経済研究
Vol. 69, No. 1, Jan. 2018

小特集：人事と組織の経済学のフロンティア

ソフトウェア開発における早期すり合わせの効果と働き方改革への示唆

水上祐治・大湾秀雄

働き方改革の議論が進む中，生産性や質を高めながら残業を削減する方法が模索されている．本論文では，早期のコーディネーションを目的に開発プロセスを変更した自動車部品メーカーにおけるソフトウェア開発のプロジェクト進捗管理データを用いて，ソフトウェア開発におけるすり合わせ時期の前倒しの影響を定量的に計測した．取り組みの有効性をみるため，残業時間，製品の質，生産性への影響を検証した．開発プロセスの変更は，(1)労働時間の平準化を通じ残業時間を減少させ，(2)出荷後に発見される欠陥率を有意に引き下げたが，(3)開発期間中の仕様変更とコミュニケーションの頻度の著しい増加の結果，生産性には有意な影響を及ぼさなかった．興味深いことに，経験(勤続年数)による欠陥率(統合テストおよび出荷後)の低減効果は，プロセス変更後有意に小さくなった．早期すり合わせが仕事と労働者の生活の質を向上させると同時に，情報共有を通じたチームでの問題解決を容易にすることで，属人的な知識や経験への過度の依存を回避させたと解釈できる．
JEL Classification Codes: J24, L15, L62, M11

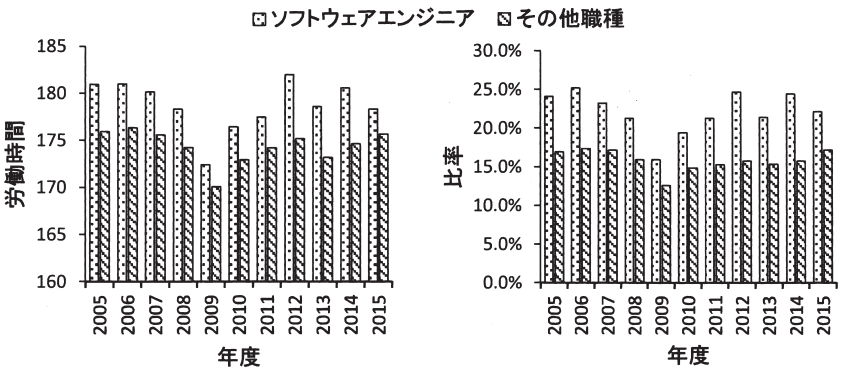
1. はじめに

働き方改革の議論の中で，システムエンジニア，プログラマーといったソフトウェア関連業務の長時間労働が話題となることが多い．賃金構造基本統計調査(賃金センサス)でシステムエンジニアおよびプログラマーの労働時間をそれ以外の非管理職と比べると，平均労働時間数で見ても月 200 時間以上の長時間労働者の比率で見ても，ソフトウェアエンジニアの労働時間が長い(図 1)．また，ソフトウェア関連業務の労働時間は，単に長さだけでなく，開発工程における不確実性の大きさや突然の不具合や顧客要請といった外的な要因によって引き起こされる残業の頻度の多さによっても特徴づけられるかもしれない．残業による疲労感は，予測可能かどうか，発生の集中度合いにもよるので，単なる残業時間合計数だけでは測れない部分もあるだろう．つまり，予め予想された残業よりも，当日や前日に突然発生する残業の方がストレスを感じるし，毎日少しずつ発生する残業よりも，3 日続けての深夜残業が 1 回発生する方がつらいと感じるかもしれない．

情報サービス業という産業あるいはソフトウェア・IT エンジニアという職種においてなぜ長時間労働が発生しやすいのかという点については，業務に関する次の 4 つの特徴が原因として挙げられる(厚生労働省 2014 年)．(1)プロジェクト・チームでの仕事が多い，(2)事前の仕様の確定が困難なケースが多い，(3)顧客先に常駐して業務に当たるエンジニアも少なくない，(4)開発プロセスの他社への委託が頻繁に起きる．これら 4 つの特徴すべてが，作業の進捗管理や製品の品質管理を難しくしている．つまり，企画やコミュニケーションが不十分な場合，その後の工程に影響が出て，時間外労働や休日出勤が増える結果になりやすい．また，仕組みとしての作業の進捗管理や製品の品質管理が十分でないということは，生産性や製品の質が個々人の経験やノウハウに依存する傾向を一層強めることになる．

一方で，関係者間の調整不足が無駄な作業ややり直しの発生を引き起こし，長時間労働につながっているのであれば，ソフトウェア・IT エンジニアという職種にあった働き方改革が必要であるということも言える．もし，作業プロ

図 1. ソフトウェアエンジニアの労働時間



(a) 月平均労働時間　　(b) 長時間労働者(月 200 時間超)比率

セスを見直し，顧客との早めのすり合わせ，仕様の早期決定，定期的なコミュニケーション等を業務効率化のために進めて行けば，長時間労働を避けることも出来るかもしれない．

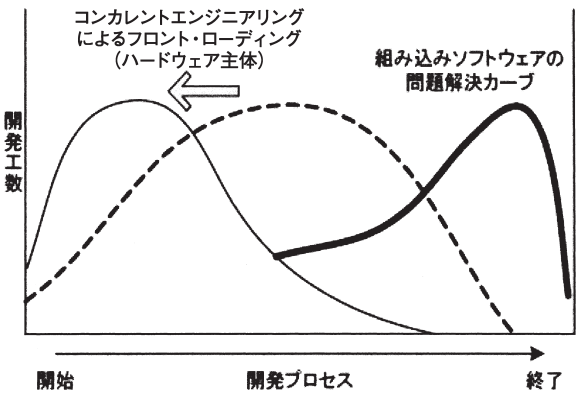
住友商事系 IT サービス会社の SCSK 株式会社は，2012 年に残業大幅削減，有給休暇取得率 100% を目指して，様々な取り組みを開始した．様々な努力によって，2-3 年で残業時間を半分近くまで削減しながら業績を改善させたことで知られている．同社は，システム開発の仕事のやり方を見直す中で，(1)システム開発を一人ではなくチームでの仕事に変え，(2)開発途中の複数段階でテストを実施し，(3)顧客との頻繁なコミュニケーションを通じて，進捗状況を共有し，新たな要望の吸い上げを図った．こうした業務プロセスの変更は，不具合や，最終段階で発生する新たな要望への対応によって引き起こされる作り直しを大幅に減らし，残業時間の削減に寄与したと言われている[1)]．

こうした業務プロセスの改善は，生産性や質を改善し，残業時間を減らす上でどの程度有効なのであろうか？また，早期コーディネーションという考え方は，生産性を上げる方法として，どの程度一般化出来ることなのであろうか？コーディネーションや情報共有といった組織内の活動が，より良い意思決定やコーディネートされた活動を通じて組織のパーフォーマンスを引き上げることは様々な研究で強調されてきた (Cremer 1981, Dessein and Santos 2006, Milgrom and Roberts 1992, Ochs 1995)．また，企業がコーディネーション能力を高めるためにチームを活用することも既に指摘されている (Kato and Owan 2011)．しかしながら，コーディネーションの仕方を改善することで，実際にどの程度企業の(質を考慮した)生産性に変化が生じうるのか検証した実証研究は管見の限りまだない．

本論文の目的は，早期のコーディネーションを目的に開発プロセスを変更した AMRD 社のプロジェクト進捗管理データを用いて，ソフトウェア開発における顧客とのすり合わせ時期の前倒しの影響を定量的に計測することである．生産性，製品の質，残業時間への影響，そしてその裏にある活動の記録を検証することで，こうした取り組みの有効性を検証する．

Ulrich(1995)や藤本(2001)らが導入した製品アーキテクチャの観点から，本研究が対象とする AMRD 社を見ると，ソフトウェア業務全体に一般化できる設定とは言い難い．ソフトウェアの製品アーキテクチャは，一般に機能ごとの切り離しが可能なモジュラー型(組み合わせ型)であろう．一方，AMRD 社が製品を供給する自動車業界はインテグラル型(すり合わせ型)である．インテグラル型産業のサプライチェーンに入ったソフトウェア業務は，製品開発プロセスや人のマネジメントにおいても修正を迫られることは十分予想できる(都留・守島 2012)．実際，AMRD 社が開発プロセスの変更を行った背景には，フェーズごとにコーディネーションが入る開発プロセスにおいて，メーカー側の

図 2. 組み込みシステムの問題解決カーブ



出典）　佐伯(2008).

期間短縮要求に応えることが出来なくなったという事情がある．しかしながら，コーディネーションが頻繁に発生する仕事のプロセスにおいて早めのコーディネーションがどのような影響をもたらすかについて，ある程度の示唆を得ることが出来たと考える．

2. 自動車産業における車載ソフト開発

自動車産業におけるソフトウェア開発の特徴を３つ挙げる．まず，不具合が発生すると製品自体の交換につながるため，費用が大きく，質に対する要求が高いということが言える．また，生産や販売の開始時期の遅れも大きな費用や機会損失を発生させるため，納期の厳守と質の要求の間のコンフリクトは大きい．このことは，他のソフトウェア開発，例えばアップデートを通じた修正が可能なパーソナルコンピュータ，携帯電話，ゲームアプリケーション用ソフトウェア開発などと比較して，長時間労働が発生しやすい環境と言える(藤本 2003)．

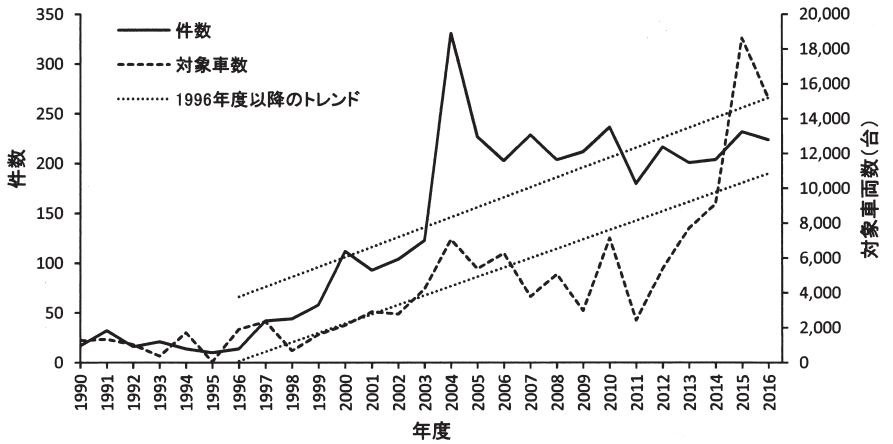
２つめの特徴として，ハードウェアとソフトウェアの設計が同時に進むため，ソフトウェア開発においては仕様変更が生じやすいということも挙げられる．つまり，自動車開発の期間短縮化でハードウェア開発作業のフロントローディング(前倒し化)が進んでいるため，開発末期で生じる問題はすべて設計変更が容易なソフトウェアで解決しようとする傾向が生じる[2)]．その結果，ハードウェアとは逆に，ソフトウェア開発のバックローディング傾向が強まっている(図 2)．

３つ目の特徴として，ハードウェア同様，企業間のすり合わせが重視される．これにはいくつかの要因がある．まず．車に使われる電子制御ユニットの増加に対する自動車メーカーの対応能力は限定的であり，サプライヤのソフトウェア開発技術力を積極的に利用する必要があることが挙げられる．また，技術的不確実性に対応するため，開発初期段階での企業間のフィードバックや情報共有が品質改善のために不可欠である．さらに，開発期間短縮化の要求は強まっており，機能設計の段階でサプライヤのインプットを入れることで，モジュール化努力と柔軟性維持の間に生じるトレードオフを緩和することが出来る．

２つ目の特徴として指摘したソフトウェアのバックローディング化，つまり開発末期における仕様変更は，車載ソフトの品質低下の原因となってきている．実際に，近年車載ソフトの質の低下が問題となりつつある．図３で示されるように，2000 年代に入って，リコール件数，リコール対象車種共に高水準で推移している．国土交通省(2004，2008，2012，2016)によると，リコールの２分の１から４分の３が開発段階における問題に起因するものであった．評価基準の低下，開発期間の短縮化，担当者の業務多寡による疲弊などが原因として挙げられている．

こうした質の低下に対処する上でも，企業間

図 3. 国産車のリコール件数の推移



すり合わせは重要になってくる．Gokpinar, Hopp and Iravani(2010)らは，米国自動車メーカーの内部データを用いて，製品設計の要求するコーディネーションのレベルに実際のコミュニケーションが追い付かないと，補償期間中の修理要求件数で測った商品の質は低下することを示した．ソフトウェアのバックローディング化の問題は，ソフトウェア開発チームの機能間，部品間のインターフェースにおける調整が十分に行われなくなってきている可能性を示唆している．早目にコーディネーションを行うことで，ソフトウェア・コードの製作(コーディング)に早目に着手出来るし，不具合の早期発見にもつながる．ソフトウェア開発のフロントローディング化の必要性は，産業全体の共通認識であると言える(Mizukami and Owan 2010)．

3. AMRD 社におけるソフトウェア開発モデルの変更

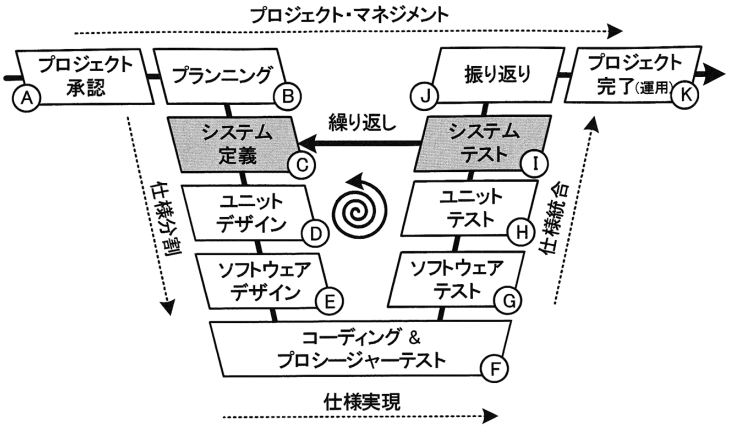
本研究では，米国自動車部品メーカー AMRD 社を対象とする．AMRD 社は，日本の自動車メーカー 3 社の 1 次サプライヤであり，2001 年に日本におけるソフトウェア開発を開始した．AMRD 社では，製品基本性能に関するコア部分の開発は米国本社で，顧客専用機能に関わる周辺アプリケーションの開発は日本支社が担当している[3]．

AMRD 社におけるソフトウェア開発は，2005 年までは TIV(Traditional Iterative V-Model)モデルに沿って行われた．TIV モデルを図 4(a)に示す．このモデルは，全体の機能設計，詳細設計，コーディング，テストという工程を順を追って進めるウォーターフォールモデルをベースにした反復型 V モデル(V-Model 97 1997)を使用しており，仕様作成工程とテスト工程が関連付けされ相互に情報を照合する仕組みとなっているのが特徴である．仕様に不確実性があるので，フェーズごとに反復する．全体の進捗状況が把握しやすいウォーターフォールモデルをベースとしつつも，フェーズに分けた反復的な開発を行うことで不確実性に対応しやすくしたという特徴を持つ．

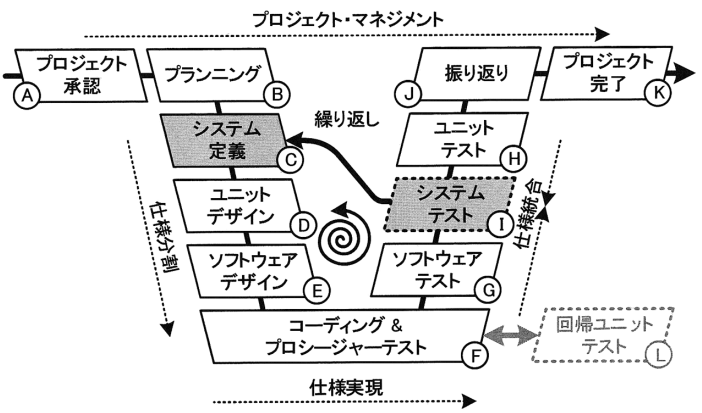
しかしながら，TIV モデルでは，次第にメーカーからの開発期間短縮要求に応えられず，プロセスの変更を提案することになった．その内容は，アクティビティ順序を変更することで，サプライヤ間の早めのすり合わせを行うということである．具体的には，プロトタイピング設計法と同様に自社製品の機能を他のサプライヤに見せることを優先して，部品単体でのテストと，自動車の機能を決定するシステムのテストの順番の入れ替えを行った[4]．AMRD 社では，この新しい開発プロセスは CLIV(Chain-Linked Iterative V-Model)モデルと呼ばれた[5]．CLIV モデルを図 4(b)に示す．

従来，フェーズの終わりに行っていたシステムテストでは，サプライヤが実際に部品を持ち寄り，つないでシステムとしての性能を確認す

図 4. 開発プロセスの変化 （TIV モデルから CLIV モデルへ）



(a) TIV モデル



(b) CLIV モデル （コンカレント化されたプロセス）

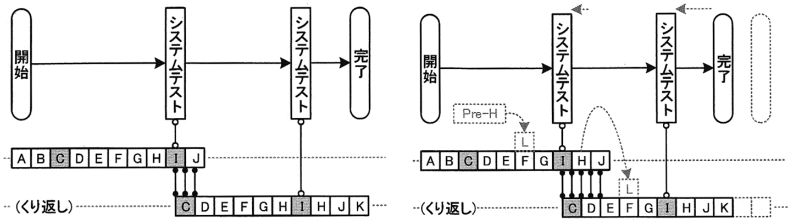
る．ここで，サプライヤの理解不足，仕様のグレーゾーンやサプライヤ間の仕様理解の不一致が顕在化し改善点が明確になる．しかし，そのために前行程に立ち戻ってのやり直しがしばしば発生する．

システムテストを早めに行うことは，余裕を持って修正や仕様変更に対処することを可能にした．また，自動車メーカー，サプライヤ間の情報共有を促進し，改善検討の機会を多く確保することも可能にした．加えて，システムテスト以降，次のフェーズの開始が可能になるため，現サイクルと次期サイクルの工程が同時進行し，これら工程間にて，プロジェクト・チーム内の情報共有が促進される．そのため，前工程に戻って修正する必要が発生するリスクが軽減された(図 5)．

このようなプロセス変更は，次のような変化をもたらすことが予想された．

①他の組織との理解の不一致を早期発見し，仕様が早期に確定することで，ソフトウェア開発のフロントローディング(前倒し)が進む．その結果，労働時間の平準化が進む．
②残業が減り，疲労が蓄積せず，余裕を持って修正作業に取り組めるため，質が向上する．
③従来は，経験を重ねるほど他社の期待を予見できるようになるので手戻りが減っていたが，早めのすり合わせにより属人的な経験への依存度が低下する．つまり，勤続年数が生産性や質に与える影響つまり企業特殊的人的資本のリターンが低下する．

図 5. プロセスフローの変化



(a) TIV モデルのプロセスフロー

(b) CLIV モデルのプロセスフロー
(コンカレント化されたプロセス)

A: プロジェクト承認
B: プランニング
C: システム定義
D: ユニットデザイン
E: ソフトウェアデザイン
F: コーディング & プロシージャーテスト
G: ソフトウェアテスト
H: ユニットテスト
Pre-H: 過去プロジェクトのユニットテスト
I: システムテスト
J: 振り返り
K: プロジェクト完了
L: 回帰ユニットテスト
: 車両レベルのタスク
: プロジェクト内の情報共有
: サプライヤ間の情報共有
: スクリプト化 & 情報展開
: 補助線

(c) 各アクティビティの詳細

表 1. プロジェクトの時期，タイプ，サイズ

ID	プロジェクト開始日	タイプ	フェーズ数	期間[週]	参加人数
1	2003 年 1 月 5 日	TIV	4	26	4.192
2	2003 年 6 月 5 日	TIV	4	22	4.545
3	2003 年 7 月 20 日	TIV	3	18	4.389
4	2003 年 9 月 7 日	TIV	3	16	4.488
5	2003 年 11 月 7 日	TIV	3	17	4.388
6	2004 年 2 月 8 日	TIV	3	17	4.482
7	2004 年 5 月 23 日	TIV	2	12	4.917
8	2004 年 7 月 11 日	TIV	1	5	4.840
9	2004 年 8 月 15 日	TIV	2	12	4.217
10	2004 年 8 月 22 日	TIV	2	13	4.662
11	2004 年 10 月 17 日	TIV	2	11	4.309
12	2004 年 12 月 16 日	TIV	3	18	4.500
13	2005 年 1 月 23 日	TIV	2	11	4.891
14	2005 年 2 月 27 日	TIV	3	13	4.877
15	2005 年 6 月 5 日	TIV	2	17	4.529
16	2005 年 7 月 24 日	TIV	3	23	4.209
17	2006 年 1 月 8 日	CLIV	4	20	3.880
18	2006 年 3 月 26 日	CLIV	3	16	3.838
19	2006 年 5 月 7 日	CLIV	4	17	3.706
20	2006 年 8 月 20 日	CLIV	3	12	4.100
21	2006 年 9 月 17 日	CLIV	3	15	4.053
22	2006 年 10 月 8 日	CLIV	5	23	4.067
23	2006 年 11 月 12 日	CLIV	3	17	4.035
24	2007 年 2 月 18 日	CLIV	4	16	4.575
25	2007 年 5 月 6 日	CLIV	2	12	4.650
26	2007 年 5 月 27 日	CLIV	4	16	4.413
27	2007 年 8 月 19 日	CLIV	4	17	5.318
28	2007 年 10 月 18 日	CLIV	2	9	5.444
29	2008 年 1 月 20 日	CLIV	5	16	4.369
30	2008 年 6 月 8 日	CLIV	4	21	5.467

TIV: Traditional Iterative V model, CLIV: Chain-Linked Iterative V model.

4. データおよび分析方法

本論文の実証研究では，AMRD 社のプロジェクト進捗管理データを用いる．6 年間にわたる 30 プロジェクトのデータを含み，そのうち TIV モデルを用いたのは，2003 年 1 月から 2005 年 12 月までの 3 年間の 16 プロジェクト，CLIV モデルを用いたのは，2006 年 1 月から 2008 年 12 月までの 3 年間の 14 プロジェクトである(表 1)．各プロジェクトは，2-5 のフェーズから構成され，分析で用いるのはフェーズ単位の記録で，観測数は 91 である．

すべてのプロジェクトは，同一のプロジェクトリーダーが管理し，TIV モデルから CLIV モデルへの変化以外は，技術や製品車種の大きな変化はなかった．

開発プロセス変更の残業時間への影響を見るために，プロジェクトメンバーの平均週残業時間の試算も行った．これは，プロジェクト管理システムのデータに基づくもので，実際の個人ベースの出退勤情報に基づくものではないため，かなりの計測誤差を含むものであることに注意する必要がある．具体的には，以下のス

テップを踏んで計算した．

①各プロジェクトの週別作業時間投入量をもとに，各週の一人当たり 1 日平均作業時間を計算する．これは個人ベースではなく，開発部門全体の平均である．
②残業時間＝max{一人当たり平均一日作業時間＋4.5，7.75}－7.75 で残業時間を計算し，それに勤務日を掛け合わせて，平均週残業時間を計算する．ここで，4.5 というのは，標準週のチームメンバーのカレンダーに基づき，ミーティング，打ち合わせ，事務作業など，プロジェクト業務以外に費やした時間数の平均である．7.75 時間は所定内労働時間数である．
③プロジェクトごとの残業時間を算出するため，その週に走っているプロジェクトの作業時間の比率で残業時間もプロジェクトに割り振る．
④最後に，フェーズごとの残業時間変数を作成するため，③でプロジェクトに割り振った週単位の平均週残業時間を使って，各フェーズ期間中の平均を取る．その際に，そのフェーズの週ごとの総作業時間をウエイトとして，加重平均を取る．ただし，フェーズが重なる週は，1 週であれば前段フェーズに振り分けて計算し，2 週重なる場合は，それぞれ 1 週割り振った．

実際には，チームメンバーによって労働投入量は異なるし，プロジェクト業務以外に費やす時間も変動するため，計測誤差はかなりある．それでも，この指標を用いて，実際に生じた残業の動きをほぼ追うことが出来ると考えた．

開発プロセス変更がもたらした開発チームの残業時間や成果物の質や生産性への影響を見るため，以下の式で表された OLS(式 1)もしくは Tobit 回帰式を推定する．

$$ln(y_{ij}) = \alpha + X_{ij}\beta + \gamma New\_V\_Model_i + \varepsilon_{ij} \quad (式\ 1)$$

y_{ij} は，プロジェクト i，フェーズ j の評価指標であり，プロジェクトによって引き起こされた週平均一人あたり残業時間の推定値（$Overtime_{ij}$），質の指標として，システムテストにおけるコード 1000 行あたり欠陥率（$Integration\_Test\_Defect\_Density_{ij}$），初出荷後 1 年以内に確認されたコード 1000 行あたり欠陥率（$Market\_Defect\_Density_i$）の 2 つ，生産性指標として，1 時間当たりの開発コード数（$Productivity_{ij}$）を用いる．出荷後欠陥率は，プロジェクトレベルの変数であるが，フェーズ単位のサンプルで推計を行う．すべての推計において，プロジェクト内でクラスタリングしたクラスター頑健標準偏差を用いて評価を行う．残業時間や欠陥率を被説明変数とする分析では，ゼロを下限とする Tobit モデルを用いる．

$New\_V\_Model_i$ は，当該プロジェクトが TIV である場合には 0 を，CLIV である場合には 1 を取るトリートメント変数である．X_{ij} は，フェーズレベルの変数であり，チームメンバー数（$Team\_Size_{ij}$），同時並行実施プロジェクト数（$N\_of\_Concurrent\_Projects_{ij}$）要求仕様数（$N\_of\_Specifications_{ij}$），仕様変更数（$N\_of\_Change\_Requests_{ij}$），平均勤続年数（$Tenure_{ij}$）を含む．チームメンバー数は，会社がプロジェクトに実際に投入したリソースである．ただし，実施するプロジェクトが増えてくると，複数プロジェクトにかかわるメンバーが増えてくるので，実際の投入時間は減少してくる．したがって，労働投入量をコントロールするために，チームメンバー数と同時並行実施プロジェクト数の両方が必要となる．要求仕様数は，受注時に要求される製品機能によって規定される仕様の数であり，プロジェクトが要求する仕事量あるいは労働投入量と言える．ただし，プロジェクトのすり合わせの中で，メーカーから新たな要求が追加されたり，グレーゾーンだった部分が明確になる中で仕様が確定され，仕様変更が行われる．この数も仕事量あるいは必要な労働投入量を決めるので，コントロールする必要がある．

生産性や質に影響を与えるメンバーの人的資本を捉えるため，平均勤続年数を加えた．車載

表 2. 基本統計量

従属変数		平均	中央値	標準偏差	最大	最小
Overtime		5.382	4.481	4.646	21.287	0.000
	TIV	6.658	6.419	4.644	21.287	0.000
	CLIV	4.335	2.966	4.423	21.152	0.000
Integration test defect density		1.390	0.000	5.679	41.270	0.000
	TIV	2.046	0.000	7.570	41.270	0.000
	CLIV	0.851	0.000	3.436	24.194	0.000
Market defect density		0.233	0.186	0.183	0.719	0.000
	TIV	0.353	0.360	0.190	0.719	0.105
	CLIV	0.134	0.148	0.101	0.296	0.000
Productivity		10.211	8.514	5.862	35.465	1.965
	TIV	10.594	10.384	4.238	25.810	4.432
	CLIV	9.897	7.596	6.945	35.465	1.965
説明変数						
New V Model		0.549	1.000	0.500	1.000	0.000
	TIV	0.000	0.000	0.000	0.000	0.000
	CLIV	1.000	1.000	0.000	1.000	1.000
N of members		3.780	4.000	1.093	5.000	2.000
	TIV	3.854	4.000	1.014	5.000	2.000
	CLIV	3.720	4.000	1.161	5.000	2.000
N of concurrent projects		2.317	2.500	0.639	3.250	1.000
	TIV	2.380	2.667	0.537	3.000	1.429
	CLIV	2.266	2.400	0.713	3.250	1.000
N of specifications		783	266	1150	5544	1
	TIV	1039	350	1408	5544	13
	CLIV	572	221	843	4056	1
N of change requests		0.879	0.000	1.604	7.000	0.000
	TIV	0.512	0.000	1.227	5.000	0.000
	CLIV	1.180	0.000	1.815	7.000	0.000
Tenure		1.869	1.777	0.593	3.157	0.857
	TIV	1.606	1.624	0.472	2.493	0.857
	CLIV	2.084	2.102	0.600	3.157	1.092

TIV: Traditional Iterative V model, CLIV: Chain-Linked Iterative V model.

電子制御ユニットのソフトウェア開発経験とメーカーや他のサプライヤとのすり合わせ経験が，生産性や仕事の質を決める重要な要因なので，この変数は多くのモデルで説明力を持つ．特に，メーカーのニーズや協働するサプライヤの期待を正しく読み取る力は，誤解に基づくやり直しや前工程への戻りを減らす上で必要となってくる．他方，当初推計に含めていた学校卒業後の総経験年数はほとんど説明力を持たなかったので，説明変数から落とした．

フェーズごとに，業務の内容が若干変化することを考慮して，第1フェーズダミー，最終フェーズダミーをコントロール変数に加えたり，フェーズ率＝現在のフェーズ番号/(フェーズ総数＋1)とその二乗を加えたりしたが，どのやり方でも結果にはほとんど違いはない．本稿では，後者を採用した推計結果を紹介する．表2に主要変数の基本統計量をまとめた．

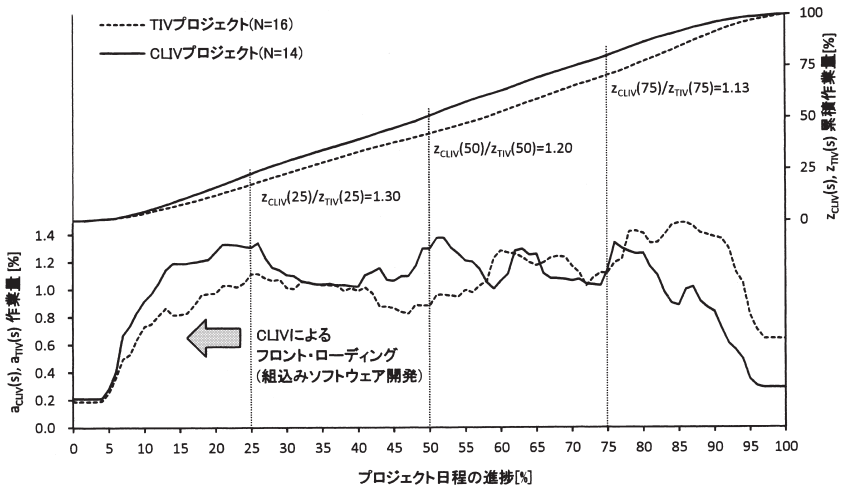
5. 実証分析

5.1 労働時間は平準化出来たか？

プロセス変更の目的が，作業の前倒しであるならば，まず実際に作業のフロントローディングが進んだかどうかを確認する必要があるだろう．プロジェクトの各週の作業時間をもとに，プロジェクト開始から終了までの作業時間の分布がどのように変化したかをまず見てみよう．

実施期間は，プロジェクトによって大きく異なるので，プロジェクトkの実施期間の経過をプロジェクト開始から当該週までの週数t_k

図 6. ソフトウェア開発の作業量の推移



ソフトウェア開発の作業量の推移：「システム定義」から「振り返り」までの作業量
作業量[%]：プロジェクト日程の進捗 1% 当りの作業量/総作業量

をプロジェクト開始から終了までに要した週数 T_k でわった t_k/T_k で表す．また，プロジェクト k の進捗状況をプロジェクト開始から t_k 週目までに行った作業時間合計 x_k をプロジェクト終了時までの総作業時間 X_k でわった x_k/X_k で示す．

まず，T_k 週分の観測値の間を線形補間することで，プロジェクト k の日程の進捗と作業の進捗の関係を関数 f_k で式 2 のように表す．

$$\frac{x_k}{X_k} = f_k\left(\frac{t_k}{T_k}\right) \qquad (式\ 2)$$

次に，TIV プロジェクト，および CLIV プロジェクトの平均進捗関数をそれぞれ式 3，式 4 のように計算する．

$$z_{TIV}(s) = \sum_{k\in TIV} \frac{f_k(s)}{\#\,TIV} \qquad (式\ 3)$$

$$z_{CLIV}(s) = \sum_{k\in CLIV} \frac{f_k(s)}{\#\,CLIV} \qquad (式\ 4)$$

すると，日程の各パーセンタイルにおける限界作業量は，それぞれ式 5，式 6 のように表せる．

$$a_{TIV}(s) = z_{TIV}(s) - z_{TIV}(s-1) \quad (式\ 5)$$
$$a_{CLIV}(s) = z_{CLIV}(s) - z_{CLIV}(s-1) \quad (式\ 6)$$

この方法を用いて，作業量が以前よりもフロントローディング化されたかどうかを確認したのが図 6 である．実際に，6 週目から 31 週目ぐらいまでは，CLIV の方が作業量は多いのに対し，78 週目からプロジェクト終了までは TIV に比べ CLIV の作業量は大きく低下している．作業がフロントローディング化したことが確認できる[6]．

作業時間の平準化が進んだのであれば，残業時間はプロセス変更後に減少したのではないだろうか？　前述した方法で，プロジェクト作業時間に基づき計算した推定平均労働時間の推移を示したのが図 7 である．9 時に出社した社員が，職務規定に沿って 45 分の昼休みと残業前 15 分の休憩を取った上で，10 時以降の深夜残業に入る閾値が 12 時間であるため，そこに基準線を引いた．この図によると，TIV プロジェクトの場合には頻繁に起きていた深夜残業が，CLIV モデルに変更になってから，大きく低下していることがわかる．もちろん，これは平均で見た頻度であるため，各社員の退出時間にばらつきがあれば，深夜残業の頻度はより大きくなることが予想される．

残業時間全体ではどうだろうか？　前述の方法で，残業時間を各フェーズに割り振った指標を被説明変数とし，Tobit モデルの推定を行った結果が表 3 である．ベースモデル(Model 1)

図 7. 深夜残業の発生頻度の比較

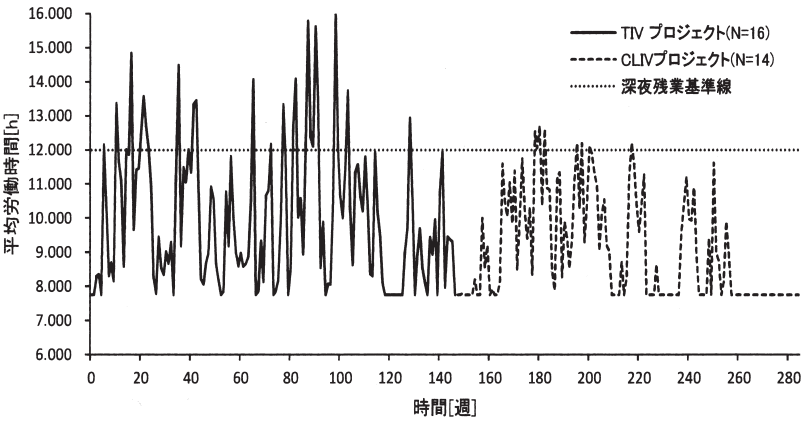


表 3. 回帰分析結果：週残業時間に与えた影響

VARIABLES	Model 1 Base	Model 2 Control for job demand	Model 3 Interaction with tenure	Model 4 Drop after financial crisis
New V Model	−2.868**	−2.166*	−2.098**	−1.953**
	(1.233)	(1.125)	(1.029)	(0.920)
*New V Model*ln(tenure)*			1.859	
			(3.611)	
ln(N of members)	−1.333	0.297	0.445	−0.291
	(1.693)	(1.358)	(1.357)	(1.698)
ln(N of concurrent projects)		3.343*	3.328*	
		(1.785)	(1.746)	
ln(N of specifications)	0.770***	0.819***	0.824***	0.805***
	(0.238)	(0.213)	(0.211)	(0.234)
ln(N of change requests)	2.107*	2.006*	1.990*	1.790
	(1.180)	(1.130)	(1.115)	(1.274)
ln(tenure)	1.336	0.245	−0.775	0.304
	(1.768)	(1.612)	(2.554)	(1.651)
Constant	3.528	−1.840	−2.196	1.879
	(3.009)	(2.694)	(2.563)	(2.738)
Observations	91	91	91	80

Robust standard errors in parentheses
*** $p<0.01$, ** $p<0.05$, * $p<0.1$

では，開発プロセスの変更によって一プロジェクトあたり 3 時間近く残業を減らす効果があったことになる．平均して 2.3 プロジェクトが同時並行して走っているため，全体では，一人あたり週平均残業時間を 6 時間程度減らす効果があった．ただし，モデル 1 は，仕事の需要をコントロールしていないため，モデル 2 は同時並行して走るプロジェクトの数をコントロールした．このモデルによると，残業を減らす効果は 2.2 時間，各プロジェクトの効果を併せた全体では，5 時間程度週平均残業時間を減らす効果があることが分かった．

しかし，残業に与えた効果は実際にはもっと小さいと考えられる．後で示すように，CLIV モデルへの変更は，仕様変更数を有意に増加させたことが分かっている．実際に，表 2 の基本統計量の比較によると，TIV から CLIV への変化で仕様変更件数は，フェーズ単位で平均 0.51 から 1.18 へ増加している．この効果を Model 2 の推計結果に基づき計算すると 2.006 ×(*ln*(1.180)−*ln*(0.512))＝1.68 時間，間接効果を通じて残業時間を押し上げる効果があった．したがって，総効果は，−2.17＋1.68＝−0.49 時間，平均 2.3 プロジェクトが同時に走ってい

表 4. 回帰分析結果：システムテスト結果に与えた影響

VARIABLES	Model 1 Base	Model 2 Control for job demand	Model 3 Interaction with tenure	Model 4 Drop after financial crisis
New V Model	0.994**	1.278**	1.904***	1.752***
	(0.455)	(0.510)	(0.479)	(0.604)
*New V Model*ln(tenure)*			3.720***	
			(0.867)	
ln(N of members)	1.097*	1.830***	1.946***	2.100***
	(0.571)	(0.555)	(0.474)	(0.525)
ln(N of concurrent projects)		1.221**	1.245***	
		(0.585)	(0.387)	
ln(N of specifications)	0.0355	0.0540	0.0845	0.0341
	(0.116)	(0.120)	(0.107)	(0.121)
ln(N of change requests)	−0.0237	−0.0684	−0.211	−0.125
	(0.293)	(0.284)	(0.254)	(0.314)
ln(tenure)	−1.494**	−2.004**	−4.459***	−2.332***
	(0.613)	(0.815)	(0.926)	(0.753)
Constant	−1.305	−3.423**	−4.208***	−2.906*
	(1.385)	(1.481)	(1.412)	(1.559)
Controls for phase	Yes	Yes	Yes	Yes
Observations	91	91	91	80

Robust standard errors in parentheses. *ln* (*tenure*) is demeaned.
*** $p<0.01$, ** $p<0.05$, * $p<0.1$

る状況では，1時間強週平均残業時間を減らす効果にとどまる．

それ以外の変数は，同時並行で進むプロジェクトの数，要求仕様数の増加がすべて残業時間を増やす方向で働くという自然な結果を得た．

5.2　質は向上したか？生産性はどうなったか？

次に，システムテスト欠陥率（$Integration\_Test\_Defect\_Density_{ij}$），出荷後欠陥率（$Market\_Defect\_Density_i$）の二つの質の指標に与えた影響を見てみよう．まず，システムテスト欠陥率への影響を Tobit 回帰モデルを用いて分析した表4モデル1を見ると，開発プロセスの変更はシステムテストで発見される欠陥を約100%つまり2倍に押し上げる効果を持ったことがわかる．これは，ユニットテスト実施前にシステムのテストを行っている，あるいは通常よりも早い段階でテストを実施していることから，予想された結果である．それ以外の変数についても，大きなプロジェクトほど欠陥率が高く（$ln(N\ of\ members_{ij})$ の係数が正で10%水準で有意），勤続年数が増えるほど欠陥率が下がる（$ln(Tenure_{ij})$ の係数が負で5%水準で有意）といった結果は予想通りであった．プロジェクトメンバー数が増えるほどコーディネーションが難しくなるし，経験が長いほど，メーカーや他のサプライヤの期待を読み取ることが上手くなるためであると解釈できる．

この結果が，需要の変化によって生じたものではないことを確認するため，モデル2では，同時並行して走っているプロジェクト数の対数を説明変数に加えた[7]．受注が増えるほど，複数のプロジェクトが走り，掛け持ちで仕事をすることになるため，質への悪影響が懸念されるようになる．プロジェクトマネージャーへのインタビューによると，プロジェクトの納期が延期されることはなく，生産性が悪く納期が延期されたからプロジェクト数が増えるといった内生性はない．ここで，変数 $ln(N\_of\_Concurrent\_Projects_{ij})$ の係数は，5%の水準で有意であり，繁忙期には質が落ちることがわかる．これによって，開発プロセス変更（$New\_V\_Model_i$）の影響が下がることはなく，結果が頑健であるこ

表 5. 回帰分析結果：出荷後欠陥率に与えた影響

VARIABLES	Model 1 Base	Model 2 Control for job demand	Model 3 Interaction with tenure	Model 4 Drop after financial crisis
New V Model	−0.103***	−0.0784**	−0.0737**	−0.0832*
	(0.0336)	(0.0356)	(0.0325)	(0.0468)
*New V Model*ln(tenure)*			0.127	
			(0.102)	
ln(N of members)	−0.00633	0.0456	0.0520	0.0152
	(0.0621)	(0.0697)	(0.0661)	(0.0754)
ln(N of concurrent projects)		0.103*	0.0998*	
		(0.0529)	(0.0518)	
ln(N of specifications)	0.000454	0.00395	0.00410	0.00377
	(0.00661)	(0.00656)	(0.00675)	(0.00704)
ln(N of change requests)	−0.0446*	−0.0473**	−0.0479**	−0.0583**
	(0.0228)	(0.0198)	(0.0188)	(0.0248)
ln(tenure)	−0.272***	−0.307***	−0.374***	−0.290***
	(0.0558)	(0.0633)	(0.0751)	(0.0619)
Constant	0.209**	0.0534	0.0363	0.171
	(0.0892)	(0.134)	(0.128)	(0.110)
Controls for phase	Yes	Yes	Yes	Yes
Observations	91	91	91	80

Robust standard errors in parentheses. *ln* (*tenure*) is demeaned.
*** $p<0.01$, ** $p<0.05$, * $p<0.1$

とが確かめられた．次に，モデル 4 では，観測期間中に始まった金融不況による影響ではないことを確認するため，2007 年の金融危機後に始まったプロジェクトを落として，推計を行ったが，ここでも $New\_V\_Model_i$ は下がることなく，逆に係数は高めに出てきた．

次に，最終的な質の指標である出荷後に発見される欠陥の発生率を被説明変数とする推計結果を見てみよう．表 5 のベースモデルを見ると，開発プロセス変更は，製品の質を有意に改善させたことがわかる．$New\_V\_Model_i$ の係数は，−0.103 で 1% の水準で有意である．つまり，開発プロセスの変更で欠陥が 1 割減ったことになる．同時並行プロジェクト数で需要をコントロールした推計（モデル 2）も行ったが，欠陥率への影響は 7.8% 減へと若干下がるものの，依然 5% の水準で有意である．しかし，残業時間のケースと同様，CLIV モデルへの変更は，仕様変更頻度を有意に増加させるので，本当の効果はこれより大きいと考えられる．いずれのモデルでも $ln(N\_of\_Change\_Requests_{ij})$ の係数は有意に負であり，この経路による影響を含めると欠陥率低下はもっと大きい．先ほどと同様，TIV から CLIV への変化で仕様変更件数がフェーズ単位で平均 0.51 から 1.18 へ増加したことによる効果を Model 2 の推計結果に基づき計算すると $-0.0473\times(ln(1.180)-ln(0.512))=-0.0395$，つまり 4% 程度欠陥率を押し下げる効果があった．したがって，間接効果を含めたプロセス変更の影響は，12% 程度の出荷後欠陥率の低下である．

ここでも勤続年数の影響は，どのモデルにおいても有意に負であり，企業特殊的な経験を蓄積するにつれ，出荷後に発見される欠陥は減少することがわかる．

金融危機後に始まったプロジェクトをサンプルから落としても（Model 4）有意性に若干悪化が生じるものの，係数に大きな変化は見られない．

5.3 生産性は改善せず．何故か？

続いて，生産性（対数）への影響を最小二乗法で測った分析を表 6 に示す．$New\_V\_Model_i$ の係数は，有意ではないものの，−0.146 と負となった．残業時間の減少，質の向上というプラスの成果が表れる一方，生産性に関してはむ

表 6. 回帰分析結果：生産性に与えた影響

VARIABLES	Model 1 Base	Model 2 Control for job demand	Model 3 Interaction with tenure	Model 4 Drop after financial crisis
New V Model	−0.146	−0.125	−0.139	−0.157
	(0.129)	(0.150)	(0.139)	(0.184)
*New V Model*ln(tenure)*			−0.391	
			(0.463)	
ln(N of members)	−0.340*	−0.294	−0.325	−0.364*
	(0.170)	(0.196)	(0.205)	(0.211)
ln(N of concurrent projects)		0.0942	0.100	
		(0.201)	(0.182)	
ln(N of specifications)	0.147***	0.150***	0.149***	0.156***
	(0.0351)	(0.0345)	(0.0347)	(0.0383)
ln(N of change requests)	0.0865	0.0833	0.0867	0.109
	(0.0889)	(0.0875)	(0.0929)	(0.0985)
ln(tenure)	0.142	0.110	0.327	0.162
	(0.232)	(0.261)	(0.284)	(0.293)
Constant	2.193***	2.053***	2.127***	2.193***
	(0.408)	(0.452)	(0.498)	(0.469)
Controls for phase	Yes	Yes	Yes	Yes
Observations	91	91	91	80
R-squared	0.371	0.373	0.384	0.360

Robust standard errors in parentheses. *ln* (*tenure*) is demeaned.
*** $p<0.01$, ** $p<0.05$, * $p<0.1$

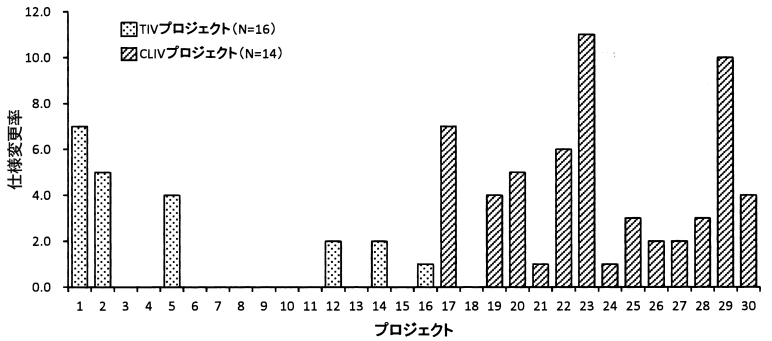
しろ低下に振れるケースも少なくないことが示された．残業時間の減少は，生産性の向上ではなく，単に作業の平準化によってもたらされたことになる[8]．もう一つ興味深いのは，プロジェクト・チームが大きくなるほど，生産性が低下する傾向が読み取れることである．コーディネーションのための打ち合わせやコミュニケーションミスによるやり直しなどによる生産性低下があるのかもしれないし，あるいは能力の高い人ほど少ない人数でプロジェクトを任せられるというセレクション効果が働くのかもしれない．表 4，表 5 と同様，需要の影響をよりコントロールしたモデル 2，モデル 4 を推計したが，開発プロセス変更の影響に変化は見られない．

早期コーディネーションによって，やり直しやミスを減らした兆候が見られる一方，生産性には改善をもたらさなかった理由はなんであろうか？　まず，早めに問題点や改善点を見つけることで，より大幅な修正が可能になったことが予想される．図 8 に見られるように，仕様変更頻度は，開発プロセス変更後大きく増加している．先にみた最終製品の質の完全は，こうした仕様変更によってもたらされた側面もあるのかもしれない．

さらに，早めのコーディネーションは，メーカーとの間やサプライヤ間での情報共有も促進し，プロジェクト期間中のコミュニケーションの量も増やした可能性がある．実際，図 9 に示されるように，AMRD 社のエンジニアとメーカーや他のサプライヤとの間で交わされた E メール送付件数は，CLIV モデルへの変更後大きく増加した．つまり，こうした手間をかけた結果，質が改善していたことになる．こうした観察結果は，前述の Gokpinar, Hopp and Iravani(2010) らの研究と整合的である．つまり，これまでは，ソフトウェア開発のバックローディング化の結果十分なコミュニケーションの時間が確保できず質の低下につながっていたが，早期コーディネーションの結果，インターフェースにおける他機能・他部品とのコーディネーションが製品設計が要求する水準に達したために質が向上したと解釈できる．

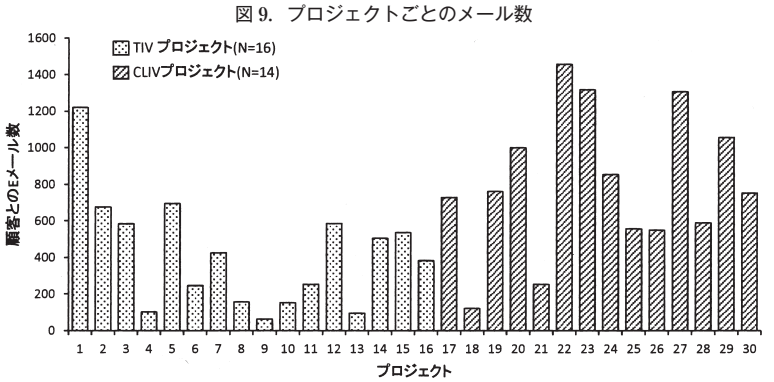
ただし，生産性への影響に関しては，仕事の質的変化を捉えられていない可能性がある．生

図 8. プロジェクトごとの遂行途中での仕様変更頻度の変化



仕様改善率：1000 仕様当りの仕様変更数

図 9. プロジェクトごとのメール数



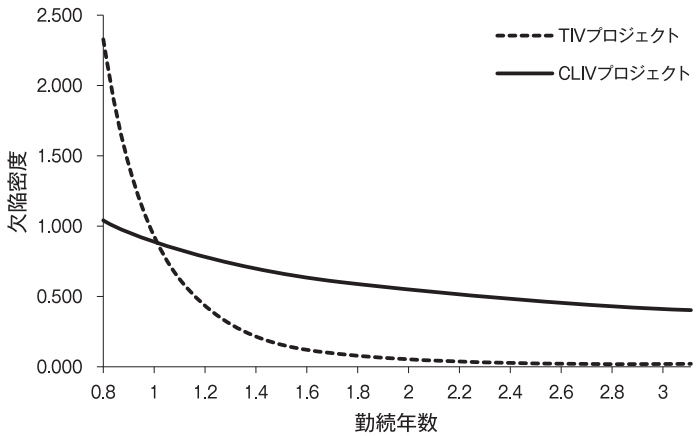
産性指標として 1 時間当たりの開発コード数を使っているが，これは難易度を考慮しない成果指標である．プロジェクトマネージャーへのインタビューによると，技術革新に伴い，高度な機能に取り組むケースが継続的に増えたという．こうした仕事の難易度の上昇という質的側面を考慮すれば，生産性は改善していたかもしれない．また，生産性への影響の分析は，短期的な影響しか捉えていない．ユニットテストを実施していない段階でのシステムテストは，ユニットレベルでの問題も結果に反映する可能性があるので，欠陥の原因を幅広く捉えて解釈する力を要求するだろう．作業者は，これら可能性を加味して，いつもよりレビュープロセスを多用するなど，検証作業を慎重に進めるであろう[9]．しかし，新しいシステムに慣れ，欠陥の原因をより効率的に予想できるようになると，当初の慎重な作業はそれほど必要ではなくなるかもしれない．その場合，生産性への影響は長期的にはプラスに転じる可能性もあるだろう．

5.4 勤続年数のリターンに変化が生じたか？

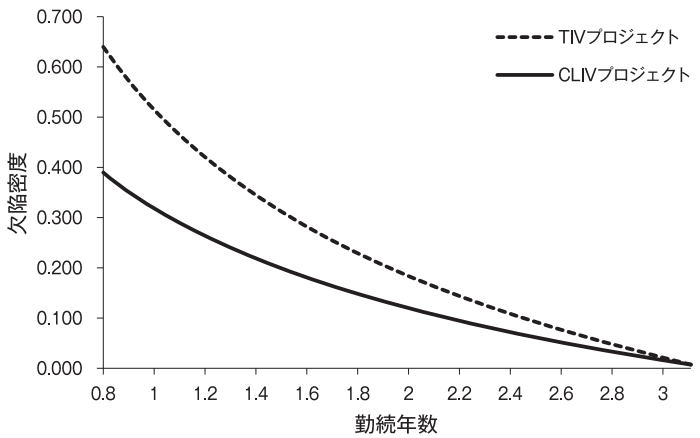
もう一つ，長期的な便益が期待できるのが，勤続年数と品質の間の関係の低下である．表 3 から表 6 のモデル 3 は，モデル 2 の説明変数に勤続年数(対数)と開発プロセス変更後ダミー($New\_V\_Model_i$)の交差項を加えたものである．有意に結果が出たのはシステムテスト結果への影響(表 4)だけであるが，いずれも経験効果を弱める方向で働いている．その大きさを確認するために，勤続年数とシステムテストの欠陥率，出荷後欠陥率(共にモデル 3 の推定値に基づく期待欠陥率)の間の関係を図示したのが図 10 である．いずれも経験による質向上の度合いは弱まっている．特にシステムテストにおいて，その効果が大きい．

経験に対するリターンが低いということは，離職率を下げるために，年功的な賃金を導入したり，必ずある一定以上の経験者がプロジェクトに入るようにチームを編成するといった必要性が低下することになり，人事面での制約を下げる効果がある．また，これまで経験者に過度

図 10. 勤続年数と品質



(a)システムテスト欠陥率との関係



(b)出荷後欠陥率との関係

に負担がかかっていたことも，未経験者である程度回せるようになるため，メンバー間での仕事の平準化が容易に出来るようになる．こうした変化は，長期的には，社内経験よりは技術力を重視した採用や配置を望ましいものとし，個人よりはチームに依存した運用を可能にするだろう．

6. 結論

結果をまとめると，TIV モデルから CLIV モデルへのプロセス変更は，(1)開発期間中の仕様変更とコミュニケーションの頻度を著しく引上げ，(2)結果，生産性を悪化させるケースも出現したが，(3)労働時間の平準化を通じ残業時間が減少し特に深夜残業がほぼ無くなり，(4)出荷後に発見される欠陥率を有意に引き下げ，(5)勤続年数の欠陥率(統合テストおよび出荷後)に与える影響を低下させた．早期すり合わせが仕事の質を向上させかつ残業時間の減少につながったと言える．また，早期のすり合わせは，属人的な経験への依存を回避させ，情報共有を通じたチームでの問題解決や相互の学習を促す働きも生じた．

プロセス変更による早期すり合わせは，仕様変更頻度で測った事後的なコーディネーションやメール送信数で測ったコミュニケーションも高めたことを確認した．また TIV モデルよりも CLIV モデルの方がすり合わせの密度は増したと考えられる．そうであるならば，評価した影響を解釈する場合にも注意が必要である．

Morita, Nakajima, and Tsuru(2017)によると，開発エンジニアのコーディネーション能力が高い場合には，製品アーキテクチャの統合度を高めることは製品の質の向上につながるが，コーディネーション能力が低い場合には，逆にシステム内の相互依存性の高まりから部品間の質のバラつきが製品全体の質を低下させやすくなる．今回 AMRD 社で製品の質が向上したのは，そこで働くエンジニアのコーディネーション能力が高かったからである可能性が残る．

今後の課題としては，現時点では，比較的短期的な効果しか測れていないということである．システムテストで生じた問題の解決能力が増したり，普段からサプライヤ同士のコミュニケーションが増すことによる調整能力向上など長期的な効果もあるかもしれない．また，CILV に対する経験不足で効果がまだ十分に出ていない可能性もある．観測期間末期や観測期間後は，金融不況の影響で離職や配置転換があったり，車種にも変更があったため，長期的な影響を測ることは極めて難しくなっている．

また，製品機能の高度化といった生産性の質的側面は測れていない．機能面での質の向上と欠陥率の低下を顧客が評価して価格引き上げを通じて収益の改善につながっているかどうかは重要な問題であるが，価格情報がデータに含まれていないため，こうした確認は取れていない．

最後に，本稿の一つの知見は，働き方改革を考える際には，職種ごとの属性や要件にあった対策を考えることが重要であるということである．早期のすり合わせが有効な職種は，ソフトウェア開発だけに留まらず，他にもあるだろう．複雑性，情報の非対称性，相互依存性が高い業務，たとえば研究，製品開発全般，プラント建設など複雑かつ不確実性の高い建設プロジェクト，M&A など投資銀行業務において，AMRD 社における取組みが役に立つ可能性がある．

（日本大学生産工学部，統計数理研究所・
東京大学社会科学研究所，経済産業研究所）

注

1) テレビ東京「カンブリア宮殿」(2017 年 1 月 12 日放映)での紹介より．

2) 日本の自動車産業では，効率化と最新の機能をタイムリーに市場投入する目的で，自動車開発の期間短縮化が進んでいる(舘岡 2002)(雨宮 2005)．特に，機構設計の三次元 CAD 化の研究が進み，ハードウェアの開発期間が短縮化された(福士 2006)．例えば，1980 年代，平均 30ヶ月であった新規の開発期間が，2000 年代後半には，一部車種で 10.5ヶ月にまで短縮されている(福士 2006)．

3) AMRD 社では，ソフトウェア・プロダクトライン開発(Boehm2004)と同等のソフトウェア・アーキテクチャを採用している．ソフトウェア・プロダクトライン開発(SPL)とは，ソフトウェアの中心機能をコア資産として整理して，それらを計画的に再利用する開発体制とソフトウェア・アーキテクチャのことを言う．一般的には，派生製品の開発が継続する場合，ソフトウェアの構造が複雑化して開発効率が低下するといわれている．SPL を適用すると，製品群の開発効率を長期的に安定維持させることができる．

4) プロトタイピング設計法は，システム開発の初期段階で，システムの振る舞いをユーザーにデモンストレーションするための試作品(プロトタイプ)を作成し，ユーザーと開発者の間で，システムに求められる性能や機能の認識を共有しようするものである．

5) CLIV は，反復型 V モデルにイノベーション・プロセス・モデルのチェーンリンクドモデル(Kline 1986)，アジャイル・プログラミングの Extreme Programing(Beck 1999)，プロトタイピング設計法を適用したものである．

チェーンリンクドモデルは，技術イノベーションの出発点は「市場の洞察・発見」であるとするモデルである．このモデルを車載ソフト開発にあてはめた場合，「システムテスト」が「市場の洞察・発見」に相当すると考えられる．Extreme Programing は，ソフトウェア開発プロセスの整合性よりもソフトウェア開発を取り巻く状況の変化を柔軟に取り入れて，開発の効率性を高めることを重視した考え方である．

6) フロントローディング化がフェーズ内で進行したかどうかを確認するため，同様な方法でフェーズごとの進捗関数を作り，その積分値を 1/2 から引くことで，フェーズごとのバックローディング指標を定義し，それを被説明変数とするような回帰分析も行った．予想されたように，TIV から CLIV への変更はバックローディング指標の低下をもたらしたが，有意な結果ではなかった．したがって，プロジェクト全体のフロントローディング化は，フェーズ内のフロントローディング化ではなく，早めに次のフェーズに着手できるという利点に由来するものかもしれない．

7) 需要要因をコントロールするため，全国新車販売台数を右辺に入れた推計も行ったが，有意ではなかった．最終需要と開発段階での発注の関係は，明確ではないし，AMRD 社の製品は，主に高級車や輸出車で使われる割合が多いため，販売台数合計では適切に需要要因をコントロールできないのかもしれない．

8) もう一つの可能性は，平行して走るプロジェク

トの数だけでは把握できない，一人当たり総業務量の減少が生じていた可能性である．

9)　レビュープロセスとは，変更結果などについて，複数の作業員が確認を行うプロセスである．

参考文献

雨宮正和(2005)「勝ち組企業の開発スピードアップへの取り組み」,『知的資産創造』野村総合研究所, Vol. 13, No. 5, pp. 76–80.

藤本隆宏(2001)「アーキテクチャの産業論」, 藤本隆宏・武石彰・青島矢一編『ビジネス・アーキテクチャ：製品・組織・プロセスの戦略的設計』有斐閣, 第一章, pp. 3–26.

藤本隆宏(2003)『能力構築競：日本の自動車産業はなぜ強いのか』, 中公新書.

福士敬吾(2006)「デジタルナレッジによる自動車の開発期間半減」『日本機械学会誌』, Vol. 109, No. 1054, pp. 740–741.

国土交通省(2004)「リコール届出内容の分析結果：平成 15 年度のリコール届出の傾向分析」国土交通省自動車交通局.

国土交通省(2008)「リコール届出内容の分析結果：平成 19 年度のリコール届出の傾向分析」国土交通省自動車交通局.

国土交通省(2012)「平成 22 年度自動車のリコール届出内容の分析結果について」国土交通省自動車局.

国土交通省(2016)「平成 26 年度自動車のリコール届出内容の分析結果について」国土交通省自動車局.

厚生労働省(2014)『働き方・休み方改善ハンドブック情報通信業(情報サービス業編)』.

佐伯靖雄(2008)「製品開発組織と開発プロセス：車載組込みシステム開発の設計と調整」,『立命館経営学』, Vol. 46, No. 5, pp. 193–219.

舘岡康雄(2002)「新車開発期間超短縮における“支援”の有効性に関する研究」,『品質』品質管理学会, Vol. 32, No. 3, pp. 370–380.

都留康・守島基博(2012)『世界の工場から世界の開発拠点へ　製品開発と人材マネジメントの日中韓比較』, 東洋経済新報社.

Beck, Kent (2005) *Extreme Programing Explained - Embrace Change*, Addison-Wesley, 1st edition, 1999, 2nd edition.

Boehm, Barry, Winsor Brown, Ray Madachy and Ye Yang (2004) “A Software Product Line Life Cycle Cost Estimation Model,” *ISESE–2004*, pp. 156–164.

Cremer, Jacques (1981) “A Partial Theory of the Optimal Organization of a Bureaucracy,” *The Bell Journal of Economics*, Vol. 11, No. 2, pp. 683–693.

Dessein, Wouter, and Tano Santos (2006) “Adaptive Organizations,” *Journal of Political Economy*, Vol. 114, No. 5, pp. 956–995.

Gokpinar, Bilal; Wallace J. Hopp, and Seyed M. R. Iravani (2010) “The Impact of Misalignment of Organizational Structure and Product Architecture on Quality in Complex Product Development,” *Management Science*, Vol. 56, No. 3, pp. 468–484.

Kato, Takao, and Hideo Owan (2011) “Market Characteristics, Intra-Firm Coordination, and the Choice of Human Resource Management Systems: Theory and Evidence,” *Journal of Economic Behavior and Organization*, Vol. 80, No. 3, pp. 375–396.

Kline, Stephen and Nathan Rosenberg (1986) *An Overview of Innovation, The Positive Sum Strategy, National Academy of Sciences*, pp. 275–306.

Milgrom, Paul, and John Roberts (1992) *Economics, Organizations, and Management*, Prentice-Hall, Inc..

Mizukami, Yuji and Hideo Owan (2010) “From Sequential to Integrated R&D Process: Estimating the Impact of a Process Change in Automotive ECU Development on Flexibility and Product Quality,” *Computers and Industrial Engineering IEEE - The 40th International Conference on Computers and Industrial Engineering Proceeding*, CD-ROM, 6 pages, Japan.

Morita, Hodaka, Kentaro Nakajima and Tsuyoshi Tsuru (2017) “Product Architecture and Intra-Firm Coordination: Theory and Evidence,” *Hitotsubashi University, Discussion paper*, A, No. 659, pp. 1–19

Ochs, Jack (1995) *Coordination Problems*, in John Kagel and Alvin Roth (eds.) *Handbook of Experimental Economics*, Princeton University Press.

Ulrich, Karl (1995) “The Role of Product Architecture in the Manufacturing Firm,” *Research Policy*, Vol. 24, No. 3, pp. 419–440.

V-Model 97 (1997), V-Model TX (2005) *German Directive 250–252*, Development Standard for IT Systems of the Federal Republic of Germany.