UMTP
UML Based Modeling Technologies Promotion

UMTPアジャイル開発事例セミナー
現場に学ぶ実践アジャイルモデリング

アジャイルにモデリングは必要か

Is “Modeling” required in Agile?

2015/7/23

株式会社ゼンアーキテクツ
岡　大勝

ZENARCHITECTS

自己紹介

岡　大勝

@okahiromasa

株式会社ゼンアーキテクツ

CEO/チーフアーキテクト

プロフィール

- 第一勧銀情報システム（現：みずほ情報総研）
  - VOS3 COBOL＆MS-Cプログラマ
- 日本ディジタルイクイップメント（日本DEC）
  - 主に生保・損保・銀行向けの資産運用システムに携わる。
  - Alpha NT, SQL Server, DECnet
- 日本ヒューレット・パッカード C&I-Financial
  - 金融機関向けのシステムアーキテクチャ設計、開発プロセス設計、運用プロセス設計を行う。
  - HP-UX, J2EE, WebLogic, Oracle Database, OO
- 日本ラショナルソフトウェア
  - 開発プロセス（RUP）およびオブジェクト指向分析設計手法の導入コンサルティング。
  - RUP, Rose, ClearCase, Functional Tester
- ゼンアーキテクツ
  - 2003年よりIT設計事務所として活動。お客さまのIT投資の最適化を目指し、アーキテクチャ中心のプロセス改善を推進。

著作/翻訳

- 「要求」の基本原則（技術評論社）2009
- 本当に使える開発プロセス（日経BP社）2012
- ディシプリンド・アジャイル・デリバリー（翔泳社）2013

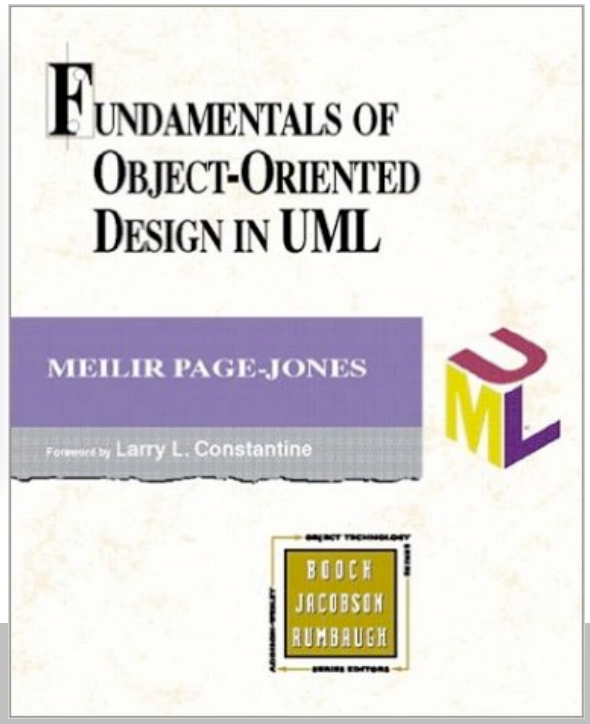
UMLは、いまどうなのか？

- 共通言語として使えて当たり前。
- 「読めない」「書けない」では話にならない。
- SWEBOKv3の5つのKAで“前提”
  - 要求・設計・実装・プロセス・モデルと手法

SWEBOK® V3.0

Guide to the Software Engineering Body of Knowledge

http://www.computer.org/web/swebok



1. アジャイルでの「モデル」を整理する

2. アジャイルプロジェクトでモデルを活用する

ZENARCHITECTS

アジャイルにモデリングは必要か？

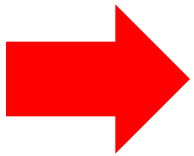
- 現実には、いつも、どんなアジャイルプロジェクトでも行われている。
  - UMLだけがモデルではない
  - 「モデル」と「文書」は異なる
  - 精緻で正確でビジュアルに描かれたものだけがモデルではない
  - 意識していない活動を含む

アジャイルに限らず、知的生産活動には
モデリングは欠かせない

アジャイルは組織活動を変革した

■ 計画管理/要求管理

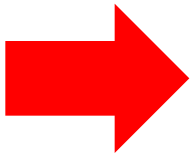
予測型
（計画駆動）



適応型
（変化駆動/アジャイル）

■ 開発工程

成果物駆動
Document Driven



ジャストイン
タイム
Just-In-Time

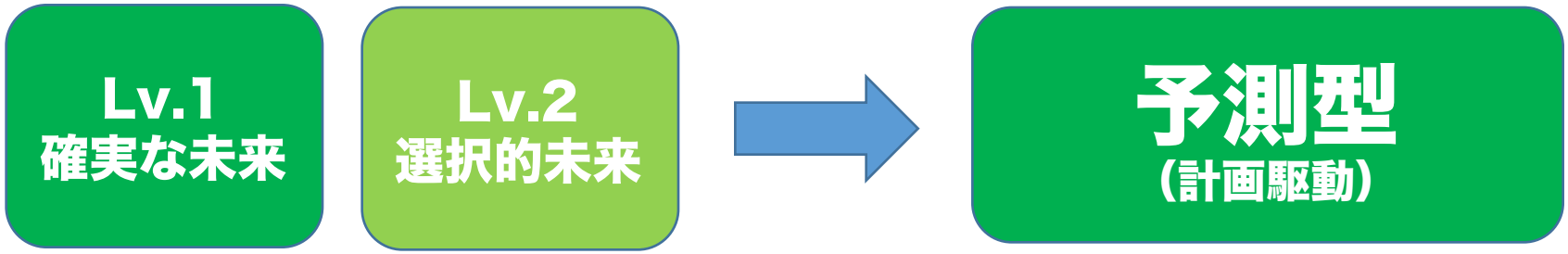
詳しくは「企業システムにアジャイルは必要か」スライドを参照
http://www.slideshare.net/hiromasaoka/agile-is-really-need-to-enterprise-system-49711192

計画できることは、計画して実施する

- 「正しい」ことが分かっている
  - よほど大きな問題が発生しない限り“うまくいく”
  - 事業者は「必要な時期」に「必要なもの」が手に入れば良い。

- 定型反復業務
- 確立された手順

PMBOK 5th

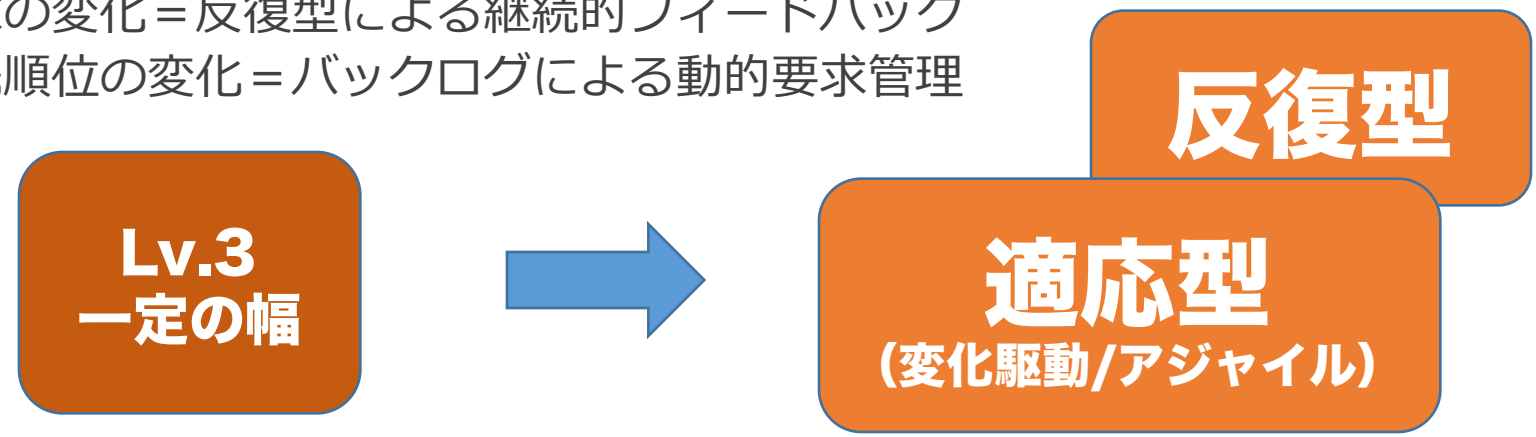


計画できないことは、確認しながら進む

- 何が「正しい」のか判断できない
- 正しいものが変化する
  - 機能
  - 技術
  - ビジネス、マーケット

- 変化への継続的な対応
- 「要求の変化」と「優先順位の変化」
  - 要求の変化＝反復型による継続的フィードバック
  - 優先順位の変化＝バックログによる動的要求管理

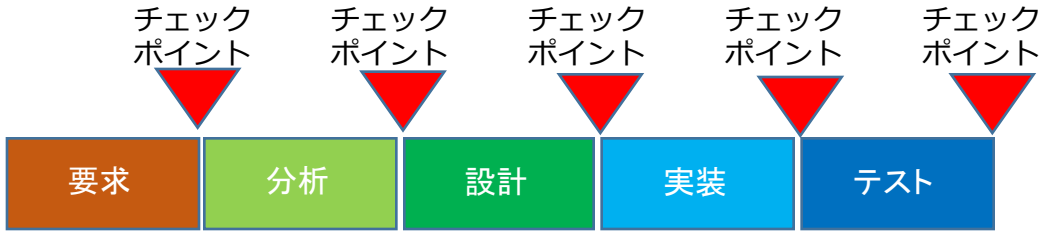
PMBOK 5th



ジャストインタイム（JIT）

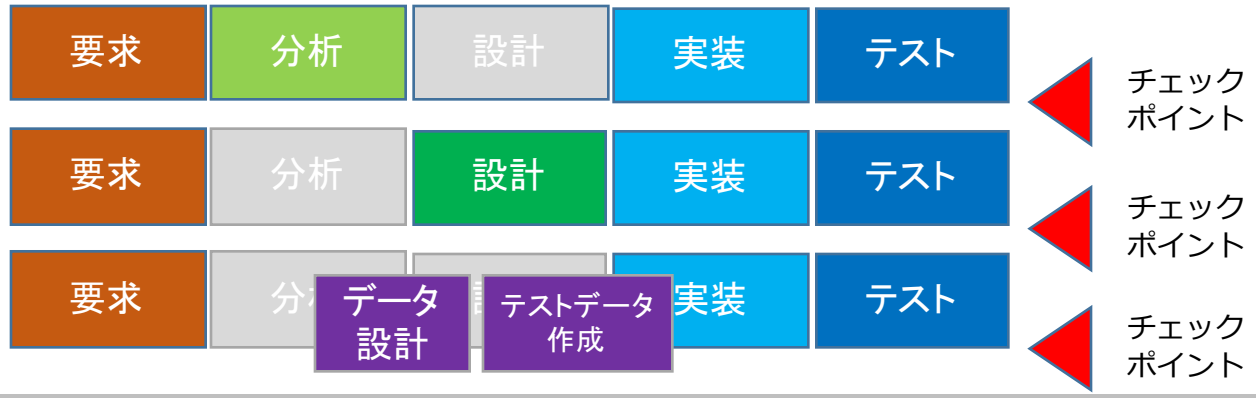
- 作業の管理を、テスト結果によって行う
  - 中間成果物は「それが必要な場合に」作成する

成果物駆動



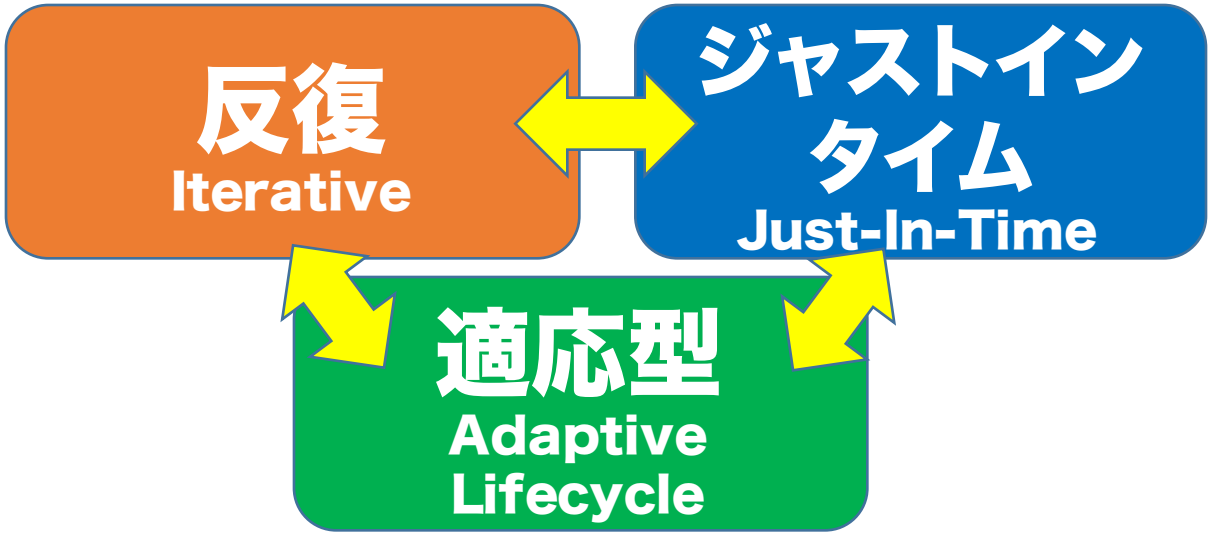
タスクそのものを「必要に応じて」実施

ジャストインタイム



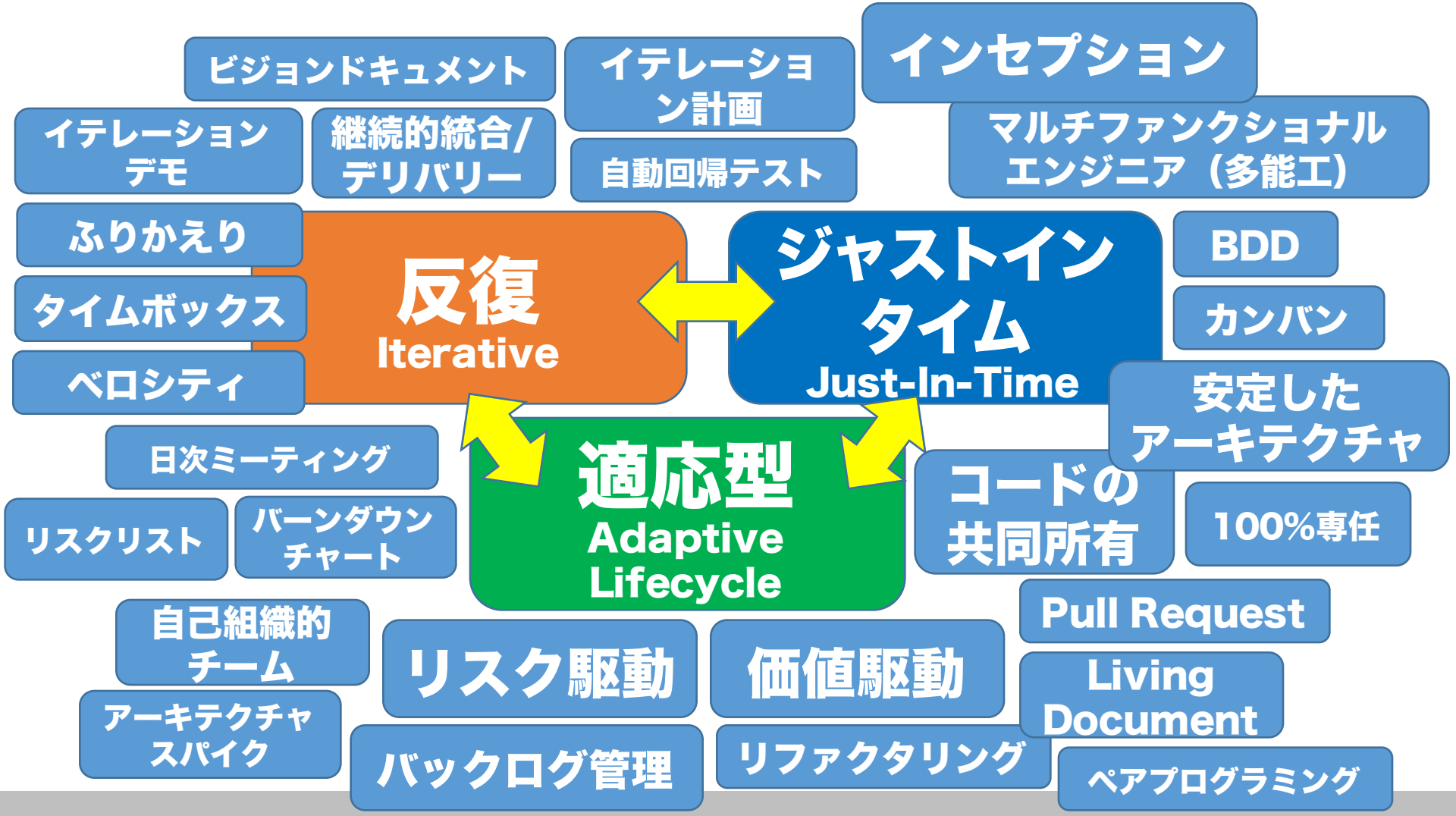
アジャイルの骨格

アジャイルは、成果物や文書に影響されない



つまり「何でもいい」

アジャイルは、成果物に依存しないよう うまく設計されている



ZENARCHITECTS

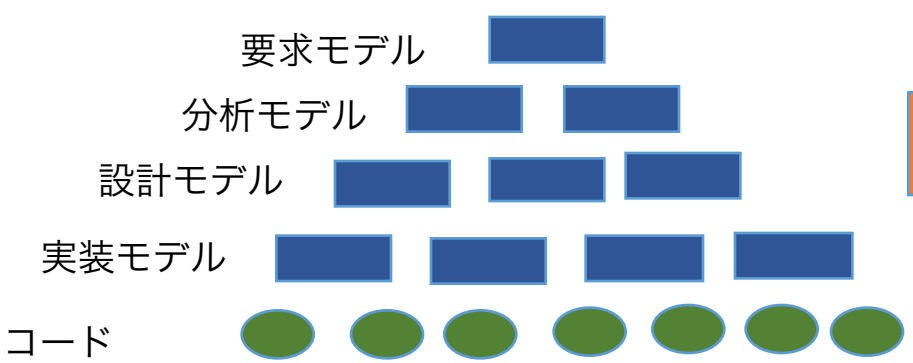
※ゼンアーキテクツがお客さまの現場で実践している主要なプラクティスを表したものです

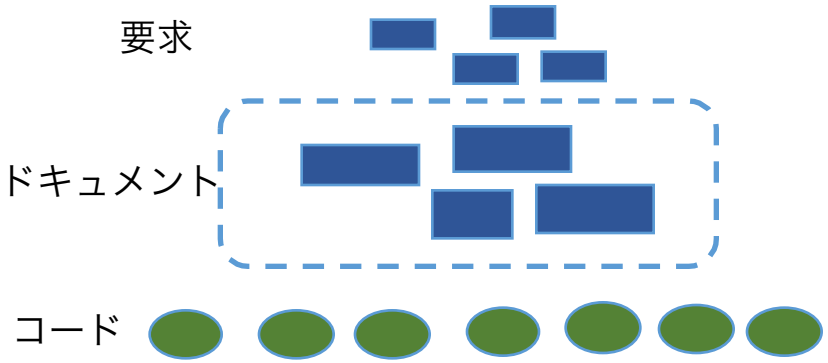
「成果物」の意味が変わった

成果物駆動
「次工程の入力」

→

ジャストインタイム
「必要に応じて作るもの」





成果物を作成する活動（＝ドキュメンテーション）は、成果物駆動型の主たる構造
コード不要の世界を目指していた

- MDA/MDD
- Executable UML

ジャストインタイムでは「ドキュメントはコードを補うもの」
「Collective Code Ownership ＝コードの共同所有」が根源的価値

- 開発言語の抽象度の高まり
- DSL

ZENARCHITECTS

何が起こったか

- 「文書化」の目的が変わった

成果物駆動

- 「ドキュメントの連続性」でプロジェクトを構築
  - 検討し残しや書き残しのないように、すべきことを「ドキュメント記載項目」で表現する

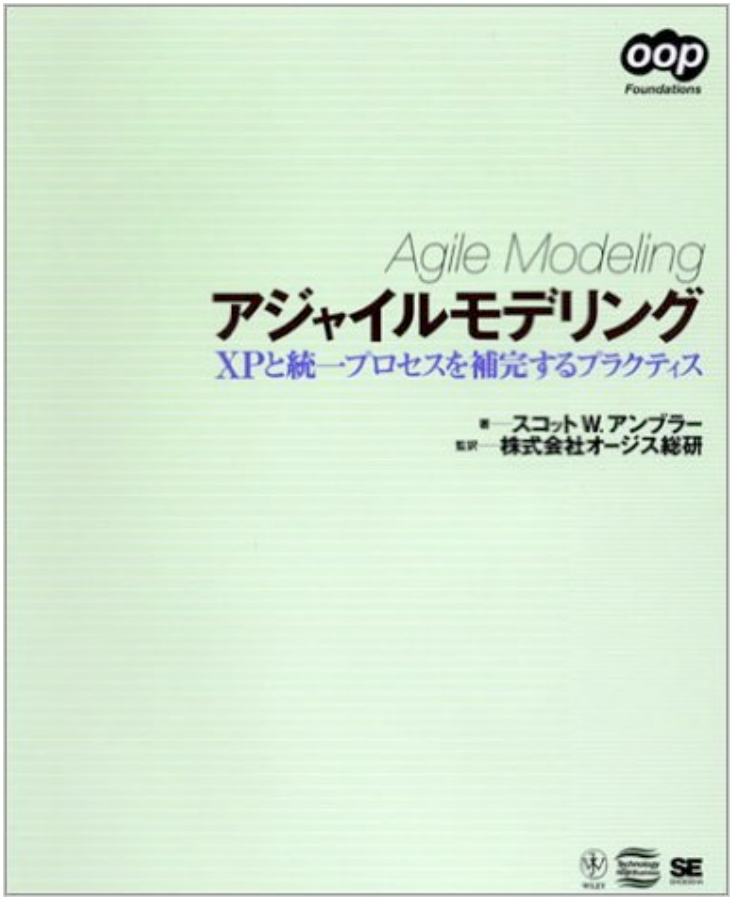
JIT

- 「動作するコードの積み重ね」でプロジェクトを構築
  - 事前に明らかにすべきことは確認する。必要に応じて、適切な方法で伝える。

「アジャイルモデリング」

- Agile Modeling
  - 2002年：英語版
  - 2003年：日本語版
- スコット.W.アンブラー 著
  - 他の著作
  - Enterprise Unified Process
  - データベースリファクタリング
  - ディシプリンド・アジャイル・デリバリー

- モデリングを“アジャイル”のコンテキストに適用するためのガイドブック。
- 「ソフトウェア開発プロジェクトで適用できる、効果的で手軽にソフトウェアをモデリングするための価値、原則、およびプラクティスを集めたもの」「アジャイルなモデルは完璧である必要はない」

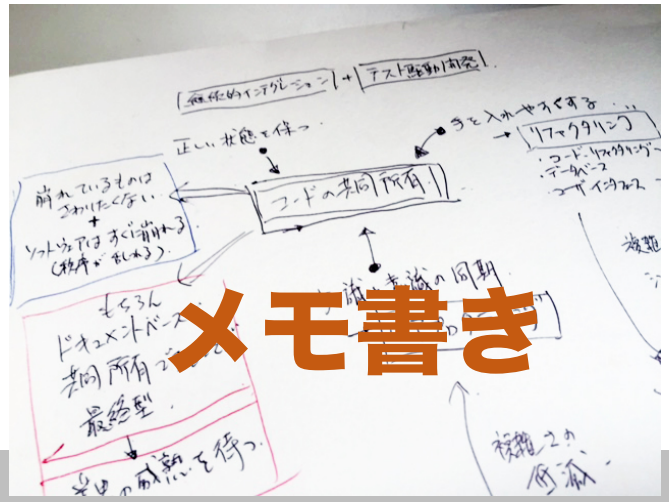


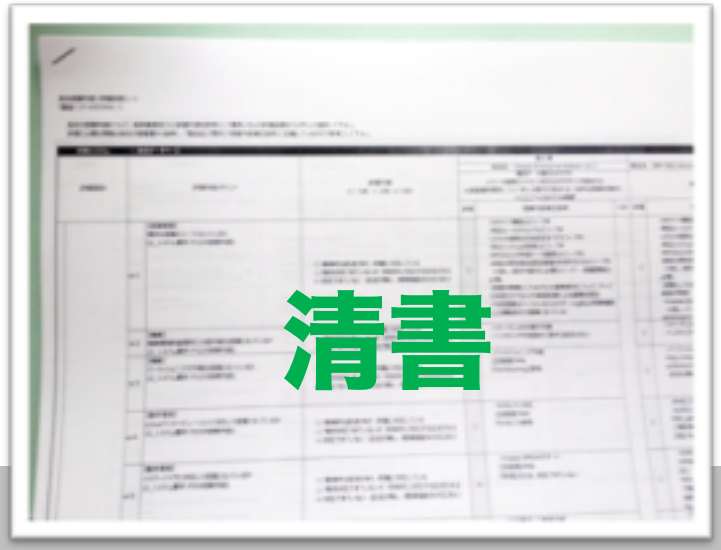
http://www.ogis-ri.co.jp/otc/swec/process/am-res/am/

「メモ書き」と「清書」の分離

- 成果物駆動では、メモ書きは捨てるもの
  - 大事なことは「正式なドキュメント」として記載

- JITでは、「メモ書き」の比率が高まる
  - 本当にメモ書きを捨てていいのか？
  - 大事なことは、どこに書くのか？





ZENARCHITECTS

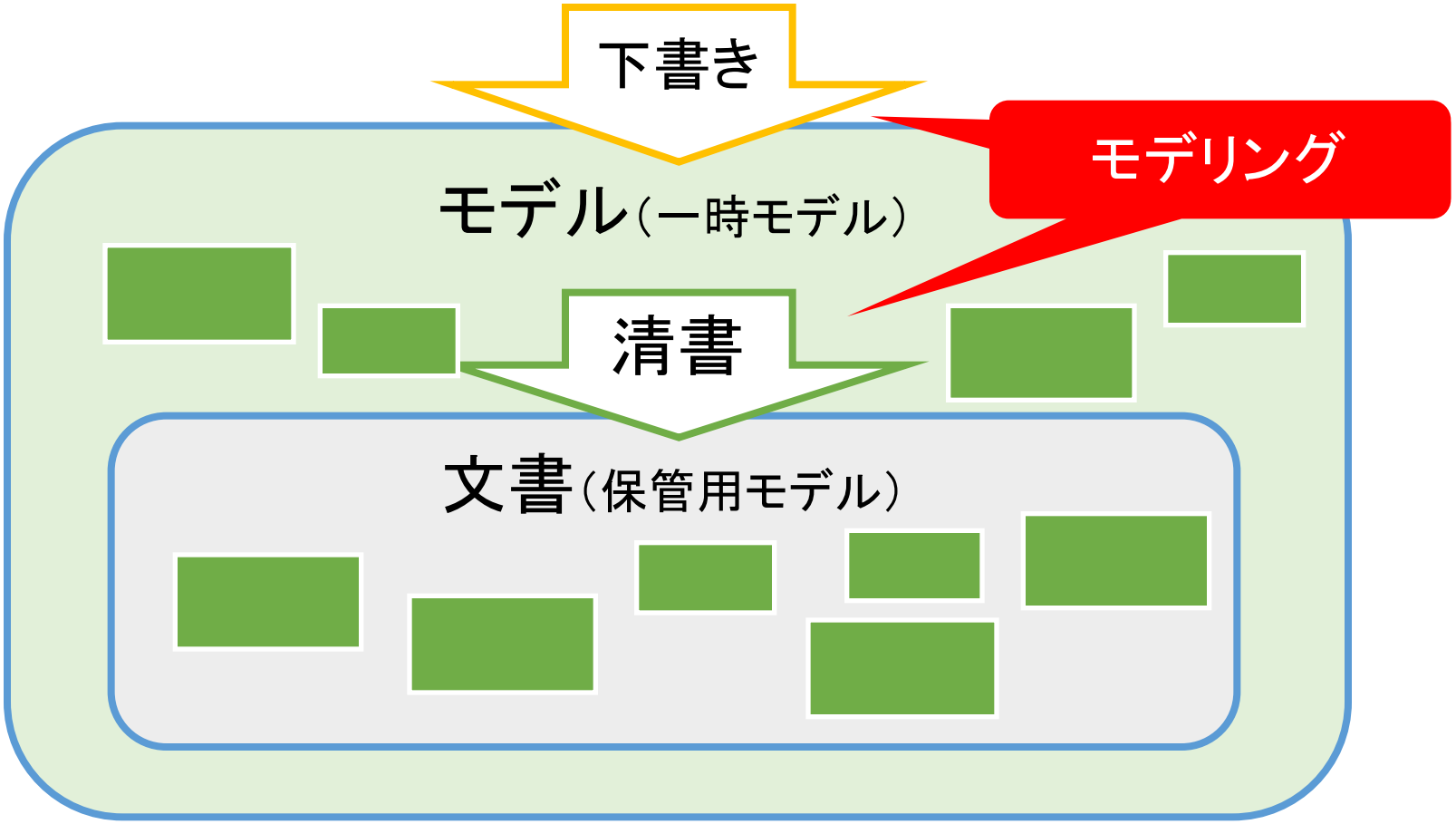
モデルとは何か

「問題の1つ以上の側面、その問題に対して考えられる解決策を抽象的に記述したもの。図や文章で示される」

「アジャイルモデリング」での定義

- 視覚的モデル
  - フローチャート, 状態マシン, ER図, DFD, OMT, Booch法, UML(ユースケース図, クラス図, アクティビティ図,..), BPMN, Value Stream Map, Lean Canvas, ユーザーストーリーマップ, 任意のアーキテクチャ図
- 任意のモデル
  - 要求一覧, 機能一覧, 業務ルール, 計算ロジック, ユーザーストーリー, CRCカード,インタフェース仕様, ポンチ絵, サンプルコード, 任意のメモ書き

モデルと文書とモデリングの関係



モデリングという活動

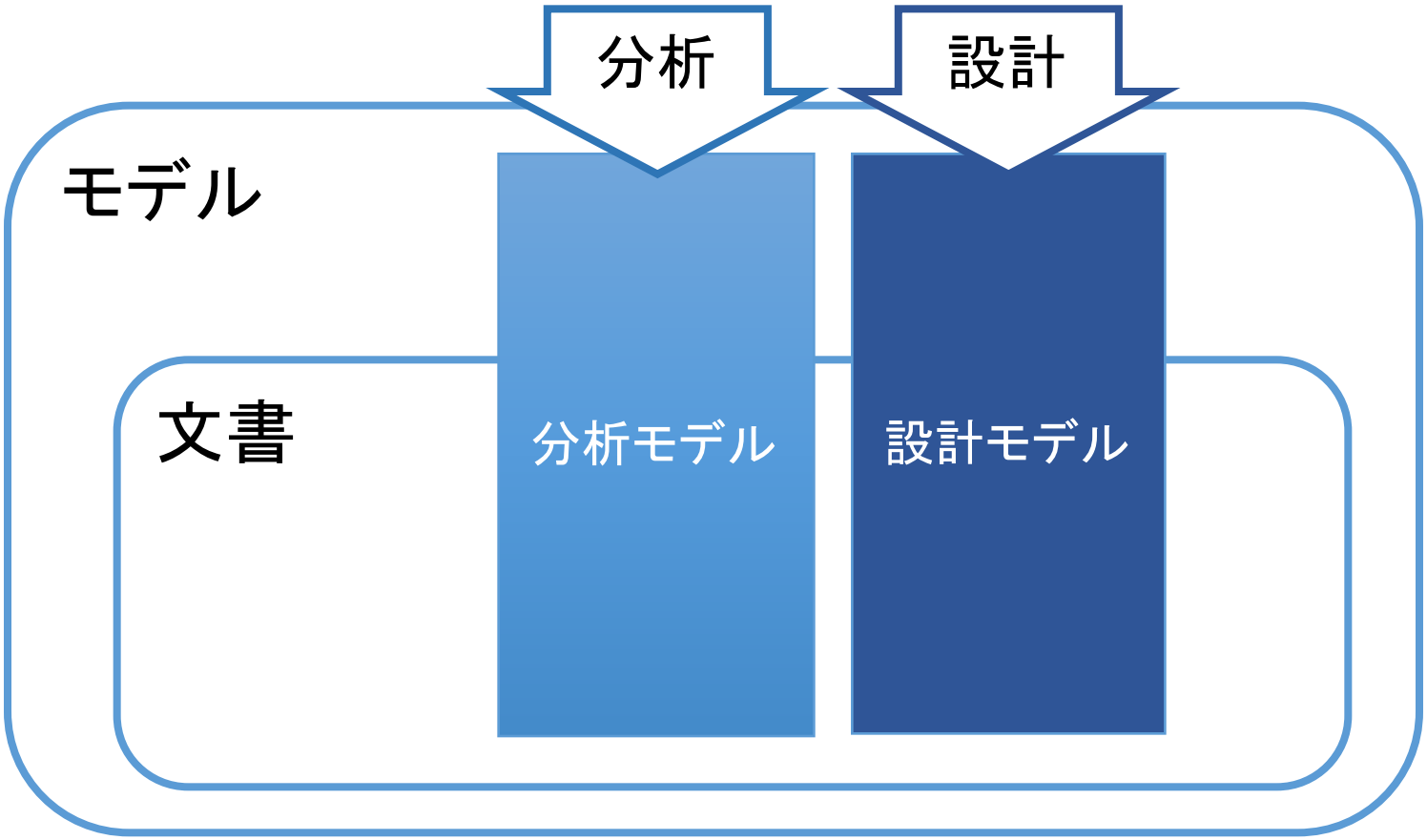
1. ありのままを表現する
   (これを“分析”と呼ぶ)
2. 作りたいもの(なりたい姿)を表現する
   (これを“設計”と呼ぶ)

分析

設計

ZENARCHITECTS

分析と設計



モデリングは多面的である

- 例えば要求モデリング
- プロダクトバックログに積まれたユーザーストーリー(ストーリー一覧)だけで、要求が適切に表現できるだろうか。

- 複数の軸で視覚的に分類して俯瞰したい
  - ユーザーストーリーマッピング
- アクターを軸にグルーピングして俯瞰したい
  - ユースケース図で全体を把握
- 業務にマッピングして俯瞰したい
  - ビジネスプロセスマッピング
- 全ての要求の優先順位を知りたい
  - プロダクトバックログ

自然に、複数のモデルを併用しているはず

【AM】なぜドキュメントを作成するのか

1. プロジェクトの利害関係者が要求しているため
2. 取り決めモデルを定義するため
   - インターフェイスを含むシステム間の相互作用
3. 外部グループとのコミュニケーションをサポートするため
   - 書いたことは誤解されやすいが、補助的なメカニズムとしては優れている
   - 「話すためにモデリングしよう」
4. 何かについてじっくり考えるため
   - 「理解するためにモデリングしよう」

“モデリングは後続工程のために行うものではない”

【アジャイルモデリング 14.1 より引用】

用語を整理する

✓文書（Document）
- 長期間維持できる方法で情報を伝える目的を持つ。

✓モデル（Model）
- 問題の1つ以上の側面、その問題に対して考えられる解決策を抽象的に記述したもの。図や文章で示される。

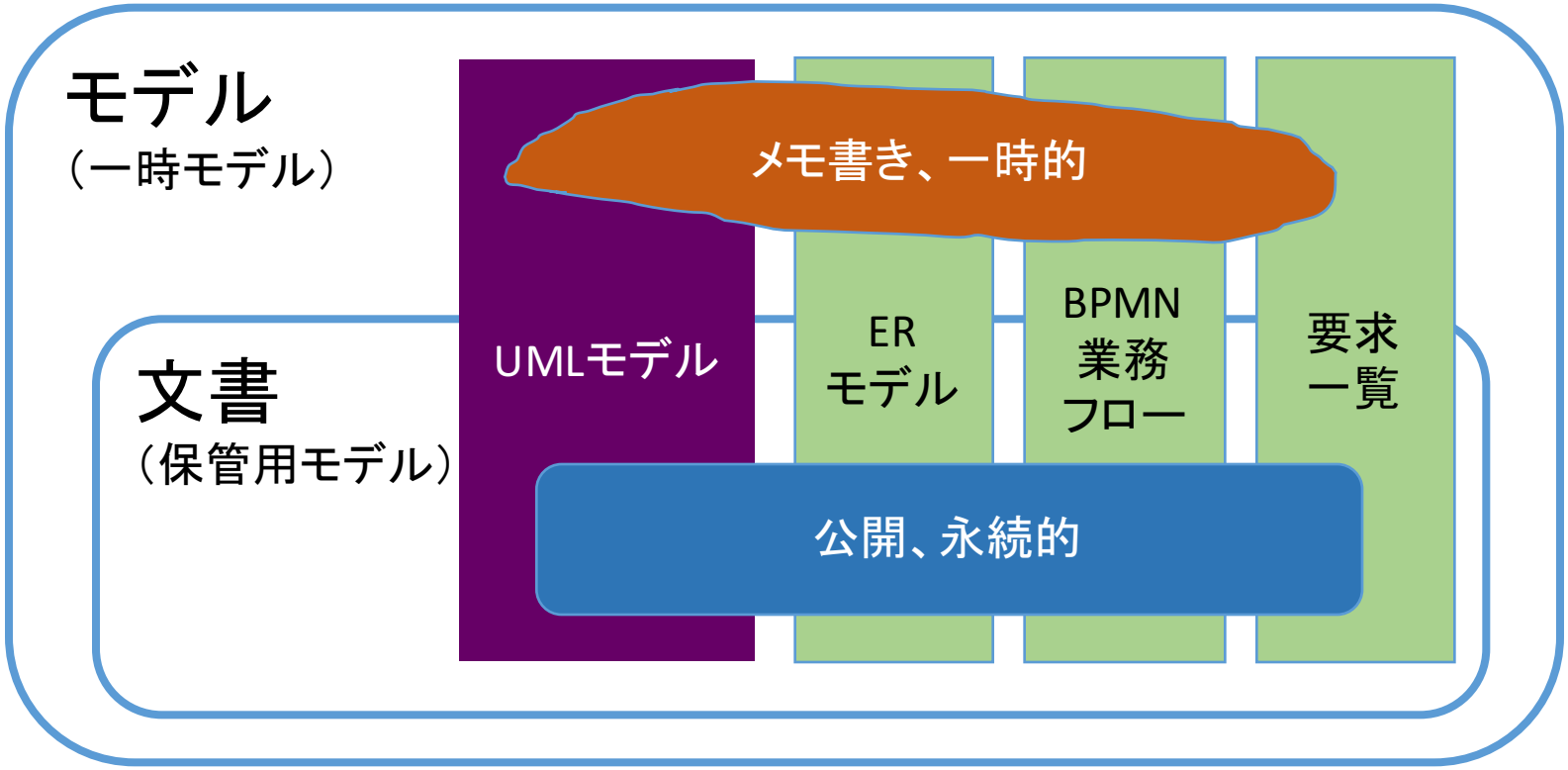
✓ドキュメント（Documentation）
- 他の人のためにシステムについて記述した永続情報。文書とコードのコメント両方を含む。

✓コード（Source Code）
- コンピュータシステムに対する一連の命令と、命令について記述したコメント。

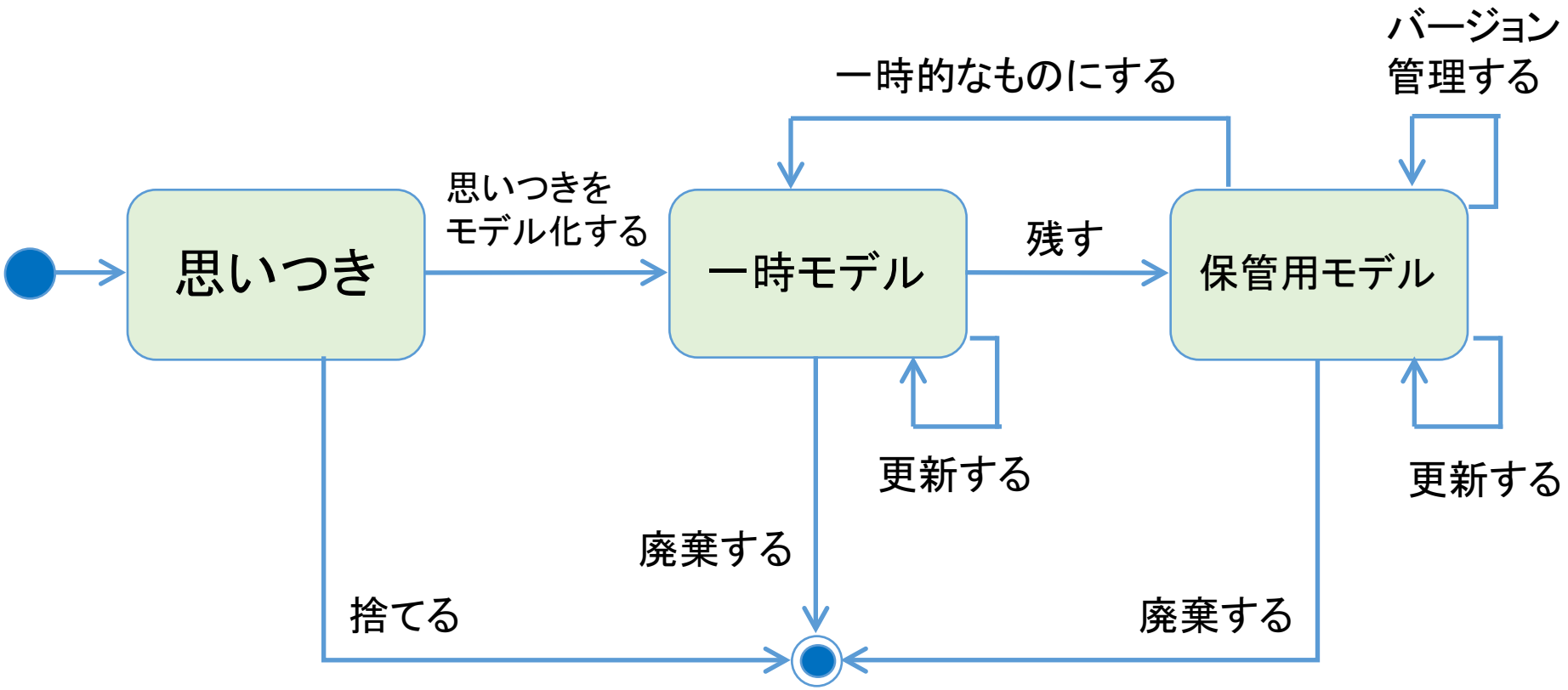
✓成果物（Artifact）
- 納品物または生産した製品。

【アジャイルモデリングでの定義】

UMLモデルの位置づけ



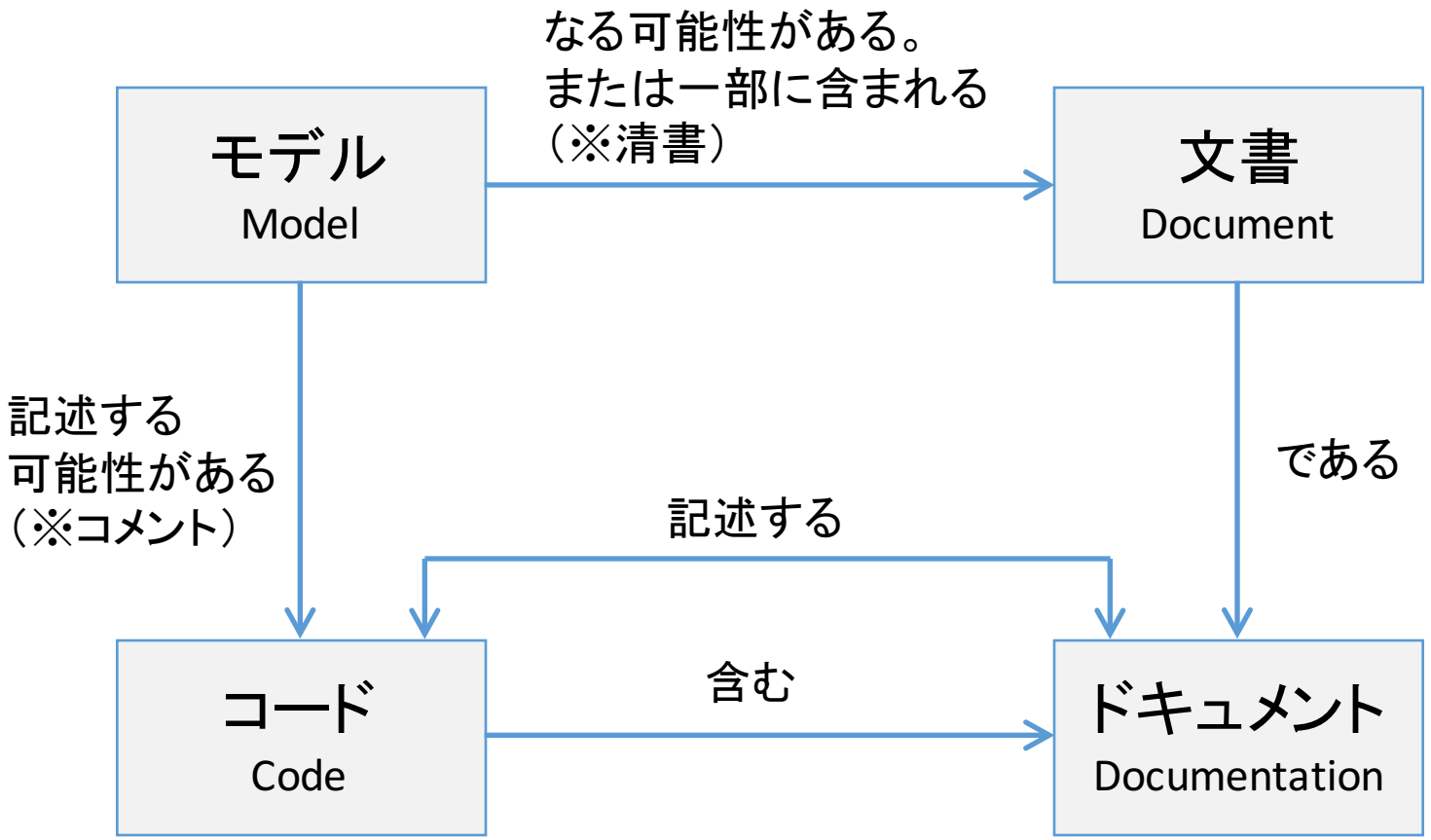
アジャイルなモデルのライフサイクル



ZENARCHITECTS

アジャイルモデリング　図14.2より

モデル、文書、コード、ドキュメント



アジャイルモデリング　図14.1より

1. アジャイルでの「モデル」を整理する

2. アジャイルプロジェクトでモデルを活用する

ZENARCHITECTS

アジャイルプロジェクトの一生

2つの期間

- 「安定するまで」
- 「安定してから」

よく耳にするのは「安定してから」の話

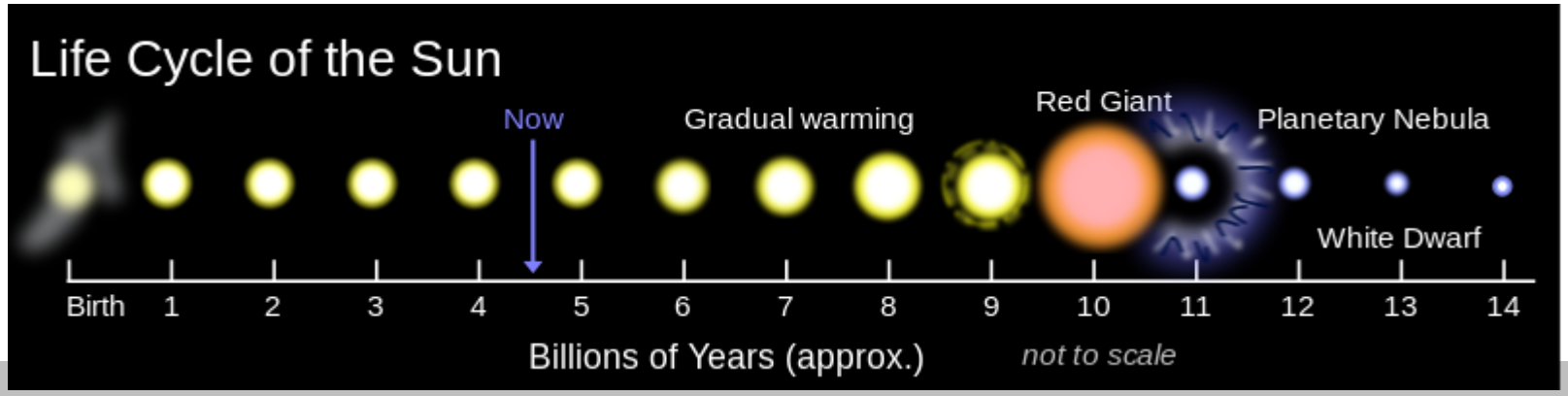


Image source by Wikipedia

アジャイルは「安定するまで」が難しい

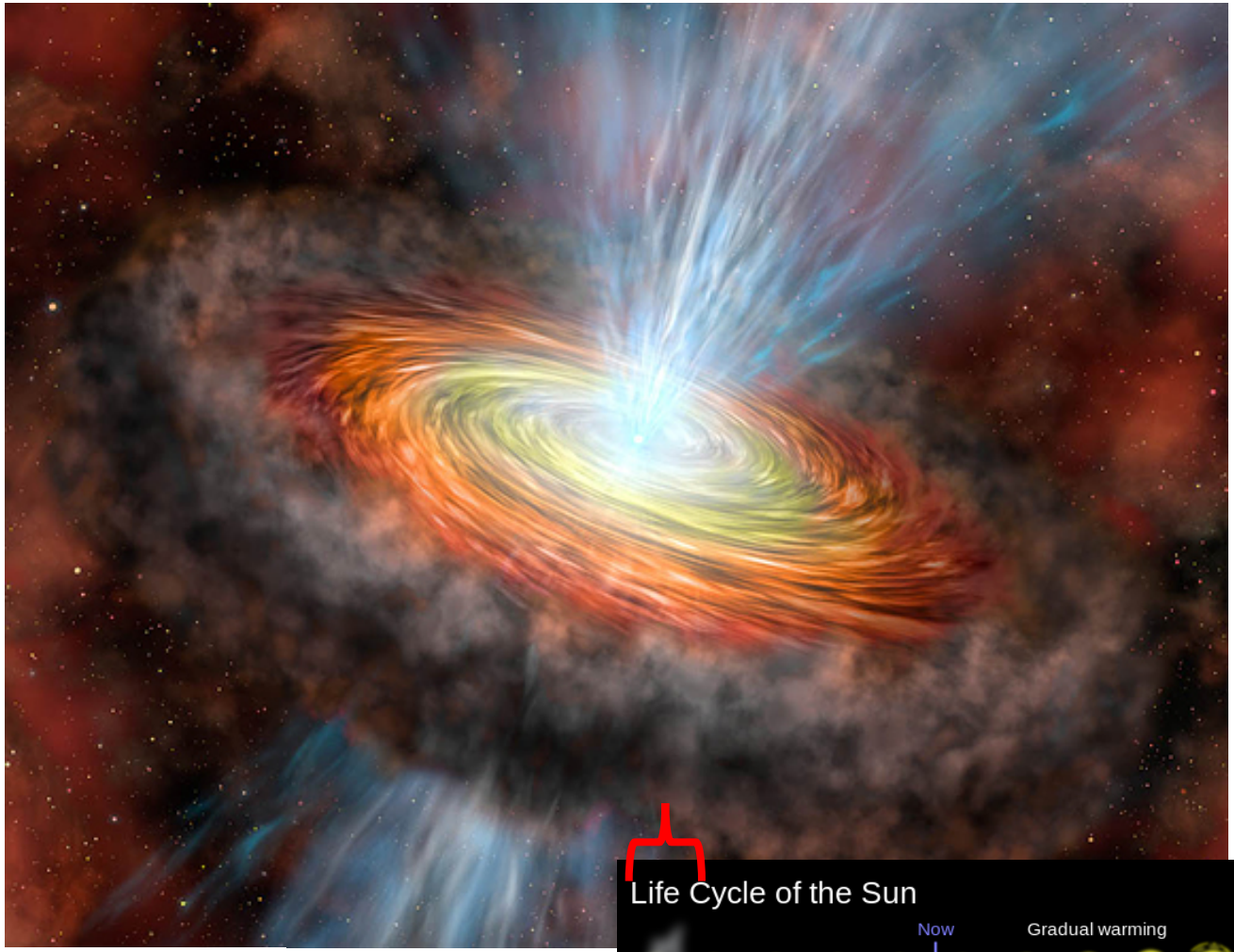
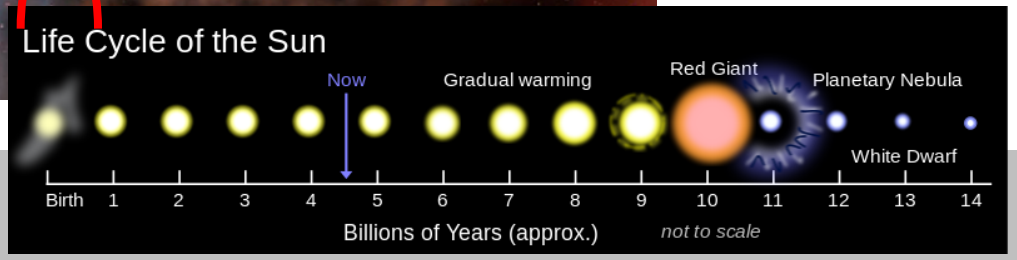


Image source by SpaceRef.com

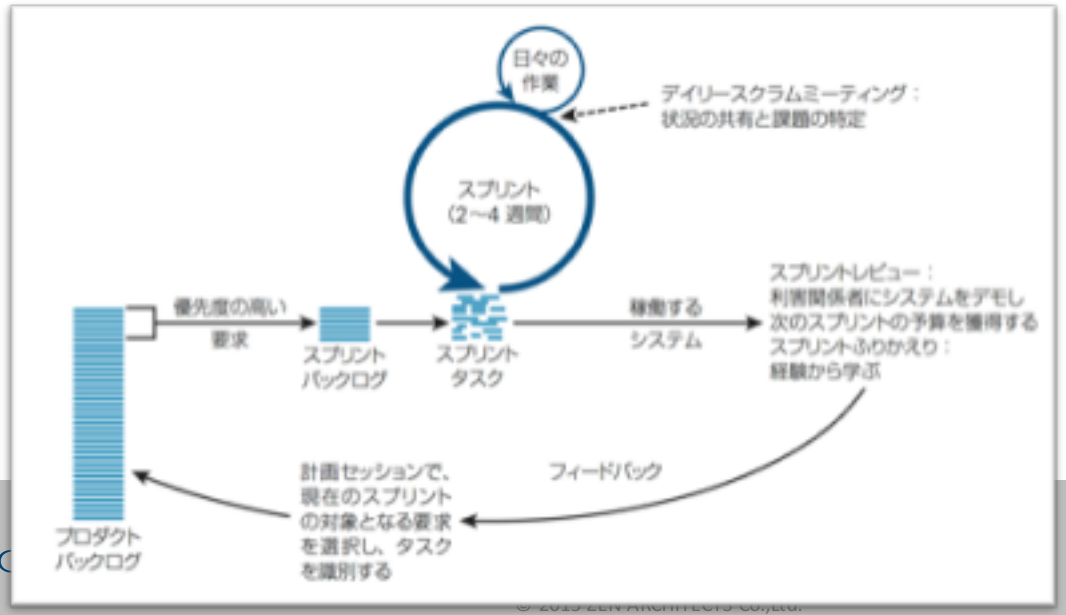


ZENARCHITECTS

アジャイル プロジェクト イニシエーション

- いきなり書けと言われても
  - どんな技術を使うのか
  - いつ、何ができるのか
- 誰が、何をすればいいのか
  - だれが決めるのか・・・

Agile Project Initiation で安定させる



アジャイルの立ち上げを安定させる

- アジャイル プロジェクト イニシエーション
  - Agile Project Initiation
  - 安定したアジャイルライフサイクルへの助走

➢Iteration Zero
  - XP

➢Sprint 0
  - Scrum

➢方向付けフェーズ
  - ディシプリンド・アジャイル・デリバリー

安定した継続的活動の確立

見積りのベースラインづくり

何を安定させるのか

アジャイルプロジェクトを安定させるための３つの観点

1. 要求を安定させる

- たくさんの“やりたいこと”を、観点を合わせ、優先順位順に並べ、スコープの目星をつける
- プロダクトバックログ：安定した入力

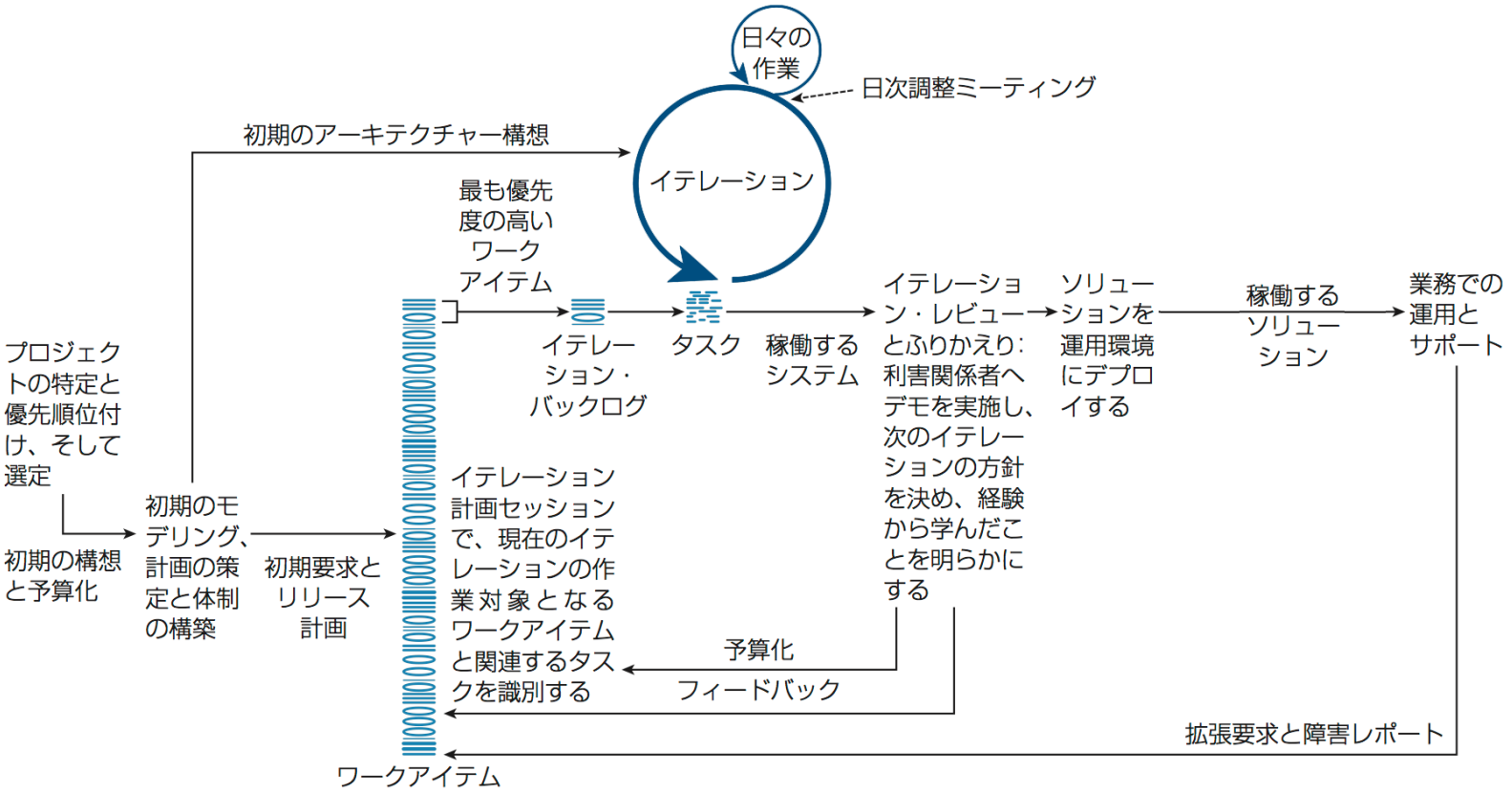
2. アーキテクチャを安定させる

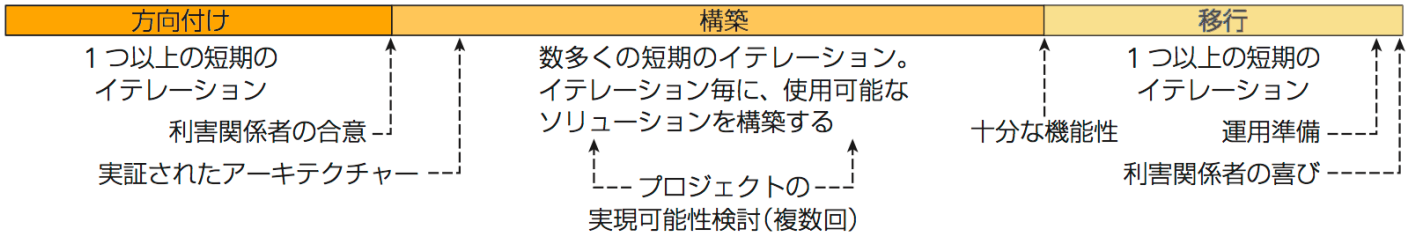
- チームの試行錯誤を最小化し、誰もが安定して開発（ジャストインタイム）が進められるよう、技術の組み合わせや書き方のパターンを共有する
- アーキテクチャドキュメント：安定した開発活動

3. 計画を安定させる

- 個々の要求の実現コストが予測できる（見積基礎値）ようになってくると、リリース時の要求実現状況のぶれ幅が小さくなる
- リリース計画：安定した計画と実績

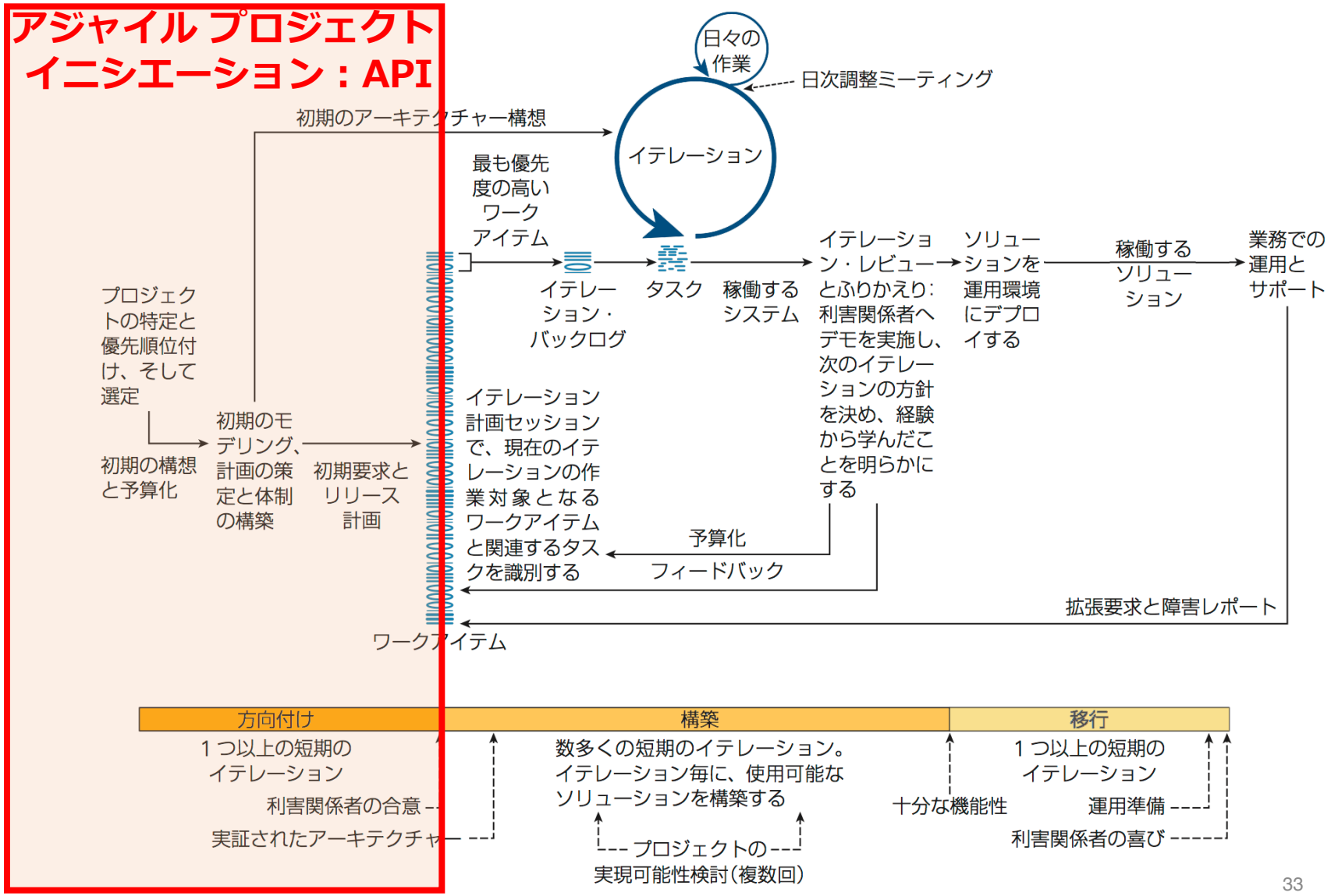
ディシプリンド・アジャイル・デリバリー（DAD）





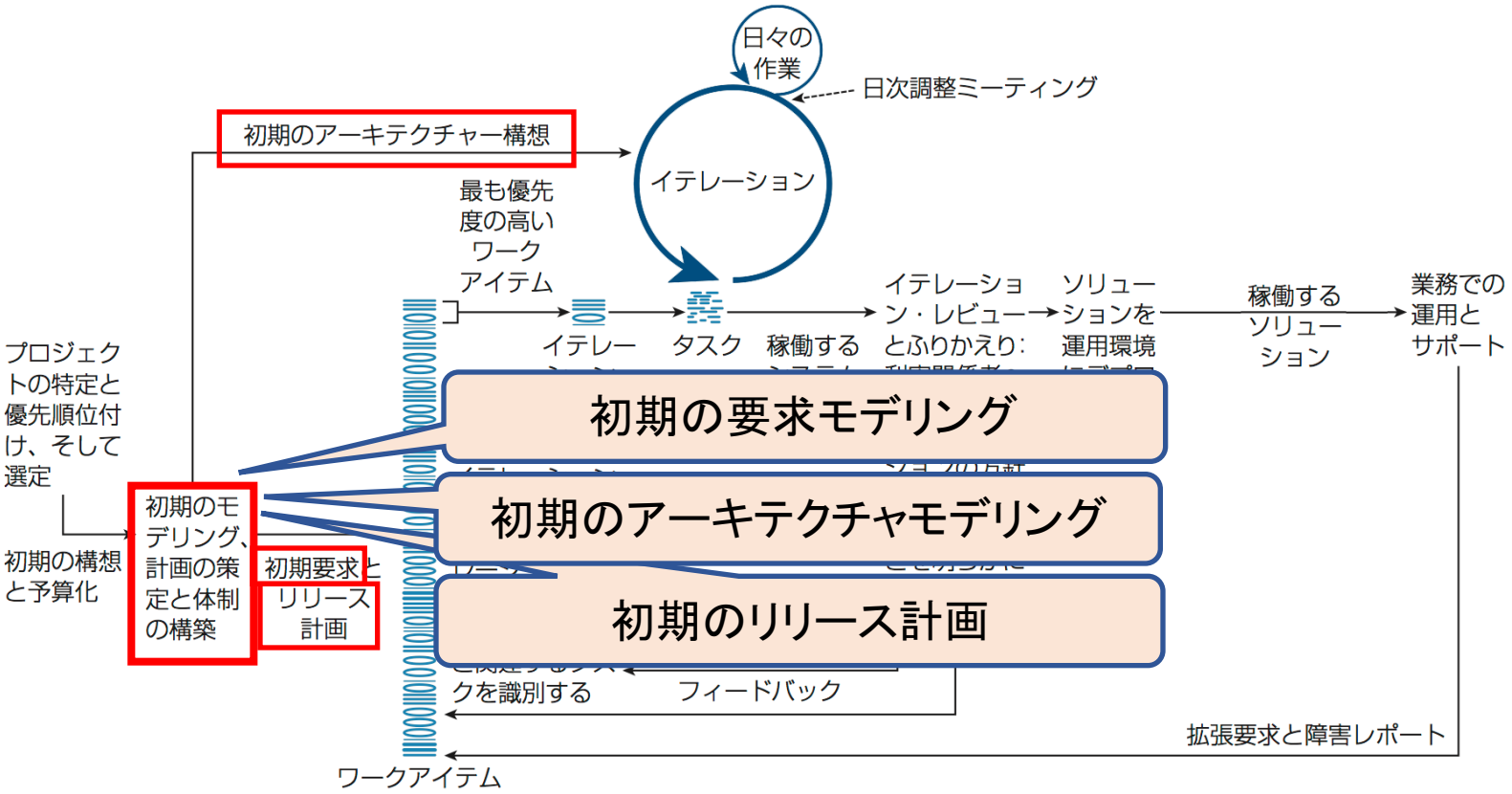
引用：ディシプリンド・アジャイル・デリバリー(2013/翔泳社)

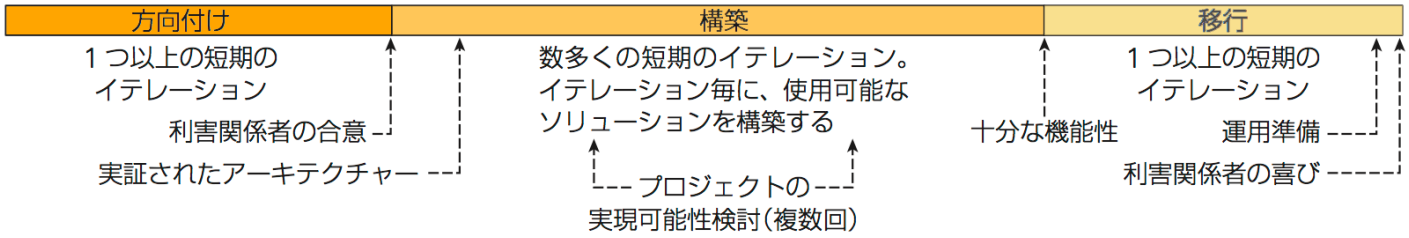
DADでのAPI（Agile Project Initiation）の位置 ＝アジャイルプロジェクトの立ち上げ



引用：ディシプリンド・アジャイル・デリバリー(2013/翔泳社)

APIで行うべき３種類の活動



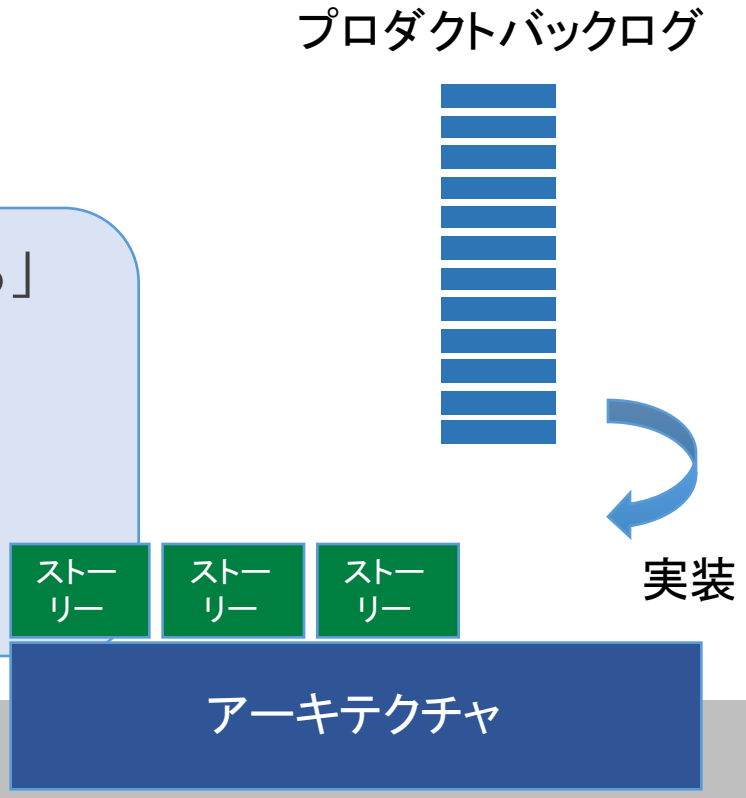


引用：ディシプリンド・アジャイル・デリバリー(2013/翔泳社)

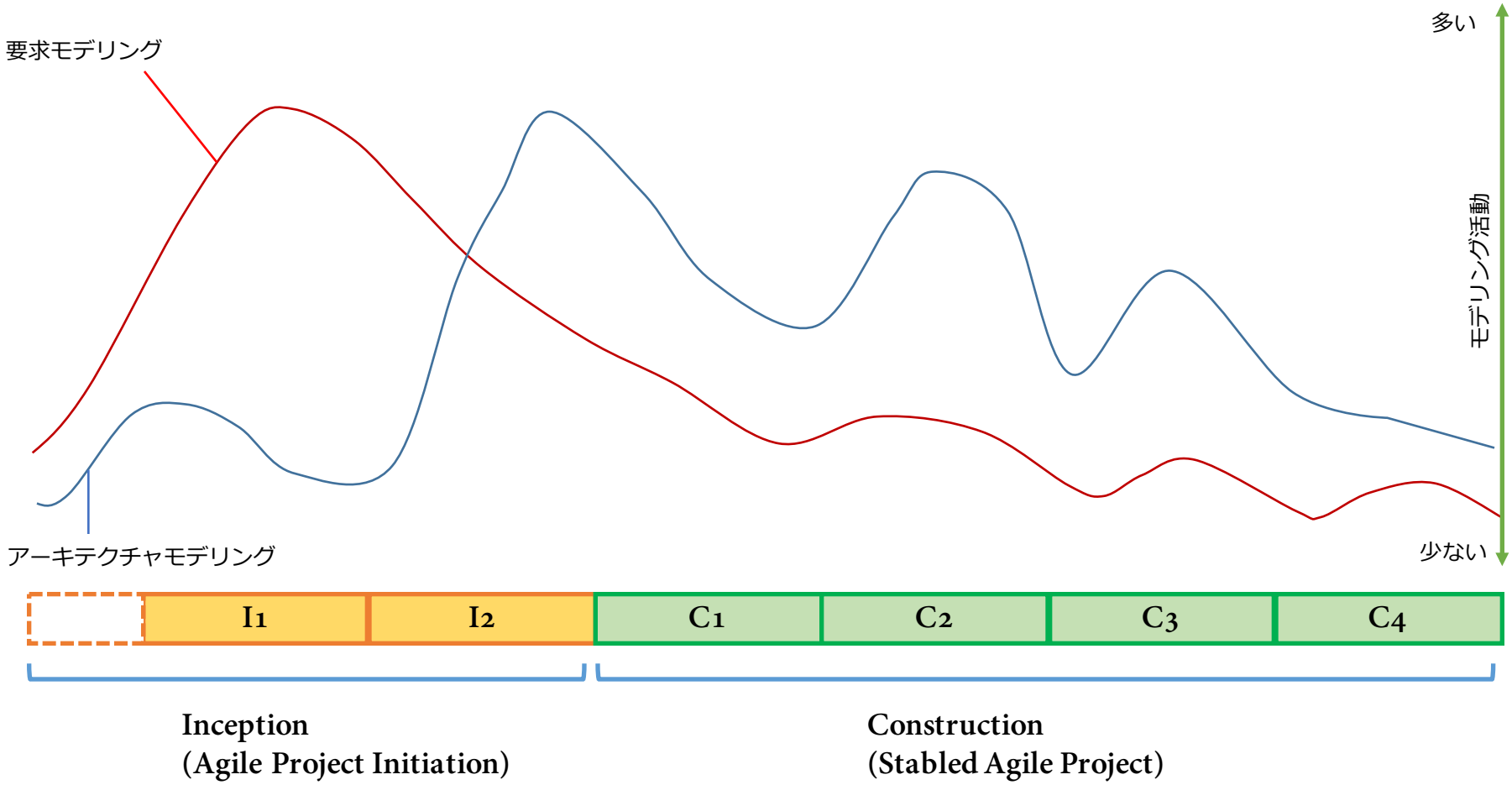
DAD方向付けフェーズのモデリング

- 初期の要求モデリング
  - IRM：Initial Requirement Modeling
- 初期のアーキテクチャモデリング
  - IAM：Initial Architecture Modeling
- 初期のリリース計画（IRP）
  - IRP：Initial Release Planning
    - →またの機会に紹介します

- 目的は「Scrumを安定して運用する」
  - 安定したプロダクトバックログ運用
  - 安定したスプリント計画
  - 安定した実装
  - 安定したリリース
  - 安定した見積り

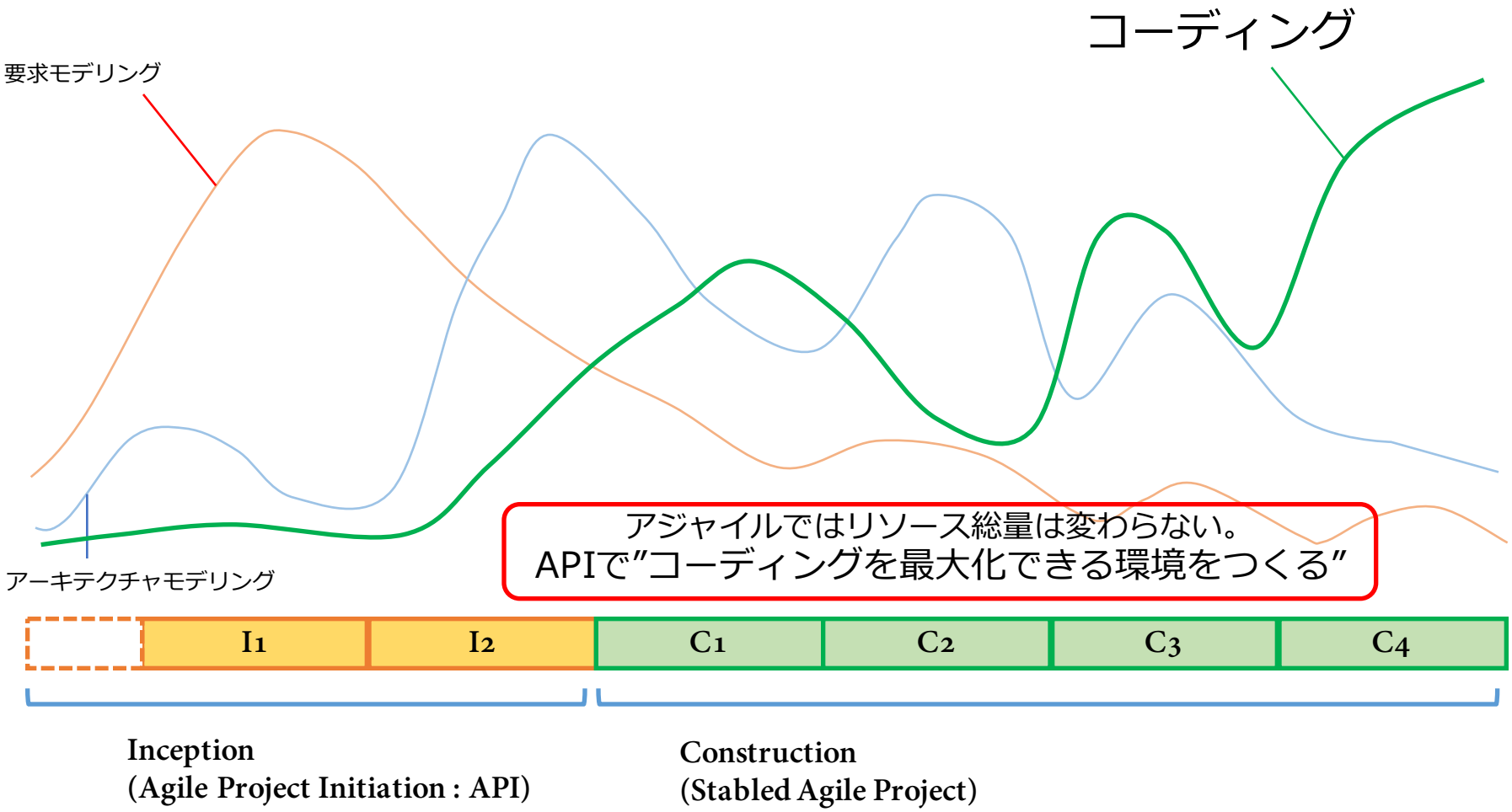


アジャイルプロジェクト立ち上げ後のモデリング活動の推移



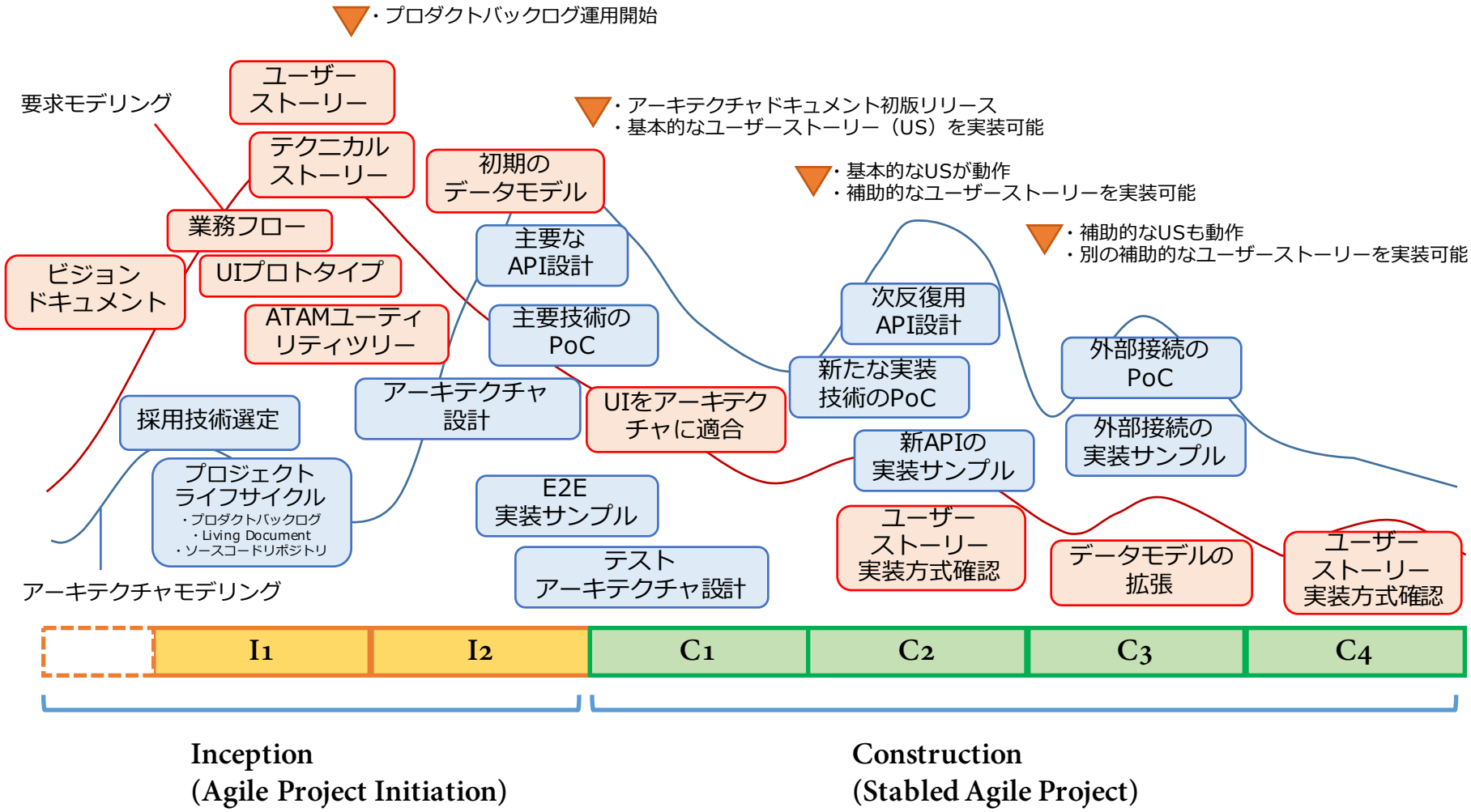
注）本チャートはゼンアーキテクツ実施の事例による推測です

アジャイルプロジェクト立ち上げ後のモデリング活動の推移



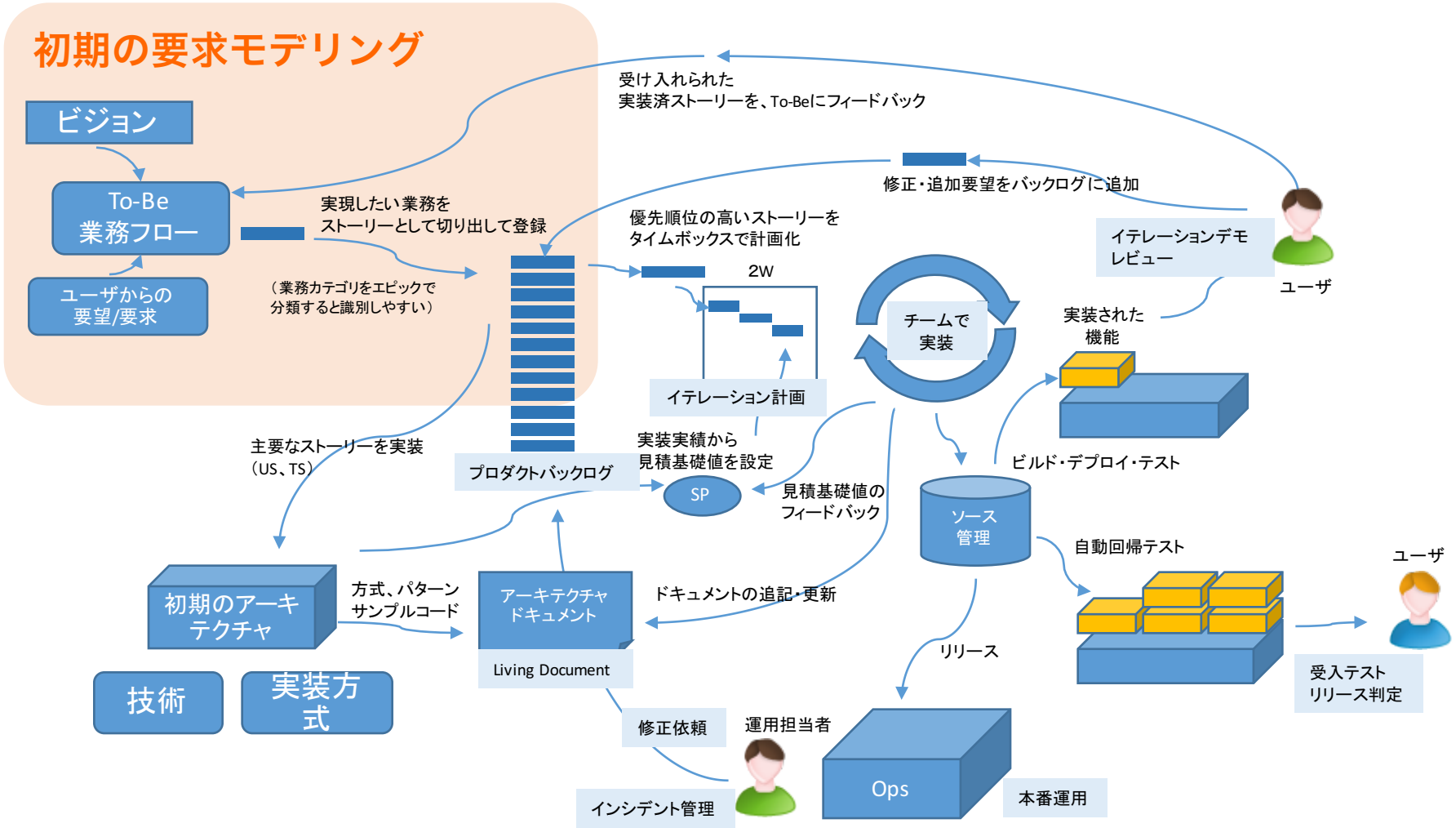
注）本チャートはゼンアーキテクツ実施の事例による推測です

アジャイルプロジェクト立ち上げ後のモデリング活動の推移



注）本チャートはゼンアーキテクツ実施の事例による推測です

IRM：初期の要求モデリング



IRMで利用可能なモデル（DADより）

- ビジネスプロセス図
- データフロー図
- ビジネスルール
- 制約事項
- コンテキスト図
- ドメインモデル
- エピック
- 機能一覧
- フローチャート
- マインドマップ
- 非機能要求（NFR）
- ペルソナ
- 要求事項
- UIフロー図
- UIプロトタイプ
- UI仕様書
- UMLアクティビティ図
- UMLユースケース図
- ユースケース仕様書
- 利用シナリオ
- 状態遷移図
- ユーザーストーリー
- バリューストリームマップ

23種類のモデルを“適切に組み合わせて”活用し、
初期の要求モデルを構築する

IRMで利用可能なモデル（DADより）

- ビジネスプロセス図
- データフロー図
- ビジネスルール
- 制約事項
- コンテキスト図
- ドメインモデル
- エピック
- 機能一覧
- フローチャート
- マインドマップ
- 非機能要求（NFR）
- ペルソナ
- 要求事項
- UIフロー図
- UIプロトタイプ
- UI仕様書
- UMLアクティビティ図
- UMLユースケース図
- ユースケース仕様書
- 利用シナリオ
- 状態遷移図
- ユーザーストーリー
- バリューストリームマップ

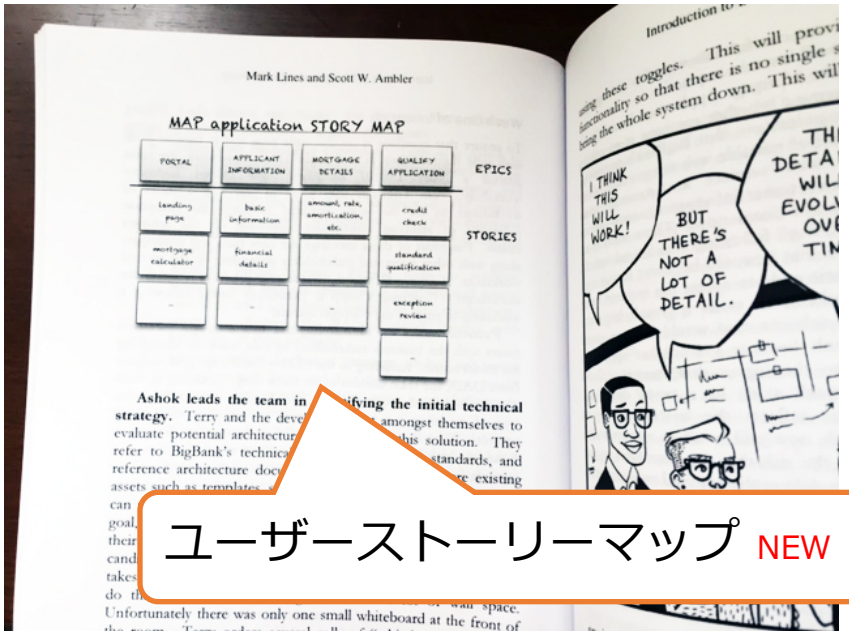
ATAMユーティリティツリー

ユーザーストーリーマップ NEW

でも23種類にはこだわらない。
目の前の要求を適切に扱うために、”適切な道具”でモデリングする。

ZENARCHITECTS

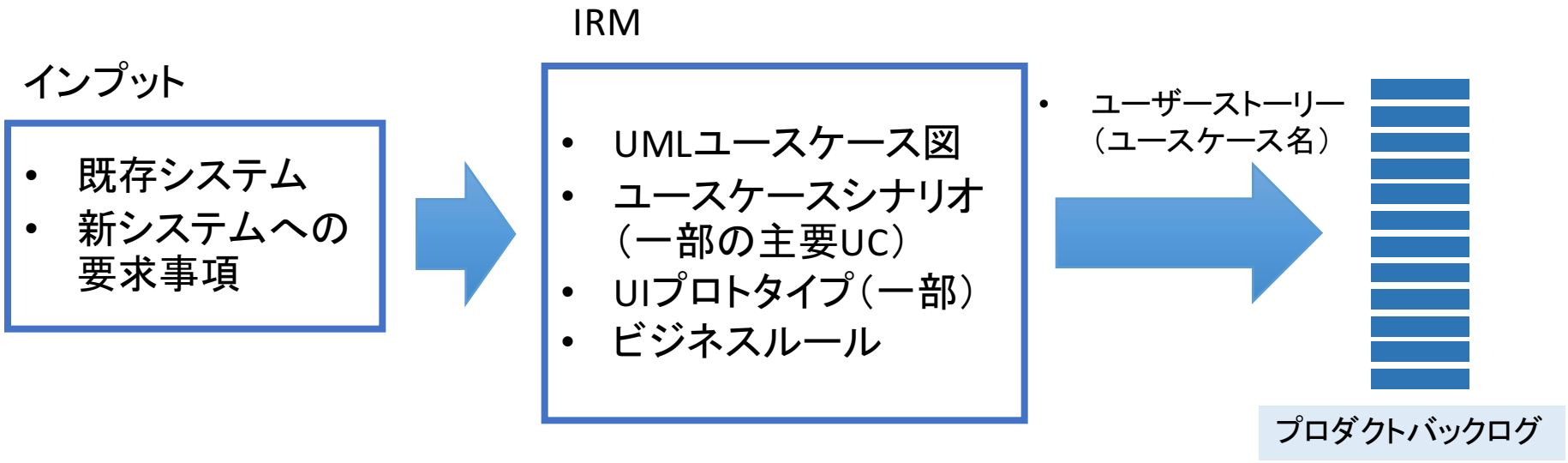
DAD新刊での
ユーザーストーリーマップ活用例



DAD新刊：Introduction to Disciplined Agile Delivery

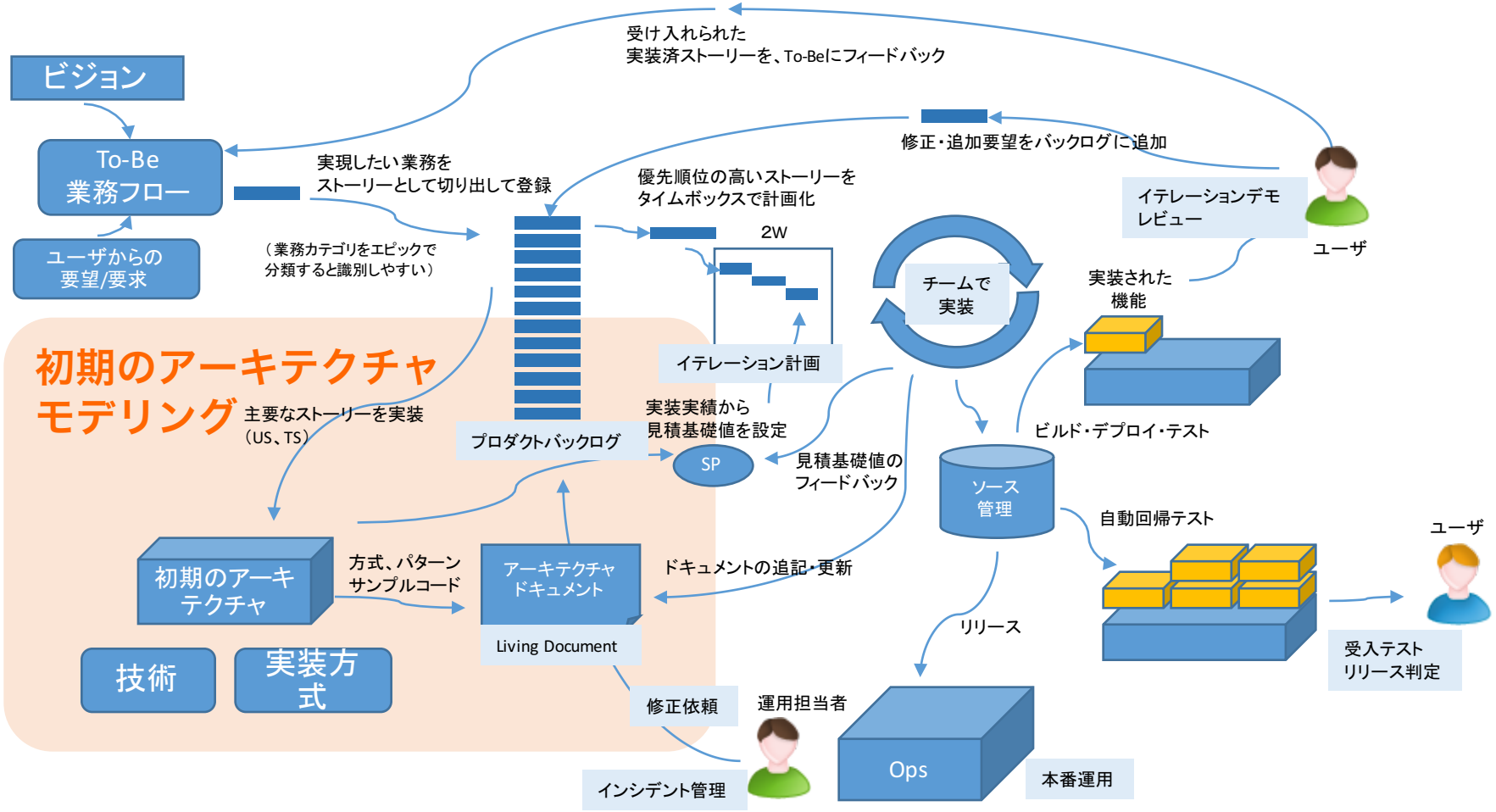
【事例】企業システム刷新時のIRM

- 新技術による次世代製造管理システムの開発



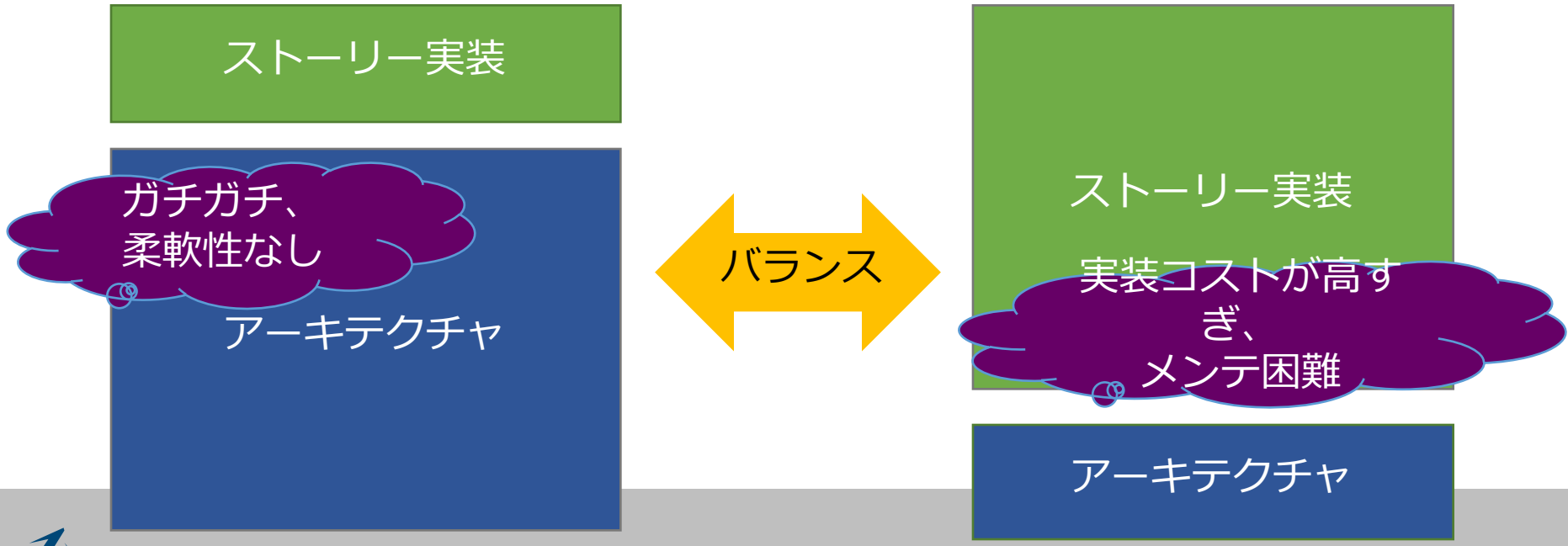
規模が大きく（数百機能）、既存システムが存在するため、UMLユースケース図でアクター観点で全体像の整理。粒度調整のため一部をシナリオ化。
ユースケース名をストーリーとしてプロダクトバックログに登録し、イテレーション内で必要に応じて（JITで）詳細化する。

IAM：初期のアーキテクチャモデリング



IAMが早期安定化のカギ

- 適切なアーキテクチャ
  - 偶発的アーキテクチャ→　創発的アーキテクチャ
- 適切なストーリー実装
  - INVESTなストーリーをそのままシンプルに実装できる
    - 抽象度の高いドメインレベルAPI
- ちょうど良いバランスを「作る」



適切なアーキテクチャをつくる

1. アーキテクチャ要求を識別・収集
   - 品質特性シナリオ（主たる機能＋非機能）
   - 品質モデル（ISO/IEC25010等）での網羅性検証

2. アーキテクチャ設計
   - 利用可能かつ適切な技術を組み合わせる
   - 自己検証（モデル、実装）

3. 客観的な妥当性検証
   - 第三者によるアーキテクチャの妥当性検証（実現性、リスク、コスト、選定技術が適切性、ストーリーの実装しやすさ）
   - ATAM(Architecture Trade-Off Analysis)によるトレードオフ分析等を活用する

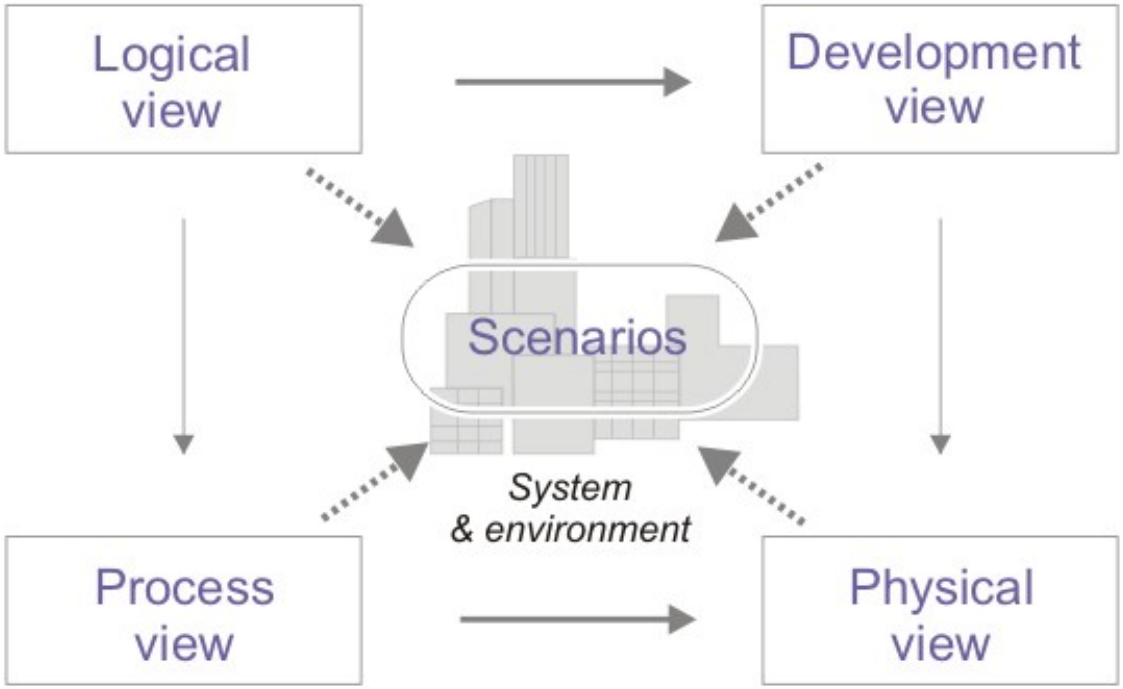
アーキテクチャを後から大きく改修するのは困難

アジャイルでも変わらない

ZENARCHITECTS

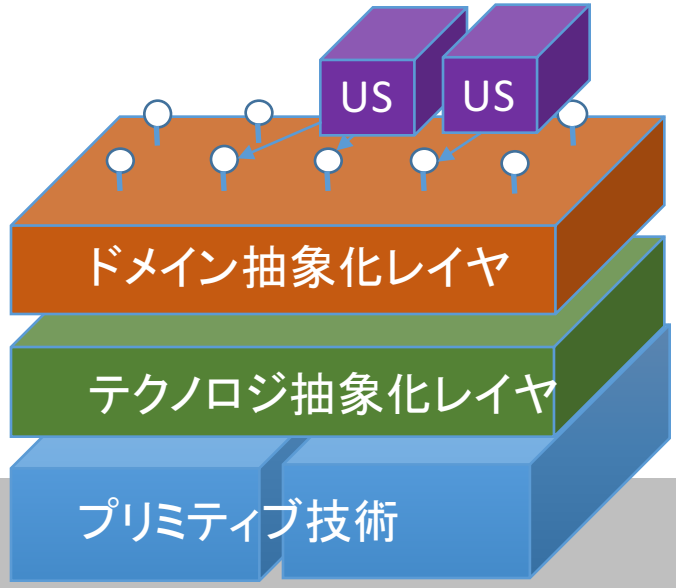
Architecture Description

- 4+1 View
- 静的ビューをシナリオで「振る舞わせる」= 自己検証モデル



アーキテクチャで
ストーリーの実装を安定させる

- 複雑さの隠蔽
- パターンの提供
- 新規性の最小化
- ドメインの言葉で実装したい（OOAD/DDD）
  - ”ユーザーストーリーがそのまま実装できるアーキテクチャ”
- シンプルなコード
  - 結果「コードの共同所有」を促進



アーキテクチャをどうやって伝えるのか

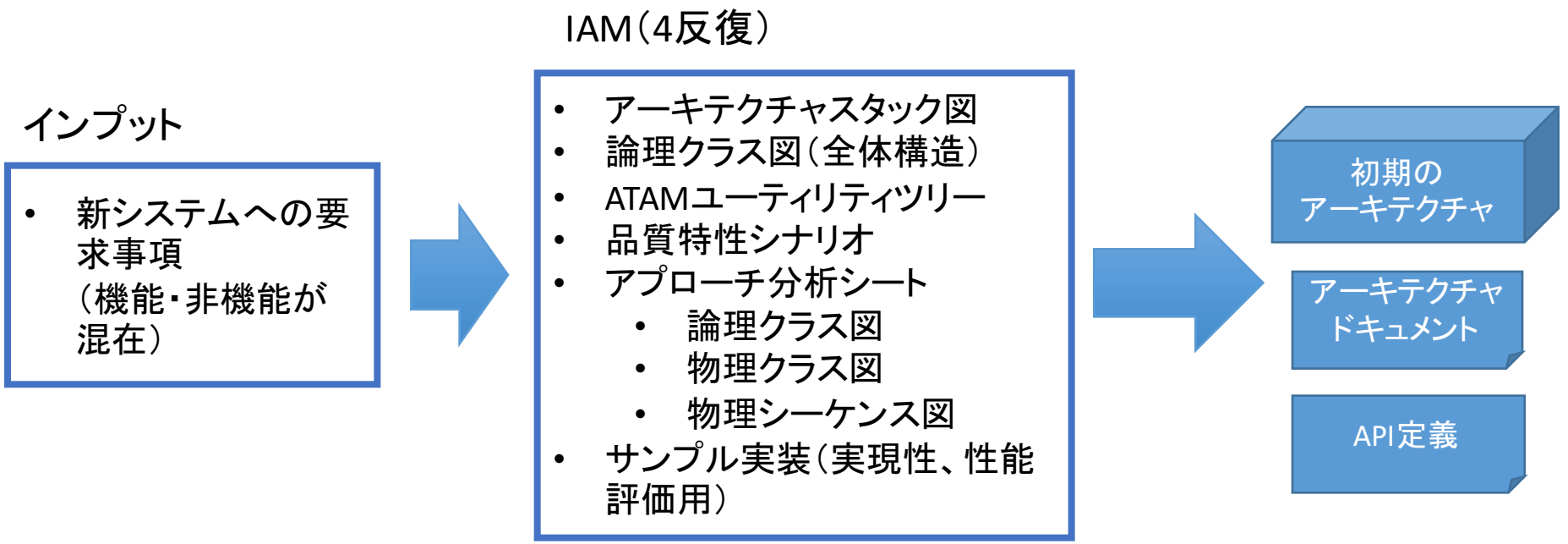
- 「アーキテクチャ設計書」と「アーキテクチャ説明書」
  - アーキテクチャ設計書（Architecture Description）は、内部の構造、振る舞い、根拠、保守の観点を示す「設計書」
  - アーキテクチャドキュメント（Architecture Document）は、アーキテクチャ（API）の使い方やサンプルを示す「説明書」

- 「しっかり書く」 or 「徐々に積み上げていく」
  - 「安定」までの戦略次第。最長6～8週。
  - 傾向として
    - アーキテクチャドキュメントと「ストーリー増加に応じて徐々に」
    - アーキテクチャ設計書は「主要なテクニカルストーリーが通ることを確認できるところまで」

- “Living Document”
  - Wikiプラットフォーム
  - 「必要に応じて、必要なことを、伝える」

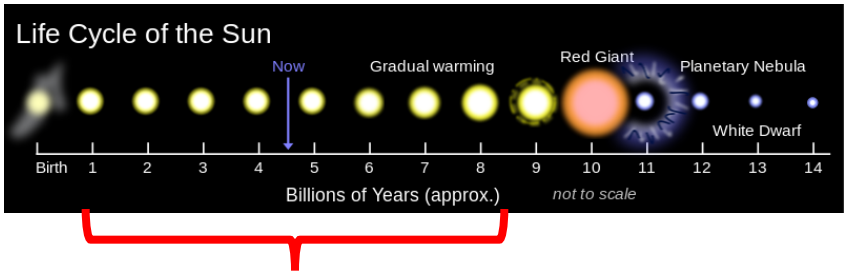
【事例】ソフトウェア製品開発でのIAM

- 新分野でのソフトウェア製品の初期開発



新分野における未知の製品開発。
要求事項の優先度と非機能のトレードオフを明確化するため、ATAMユーティリティツリーを活用。妥当性の客観的評価のためアプローチ分析シートを用いて机上・実証両面で検証。
性能・保守性・拡張性・ストーリー実装性でバランスのとれたアーキテクチャを構築。

「安定してから」のモデリング

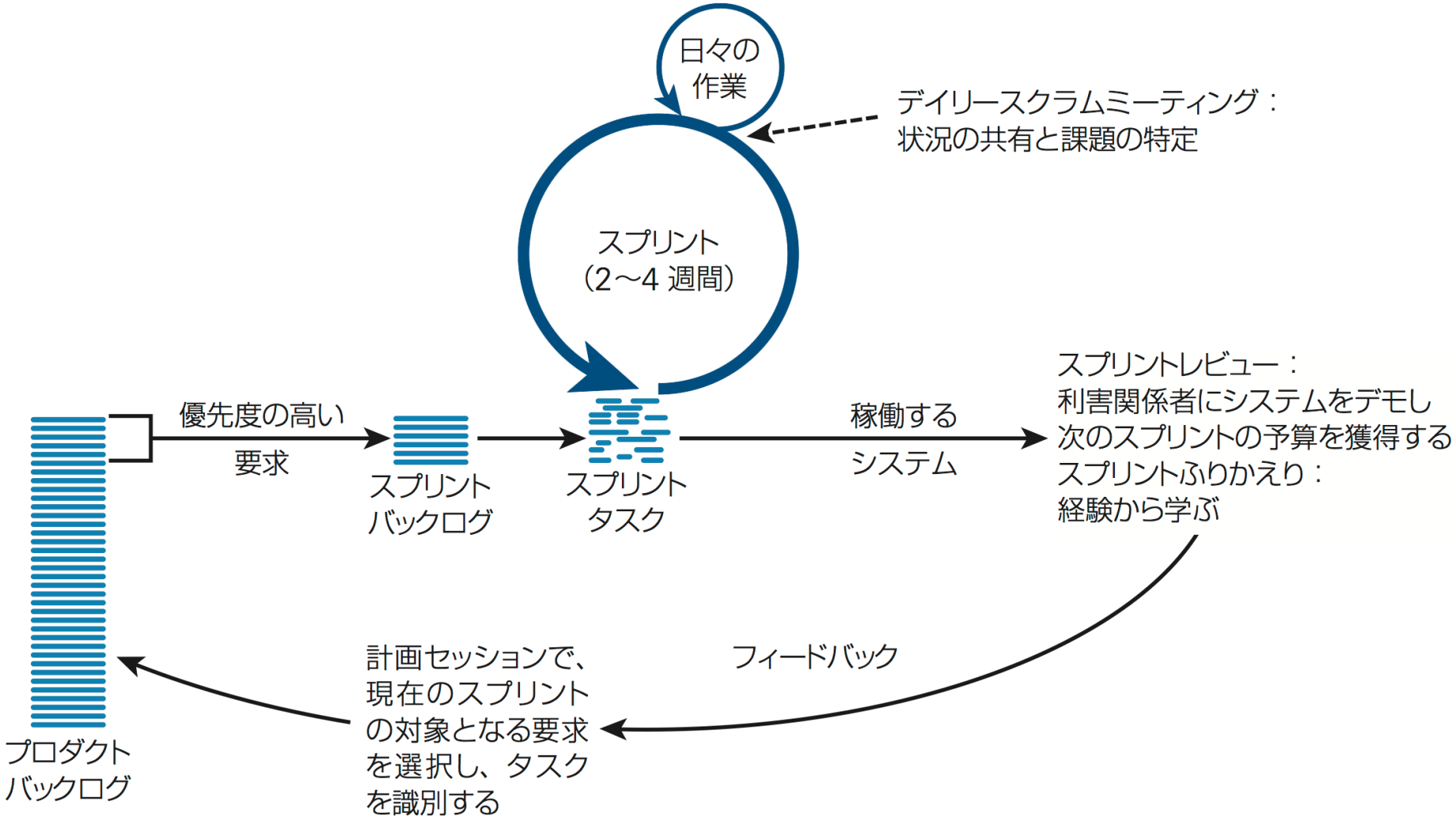


- JITモデリング
- 最小で良い
- 「コードの共同所有」と、
  それを支えるアーキテクチャの安定
- “新しいこと”を扱うときに、軽く書く
  - 新しいテーブル
  - 新しい処理方式
  - 新しいモジュール
  - 新しいタイプの要求

Image source by Wikipedia

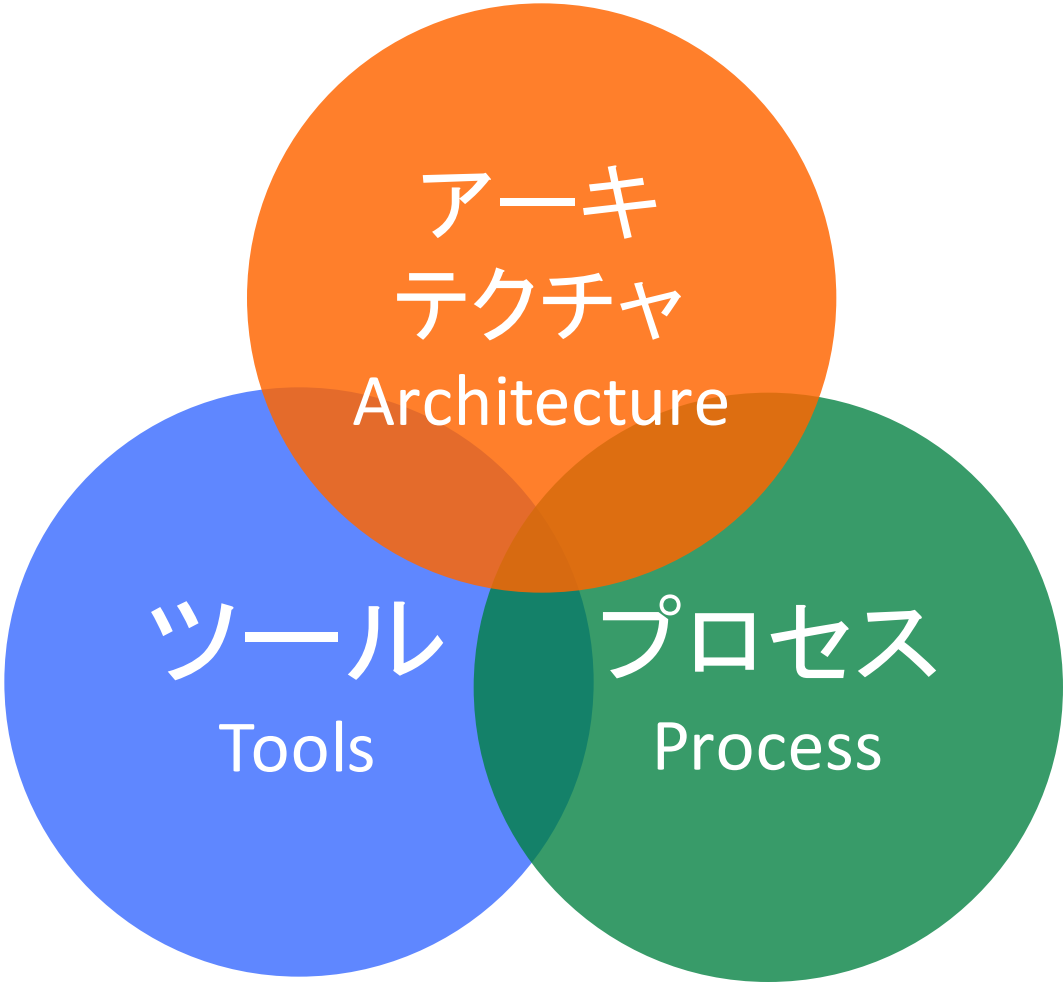
ZENARCHITECTS

スクラムの基本的な進め方

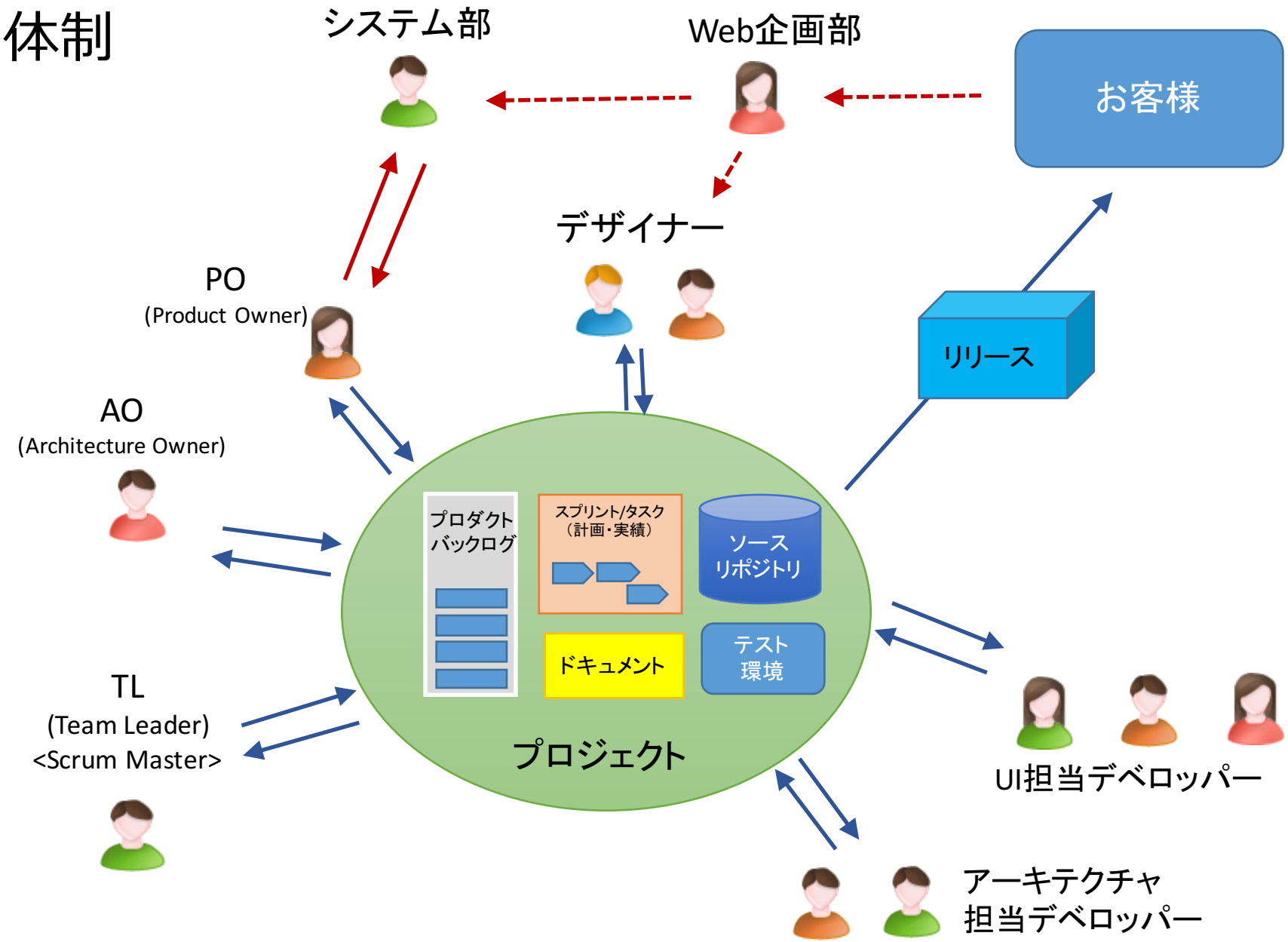


金融機関中規模チームでのDAD適用例

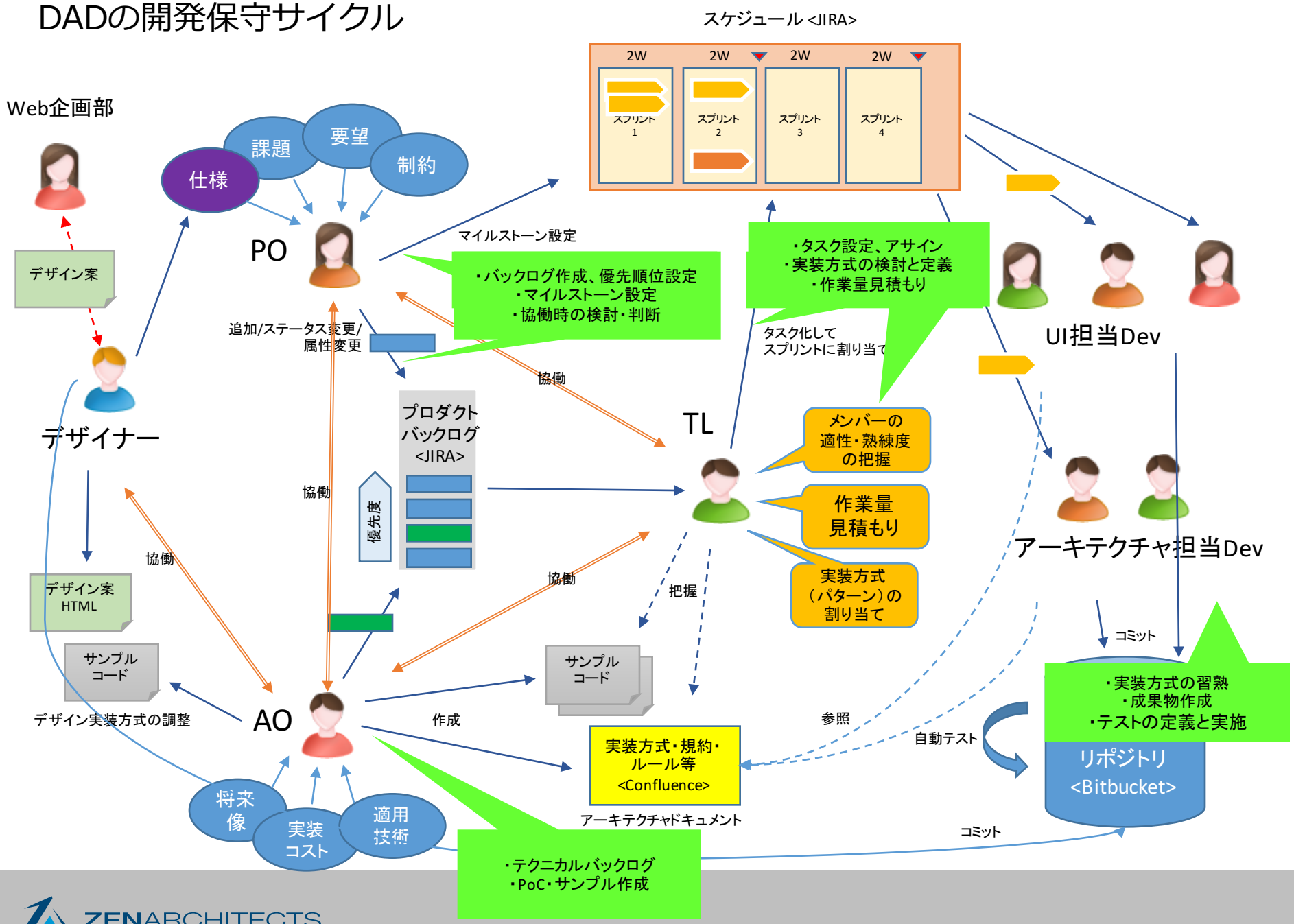
- ツール
  - JIRA http://www.atlassian.com/
  - Confluence
  - Bitbucket
  - Jenkins
  - SWAT http://www.smartekworks.com/
- アーキテクチャ
  - SPA/HTML5/knockout.js
  - MSA/PHP/Laravel/MySQL
  - RHEL
- プロセス
  - Disciplined Agile Delivery



体制



DADの開発保守サイクル



ZENARCHITECTS

【AM】アジャイル ドキュメンテーション戦略

- ✓動かない仕様よりも実行可能な仕様を選ぶ
- ✓ドキュメントを要求として扱う
- ✓ドキュメントは情報を整理する場であり、思索にふける場ではない
- ✓簡潔さを保つ
- ✓ドキュメントは、あなたの置かれた状況に合わせて十分なものを作ればよい
- ✓情報を捕まえる「よりよいやり方」を懸命になって探しなさい
- ✓ドキュメントを書く正当な理由を検討しなさい
  - ✓プロジェクトの利害関係者がそれを求めた
  - ✓APIを定義する
  - ✓外部のグループとコミュニケーションを円滑にする

【AM】現実に「UMLをどう使うか」

1. UMLをモデリングの中核として使う(もちろん全てではない)
2. 表記法の重要な部分だけを採用する
   (全て使おうとしない。重要な20%から取りかかる)
3. すべての開発者にUMLの教育を行う
   (コミュニケーションのための当たり前の道具)

UML推進者が注意すべきこと

「～にUMLをどう適用するか」ではなく「ここで～をモデリング(分析や設計)を活用するにはどうすればいいか」と考えた方が良い。

【アジャイルモデリング 15.5 より引用】

つまり「アジャイルにUMLをどう適用するか」ではなく「アジャイルでどうモデリングを活用するか (そのうちのどこでUMLが適切なのか)」と考えていきたい。

ZENARCHITECTS

zenarchitects.co.jp

facebook.com/zenarchitects