AgileがWaterfallより有利となる開発条件の検証

Verification of the development conditions that Agile is advantageous than Waterfall

駒井忍† 　鈴木健 　廣瀬嘉代
Shinobu Komai†, Ken Suzuki, and Kayo Hirose
†マレーシア工科大学　マレーシア日本国際工科院　技術経営学部
† Faculty of Management of Technology, Malaysia Japan International Institute,
Universiti Teknologi Malaysia

要旨

システム開発において Agile が Waterfall に比べて有利になる条件を機能数、一要件定義工数、一機能開発工数、作業影響率、作業人月をパラメータとして変化させ、要件定義工数合計と開発工数合計を算出することにより検証した. この結果同条件で二つの開発方法を実施した場合、ある機能数までは Agile が有利だが、それ以降は Waterfall のほうが有利となる均衡点があることがわかった.

1. はじめに

要件定義（SRD）フェーズは不確実なユーザの要求を知る最初のフェーズであり、システム開発における鍵となる[1]. しかし、要件定義フェーズに関連した多くの失敗事例が日本で報告されている[2], [3]. 一方、Forrester Research, Inc.は 2009 年第 3 四半期に実施した調査結果を公表し、「Agile 型開発が主流になった」と述べ、Agile 型開発の採用が 35%となり、Waterfall 開発の 13%を大きく上回ったと報告した[4]. しかしながら要件定義フェーズの視点からシステム開発を考察した研究事例はほとんどない．本論文では、3 つの先進的な IT システム開発事例において Waterfall 型開発が問題となった失敗事例を克服するための Agile 開発の可能性を検討した．先ず、Agile 開発で対象とする要件数、要件定義の工数、要件定義後の開発工数、および開発作業の手戻り率をパラメータとして設定し、手戻り工数を計算する表計算プログラムを開発した．次に、Agile 開発の「必要な要件から順番に、迅速に開発する」という特徴から「ユーザにとって価値のある機能を迅速に提供し、無駄な要件は作らない」メリットと、「開発作業で手戻り工数が発生する」デメリットに着目し、3 つの事例を表計算プログラムに当てはめて考察した.

2. 研究手法について

2.1. Agile 開発について

Agile 開発の特徴は部分先行開発にある．部分先行開発とは、ユーザ要件が明確になった機能、あるいは機能の集まり毎に開発に着手することである．部分先行開発後、他のユーザ要件が確定することによる先行開発部分への影響即ち手戻りをどれだけ想定するかが重要なポイントとなる. Agile の特徴は、「必要な要件から順番に、迅速に開発する」という点にある．結果として、ユーザにとって本当に価値のある機能を迅速に提供でき、無駄な要件は作らないで済むというメリットが生まれる[5]．しかし、先行開発作業に対する手戻り工数が発生するデメリットがある．本研究では、まず無駄な要件を作らない「メリット＝未着手の要件＋開発作業の工数」と「デメリット＝既に開発を終了した作業に関する手戻り工数」との「工数の均衡点」を以下のシミュレーションテーブルを構築して見極めた.

2.2. Agile 開発シミュレーションテーブルの構築

以下にシミュレーションテーブル構築の考え方を述べる.

本研究では、要件が固まった機能から開発し、開発着手済みの要件に対しては確認済みのため変更はないものとする．Agile で開発する機能数を N として、要件が確定した機能、あるいは機能の集まりから順次開発に着手する．要件確定のための工数 Ri を（R1, R2, R3, --, Rn）とし、要件確定後の開発工数 Di を(D1, D2, D3, --, Dn)として与える. 要件確定のための工数 R1＝1 とし、その他を R1 で規格した値 R、開発のための工数 D1＝2 として、その他を D1 で規格化した値 D で与える．要件確定のための工数は R=1 に対する値を 0.6 から 1.4 まで 0.1 刻み増加する変数とし、開発のための工数は D1=2 に対する値 1.2

から 5.5 まで 0.5 刻みに増加する変数とする．上記の要件確定のための工数は「要件定義フェーズ」の工数であり、開発のための工数は「システム基本設計、プログラム設計、プログラミング、およびプログラム単体テスト・フェーズ」の工数とする．

以下のように手戻りがない Agile 開発の合計工数は Waterfall 開発工数と同じになる．

$\sum_{i=1}^{n} Ri + Di$　　**式 1**

表 1 手戻りがない Agile 開発の工数

機能数(N)	N=1	N=2	N=3	・	・	N=n	合計工数
要件定義(Ri)	R1	R2	R3	・	・	Rn	
要件定義工数	1	0.6-1.4	0.6-1.42	・	・	0.6-1.4	$\sum_{i=1}^{n} Ri$
開発作業(Di)	D1	D2	D3	・	・	Dn	
開発作業工数	2	1.2-2.8	1.2-2.8	・	・	1.2-2.8	$\sum_{i=1}^{n} Di$
工数計(Ri+Di)	3	1.8-4.2・	1.8-4.2	・	・	1.8-4.2	$\sum_{i=1}^{n} Ri + Di$

以下に Agile 開発における手戻りが必要なケースを列挙し、対応方法を記述する．

1) 開発したモジュール間の相互利用を考慮した手戻り工数の計算

上記表 1 を基に、ユーザ要求が確定したと看做して開発に着手した機能の開発後の他のユーザ要求の確定によって開発に着手した要求による設計変更が生じる場合の手戻りの計算表を考える．この場合の Agile 開発の工数は、要件の変更ではなく開発したモジュール間の相互利用を考慮した手戻り工数として計算する．以下の表 2 のように、開発作業 Di=(D1, D2, D3, --, Dn)で相互に設計変更が生じる割合を開発作業変更率 a=0～100%として与える．また、その設計変更によって生じる手戻り工数の割合を開発作業影響率 b=0～100%として与えて計算する．手戻り工数については、開発作業の工数は j 個目の Agile 開発が、i 個目単独の Agile 開発工数 Dii に影響する手戻り工数を Aij として、i 個目の開発工数に対する手戻り工数は Aij (j=1～i-1, i+1～n）と考えられるが、便宜上すべての要件の開発において均等に発生するものとして Aij は Di*a/100*b/100 として計算する．

表 2 Agile 開発工数計算方法

機能数(N)	N=1	N=2	N=3	・	・	N=n	合計工数
要件定義(Ri)	R1	R2	R3	・	・	Rn	
要件定義工数	1	0.6-1.4	0.6-1.4	・	・	0.6-1.4	$\sum_{i=1}^{n} Ri$
開発作業(Di)	D1	D2	D3	・	・	Dn	
開発作業工数	2	1.2-2.8	1.2-2.8	・	・	1.2-2.8	$\sum_{i=1}^{n} Di$
工数計(Ri+Di)	3	1.8-4.2	1.8-4.2	・	・	1.8-4.2	$\sum_{i=1}^{n} Ri + Di$
開発作業変更率(a)			手戻り				10%-100%
開発作業影響率(b)			手戻り				10%-100%
手戻り工数	D1*a*b/100	D2*a*b/100	D3*a*b/100	・	・	Dn*a*b/100	$\sum_{i=1}^{n} Di * a * b/100$

2) システムテストで発見されたプログラム不具合修正のための手戻り

一般的に Waterfall ではシステムテストは開発終了後に実施するが、Agile 開発は要求の集まり毎に複数回システムテストし、そこで得られた知見や経験を開発作業にフィードバックしながら効率を上げていく[6]．この効率化については、上記表 2 の右側にテスト工数をシミュレーションできるように拡張して検討することは可能と思われる．しかし、知見や経験の数値化は大変難しい．複数の事例から実際のテスト工数を把握しながらデータを分析すればできそうだが、将来の研究の課題とする．

3) ミドルウェアの適合性に関する手戻り

使用を見込んでいたミドルウェアを使ってプログラム開発を進めていたが、要件の確定が進むにつれて、そのミドルウェアの使用が不適切であると判明し新たなミドルウェアへの変更が生じる場合の部分

先行開発に関する無駄な工数の手戻り．Waterfall では要件が完全に確定するまで開発に着手しないため、無駄な開発工数は発生しない．どの時点でミドルウェアの使用が不適切であると判断するかについては、その判断基準を設定することが重要である．この点も将来の研究の課題とする．

上記以外に以下の手戻りの可能性が考えられるが、いずれも Waterfall も同様とする．

4) 上記 2)項の手戻りが開発中の他のアプリケーションに影響する場合の手戻り

事例 3 で CTI, IVR, Soft phone, and Middleware 基盤の 4 つのモジュールの関連性が高かったため発生した．これは一つの要件を同時に 4 つの開発チームで開発したためであるが、Waterfall も同様とする．

5) システム基本設計で発見された要件定義への手戻り

6) プログラム単体テストで発見されたプログラム不具合のための手戻り

7) ユーザによる検収テストで発見されたプログラム不具合修正のための手戻り

8) ユーザによる検収テストで指摘されたプログラム品質要件改善のための手戻り

9) 品質要件改善のためのプログラム設計、システム基本設計、および要件定義に関する修正の手戻り

10) ミドルウェア不具合修正のためにベンダーから提供されるミドルウェア置換後のシステムテストで発見されるプログラム不具合修正のための手戻り

11) ミドルウェアとアプリケーションの間での手戻り

アプリケーション（プログラム）からミドルウェアにパラメータを与えたときの動作は以下の二点が考えられる．

(1) パラメーターエラー

ミドルウェアがチェックしてプログラマーにエラーとして返すため、プログラム作成工数に含まれる

(2) パラメータ間違い

プログラムで間違ったパラメータをミドルウェアに与えた場合は、ミドルウェアが間違ったデータを返すため、プログラム単体テストでは分からない場合がある．そのため、システムテスト、あるいはユーザによる検収テストでプログラムの不具合として発見される．

上記 1)項で説明した Agile 開発工数計算方法の考え方に基づいて、機能数 N と各パラメータを変化させながら、表計算ソフトを利用して表 3「Agile Waterfall Simulation Table I (以下 AWST-I)」を構築した．

表 3 Agile Waterfall Simulation Table の例

N	10										
R2-Rn	0.6										
D2--Dn	1.2										
a	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	
b	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	
N	1	2	3	4	5	6	7	8	9	10	
Ri	1	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	6.4
Di	2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	12.8
Ri+Di	3	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	19.2
Hand over	0.28	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	0.168	1.792
Dit	2.28	1.368	1.368	1.368	1.368	1.368	1.368	1.368	1.368	1.368	14.592
Total	3.28	1.968	1.968	1.968	1.968	1.968	1.968	1.968	1.968	1.968	20.992

N	10										
R2-Rn	0.6										
D2--Dn	1.4										
a	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	
b	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	
N	1	2	3	4	5	6	7	8	9	10	
Ri	1	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	6.4
Di	2	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	14.6
Ri+Di	3	2	2	2	2	2	2	2	2	2	21
Hand over	0.28	0.196	0.196	0.196	0.196	0.196	0.196	0.196	0.196	0.196	2.044
Dit	2.28	1.596	1.596	1.596	1.596	1.596	1.596	1.596	1.596	1.596	16.644
Total	3.28	2.196	2.196	2.196	2.196	2.196	2.196	2.196	2.196	2.196	23.044

この表では、D2-D10 が 1.4 のときに「未着手の要件＋開発作業の工数」＜「既に開発を終了した作業に関する手戻り工数」として「工数の均衡点」（赤で塗りつぶして表示）が確認できた．

3. 研究の成果

3.1. シミュレーションテーブルから工数比較テーブル(AWST-II)の作成

上記 1)項で説明したシミュレーションテーブル構築の考え方に基づいて機能数 N と各パラメータを変化させながら、Agile と Waterfall の工数比較表=Agile Water fall Simulation Table II (以下 AWST-II と略す)を構築した．具体的には要件数 N を 10 に固定し、要件定義工数 R2-Rn を 0.6 から 1.4 まで 0.1 刻みで 9 通りに変化させる．同様に開発工数 D2-Dn を 1.2 から 2.8 まで 0.2 刻みで 9 通りに変化させ、開発作業変更率 a、および開発作業影響率 b を各々10%から 100%に 10%刻みで 10 通りに変化させて、各機能数 N に対して 8100 行分（通り）のテーブルを構築した．その結果、例えば以下の表 4 のように工数の均衡点が見つかった．

表 4 Agile Waterfall Simulation Table II の均衡点例

1<n<= Nmax, nInteger		
N (機能数)	10	Max 128
R1 (要件定義工数1)	1	固定
R2 (一機能要件定義工数)	0.6	0.6-1.4,0.1刻み
D1 (開発工数1)	3.5	固定
D2 (一機能開発工数)	1.2	1.2-2.8,0.2刻み
a (開発作業変更率)	10%	10% - 100%,10%刻み
b (開発作業影響率)	10%	10% - 100%,10%刻み
K (Man Power)	1.0666667	月平均要件定義工数
L (Month)	6	固定

色	意味
	固定値
	変数
	計算結果
	計算結果(合計)

機能数	要件定義工数(1)	要件定義工数(2-n)	要件定義工数計	開発工数(1)	開発工数(2-n)	開発工数計	Waterfall工数合計	設計変更率	開発影響率	Agileの要件定義工数	要件定義工数差	手戻工数合計	Agile工数合計	Agile工数合計-WF工数合計
N	R1	R2	ΣRi	D1	D2	ΣDi	ΣRi+ΣDi	a	b	ΣRij	ΣRi-ΣRij	ΣAij	P(ΣAgile manpower)	P-(ΣRi+ΣDi)
10	1	0.6	6.40	3.5	1.2	14.30	20.70	30%	60%	3.52	2.88	2.57	20.39	-0.31
10	1	0.6	6.40	3.5	1.2	14.30	20.70	30%	70%	3.52	2.88	3.00	20.82	0.12
10	1	0.6	6.40	3.5	1.2	14.30	20.70	40%	50%	3.52	2.88	2.86	20.68	-0.02
10	1	0.6	6.40	3.5	1.2	14.30	20.70	40%	60%	3.52	2.88	3.43	21.25	0.55
10	1	0.6	6.40	3.5	1.2	14.30	20.70	50%	40%	3.52	2.88	2.86	20.68	-0.02
10	1	0.6	6.40	3.5	1.2	14.30	20.70	50%	50%	3.52	2.88	3.58	21.40	0.70
10	1	0.6	6.40	3.5	1.2	14.30	20.70	60%	30%	3.52	2.88	2.57	20.39	-0.31
10	1	0.6	6.40	3.5	1.2	14.30	20.70	60%	40%	3.52	2.88	3.43	21.25	0.55
10	1	0.6	6.40	3.5	1.2	14.30	20.70	70%	20%	3.52	2.88	2.00	19.82	-0.88
10	1	0.6	6.40	3.5	1.2	14.30	20.70	70%	30%	3.52	2.88	3.00	20.82	0.12
10	1	0.6	6.40	3.5	1.2	14.30	20.70	80%	20%	3.52	2.88	2.29	20.11	-0.59
10	1	0.6	6.40	3.5	1.2	14.30	20.70	80%	30%	3.52	2.88	3.43	21.25	0.55
10	1	0.6	6.40	3.5	1.2	14.30	20.70	90%	20%	3.52	2.88	2.57	20.39	-0.31
10	1	0.6	6.40	3.5	1.2	14.30	20.70	90%	30%	3.52	2.88	3.86	21.68	0.98
10	1	1.4	13.60	3.5	2.6	26.90	40.50	50%	40%	7.48	6.12	5.38	39.76	-0.74
10	1	1.4	13.60	3.5	2.6	26.90	40.50	50%	50%	7.48	6.12	6.73	41.11	0.61
10	1	1.4	13.60	3.5	2.6	26.90	40.50	60%	30%	7.48	6.12	4.84	39.22	-1.28
10	1	1.4	13.60	3.5	2.6	26.90	40.50	60%	40%	7.48	6.12	6.46	40.84	0.34
10	1	1.4	13.60	3.5	2.6	26.90	40.50	70%	30%	7.48	6.12	5.65	40.03	-0.47
10	1	1.4	13.60	3.5	2.6	26.90	40.50	70%	40%	7.48	6.12	7.53	41.91	1.41
10	1	1.4	13.60	3.5	2.6	26.90	40.50	80%	20%	7.48	6.12	4.30	38.68	-1.82
10	1	1.4	13.60	3.5	2.6	26.90	40.50	80%	30%	7.48	6.12	6.46	40.84	0.34
10	1	1.4	13.60	3.5	2.6	26.90	40.50	90%	20%	7.48	6.12	4.84	39.22	-1.28
10	1	1.4	13.60	3.5	2.6	26.90	40.50	90%	30%	7.48	6.12	7.26	41.64	1.14
10	1	1.4	13.60	3.5	2.6	26.90	40.50	100%	20%	7.48	6.12	5.38	39.76	-0.74
10	1	1.4	13.60	3.5	2.6	26.90	40.50	100%	30%	7.48	6.12	8.07	42.45	1.95
10	1	1.4	13.60	3.5	2.8	28.70	42.30	30%	70%	7.48	6.12	6.03	42.21	-0.09
10	1	1.4	13.60	3.5	2.8	28.70	42.30	30%	80%	7.48	6.12	6.89	43.07	0.77
10	1	1.4	13.60	3.5	2.8	28.70	42.30	40%	50%	7.48	6.12	5.74	41.92	-0.38
10	1	1.4	13.60	3.5	2.8	28.70	42.30	40%	60%	7.48	6.12	6.89	43.07	0.77
10	1	1.4	13.60	3.5	2.8	28.70	42.30	50%	40%	7.48	6.12	5.74	41.92	-0.38
10	1	1.4	13.60	3.5	2.8	28.70	42.30	50%	50%	7.48	6.12	7.18	43.36	1.06
10	1	1.4	13.60	3.5	2.8	28.70	42.30	60%	30%	7.48	6.12	5.17	41.35	-0.95
10	1	1.4	13.60	3.5	2.8	28.70	42.30	60%	40%	7.48	6.12	6.89	43.07	0.77
10	1	1.4	13.60	3.5	2.8	28.70	42.30	70%	30%	7.48	6.12	6.03	42.21	-0.09
10	1	1.4	13.60	3.5	2.8	28.70	42.30	70%	40%	7.48	6.12	8.04	44.22	1.92
10	1	1.4	13.60	3.5	2.8	28.70	42.30	80%	20%	7.48	6.12	4.59	40.77	-1.53
10	1	1.4	13.60	3.5	2.8	28.70	42.30	80%	30%	7.48	6.12	6.89	43.07	0.77
10	1	1.4	13.60	3.5	2.8	28.70	42.30	90%	20%	7.48	6.12	5.17	41.35	-0.95
10	1	1.4	13.60	3.5	2.8	28.70	42.30	90%	30%	7.48	6.12	7.75	43.93	1.63
10	1	1.4	13.60	3.5	2.8	28.70	42.30	100%	20%	7.48	6.12	5.74	41.92	-0.38
10	1	1.4	13.60	3.5	2.8	28.70	42.30	100%	30%	7.48	6.12	8.61	44.79	2.49

3.2. Agile Waterfall Simulation Table (AWST-II) の検討

表 4 で確認した AWST-II における「工数の均衡点」については、対象となるシステム開発の機能数 N、要件定義工数 Ri、開発工数 Di、開発作業変更率 a、および開発作業影響率 b が以下の式を満たすときに「未着手の要件＋開発作業の工数」＞「既に開発を終了した作業に関する手戻り工数」となる．

$$Rn + Dn >= \sum Aij \quad (式2、但し\ Aij = Di * a * b/100\ )$$

上記の結果から Agile 開発は「無駄な機能の開発をしない工数が手戻り工数より大きい」場合に少ない工数ですむことが確認できた．また、構築した AWST-II のどこかの行に Agile 開発を選択する対象のシステム開発が当てはまることが分かった.

3.3. IT システム開発事例での検証

上記で構築した AWST を以下の表 5 にある 3 つの IT システム開発事例で検証する．この内容は実際の IT システム開発プロジェクトのソフトウェア構造を表している.

表 5. IT システム開発におけるソフトウェア構造

ソフトウェア構造	事例１	事例２	事例３
①基盤ソフト（システム）	O/S: Windows 2003 Server 数：2	O/S: Windows2003 Server 数：2	O/S: Windows2003 Server 数：10
②基盤ソフト上のミドルウェア	数：2 * Java Synchronization Identity Management System	数：2 * Identity Integration Management System	数：6 Call Center Integration System, Database Software, and Voice Exchange System
③ミドルウェア上のアプリケーション	数：2 Java and Windows Connector	数：2 HP/Unix and Windows Connector	数：4 Computer Telephony Integration (CTI), Interactive Voice Response (IVR), Soft Phone, Middleware Infrastructure
④連携したアプリケーションソフト	数：1 Human resources management System	数：1 Financial Intelligence System	数：1 Customer Relationship Management System

開発規模から推定される工数計算のパラメータを表 6 のように設定し、各パラメータを AWST に入力してシステムごとに均衡点を確認、検討した.

表 6 事例から推定される工数算のパラメータ

パラメータ	事例１	事例２	事例３
N：要件の数	3	11	101
要件定義工数(R2, R3, ---, Rn)	0.6	0.6	0.6
開発作業工数(D2, D3, ---, Dn)	2	2	2
開発作業変更率(a)	30%	40%	40%
開発作業影響率(b)	20%	30%	30%

上記の結果から、事例 1 では推定される Agile の手戻り工数は 0.36、最終要件 N=3 の「要件定義＋開発作業工数」は 2.6 となる．この事例のように要件数の小さいシステム開発では無駄な要件を作らない Agile 開発のメリットをそれほど享受できない．結論としては、Water fall 開発の開発期間が許されるなら無理に Agile を志向しなくてもよい.

事例 2 では推定される Agile の手戻り工数は 2.64、最終要件 N=11 の要件定義＋開発作業工数は 2.6 となる．この事例では、N=10 までの手戻り工数が 2.4 なので、最終要件 N=11 の要件定義＋開発作業を無駄な要件として確認できれば、Agile 開発の工数が 28.8 と少なくてすむ.

事例 3 では、表 6 のように推定される Agile の手戻り工数の累積値は N=92 の 22.08 が、N=93 以降の Ri+Di の合計値 23.4 に対して均衡点となる.

この条件式は以下のようになる.

$\sum Aij =< (Nmax - Ni) * (Ri + Di)$（式 3）

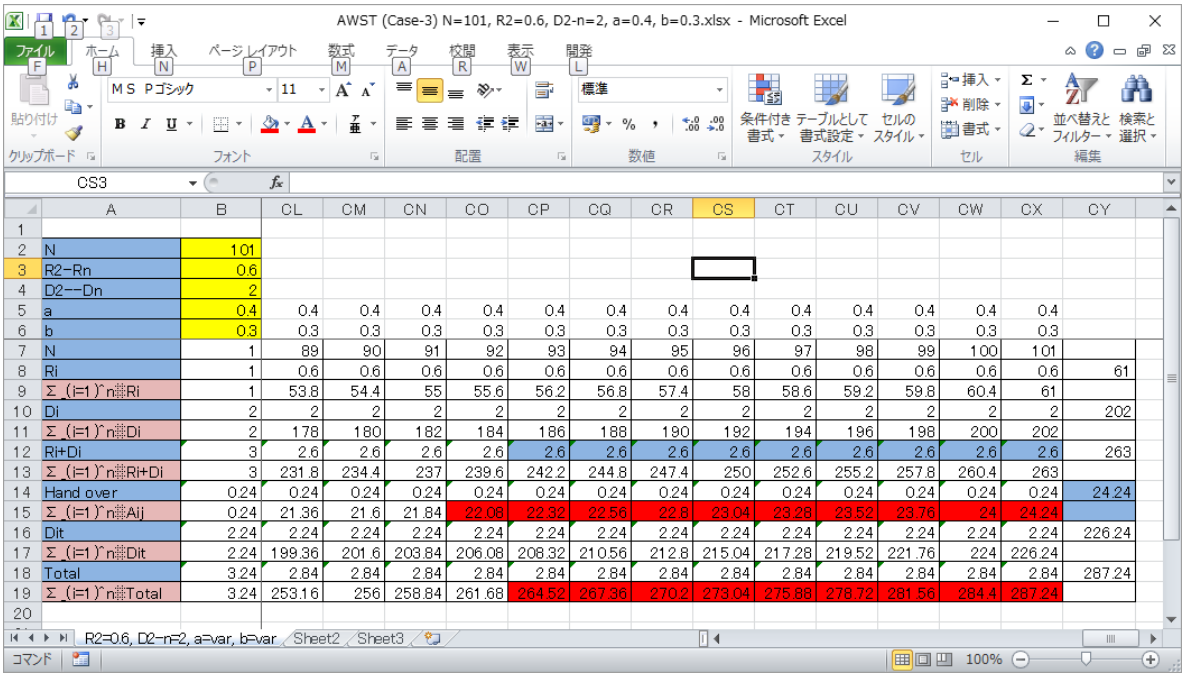
また、この均衡点は以下の式で計算できる.

① $\mathrm{Rounddown}\left(\mathrm{Nmax} * \frac{\mathrm{Ri+Di}}{\mathrm{abDi+Ri+Di}}, 0\right)$ → ここまでは Agile が有利

② $\mathrm{Roundup}(\mathrm{Nmax} * \frac{\mathrm{Ri+Di}}{\mathrm{abDi+Ri+Di}}, 0)$ → 上記以降は Waterfall が有利

実際に Nmax=101, Ri=0.6, Di=2, a=0.4, b=0.3 を当てはめると①は92 となり ②は93 となるので表 6 と合っている.

表 7 均衡点の確認



	A	B	CL	CM	CN	CO	CP	CQ	CR	CS	CT	CU	CV	CW	CX	CY
2	N	101														
3	R2-Rn	0.6														
4	D2--Dn	2														
5	a	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	
6	b	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	
7	N	1	89	90	91	92	93	94	95	96	97	98	99	100	101	
8	Ri	1	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	61
9	Σ_(i=1)^n※Ri	1	53.8	54.4	55	55.6	56.2	56.8	57.4	58	58.6	59.2	59.8	60.4	61	
10	Di	2	2	2	2	2	2	2	2	2	2	2	2	2	2	202
11	Σ_(i=1)^n※Di	2	178	180	182	184	186	188	190	192	194	196	198	200	202	
12	Ri+Di	3	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6	263
13	Σ_(i=1)^n※Ri+Di	3	231.8	234.4	237	239.6	242.2	244.8	247.4	250	252.6	255.2	257.8	260.4	263	
14	Hand over	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	24.24
15	Σ_(i=1)^n※Aij	0.24	21.36	21.6	21.84	22.08	22.32	22.56	22.8	23.04	23.28	23.52	23.76	24	24.24	
16	Dit	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	226.24
17	Σ_(i=1)^n※Dit	2.24	199.36	201.6	203.84	206.08	208.32	210.56	212.8	215.04	217.28	219.52	221.76	224	226.24	
18	Total	3.24	2.84	2.84	2.84	2.84	2.84	2.84	2.84	2.84	2.84	2.84	2.84	2.84	2.84	287.24
19	Σ_(i=1)^n※Total	3.24	253.16	256	258.84	261.68	264.52	267.36	270.2	273.04	275.88	278.72	281.56	284.4	287.24	

4. まとめ

本研究では、Agile 開発が「必要な要件から順番に、迅速に開発する」という特徴と、「ユーザにとって本当に価値のある機能を迅速に提供でき、無駄な要件は作らないで済む」というメリットに着目した. 具体的には、手戻りがない Waterfall 開発と Agile 開発のデメリットである先行開発作業に対する手戻り工数について一般化し、定式化して両方の開発工数を計算により求めた. 本研究で構築した Agile Waterfall Simulation Table を利用すると、Agile 開発の特徴である「無駄な要件を作らないメリット＝未着手の要件＋開発作業の工数」と「デメリット＝既に開発を終了した作業に関する手戻り工数」の比較ができる. このような考え方を発展させてシステム開発の手法を検討することが重要であることが分かった. 本研究で得られた成果は、今後のシステム開発における開発手法の選択に対して大きな指針となる.

参考文献

[1] 石井信明, “要件定義におけるプロジェクトマネジメントフレームワークの提案,” 文教大学情報学部『情報研究』第35号, pp. 47-66, 2006.
[2] 日本経済新聞社, “ＩＢＭに74億円の賠償命令、スルガ銀裁判の深層,” 2012.
[3] 日経コンピュータ, “55億円無駄に、特許庁の失敗,” 2012.
[4] D. K. Taft, “Agile Development Hitting the Mainstream, Report Says,” 2010.
[5] みずほ情報総研, “ユーザ企業におけるアジャイル開発のメリットと課題,” 2012. [Online]. Available: http://www.mizuho-ir.co.jp/publication/column/2012/1225.html.
[6] H. Kniberg, “リーン開発の現場、カンバンによる大規模プロジェクトの運営,” オーム社, 2013.