アジャイル開発の試行と検討

2015年11月18日

JASA中部 アジャイル研究会

萩原電気 水谷紘也

一般社団法人 組込みシステム技術協会 Japan Embedded Systems Technology Association

アジェンダ

- はじめに
- 研究会の活動目的
- プロトタイプ開発での事例紹介
- 組み込みソフト開発での事例紹介
- 現場での適用事例に対する検討
- まとめ

はじめに

■ 研究会発足背景

- 組み込みソフト開発を行ううえで抱えている問題を解決する

■ 研究会所属メンバーの特徴

- ソフト開発を請負で実施している会社のメンバーが多い
  (プロダクトオーナーが顧客)

■ 昨年までの主な活動実績

- 契約方法に対する検討
- IPA発行のアジャイルガイドラインに対する検討

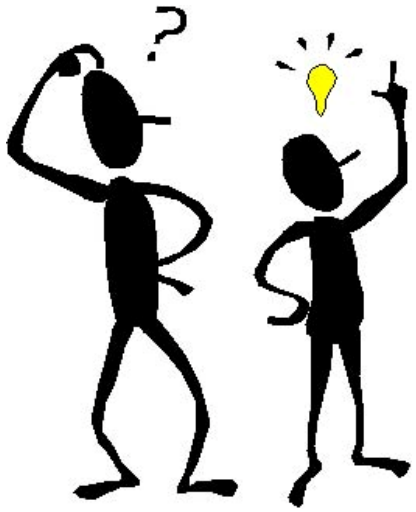
研究会の活動目的

アジャイルってどんなの？？？

とりあえず、やってみよう！

どうやって業務へ取り入れよう？

そもそも解決したい問題とは？



【本年度の活動対象】

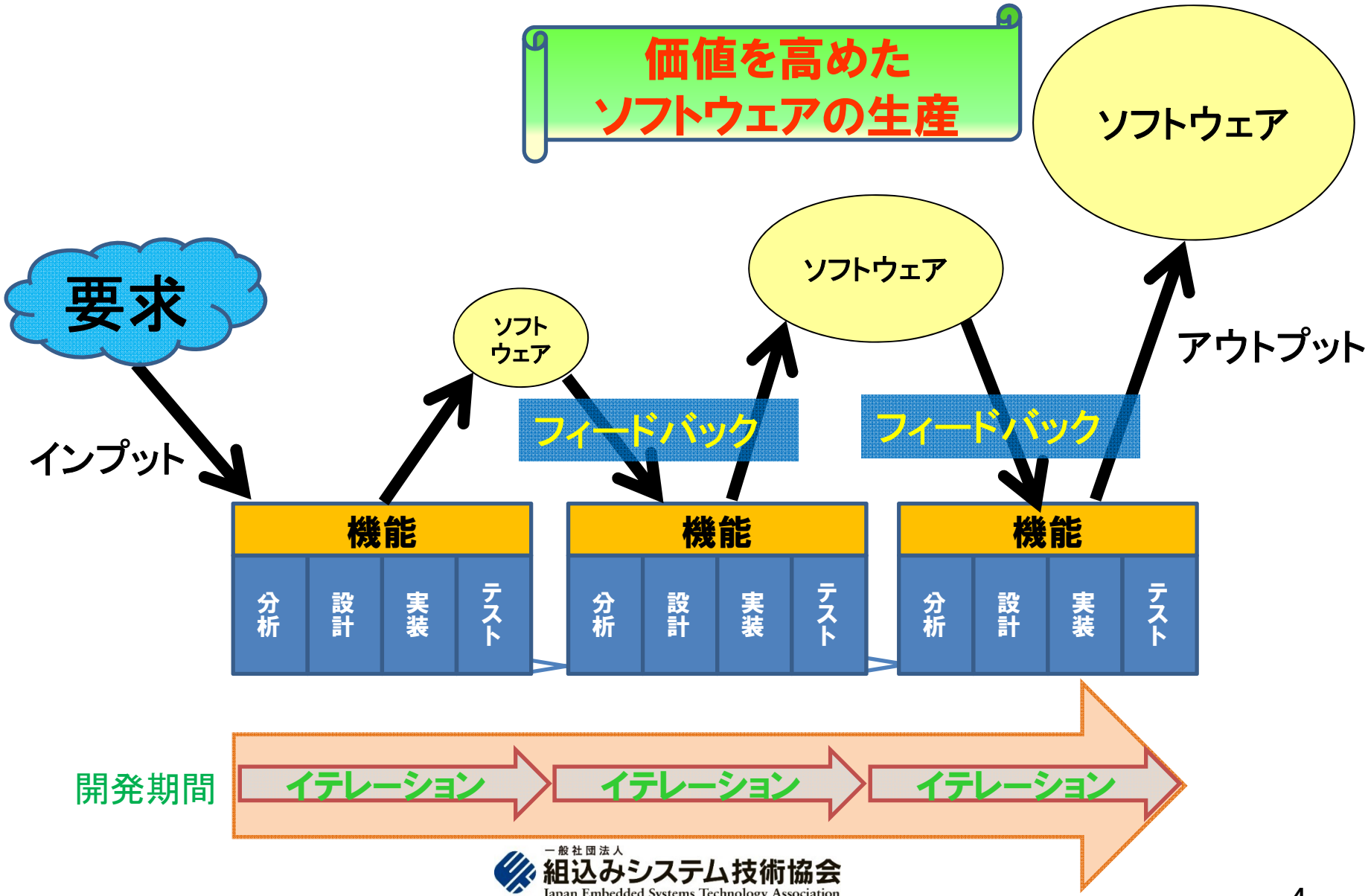
この考え方は業務に有効では！

さっそく、取り入れてみよう！

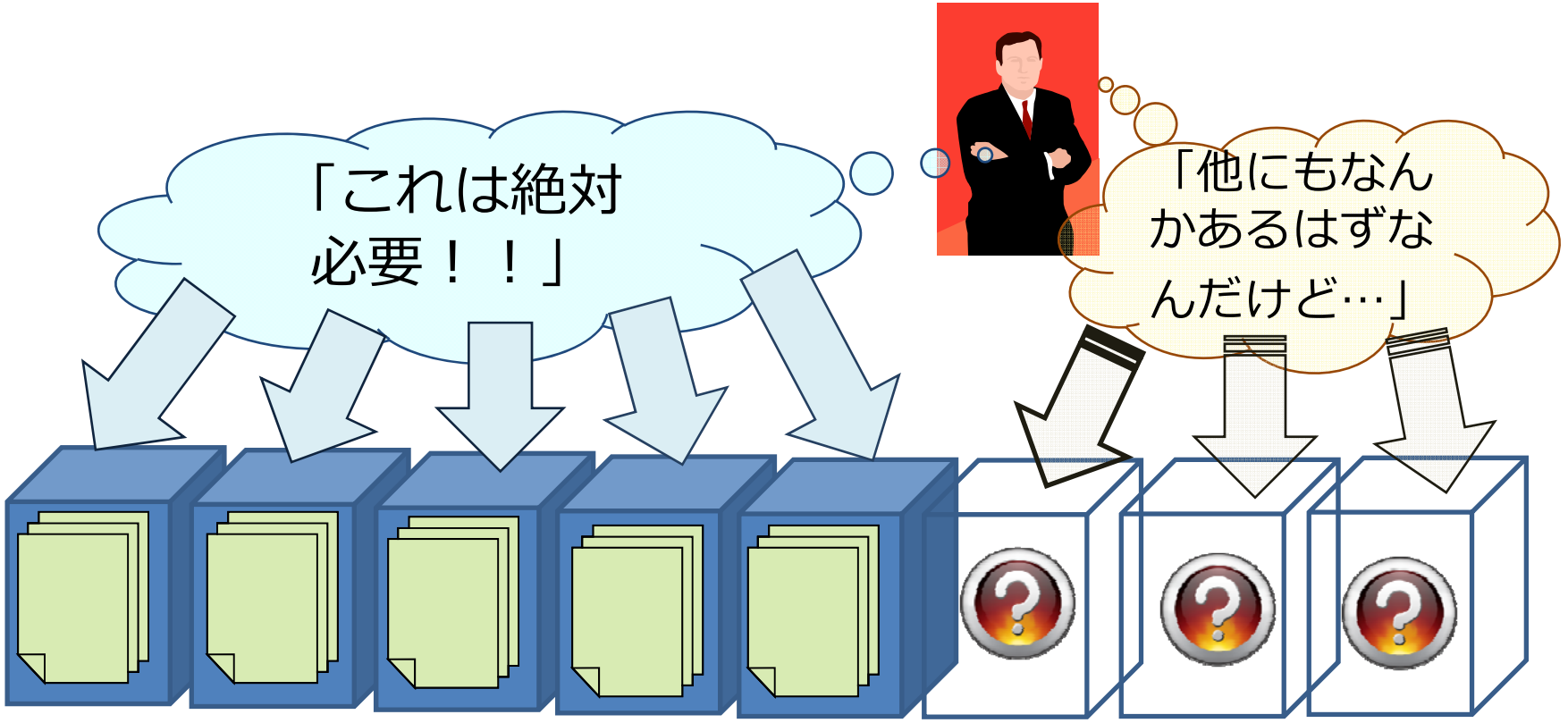
アジャイル的な考え方ができるようになってきた！

一般社団法人
組込みシステム技術協会
Japan Embedded Systems Technology Association

アジャイル開発について



アジャイルな契約



こっちは100%
確実にやりますよ。

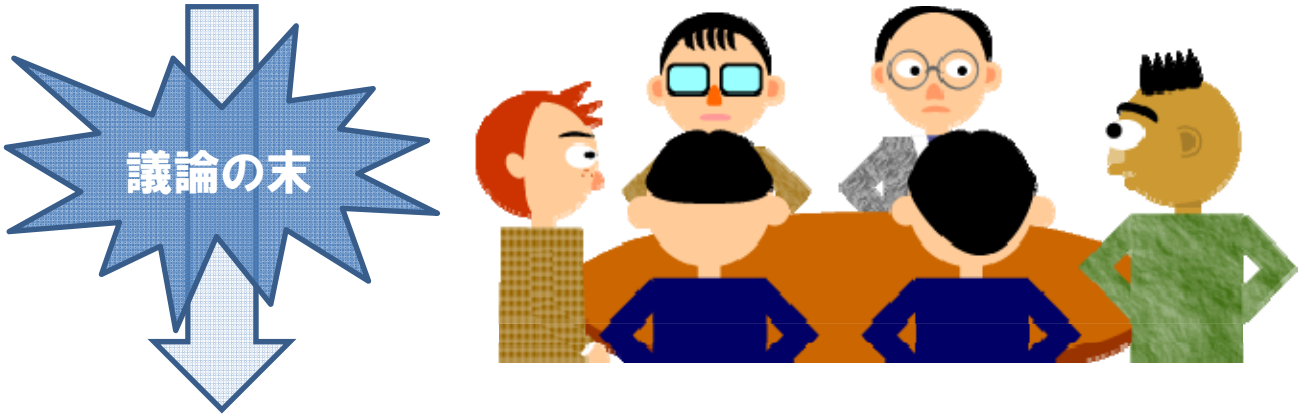
箱におさまる量なら、要
件が確定次第やりますよ。
準備もしておきます。

一般社団法人
組込みシステム技術協会
Japan Embedded Systems Technology Association

IPA発行のアジャイルガイドラインに対する検討

IPA発行のアジャイルガイドラインのプラクティスが
組み込み業界で実施できるかを検討

各社の見解を集計「できない」数→全体の**30%**



「できない」数→全体の**1%弱**

「できる」になった理由
- **実施していないから「できない」を選択していた**
- **適用「できる」方法が分かった**
- **開発体制の問題**

etc…

できないと考えた項目

■　問題のある「技術・ツール」

プラクティス名	内容	問題の原因
テスト駆動開発 	テストコードとプログラムを平行開発。最低限の機能から始めて、少しずつ機能拡張する。	ソフトウェアは少しずつ作れても、ハードウェアは完成しないとでてこない。ので、その時点ではテストができない。
ユニットテストの自動化	コスト（リソース）削減のため、既存のテストを自動実行できるようにする。	環境（特にハードウェア）が揃わないとテストできない領域がある。コスト面で自動化は見合わない。
逐次の統合	複数の修正を一度に結合しない。一結合一機能で動作確認する。	「ユニットテストの自動化」が前提である。
継続的インテグレーション	変更のある/なしに関わらず、定期的に結合・確認を実施する。	「ユニットテストの自動化」が前提である。

「自動化」と「環境」の相性が課題。

アジャイル研究会での試行

■ プロトタイプ開発にアジャイル的な要素を適用

- 会社の開発プロセスにアジャイルが適合しない
- どこまで機能を入れるか決まらない
- まずは動くものを顧客に見せたい

■ 同様な開発をウォータフォールとアジャイルで実施

・従来の手法でウォータフォール開発を実施

・最初の開発がうまくいかず、一部をアジャイル的に開発実施

■ 現場で適応事例の例

・管理的なアプローチ

・技術的なアプローチ

プロトタイプ開発での事例紹介

■ アジャイルを採用した理由

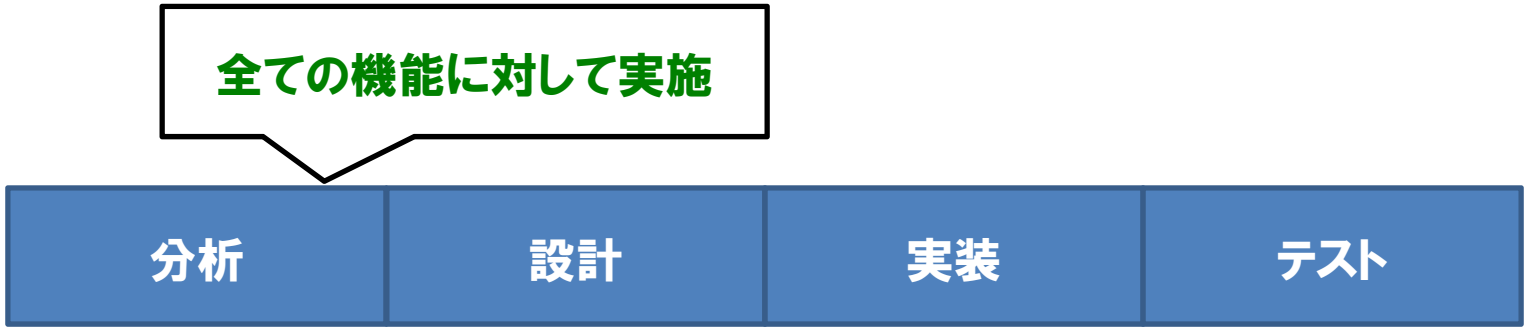
- まずは動くものを顧客に見せたい
- どこまで機能を入れるか決まらない

■ 開発業務の特徴

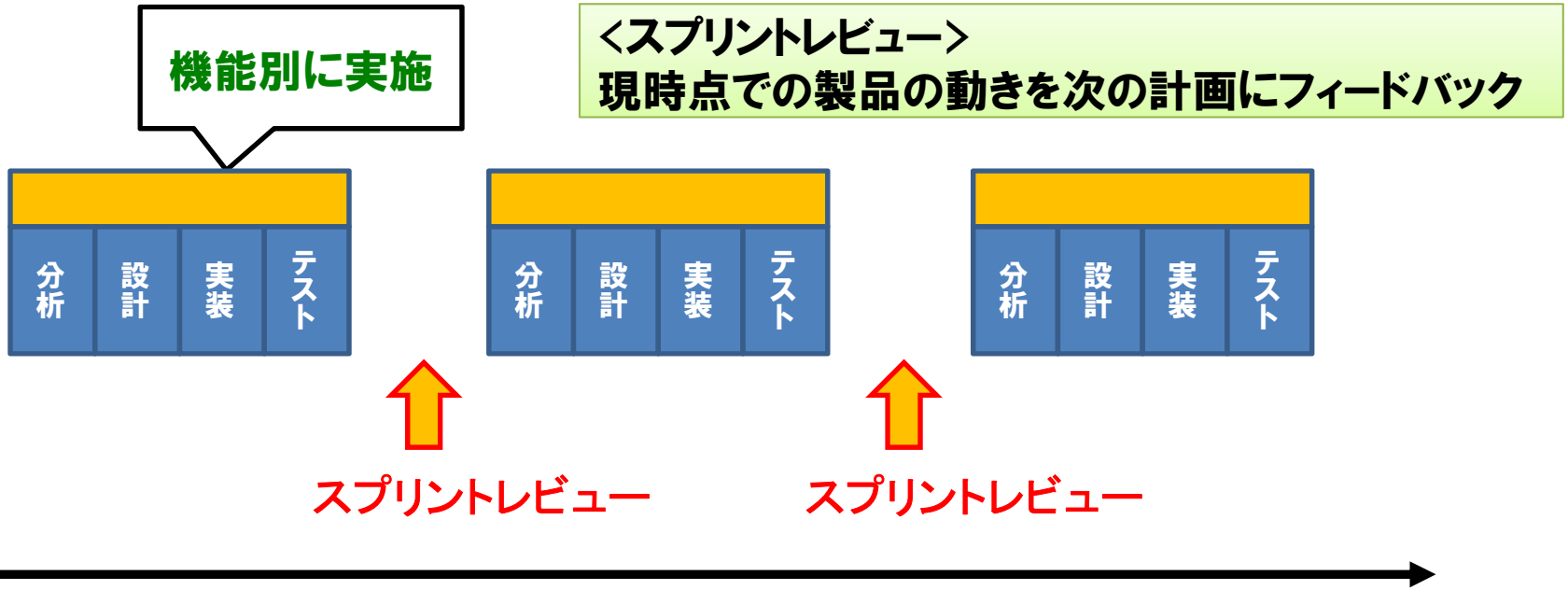
- 販促用デモセット開発→プロトタイプ開発
- 通信アナライザー開発
- PC上でのデモができるように作成
- 複数の機能を持ったソフトを作成

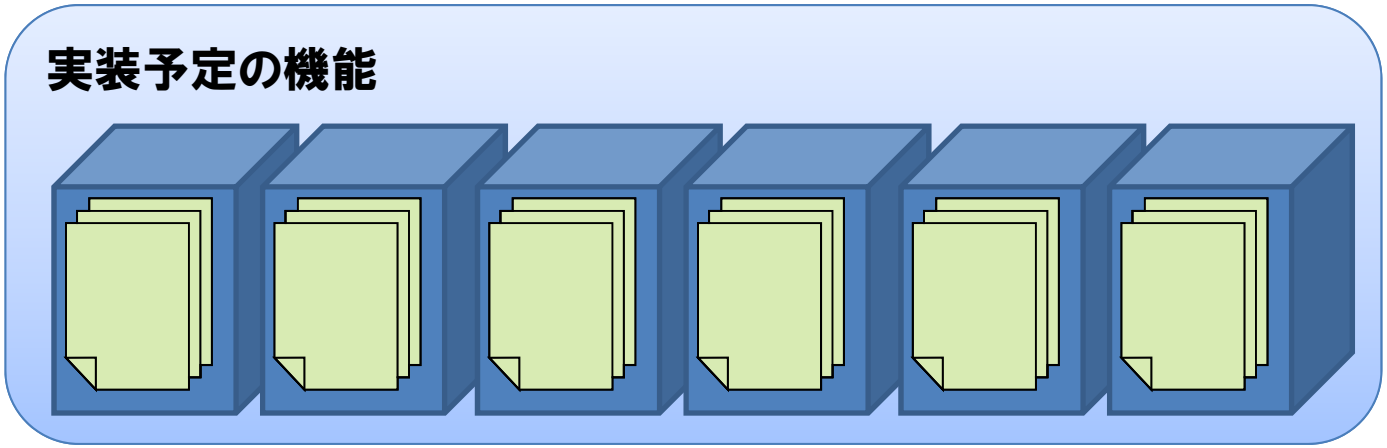
開発の進め方

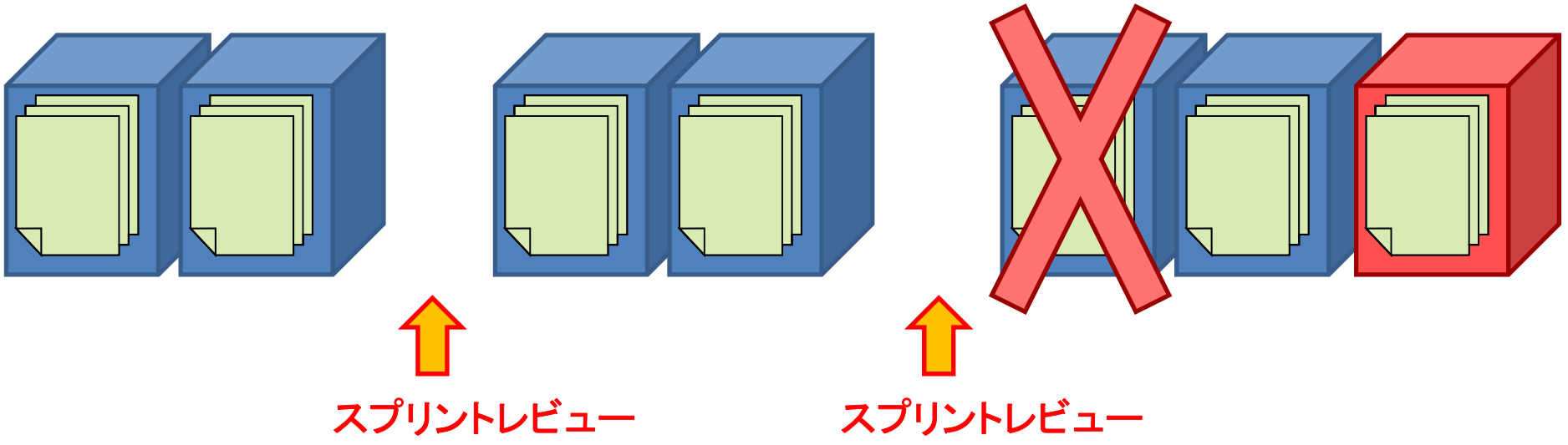
【従来】


全ての機能に対して実施
分析
設計
実装
テスト

【今回】


機能別に実施
<スプリントレビュー>
現時点での製品の動きを次の計画にフィードバック
分析
設計
実装
テスト
分析
設計
実装
テスト
分析
設計
実装
テスト
スプリントレビュー
スプリントレビュー
時間


実装予定の機能


スプリントレビュー
スプリントレビュー

結果に基づいた議論

■ ふりかえり結果

- 顧客の反応を早く入手

↓

- 不要な機能を作らずにすんだ
- 新たな価値を提供することができた

■ 注意点

量産開発においては
社内の標準プロセスに準拠していることが前提

組み込みソフト開発での事例紹介

- 同じような2つの開発を異なるプロセスで実施
  1．ウォーターフォール
  2．ウォーターフォールから一部プロセス変更

- 開発業務の特徴
  ・組み込みのソフト開発
  ・量産品のソフト開発
  ・自社製品の開発

ウォーターフォールでの失敗

■ 問題が発生

1．開発の進捗遅れが発生

2．不具合が多発

製造後に再度工場を稼動させ、ソフト書き換えを実施する事態になった

■ 原因

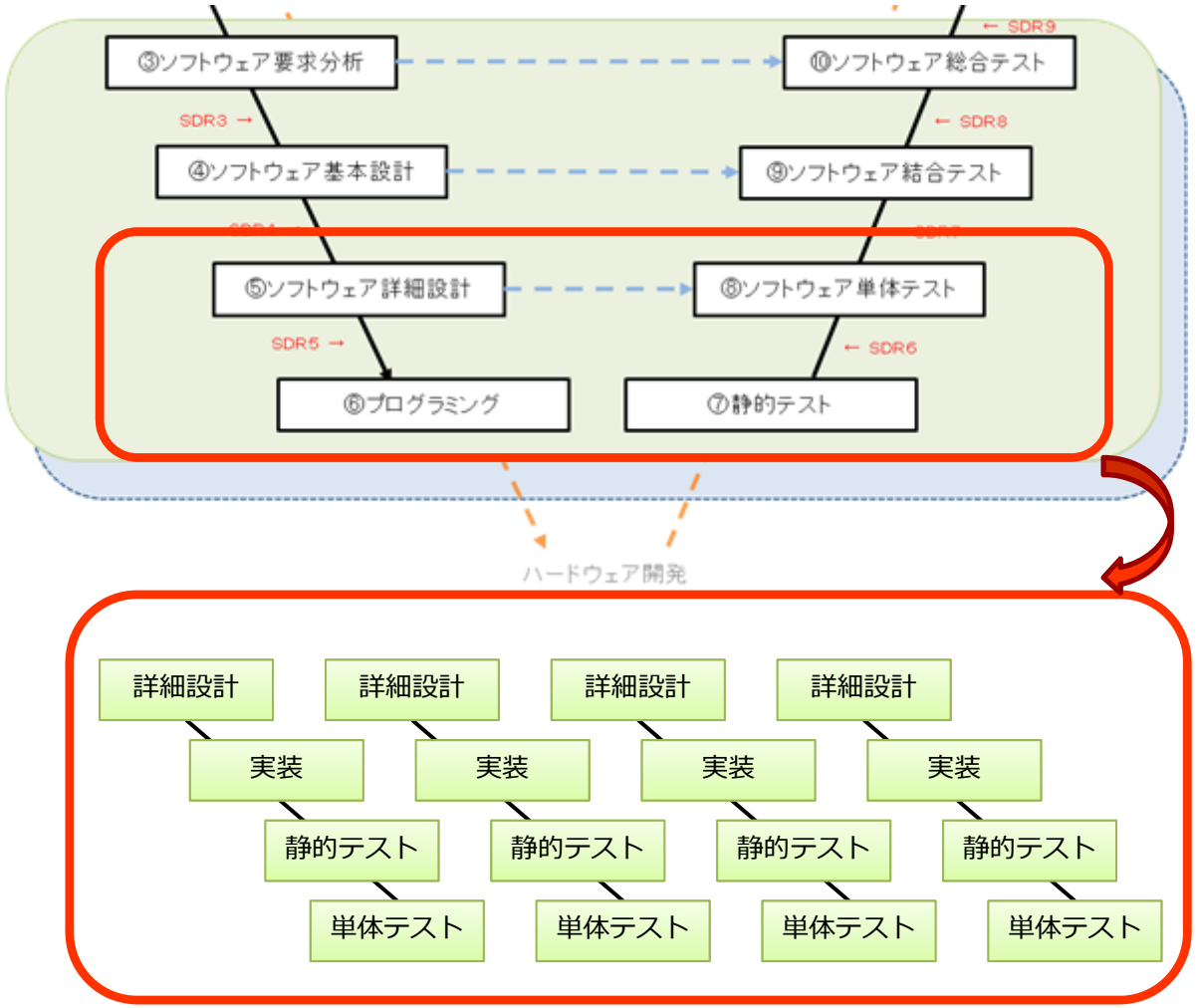
・進捗遅れ

- 見積りが甘かった
- ハードの仕様決定が遅かった

・不具合多発

- 納期を遵守するため、テストにかける時間を削減した

開発プロセスの一部を変更



変更プロセス内の基本方針

<原則>

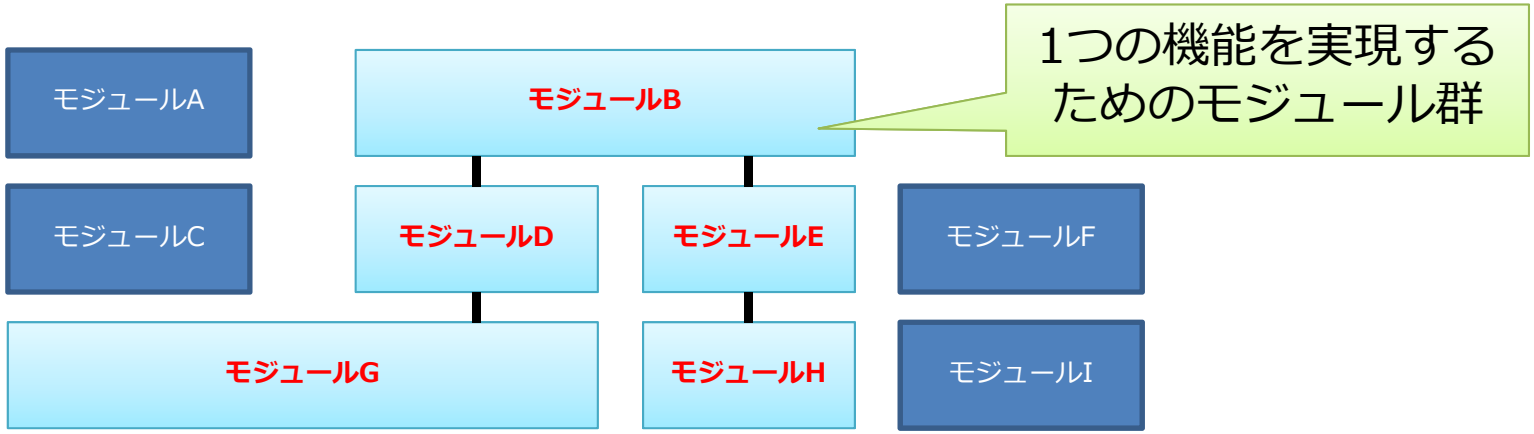
・止まらない。
（他の仕様決定を待たない）
・戻らない。
（仕様変更があっても「今」は無視）
・優先順位の高い機能から順に
（基本設計で機能に順位付けを実施）
・それぞれの箱の中でやるべき事は変えない

変わったところ

- 詳細設計～単体テストを実施する単位
  - モジュールごと
    (モジュール分けやその粒度は基本設計で決定済み)
- ハードとの関係性
  - 結合テストまでは必要ない
  - ハード依存モジュールは独立性を重視
    (仕様変更があっても上位モジュールへ影響を与えない設計とする)
  - 未FIXの仕様は適当（適切）に仕様を仮FIXさせる

詳細

- 機能はいくつかのモジュールが結合されて実現される。
- 詳細設計はモジュール毎に行っている。



結果

- 幾つかの機能の優先順位を落としたが
  不具合は発生しなかった
  (進捗遅れが発覚した時点で、優先度の低い機能は後の回した)

- 搭載した機能に対しては規定されたプロセスを全て実施できた
  (レビューの省略、テストパターンの削減などは行っていない)

ウォーターフォールとの比較

■ 同じところ

- やるべき作業は変わらない

■ 異なるところ

- 製品としての価値

<ウォーターフォール>

製品としての価値は”1” or ”0”

-ある程度完成(全ての機能を搭載)
※ただし部分的にテストが未完了

<一部プロセス変更>

製品としての価値は”0.7”ぐらい？

-優先度の高い機能は搭載(テスト実施済み)
※ただし一部優先度の低い機能は未搭載

押さえるべきポイント

- 進捗状況の見える化
  (見通しがよくなる状況を作る)
  - モジュール単位で完結させる
  - 止まっている状況をなくす
  - 作業が進むごとに見積り精度が向上する
    (実績値に基いて精度が向上する)

- 品質の確保(やるべきことを変えない)
- プロダクトオーナーは必須
  (進捗状況を迅速な意思決定に繋げるため)

現場での適用事例に対する検討

- プロジェクトマネジメント的要素
  - 朝会
  - タスクボード
- 技術的要素
  - リファクタリング
  - TDD(テスト駆動開発)

朝会

①問題点
　同グループメンバの作業状況を把握できず、マイルストーンに一部作業が間に合わなかった、作業負荷が特定のメンバのみに掛かった

②目的
　グループメンバの作業内容・作業進捗を共有し、管理すること

③対策
・毎朝グループメンバが集まり、各自の進捗・本日の作業を報告する
・(5，6人のグループで)平均15分程度
・作業遅れがある場合、順調に進んでいるメンバに作業を割り振る、リスケジュールをする等、作業調整を行う
・リーダが進行役となり、メンバの状況を把握し作業調整を行う

請負開発ではプロダクトオーナーとの情報共有が滞らないように

タスクボード

①問題点
　必要な残作業が見えておらず、マイルストーンに作業が間に合わなかった

②目的
　作業グループ内での、マイルストーンまでの残タスクを見える化し、コントロールすること

③対策
- タスクを付箋に記載する
- 付箋には、タスク名、作業予定メンバ名、マイルストーンを記載する
- タスクボードにタスク(付箋)を貼りだす
- タスクボードはグループメンバが常に見える場所に設置する
- タスクボードにはTODO,DOING,DONEを設け、そのタスクのステータスに応じた場所に付箋を貼り付ける
- タスクボードを確認・更新する時間を、毎日設ける

リファクタリング

①問題点
　すぐに動くものをという要求にこたえるため、人数をかけて一気に作り上げた。その際、同じような機能のため、クローンコードが多数できてしまっていたり、さまざまなアーキテクチャが入り乱れることとなり、その後の修正が大変になった。(保守性の低下)

②目的
　ニーズを想像し、今後の仕様変更や仕様追加に耐えれるようアーキテクチャを修正すること。

③対策
・全体のアーキテクチャ設計をしなおす
・ソフト単位を統一
・層を決め、配置しなおす
・アーキテクチャを統一させる

TDD(テスト駆動開発)

①問題点
　使うことを考えると簡単に見つかる問題がテストにて発見される。特に後工程で見つかることも多いため戻り作業も多くなりがちで無駄な工数が取られている。

②目的
　テストを先に考えることで実際にその機能を使う状況という視点をいれる。そうすることで、仕様の行間や矛盾点を早期発見し、無駄な工数を削減する。

③対策
- 機能の動く背景などを機能を使う状況を考え、チェックリストを作成する
- 要求者とチェックリストにて要求の確認を行う
- 仕様に「なぜこの仕様になったか」をフィードバッグする

現場での適用事例に対する課題

- プロジェクトマネジメント的要素
  - プロダクトオーナーとの合意形成が迅速に行えるか
    (請負開発では、顧客がプロダクトオーナー)
- 技術的要素
  - リファクタリング：タイミングが重要
    (あらかじめ計画を立て、顧客と合意したうえで実施する)
  - TDD：ハードとソフトの境界が重要
    (ある程度まではシミュレータが準備できれば実施できる)

まとめ

- プロダクトオーナーとの迅速な意思決定
- 決められたマイルストーンにおける製品の価値の最大化を目指す
- 開発プロセスの一部にアジャイルにすることでも、大きな効果が得られる

最後に

「アジャイル開発の試行と検討」

2015/11/18 発行

発行者　一般社団法人 組込みシステム技術協会
東京都中央区日本橋大伝馬町6-7
TEL: 03(5643)0211 FAX: 03(5643)0212
URL: http://www.jasa.or.jp/