推薦の言葉

- プロフェッショナルな仕事をしたいですか？
- 楽しく仕事をしたいですか？

本書は、この2つの問いに「Yes」と答えた人のための本です。それには、受託開発現場の考え方を変えること、やり方を変えること、組織を変えること、そのすべてに先立って、あなた自身が変わること。

本書の読者が、お客さまから「あなたがいてくれたから成功できた」、後輩から「先輩と仕事ができてよかった」と言ってもらえる、そんな現場が日本に増えることを願います。

平鍋 健児
㈱チェンジビジョン 代表取締役社長、㈱永和システムマネジメント 副社長

本書は受託開発プロジェクトの見積り、計画、要求、開発、テスト、運用を取り上げ、具体的な事例やアウトプットを交えて、シンプルな形で、非常にわかりやすく解説している。そして、自分を変え、仲間を増やしながら、組織を変えていくという受託開発チームの成長を促すためのノウハウが見事に綴(つづ)られている。

受託開発をこれから行うエンジニアや、受託開発に疲れ果てたエンジニアに勇気を与える良書である。

萩本 順三
㈱豆蔵 取締役、内閣官房 情報通信技術担当室 電子政府推進管理局（GPMO）補佐官

もしあなたが受託開発に携わっているのであれば、本書に書かれているのは知っていて当然のことばかりです。ですが当たり前のことをきちんと理解していることは大切なことです。それらの「当たり前」を過不足なく伝えているという点において、本書には十分な意義があります。自分の仕事を省みるツールとして本書をお薦めいたします。

羽生 章洋
㈱スターロジック 代表取締役兼CEO

血も滴るばかりの良書である。

著者の岡島氏はたたき上げ現役バリバリのリーダーである。ときには失敗し、苦悩し、考え抜き、自らを変革させ、行動してきたことだろう。であるからこそ発することができる、血の通ったメッセージが本書には満ち溢れている。

我々、開発の現場にいる者に必要なものは、このような生々しくも生き生きとしたメッセージなのではないだろうか。

ひろせ まさあき
KLab㈱ Kラボラトリー システムアーキテクト

自分の仕事を憎むにはあまりにも人生は短い——私たちの仕事は、受託開発です。本書は、私たちが自分自身の仕事に誇りと希望を持てるようにするための一冊です。私は本書に書かれていることを確かに岡島さんから学びました。きっとあなたの役にも立つはずです。あなたの仕事に誇りを持てるようにできるのは他でもない、あなただけです。

角谷 信太郎
㈱永和システムマネジメント サービスプロバイディング事業部 チーフプログラマ

はじめに ——仕事を生業にする

「生業（なりわい）」という言葉があります。「生活のための仕事」という意味なのですが、語源は「五穀が実るように務めるわざ」だそうで、なかなか味わい深い言葉です。

一朝一夕に「五穀が実るように務める」ことはできません。地を耕し、種を蒔き、雑草を刈り、収穫のためにじっくり働く覚悟が必要です。より実り多くするために日々努力して技術を高め、時には天候のようにコントロールできない問題とも付き合わなくてはいけません。

私は「生業」という言葉を重く受け止めます。生業とは、簡単にやめることのできない、まさに一生物の仕事のことだと思います。

あなたはソフトウェア開発という仕事を生業だと思えますか？

この質問に自信を持って「はい」と答えられる人は少ないと思います。「プログラミングは大好きだけど……」「業界の構造に不満があって……」「どうせ自分の会社は……」などといった躊躇（ちゅうちょ）を感じているのでしょう。

その一方で、開発者の中には苦労しながらも、仕事を生き生きと楽しんでいる人がいるのも事実です。私の周りにもそんな人たちがたくさんいて、彼らはソフトウェア開発という仕事をまさに生業と感じているかのようです。

私は、仕事を生業に感じられる人たちには共通する特徴があると思います。

まず、技術を持っています。ここでの技術とは、プログラミング言語や開発プロセスに関する知識だけではなく、コミュニケーションやビジネスに関するノウハウも含んだ幅広いものです。

そして主体的です。業界の構造や会社の問題をあれこれ悩んで止まってしまうより、自分の行動や考え方を変えて先に進むことを重視します。

さらに、仕事と生活の両方を大切する価値観を持っています。仕事は人生における目的でも手段でもなく、仕事での経験と生活での経験が相互にプラス作用し、人生を豊かにすることに気がついています。

私は、このような人たちが増えることで、業界全体が変わると信じています。今の受託開発、というよりはソフトウェア業界は行き当たりばったり過ぎます。今年さえ乗り越えられればいいという考えではダメで、年々着実に成長できるような業界にしなくてはいけません。

一つのプロジェクトの成功や満足な仕事ができた喜びだけにとどまらず、そこで得られたことを次の世代の種として蒔くことができる。プロジェクトがうまくいかなかったのであれば、技術を改良し次のチャレンジでは成功させたいと努力できる。ソフトウェア開発の現場にそんな開発者をもっと増やしたい。それが私の目標であり、この本がその一端を担えれば幸いです。

2008年3月　岡島幸男

受託開発の極意:目次

推薦の言葉 4

はじめに 6

第0章
受託開発を楽しむには 15

0.1 問題は現場から解決する 16
0.2 サービス中心主義 17
0.3 全部自分でやってみる 19
0.4 まずあなたから始めよう 20

第1部
受託開発の手ほどき 22

第1章
お客さまに関心を持つ 23

1.1 まずお客さまに関心を持つ 24
1.2 なぜ関心を持てないのか 26
1.3 体制と役割でコミュニケーション量を増やす 26
窓口型チーム 26
ソーシャルなチーム 27
1.4 打ち合わせでは役割を工夫する 28
責任者 29
書記 30
Q&Aメンバー 31
1.5 「気持ちいい関係」がコミュニケーション量を増やす 32
相手の知りたいことを優先する 32
親しき仲にも礼儀あり 33
「問題対私たち」の構図を作り上げる 33
1.6 お客さまの立場でプロジェクトを眺める 34

第2章
サービスは見積りから始まっている 35

2.1 見積りとコミットメント 36
2.2 サービスはすでに始まっている 37
見積り根拠の納得感 37
提案のバリエーション 39
プラスα 40
2.3 見積書作成の流れ 41
①お客さまの「やりたいこと」をヒアリングする 41
②「もの」を数え、必要な工数を見積る 42
③理解できないことはお客さまに追加ヒアリングする 43
④見積書を提出する 44
⑤交渉成立 44
2.4 見積り手法を身につける 44
ユースケースポイント法 45
ファンクションポイント法 46
タスク分解法 47
2.5 最適な見積り手法を選択する 48
2.6 実績データが見積り精度を高める 49

第3章
要件さえつかめば大丈夫 51

3.1 決まらないくせに変わりやすいのが要件 52
3.2 要件定義の難しさ 54
3.3 要件ヒアリングのコツ 55
手段ではなく目的の話をする 55
積極的に聴いたうえで理由を尋ねる 56
追い詰め過ぎない 57
進捗を示す 57
記録を残す 58

3.4 要件をまとめる 59
業務要件 59
システム要件 61
3.5 要件を軽くするには 62
3.6 丸投げされても前進する 63
3.7 丸投げドキュメントの読み方 64
①既存ドキュメントの一覧を作成する 64
②既存ドキュメントの目的を調べる 65
③自分たちが作る「もの」にフォーカスする 65
④「もの」とIN・OUTを意識する 65
⑤説明ドキュメントを書く 66
⑥シナリオを具体的に書きながら検証する 67
3.8 コストに見合う価値を示す 68

第4章
保守性にこだわった**設計・実装・テスト** 71

4.1 設計で品質を作りこむ 72
4.2 高レベル設計ではドキュメントを重視する 73
ドキュメントは必要 73
役立つドキュメントの書き方 73
基本設計書 74
ユースケース記述 75
テーブル定義書 77
画面・帳票設計書 77
4.3 低レベル設計はテスト駆動で行う 78
テストコードで内部処理を書く 79
4.4 テストコードで品質と保守性を向上させる 80
テストの効果を測定する 80
ほとんどの問題は単体テストで発見できる 81
カバレッジは重要 82
テストを増やせば品質は上がる 83
4.5 テストは計画が重要 84
最初に計画を立てる 84
単体テストの確認ポイント 85
結合テストの確認ポイント 86
システムテストの確認ポイント 86
4.6 テストケースの質を高める 86
網羅しやすいこと 88
読みやすいこと 88
保守しやすいこと 89

第5章
運用が最上流 93

5.1 作っているより使っている時間のほうが長い 94
5.2 お客さまとの信頼関係も保守する 95
5.3 保守の決め手はナレッジとマインドの伝承 95
テスト資産を活かす 96
ドキュメント資産を活かす 97
作業の自動化資産を活かす 99
5.4 トラブル時の心構え 100
事実を重視する 100
教訓にする 100

第6章
計画とスケジュールの管理 105

6.1 3つのスケジュールを使い分ける 106
スケジュール案 106
マスタスケジュール 107
進捗管理用スケジュール 107
6.2 自分のスケジュールは自分で管理する 111
6.3 見積りがはずれる原因と対策 111
極端に心配性 112
割り込み時間を考慮していない 112
品質に関する認識がずれている 113
6.4 実績データを活用する 113
実績データを集める 113
時間配分の目安を求める 114

6.5 実現可能なスケジュールを書く 115
6.6 主体性を持って計画する 117

第7章
チームで成功を目指す 119

7.1 チームカラー ＝ リーダーのカラー? 120
7.2 チームカラーは日常に宿る 120
7.3 リーダーとの関係構築力 121
「バリバリ」リーダーとうまくやる 123
「サクサク」リーダーとうまくやる 123
「よしよし」リーダーとうまくやる 124
「ふむふむ」リーダーとうまくやる 124
上下関係を超えていけ 125
7.4 「普段着」の行動力 125
目的と課題の共有 126
プラスαの行動 126
冷静と情熱のバランス 127
チーム優先 128
7.5 交渉力 129
原則1: 勝ち過ぎない 130
原則2: 代替案を持つ 130
原則3: 気づきと築き 131

第2部 人と組織を変えること 136

第8章
自分を変える 137

8.1 変わりたい理由 138
8.2 変われない理由 139
8.3 変わるとはどういうことか? 140
連続性がある 140
フィードバックを求める 141
元に戻ろうとする 141
8.4 正しい変え方を身につける 142
8.5 Know: 今の自分を知る 143
8.6 Imagine: 変わったあとの姿を思い描く 144
8.7 Move: 行動する 145
チャンスを逃さない 145
言葉使いに気をつける 146
自分を客観的に評価する 146
成果をアピールする 147
自分にプレッシャーをかける 147
周りの人を巻き込む 147
8.8 Love: 変われた自分を好きになる 149

第9章
仲間を増やす 151

9.1 他人に関心を持つことから始める 152
自分のことを話さない 152
メタマニアの勧め 152
9.2 仲間をもっと好きになる 153
勉強会 153
改善グループ 154
社内ブログを立ち上げる 155
9.3 活動を継続させるリーダーシップ 156
目的・課題・次のアクション 156
空気の読み過ぎに注意 157
成果を上げること 157

第10章
組織を変える 161

10.1 誰を変えるのか? 162
10.2 組織を変える難しさ 163

課題1: 変化後のイメージの共有 163
課題2: 組織ミッションとの整合性 164
課題3: コンセンサスのコスト 164
10.3 組織の変え方 164
10.4 Keep: 組織ミッションとの整合性を保つ 165
10.5 Interest: 関心を持ってもらう 166
10.6 Manage: 変化を管理する 167
10.7 Lead: 身をもって示す 167
10.8 挫折したら「自分」に立ち戻る 168
10.9 まずあなたから始めよう 169

付録 171

付録A プロジェクトファシリテーション入門 172
朝会 172
かんばん 174
ふりかえり 176
付録B 解説付き参考文献 178

あとがき 182

索引 185

コラム

半分は受託開発 21
技術はミームだ 69
「設定よりも規約」で保守性を向上させる 91
開発プロセスと中庸さ 102
受託開発組織の課題 132
哲学もほどほどに 159
ビジョン共有ワークショップ 166

第0章 受託開発を楽しむには

問題は現場から解決する
サービス中心主義
全部自分でやってみる
まずあなたから始めよう

0.1
問題は現場から解決する

私はここ最近、どうしたら受託開発という仕事をもっと楽しめるかを考えています。現場の開発者は「やらされている感」「顧客に振り回されている感」「自分の仕事が何の役に立っているのかわからない感」といった目に見えにくい問題に悩んでいます。

私はマネージャであり、開発者に仕事のやりがいや楽しさを感じてもらうことは重要なミッションです。しかし、それ以前にソフトウェア業界全体に漂うマイナスのイメージや雰囲気の悪さ、一種の閉塞感が気になります。最近では新3K（きつい・帰れない・給料安い）などと言われ、業界全体に活気がありません。

これらの問題点を解決するのに、「日本のソフトウェア業界の構造問題であるから、もっと受託開発の比率を下げるべきだ」などと言われてもどうしようもありません。評論家はお呼びでないのです。

問題解決のためには、もっと現場からアプローチしなくてはいけません。プロジェクトにおける主役は現場の開発者一人一人なのです。

マネージャを過信してはいけません。とても良い働きをして、心から現場のことを考えているマネージャであっても、できることとできないことがあります。本当に優秀なマネージャはこのことをわきまえています。あなたがやるべきことはあなたにやらせるでしょう。そしてなにより、優秀なマネージャがいつもあなたのプロジェクトにアサインされるとは限らないのです。

経営者に期待しすぎてはいけません。明確なビジョンと戦略を持つ経営者は会社を良い方向に導く可能性が高いでしょう。あなたのところの経営者はいつか理想の職場を作り上げるかもしれません。でも、今あなたのプロジェクトのためだけに手を差し伸べてくれる理由はないのです。

最も大きな問題は、体制とか、組織とか、社会とかそんなものを理由にして開発者自身があきらめてしまうことです。受託開発だから、下請けだから、お客さまや上の言うことだから、などといった受け身の姿勢が問題を大きくします。

0.2
サービス中心主義

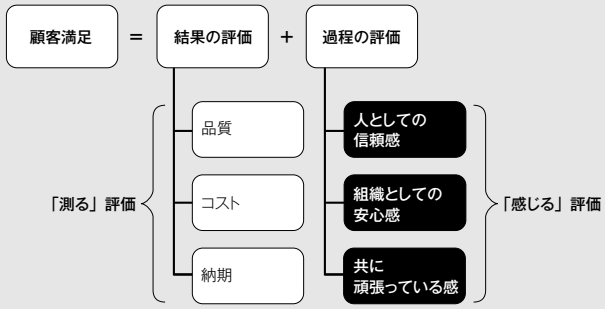
まず、受け身の姿勢を変えなくてはいけません。人に振り回されて悩むくらいであれば、こちらから攻め返してやりましょう。きめ細やかなサービスで顧客満足をコントロールするのです。

顧客満足を十分に得られれば、それは「振り回し」に対する抑止力になり得ます。また、信頼を得てリピーターになってもらえれば、慣れによってお客さまの「癖」がつかめます。これにより振り回される前に手を打つことも可能になるでしょう。

そのためには、お客さまの問題を解決してあげることが受託開発の本質であり、その質こそが競争力であると認識しなくてはいけません。自分たちが作ったソフトウェアやシステムだけが、あなたやあなたの組織に対する満足度を決めるわけではないのです。

図0-1は私が考える「顧客満足の公式」です。

図0-1　顧客満足の公式



顧客満足は品質・コスト・納期といった「結果として評価できるもの」(「測る」評価)だけでなく、人としての信頼感・組織としての安心感・共に頑張っている感などの「過程を通して評価を感じられるもの」(「感じる」評価)からも得ることができるのです。

品質・コスト・納期にはバランスが必要であり、トレードオフが発生します。つまり最高の品質の物を、最短の納期で仕上げ、さらにコストを度外視して実現することは不可能なのです。

しかし、「感じる」評価にはトレードオフは発生しません。それぞれ青天井に伸ばすことができ、その分だけ顧客満足を高めることができます。

お客さまは結果だけでなく、途中経過にも関心があります。問題解決に臨むあなたという人間の立ち居振る舞いもサービスの一部であり、重要な競争力なのです。

技術力や結果だけでなく、過程も重視しましょう。これを「サービス中心主義」と呼びます。**サービス中心主義とは、結果としての評価はもちろんのこと、過程としての評価をさらに重視し、顧客満足度を高める考え方です。**

「サービス中心」はこの本の重要なテーマであり、本書に一貫して織り込んである哲学です。

0.3 全部自分でやってみる

そしてなにより、仕事を全部自分でやりたいと思う。やってみる。この姿勢が重要です。「やらされている感」をなくすには自分たちでやるしかありません。自分の任されている工程ではないから関心を持たない。まだ若いから今は知らなくていい。このような気持ちを切り替えてみましょう。

やるからには、うまくいくやり方を学ぶ必要があります。**この本は受託開発という仕事を全部自分でできるようになるための手ほどきです。**第1部では、見積りから保守・運用といった開発者の仕事を説明するのはもちろん、お客さまとの接し方やチーム内でのコミュニケーションについても手ほどきします。

もちろんやり方を学んでも、実際に自分や自分のチームで全部やれるかどうかはわかりません。下請けで一部の工程を請け負っているのであれば、現実問題としてそれをはみ出した仕事をするのは難しいでしょう。

しかし、自分たちの請け負っている仕事・成果物がどうつながってお客さまに価値をもたらすのか、そしてほかのチーム・会社との関係はどうなっているのか。この部分に着目できないと、いつまでたっても自分の仕事が何の役に立っているのか思

い悩むでしょう。

0.4
まずあなたから始めよう

今はできなくても、いつか自分の手で全部やってみればいいのです。今のままの組織や会社でできないのなら、あなたがそれを変えていけばいいのです。

自分を変える。組織を変える。この本のもう一つのテーマです。第2部では人と組織に目を向けます。自分と向き合い、自分をより良く変えていくこと。そして仲間を増やし、良い文化を培っていくこと。これが組織の改善であり、楽しくやりがいある仕事を実現する一番の近道です。

「Social Change Starts With You」は、XP（エクストリームプログラミング）の提唱者、ケント・ベックの言葉です*。訳すなら「人との関わりはあなたから変えていくのだよ」といったところでしょうか。彼がこの言葉に込めた意味は、「私ではなく、あなたが変えるのだよ！」ということです。変化とは、自分から始め、そしてまわりの人へと広げていくものなのです。

他人から影響を受けるのはかまいませんが、依存してはいけません。人は他人を変えることができません。現状に不満を持ち、何かを変えたいと願うのであれば、まずあなたから始めるしかないのです。

＊「『"social change starts with you"... you? ... me? not you?』みたいなことをこれまたモゴモゴ言ってたら、腰リールを下げたKentBが力強く『Social Change starts with **YOU!**』と言ってくれた。それで我に返った。」（「角谷HTML化計画」http://kakutani.com/20060910.html#p02）

Column
半分は受託開発

受託開発（受注ソフトウェア開発）は、総務省による「日本標準産業分類」では、小分類だと「ソフトウェア業」、中分類だと「情報サービス業」に分類されます。

「ソフトウェア業」は受託開発とパッケージソフトウェア開発のことです。「情報サービス業」はそれに加えて、データ入力や計算センターなどの情報処理サービス、システムの運用を請け負うシステム等管理運営委託、独自のデータを収集・加工し提供するデータベースサービスなども含みます。

経済産業省の調査によると、受託開発は売上ベースで「ソフトウェア業」の86.4%を占めます*。「情報サービス業」全体で見ても46.3%です**。

このような受託開発偏重に対する問題点の指摘は多々あります。しかし数字に基づく事実としては、現時点での日本経済における受託開発の貢献度は非常に高いのです。

本書では「受託の割合が高すぎる」「ソフトウェアの対米輸入超過」など、産業構造に関する問題を直接は扱いませんが、日本の受託開発をとりまく事実認識として知っておいてください。

＊「経済産業省経済産業政策局調査統計部　平成18年特定サービス産業実態調査報告書　ソフトウェア業，情報処理・提供サービス業編」
http://www.meti.go.jp/statistics/tyo/tokusabizi/result-2/h18/pdf/h18-t-01.pdf

＊＊「経済産業省経済産業政策局調査統計部　平成17年特定サービス産業実態調査報告書　情報サービス業編」
http://www.meti.go.jp/statistics/tyo/tokusabizi/result-2/h17/pdf/h17-t-02.pdf

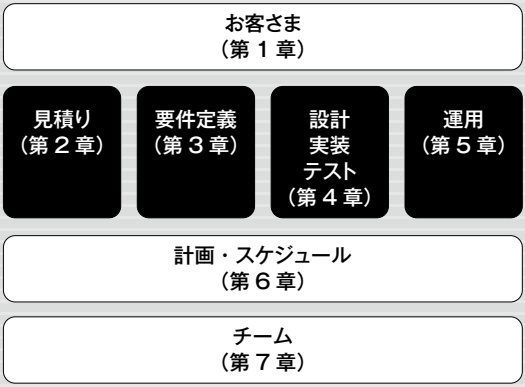
第1部

受託開発の手ほどき

受託開発は大まかに言って「見積り」から始まり、「要件定義」「設計・実装・テスト」「運用」の順に仕事を進めます。

そしてこれらの仕事を円滑に進めるためには、「計画」や「スケジュール」を立てるスキルや、あなたと共に働く「チーム」という土台も必要です。

第一部では、まず受託開発には欠かせない「お客さま」に関する話題からスタートします。次に、見積りから運用、スケジュールの立て方それぞれについて仕事のやり方と心構えを説明し、最後にチームを題材にコミュニケーションに関する考察を行います。



第2部

人と組織を変えること

第2部のテーマは「変化」です。

自分を中心に置くと、「組織」は一番外側の殻のようなものです。組織に問題があってそれを良い方向に変えたい場合、外から変わることを期待するのではなく、まず「自分」からスタートし、中から変えていく必要があります。流れを自分から巻き起こし、徐々に「仲間」を巻き込むのです。

まず、第8章で自分の変え方を説明します。続いて第9章では仲間を増やすためのコミュニケーションについて説明します。そして最終章では組織を良い方向に変えていくためのやり方を提言します。