Vol.2013-IS-124 No.6
2013/6/7

アジャイル開発の見積もり方法の考察

吉田　知加[†1]　　木野　泰伸[†1]

日本ではここ数年来，アジャイル（反復型）開発が注目される半面，ウォーターフォール型が定着した市場で，未だにその適用率は欧米に及ばず低迷している．その主たる理由としては，変更を受け入れる為の見積もりと契約の難しさにある．本研究では Ruby によるアジャイル開発を事業戦略のひとつとする IT 企業のプロジェクト事例よりアジャイル開発での見積もりの実態の分析と，精度向上のための手法について検討する．

この見積もり手法を考察することにより，アジャイル開発の時間・品質・コストの不確実性を削減し，利用範囲を拡大できることを目的とする．

A Way of Estimation for Agile Software Development

CHIKA YOSHIDA[†1]　YASUNOBU　KINO[†1]

An agile development method has been seen as new software development method in Japan.　However it not so familiar than Water whole model in Japan and the ratio of the adaptation has been lower than in US and Western countries. The main reason should be the difficulties of way of estimation to conclude the contract due to accept the changes.

This research shows an effective and dependable way of estimation for agile development by Ruby with real case of a project of an IT company who think agile development as a business strategy. It aims to reduce the uncertainty of time, quality and cost by using this estimation method for expanding the use of an agile development.

1. はじめに

IT(情報技術)がビジネスの戦略を実現する差別化要因になっている現在，経営環境変化に柔軟に対応でき，迅速なシステム開発が可能な「アジャイル開発」が注目されている．特に欧米ではウォーターフォールを凌いで一般的開発方法となっている．日本では未だ，ウォーターフォールが主流で，反復型開発を採用したプロジェクトは，全体の僅か2.8%とされている.[1)]

その背景には，変更に対応するゆえの初期段階での見積もりの難しさ，契約時の成果物をはじめとする誓約要件の合意の難しさがある．今回の研究は，アジャイル開発の見積もりもり手順を明確化し，その精度を高めることで市場を拡大することにある．しかし，この研究に際しアジャイル開発のデータ採集を試みたが，次の事由により予想以上にその採集は容易でなかった．

①アジャイル開発自身を実施している企業が少ない．

②研究に必要なデータ項目を採集し残している企業が少ない．

その中で島根県に本社をおきRubyの開発事業を戦略のひとつとするB社は2-3年前より業務アプリケーション開発を進めており，その内一つのプロジェクトデータを今回の研究に提供頂いた．

B社のアジャイル開発へ事業推進の狙いは，従来の受託開発事業から顧客とのビジネス創出型事業へ主軸を移すことであり，将来に向けた新規分野への展開とともにIターンも含めた雇用確保を実現することにある．また島根県がRubyによるIT産業の振興事業を積極的に推進していることも，B社の戦略実現に大きな追い風となっている.ただ，B社の一プロジェクトでは検証が難しい．他数社にコンタクトしデータ提供を試みたが，アジャイル開発を遂行し，そのプロセスで計測･検証に必要なデータを記録･保管し，かつそれらを公開できる企業はなかった.そのようなおり，筑波大学大学院システム情報工学研究科コンピューターサイエンス専攻の PBL（Project Based Leaning）で Ruby を使ったアジャイルによる開発プロジェクト（以下 S プロジェクト）がスタートすると聞き，開始時から必要データの採集と情報提供の協力を依頼した．これにより B 社のデータとともに見積もり手法の考察へ進めることができた．

この 2 つのプロジェクトデータから，計画時見積もり手法の精度と各イテレーションでの生産性分析を行い，これからの研究に繋げたいと考える．

2. Ruby言語とアジャイル開発

従来のウォーターフォール型の開発手法では，すべての機能を開発するまで本番運用ができず，稼動開始までに長

†1 筑波大学
University of Tsukuba

期間を要する.さらに，完成時にはニーズとかけ離れたシステムになるリスクがある．これに対し，スモールスタートで早期に本番運用を開始し，徐々に拡張していくインクリメンタル型の開発ができるアジャイル開発手法は，早期に投資の回収が可能となり，システム開発の投資効率が改善できる.また，ユーザニーズを吸い上げながら徐々に拡張することによってユーザにとって価値のある機能を作り込むことができる利点がある．

これらの事由によりRubyによる開発には，短期間で機能を実装できその特徴を最大限活かすことができるアジャイル開発が向くことがわかる．

表 1　Rubyの特徴
Table 1 Characteristics of Ruby

特徴	利点	効果
オブジェクト指向	テストし易い． モジュール分割が可能	変更可能性の向上
動的型付け	メタプログラミングが可能	プログラムの簡素化
インタプリタ型	プログラム動作までの時間が短い．	動作確認頻度の向上

また，上記のRubyの特徴を活用すれば，短期間でシステムをサービス・インすることができ，かつシステム拡張に要するコストの増大を抑えた開発が可能である．しかし，現状のシステム開発契約の主流である請負契約では，以下の課題がある．

- 計画時にシステム全体の規模・工数を見積もり，それらを基準として算出した契約金額で契約締結するため，計画時に総システム概算が必要となる．
- 開発後，システムを納入，検収後でなければ，本番運用を開始できず，インクリメンタル開発で頻繁にリリースを行うためには，その都度，検収が必要となり，現在のインクリメンタル開発の適用を困難なものとしている．

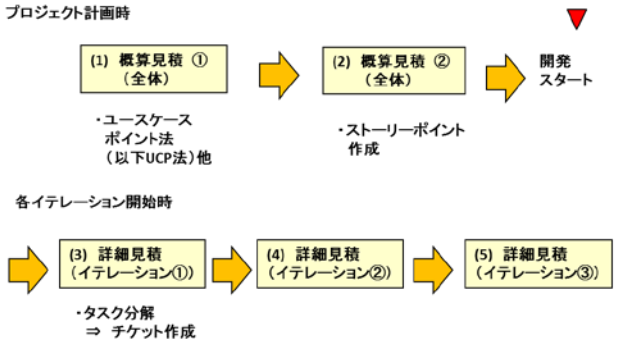
これらの解決法を考えた場合，現状の商習慣の中でその普及を推進する為には，契約そのもののアジャイル開発向け見直しと，計画時に共通な基準によるシステム全体の規模・工数を把握する必要がある．

3. アジャイル開発の見積もりプロセスと手法

アジャイル開発では，プロジェクト開発時（初期契約時）と各イテレーション開始時に見積もりを実施する．

そのプロセスを図 1 に示す．

図 1　アジャイル開発見積もりのプロセス
Figure 1 Process of Estimation for Agile Development



(1)　概算見積①
全体概要見積もりではプロジェクト計画時行い受入要件の機能規模を見積もる．UCP 法は非機能要件（技術，環境要因・リスク）を含めた全体工数も算出できる．

(2)　概算見積②
ストーリーポイントは要件定義後に各要件の規模を把握するために行い，その累計により全体規模を把握する．それらはタスク分解のベースになる．

(3) ~ (5)　詳細見積
各イテレーションの開始時に実施する．ストーリーから最小作業単位のタスクに分解し規模，工数，時間を見積もる．

プロジェクト開始時の見積もりは，要件変更が多く迅速性が重視されるアジャイル開発の場合，完了時との乖離が大きく難しいと言われているが，システム入出力が明確となる設計段階でなければ見積もれないファンクションポイント法（1979.　A.J.Albrecht）[2] に比べ開発の初期段階で利用できるユースケースポイント法（1993.　G.Karner）[3] が適応されている．また詳細見積もりでは，ユーザーストーリーポイント（USP）をボトムアップ法でさらにタスク（チケット）に分解しその工数を見積もる方法が，プロジェクト計画時，また各イテレーション開始時に実施される.その時点では前イテレーションまでの残タスクや要件変更への調整が行われる。

今回のB社とプロジェクトとSプロジェクトのデータの見積もり状況を以下に示す．

表 2 プロジェクト見積もりデータ状況
Table 2 Aspects of estimation data of the projects

実施時期	単位	見積内容	B社プロジェクト	Sプロジェクト
プロジェクト計画時	ユースケースポイント	(1) 概算見積(全体①)（UCP法）	―（経験による工数算出）	○
プロジェクト計画時	ストーリーポイント	(2) 概算見積(全体②)（各ストーリーポイント累積）	―	○
各イテレーション開始時	タスク（チケット）	(3)詳細見積（イテレーション内各タスクの工数累積）	○	○
各イテレーション開始時	タスク（チケット）	(4)詳細見積（イテレーション内各タスクの工数累積）	○	○

S プロジェクトでは，その教育目的もありユースケース図を作成しユースケースポイント法による見積もりを実施している．B 社は，実際の開始時には経験による期間・要員ベースで見積もりもられた為，全体の特定の手法による見積もりは実施していない．この様に，同じ Ruby によるアジャイル開発もその見積もり･管理方法は一様ではない．

4. 分析

4.1 UCP法概算見積もりと実績値の比較

全体見積もりと実績の比較の目的は計画時見積もりの分析による精度の向上にある．全体概算見積もりは，S プロジェクトにて USP 法とストーリーポイントの合計で実施されている．USP 法での見積もりはその前提として，ユースケースモデルを作成しユースケース図，ユースケース記述を行いそのアクタ，ユースケースの重みづけでポイントを設定する．
その手順と S プロジェクトの数値は以下のとおりである．

(1) Use Case 図の作成
(2) Inter Face による Actor の重みづけ（複雑/平均/単純）
(3) 処理数(Transaction)による機能(Use Case)の重みづけ（複雑/平均/単純）
(4) 調整前の機能量(UUCP)の算出
(5) 技術要因（TCF）と環境要因（EF）の重みづけ
(6) UCP の算出
UCP = 調整前の機能量（UUCP）×技術要因係数（TCP）×環境要因係数（EF）
● S プロジェクト ＝ UUCP × TCP × EF
＝174（UUCP）× 0.89（TCP）× 0.95（EF）
(7) 危機指数の割り出し
● S プロジェクト＝ 危機指数（2）
1 UCP に対応する人時間・・20
工数＝ UCP × 20 ＝ 2942 人時間
1 日 8 時間とした場合 367.8 人日
1 か月 160 時間とした場合 18.39 人月
＜実績工数＞ 1704 人時間

以上から約 42%の乖離があったことがわかる．この数字は決して高精度とは言えないが，この S プロジェクトひとつで断定することは難しい．

4.2 ストーリーポイントによる全体見積もり

ストーリーポイントでの全体見積もりも S プロジェクトのみで実施されている．ストーリーは，“要件”であり，B 社では各イテレーションの開始前にオーナー（顧客）とのストーリーの概要を設定し，そのポイントを 1, 2, 3, 5, 8 の数字で設定する．
ストーリーはプロジェクト進行過程で各イテレーション開始前に更新される．変更推移を以下のグラフに示す．

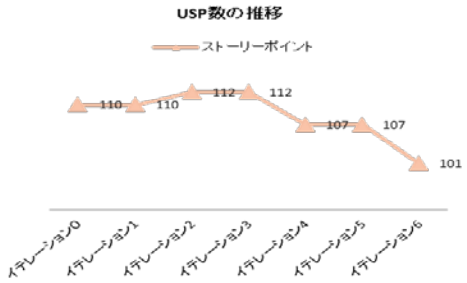


図 2 USP 数の推移
Figure 2 Change of Number of USP

4.3 詳細見積もりの精度

イテレーション毎の詳細見積もりは，B 社，S プロジェクトともに実施している．但し，イテレーション期間は，B 社プロジェクトでは，1 か月だが，S プロジェクトでは 1～2 週間と異なる．また，S プロジェクトにはプロジェクト開始時に所要時間の計測を依頼できたが，B 社プロジェクトでは各タスク・ストーリー完了までの所要時間がわからなかった為, 分析できていない. B 社プロジェクトでは，各イテレーション開始前に顧客とそれぞれのストーリー（要件）を固め，ストーリー毎に 1 枚の付箋紙にその概要を記入した．そしてそのストーリーのポイント数をその付箋紙の裏側に記入し，後でシステム（このプロジェクトでは Excel）に移行するという手順で実施している．
B 社プロジェクトの各イテレーションの予想／実績工数比較を以下の表 3 に示す．

表 3 B 社プロジェクト予想・実績工数比較
Table 3 Comparison of Plan and Actual man-hour for B company project

B社プロジェクト　　工数：単位人日

イテレーション（1か月）	予想工数	実績工数	乖離%
イテレーション1	17	13.9	22%
イテレーション2	23	26.1	▲12%
イテレーション3	26	25.8	0.8%
平均			11.6%

B 社プロジェクトの数値から，イテレーションが進む度に, 予想工数と実績工数の乖離が 22%から 0.8%と減少していることが読み取れる. プロジェクトが進行していくほど，工数見積もりの精度が高くなっていることが伺える．
次に，S プロジェクトの予想／実績工数比較を表 4 に示す．

表 4　S プロジェクト予想・実績工数比較
Table 4 Comparison of Plan and Actual man-hour for S project

Sプロジェクト　　工数：単位人時間

イテレーション（1-2週間）	予想工数	実績工数	乖離%
イテレーション0	75	75	0%
イテレーション1	135	135	0%
イテレーション2	414.5	414.5	0%
イテレーション3	187.5	181.5	0.8%
イテレーション4	42.8	37.8	16%
平均			0.16%

S プロジェクトでは，予想工数・実績工数がほぼ等しい数値である結果であるが，最後のイテレーション 4 では見積もり工数を下回る実績が出ていたことがわかる．

5. 評価

今回この 2 つのアジャイル開発プロジェクトのデータをアジャイル開発の見積もり精度向上を目的として分析した．

アジャイル開発の生産性の把握とそれを論理的に反映した初期見積もりの実現を目指すからである．今回の 2 つのプロジェクトでは表 5 の様な集計単位で予想・実績データが採集されていた．

表 5　プロジェクトデータの採集単位
Table 5　The Unit of Collected Project data

	Sプロジェクト		B社プロジェクト	
必要データ項目	見積	実績	見積	実績
規模（サイジング）	UCP	-	-	LOC
	ストーリーポイント	ストーリーポイント	ストーリーポイント	ストーリーポイント
工数	タスク	タスク	タスク	タスク
所要時間	-	タスク	-	-

しかし，これらのデータからは有効な生産性は得られなかった．その理由は次のとおりである．

- 生産性は（生産量／単位工数 or 単位時間）で求めるが生産量を何とするかが未定義であること，B 社では全体ステップ数の実績が採集されており全体の生産性実績は把握できたがイテレーション毎はとられてない．またアジャイル開発の生産量単位としてステップ数が適切か検討すべきところである．
- 2 プロジェクトに共通して規模算出（サイジング）の単位としてストーリーポイントが使われているが，ブレイクダウンしたタスクが工数算出の最小単位となっており，数式化されていないためにそのリンクが不明瞭であること，B 社プロジェクトではストーリーポイントそのものが工数である捉え方になっている．
- 所要時間の採集は S プロジェクトのタスクに対する実績のみであった．工数は人／時間で出ていたが，より詳細な計画を立てるために，タスク毎の所要時間が必要になる．

6. 今後の分析

この評価により生産性を求める上で，何のデータをどのタイミング採集しなければならないかを認識できた．

データ提供による協力を頂いた S プロジェクトは教育を目的としていることから，また B 社プロジェクトは，県の振興プロジェクトとして報告されていることから，このような計測データを頂けた．しかし，実際の企業がこれら必要データをすべて採集，管理されているか今までのヒアリングからも期待が薄い．そこで今後の進め方として新規プロジェクトへ以下のアプローチによる研究を進める．

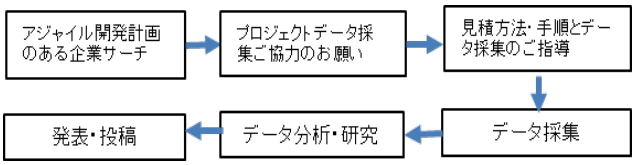


図 3　データ採集と分析手順
Figure 3　Process for　Data collection and Analysis

また操作の標準化の為に共通プロジェクト管理ツールとして“Redmine”（Web 上で動くオープンソースのアプリケーションフレームワーク）を活用し推進したい．

7. おわりに

この研究では今回のデータ評価にもとづいて，複数のプロジェクト事例から，アジャイルに適切な見積もりプロセスの有効性を検証し提案できることを目的としている．また，見積もりの精度や生産性を研究する上で変更・品質管理標準についても考察を広げてきたい，

謝辞

本研究にご協力いただいた，株式会社プロビズモ　女鹿田晃和様，筑波大学大学院コンピューターサイエンス専攻 SANITY プロジェクト永田達也様に深く感謝申し上げます．

参考文献

1)「ソフトウェア開発データ白書 2009」日経 BP 出版センター，2009 年
2)　A.J.Albecht, ”Function Point Analysis, Encyclopidia of Software Engineering, 1994
3)　G.Karner, ”Use Case Points – Research Estimation for Objectory Projects” Objective Systems SF AB (Rational Software) 1993