JAIST Repository
https://dspace.jaist.ac.jp/

Title	受託システム開発会社におけるよい戦略と分業構造
Author(s)	米澤, 英昭; 若林, 秀樹; 田村, 浩道
Citation	年次学術大会講演要旨集, 38: 350-354
Issue Date	2023-10-28
Type	Conference Paper
Text version	publisher
URL	http://hdl.handle.net/10119/19251
Rights	本著作物は研究・イノベーション学会の許可のもとに掲載するものです。This material is posted here with permission of the Japan Society for Research Policy and Innovation Management.
Description	一般講演要旨

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

１Ｅ０７

受託システム開発会社におけるよい戦略と分業構造

○米澤英昭，若林秀樹，田村浩道（東京理科大 ＭＯＴ）
8822242@ed.tus.ac.jp

1. はじめに

日本における IT 産業は、多重下請けの産業構造になっており２つの問題を抱えている。

１つには、１９８０年代以降、コスト削減を目的にシステム部門の子会社化や外注依存が進行したことである。これにより、システムのブラックボックス化やベンダロックインが生まれ、システム開発の機敏性を失ってしまった。これについては、内製化への回帰が出てきている。

２つ目の問題は、システム開発の工程を知的集約的なものと労働集約的なものとに分け、分業が進行したことである。これにより、労働集約的な工程の生産性の向上は遅れ、また、知的集約的な工程において技術や知識の蓄積に問題が出てきた。これに対しては明確な解決策が見いだせていない現状があるのではないだろうか。そこで、この２つ目の問題点について考察をし、対策することが急務である。

他方、今後、受託システム開発の市場はどうなるだろうか。ＤＸ関連の需要や日本の長期雇用の労働環境やＩＴ人材不足の状況を踏まえると、市場の成長も見込まれている[1]。

近年、システムの巨大化や技術の高度化・専門化が進行していく中で、インフラの選択、ＯＳＳやパッケージソフトウェアの選択、ユーザーとの開発分担の中でどこを手掛けるかが重要になってきている。受託システム開発会社においては、今後さまざまな面でより一層の独自性を発揮しなければ生き残っていくことが難しくなっていくだろう。

これまでのベンダー依存、多重下請け構造は、必ずしも最適な分業体制になっていない。分業体制の見直しにより、生産性を向上させ独自性を発揮してくことができるのではないだろうか。そこで、本稿では、受託システム会社が、生産性向上のために、かつ独自性も得られるための、最適な分業体制についてとりうる類型を考察する。

2. 先行研究

ソフトウェアビジネスに関する研究では、クスマノ（2004）[2]によるものがある。20 年近く前に発表されたものであるが、製品企業、サービス企業（あるいはハイブリッド・ソリューション企業）と体系づけた研究は、現在でも有効な普遍的な研究だろう。この研究では、日本のソフトウェアビジネスの類型をソフトウェアファクトリーと分類し生産性を評価したものの、ビジネスとしての評価は決して大きいものではなかった。しかし、ソフトウェアファクトリーたる所以は、その開発プロセスにあるもので、ソフトウェアビジネスとしては、サービス企業あるいはハイブリッド・ソリューション企業のそれと変わるものではないだろう。

また、多重下請けや分業構造に課題を見出す研究やウォーターフォールとアジャイルの開発プロセスの違いに焦点をあてた研究[3]も多くあるが、現在の日本における産業構造や企業の組織構造を前提とした場合に明確な解決策を提示するものではないだろう。

例えば、平井（2017）はソフトウェア開発を対象に、下流工程における開発作業に知的集約的な面が備わっていることを明らかにした。しかし、下流工程のすべての開発作業を知的集約的なものとすることは、人材やコストの面からも現実的ではないだろう。

そこで本稿では、近年の DX、アジャイル、自動化ツール等の先端技術のトレンドやビジネスモデルも踏まえて、受託システム開発ビジネスにおける戦略とその分業体制について考察する。

---

[1] ２２年の国内市場規模は６兆７３４億円。２２～２７年の年間平均成長率は２．９％で推移すると IDC は予測している。(2023/09/04 調査)

[2] 「ソフトウェア企業の競争戦略（2004）」ダイヤモンド社

[3] 平井によるソフトウェア開発における知識労働と分業構造に関する研究や奏らによるアジャイル開発とウォーターフォール開発の比較などがあげられる。

3. 現状の分業構造に関する分析

これまでの分業体制は、歴史的な経緯の中で形成されたもので、主にコスト削減が目的となっていた。受託システム開発のビジネスにおけるＫＳＦには、人員の稼働率や生産性、技術・スキルなどがあり、これらを意識した分業体制の構築が必要ではないだろうか。

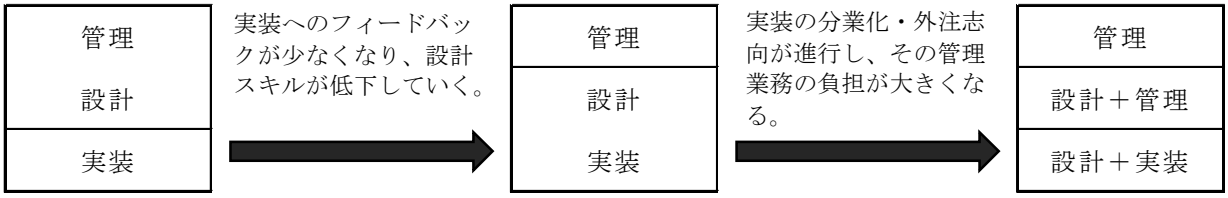
あるべき分業体制を考察する前に、現状の分業体制がどうなっているかを紹介する。国内のシステム開発における分業体制としてよく見られるものは、管理、設計、実装[4]を分担するもので、開発工程との対応は以下のようになる。

<table>
<tr><td>要件定義</td><td>基本設計</td><td>詳細設計</td><td>コーディング</td><td>単体テスト</td><td>結合テスト</td><td>システムテスト</td></tr>
<tr><td colspan="7">管理</td></tr>
<tr><td colspan="2">設計</td><td colspan="3">－</td><td colspan="2">設計</td></tr>
<tr><td colspan="2">－</td><td colspan="3">実装</td><td colspan="2">－</td></tr>
</table>

（出所：筆者作成）

図 1　現状の分業体制

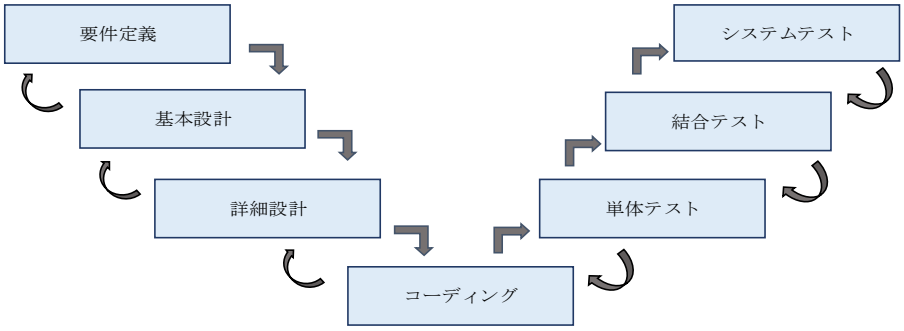
1980 年代以降、管理および設計は知的集約的、実装は労働集約的なものとして分業化が行われた。したがって、管理および設計といった上流の作業は自社で行い、実装など労働集約的な作業を外注先に依頼するといった形態がとられた。しかし、これが時間の経過とともに、以下のように、その実態に変化が生まれてきた。



（出所：筆者作成）

図 2　分業体制の変化

1 つに、設計が実装側にシフトしていった点があげられる。受託システム開発では、従来からウォーターフォールの開発プロセスが採用されることが多い。分業構造との親和性もあるが、計画・管理のしやすさというメリットがあるためだろう。ウォーターフォールの場合、以下のように工程を進めていきつつ、不具合や変更が発生した場合には前工程へのフィードバックが行われる。設計へのフィードバックを、設計担当ではなく実装担当が対応していくこともあり、これにより設計担当への技術やスキルの蓄積が減り、最終的には設計に必要なスキルがなくなってしまうのである。[5]



（出所：筆者作成）

図 3　ウォーターフォール開発プロセスのＶ字　フロー

---

[4] 上流工程に位置付けられる開発作業を「設計」、下流工程に位置付けられる開発作業を「実装」と定義している。

[5] ソフトウェア開発が標準化された単純作業ではない点については、クスマノ（2004）らが指摘してきた点である。

さらに実装担当の外注やその細分化が進むと、その管理作業の負担が膨らむ。外注先が実装担当の管理を行うことになる。

4. あるべき分業体制

それでは、どのような分業体制が望ましいのだろうか。まずは、受託システム開発のビジネスモデルの観点から分析をする。受託システム開発はソフトウェアビジネスの類型としては、「サービス企業」[6]に分類されるだろう。この類型では、顧客との信頼関係を維持して稼働率を高い水準とすることが重要となる。受託開発という特性上、稼働率は顧客の需要に依存してしまう。したがって、この変動を少なくするための取り組みが必要となる。具体的には、以下の２つの観点があるだろう。

・顧客に多様性を持たせて需要の変動リスクを少なくする
・開発担当者の取り扱える製品に多様性を持たせて需要変動時の柔軟性を大きくする

前者は、ある製品のシェアを高めることと同義であり、このことはある製品・技術・知識に関する再利用性を高めることにもつながるため、生産性向上の観点でも最も望ましい形だろう。シェアを高めるためには機能性、コストや品質などで競合優位性を高める必要があり、ある分野での技術・知識の蓄積が不可欠である。規模の経済や経験曲線効果が有効に働くため、競合に対してどれだけ早期に開発実績を獲得できるかが重要であり、そのためには先を見据えた業界動向や技術トレンドの調査、顧客と協働してのシステム企画の活動が必要となる。

それに対し後者は、ある開発担当者がどれだけ柔軟に様々な業界・製品に対応することができるかである。そのためには、開発担当者のモチベーションを高める工夫と、製品ごとにある属人性、顧客や業界の固有性を下げる工夫の両面での対応が必要と考える。開発作業に関する共通的な技術を広く獲得することや、開発作業のプロセスの標準化を推進することが必要となる。開発プロセスや技術に関する属人的な技術の再利用を促進するという観点でも生産性向上に寄与するものだろう。

ここからは、以下のような業務に分類し、分業体制をとるべきことの示唆を得ることができるだろう。

・シェアを獲得できる分野での新規受託、製品開発業務
・既存の特定製品の標準化、改善業務
・多様な製品・業界を対象とした受託システム開発業務

次に生産性を観点として分析を進める。生産性の考慮には、その目的に応じてインプットとアウトプットのレベルをあわせておくことが必要だろう。本稿では、そのレベルを３階層に分けて分析していくこととする。

レベル１	決まった時間でどの量の仕事を終えることができたか。開発チームの生産性評価レベル。
レベル２	決まった時間でどのくらい価値が期待される仕事ができたか。製品企画チームの生産性評価レベル。
レベル３	決まった時間でどのくらいの価値が実現できたか。部門の生産性評価レベル。なお、本稿では分析対象として取り上げないこととする。

（出所：広木(2022)『開発生産性について議論する前に知っておきたいこと』をもとに筆者作成）

図 4　生産性の３階層

これら各レベルを、システム開発の原価構造と対応させると以下のようになるだろう。

[6] クスマノが「ソフトウェア企業の競争戦略」の中で類型化したもので、製品企業、サービス企業、ハイブリッド・ソリューション企業の類型がある。

調達品費（他社製品）	他社製ソフトウェアライセンス等費用	レベル１生産性
リスク費	システム開発にかかる原価に対して一定の料率を乗せたもので、プロジェクトの規模や難易度に応じて設定される	
労務費	自社開発担当者の人件費	
外注費	委託にかかる費用	
経費	交通費、オフィス・機器賃貸費	
販売管費	マーケティングや提案活動にかかる費用	レベル２生産性
自社製品費	自社製ソフトウェアのライセンス等費用	
間接費	部門配賦など	レベル３生産性

（出所：戸川(2018) 日経クロステック『システム開発費の内訳は？IT コストの謎」をもとに筆者作成）

図 5　システム開発費の内訳

レベル１の生産性は、いかに効率的に開発作業を進めるか、いかに人員の稼働率を高めるか、また、いかにリスクの顕在化を回避できるか、ということになるだろう。レベル２の生産性は、いかに再利用性の高い製品を持てるかになるだろう。

レベル１のいかに人員の稼働率を高めるか、レベル２のいかに再利用性の高い製品を持てるかについては、先のビジネスモデルの観点で取り上げた内容であるため、ここではレベル１のいかに効率的に開発作業を進めるか、いかにリスクの顕在化を回避できるかを観点が分析対象として残る。ただし、システム開発のリスクについては、様々な要因があるため本稿での分析対象からは外すこととする。

効率的に開発作業を進める点については、前章で分析したとおり、工程による分業により設計と実装を完全に切り離すことは短期的には有効であるものの、長期的な観点からは技術の流出リスクがあるため避けなければならない。短期と長期のバランスが重要である。適切な開発プロセスや自動化ツールの適用によってＱＣＤをコントロールしていくことが長期的に必要だろう。

このバランス（短期、長期）をとるためには製品や顧客、開発規模に応じて最適化を図るべきである。すなわち、様々な KPI を考慮しながら経営者側が組織設計、キャリア形成を図っていくことが必要だろう。

5. 考察

前章においてビジネスモデルと生産性を観点に分業体制を考察した。

前段のビジネスモデルを観点とした分析から得られた分業体制は、予備校などをはじめとした教育の分野との類似性を見出すことができないだろうか。「シェアを獲得できる分野での新規受託、製品開発業務」は、カリスマ講師の担う業務であり、そこから教材の標準化、標準化された教材を使った規模の拡大といった業務につながっていくだろう。

後段の生産性を観点とした分析からは、受託システム開発会社におけるキャリアパスのマネジメントの重要性が、分業体制や生産性に影響を与えうることの示唆を得ることができるだろう。

6. まとめ

本稿における分析から、受託システム開発のビジネスにおける教育分野のビジネスとの類似性や、キャリアパスのマネジメントの重要性に関する示唆を得ることができた。このことから、分析により導きだされた分業体制を実現するために、具体的にどのような組織設計、ＫＰＩ、人材マネジメントをしなければならないか、今後分析していくことが必要だろう。

参考文献

[1] マイケル・Ａ・クスマノ，2004，ソフトウェア企業の競争戦略，ダイヤモンド社
[2] 及川卓也，2019，ソフトウェア・ファースト，日経ＢＰ
[3] Camille Fournier，2018，エンジニアのためのマネジメントキャリアパス，O’REILLY
[4] 平井直樹，2017，ソフトウェア開発プロセスにおける分業構造と知識労働
[5] 広木大地,2022，開発生産性について議論する前に知っておきたいこと - Qiita
[6] 紛争事例に学ぶ IT ユーザーの心得，2017，細川義洋

[7] 奏泉寺他， 2020， リーンのコンセプトによるアジャイル開発の生産性指標の提案とウォーターフォール開発との比較
[8] 田中辰雄，2010，日本企業のソフトウェア選択と生産性，経済産業研究所
[9] 峰滝他，2007，日本のソフトウェア産業の業界構造と生産性に関する実証分析，経済産業研究所