アジャイルと
派生開発手法の活用による
開発改善について

2017/11/15

JASA中部

アジャイル研究会

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

はじめに

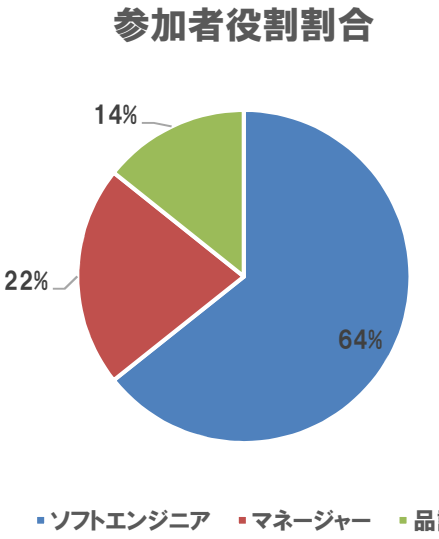
- 今年のアジャイル研究会はアジャイルプロセスからさらに外れています。
- アジャイル研究会でのアジャイルの定義
  - アジャイル研究会が発足して様々な議論したところ以下の定義となった
    - アジャイルはマインドである
    - スクラム＝アジャイルではない
- アジャイルな考え方をもって開発を良くしていきたいと思っているメンバーがあれこれやった結果をご紹介する

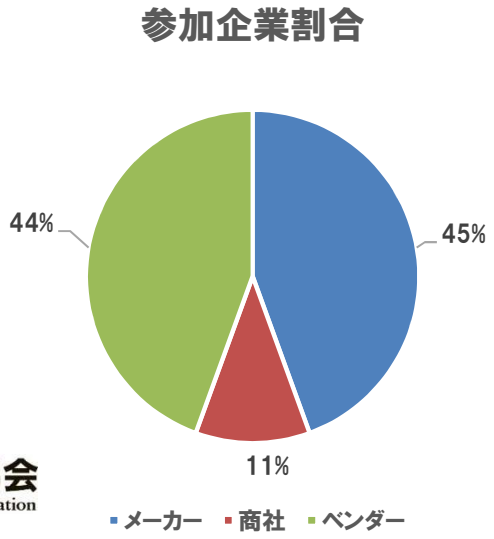
Agenda

1. 研究会紹介と今回のターゲット
2. 今回改善のテーマ悩みの分析
3. AFFORDDとの連携
4. USDMについて
5. 実改善例

研究会紹介

- 秋谷勤
  - 2016年～　研究会会長
- 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
  - アジャイル開発を行ったことがないメンバーが『アジャイル』というキーワードだけで集まった
- 研究会所属メンバー割合





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

ターゲット

- 組込みエンジニア
- 以下のような構造のソフトウェアを作成しているエンジニア
  - ハードウェアとのソフトが切り分けられている
  - 機能単位の作りこみが可能

会員の悩み事

■ アジャイルな考え方でソフトウェア開発をよくするために会員の悩み事を募った。

- マネージャーがやることがわかっていない
- 不具合修正でデグレードが発生する
- いつまでたっても開発が終わらない（終わりがわかっていない）
- いつも忙しい
- 途中状態のものをレビュー（監視）されたくない
- 何をなおしたらよくなるかわかっていない
- 分業が進んでいて連携がうまくいっていない
- 他部署（品質）との連携がうまくできていない
- 機能が大きすぎてうまくできない

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

悩みの分析

■　出てきた悩み事の分析、深堀を行った

結果

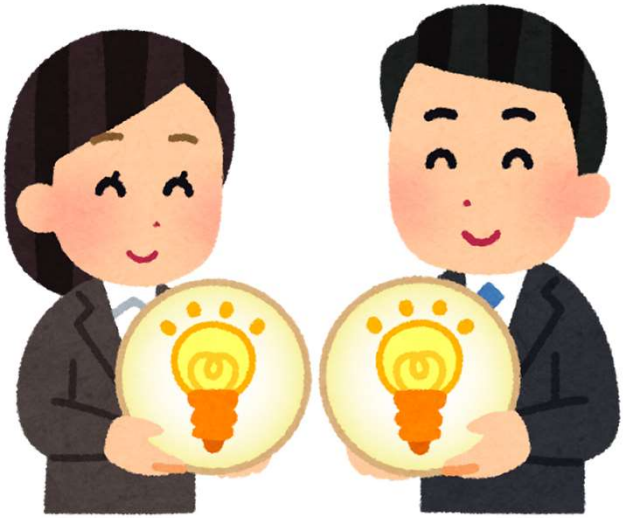
共通した悩みとして
コミュニケーションの問題が現れた！
今回はこれを解決する！

コミュニケーションの改善

■ 悩みの分析中にいくつかの改善策も出てきた

- コミュニケーション結果の見える化
- 伝えたいこと、理解したいことの構造化
- ドメイン用語のトレーサビリティ

これらができれば悩みが減るのではと考えた！



AFFORDDとの連携

- コミュニケーションの改善で困っている点を深堀していくとお互いの認識の粒度のズレの問題が多いことが分かった
- 情報の構造化という点でUSDMという記述法が適しているのでは？という意見が出た

- USDMは派生開発プロセスである**XDDP**の一部であった
- **XDDP**を推奨している派生開発推進協議会（AFFORDD）連絡をとりお話をきくことに

XDDPの概要

■　XDDPは以下の特徴を持っているとのこと

1. 変更内容を「変更の要求」と「機能追加の要求」に分け、
   別々のプロセス（変更プロセスと追加プロセス）で対応する

2.「変更の要求」に対しては
「変更要求」と「変更仕様」に分け、階層構造の中で扱う

3. 変更仕様をソースコードのレベルで捉える

4. 基本的に「差分」で作業を進める

5. 「変更３点セット」の成果物で協同作業を円滑にする

要求（実現したい事）をはっきり表現するのがポイント

要求を表現する

例えば、

・10%はもち米にしてね
・水は線より5mm少なくしてね

もち米は水を吸うから
ふっくらにするためには
少なめにしないとネ。

要求あいまい…

固めのご飯が
食べたいのか？
弱火で炊くか…

より、

ふっくらご飯が食べたいの。
・10%はもち米にしてね
もち米は水をよく吸うから、
・水は線より5mm少なくしてね

要求ばっちり！

ふっくらご飯が
食べたいのね。
炊き方そのままだ

⇒要求が伝わると間違いない

USDMの概要

■ USDMは、以下の特徴を持っている

・『**要求**』と『**仕様**』を階層構造で表現

・『**仕様**』は動詞で振舞いを**定義**

<USDMを用いた要求仕様書>

要求	A.1	Setボタンを押すことにより、タイマ時間を設定できる	
	理由	ボタン1つで簡単にタイマ時間を設定できるようにしたい	
	□	A.1-1	タイマ時間は1分～99分の間で設定することができる
	□	A.1-2	Setボタンが100ms以上、500ms未満の間で押される毎に、タイマ時間に１分加算し、内部タイマの秒数を0にする
	□	A.1-3	タイマ時間が99（分）の時に、Setボタンが100m以上、500ms未満の間で押された時は、タイマ時間および内部タイマの秒数を０にする
要求	A.2	Startボタンを押すことにより、タイマーを起動する	
	理由	任意のタイミングで計測をスタートしたい	
	□	A.2-1	
	□	A.2-2	

要求：
実現してほしいこと

理由：
要求に至った背景

仕様：
要求を満たすべき
具体的な振舞い

USDMを採用

- **まさにこれだ！**

- コミュニケーションを補強するツールとして**XDDPの技法である**USDMを取り入れてみることにした

適応事例2点

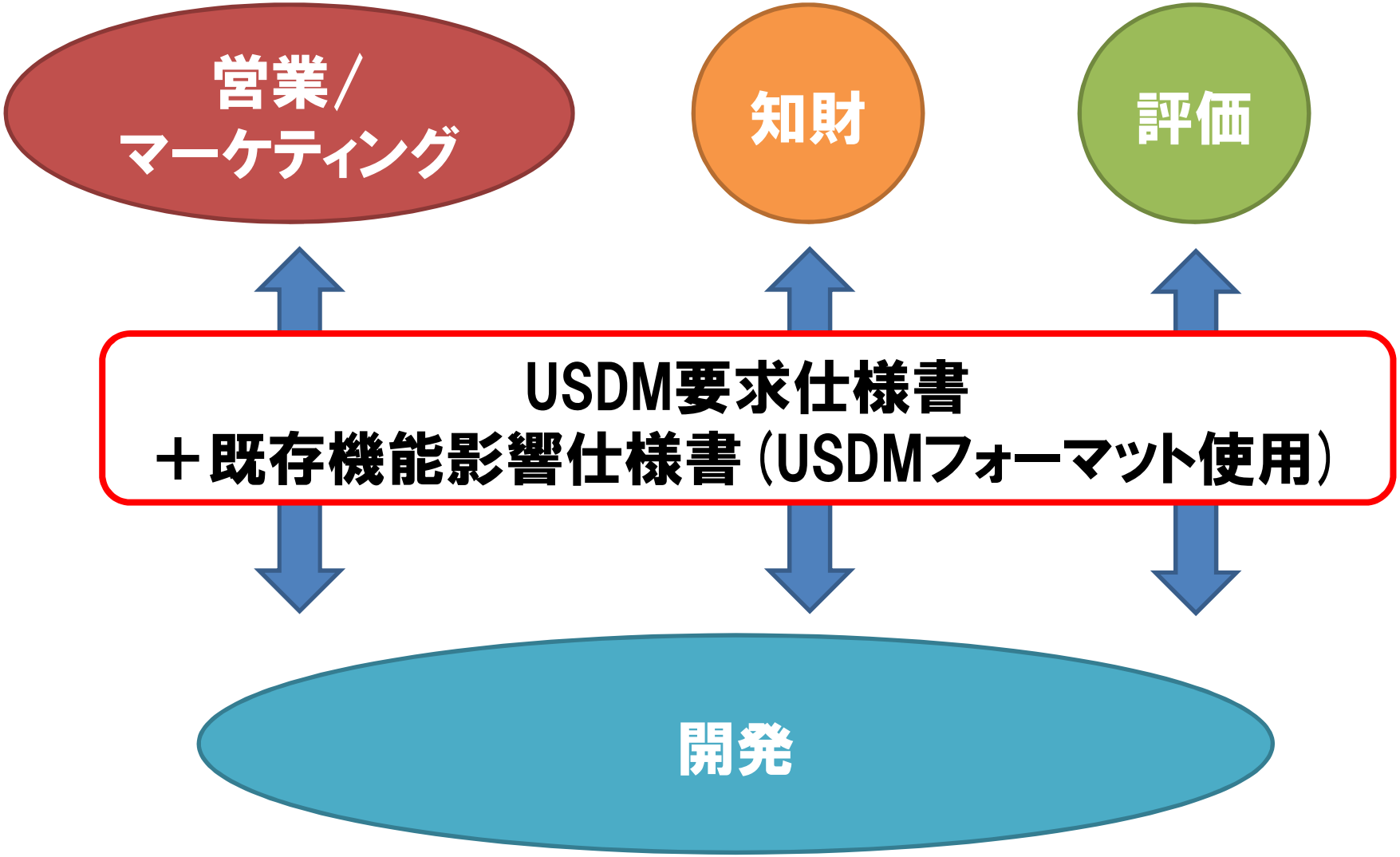
■ 他部署との連携をスムーズにした例

- 様々な部署がそれぞれ必要とする情報が違うものをまとめた例

■ 厳しい納期要求に答えた例

- ソフトウェアをあまり知らないハード側のエンジニアとのコミュニケーションツールとして

他部署との連携をスムーズにした例



既存機能影響仕様書

要求	A.1	Setボタンを押すことにより、タイマ時間を設定できる	
	理由	ボタン1つで簡単にタイマ時間を設定できるようにしたい	
	□	A.1-1	タイマ時間は1分～99分の間で設定することができる
	□	A.1-2	Setボタンが100ms以上、500ms未満の間で押される毎に、タイマ時間に１分加算し、内部タイマの秒数を0にする
	□	A.1-3	タイマ時間が99（分）の時に、Setボタンが100m以上、500ms未満の間で押された時は、タイマ時間および内部タイマの秒数を０にする
要求	A.2	Startボタンを押すことにより、タイマーを起動する	
	理由	任意のタイミングで計測をスタートしたい	
	□	A.2-1	
	□	A.2-2	

影響する既存の機能の変更概要を書く

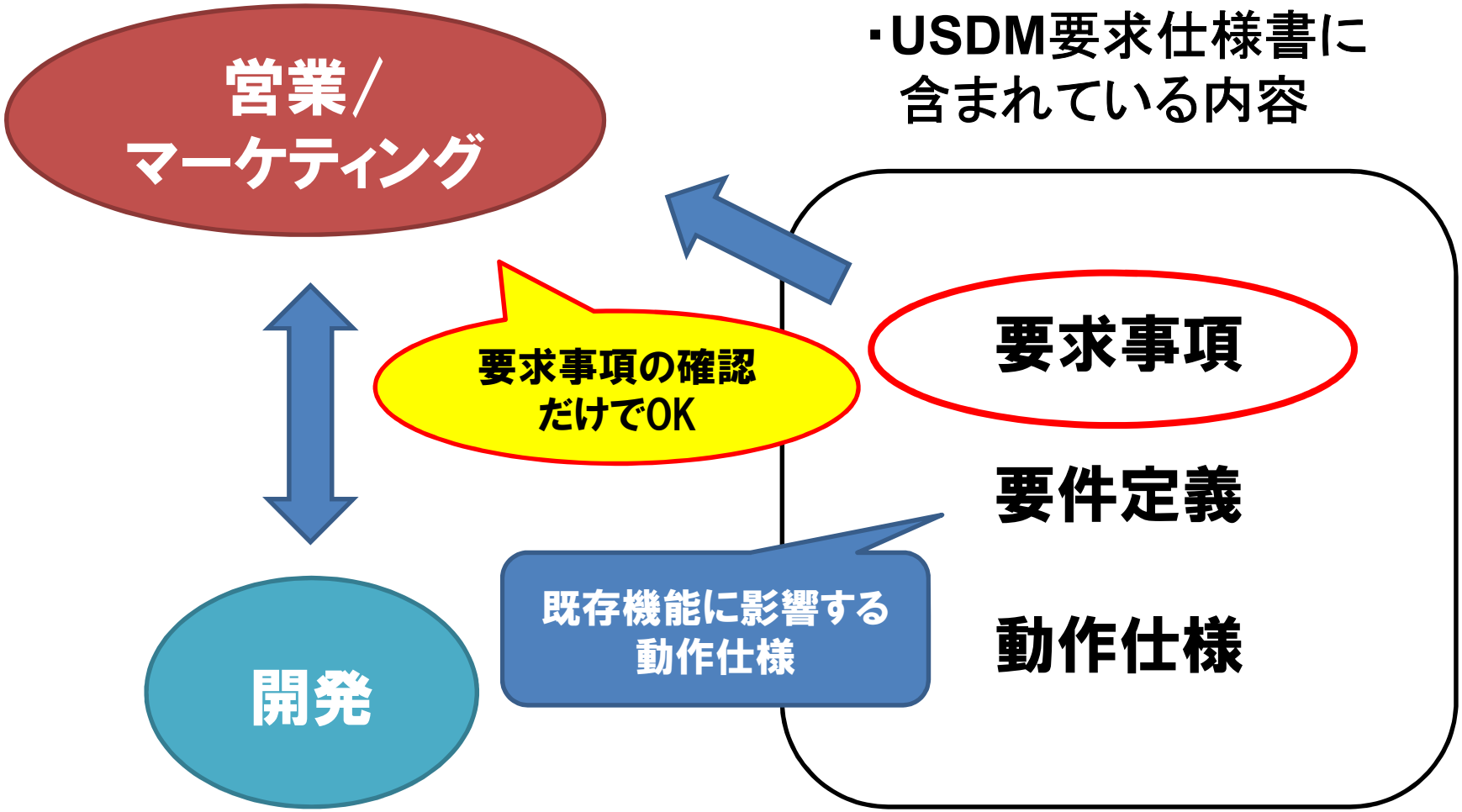
例: XX機能の動作中は
YY機能の動作を制限する。

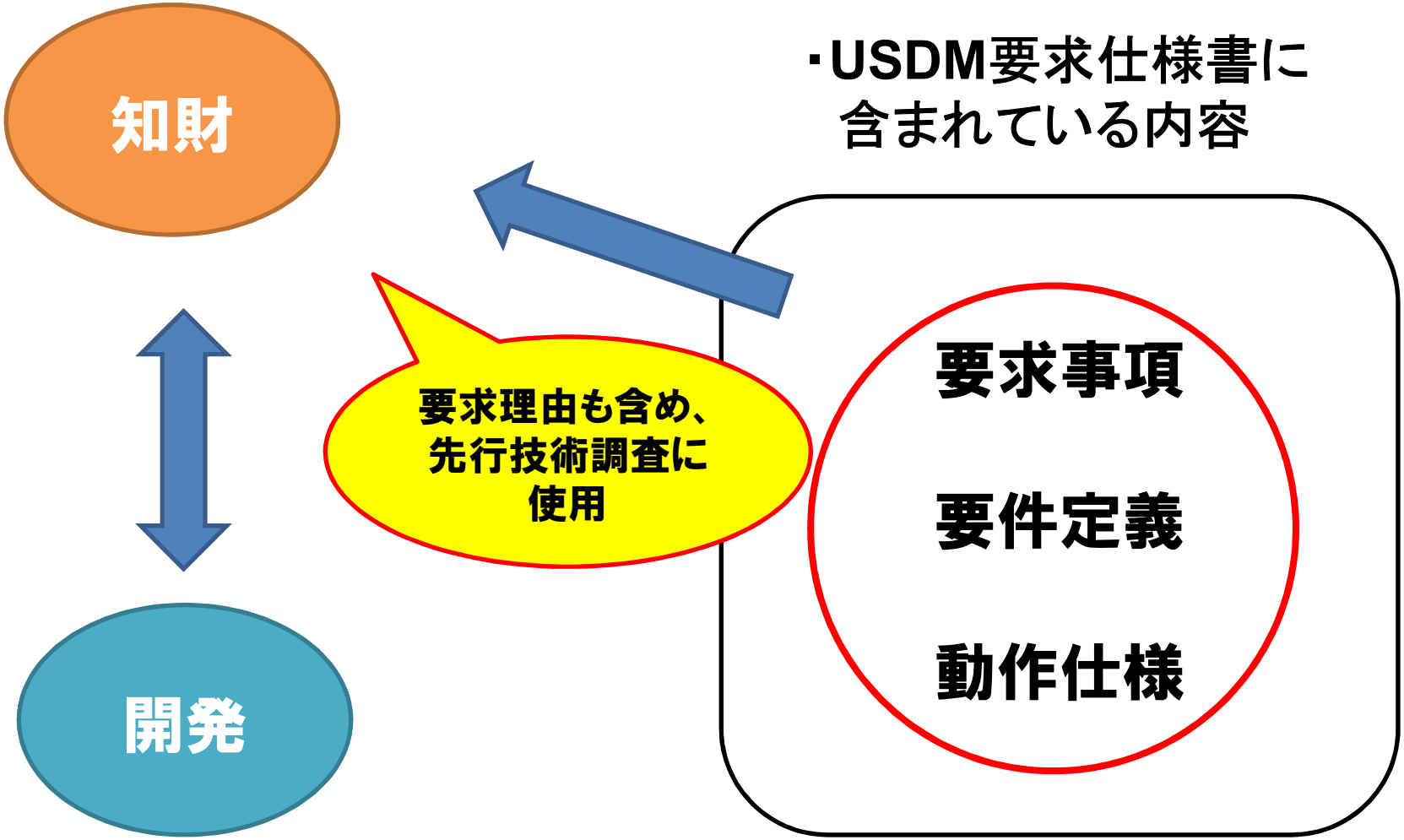
既存の機能の変更後の動作仕様を書く

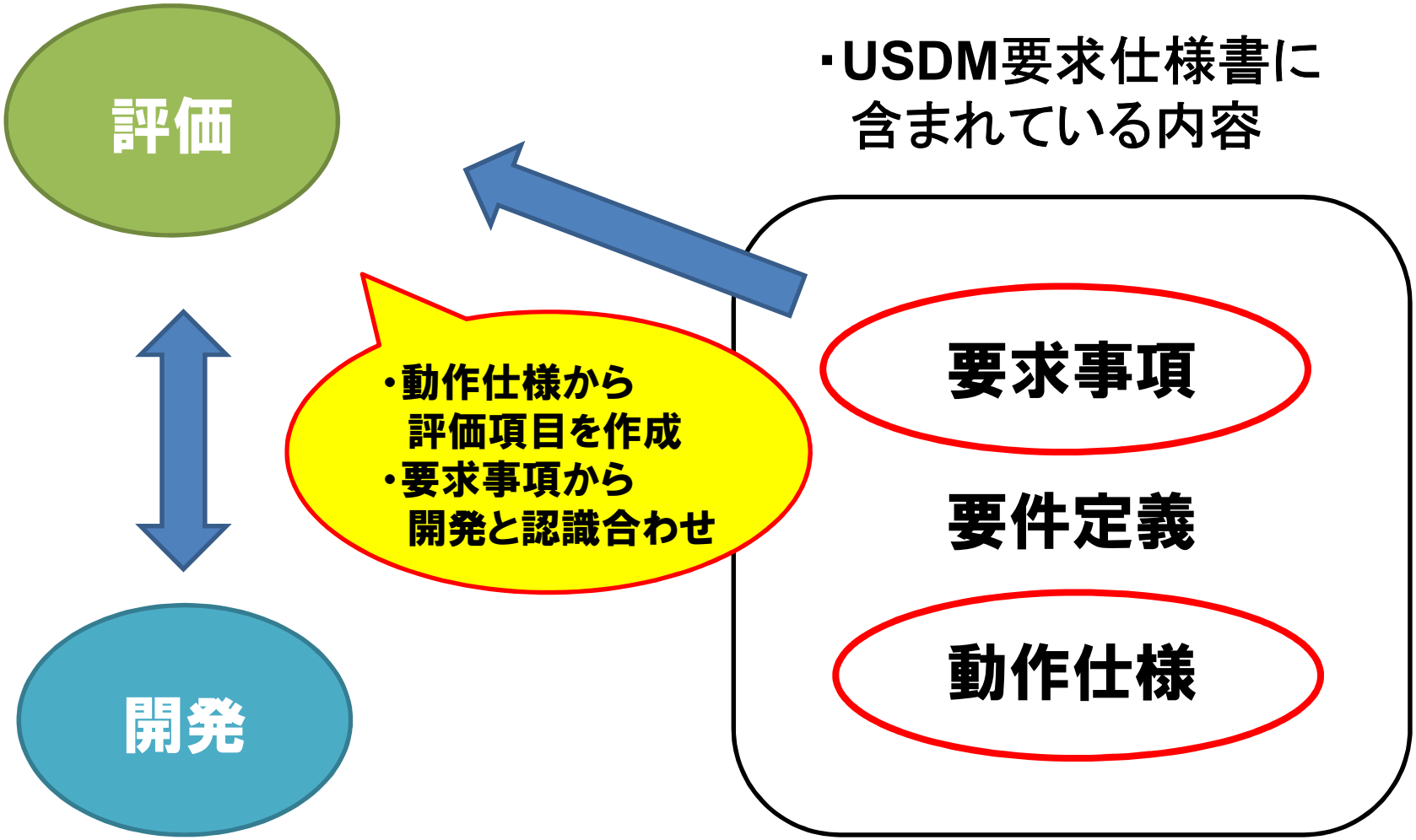
例: XX機能の動作中は
YY機能の設定変更時にZZする。

ソフトウェアの変更点がまとまっていて管理しやすい！

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


・USDM要求仕様書に
含まれている内容
営業/
マーケティング
要求事項の確認
だけでOK
要求事項
要件定義
既存機能に影響する
動作仕様
動作仕様
開発


知財
開発
・USDM要求仕様書に
含まれている内容
要求理由も含め、
先行技術調査に
使用
要求事項
要件定義
動作仕様


評価
開発
・USDM要求仕様書に
含まれている内容
・動作仕様から
評価項目を作成
・要求事項から
開発と認識合わせ
要求事項
要件定義
動作仕様

効果

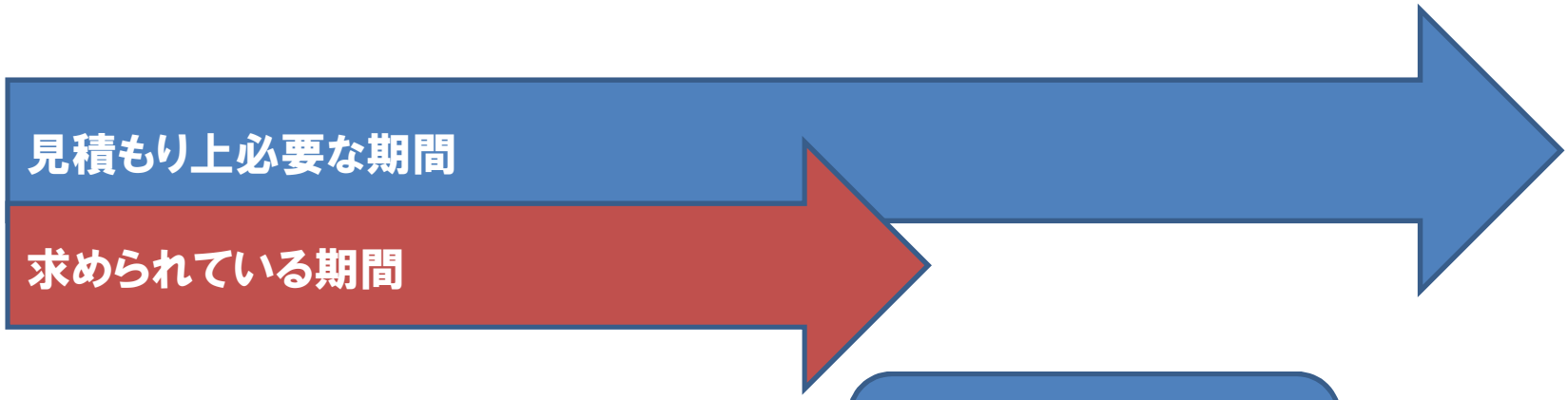
■ 良かった点

- 他部署との動作仕様の認識違いを少なくできた。
- 要求仕様の漏れの早期発見できた。
- 要求に対しての動作仕様が記載されているため、評価項目の作成がスムーズに進められた。

■ 悪かった点

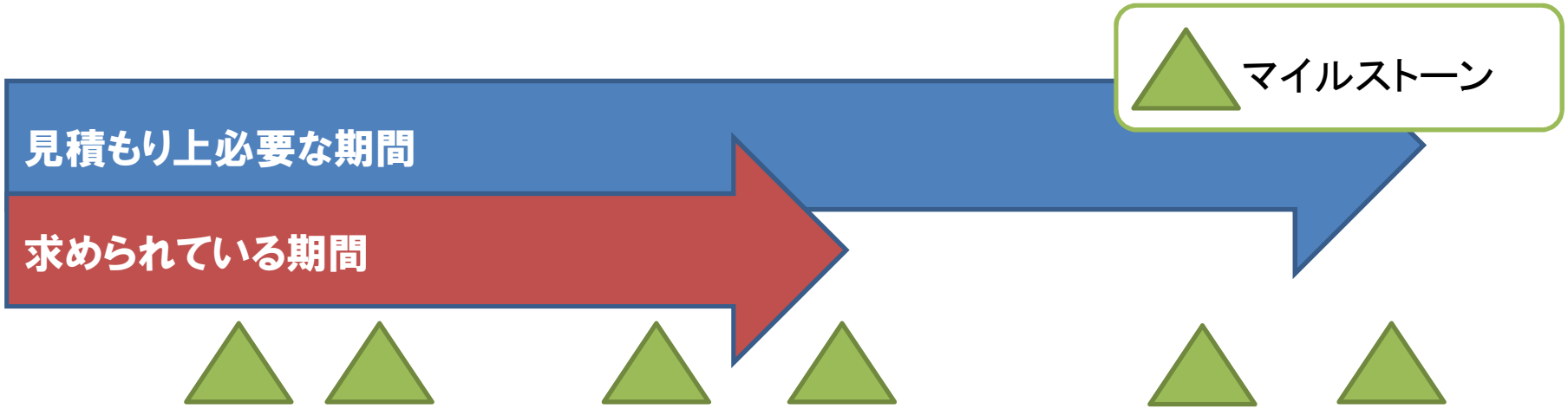
- 仕様書の作成/更新に時間が取られてしまった。
- 実装機能の規模が大きくなってきた場合に要求と仕様を記載する階層が多くなってしまい検索性が低下してしまった。

厳しい納期要求に答えた例



いつものやり方では
絶対無理！

実際の適応状況



- この時の開発はハードも並行で開発しており、すべてがその時点でできている必要が無かった
- 影響が少ない個所は希望納期後でも問題がないことが分かった

USDMの活用

要求	A.1	Setボタンを押すことにより、タイマ時間を設定できる	
	理由	ボタン1つで簡単にタイマ時間を設定できるようにしたい	
	□	A.1-1	タイマ時間は1分～99分の間で設定することができる
	□	A.1-2	Setボタンが100ms以上、500ms未満の間で押される毎に、タイマ時間に１分加算し、内部タイマの秒数を0にする
	□	A.1-3	タイマ時間が99（分）の時に、Setボタンが100m以上、500ms未満の間で押された時は、タイマ時間および内部タイマの秒数を0にする
要求	A.2	Startボタンを押すことにより、タイマーを起動する	
	理由	任意のタイミングで計測をスタートしたい	

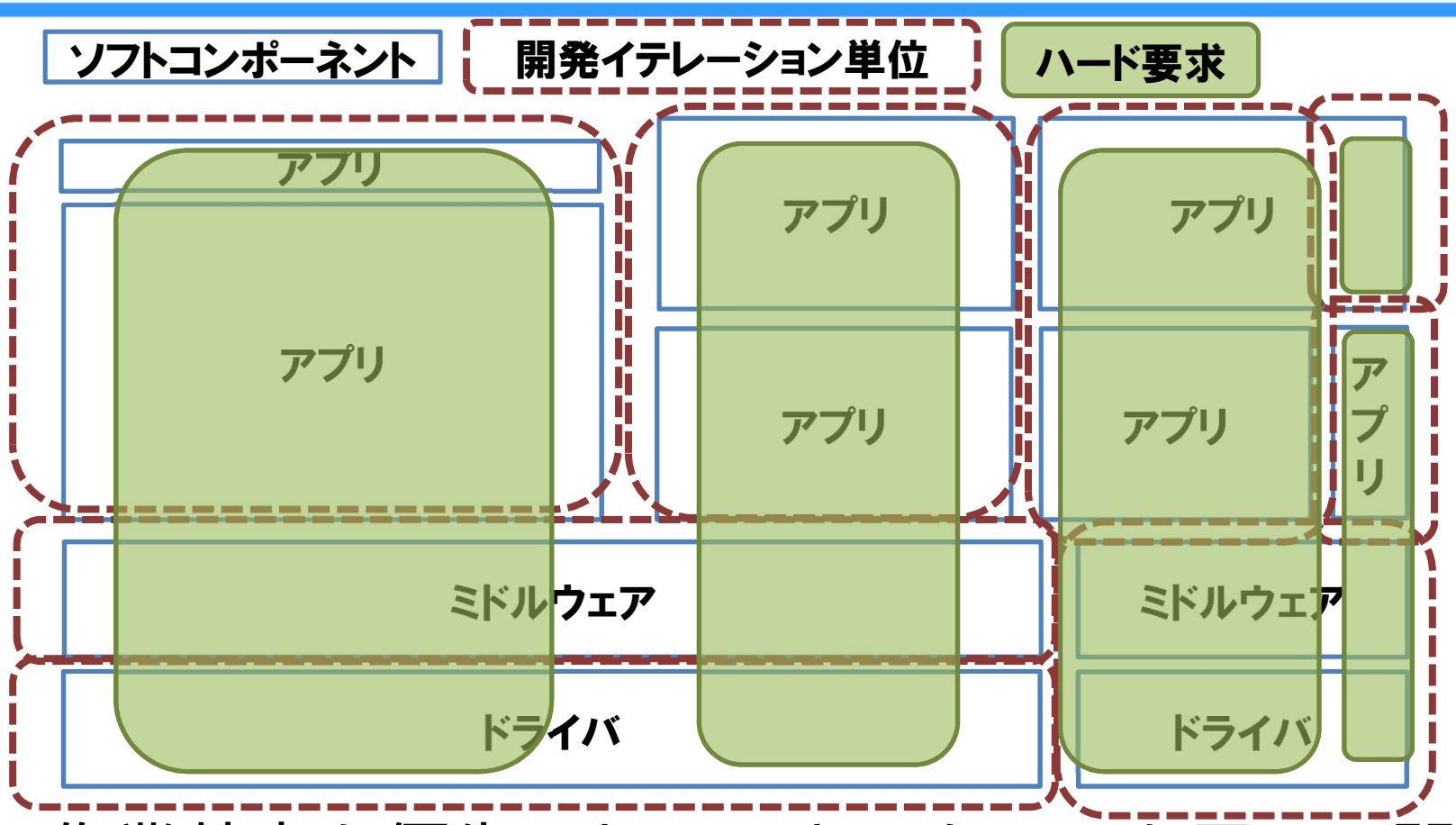
要求項目に
ハード開発者の
言葉を使用

仕様に機能化した
仕様を並べる

要求していることと
ソフトで作るものが
つながっているので
わかりやすい！

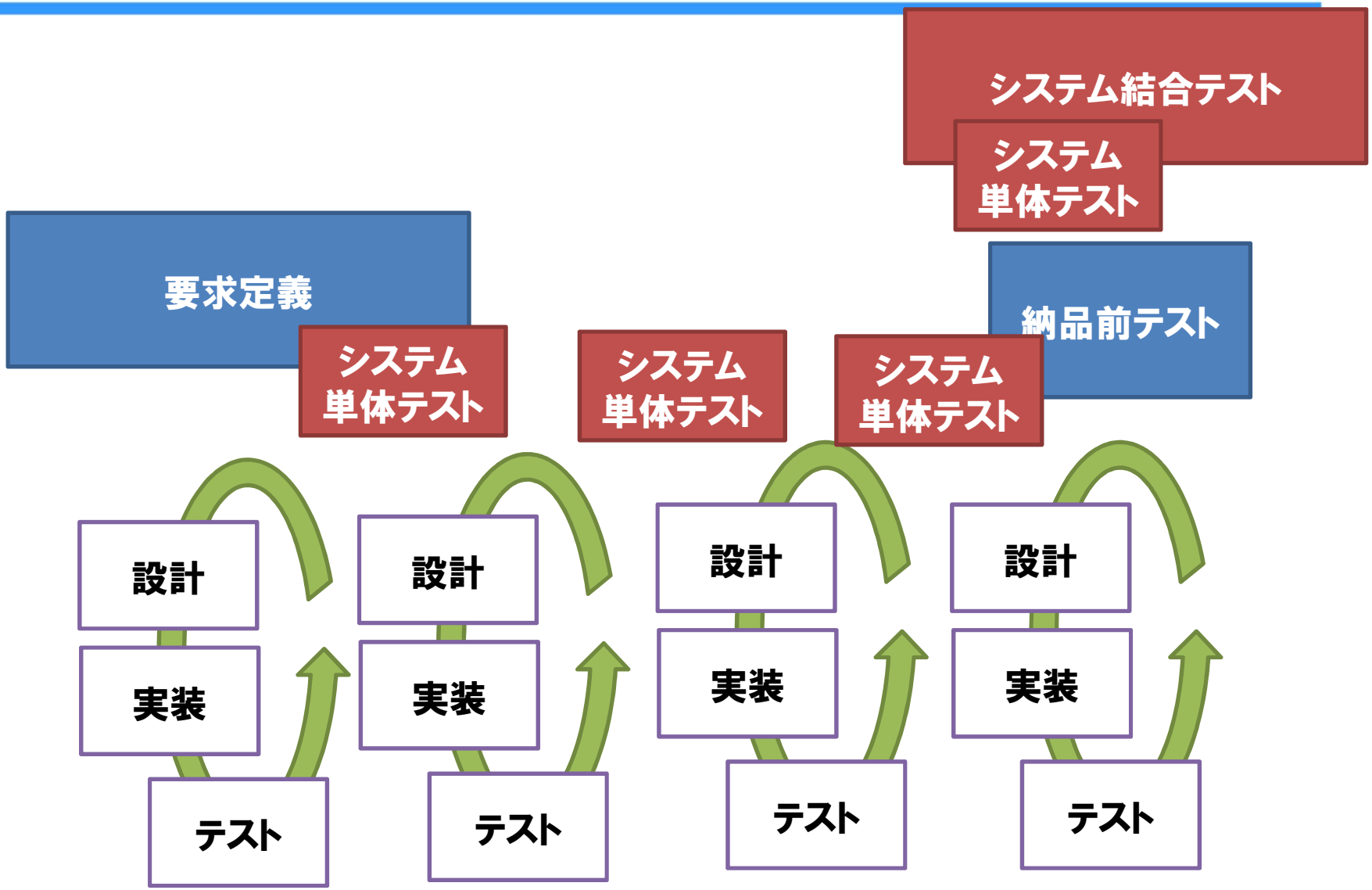
一般社団法人
組込みシステム技術協会
Japan Embedded Systems Technology Association

開発状況



- 作業効率を優先したアーキテクチャを用い、開発単位を分割
- テスト可能な単位で分割し、開発を進める

実際の適応状況


システム結合テスト
システム
単体テスト
要求定義
納品前テスト
システム
単体テスト
システム
単体テスト
システム
単体テスト
設計
実装
テスト
設計
実装
テスト
設計
実装
テスト
設計
実装
テスト

効果

■ 良かった点

- アジャイルの効果が出せた！
  - 1イテレーション事で動作ができるので、実査に動かすことで見えてくる変更点が早く発見できた
- ハード開発者との意思疎通ができた！
  - 細かい点まで仕様の合意が取れた

■ 悪かった点

- 保守性が悪い部分ができてしまった
  - 一番最初の機能の期間が短く、納期優先の設計になってしまい変更がしづらくなってしまった
- マネージメント力不足
  - 要求は永遠と出るので更新し続ける計画が必要

まとめ

- アジャイルにはイテレーションの単位をどうするかという技術要素がある
  - USDMを利用することで優先の議論や1イテレーションにする単位が決めやすいため効果的と感じた
- USDMも徐々に拡張していくという方法もよいと感じた
  - しかし大きくなりすぎたUSDMは見づらさを生んだ
  - USDMの構造化など考える必要がある

- XDDPには他のツールもあり、自分たちの問題が、当てはまるようであれば一部だけでも活用していけるということが分かった

最後に

「アジャイルを活用した開発改善」

2017/11/15 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/

本書の著作権は一般社団法人組込みシステム技術協会(以下、JASA) が有します。
JASAの許可無く、本書の複製、再配布、譲渡、展示はできません。
また本書の改変、翻案、翻訳の権利はJASAが占有します。
その他、JASAが定めた著作権規程に準じます。

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

© Japan Embedded Systems Technology Association 20YY