モデルベース開発における アジャイルＸＤＤＰの適用

スクラムＸの提案と実践

2018年 5月 18日

Ｔ06研究会 アジャイルとＸＤＤＰの融合

アジェンダ

- スクラムＸ 2.0の紹介
- スクラムＸ 2.0の実践例

第1部

スクラムX 2.0の紹介

第1部 スクラムX 2.0の紹介

- T06研究会の紹介
- スクラムＸ 1.0とは
- スクラムＸ 1.0の課題と対策
  - アジャイルUSDM
  - モデルで変更の影響を分析

第1部 スクラムX 2.0の紹介

- T06研究会の紹介
- スクラムＸ 1.0とは
- スクラムＸ 1.0の課題と対策
  - アジャイルUSDM
  - モデルで変更の影響を分析

T06研究会の概要

- **研究テーマ**

  XDDPとAgileの融合。2014年に派生開発推進協議会の中でAgileに興味を持つメンバーが集まり立ち上げ。以下精力的に活動してきた。
  2015年：２つのAgileXDDPの提案
  2016年：XDDPを活用してアジャイル派生開発に挑む(ET2016)
  2017年：スクラムＸ(Ver 1.0)の提案

- **概要**

  以下の3つのテーマを研究し、**モデルベース開発をXDDPで派生開発をアジャイルに開発する**ためのプラクティスを定義する。
  ① XDDPにAgileプラクティスを取り入れることでXDDPを改善
  ② AgileにXDDPの手法を取り入れることでAgileを改善
  ③ **モデルベース開発 + XDDP + Agile**

- **参加メンバー**

  斎藤　賢一、佐々木　俊正、小田 武弥、葛西 孝弘、加藤 康之、永田 敦、星野 充史、
  本田 英稔、八木 将計、山崎 伸洋

2015年：2つのAgileXDDPの提案

変更のAgileXDDP

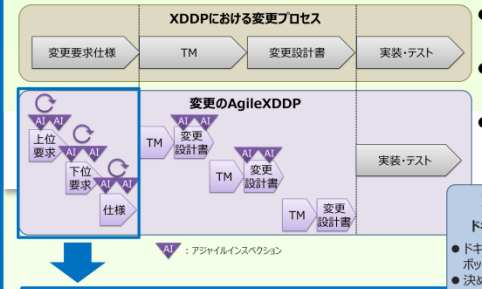
問題の深堀：

- 詳細な仕様に目が向き、要求の定義、確認がおろそかになる
- 思い込んだまま進んでしまい、仕様モレやミスに気付かない
- 最後まで書ききると、手戻りによる書き直しを防ぎたくなる
- USDMを正確かつ網羅的に記述するスキルが必要
- USDMとして記載する要求の粒度が難しい

解決すべき課題

変更3点セットの効率的なレビューの機会と記述スキルの向上

変更のAgileXDDP

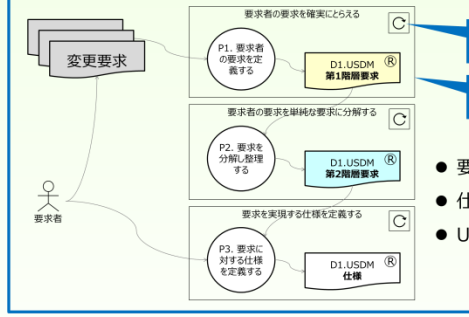


- 変更要求仕様・TM・変更設計書をイテレーティブに行う。
- 変更要求を確実に抑えてから下位要求・仕様を固める。
- アジャイルインスペクションにより欠陥の早期発見と記述スキルを向上

アジャイルインスペクションとは

ドキュメントと書き手の品質レベルを上げる

- ドキュメント作成途中にサンプリングし短いタイムボックスでレビューする
- 決めた品質になるまで修正／サンプリング／レビューを繰り返す



- 要求の勘違いを早期に防止
- 仕様ミス・勘違いゼロ
- USDM記述のスキル向上

機能追加のAgileXDDP

問題の深堀：

- 機能追加時に既存機能への影響の確認が十分に行われていない
- デグレードによる手戻りが大きく、なかなか品質が安定しない
- 単体テストで安心しても、統合テストでバグがバラバラ出る

解決すべき課題：

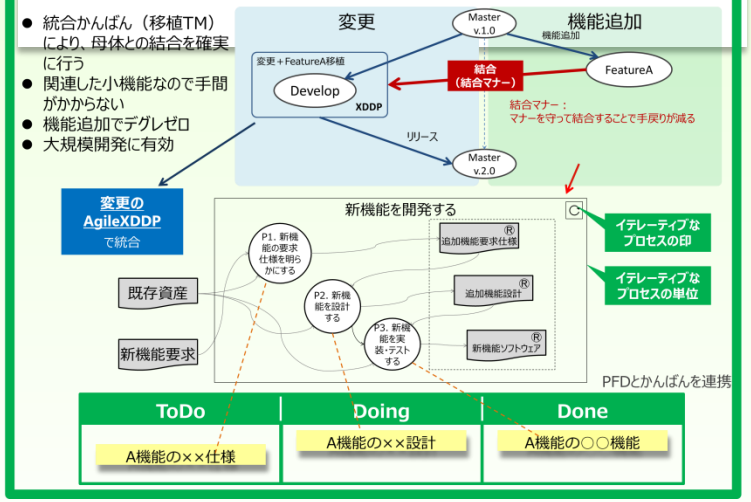
Agileのスピード感を保ちつつ、デグレゼロで機能を追加する

機能追加のAgileXDDP

- 機能ごとに開発を行い、その後XDDPのプロセスを用いて移植（結合）を行う

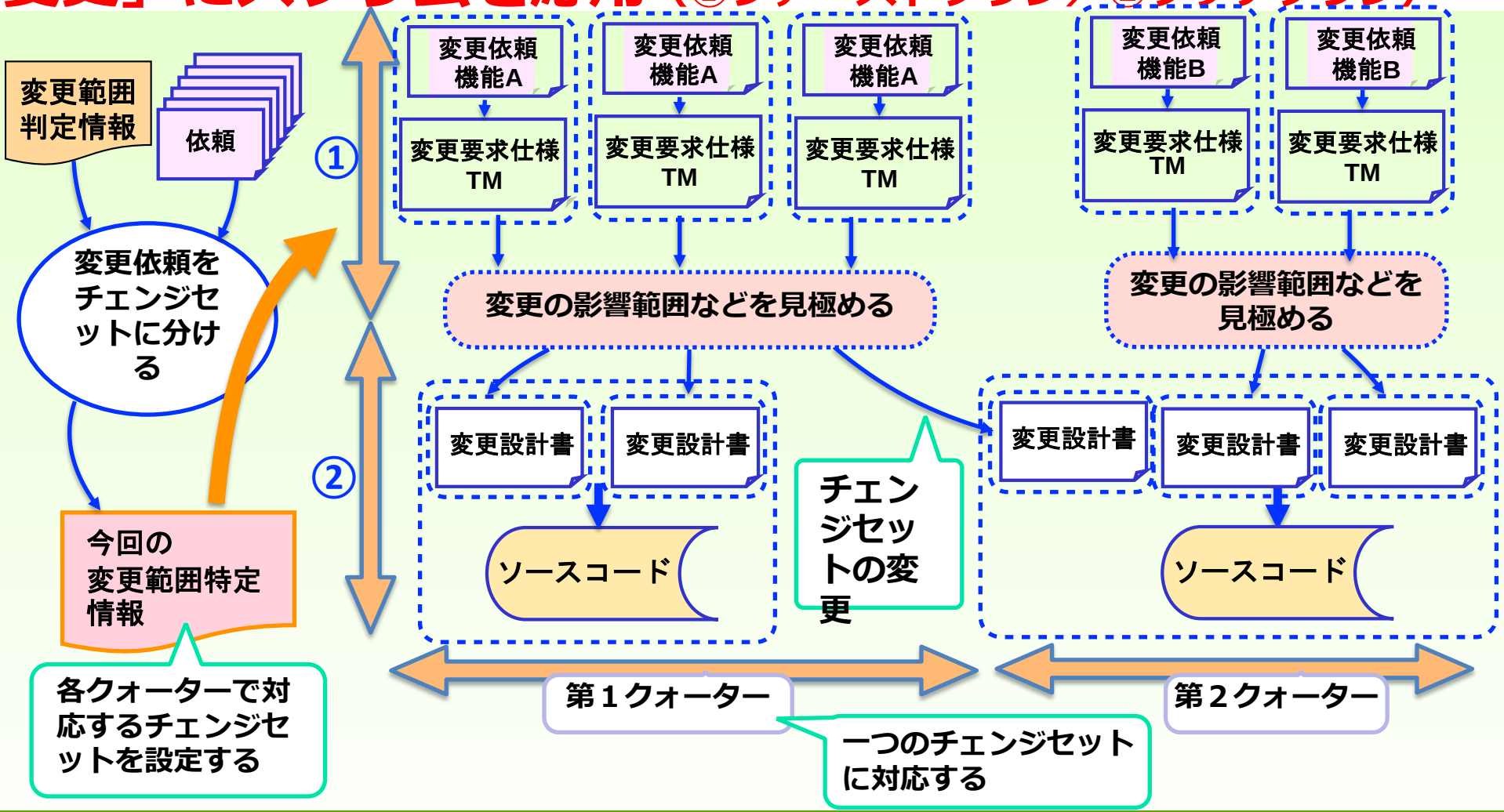
AgileXDDPのブランチ管理・開発のイメージ

- 統合かんばん（移植TM）により、母体との結合を確実に行う
- 関連した小機能なので手間がかからない
- 機能追加でデグレゼロ
- 大規模開発に有効



2016年：XDDPを活用してアジャイル派生開発に挑む(ET2016)

- **「変更」にスクラムを応用（①ファーストダウン／②タッチダウン）**



2017年：スクラムX 1.0 の提案

派生開発推進協議会　T6研究会

スクラムX

Scrumの特徴・課題

■特徴
- アジャイル開発は動くソフトウエアにインクリメンタルに追加および変更開発を繰り返していく(差分開発)
- 仕様の表現や詰め方、設計のやり方についてはScrumでは規定していない

■課題
- 開発チームの行うプロセスが明確でないため、調べた内容を残す方法がなく、最適な仕様と設計の吟味ができなくなり、修正前の影響範囲などのレビューが難しく、手戻りが発生する

XDDPの特徴・課題

■特徴
- 変更は既存資産に対する「差分」で進め、既存資産の不明な部分はスペックアウトで明らかにする
- 変更3点セット「変更要求仕様書」「TM(Traceability Matrix)」「変更設計書」により、具体的な変更方法を早期に確認する
- 変更方法確認後、一気にソースコードを変更することで結合後の混乱がない

■課題
- XDDPを適用すべき状況において、プロダクト要求による段階的なリリースや、要求の優先順位の変更が求められるような開発でその都度現場で考える必要がある

ScrumとXDDPの問題を解決する「スクラムX」

- Scrumはビジネス重心であり、既存システム(ソースコード)の理解が難しい場合はうまく進まない
- 派生開発の要求にも優先順位があり、優先順位が高いものは早くリリースしたい
- ScrumとXDDPの融合により、上記課題を解決
  ⇒ スクラムXの提案 (X：eXtreme, eXpend, eXtension, Xddp)
- **スクラムX**の特徴
  - 最初に変更に対する既存システムの不明点を明確にし、プロジェクトの要求の優先順位を、ビジネスの側面だけでなく、システムへの影響を加味して進める
  - 役割：Scrumのプロダクトオーナー、スクラムマスターに**「スカウター」**を追加
  - プロセス：ファーストダウン・タッチダウンをクォーター単位で実施
    - **ファーストダウン**：変更の影響範囲を見極める
    - **タッチダウン**：ファーストダウンの範囲で変更設計・ソースコードを変更
    - **クォーター**：ファーストダウン・タッチダウンを1つの単位としたもの。リリース、デプロイの単位

スクラムXの効果

- 実施するプロセスが明確になり、レビューが行えるようになり、抜け漏れなどがなくなる
- XDDPの課題であった計画的な段階リリースが可能になる

「アジャイル開発との連携」

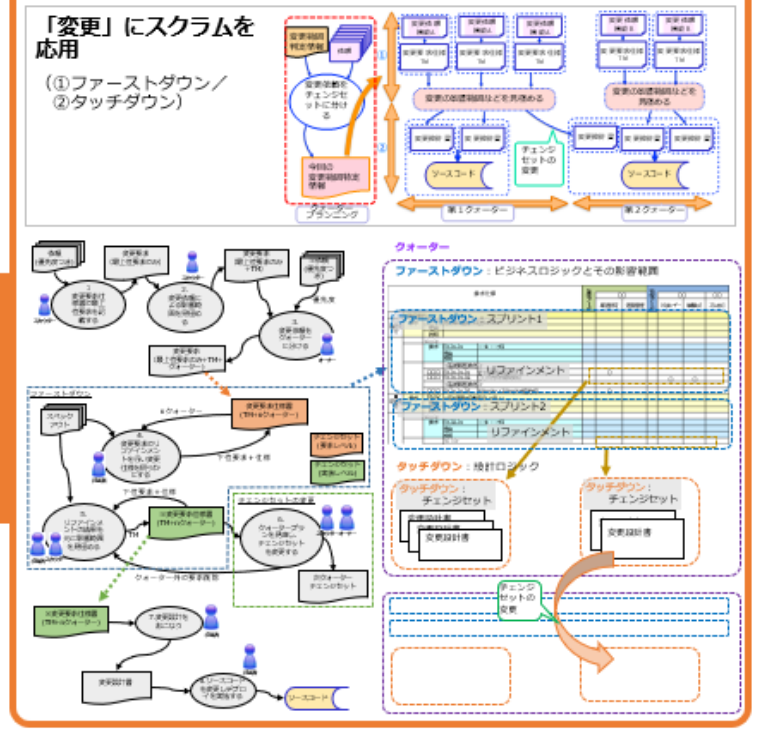
スクラムXの役割

プロダクトオーナー	■プロダクトについて**何をどの順番で**作るのかについて責任を持つ
スクラムマスター	■プロジェクトへのスクラムの導入、**エンジニアリングプロセスの観点**で開発チームをマネジメントする
スカウター	■システムの構造を理解し、**変更要求による影響範囲**を見極める（既存設計を偵察、すなわち、スカウティングする役割から、スカウターと呼ぶ） ■内部の作りによって要求の**優先順位の見直し**をプロダクトオーナーに**提案する**（戦術参謀）
開発チーム	■開発チームはプロダクトバックログアイテムを開発し、**リリースする**ことに責任を持つ ■バックログを**リファインメントする**（USDMで要求を仕様化する）

スクラムXのプロセス

「変更」にスクラムを応用

（①ファーストダウン／②タッチダウン）

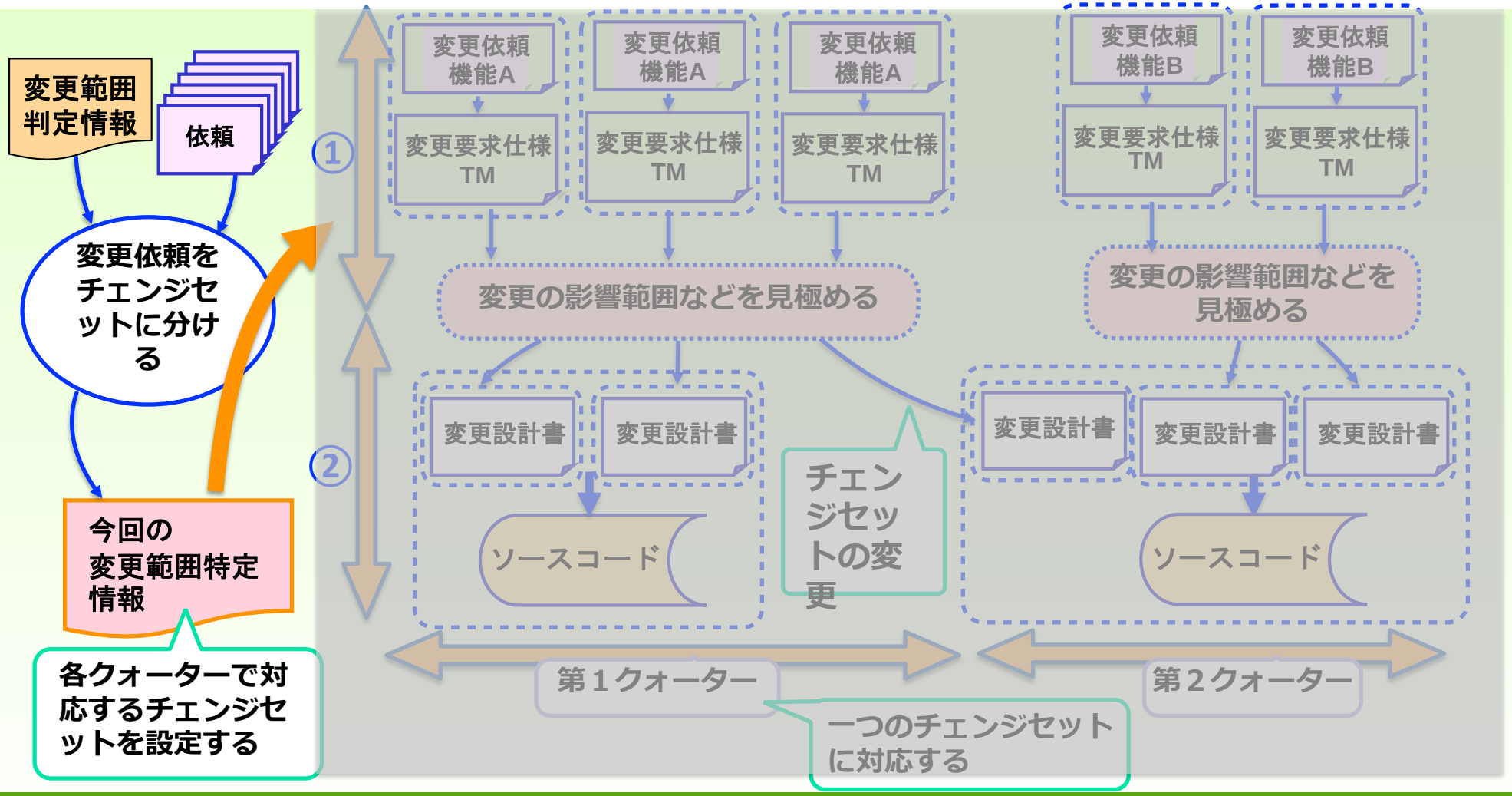


第1部 スクラムX 2.0の紹介

- T06研究会の紹介
- スクラムＸ 1.0とは
- スクラムＸ 1.0の課題と対策
  - アジャイルUSDM
  - モデルで変更の影響を分析

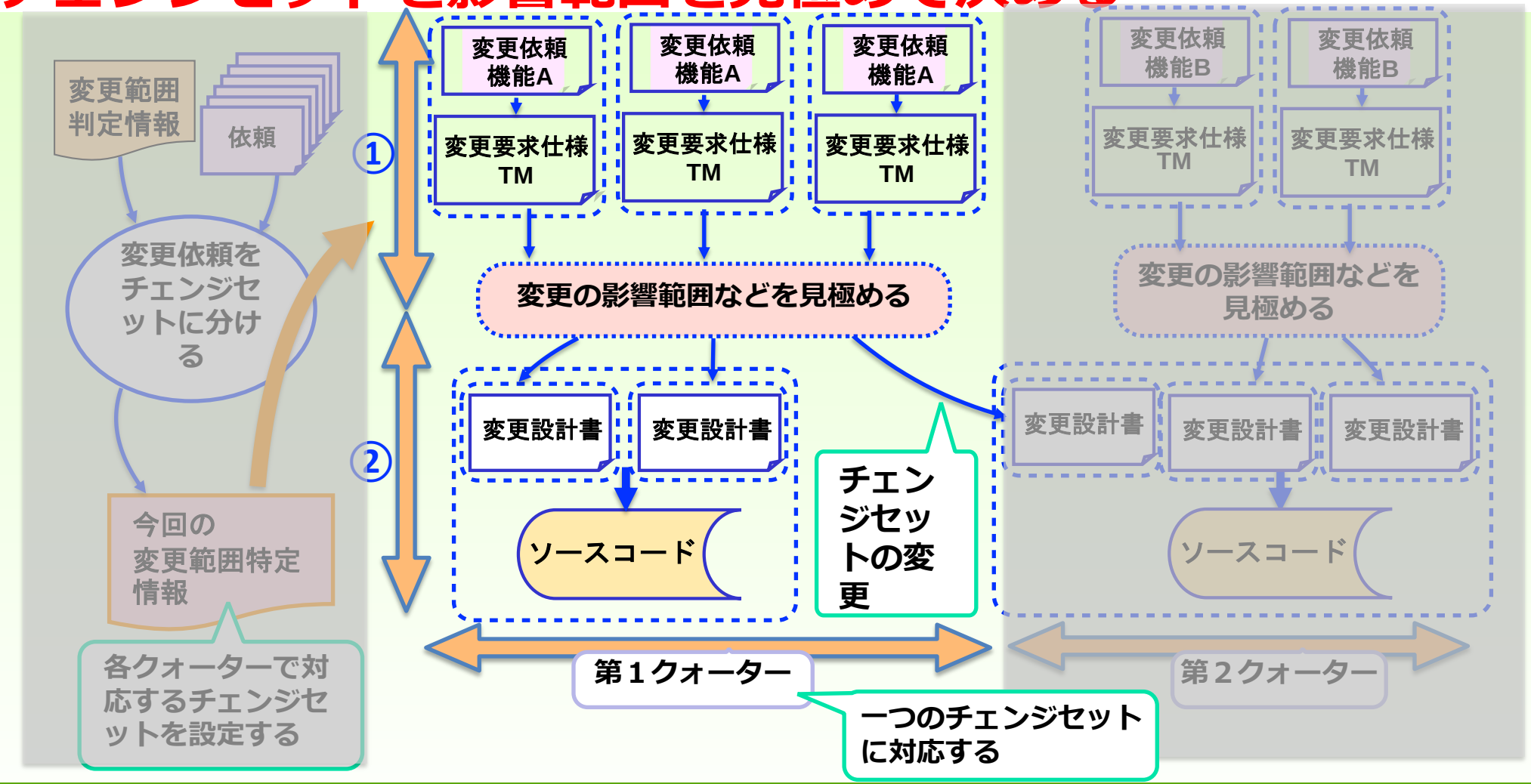
スクラムＸ 1.0とは

- **段階的リリースを可能にするためにチェンジセットの概念を導入**



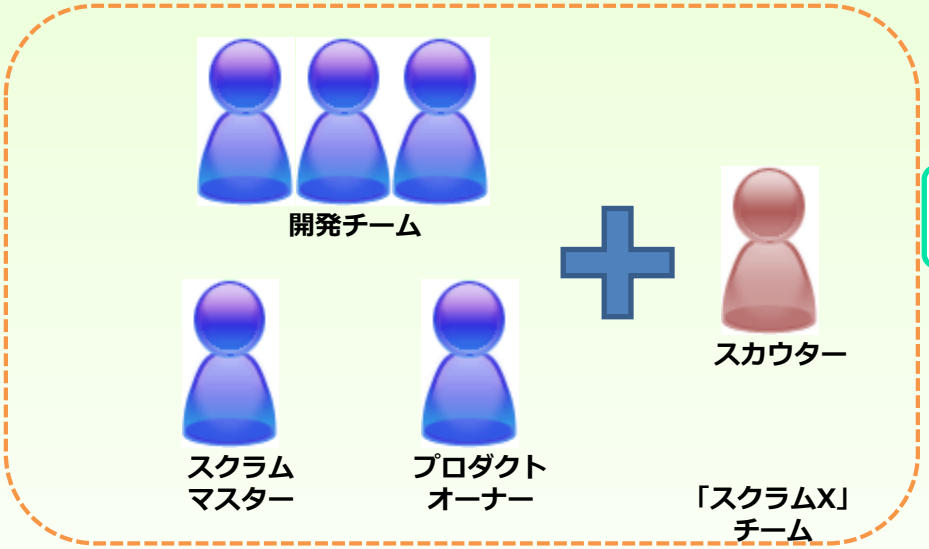
スクラムX 1.0とは

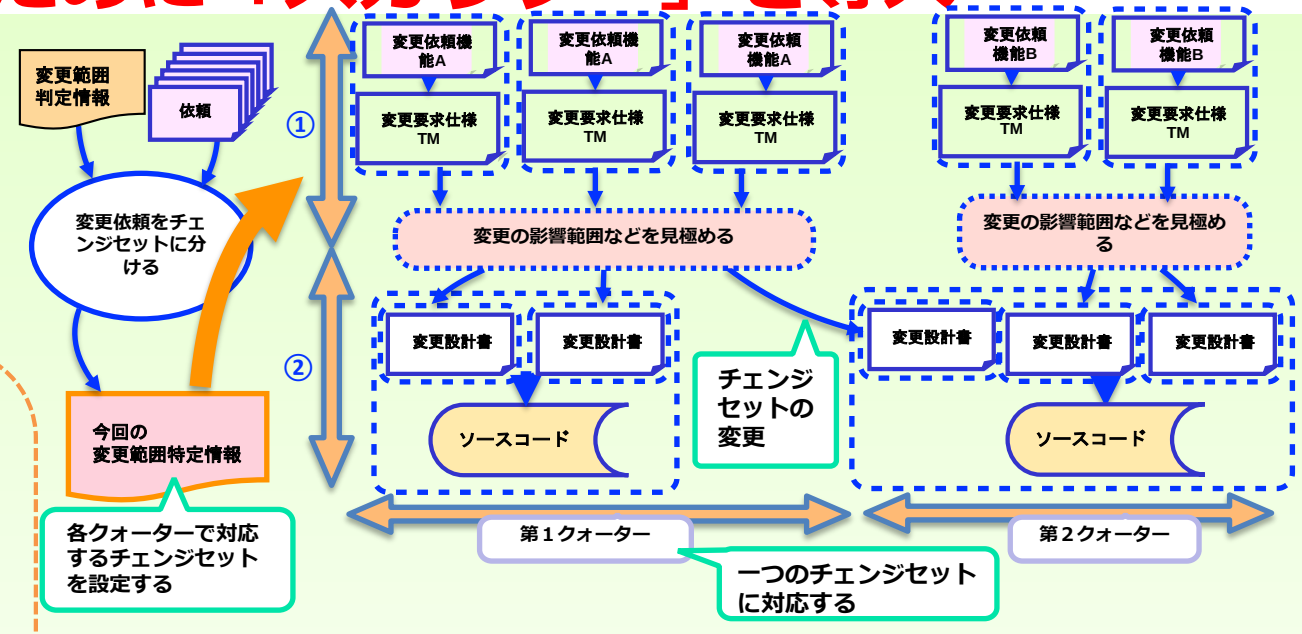
- チェンジセットを影響範囲を見極めて決める



スクラムＸ 1.0とは

●影響範囲を見極めるために「スカウター」を導入





スクラムのロールとしてあらたにスカウターを導入

■システムの構造を事前に調査し、**変更要求による影響範囲**を見極める（既存設計を偵察、すなわち、スカウティングする役割から、スカウターと呼ぶ）

■内部の作りによって要求の**優先順位の見直し**をプロダクトオーナーに**提案する**（戦術参謀）

第1部 スクラムX 2.0の紹介

- T06研究会の紹介
- スクラムＸ 1.0とは
- スクラムＸ 1.0の課題と対策
  - アジャイルUSDM
  - モデルで変更の影響を分析

スクラムＸ 1.0の課題と対策

課題①：実践不足

- スクラムＸ1.0は実践するためにはプロセスが荒い

課題②：ユーザーストーリーの落としこみ

- ユーザーストーリーを精度よく変更要求に落とし込みたい(ユーザーストーリーとＵＳＤＭにギャップがある)

課題③：影響分析の検討もれ

- ソースコードのグレップだけでは変更箇所の影響範囲が十分に検討しきれない
- スカウターが影響範囲を早く正確に判断するためにはＴＭだけだとたりない(ＴＭでは構造までしかわからない)

対策①：プロセスを再設計した

- プロセスを詳細化し、成果物も定めた

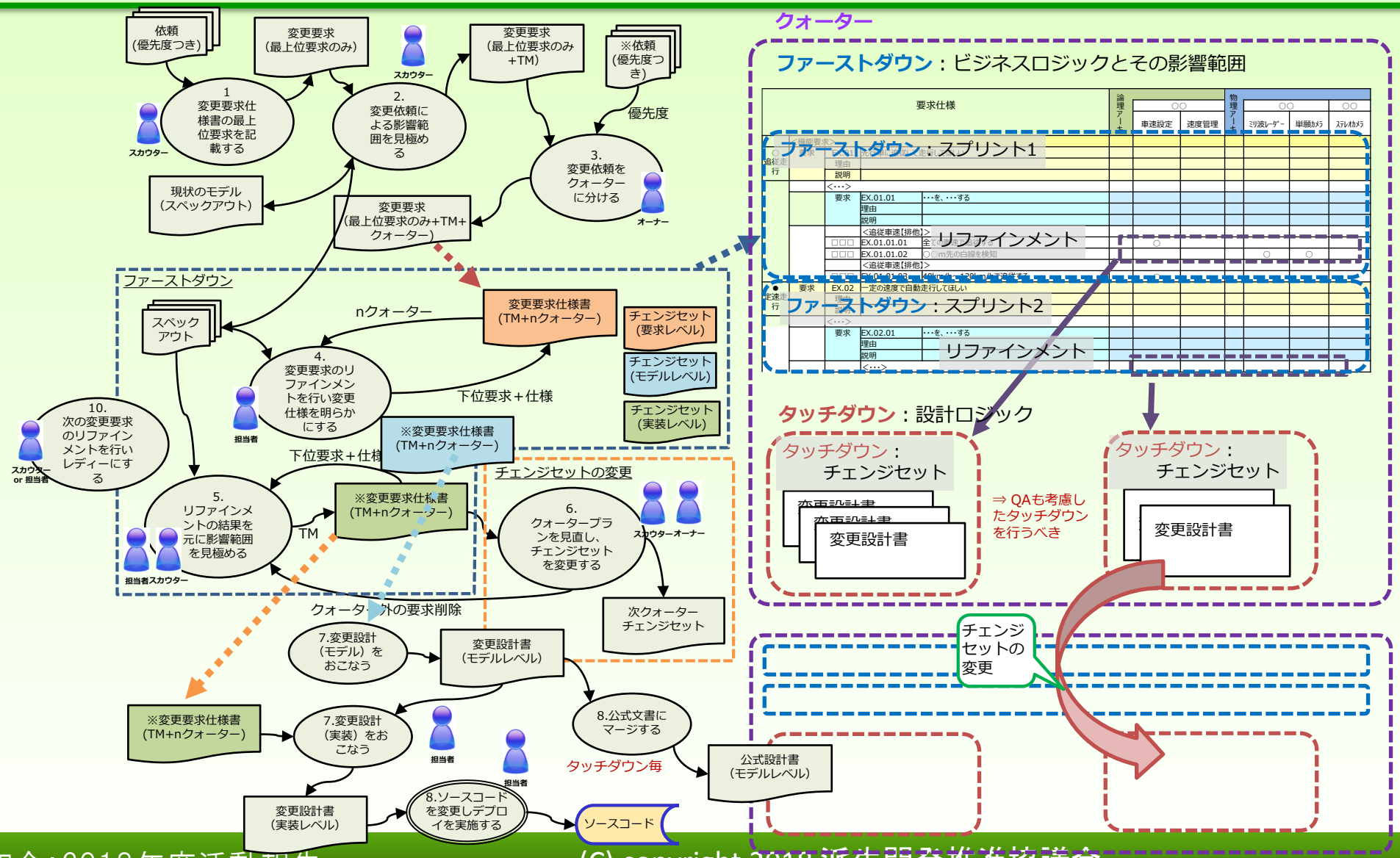
対策②：アジャイルUSDMを考案した

- ユーザーストーリーを変更要求に落とし込むために軸を定めた

対策③：モデルで変更の影響を分析した

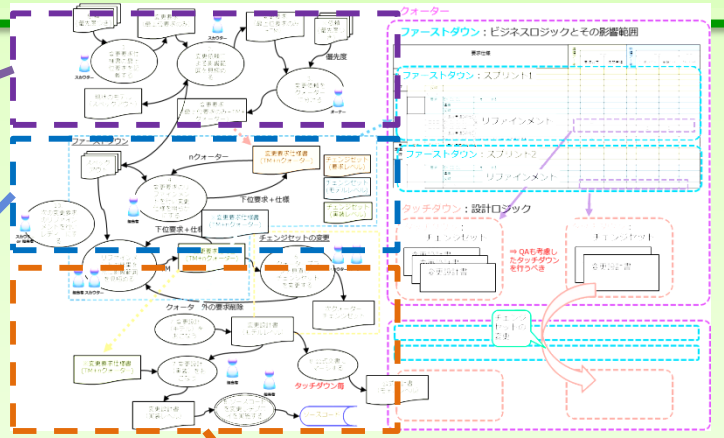
- スペックアウトやリファインメントにモデルを用いることでさまざまな観点で変更の影響を分析することができた
- ＴＭだけではなく、モデルに直接変更箇所を記すことでその箇所を変更することで及ぼす影響が分析できるようになった

対策①：再設計したプロセス

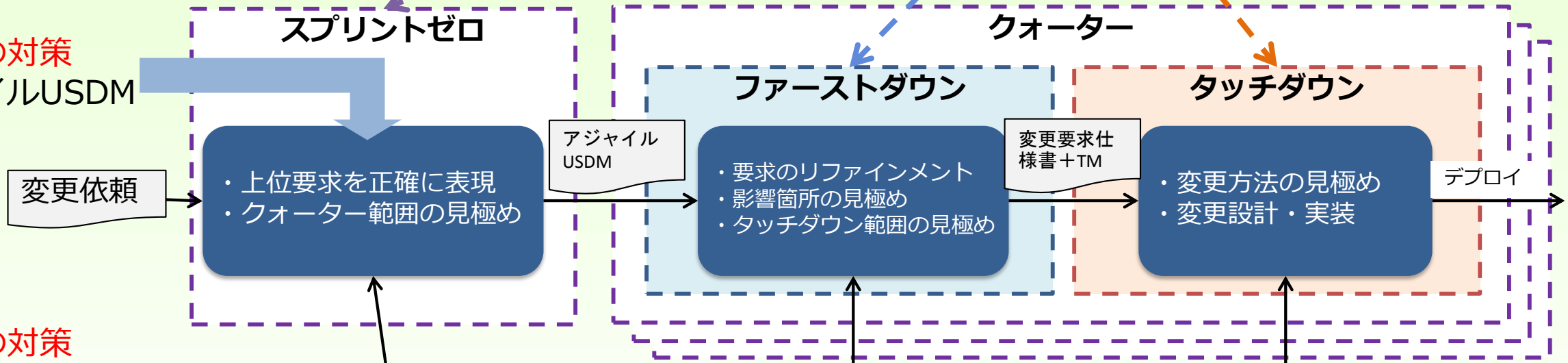


対策①：再設計したプロセス

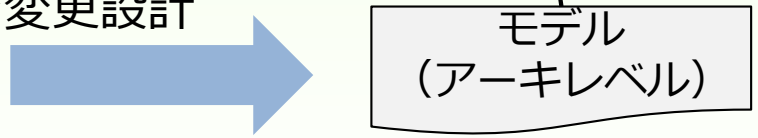
課題①の対策
スクラムのフレームワークに整合するよう、
プロセスを再設計



課題②の対策
アジャイルUSDM



課題③の対策
モデルで変更設計



対策②：アジャイルUSDM

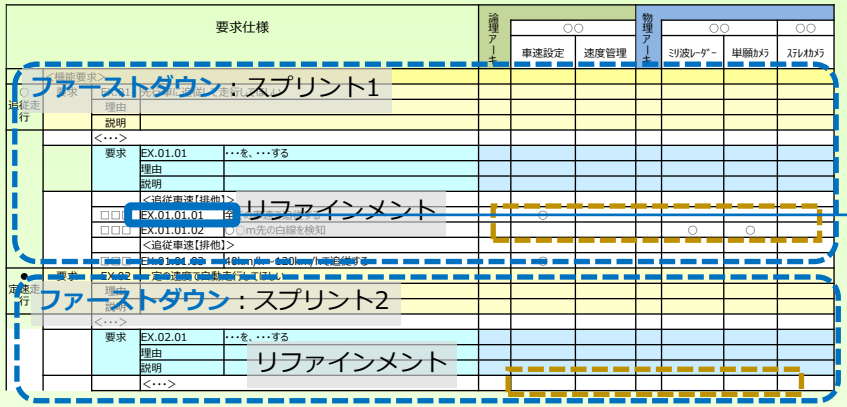
要求仕様				
	<機能要求>			
追従走行	要求	EX.01	先行車に追従して走行してほしい	
		理由	○○だから	
		説明		
		<・・・>		
		要求	EX.01.01	・・・を、・・・する
			理由	
			説明	
			<追従車速【排他】>	
		□□□	EX.01.01.01	全ての車速で追従する
		□□□	EX.01.01.02	○○m先の白線を検知
			<追従車速【排他】>	
		□□□	EX.01.01.03	40km/h～120km/hで追従する
定速走行	要求	EX.02	一定の速度で自動走行してほしい	
		理由		
		説明		
		<・・・>		
		要求	EX.02.01	・・・を、・・・する
			理由	
			説明	
			<・・・>	

ユーザーストーリーから変更要求を導き出すためのフレームを考案

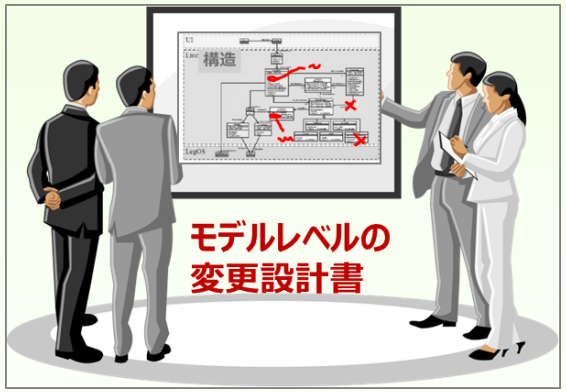
つなげて読むと上位要求、
セル1つ1つは下位要求

ラベル（USDMの上位要求のラベル）		（だれが）どんな状況で	どんなとき	どうなったら	どうなる	理由
衝突回避ブレーキ(新規)	要求	走行中	前走車がいるとき	ぶつかりそうになったら	ブレーキをかける	
	仕様	走行中の仕様 30km/h～130km/hを走行中とする ・・・	前走車の認識仕様	TTCの仕様	ブレーキをかける仕様	
衝突回避ブレーキ(変更)	Before要求	走行中	前走車がいるとき	ぶつかりそうになったら	ブレーキをかける	
	After要求	走行中	前走車がいるとき	ぶつかる手前で	警報を出す	
	仕様					

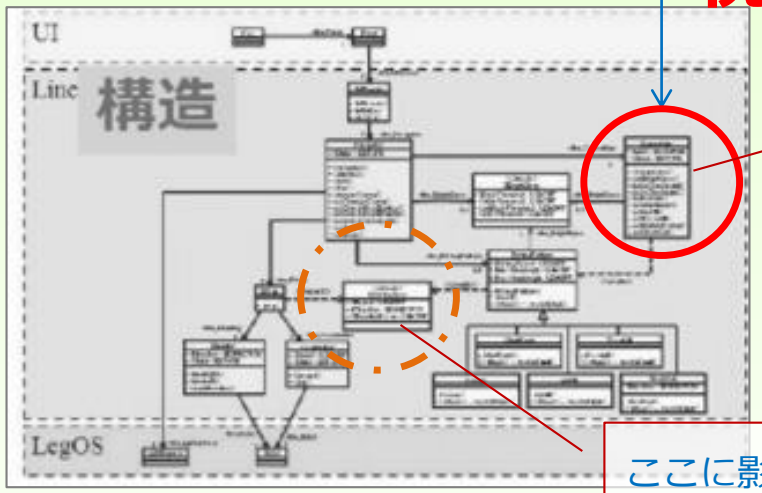
対策③：モデルで変更の影響を分析



どう変更しようとしているかをメモで残す



影響範囲を視覚的に確認



ここをかえる

ここに影響あり

第1部 スクラムX 2.0の紹介

- T06研究会の紹介
- スクラムＸ 1.0とは
- スクラムＸ 1.0の課題と対策
  - アジャイルUSDM
  - モデルで変更の影響を確認

対策②詳細：アジャイルＵＳＤＭとは

アジャイルの要件定義

◆ユーザーストーリーは「カード」「会話」「確認」が基本、最小限の記述
◆カスタマージャーニーマップは「サービス全体のデザインを考えるツール」

・要求表現は書き手に依存
・記述内容の粗さ、細かさ
・網羅性を確認しにくい

課題

ユーザーストーリーを精度よく変更要求に落とし込みたい
(ユーザーストーリーとＵＳＤＭにギャップがある)

しかし、USDMは書き方の難易度が高くヘビーなイメージがありユーザーストーリーのようなスピード感が足りない

アジャイルUSDMの提案

・最小限の記述でUSDMの特徴を表現できる
・振る舞いの観点により、要求の書き方に悩まずに記述できる
・要求から仕様を直感的に引き出しやすい

ラベル	要求仕様	(誰が)どんな状況で	(誰が)どんなとき	(誰が)どうなったら	(誰が)どうなる	理由	説明
衝突回避ブレーキ	要求	走行中	前走車がいるとき	ぶつかりそうになったら	ブレーキをかける		
	仕様	走行中の仕様	前走車の認識仕様	TTCの仕様	ブレーキをかける仕様		
追従走行	Before要求	走行中	前走車がいるとき	ぶつかりそうになったら	ブレーキをかける		
	After要求	走行中	前走車がいるとき	ぶつかる手前で	警報を出す		
	仕様						

対策②詳細：アジャイルＵＳＤＭとは

ユーザーストーリー：
荷物の集配依頼を受けたら情報を送る

これを早く正確に変更要求に落とし込みたい

手順1
ユーザーストーリーを黄色セルに転記する

ラベル	要求仕様	(だれが) どんな状況で	(だれが) どんなとき	(だれが) どうなったら	(だれに) どうなる	理由	説明
		荷物の集配依頼を受けたら情報を送る					
	要求						
	仕様						

手順1

対策②詳細：アジャイルＵＳＤＭとは

手順2
ラベルをつける

ラベル	要求仕様	(だれが) どんな状況で	(だれが) どんなとき	(だれが) どうなったら	(だれに) どうなる	理由	説明
		荷物の集配依頼を受けたら情報を送る					
集荷依頼	要求						

手順2

手順3
「振る舞いの観点*」のテンプレートに沿って要求を記述する

*USDMの特徴と、過去のUSDM記述から、記述として抑えるべきポイントを導出

ラベル	要求仕様	(だれが) どんな状況で	(だれが) どんなとき	(だれが) どうなったら	(だれに) どうなる	理由	説明
		荷物の集配依頼を受けたら情報を送る					
集荷依頼	要求	オペレータが集荷受付中					

手順3

振舞いの観点

対策②詳細：アジャイルＵＳＤＭとは

ラベル	要求仕様	(だれが) どんな状況で	(だれが) どんなとき	(だれが) どうなったら	(だれに) どうなる	理由	説明
		荷物の集配依頼を受けたら情報を送る					
集荷依頼	要求	オペレータが集荷受付中	配達員が荷物を配達しているとき	オペレータが荷物の集配依頼を受けたら	担当地域の配達員の端末に住所や氏名等の情報を送る		

残りの欄も埋める

対策②詳細：アジャイルＵＳＤＭとは

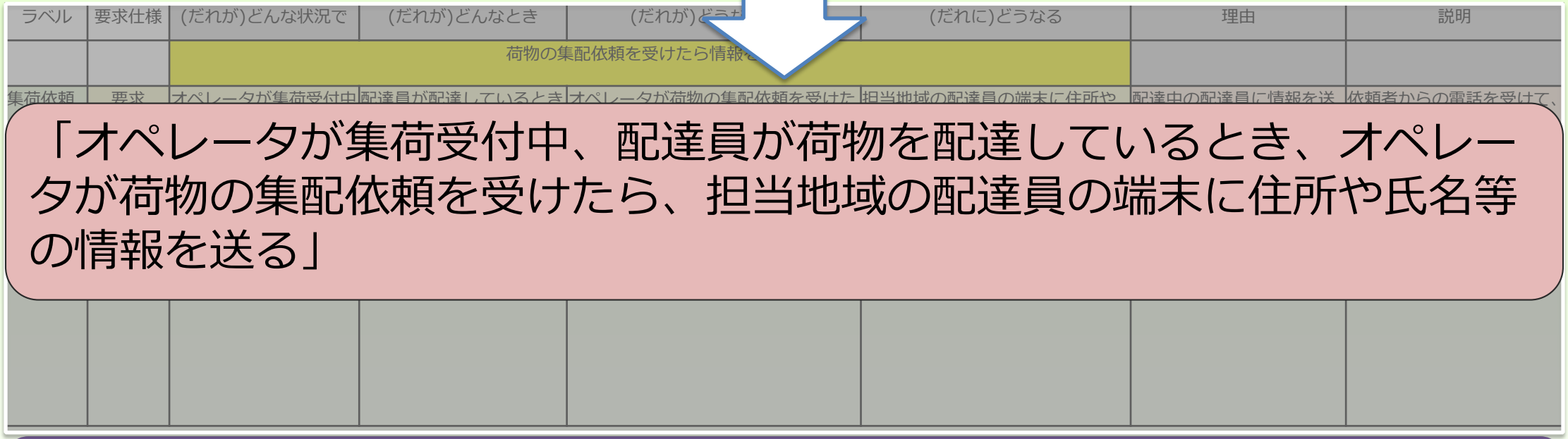
手順4
要求の「理由」と「補足説明」を記入する

ラベル	要求仕様	(だれが)どんな状況で	(だれが)どんなとき	(だれが)どうなったら	(だれに)どうなる	理由	説明
		荷物の集配依頼を受けたら情報を送る					
集荷依頼	要求	オペレータが集荷受付中	配達員が荷物を配達しているとき	オペレータが荷物の集配依頼を受けたら	担当地域の配達員の端末に住所や氏名等の情報を送る	配達中の配達員に情報を送ることで、配達ルートの中で荷物を集めたい	依頼者からの電話を受けて、集荷先の氏名や住所、メールアドレスなどを聞き出して入力したデータが画面上に表示されている

手順４

対策②詳細：アジャイルＵＳＤＭとは

荷物の集配依頼を受けたら情報を送る



ラベル	要求仕様	(だれが)どんな状況で	(だれが)どんなとき	(だれが)どうな	(だれに)どうなる	理由	説明
		荷物の集配依頼を受けたら情報を					
集荷依頼	要求	オペレータが集荷受付中	配達員が配達しているとき	オペレータが荷物の集配依頼を受けた	担当地域の配達員の端末に住所や	配達中の配達員に情報を送	依頼者からの電話を受けて、

「オペレータが集荷受付中、配達員が荷物を配達しているとき、オペレータが荷物の集配依頼を受けたら、担当地域の配達員の端末に住所や氏名等の情報を送る」

振舞いの観点をユーザーストーリーに当てはめることでユーザーストーリーが洗練されました。

対策②詳細：アジャイルＵＳＤＭとは

手順5
この段階ですでに仕様がわかっている場合には仕様を記載する

ラベル	要求仕様	(だれが)どんな状況で	(だれが)どんなとき	(だれが)どうなったら	(だれに)どうなる	理由	説明
		荷物の集配依頼を受けたら情報を送る					
集荷依頼	要求	オペレータが集荷受付中	配達員が配達しているとき	オペレータが荷物の集配依頼を受けたら	担当地域の配達員の端末に住所や氏名等の情報を送る	配達中の配達員に情報を送ることで、配達ルートの中で荷物を集めたい	依頼者からの電話を受けて、集荷先の氏名や住所、メールアドレスなどを聞き出して入力したデータが画面上に表示している
	仕様			<集荷情報入力画面の表示> <担当配達員の検索> <配達員情報の表示>	<集荷指示データの編集> <集荷担当者への送信>		

手順5

対策②詳細：アジャイルＵＳＤＭまとめ

特徴まとめ

- 「振る舞いの観点」のテンプレートに沿って要求を記述
- 観点に沿った要求記述全体を「上位要求」、観点の1つ1つを「下位要求」と捉え、要求の階層を表現
- １つ１つの下位要求に対して仕様を記述できる

ラベル　振る舞いの観点　上位要求　⇔ **下位要求を1文にすると、上位要求になる**

ラベル	要求仕様	(だれが)どんな状況で	(だれが)どんなとき	(だれが)どうなったら	(だれに)どうなる	理由	説明
		荷物の集配依頼を受けて、担当地域の配達員の端末に住所や氏名等の情報を送る					
集荷依頼	要求	オペレータが集荷受付中 下位要求	配達員が配達しているとき 下位要求	オペレータが荷物の集配依頼を受けたら 下位要求	担当地域の配達員の端末に住所や氏名等の情報を送る 下位要求	配達中の配達員に情報を送ることで、配達ルートの中で荷物を集めたい	依頼者からの電話を受けて、集荷先の氏名や住所、メールアドレスなどを聞き出して入力したデータが画面上に表示している
	仕様			<集荷情報入力画面の表示> <担当配達員の検索> <配達員情報の表示> 仕様	<集荷指示データの編集> <集荷担当者への送信> 仕様		
		荷物に貼られた配送シートの情報を読み取って、配送データを生成して配送DBに新規登録する					
荷物受取	要求	配達員が配達中	集荷指示を受け取った後	配達員が荷物を受け取ったら	荷物に貼られた配送シートの情報を読み取って、配送データを生成して配送DBに新規登録する	一旦受け取った荷物は、できるだけ早くシステムに登録しておきたい	
	仕様						

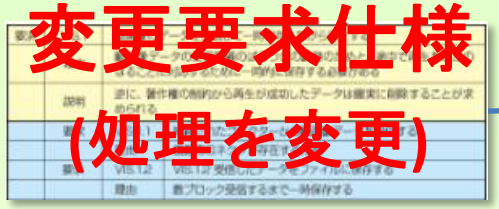
各下位要求に対し、仕様を記述できる

第1部 スクラムX 2.0の紹介

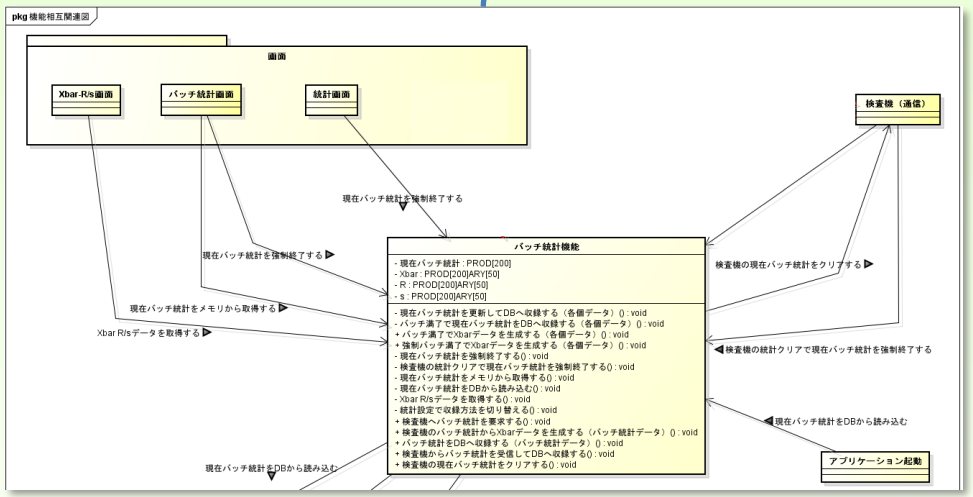
- T06研究会の紹介
- スクラムＸ 1.0とは
- スクラムＸ 1.0の課題と対策
  - アジャイルUSDM
  - モデルで変更の影響を分析

対策③詳細：モデルで変更の影響を分析

変更要求仕様
(処理を変更)

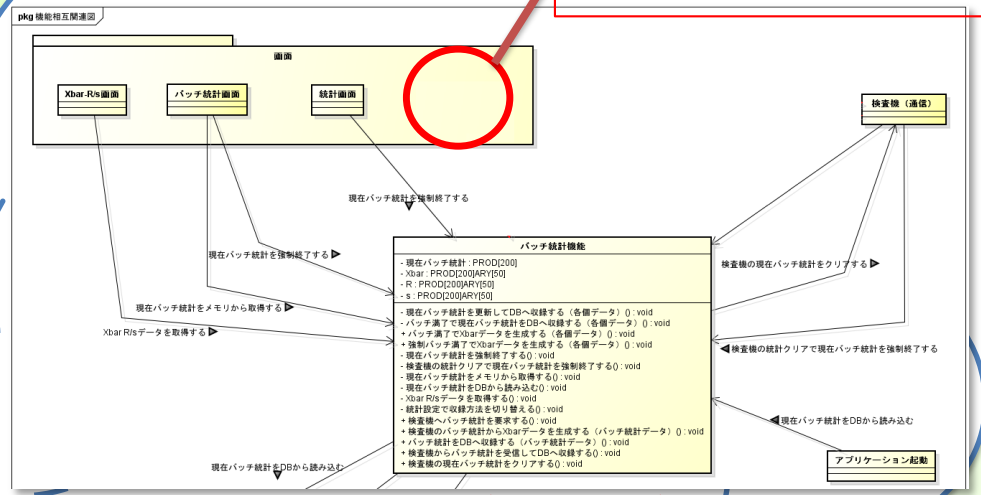


5-1.
変更の影響範囲を見極める



BEFORE（ベースモデル）

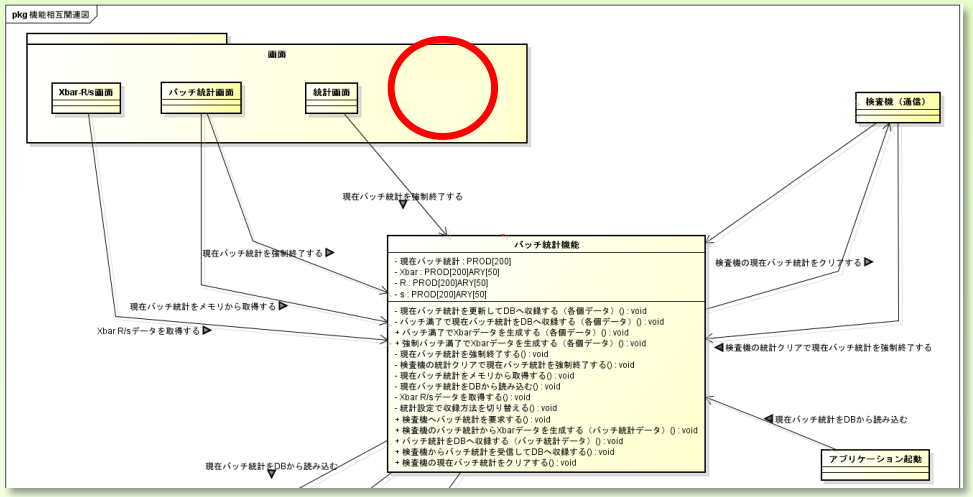
クラスを追加しよう

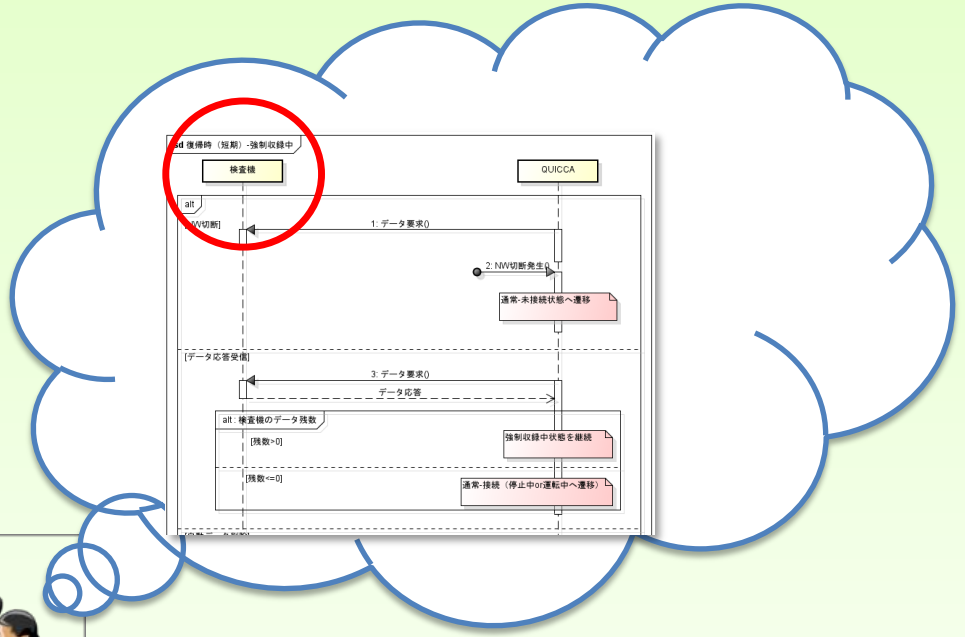


AFTER（メモ）

スカウターが影響分析

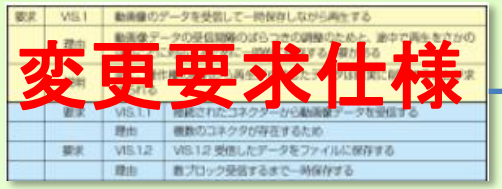
対策③詳細：モデルで変更の影響を分析



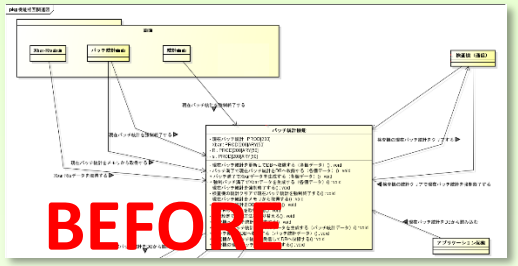


クラス追加するならシーケンスも
変更が必要そうだ

対策③詳細：モデルで変更の影響を分析



変更要求仕様

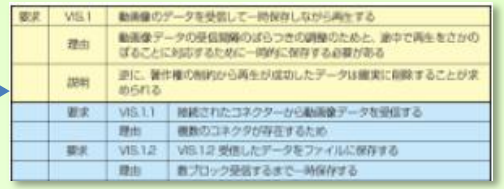


BEFORE

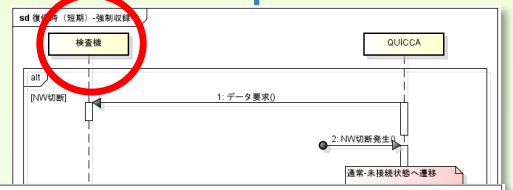
（ベースモデル）

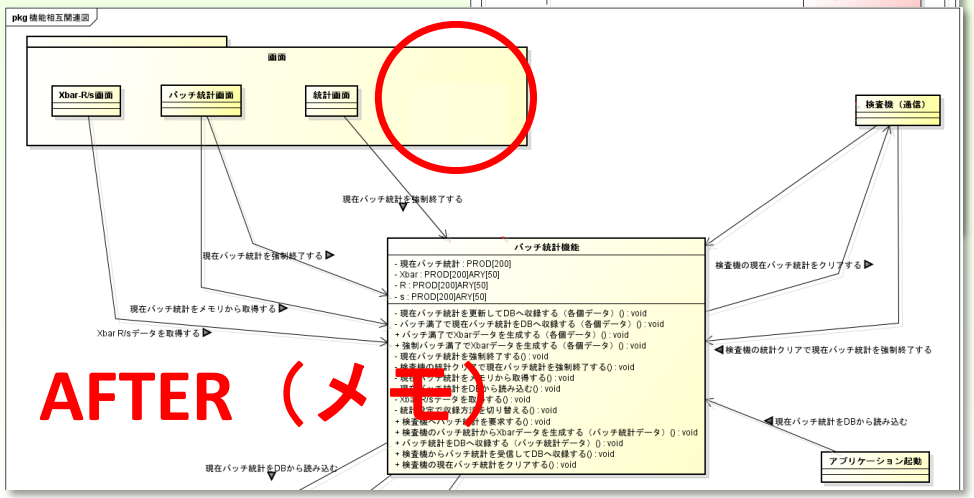
5-2.
レビュー

TM

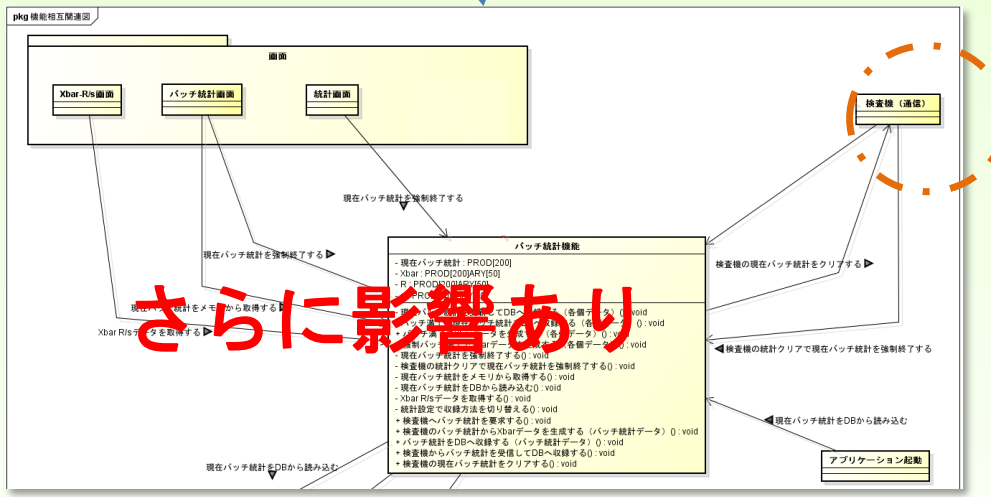


レビュー結果



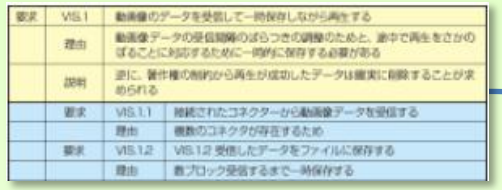


AFTER（メ

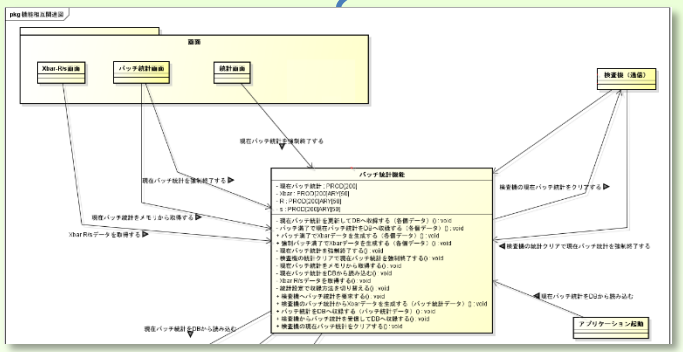


さらに影響あり

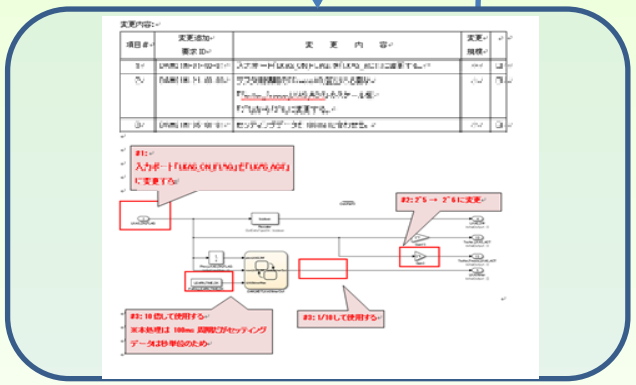
対策③詳細：モデルで変更の影響を分析



7
変更設計をおこなう

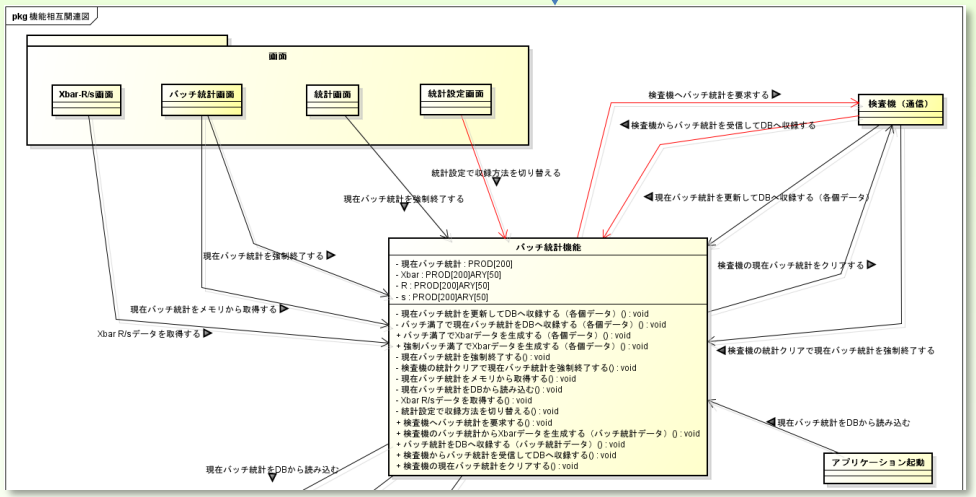


AFTER(メモ)



変更設計書を作成

8
ソースコードを変更しデプロイを実施する



AFTER(モデル正式版)

スクラムX 2.0のまとめ

プロセスを詳細化し、成果物も定めた

- より実用的なプロセスになった

アジャイルUSDMを考案した

- ユーザーストーリからUSDMの表現へ
  - ユーザーストーリとUSDMのギャップを埋めていく
- 早く正確な要求仕様を作ることができる

モデルで変更の影響を分析した

- 要求仕様による変更がシステムのどこにどのように影響するかモデルで俯瞰
- スカウターがクオータに対するバックログのスコープを決めやすくなる

第2部

実践編
スクラムX 2.0の部分適用

第2部 スクラムX 2.0の部分適用

- プロジェクト紹介
- プロジェクトの問題点
- スクラムＸ 2.0で問題解決
  - 問題点①にスクラムＸを適用
  - 問題点②にスクラムＸを適用

第2部 スクラムX 2.0の部分適用

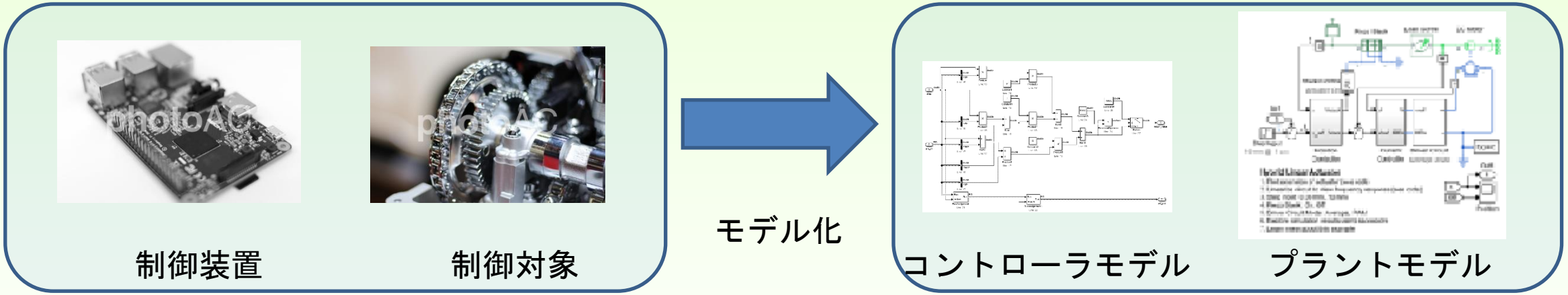
- プロジェクト紹介
- プロジェクトの問題点
- スクラムＸ 2.0で問題解決
  - 問題点①にスクラムＸを適用
  - 問題点②にスクラムＸを適用

プロジェクト紹介

- プロジェクト概要
  - 組込車載製品の開発
- 開発の特徴
  - MATLAB/Simulink ©を用いたモデルベース開発
- 開発体制
  - ベテラン エンジニア1名
  - 中堅 エンジニア2名
  - 若手 エンジニア2名
  - スクラムマスター(PM) 1名

モデルベース開発とは

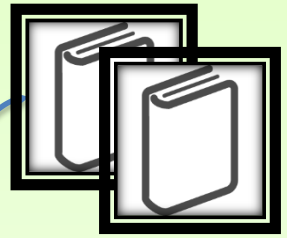
- **モデルベース開発とは**
  - MATLAB/Simulink©等のＣＡＥツールによって制御装置と制御対象をモデル化し、それを実行可能な仕様書として用いることで製品ライフサイクル全般に渡った品質向上と開発効率向上を目指した開発手法のことである



引用：ＪＭＡＡＢ
http://jmaab.mathworks.jp/MBD_definition/index.html

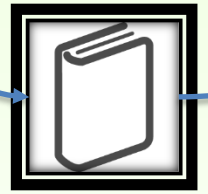
弊社の一般的な作業工程

凡例
お客様作業
弊社作業

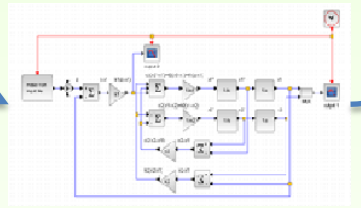


変更/追加要求
既存モデル

要求分析 → 各種設計 → モデリング → モデル検証 → 受け入れ検査

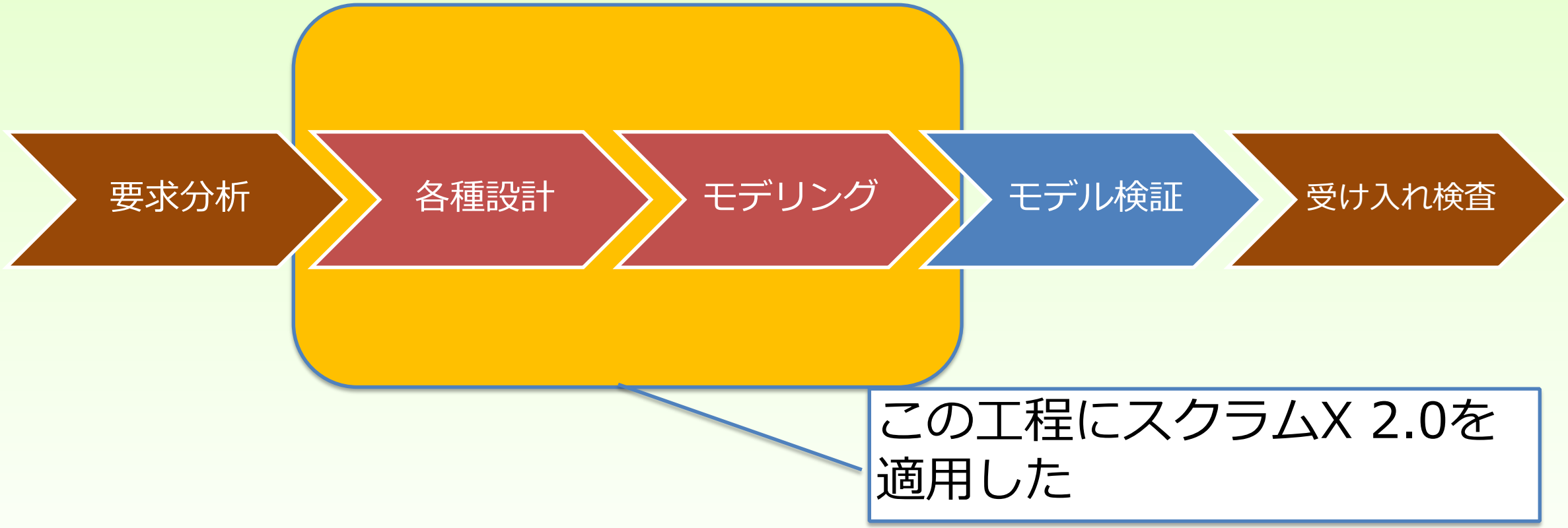


設計仕様書



Simulink©モデル

スクラムX 2.0を適用

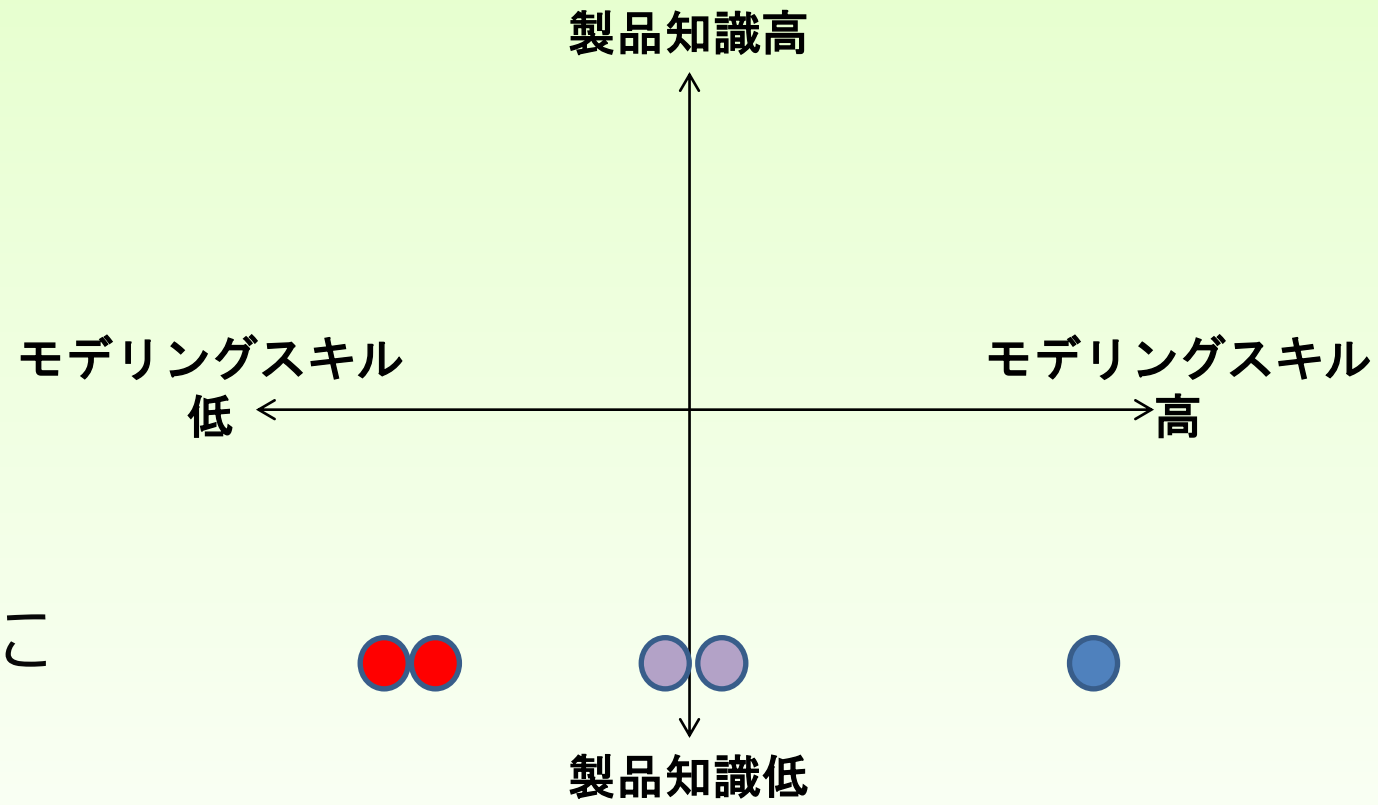


第2部 スクラムX 2.0の部分適用

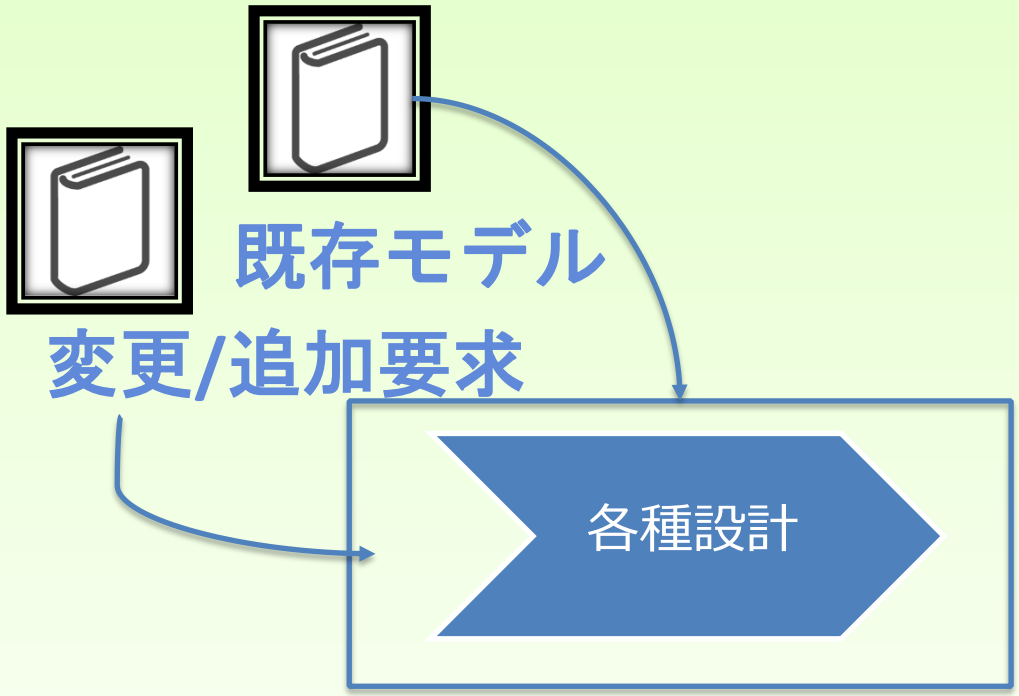
- プロジェクト紹介
- プロジェクトの問題点
- スクラムＸ 2.0で問題解決
  - 問題点①にスクラムＸを適用
  - 問題点②にスクラムＸを適用

本事例における開発体制の問題

- **開発体制**
  - ベテラン1名
  - 中堅2名
  - 若手2名
- **体制の問題点**
  - 製品知識ゼロ
  - モデリングスキルにばらつきあり



製品知識不足によって発生する問題



変更/追加要求が既存機能やモデルのどこにどのように影響するのかがわからない

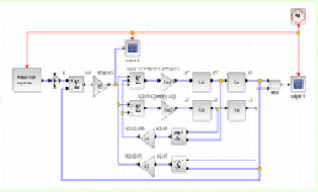
製品知識不足によって設計の品質に問題が発生することが予想された

モデリングスキルのばらつきによる問題

正しくモデルが作られているか不安
開発完了に間に合うか不安

若手開発者

モデリング



Simulink©モデル

開発管理者

モデリングスキルのばらつきによってモデルの品質に問題が発生することが予想された

問題点まとめ

①変更追加要求に対する影響範囲がわからない

- 要求の変更/追加が既存機能やモデルの「どこに」「どのように」影響するのかがわからない

②モデリングスキルが低いメンバーの品質への不安

- 要求を正しくモデルに反映できるか不安（要求とモデルの一致性に不安）
- 作成されたモデルの設計品質に不安

上記問題により手戻りが発生しデリバリが遅れる懸念がある

第2部 スクラムX 2.0の部分適用

- プロジェクト紹介
- プロジェクトの問題点
- スクラムＸ 2.0で問題解決
  - 問題点①にスクラムＸを適用
  - 問題点②にスクラムＸを適用

問題点①に「スクラムＸ 2.0」を適用

問題点	対策		効果
変更に対する影響範囲がわからない	対策	ＸＤＤＰ	• 変更による影響をＴＭで明示できる
	さらに対策	スクラムＸ	• スカウターによって変更のキーポイントを早期に発見できる • スプリントゼロでチェンジセットを検討し、分割して複数回デリバリーできる

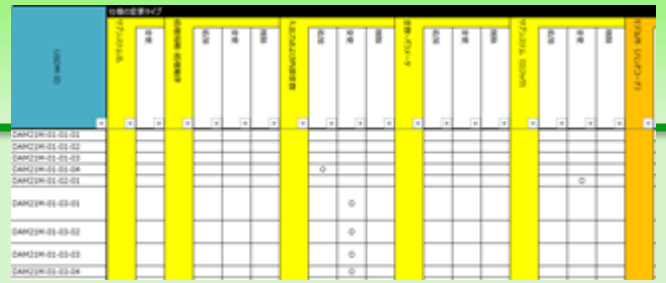
問題①の対策：XDDPの導入

問題

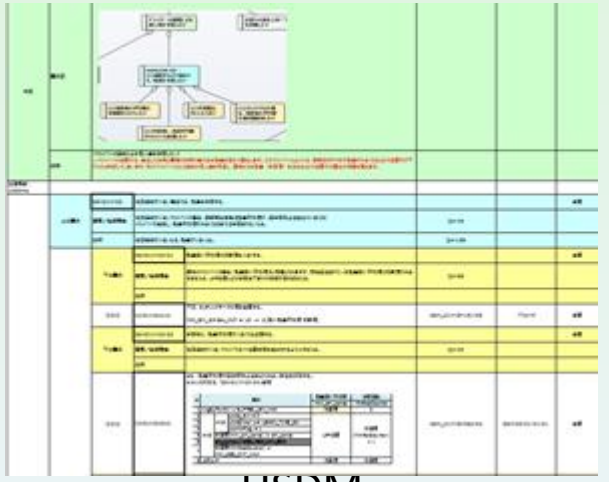
①変更追加要求に対する影響範囲がわからない

対策：XDDPの導入

適切な変更方法を見出すために、
変更追加の対象個所の構造や設計の
意図をスペックアウトし、既存システムの「要求」＋「仕様」を深堀し
USDMへ書き出す



スペックアウト資料（影響分析）



USDM

問題①の対策に対するあらたな課題

- でも・・・
  - 「製品知識ゼロ」「スキルにばらつきあり」で誰が影響分析できるのか？
  - 影響分析が不足することで手戻りが多く発生しデリバリーに影響するのでは？
  - 変更がどのように影響するかがわからないのに、既存の開発方針に則って製品を一括デリバリーして大丈夫？

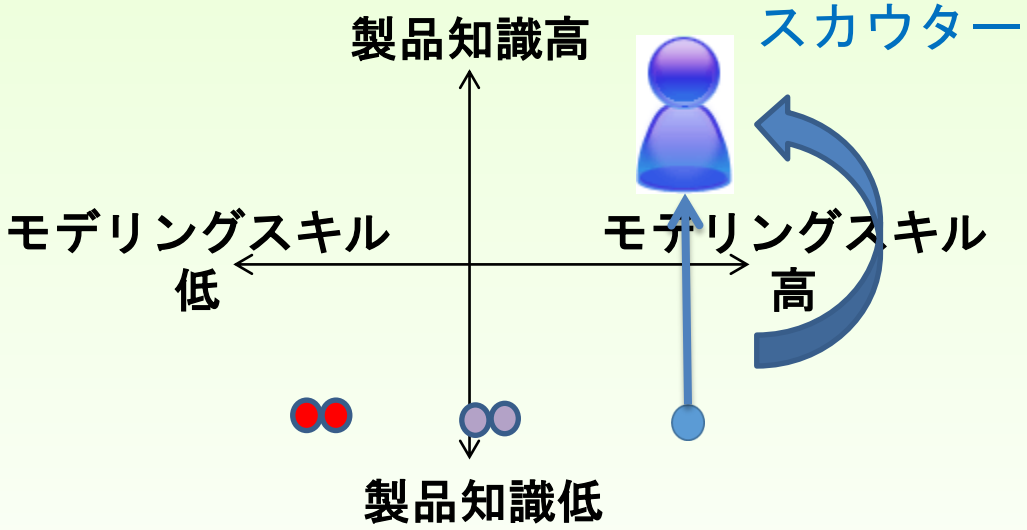
「スクラムＸ」 の主な特徴

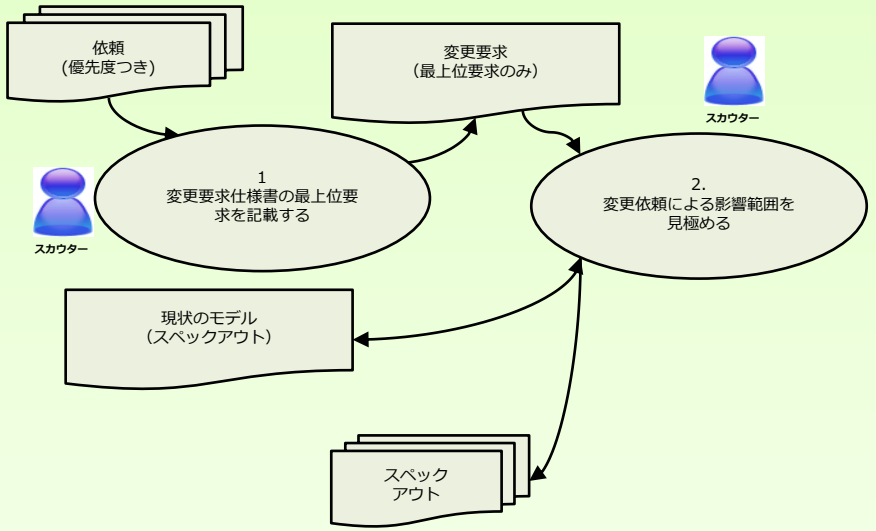
- Scrumで定義されている役割に　「スカウター」を追加
- スプリントゼロで変更に対する既存システムへの影響を明確にし、プロジェクトの要求の優先順位を、ビジネスの側面だけでなく、システムへの影響を加味して決定する

対策：スクラムＸの導入

問題①の対策：スクラムXの導入

- スカウターの導入
- スプリントゼロの実施
- 複数回デリバリーの実施





モデリングスキル「高」のメンバーを「スカウター」とし、「スプリントゼロ」でドメイン知識の習得させることで製品への影響分析精度を向上させることができた。

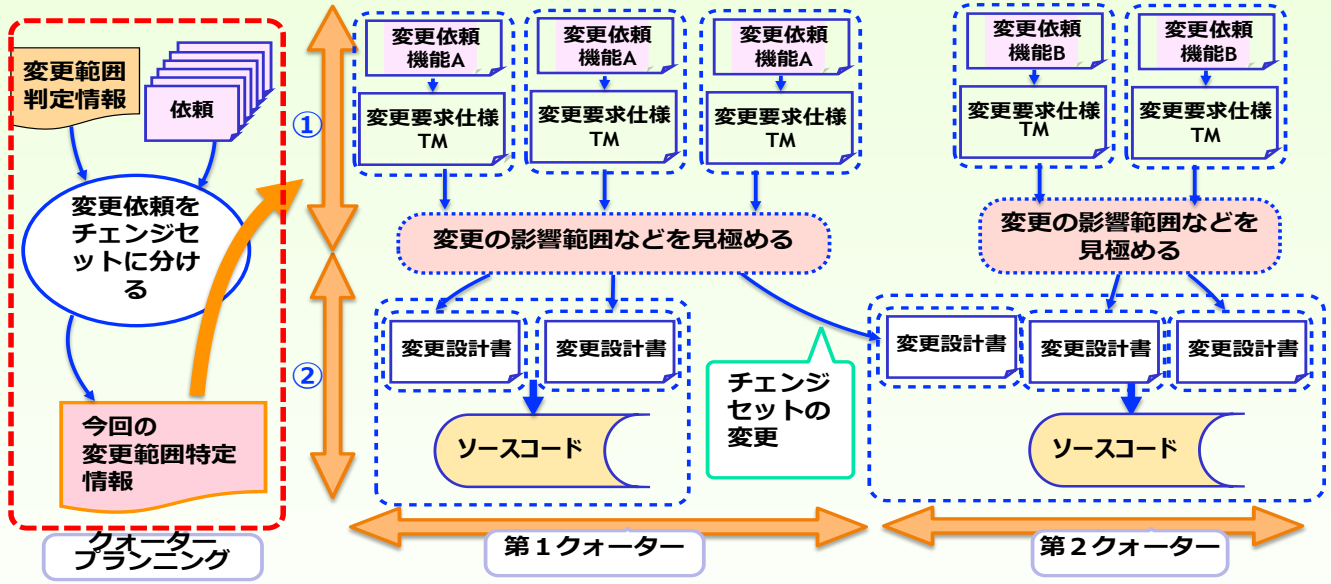
問題①の対策：スクラムＸの導入

スクラムＸ導入結果

スカウター&スプリントゼロを実施したことで
①スプリントゼロでスカウターが事前にスペックアウトしたことで、変更のキーポイントとなるところを早期に発見でき事前に要求を整合できた。
②処理負荷、メモリリソースなど弊社では調査できずに結果次第で手戻りが発生する箇所に対してスプリントゼロで気づき、**分割してデリバリーすることで手戻りの発生を防いだ。**

スカウター

第一クォーターでは処理負荷、リソース関連のチェックをする
第二クォーターは動作確認が必要な要求を実装する。・・・



第2部 スクラムX 2.0の部分適用

- プロジェクト紹介
- プロジェクトの問題点
- スクラムＸ 2.0で問題解決
  - 問題点①にスクラムＸを適用
  - 問題点②にスクラムＸを適用

問題点②に「スクラムＸ 2.0」を適用

問題点	対策		効果
モデリングスキルが低いメンバーの品質への不安	対策	ＸＤＤＰ	• 変更設計書を記入し、変更設計書をレビューすることで品質を担保
	さらに対策	スクラムＸ	• 変更設計書にモデルを記入することで、変更に対する影響を早く正確に知ることができる • モデリングスキルが低いエンジニアを育成できるため、モデルの設計品質が向上する

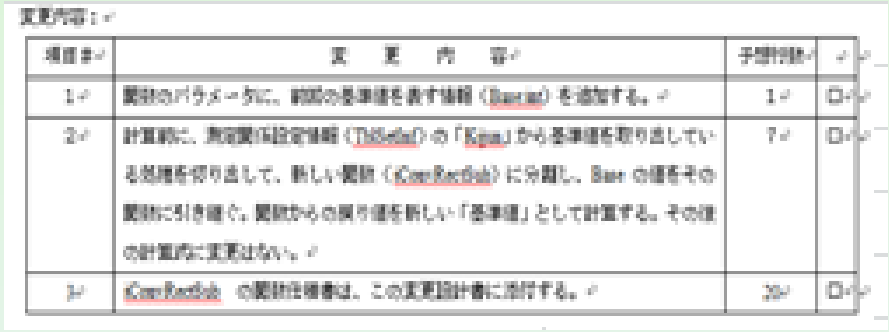
問題②の対策：XDDPの導入

問題

②モデリングスキルが低いメンバーの品質への不安

対策： XDDPの導入

「要求＋仕様」に対して、適切な変更手段を示すために、「変更設計書」へ書き出す。
変更設計書をレビューすることで、品質を担保する。



変更内容：

項目#	変　更　内　容	予想時間	
1	関数のパラメータに、初期の基準値を表す情報（Base）を追加する。	1	□
2	計算前に、無効関数設定情報（ThisData）の「Kpa」から基準値を取り出している処理を切り出して、新しい関数（ConFunc2nd）に分離し、Base の値をその関数に引き渡す。関数からの戻り値を新しい「基準値」として計算する。その後の計算内に変更はない。	7	□
3	ConFunc2nd の関数仕様書は、この変更設計書に従属する。	30	□

一般的な「変更設計書」

問題②の対策に対してさらに追加対策を実施

- さらに・・・
  - モデリングスキル低のエンジニアを育成するために
    - モデリングスキルの高いスカウターがモデルの変更範囲に当たりをつけてモデリングスキルが低い担当者が変更設計をする
    - 変更設計を短期間で行いレビューする
  - 成果物はオートコードされるのでモデルが最終成果物。なのでモデルで変更設計を表現したい

スクラムXの導入

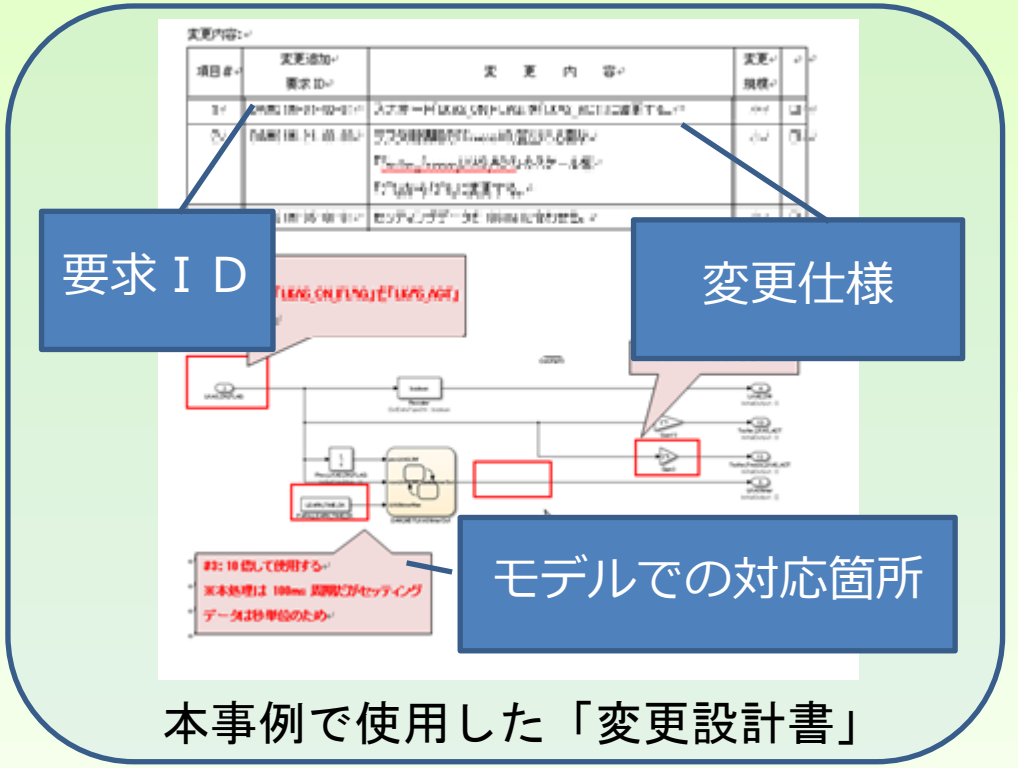
問題②の対策：スクラムＸの導入

変更設計書にモデルを導入

変更設計書に**モデル**を記載する

- 要求ID,変更仕様は変更要求仕様書の内容を転記する。それをモデルのどこで実現しているかをモデルで記載する

- Before/Afterをモデルのキャプチャで表現し、Afterのモデルと変更仕様とをトレースできる情報を変更設計書に記載する



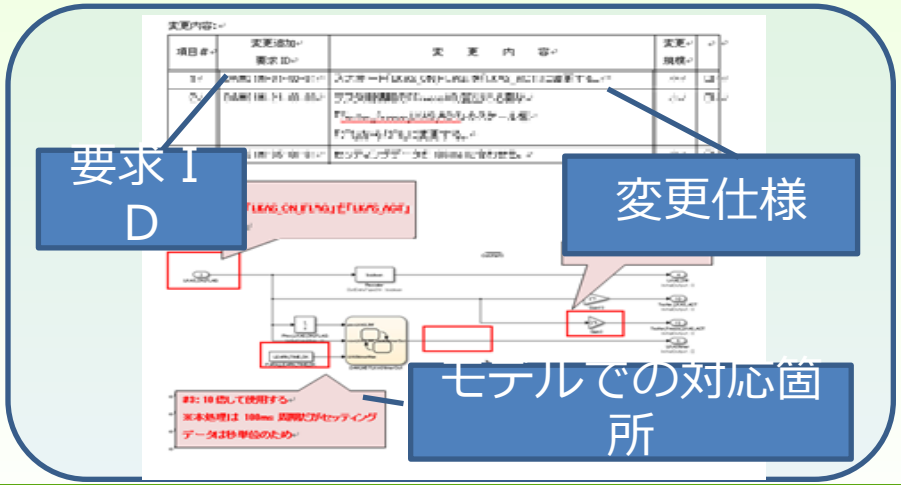
本事例で使用した「変更設計書」

問題②の対策：スクラムＸの導入

スクラムＸ 2.0導入結果

変更設計書に**モデル**を記載することで
①要素間の関連が把握しやすくレビューの品質が向上した
②スキルの低いメンバーを育成することができた
（短期間にレビューを回すことで正しいモデルが何かを理解させることができた）
③スカウターがアーキテクトとしてレビューすることでモデルの品質が安定した

「要求＋仕様」＋「変更方法」が一目瞭然なので、変更作業・レビューともに質とスピードが向上した



スクラムXの効果まとめ

①スカウターの導入＋スプリントゼロ+複数回デリバリー

- スカウターによって変更のキーポイントを早期に発見できた
- スプリントゼロでチェンジセットを検討し、分割して複数回デリバリーすることで手戻りが回避できた

②変更設計書＋モデル

- 変更設計書にモデルを記入することで、モデルに対する影響を早く正確に知ることができた
- モデリングスキルが低いエンジニアを育成でき、設計品質が安定した

最後に

Ｔ６研究会の今後の課題

- 検証プロセスの考察
- アジャイルＵＳＤＭを含めた事例収集
- ガイドライン化
- etc

まだまだ課題がありますのでＴ６研究会でまっています

ご清聴ありがとうございました

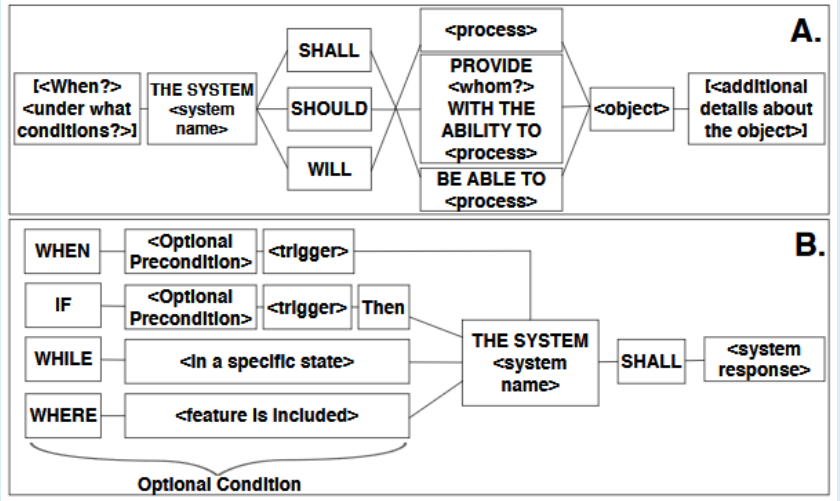
Appendix

世の中の要求表現とアジャイルUSDM

世の中の要求表現例

- IEEE 29148
  Systems and software engineering - Life cycle processes - Requirements engineering
  - 5.2.4 Requirements construct
    - 要求事項の構成例

      [条件][主体][動作][対象][制約]
- ボイラープレート
  - 必要な情報項目を予め定型フォーマットとして配置し、書き手は項目の情報を埋めることで要件を記述する方法



Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand ,
Requirement Boilerplates: Transition from Manually-Enforced to Automatically-Verifiable Natural Language Patterns

アジャイルUSDMの要求表現

条件：While、Where　　条件：When、If　　動作：能力、レスポンス

(だれが)どんな状況で	(だれが)どんなとき	(だれが)どうなったら	(だれに)どうなる

「振る舞いの観点」は、世の中の要求表現にもほぼ対応し、かつ、要求の書き方に悩まずに記述できるようになる

スクラムX 2.0 プロセス