Visual 開発ツール Node-RED の導入によるプロセスの変化と考慮点

阪井 誠
株式会社 SRA
sakai@sra.co.jp

要旨

本論文では Node-RED を導入したソフトウェア開発プロジェクトの経験者にアンケートを行い，ソフトウェア開発にどのような変化があったのかを調査した．経験者 8 人に対するアンケートの結果，品質と開発期間の評価が高かったものの，ツールの導入だけでは必ずしも効果が得られず，(1)ツールの知識やノウハウを共有すること，(2)特性を活かした設計で品質を作りこむこと，(3)実装を繰り返して常に確認すること，(4)上流から利用すること，が効果的なソフトウェア開発につながることが分かった．

プロセス改善を目的として多くの企業でツールが導入されているが，必ずしも成功していない，本研究の知見を参考に，より効果的なプロセス改善が行われることが期待される．

1. はじめに

「プロセスやツールよりも個人と対話を」とアジャイルソフトウェア開発宣言には書かれている[1]．これはソフトウェア開発におけるプロセスやツールの重要性を認めたうえで，個人との対話の重要性を宣言したものである．特に開発言語のように成果物と直接関わるソフトウェア開発ツールは，プロジェクトの成否に大きく影響する．このため，ツールを先に決定しておいて，経験のある開発者を採用することも多く行われている．このようにソフトウェア開発に大きな影響を与えるので，新しいツールを導入する機会は少なく，開発プロセスに大きな影響を与えるにも関わらず，その知見はあまり多くない．

その一方で，計算機の性能向上や技術の発展によって，効率的なソストウェア開発が可能になった．上流からのアプローチとしてはデータモデリングや画面・帳票まで自動生成する GeneXus[2]，モデルをベースにプログラムのパターンを定義する事で仕様書を実行する X-TEA Driver [3]などがある．逆に下流からのアプローチとしては，本論文で扱う Node-RED をはじめとして，高機能なシステムを簡単に作成するビジュアル開発ツールが数多く提案されている．

Node-RED は Visual IoT ツールと呼ばれており[4]，サーバーサイド Javascript 環境である Node.js およびそのミドルウェアである Express.js 上で動作する．Node.js はイベント駆動で動作し，いわゆるクライアント 1 万台問題が生じないと言われるほど高性能であるが，非同期処理プログラミングは容易ではない．Node-RED はプログラムコンポーネントであるノードを線でつなぐ事で順次処理と非同期処理を簡単に記述できる．また，多様で多機能なノードが豊富にある，一瞬でデプロイが可能，といった特徴から快適で効率的な開発を実現している．すでに日本でもユーザ会のほか[5]，書籍や雑誌の記事もあり[6]，IoT ツールとしてだけでなく，Web サービスなど様々な開発が行われている

これらのツールを導入するとソフトウェア開発プロセスは大きく変わると思われる．しかし，ツールの技術情報は多いものの，プロセスの変化に関する報告は GeneXus に関して開発プロセスを変えるものとして報告があるなど[2]，ごく一部に限られている．また，上流からのアプローチであれば開発規模も比較的大きく，コンサルタントを依頼できる可能性があるが，下流からのアプローチのツールを導入する場合は，当初は規模が小さいことが多いので外部に委託することが難しい．このことが，プロセスを大きく変えるツールの導入を難しくしている．

本論文では Node-RED によるソフトウェア開発の経験者にアンケートを行い，ソフトウェア開発のプロセスにどのような変化があったのかを調査し，その原因を考察した．アンケートを実施した経験者の開発対象は，社内サービス，プロトタイプ，テストダブル(ドライバ，モック，スタブ)，自社パッケージ，ユーティリティなどである．また，アンケートで調査した内容は，品質，コスト， 開発期間に対する選択式の評価および自由記述，要件定義，設計，プログラミング，テスト，リリースに対して従来との違いについて，自由記述で行った．

以降の章では，Node-RED の概要と長所・短所，アンケートの方法と結果，考察，そして最後にまとめの順に報告する．

2. Node-RED

2.1. Node-RED の基本

Node-RED は Visual IoT ツールと呼ばれ，Web ブラウザ上のエディタでプログラミングする．左の領域にある長円のプログラムモジュールをノードと呼び，中央の編集領域に配置し，ノード間を接続してフロー（処理）を作成することでプログラミングする．ノードに名前を付加できるが，単に配置するだけでも設定に応じた内容が表示される．

最もシンプルなフローは図1のようになっている．左端がインジェクトノードでデフォルトの timestamp がセットされている，このノードの左にあるボタン部分をクリックすると，現在時刻が msg オブジェクトとして次のノードに送られる（クリックでなく，定期実行することもできる）．右端にあるのがデバッグノードで，受け取った msg オブジェクトの一部またはすべてを右側のデバッグタブに表示する（コンソール出力も可能）．インジェクトノードに文字列等をセットすることも可能だが，ここでは中央のファンクションノードで“Hello world!”を msg.payload に代入している．

ここで，インジェクトノードの代わりに http ノード，デバッグノードと並列に http response ノードを接続すると，レスポンスボディに“Hello world!”を返す web API になる（図2）．デバッグノードの出力であるmsgオブジェクトは JSON 形式なので，コピーして別のインジェクトノードのデータとしてペーストすれば，後続の処理を手分けして開発できる．インタフェースがすべて msg オブジェクトに統一されているので，処理の変更や追加が容易なのである．

編集領域は複数のタブで構成されていて，複数のフロー間をリンクでつなぐこともできるので，大きなシステムは，タブを切り替えて開発することが可能である．フローを修正した場合は右上のデプロイボタンが赤くなり，これを押すことで瞬時に実行が可能になる．

標準で用意されているノードには，データをセットする

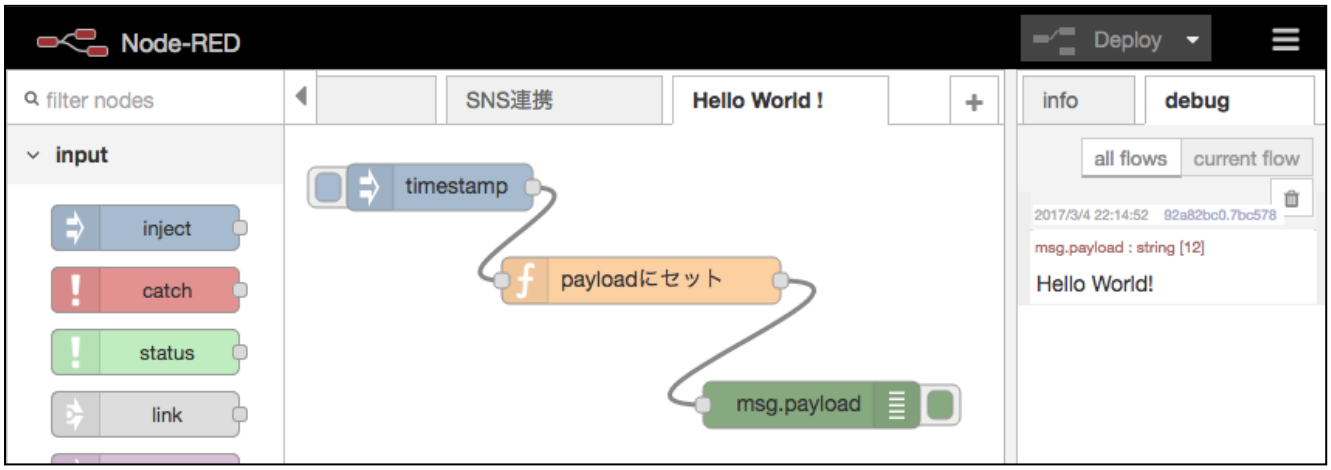

図 1 基本構造

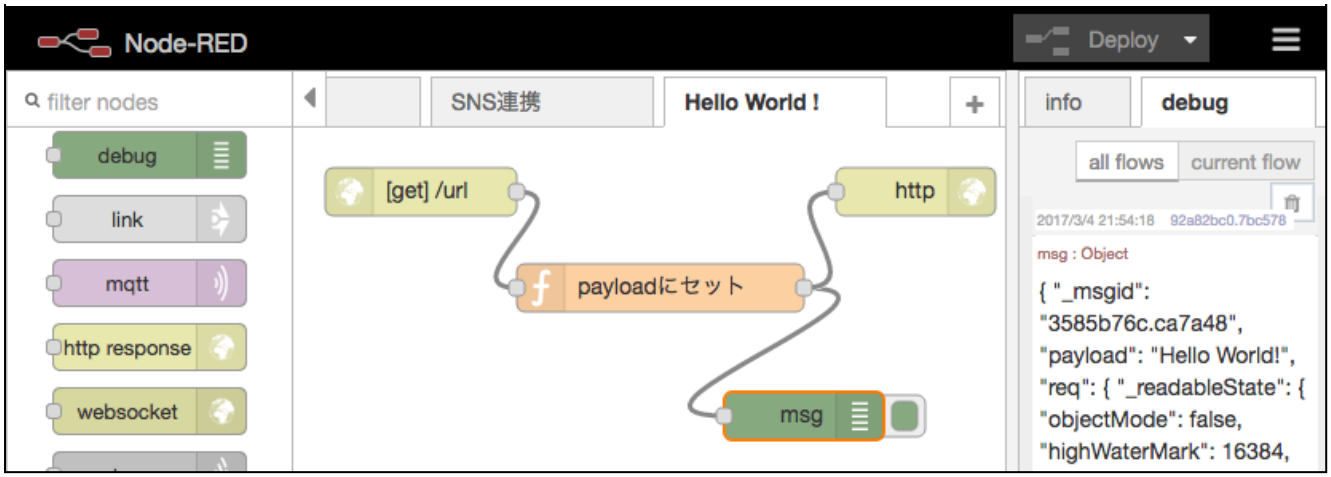

図 2 Web サービスの例

チェンジノードや，分岐処理をするスイッチノードなどがあり，簡単なプログラムであればプログラムコードを書かずに設定だけ実装できる．また，標準ノードのファンクションノードでは javascript でプログラムを書くことができるほか，フローの一部のノード群をサブフローとして一つのノードにまとめることができる．サブフロー以外にも Javascript と HTML でカスタムノードを作成できる．サブフローには入口，出口，名称以外は設定できないが，カスタムノードは標準ノードのように設定用の UI を持ち，多くのカスタムノードがライブラリとして npm(Node Package Manager)で公開されている．

2.2. 長所

Node-RED の長所をまとめると以下のようになる．

非同期処理が簡単に扱える：図1の様にノードを一列に並べると順次処理になる．図 2の http と表示されている http レスポンスノードと，msg と表示されているデバッグノードの様に，ノードの出力を複数のノードに接続すると非同期に処理が行われる．また，配列やオブジェクトを分割して非同期処理する split ノードがあるほか，ファンクションノードから node.send()メソッドで非同期に msg オブジェクトを送出することも可能である．

アルゴリズムが可視化される：ノードの名称とその順序で処理が可視化される．ノードに和名を付けたり，コメントノードでドキュメントを記述することも可能である．

多機能なノード（モジュール）：Node-RED には HTTP，TCP，UDP，WEBソケット，MQTT，Twitterなど各種通信が簡単に行えるノードが標準であるほか，さらに多くの通信やデバイス制御，DB アクセス，SNS，BOT などが数多くノードがコントリビュートされている．中でも Dashboard というライブラリは，Web 画面を簡単に作成でき，デプロイなしにデータの入出力ができるので，テスト用のドライバ，スタブ，モックなどいわゆるテストダブルの開発が簡単にできる．

デプロイが一瞬：コンパイルやビルドが不要で，デプロイが 1 瞬で実行できる．常に動作を確認しながら開発できるので，仕様通りに動作する部分を徐々に増やしながらインクリメンタルに開発することで，品質の高いソフトウェアを開発可能である．

再利用が容易：カスタムノードやサブフローによるモジュール化のほか，Node-RED 内でフローの一部をコピー＆ペーストできる．また，フローの一部や全体をエクスポートして，他の Node-RED 環境にクリップボードを介してインポートして簡単に利用できる．さらに，作業分担の界面にデバッグノードをつないで msg オブジェクトをダンプし，その値を設定したインジェクトノードで開始すれば繋がっていないフローを別々に開発できる．これらの方法を組み合わせると，複数人で効率的に開発できる．

2.3. 短所

Node-RED の長所をまとめると以下のようになる．

単体テスト：カスタムノードにユニットテストを組み込むことは可能だが，フローの一部をテストする標準的な方法がない．このため，フロー全体をブラックボックステストする必要がある．

発展途上：Node-RED の機能は日々進化しているが，（改善しつつあるものの）複数人で同時編集できないなど，まだ不十分なところがあるほか，一部のノードに不具合が残っている．

方式設計が重要：小規模なシステムは実装を優先しても問題ないが，多機能なシステムを小さな工数で作れてしまうので気づかないうちに複雑になり混乱することがある．規模が大きくなる場合は，従来のソフトウェア開発と同じように，msg オブジェクトや永続化対象のデータ構造や，ソフトウェアアーキテクチャをあらかじめ設計する必要がある．

ループが特殊：複数のデータを処理する場合，ファンクションノード内のループ，ファンクションノード＋node.send()，split ノード＋join ノードなどの方法があるが，複数のノードで順次処理を繰り返す場合は，フローがループ状になるので，慣れるまではわかりにくい．

マージ・保守に工夫が必要：従来の言語のように diff で差分をとったものを patch でマージすることができない．このためバージョン管理が容易でない．開発環境のコードを本番環境にマージするには手作業が必要になる．

3. アンケート

3.1. アンケート方法

Node-REDを用いた開発を行っている8人の経験者にアンケートを行った．

各経験者は Node-RED を用いて社内サービス，プロトタイプ，テストダブル（ドライバ，モック，スタブ），自社パッ

ケージ，ユーティリティなどを開発している．全てのプロジェクトはいわゆるウォーターフォール型開発の工程を持っていたが，ばらつきはあるものの厳格な工程完了の審査は行われていない．Node-RED は生産性が高いことから選択された．

アンケートはメールで依頼し，いわゆるQCDとプロセスの変化をアンケートした．品質，コスト， 開発期間に対しては 4 段階の選択式（重複選択可）の評価と自由記述でアンケートした．従来のプロセスとの違いは，要件定義，設計，プログラミング，テスト，リリースに対して自由記述でアンケートした．

3.2. アンケート結果

品質，コスト，開発期間のアンケートは選択肢毎の比率を集計し（図 3），コメントを確認した．複数選択を許しているので，合計は 100%でない．評価は総じて良かったが，複数選択なので評価の良かった点だけでなく，Node-RED の問題点も明らかにすることができた．

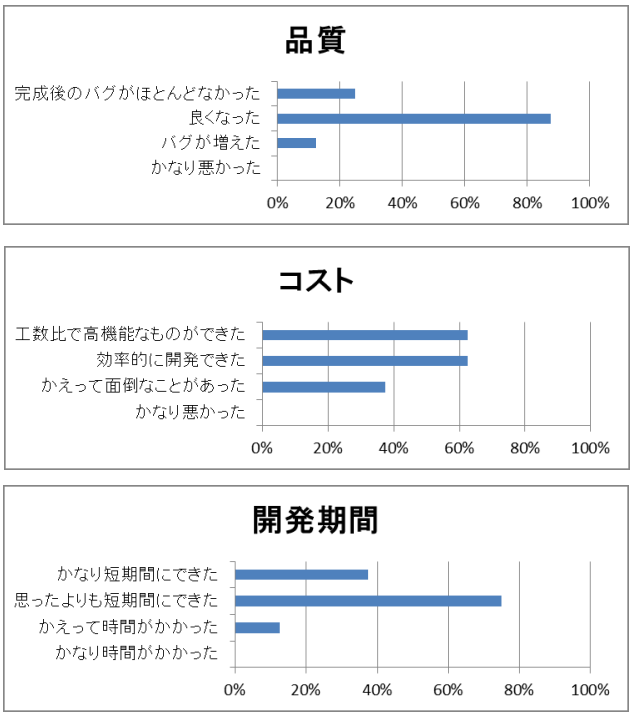

図 3 品質，コスト，開発期間のアンケート結果

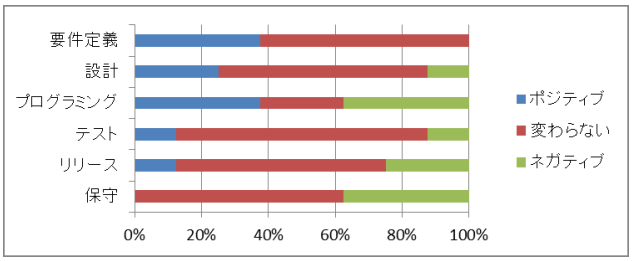

図 4 プロセスの変化のアンケート結果

品質に対して評価が良かった理由は以下のような内容だった．

- サクサクと実装，実行，確認・修整の作ってのループが良かった
- 内製にこだわるよりも品質が良い
- コード量が減った
- 試作に有効
- 非同期処理が容易（promise は間違いやすい）
- フローを意識してシンプルな作りになった

このようにデプロイが一瞬，高機能，ビジュアルなど NodeRED の長所が品質に対して評価が高かった．逆に評価が低かった理由は以下のような内容だった．

- 単体テストができない
- コード検索ができないのでバグが見つけにくい

これらはマイクロサービス化などの工夫はできるものの，Node-RED の短所が品質に影響を与えていたと考えられる．バグがほとんどなかったとしている回答者も 2 名いるので開発プロセスの違いが大きいと考えられる

コストに対して評価が高かった理由は以下のような内容だった．

- 非同期処理が容易
- 高機能なコンポーネントが多い
- 設計からテストまでシームレスにでき効率が良い
- 処理部に注力できた

このように長所が評価された．逆にコストに対して評価が低かった理由は以下のような内容だった．

- diff が取れない
- ドキュメントが少ない
- 必要なノードを探すのに時間がかかった
- 繰り返し処理に苦労した

Node-RED の短所が指摘されたが，下の 2 つは知識やノウハウを共有することで改善できると考えられる．

開発期間に対して評価が高かった理由は以下のような内容だった．

- 非同期処理が容易

- カスタムノード作成で効率化できた
- 開発のスピード感が半端ない
- 他の人のフローを簡単にインポートできる
- 実装にいきなり入れる
- 効率よく開発できる
- 大きな手戻りがなかった

ここでは品質で挙げられた点のほか，インポート・エクスポート，常に動作させることで高品質ななること，が挙げられた．逆に開発期間に対して評価が低かった理由は以下のような内容だった．

- 自作部分の作りで効率や保守性が変わる
- ドキュメントが少ない
- 品質の悪いノードがあった
- 必要なノードを探すのに時間がかかった

ここではサブフローやカスタムノードなど再利用の仕組みをどう使うか，あるいは全体の作りなど，Node-RED に適した設計が重要であること，発展途上のツールであることなどが指摘された

プロセスの変化は，自由記述で書かれた内容からネガティブ評価，ポジティブ評価，変化なし（およびどちらともいえない）の 3 段階を判断して集計した（図 4）．上流においてプロトタイピングが可能な点に対して評価が高かった反面，プログラミング，テスト，リリース，保守に関しては管理面の課題が多く挙げられていた．プロセスの変化の詳細は考察で検討する．

4. 考察

4.1. 品質，コスト，開発期間

アンケートの結果，特に品質と開発期間に対しての評価が良かった反面，コスト面ではある程度の経験が必要であることが指摘された．

Node-RED は高機能なノードを組み合わせて，とりあえず動かすことが簡単．デプロイも一瞬なので作りながら設計と実装を進めると，品質が確認されたフロー（処理)を増やしていくことができる．その反面，単体テストができない，コード管理が困難であるなどツールの特性を理解する必要があり，それがアンケートに反映されたと考えられる．

プロジェクト毎の特性で見ると，知識が不足している回答者を除くと，要件が曖昧で実装しながら詳細な仕様を確認していたプロジェクトの評価が高く，そのような実装を進めないと詳細仕様を決め難い開発に向いていると考えられる．ただし，データの定義やアーキテクチャはしっかりと設計する必要がある．

4.2. プロセスの変化

プロセスの変化に関しては，上流はポジティブな評価だったが，プログラミング・テスト・リリースに関しいては評価が拮抗するかやや低評価，保守に関してはネガティブな評価のみがあった．

上流に関してのポジティブな意見は以下のようなものであった．

- プロトタイプが早くできると説明も早く，意見を貰いやすい
- 曖昧な要求でもとりあえず作り始めることができる
- 作ったものから要件を確定していくことが可能
- 大まかな処理の流れをすぐにフローとして実装可能
- 設計とプログラミングのイテレーションが容易

このようなコメントをした回答者は，設計以降の問題を指摘しながらも，全体としては好意的な表現であった．具体的には，以下のような内容だった

- Inject ノードにより，実装したフローをすぐに動作させやすく，デバッグノードでの確認も容易
- flwos.json（Node-RED の保存ファイル）などほんの少数のファイルをリリースするだけでよく，管理しやすい
- ユニットテストはカスタムフローや API 単位しかできないが，不安は少なかった
- コード管理系だけどうすれば・・・Node-RED だけで開発環境になればすばらしいものになりそう
- （保守は）基本的に容易だが，開発環境を残しておかないと詳細な内容を確認し辛かった
- ユニットテストはカスタムフローや API 単位しかできないが，不安は少なかった

それ以外の設計以降でネガティブな表現をした回答者は，上流のプロセスに変化がないとしていた．また，QCD の項目で良い評価とされていた内容を書きながらも，Node-RED 流の開発スタイルをつかみ切れていない様子であった．

- 単体テストをどのように行うのかわかりませんでした
- 複数人数での開発が少し手間取る
- 外部リソースのノードの設定が外だしに出来なくて，

リリース後にとても煩わしかった
- Node-RED の癖にあわせた設計は必要．設計次第
- 簡単な反面，リリース後の不具合も増える可能性がある
- コード全体の検索が出来ないため，複雑なシステムは保守しにくくなる

これらから，Node-RED を利用する場合は，上流から実装を開始する．特性を理解して設計する．動作をきちんと確認しながら作り上げる．といったことが必要だと考えられる．

4.3. Node-RED の特性と結果

以上のことをまとめると，実装を進めないと詳細仕様が決め難いプロジェクトに有効であり，その実施には以下のような点が重要であると考えられる．

- ツールの知識やノウハウを共有すること
- 特性を活かした設計で品質を作りこむこと
- 実装を繰り返して常に確認すること
- 上流から利用すること

プロセスの視点でいうならば，Node-RED を導入する場合は，他のツールと同じように情報共有や教育が重要であるだけでなく，既存のプロセスをそのまま適用するのではなく，積極的に変更することが，プロセス改善につながると考えられる．

5. おわりに

アジャイルソフトウェア開発宣言には「計画に従うことよりも変化への対応を」と書かれている[1]．新しいツールや技術を導入するには，従来通りのプロセスのままでは難しく，適切に変化させなくてはならない．

本論文では Node-RED によるソフトウェア開発の経験者 8 人にアンケートを行い，ソフトウェア開発プロセスにどのような変化があったのかを調査し，その原因を考察した．アンケートの結果，新しいツールを導入する際には単にツールを利用するだけでなく，ツールの知識やノウハウを共有して特性を活かした設計を行い，実装と確認によって品質を上流から作りこむなど，主体的にプロセスを変更することが，プロセスの改善につながっていたことが分かった．また，Node-RED は発展途上であるので，バージョンアップのフォローも必要になるであろう．

書籍「ソフトウェア開発 55 の真実と 10 のウソ」には，“ソフトウェアの世界には，「一つのツールや技法が何にでも当てはまる」と信じる人が実に多い．”と書かれている[7]．ソフトウェア開発ツールにはそれぞれの特性があり，よく理解したうえでふさわしいプロセスで利用しないとその効果を十分に得ることはできない．本論文の結果は，要件が曖昧で高い生産性が求められるプロジェクトに適した Node-RED という 1 ツールの導入に関しての調査結果から得られたものであるが，このような視点は他のツールにも重要である．

近年，アジャイル開発の再定義の一つとして注目されているモダンアジャイル[8]は次の 4 つの基本理念によって定義されている．

- 人々を尊重する
- 安全な状態を前提とする
- 素早い実験と学習
- 価値を継続的に届ける

今回調査したプロジェクトは一定期間のタイムボックス管理を行う一般的なアジャイル開発ではない．しかし，アンケートの結果から得られた知見は，モダンアジャイルの基本理念と通じるものであり，Node-RED がソフトウェア開発のアジリティ（機敏さ）を高めるものであると考えられる．

今後も Node-RED を利用するプロジェクトを増やして，そのノウハウを蓄積するとともに，より良いプロセスに関する知見も増やしていきたい．

謝辞

アンケートのご協力いただいた西根さん，陣内さん，奥嶋さん，澤本さん，畑中さん，中島さん，佐々木さんに感謝いたします．

参考文献

[1] Beck 他，アジャイルソフトウェア開発宣言，http://agilemanifesto.org/iso/ja/manifesto.html，2001.
[2] 松山，鯵坂,飯ヶ谷，情報システム開発におけるソフトウェア資産の上流シフトへの対応，ソフトウェア・シンポジウム 2016 in 米子, 2016.
[3] 渡辺，X-TEA Driver，ディービーコンセプト，http://dbc.in.coocan.jp/xeadDriver.html，2004..
[4] JS Foundation，Node-RED is a visual wiring tool for the Internet of Things，https://nodered.org/.
[5] Node-RED User Group Japan https://nodered.jp/,

[6] 桑野, ブラウザでお絵描き I/O!Node-RED で極楽コンピュータ・プログラミング (インターフェース SPECIAL), CQ 出版, 2016.
[7] Glass, ソフトウェア開発 55 の真実と 10 のウソ, 日経 BP 社, 2004.
[8] Smith, Agile 2016 Keynote: Modern Agile, https://www.infoq.com/news/2016/08/agile2016-modern-agile, 笠原, Agile 2016 の基調講演: モダンアジャイル , https://www.infoq.com/jp/news/2016/08/agile2016-modern-agile, 2016.