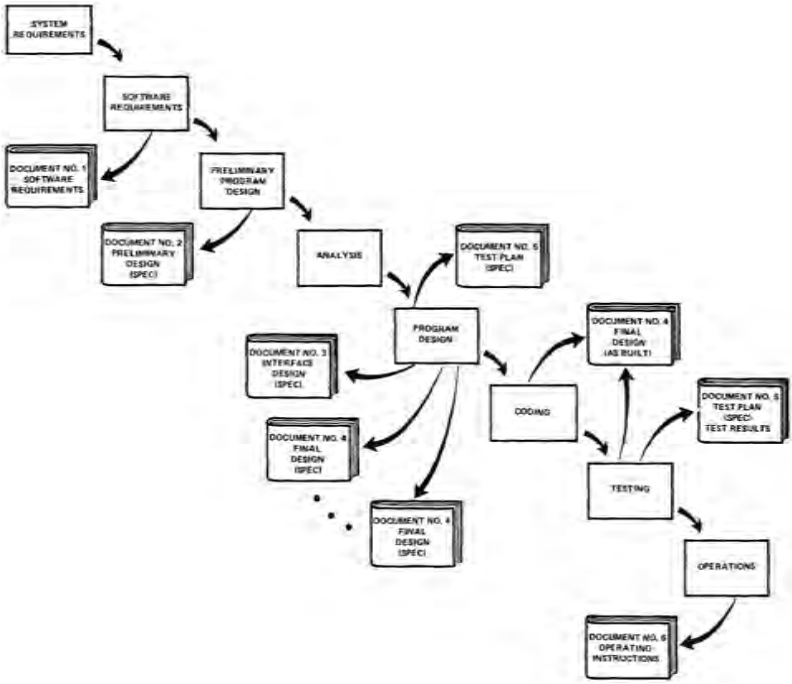

Figure 6. Step 2: Insure that documentation is current and complete − at least six uniquely different documents are required.

図６　STEP ２：ドキュメントが最新かつ完成であると保証せよ。少なくとも６つの異なる独自のドキュメントが要求される。出所 Royce［1970］，p.333

STEP ３　２度行う　図７

もし問題のコンピュータプログラムが初めて開発されたものなら，運用配置の為に顧客に最終的に納入されるバージョンは，危険な設計や運用の領域に関する範疇においては，実は２番目のバージョンとなるよう，準備しなさい。

図７では，この事がシミュレーションによってどのようにもたらされるのかを図示している。

全体の労力が30ヶ月かかるなら，試行モデルのこの初期開発の期間は10ヶ月間となるかもしれない。

この場合，関係する人員の一部には非常に特別な種類の広大な能力が要求される。

彼らは，分析，コーディングそしてプログラム設計に対する直感的な感覚を持たなければならない。

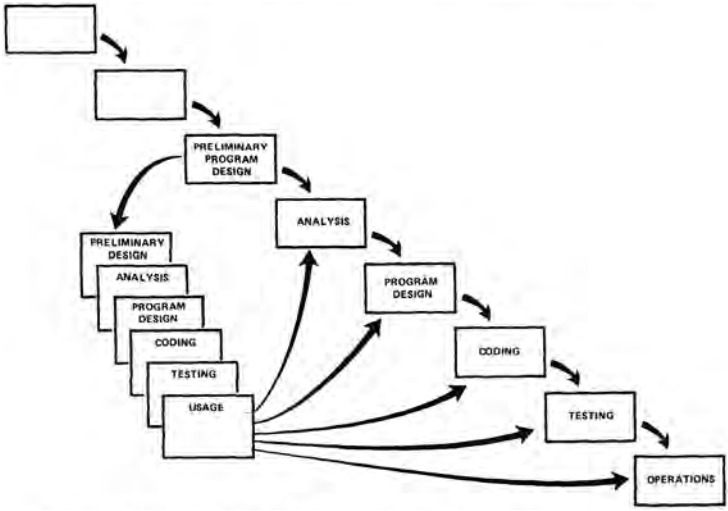

Figure 7. Step 3: Attempt to do the job twice － the first result provides an early simulation of the final product.

図７　STEP ３：作業を二度やることを試みよ　最初の結果は最終製品の初期シミュレーションを提供する。　出所 Royce［1970］，p.334

STEP 4　テストを計画，管理，モニタする

プロジェクト資源の最大の使用者はテストフェーズである。

前述した３つの推奨, 分析とコーディングの前にプログラムを設計すること，その完璧なドキュメントを作ること，パイロットモデルを作ること，これらはすべて，全体的なテストフェーズに入る前に問題をあらわにし解決することを目的としている。

図８はテストへのいくつかの追加の見地をあげている。

１）テストはテスト専門家に。

２）多くの検査は目視検査で簡単に発見できる。

３）全ての論理パスをテストせよ。

４）単純なエラーを取り除いてからテストエリアへ。

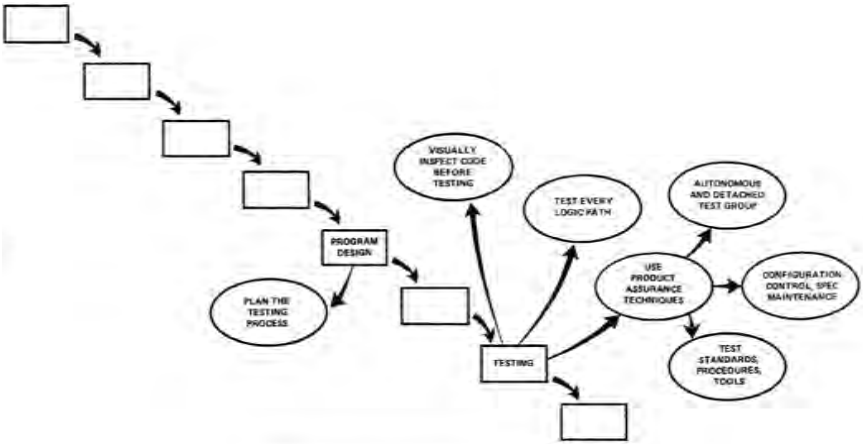

Figure 8. Step 4: Plan, control, and monitor computer program testing.

図８　STEP ４：コンピュータプログラムテストを計画, 管理, モニタせよ。

出所 Royce［1970］, p.336

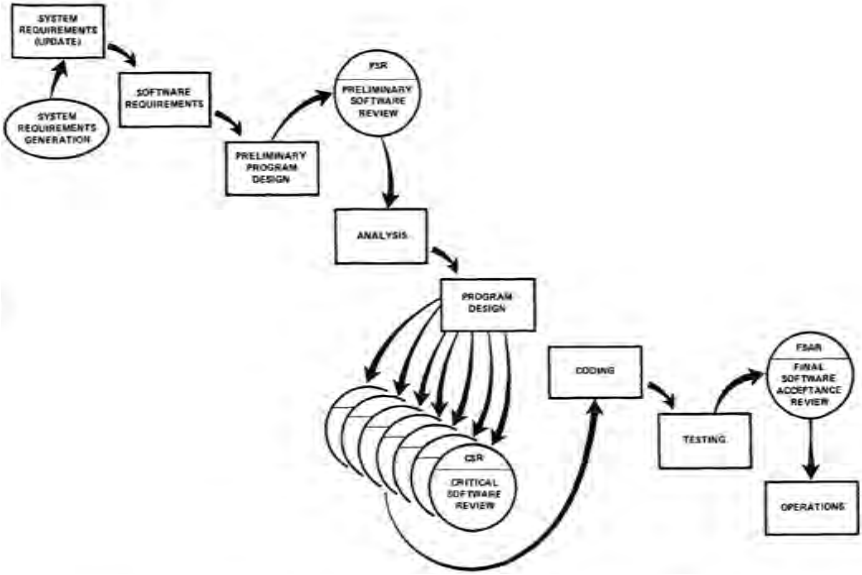

Figure 9. Step 5: Involve the customer − the involvement should be formal, in-depth, and continuing.

図９　STEP ５：顧客を巻き込め－巻き込みは公式に，徹底的に，そして継続的に。　出所 Royce［1970］，p.337

STEP ５　顧客を巻き込む

最終納品前の早い段階で顧客を巻き込むことは重要である。

図９は，顧客の識見・判断・関与が開発努力を元気づける要求定義後の３つのポイントを示す。

要約

図10は，リスキーな開発プロセスを望まれる製品を提供するそれに変換するのに必要と感じる５つの追加項目をまとめたものである。

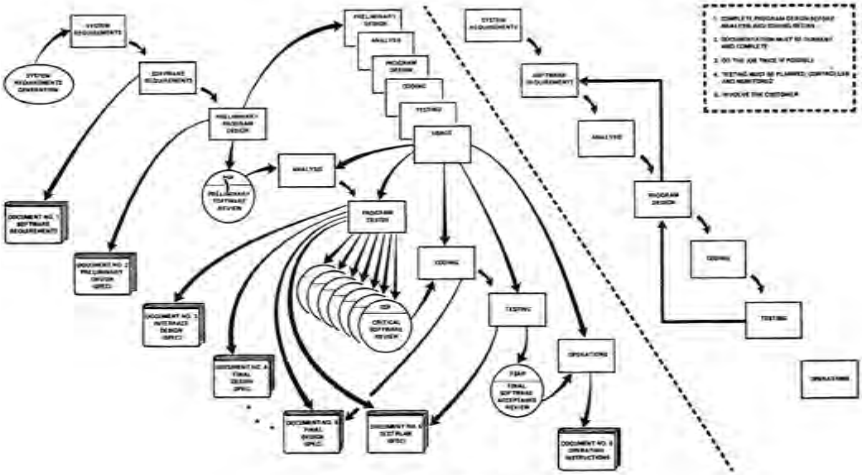

Figure 10. Summary

図10　要約　　出所 Royce［1970］，p.338

以上，Royce論文の内容を紹介したが，図の流れを中心に簡単にまとめると以下の内容であった。

図１の分析とコーディングだけでは大規模ソフトウェア開発はできず，図２の工程が必要になること，そしてそれは上から下へ流れるだけでなく，前工程の修正のため図３のように後戻りが有効であること。さらに開発リスクを排除するためには，図５のように分析やコーディングの前に事前のプログラム設計を終わらせ，図６のようにドキュメントを最新かつ完璧にし，図７のように納入されるバージョンが２番目になるように作業を２度やり，図８のようにテストを計画，管理，モニタし，図９のように顧客を巻き込むという５つの追加項目がいる。これらを要約し図にしたのが図10である。

なお，次節以降，Royceの主張する開発モデルと現在使われているウォーターフォールモデルとを区別するため，この論文では便宜的にRoyceの主張する開発モデルをRoyceモデルと呼ぶことにする。

3．ウォーターフォールモデルの実態

前節でRoyceの主張するRoyceモデルを見てきたが，そこにはウォーターフォールという単語は出てこなかった。それでは現在，どのような意味でウォーターフォールモデルという用語が使われていて，それはどう評価されているのだろうか。

情報処理推進機構（Information-technology Promotion Agency Japan，以下IPAと略）が発行するソフトウェア全般に関する個々の作業内容，用語の意味などを示した共通フレーム2007には，開発モデルやライフサイクルモデルという用語はあるもののウォーターフォールモデルという用語は定義されていない。共通フレームは開発モデルに依存しないため，共通フレームの実際の適用にあたっては，開発モデルを選択し，共通フレームで定義された作業項目をそれにマッピングせよとあり，ウォーターフォールモデルへの適用例では，ウォータフォールモデルを次のように紹介している。

「ウォータフォールモデルは，それぞれの作業ステップを一回だけ通して行うというのが特徴である。利用者のニーズの決定からはじまって，要求の定義，システムの設計，システムの作成，テスト，修正，そして導入，運用に至る」（IPA［2009］a, pp.277-278）

また，そのIPAが主催する情報処理技術者試験において，ソフトウェア開発モデルの説明を問う問題で，「ウォータフォールモデルは，開発を上流から下流に一方向に進めるモデルであり，開発効率を高めるには，各工程内でのレビューやテストによって品質を確保し，前への工程への逆戻りが起こらないようにする」が正解というのが出題されている。（IPA［2009］b, p.15）

その情報処理技術者試験の受験参考書類でも，そこで説明するウォーターフォールモデルとはやはり上述のIPA試験問題の内容である[4)]が，明確な定義

4）IPAの試験カリキュラムは2009年春に改定され，それまであったウォーターフォールモデルという用語が削除された。

づけがされていないため，受験参考書によって記述もまちまちであり，上述の共通フレーム2007にあるウォーターフォールモデルの設計プロセスを外部設計，内部設計，プログラム設計に分けて，「基本計画（現状分析，要求分析含む），外部設計，内部設計，プログラム作成，テスト，運用・保守という開発工程の各段階を上流から下流へ逆流することなく進める」と説明するものも複数ある。（TAC［2008］，平井［2003］）

上記を図示したのが図11である。

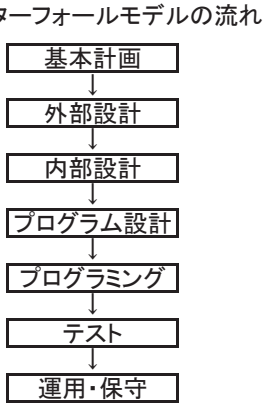

図 11　ウォータフォールモデルの流れ

出所：筆者作成

この図11は次のように説明される。

基本計画では要求分析，現状分析を行い，システムの概要，概算の開発予算と期間などを明らかにする。外部設計では画面レイアウト，帳票レイアウトなど利用者がどのようなシステムとなるのかを理解できるようにし，利用者の確認をとる。内部設計ではそのシステムをプログラム単位に機能分割をする。プログラム設計ではプログラムをモジュールに分割し，そのモジュールごとにプログラミングを行う。テストについてはさらに次のように説明している。

プログラミングに対応するのがモジュール単位で行う単体テスト，プログラ

ム設計に対応するのがモジュールを結合して行うプログラムテスト，内部設計に対応するのがプログラム間の連携を確認するシステムテストであり，またこのシステムテストは,外部設計にも対応した利用者を巻き込んだテストとなる。そして，基本設計に対応するのが運用テストであり，その後，実際の運用・保守となる。

これを図にしたのが図12であり，一般にＶ字モデルと呼ばれる。

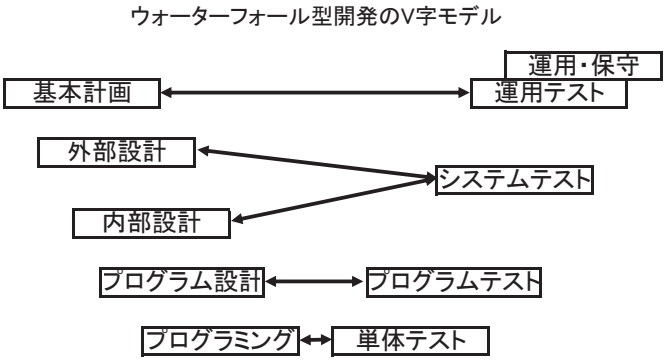

図 12　ウォーターフォール型開発のＶ字モデル

出所：筆者作成

ウォーターフォールモデルは，その明確な定義づけがされていないため，同じ情報処理技術者試験向けの受験参考書であってもその記述が多様である。これは受験参考書だけの話ではなく,研究者,実務者にとっても同じことであり,ウォーターフォールモデルの定義は個々人の解釈により差異がある。

そのような状況下，それでもウォーターフォールが推奨され続けている理由として，Larman［2003］はウォーターフォールモデルの失敗例を挙げながら，以下のことを挙げている。

1．ロイスの論文を読んだ人がほとんどいなく，単純な一度きりのライフサイクルだと思われていること。

2．ウォーターフォールが最も簡単なプロジェクト以外ではうまくいかないことを理解している人がほとんどいないこと。

3．一度で終わらせるウォーターフォールは，説明したり記憶したりするのが簡単であること。

4．ソフトウェアプロジェクトが新製品開発のパラダイムでなく，予測可能な製造のパラダイムと結び付けられて考えられてきたこと。

5．秩序正しく，予測が可能で，説明がつきやすく，測定可能なプロセスであり，文書を中心とした単純なマイルストーンが存在するという幻想をウォーターフォールが与えたため。

6．ウォーターフォールの価値や事前に大規模な仕様を作成するという目標が，要求工学では適切，理想的であるとして推奨され続けているため。

7．ソフトウェアエンジニアリング研究所の能力成熟度モデルがソフトウェアプロセスエンジニアに影響を与え，ゲートで制御された文書駆動のウォーターフォールプラクティスが使われるようになったため。

8．初期のプロジェクトマネジメントのための知識体系（Project Management Body of Knowledge : PMBOK）がウォーターフォールとの相性が良かったため。

（Larman［2003］：訳書，pp.129-132）[5)]

Larman［2003］はこのように述べ，反復型開発（iterative development），進化型開発（incremental development）であるアジャイル開発（agile development）を推奨している。[6)]

5）原文を要約して引用

6）先に紹介したIPAの2009年のカリキュラム改訂であるが，ウォーターフォールモデルに取って代わったのがアジャイル開発である。

4．ウォーターフォールモデルの起源

Royceの主張するRoyceモデルと現在使われているウォーターフォールモデルをそれぞれ見てきたが，Royceモデルはウォーターフォールモデルを批判したものであった。それでは，なぜ，Royceモデルがウォーターフォールモデルの起源といわれ，Royceモデルでは一切使われていなかったウォーターフォールという名で呼ばれるようになったのであろうか。

先行研究調査で紹介した菅野［1996］では，1968年の西ドイツ，ガルミッシュで開かれたNATO後援の国際会議で提言された，ソフトウェアを開発する段階をいくつかの工程（フェーズ）に分け，各工程の終了を意味するドキュメントを作成して，進捗を管理するとともに作成したドキュメントを検査することによって早いうちから品質の作り込みをしようとする考えがウォーターフォールモデルの起源であるとの指摘があった。

この1968年のNATO国際会議では，NaurとRandellによる議事録によると「ソフトウェア危機」についての議論があった。（Naur and Randell［1969］）

磯田［1996］によれば，「ソフトウェア危機」という言葉がいい出されたのは1960年代後半のようだと述べ，当時の様子を次のように語っている。

「当時，OS/360やMulticsといった大規模システムが開発されたが，技術が未熟だったためいずれも手痛い失敗を経験した。劣悪な品質に悩み，工期と経費が予想を大幅に超過したのだ。」

「このような状況の中で，人々がソフトウェアは『危機』に直面していると考えたのももっともなことだった。」

「これを解決するためにはソフトウェアも他の分野と同様に工学的手法に基づいて開発すべきとの考えのもとに，NATO科学委員会主催のソフトウェア会議が1968年にドイツのガーミッシュで開催された。この会議は『ソフトウェア工学』という言葉を初めて公式に使用した会議として名高い。」

またLarman［2003］も米国の軍事事情とソフトウェア開発の関連を詳しく述べている。

このような背景と先行研究調査より，ウォーターフォールモデルの出所は，１．軍事用ソフトウェアに関連する，２．Boehmがキーマンである，という二つの推測を立て，まずはBoehmに関する著作を整理し，彼のウォーターフォールモデルに関する著作を時代の降順に調査する方針を立てた。さらに，推測が外れた場合に備えて，1970年代及び80年代のACM，IBM，IEEEなどが発行する米国コンピュータ関連雑誌に時間の許す限り目を通すことにした。

まず，Larman［2003］よりBoehmに関する記述を見てみる。

「ベームは，1950年代半ばからソフトウェア開発に携わり，TRW社（経験の豊かな大規模システムの受託会社）のチーフサイエンティストとして働いていた。」と紹介され，スパイラルモデルの提唱者であり，スパイラルモデルを提唱した論文で，ウォーターフォールではうまくいかないことに言及していることが紹介されている。（Larman［2003］：訳書，pp.118-119）

Winograd［1996］では，ソフトウェア・ライフサイクル・モデルの紹介で，ウォーターフォールモデルとスパイラルモデルがどうやって生み出されたのかは，Boehm［1976］を参照するようにと書かれている。（Winograd［1996］：訳書，p.103）

また，Boehmに関してインターネットで調査したところ，出典は明らかにされてないが，Boehmがウォーターフォールという用語を最初に使ったことを否定したという記述のあるウェブサイトを発見した。（Weisert［2003］）

Boehm自身の著作に取りかかるにあたりまず，Larman［2003］の紹介するスパイラルモデルに関する論文をスタートとする。入手した論文はBoehm［1988］である。[7)]

Boehm［1988］では，Royce［1970］を引用し，Royceモデルをウォーターフォールモデルと呼び，次の図を掲げ，Royceモデルを批判している。

7）Larman［2003］によるとこの初出は1985年とある。

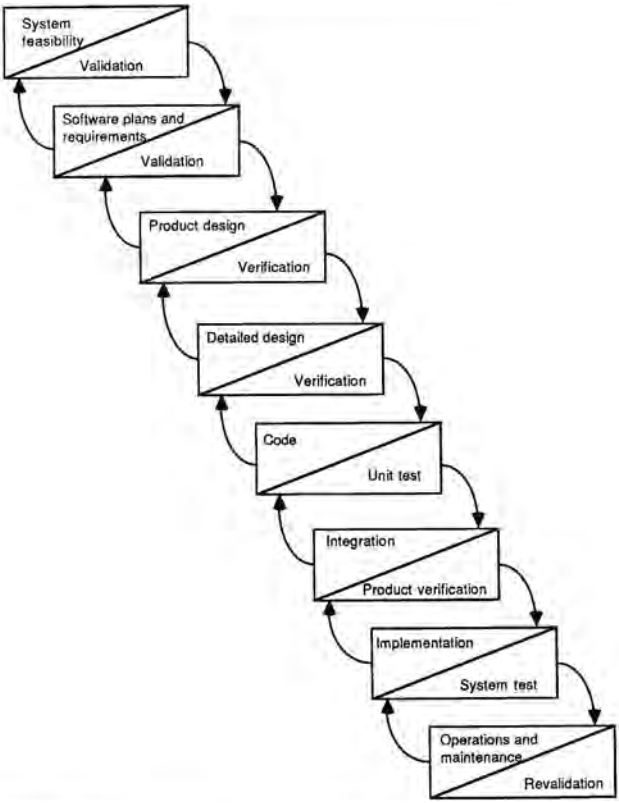

Figure 1. The waterfall model of the software life cycle.

図13　ソフトウェアライフサイクルのウォーターフォールモデル。

出所：Boehm［1988］，p.62

この図を２節で紹介した図３と見比べてみると，形は図３に近いがステップの数，その内容とも異なっている。さらに何のことわりもなくRoyceモデルをウォーターフォールモデルと呼んでいる。

Boehm［1988］のこの図13の解説部分には，Boehm［1981］が引用されている。

Boehm［1981］の４章２節のタイトルは「4.2 THE WATERFALL MODEL」

であり，そこには，前掲の図13と全く同じ図が掲載されている。（図14）

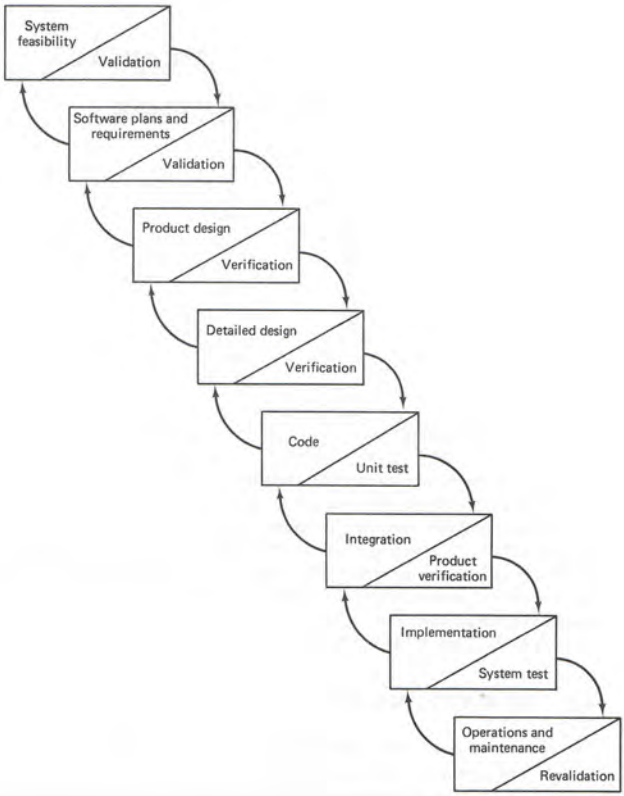

FIGURE 4-1　The waterfall model of the software life-cycle

図14　ソフトウェアライフサイクルのウォーターフォールモデル

出所：Boehm［1981］，p.36

Boehmは図14をウォーターフォールモデルだといい，そのオリジナルバージョンはRoyce［1970］だと述べ，1960年代後半の米国空軍などの影響を受けていると述べている。そしてさらに，図14を詳しく解説している。

Boehm［1981］はソフトウェアの開発工数や期間を見積もる手法である

COCOMOが発表された本であり，多くの人の目に触れたはずであり，ウォーターフォールモデルの提唱者はRoyceであるとの誤解を広めたことになる。

さらに，Boehmの著作を遡って調査することにした。

Winograd［1996］で紹介されているBoehm［1976］では，Royce［1970］は参照されてなく，ウォーターフォールの言葉もなかった。

ここで図示されている図15は前掲の図14と同じように各ステップが前後のステップと双方向の矢印で結ばれているが，図14とはステップ数とその内容が違う。図15のタイトルはソフトウェアライフサイクルである。

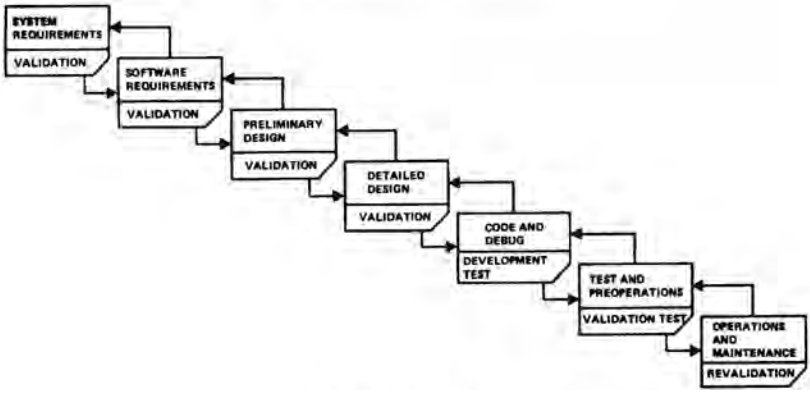

Fig. 2. Software life cycle.

図15　ソフトウェアライフサイクル。　　出所：Boehm［1976］，p.1227

この図15と同内容の図が，Wolverton［1974］にあり，それが図16である。BoehmもこのWolvertonもTRW社に属しており，それはRoyceもそうである。そこで図15，16で表現されている開発ステップはTRW社で共有されているのではないかと考え，著者がTRW社に所属し，そして図15，16のような開発ステップ，ソフトウェアライフサイクルについて図示している論文を探してみた。

そして発見したのがBell and Thayer［1976］である。この論文で彼らは，「Royceは滝（ウォーターフォール）のような概念の開発行為を紹介した」と

述べ，図17を掲げている。

なお，原著の該当部分の英語表記は以下のとおりである。“”でくくられた“waterfall”は原典そのままである。

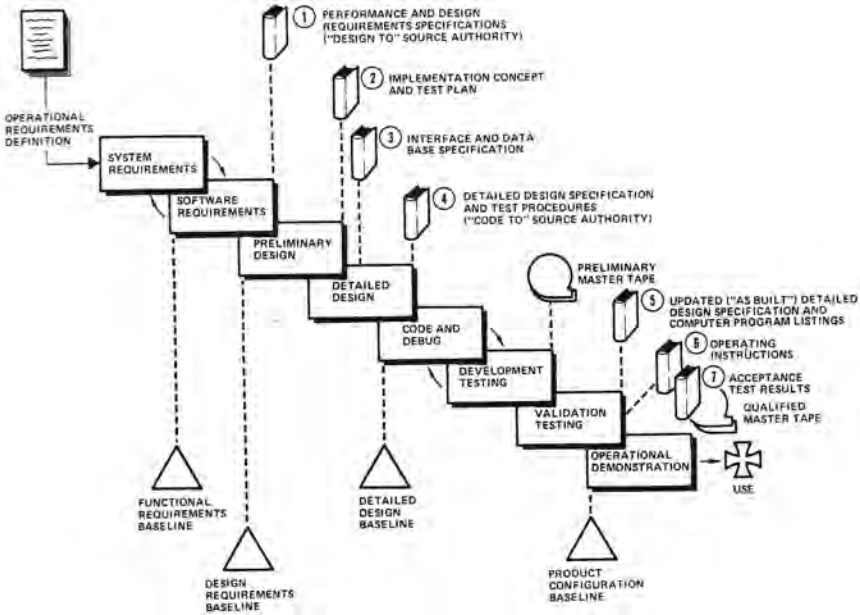

Fig. 1. Typical custom software development and test steps, showing seven unique documents.

図16　７つの独自のドキュメントを示した典型的な顧客向けのソフトウェア開発とテストのステップ。　出所：Wolverton［1974］，p.617

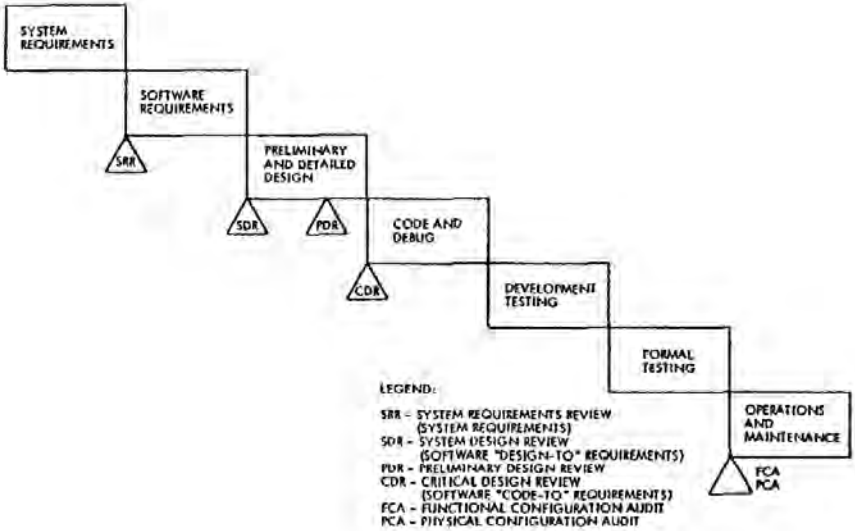

Figure 1. Development Phases of the System Development Cycle

図17　システム開発サイクルの開発段階

出所：Bell and Thayer［1976］，p.67

The same top-down approach to a series of requirements statements is explained, without the specialized military jargon, in an excellent paper by Royce; he introduced the concept of the "waterfall" of development activities. In this approach software is developed in the disciplined sequence of activities shown in Figure 1.

（Bell and Thayer［1976］，p.62）

この発見の後，これ以前でウォーターフォールという用語を使った論文がないか可能な限り調査をしたが発見できなかった。となると，そのような論文が発見されない限り，ウォーターフォールという用語を最初に使ったのはBellとThayerであったということになる。

しかし何故BellとThayerはRoyceモデルをウォーターフォールと呼んだのだろうか。

Willams［1975］では，前掲の図16と同様の図を「ソフトウェア開発アプローチ」として掲げ，こう説明している。「ソフトウェア開発のサイト防衛アプローチはより形式的な段階的製造過程に作りあげられた。」（図18）

サイト防衛アプローチというのは１億ドルかけた弾道ミサイルによる防衛のためのTRW/軍サイト防衛ソフトウェアプロジェクトで用いられたものである。（Larman［2003］：訳書，p.101）

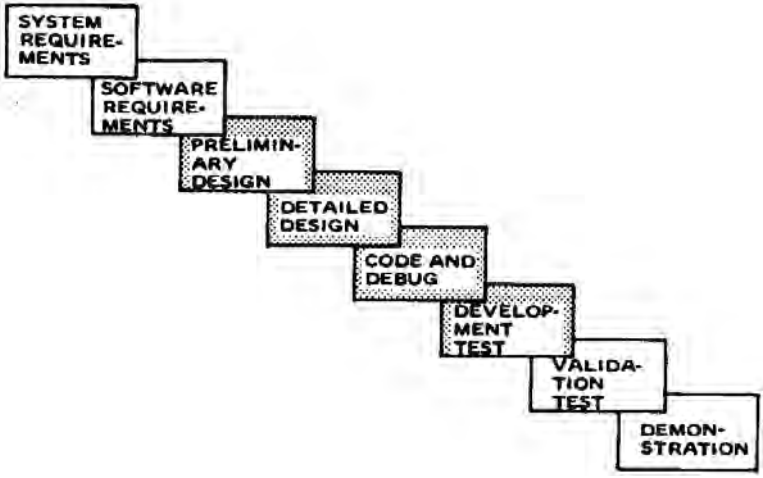

Figure 1. Software Development Approach

図18　ソフトウェア開発アプローチ　　出所：Willams［1975］，p.3

この論文ではウォーターフォールという用語はでてこないが，Royceの論文は参照されており，開発リスクを排除するために基本的なアプローチに加えなければならないものとして，Royceの５つステップを紹介している。

これを勘案すると，BellとThayerがウォーターフォールといったのはRoyceモデルのことをいったのではなく，この図18で表現されるTRW社で取り組ん

でいたサイト防衛アプローチを指していたと考えられる。そして，BellとThayerがRoyce［1970］をとりあげたのは，このトップダウンアプローチを軍事専門用語なしで説明した素晴らしい論文だったからである。

5．考　　察

「1．はじめに」で述べたように，本論文での調査研究の対象は，「Royceがウォーターフォールモデルの提唱者である」という誤解を解くために，

1）Royceの主張はどのようなものか

2）現在のウォーターフォールモデルの定義とそれに対する批判はどのようなものか

3）誰が最初にウォーターフォールモデルという用語を使ったのか

を明らかにするというものである。

「2．Royceの主張」で検証したRoyceの主張は，ソフトウェア危機といわれる時代，大規模ソフトウェアの開発には，従来の職人芸的なソフトウェア開発ではなく，製品製造過程のようなトップダウンアプローチが求められているが，それはそのままだとうまくいかず，それを成功させるためには5つの追加項目が必要だといっている。「4．ウォーターフォールの起源」での検討を踏まえると，このトップダウンアプローチとはサイト防衛アプローチを意味すると考えられる。

現在のウォーターフォールモデルと呼ばれているのは，まさしくこのサイト防衛アプローチにみられるトップダウンアプローチなのである。そして，それではうまくいかないという批判が，アジャイルに代表される他の開発モデルの台頭を生んでいる。

ウォーターフォールという用語を初めて論文に使ったのはTRW社に属するBellとThayerであると述べたが，TRW/軍サイト防衛ソフトウェアプロジェクトという大プロジェクトを引き受けたTRW社内におけるサイト防衛アプローチなどのトップダウン的なアプローチをウォーターフォールと彼らは名づけた。

当然それは同じTRW社のBoehmも耳にしているはずで[8)]，Boehm［1981］で，このトップダウンアプローチにRoyceの指摘した隣接ステップへの相互移動をつけた図を示し，これがウォーターフォールモデルというソフトウェアライフサイクルだと述べ，そのオリジナル版は，ここでいうオリジナルとは隣接ステップへの相互移動であるが，Royce［1970］にあると述べたのである。Boehmはソフトウェア開発工数見積のCOCOMOやスパイラルモデルを提唱したソフトウェア界の大御所であり，彼のその発言が，その後，サイト防衛アプローチに見られるトップダウンアプローチによるソフトウェア開発をウォーターフォールモデルと呼び，そのオリジナルの提唱者はRoyceであるという誤解を生んだと考えられる。

このような結果を見ると，先行研究調査で紹介した菅野［1996］の主張は，幾つかの重要な事実が抜けてはいるが，全くその通りであったといえる。また，もうひとつ紹介したウェブサイトの記述（Weisert［2003］），「Boehmはウォーターフォールという用語を最初に使ったことを否定している」という記述もまた正しかったのである。また，Boehmがウォーターフォールモデルのオリジナルは Royceだと指摘したことにより，その誤解が拡がった。

1970年といえば今から40年以上前である。当時活躍されていた技術者達は現在高齢になられているであろう。また，40年前に今のコンピュータを取り巻く状況を予見できた人はそれほど多くはないはずである。40年前にその当時行っていることで，どんなことの記録が40年後に必要になるかを推測し，それを記録に残そうとした人はほぼいないであろう。今のうちに，ソフトウェア危機と呼ばれた時代にどういう出来事があったのかを現代の目で整理し記録に残すことはソフトウェア工学の今後にとって有意義なことである。

8）Boehm［1976］でBell and Thayer［1976］を参照している。

６．結　　論

今回の調査対象に対する回答を述べ本論の結論とする。

１）Royceの主張はどのようなものか

・大規模ソフトウェア開発にはサイト防衛アプローチのようなトップダウンアプローチが必要（図２）

・しかし，それには隣接する段階での双方向の往来が必要（図３）

・さらに，５つのステップが必要である。（図10）

２）現在のウォーターフォールモデルの定義とそれに対する批判はどのようなものか

・現在，ウォーターフォールの明確な定義はされていないが，「ウォータフォールモデルは，それぞれの作業ステップを一回だけ通して行うというのが特徴である。利用者のニーズの決定からはじまって，要求の定義，システムの設計，システムの作成，テスト，修正，そして導入，運用に至る」と説明されている。

・要求を完全に整理することは困難であるし，また開発の後期にならないと問題が表面化しないなど，Royceが指摘したのと同様な批判がある。

３）誰が最初にウォーターフォールモデルという用語を使ったのか

・BellとThayerの論文（Bell and Thayer［1976］）にて初めてウォーターフォールという用語が使われた。

「Royceがウォーターフォールモデルの提唱者である」という誤解については，ウォーターフォールという言葉を最初に使ったのはBellとThayerがTRW社におけるサイト防衛アプローチにみられるトップダウンアプローチに対してであり，Royceはそれを批判していたという事実をもって，その誤解を解くものである。また，Boehmがウォーターフォールモデルのオリジナルは Royceだと指摘したことにより，その誤解が拡がった。

以上，調査対象とした疑問は解けたのであるが，ひとつ疑問が残る。なぜBoehmをはじめとするTRW社の関係者がウォーターフォールの出所について

明らかにしなかったのか。今後機会があれば調査したい。

参考文献

Andriole,S.J. and P.A.Freeman［1993］“Software Systems Engineering : The Case for A New Discipline,” Software Engineering Journal, 8(3), pp.165-179

Bell,T.E. and T,A.Thayer［1976］“Software Requirement : Are They Really A Problem?,” ICSE '76: Proceedings of the 2nd international conference on Software engineering, pp.61-68

Boehm,B.W.［1976］“Software Engineering,” IEEE Transactions, 25(12), pp.1226-1241

Boehm,B.W.［1981］*Software Engineering Economics,* Prentice Hall.

Boehm,B.W.［1988］“A Spiral Model of Software Development and Enhancement,” Computer, 21(5), pp.61-72

Brooks,F.P.［2010］*The Design of Design,* Person Education.（松田晃一・小沼千絵訳［2010］『デザインのためのデザイン』ピアソン桐原）

Brooks.F,P.［1995］*The Mythical Man-month, Aniversary Edition,* Person Education（訳書：滝沢徹・牧野祐子・富澤昇訳［2010］『人月の神話』ピアソン桐原）

平井利明監修執筆・斎藤裕美［2003］『情報処理技術者テキスト 基本情報処理技術者プラスアルファ：Ⅳシステム開発とその運用』実教出版

廣瀬健・高橋延匡・土井範久［1990］『コンピュータソフトウェア事典』丸善

IPA［1994］情報処理技術者試験出題範囲
http://www.jitec.ipa.go.jp/1_13download/hani.pdf　2013年２月15日参照。

IPA［2009］a 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2007第２版：経営者，業務部門が参画するシステム開発及び取引のために』オーム社

IPA［2009］b 平成21年度春期データベーススペシャリスト午前Ⅱ問題問25
http://www.jitec.ipa.go.jp/1_04hanni_sukiru/mondai_kaitou_2009h21_1/2009h21h_db_am2_qs.pdf　2013年２月15日参照

磯田定宏［1996］「ソフトウェア危機はその後どうなったか」『電子情報通信学会誌』79(12), pp.1242-1244

菅野孝男［1996］『改訂ソフトウェア開発のマネージメント』新紀元社。

Larman,C.（2003）*Agile and Iterative Development : A Manager's Guide,* Wesley Professional Press.（児高慎治郎・松田直樹監訳［2004］『初めてのアジャイル開発』日経BP社）

McBreen,P.［2002］*Software Craftsmanship : The New Imperative,* Person Education.（村上雅章訳［2002］『ソフトウェア職人気質：人を育て，システム開発

を成功へと導くための重要キーワード』ピアソン・エデュケーション）

Naur,P. and B.Randell. eds.［1969］*Software Engineering : Report on a conference sponsored by the NATO SCIENCE COMMITTEE*
http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF　2013年２月15日参照

Royce,W,W.［1970］“Managing the Development of Large Software Systems,” *In ICSE '87 Proceedings of the 9th international conference on Software Engineering,* pp.328-338

Royce,W.［1998］*Software Project Management,* Addison-Wesley Professional（日本ラショナルソフトウェア監訳［2001］『ソフトウェアプロジェクト管理』ピアソン・エデュケーション）

TAC情報処理技術者講座［2008］『基本情報技術者 試験対策テキスト：Ⅱシステムの利用と開発編』TAC株式会社

玉井哲雄［2008］「ソフトウェア工学の40年」『情報処理』49(7) pp.777-784

Weisert,C.［2003］http://www.idinews.com/waterfall.html　2013年２月15日参照

Willams,R.D.［1975］“Managing The Development of Reliable Software,” *In Proceedings of the international conference on Reliable software,* pp.3-8

Winograd,T.［1996］*Bring Design to Software,* Addison-Wesley Professional（瀧口範子訳［1998］『ソフトウェアの達人たち：認知科学からのアプローチ』アジソン・ウエイレス・パブリッシャーズ・ジャパン）

Wolverton,R.W.［1974］“The Cost of Developing Large-Scale Software,” IEEE Transactions, 23(6), pp.615-636