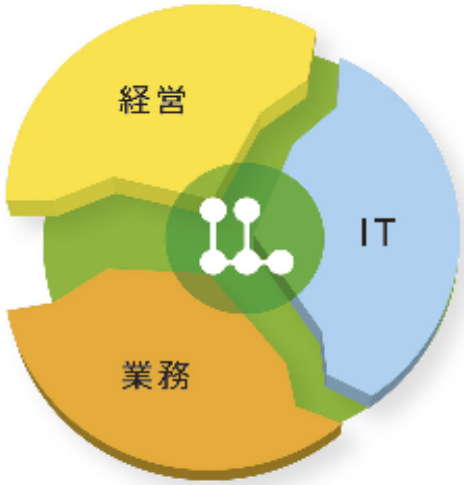
基幹系システム開発における
アジャイル開発手法適用事例のご紹介

2010/3/5

ウルシステムズ株式会社
http://www.ulsystems.co.jp
mailto:info@ulsystems.co.jp
Tel: 03-6220-1420 Fax: 03-6220-1402

ウルシステムズについて

ウルシステムズは、ビジネスとITのギャップを埋め
お客様のビジネスを成功に導くことをミッションとする
コンサルティングカンパニーです



UL Systems, Inc.

- 社名 ウルシステムズ株式会社 (UL Systems, Inc.)
- 所在地 〒104-6014 東京都中央区晴海 1-8-10
  トリトンスクエア タワーX 14階
- 設立 2000年7月25日
- 代表者 代表取締役社長 漆原 茂
- 資本金 8億1,020万円(2008年3月31日現在)

主な取引先

SONY
MITSUBISHI 三菱マテリアル株式会社
KDDI
Benesse
エネルギー・フロンティア TOKYO GAS
TOHO TITANIUM
Microsoft
時事通信社
ANA
hp invent

ウルシステムズのコンサルティング
～お客様側からギャップを埋める～

GAPS
UL Systems, Inc.

ウルシステムズは、ビジネスを成功に導くシステム開発がうまくいかない原因は
「ビジネスとITのギャップ」にあると考えております。

ゴール（目的）のギャップ	経営目的や狙いが開発側に正しく伝わっていない
アクティビティ（業務）のギャップ	業務上必要な処理内容が開発側に正しく伝わっていない
プロセス（工程）のギャップ	プロジェクト管理が破綻している
スキル（技術）のギャップ	開発体制内に必要な技術力が不足している

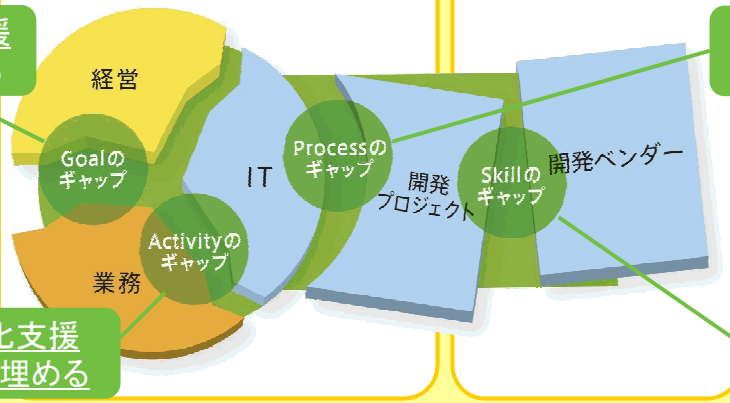
ウルシステムズは、お客様の立場から「ビジネスとITのギャップ」を埋める
4つのコンサルティングサービスをご提供します。

これら「ビジネスとITのギャップ」は発注側の課題であり、開発ベンダーだけでは解決できません。発注側であるお客様ご自身が主体となってギャップを埋めていくことが必要不可欠です。ウルシステムズは、お客様の立場からギャップを埋めるご支援をいたします。

4つのコンサルティングサービス

ビジネスイノベーション支援
ゴールのギャップを埋める

プロジェクトマネージメント支援
プロセスのギャップを埋める



業務オペレーション最適化支援
アクティビティのギャップを埋める

先端IT導入
スキルのギャップを埋める

ギャップを埋める4つのコンサルティングサービス

GAPS
UL Systems, Inc.

【ビジネスイノベーション支援】
ゴール（目的）のギャップを埋める

お客様のビジネス革新を実現するための事業戦略と、それを支えるIT戦略の立案を行います。ビジネスの目的（Goal）とシステム化計画との不整合によるギャップを埋め、ビジネスを成功に導きます。

- 新規事業立案
- システム化計画立案
- 会社ITグランドデザイン

【業務オペレーション最適化支援】
アクティビティ（業務）のギャップを埋める

お客様の事業戦略を確実に実現する業務要件を策定します。業務（Activity）の理解不足によるシステム要件とのギャップを埋め、ビジネスの目的を達成する適切な業務オペレーションを実現します。

- ビジネスモデリング
- 業務分析・設計
- 要件定義

【プロジェクトマネージメント支援】
プロセス（工程）のギャップを埋める

お客様ご自身のプロジェクトマネージメント力を強化することで、ベンダー依存から脱却したお客様主導でのプロジェクト遂行を支援します。システム開発工程（Process）の不手際から生じるギャップを埋め、予定通りのコスト・期間・品質でのプロジェクト遂行を可能にします。

- プロジェクト管理支援
- RFP作成・ベンダー選定支援
- 開発プロセス標準化
- システム要員育成

【先端IT導入】
スキル（技術）のギャップを埋める

最先端のIT技術を活用したシステム構築を支援します。開発ベンダーに必要なスキル（Skill）の不足によるギャップを埋め、要求された機能や性能を十分満たすシステム開発を可能にします。

- システムディベロップメント
- アーキテクチャ設計・構築
- 開発環境の整備

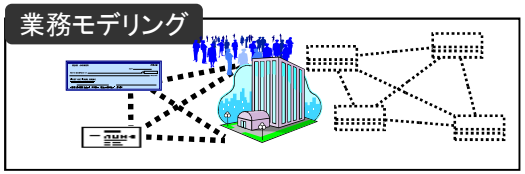
事例：百貨店様
基幹システム再構築プロジェクト推進支援

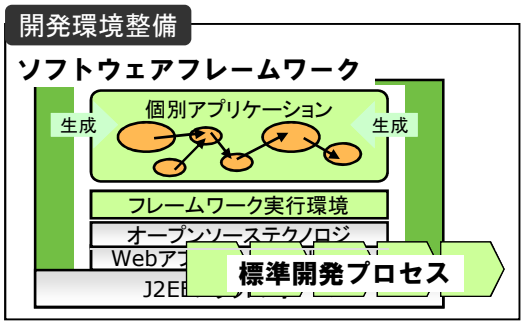
GAPS
UL Systems, Inc.

情報システム部門の技術力向上を支援、お客様主体の基幹システム再構築を推進

背景と課題

- お客様は、首都圏に多数の店舗を持つ大手百貨店の情報システム部門。分社化による企業構造の変化に伴い流通フロー変更が発生するなどビジネスが急激に変化し始めていた。
- ビジネスの変化に対応できる生産性・保守性の高い基幹システムが求められていたが、基幹系再構築は大規模開発となるため開発費の抑制が必須であり、ダウンサイジングにより運用コスト削減を実現する必要があった。
- お客様は汎用機の利用が長く、社員・開発パートナーともJavaをはじめとするオープン系技術の経験が不足しており、効果的な教育や技術支援が必要とされていた。





アプローチ

- 業務モデリングによるシステム設計
  - 業務を適切にモデリングすることで、ビジネスの変化に強いシステムを設計。
- 標準化・フレームワークを利用した開発環境の整備
  - 生産性を高めるため、ソフトウェアフレームワークを導入。あらかじめ準備されているソフトウエア部品の利用により開発範囲を限定し、開発費を抑制。
  - お客様自身によるシステム開発を目的として開発プロセスを標準化、状況にフィットする開発環境を整備。
- 技術教育と安定した開発体制の確立
  - 大規模開発をお客様側自身で運営するためのフレームワークチーム体制の確立と技術移管を実施。

導入効果

- 基幹システムは無事カットオーバーを迎え、その後安定稼動中。今後、お客様は今回の基幹システムで培った技術をベースに、他事業を支える顧客系システムなど主要基幹システムを順次刷新していく計画である。
- 開発者100名を超える大規模プロジェクトという極めて困難な状況下において、所定のシステムを当初計画通りカットオーバーさせることに大きく貢献。開発の生産性・保守性を向上させることでビジネス環境変化への対応能力が高まった。
- システムの稼働だけでなく、開発プロセスの標準化、ソフトウェアフレームワークの導入、教育による定着といった施策により、社内の他のプロジェクトにも展開可能な基盤となるソフトウエア資産の蓄積と開発ノウハウの共有を短期間で実現できた。

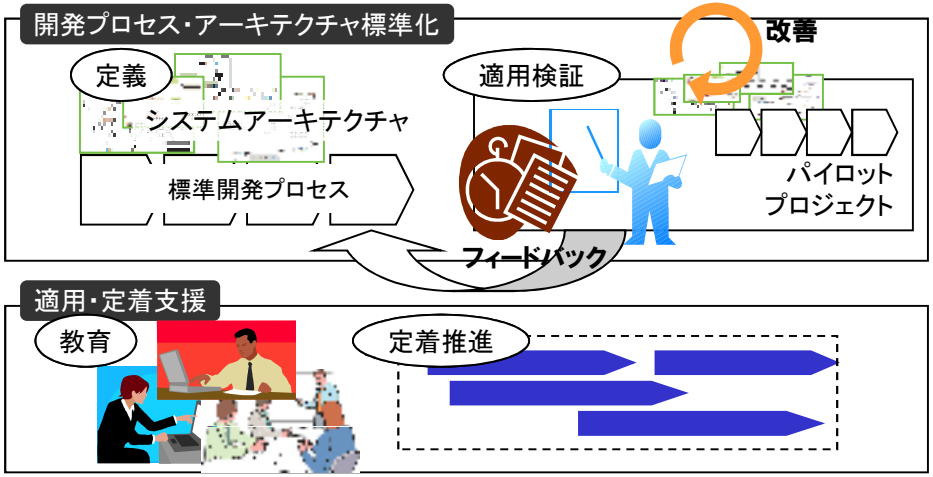
事例：大手自動車メーカ様
開発プロセス導入支援

GAPS
UL Systems, Inc.

オブジェクト指向に基づく開発プロセスの定義・導入により、システム開発効率の向上を実現

背景と課題

- お客様は、デザイン性、先進技術、生産能力の点で国内外の評価が高い国内大手自動車メーカー。汎用機で構築された既存のシステム基盤は、柔軟性に欠けており、競争力確保のために、オープン系技術を使ったITシステム基盤の再構築が急務であった。
- システム開発はSIベンダに委託していたが、生産性や品質が期待どおりに向上しておらず、保守コストが徐々に増大してきていた。



アプローチ

- 開発プロセスとシステムアーキテクチャの標準化
  - 生産性と品質を向上させるために、要件定義からシステム開発までの開発プロセスを定義し、適用を検証。
  - オープン系技術を利用したシステムアーキテクチャを定義。
- 開発プロジェクトにおけるシステムアーキテクチャ活用と、開発プロセス定着の支援
  - 開発委託先のSIベンダに対し、システムアーキテクチャを活用するための技術支援と開発プロセス定着のための教育を実施。

導入効果

- 約10プロジェクトに開発プロセスを適用し、効果を測定した結果、平均20%程度の開発効率向上が確認された（40人月～30人月規模）。
- 特に上流工程での効果が高く、プロジェクト間におけるドキュメントの再利用などが頻繁に実施される状態となった。

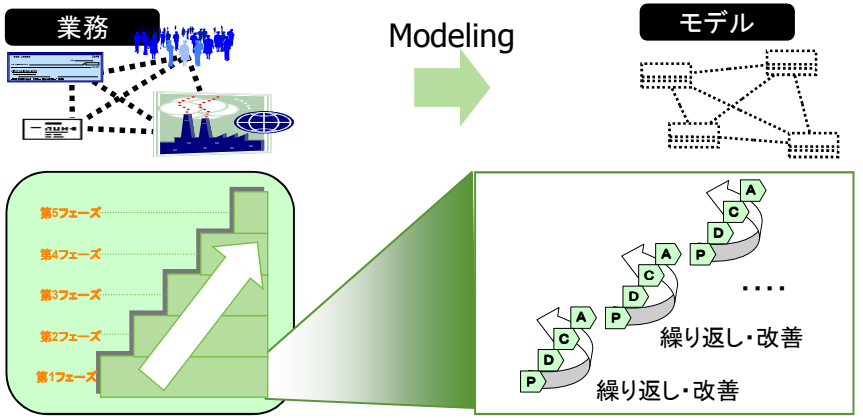
事例：大手製造業様
SCMシステム再構築支援

GAPS
UL Systems, Inc.

ビジネスモデリング＆アジャイル開発支援によりSCM改革プロジェクトを推進

背景と課題

- お客様は、複数の事業カンパニーやグループ会社を持つ大手製造業。各事業が市場環境の変化に適応し、競争力をさらに高めるためにはSCM改革を推し進める必要があった。
- 事業部門の要求を情報システム部門がしっかりと把握して合意形成できないままに開発し、満足が得られないことが過去に何度かあった。
- 従来の開発の進め方ではシステムが利用可能になるまでに時間がかかり過ぎ、経営陣や事業部門からはもっと早くビジネス上の成果を評価できねばならないと強く要望されていた



システム分割・段階的リリース　　短期間のPDCAサイクル

アプローチ

- ビジネスモデリングの実施
  - 事業部門の要求を正確に把握し、共通認識を形成するためにUMLで記述した視覚的なモデルをベースに要求を検討。ビジネスの「めざす姿」を明確化する。
- アジャイル開発の適用
  - ビジネスモデルの妥当性を実際に動くソフトウェアで評価することを目的としてアジャイル開発を採用。
  - 1ヶ月単位の短いPDCAサイクルを回し、机上の検討だけでは得がたい現場ユーザからのフィードバックを素早く取り入れ、システムの継続的な改善を図る。
- お客様独自の開発プロセスの定着化
  - 頻繁なフィードバックによるお客様独自の開発プロセスの進化、OJTによるメンバー教育。

導入効果

- めざす姿を3ヶ月で作成した後に、1ヶ月単位でシステム開発サイクル（イテレーション）を回し、現場に段階的に機能をリリースした。早い時期に本番でシステムを利用できたことにより、現場の参画意欲が急速に高まり、ビジネスモデリングワークショップでの議論の質も向上した
- 弊社のビジネスモデリング方法論を、お客様の環境や風土・文化、SCM改革プロジェクトのミッション、アジャイル開発との適合性などさまざまな要因を考慮した上でカスタマイズし、継続的に評価・改善して適用したので、プロジェクトチームの時々の状況に応じて最適な作業を実施することができた。
- 最初に着手したプロジェクトでの成果が認められたため、別事業部のSCM改革プロジェクトも同じスタイルで実施することになった。

アジャイル開発事例のご紹介

システム開発を取り巻く環境と要件定義への影響

UL Systems, Inc.

	システム開発の環境変化	要件定義への影響
経営層	IT投資にも設備投資と同等の効果を求める	・ビジネス環境の変化にも対応し、価値を提供できるシステムの構築が求められる
	システム開発の短納期・低コストの要求	・要求定義フェーズにかけられる時間が減少し、十分な検討が行えない
ユーザ部門	システム領域拡大によるプロジェクトステークホルダーの増加	・ステークホルダー間の利害調整に時間がかかり、結論がでない
	属人的業務の単なるシステム化による要求の増大	・ユーザ部門の力が強く必要な機能がどんどん膨らんでいってしまう
情報システム部門	システム関連業務のアウトソーシング化	・業務とITの両方に精通した人材が不足し、業務からシステム機能への翻訳がうまくいかない
	技術の専門性の増加	・システム構築技法の専門性が高まりすぎていて、何を選択したらよいかわからない

要件定義手法を含めた開発手法の見直しが急務

ビジネスを可視化し、要件の土台となる知識をステークホルダーで共有する

要件をビジネスからシステム機能まで構造化して整理する

抽出した要件を短いサイクルで検証し、要件の質を向上させ変化に対応する

アジャイル開発手法の適用

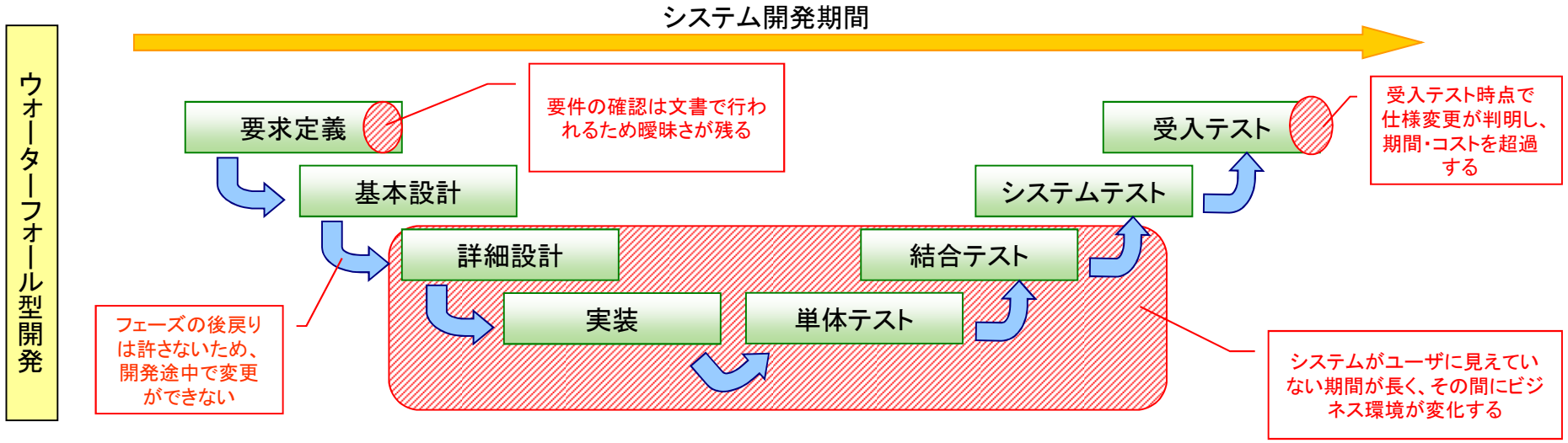
開発手法の分類と概要

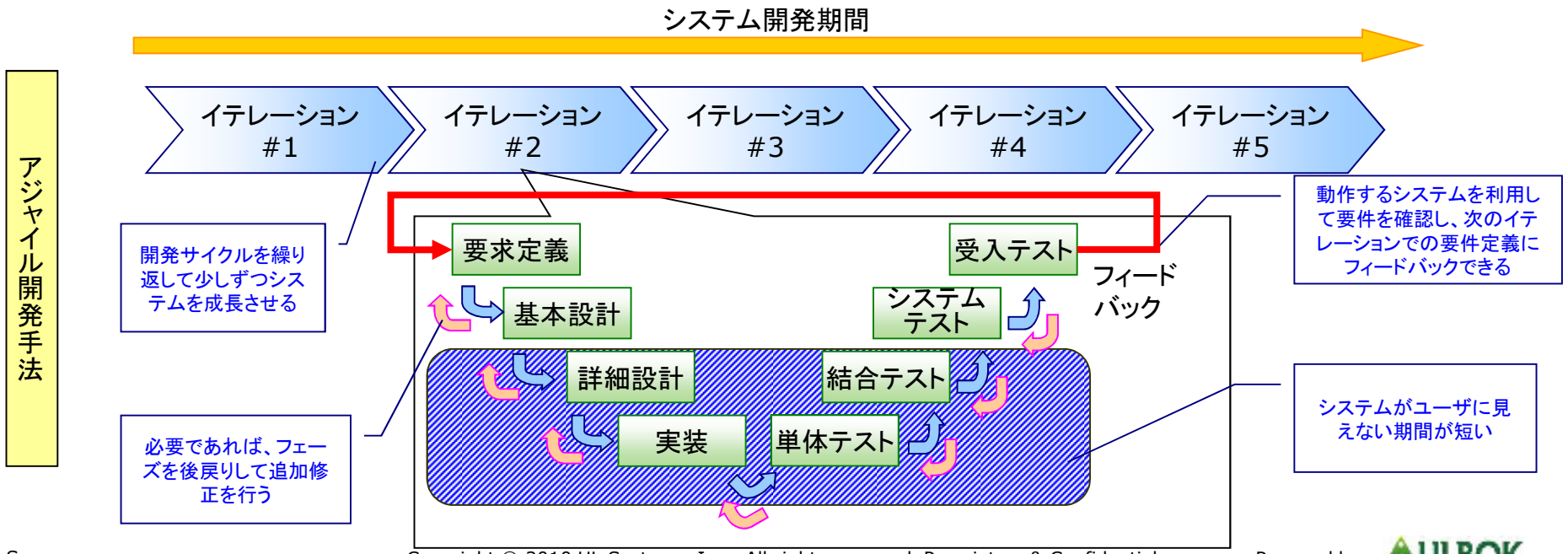
- 開発手法は以下のように分類することができます。

開発へのアプローチ	開発手法・方法論の分類	重視するポイント・前提条件	具体的な開発手法の例
スクラッチ開発（計画型 ↑ ↓ 創発型）	**ウォーターフォール**	開発に必要な要求定義～テストの各作業を予めプロセスとして定め計画的に実行してシステムを構築する。各プロセスの成果物・検証・評価の定義と標準化を重視する。	■ ベンダー標準開発手法
	バランス型	プロジェクトの初期段階で、対象とするシステムの全体像を描き、全体計画を立案する。その上で、実際のシステム構築は2週間～1ヶ月程度の短いイテレーションを単位に行う。	■ RUP ■ FDD
	アジャイル開発	要件を満たす最小限の成果物をごく短期間で提供し続け要求を創発しながらシステムを構築する。ユーザーを含む全参画者のコミュニケーションと暗黙知を重視し、高いスキルと士気を前提とする。	■ XP ■ Crystal ■ ASD ■ Scrum
試作利用	**プロトタイピング**	採用技術と要件との一致を計画時点で見極めるのが難しいという前提で、技術検証や設計方針確定を目的(※)に、試作品構築後にシステムを構築する。技術リスク低減を重視する。	（ー）

※.画面モック作成などによる要件明確化を目的としたプロトタイピングは「ウォーターフォール型」に分類する。

ウォーターフォール開発手法とアジャイル開発手法の比較





アジャイル開発の特徴と効果

アジャイル開発の特徴

くり返し型の開発

開発期間を短いサイクルに区切り、実際に動くソフトウェアを見ながら、要件の精度を向上させて行きます。

イテレーティブ（反復的）

段階的開発

システム全体を分解し、基幹となる部分や効果の高い部分から順番に作り、段階的に全体を仕上げて行きます。

インクリメンタル（暫時的）

柔軟な開発

最初に立てられた“曖昧な”計画を、走りながら検証した上で、積極的にやり方を改善し、ゴールをめざします。

創発

自律的なチーム

開発を進めるための最善策をメンバのひとりひとりが主体となったチームで自律的に判断します。

自己組織化

アジャイル開発の効果

質の高い要求を獲得し、満足度の高いシステムを構築できる

- 要求を実システムを用いて確認出来るため満足度の高いシステムが構築できる
- 重要なものから優先して開発することにより、高い価値を提供することができる
- 開発の進捗を動作するシステムを用いて確認することが可能

無駄な作業を削り、真に必要なものに集中する

- 文書化などの無駄な作業を削ることにより開発コストの削減をはかる
- 保守運用のしやすい構造を手に入れることにより保守運用作業を減少させられる
- 重要性の低い使われない機能を作る無駄を省くことができる

ビジネスの変化に迅速に対応できるスピードを身につける

- ビジネス環境の変化に合わせるように開発開始後の仕様変更にも柔軟に対応する
- 要求が固まったところからリリースできるため、早期にシステム効果を享受することができる
- PDCAサイクルを短期間で回すことでチームのパフォーマンスを向上できる

アジャイル開発によるコスト削減の考え方と実績

UL Systems,Inc.

必要なものだけを作る

ドキュメントベースでの要件定義ではユーザ側の要望により、必要な機能がどんどん膨らみ、重要度が低い機能まで作ってしまった結果、開発コストがふくらんでしまうことがある。

アジャイル開発では、動作するシステムを目にしながら、機能の優先度をイテレーション毎にユーザと調整していく。結果として、使われない無駄な機能を開発することが少なくなり、全体の開発コストを下げられる。

要求変更への対応を早期に盛り込む

ウォーターフォール型の開発では要求の欠陥の発見が開発フェーズの終盤（ユーザテスト段階）になってしまうため、やり直し作業が増えて対応コストが増大する。

アジャイル開発では要求の欠陥は各イテレーション後に実施するフィードバックにより把握することができるため、対応を迅速に行うことができ、対応コストを抑えることができる。

不要なドキュメント作成など無駄な作業を減らす

ウォーターフォール型の開発では、各フェーズにおいて成果物として様々なドキュメントを作成するが、その作成・メンテナンスに多くの工数がかかってしまう。

アジャイル開発では、動作するシステムの構築に集中するため、中間成果物などはコミュニケーションで代替する。結果として無駄なドキュメントの作成を省き、開発工数を削減できる。

チームの能力向上によるパフォーマンスアップ

従来の開発では開発終盤になるまで計画との実際の遅れが顕在化せず、能力の向上や改善活動を実施する時間がない。

アジャイル開発では、各イテレーション終了後に振り返り活動を実施することができ、開発生産性向上のための取組みを組み入れてチームの能力向上をはかることが可能。結果として開発コスト削減に寄与する。

アジャイル開発を実施した企業の65%が10%以上のコスト削減に成功

出展：VersionOne社 “The State of Agile Development”

弊社が直近で参画したプロジェクトでは納期・機能を満たした上で予算を3%下回ることに成功

プロジェクトサマリ

お客様

東邦チタニウム株式会社

（金属チタン総合メーカー）

プロジェクトの経緯

建設中の新工場で使用する生産管理システムを整備するタイミングで、コントロールセンターとしての生産管理業務の見直しや、部分最適で作成された既存システムとの連携・統合に関する検討が必要だと判断された。

プロジェクト概要

新工場や新組織におけるあるべき業務を描き出す業務改革から開始し、業務要件・システム要件の定義を経て、アジャイル開発手法を適用したシステム構築まで実施し、新工場稼動開始とともにシステムリリースを実施した。

アジャイル開発手法適用の理由

- 東邦チタニウム様では生産管理という視点でのシステム構築経験がなく、動作するシステムを見ながら要求をつめる必要があった。
- 新工場の業務はまだ確定しておらず、仕様を早期に凍結することができないと考えられた。

プロジェクトの特徴

- 業務改革検討において設定した業務改革テーマの検討時には、システム構築の視点からの検証を実施し、検討内容がシステム上実現可能であることを確認した。
- 短期間（1ヶ月）での開発を繰り返して実施することで、動作するシステムを用いてユーザに要件の確認を行い、満足度の高いシステム開発が実施できた。

プロジェクト規模

- 全体期間：2006年5月～ 2008年5月
- 開発規模

  UC数 268／画面及び帳票数 311

  生産管理（生産計画、製造指示、進捗管理）

  在庫管理、品質管理、原料管理、

  他システム（販売、検査、物量）連携
- 開発工数

  （イテレーション開始後～システムリリース：11ヶ月間）

  設計担当　：　74　人月

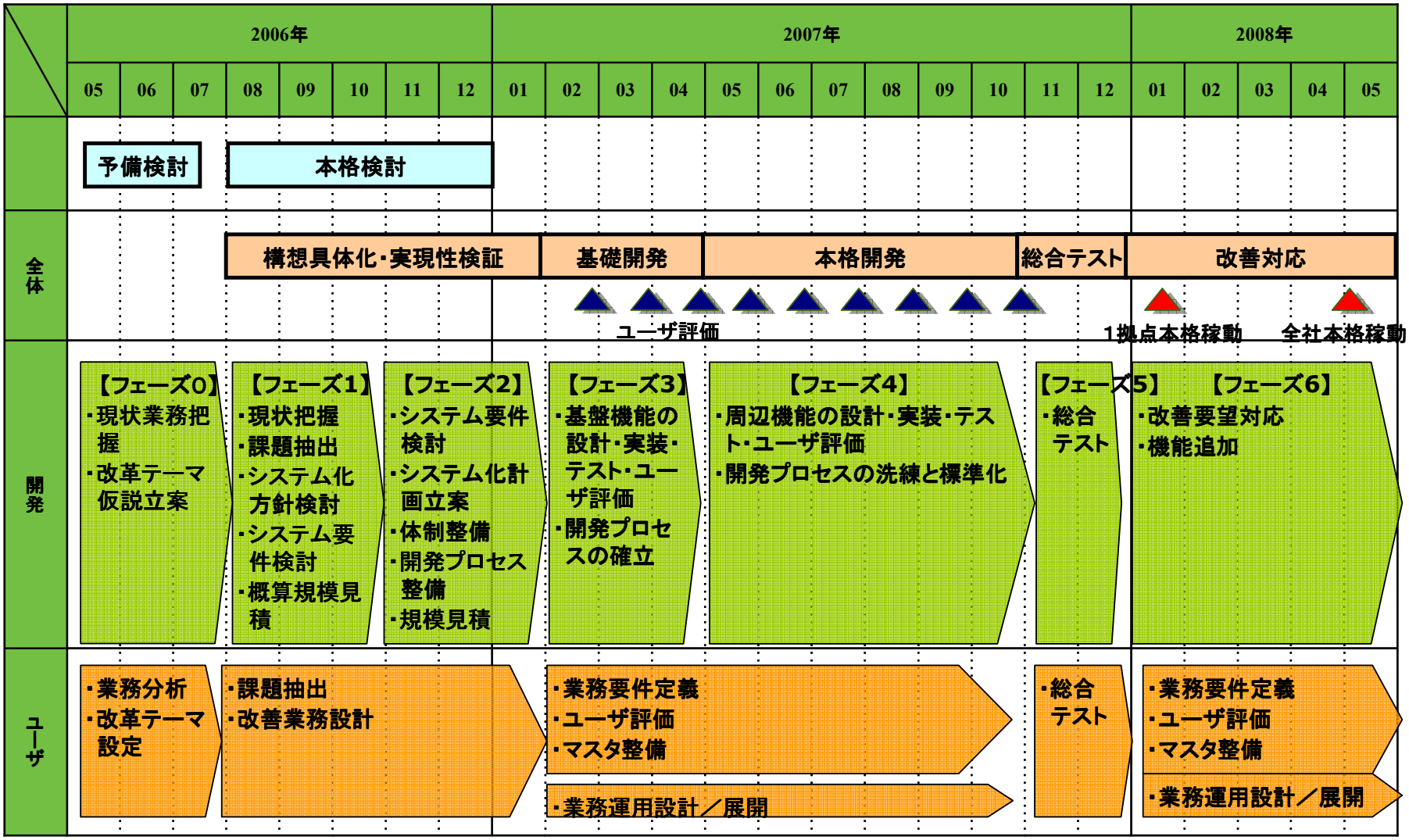
  アーキテクト：　36　人月

  開発担当　：　47．5人月

  計　：157．5人月

プロジェクト全体スケジュール

構想・計画フェーズに6ヶ月、実装フェーズに基礎開発と本格開発合わせて9ヶ月を要した



プロジェクトにおける弊社支援内容

プロジェクトの企画フェーズからシステムリリースまでプロジェクトを通じて成功に必要な支援を行った

【フェーズ1】
- 現状把握
- 課題抽出
- システム化基本方針検討
- システム要件検討
- 概算規模見積

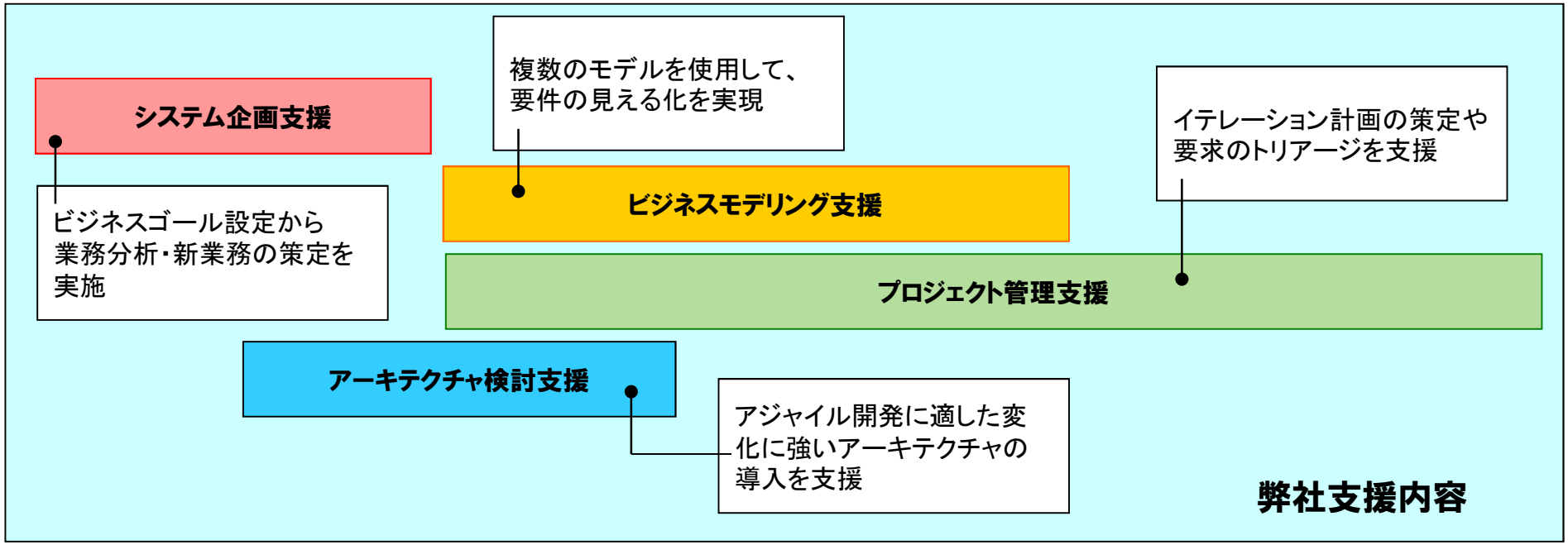
【フェーズ2】
- システム要件検討
- システム化計画立案
- 開発体制整備
- 開発プロセス整備
- 規模見積

【フェーズ3】
- 基盤機能の設計・実装・テスト・ユーザ評価
- 開発プロセスの確立

【フェーズ4】
- 周辺機能の設計・実装・テスト・ユーザ評価
- 開発プロセスの洗練と標準化

【フェーズ5】
- 総合テスト

【フェーズ6】
- 改善要望対応
- 機能追加



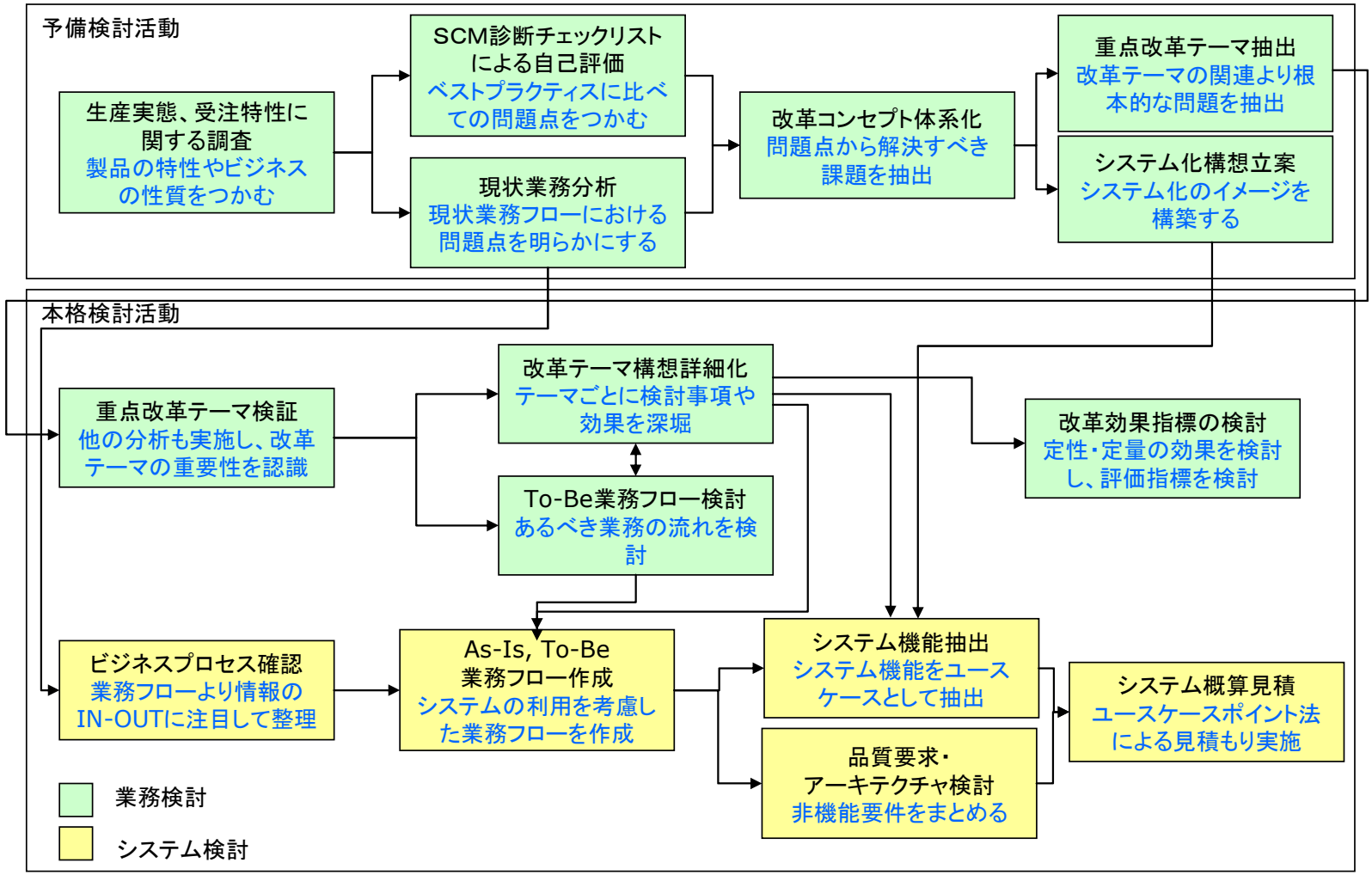
システム企画
業務検討活動からシステム検討・システム開発への流れ

ビジネス上の目標から各システム機能の要件までをトレース可能にするような活動を実施した

予備検討活動	本格検討活動	システム検討活動	システム開発
・仮説立案 ・現状業務の問題点の把握 ・業務のあるべき姿の立案 ・活動の期待効果予測	・仮説検証 ・業務のあるべき姿の合意形成 ・あるべき姿を実現するための業務プロセス定義 ・活動の期待効果試算	・システム化対象業務の決定 ・業務要件からシステム機能要件を抽出 ・システム規模の概算見積	・機能要件の詳細定義 ・システムの実装 ・短期イテレーションによる要件の確認検証
ビジネスゴールやあるべき姿として何を改革するか	改革を実現するためにどのように業務を改革するか	新業務を実現するために何をシステム化するべきか	各システム機能はどのように要件を達成するか

システム企画
事例プロジェクトでの例

仮説立案のための予備検討活動と仮説検証のための本格検討活動を組みあわせる

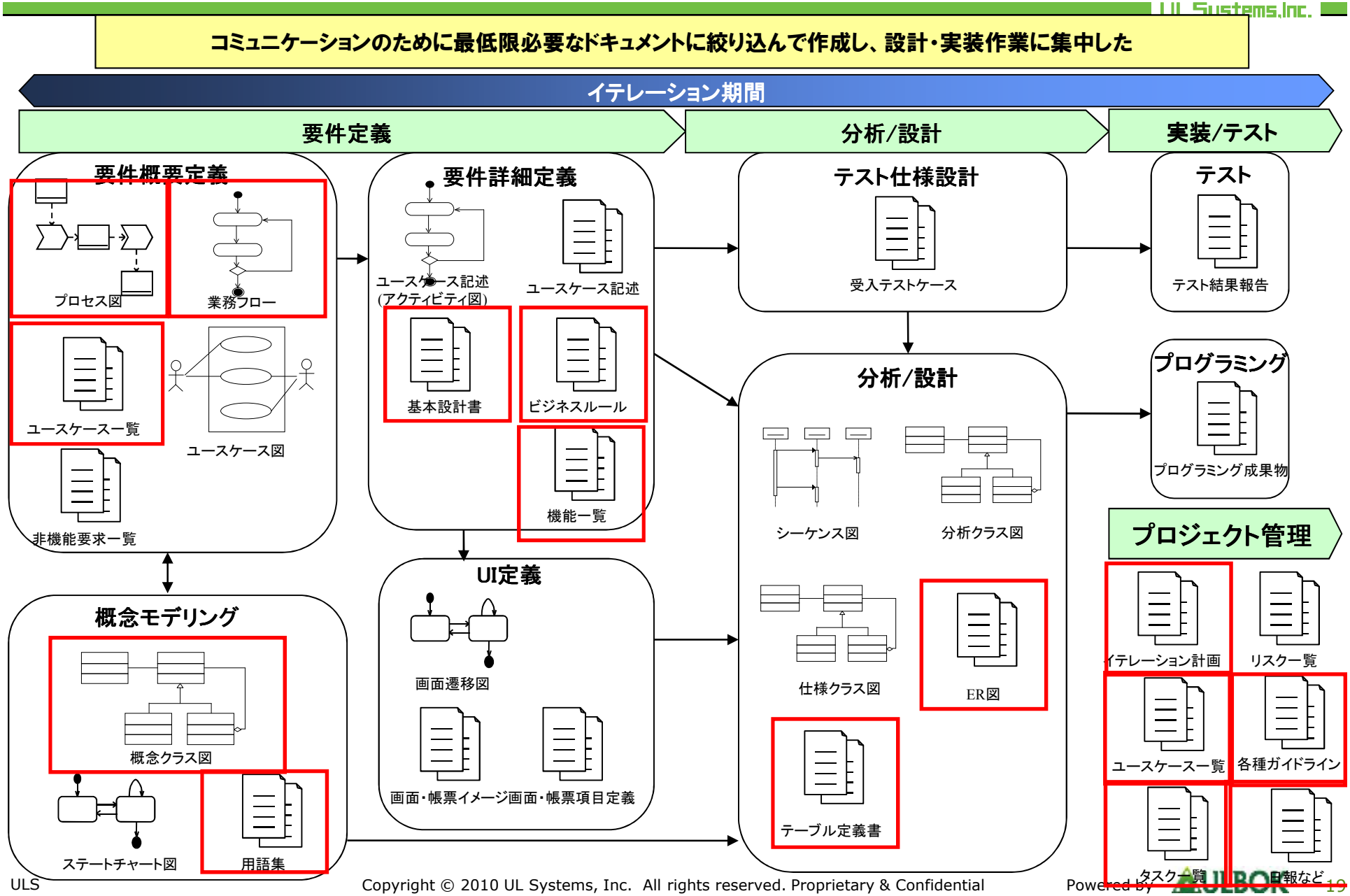


システム企画
主な改革テーマとその実現方法

業務における改革テーマをシステム化の視点からブレークダウンし、要求仕様へと具現化した

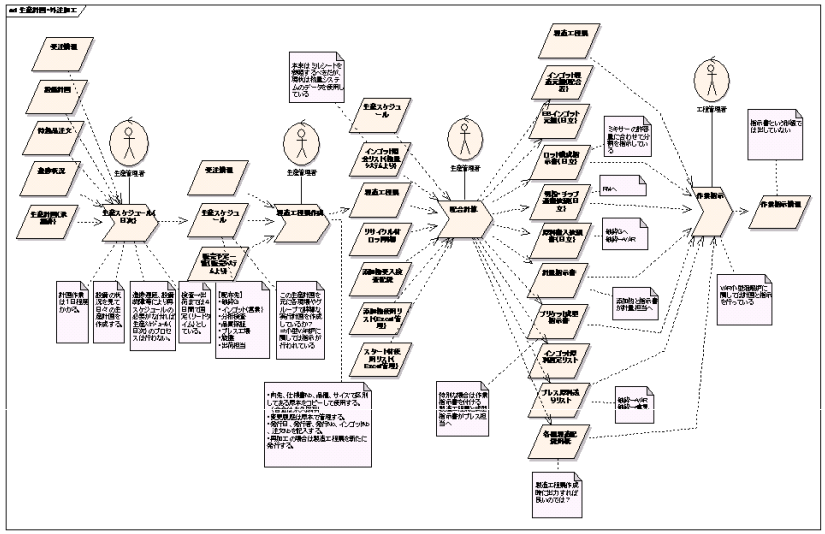
改革テーマ	背景	検討ポイント	システム化内容
工程進捗の可視化	工程全体の進捗管理は実施されておらず、装置毎に生産実績等を手作業で集計・管理していたため、情報をタイムリーに共有出来ず、リードタイムの長時間化や業務効率の悪化が発生	工程における実績管理のイメージを共有し、個別の工程について、どのような情報をどのようなタイミングで取得・管理するべきかを調査・検討	各製造工程について、進捗、物量、作業の実績入力を実施できるようにし、リアルタイムな生産状況の把握を可能にした 各工程での入力項目はマスタ化して必要情報が確実に記録できるように制御した
基準情報の運用統一化	受注から出荷まで一貫した情報にもとづいて製品が作られておらず、必要な情報は各部門での管理となっていたため、情報の紐付けに人手によるコストが発生	現状の識別番号利用状況を調査後、基準となる情報を管理するために必要なマスタの構造や、製造に必要な各種番号についての体系化を検討	品目マスタの整備を行い、仕様に関する情報や工程に関する情報の紐付けを行って、受注～製造～出荷までの情報共有を実現した また、インゴットNoなどのコード体系を分かりやすく統一した

ビジネスモデリング
アジャイル開発プロセスと成果物

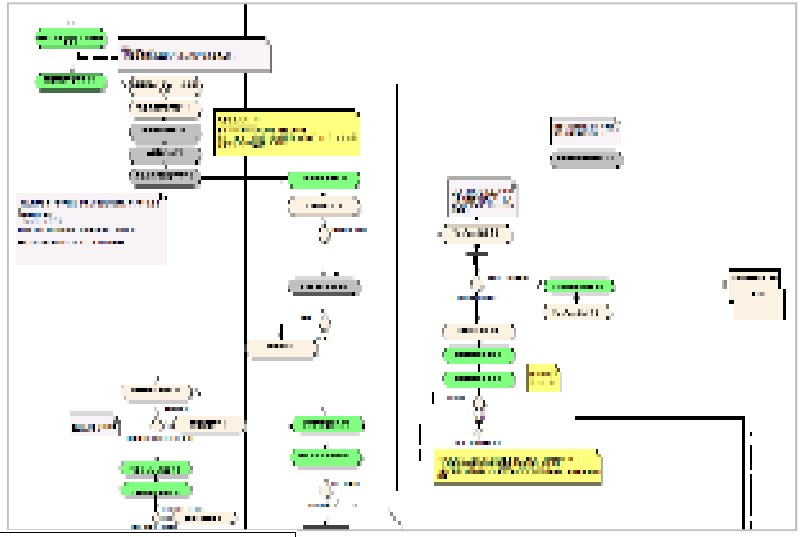


ビジネスモデリング
プロジェクトでの作成例

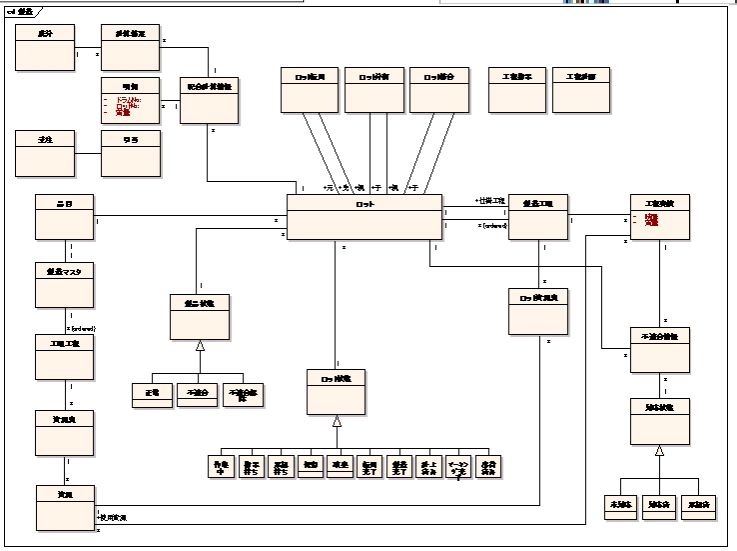
モデルを描きながらユーザと対話することで、要求の見える化をはかった



■ビジネスプロセス図
業務の概略と情報のIN-OUTを整理する



■システム化業務フロー
あるべき姿よりTo-Beの業務を描き、システム機能を抽出する



■概念モデル
ドメインの全体像を描き、その構造を理解し、データモデルへとつなげる

プロジェクト管理
プロジェクト管理における施策

プロジェクト管理についてアジャイル開発を成功させるための施策を実行した

項目	プロジェクトにおける施策	マネジメント項目
コミュニケーション	■ **コミュニケーションの頻度をあげる** 決まった場所、時間で実施するスタンドアップミーティングを活用し、開発チーム全体で状況を把握し共有しつつ、チームにリズムを与える。また、顧客には開発チームと同じ部屋もしくは近い位置にいてもらい、要件検討の打合せや仕様の問い合わせを密に行える環境を作る。 ■ **コミュニケーションの質を維持する** ドキュメントは仕様を理解するための最小限のものに絞り、維持すべきものは即座に更新しておく。また課題管理もチーム全体で共有できる状況をつくっておく。	コミュニケーション管理
チームの成長	■ **チームのメンバ構成に気を配る** スキルの高いメンバと低いメンバの組み合わせにすることで多様性を出し、メンバの成長を促していく。進め方が合わないメンバは、はやめにバスから降ろす。 ■ **成長のきっかけを与え続ける** 開発メンバに自分の力で目標を立てさせ、納期を申告させる。日報、スタンドアップミーティング、進捗会議、イテレーション後の反省会などで振り返りを実施し、生産性を向上させる。	リソース管理 調達管理
ゴール指向	■ **固定された期間内のスコープやスケジュールの決定は必ず顧客が行う** プロジェクトで優先すべき項目を決定しておいた上で、期間を固定した各イテレーションにおいてどの機能を実装していくのか、機能改善と新規作成のバランスをどのようにとるか、などは顧客が決定するように準備する。 ■ **イテレーションのゴールを設定する** 各イテレーションでは開発チームのゴールだけではなく、顧客のゴールやシステム連携のゴールなども必ず計画する。プロジェクト全体がイテレーションを軸に動くように仕向ける。	スケジュール管理 スコープ管理 コスト管理 リスク管理 品質管理

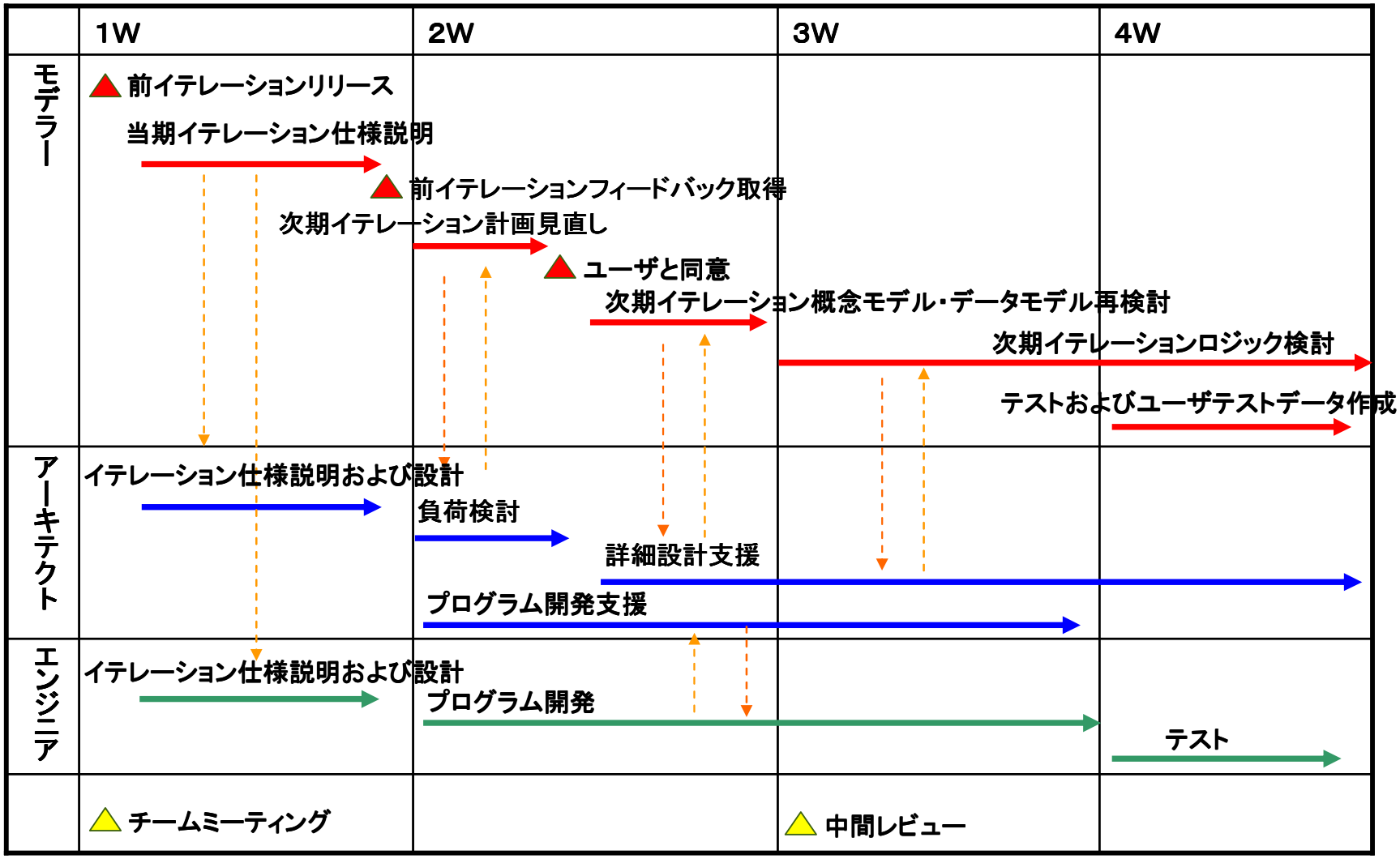
プロジェクト管理
開発計画（イテレーション計画）例

1ヵ月の各イテレーションの達成目標（ゴール）を明確にし、実績に基づいて都度見直しを実施した

イテレーション	システムゴール		システム連携ゴール	現場取組ゴール
第一 （2007/2）	・製造オーダ関連機能実装 ・実績登録実装（製造） ・原料受入機能実装	UC：42 MD：23		・現場運用ルール確定 ・必要帳票類確定
第二 （2007/3）	・月次計画実装 ・実績登録実装（出荷） ・輸送計画・指示機能実装	UC：63 MD：24	・スケジューラ連携IF確定	・品目系マスタ、製造系マスタ整備 ・受払・棚卸仕様確定
第三 （2007/4）	・週次生産計画 ・原料輸送実績登録機能実装 ・外注工程指示・実績登録実装 ・配合計算関連帳票実装	UC：18 MD：24	・スケジューラ連携機能実装	・業務利用イメージ確定
第四 （2007/5）	・生販在計画実装 ・原料所要量計画実装 ・実績登録機能完了 ・品質管理機能実装完了 ・主要帳票印刷機能実装	UC：65 MD：41	・販売システム調査完了 ・検査システム調査完了 ・物量システム調査完了	・第一次リリースフィードバック完了
第五 （2007/6）	・在庫管理機能実装 ・受払機能実装 ・各種帳票印刷機能実装 ・既存システム連携機能実装	UC：（22） MD：54	・既存システム連携IF確定 ・既存システム改修 （マスタ連携・受注連携・分析検査連携） ・原料倉庫連携IF確定	
第六 （2007/7）	・棚卸管理機能実装 ・既存システム連携機能実装 ・原料倉庫連携機能実装	UC：（26） MD：42	・既存システム改修 （出荷連携・ｽﾎﾟﾝｼﾞ照会連携）	・端末設置場所、台数、ネットワーク構成確定
第七 （2007/8）	・ハンディターミナル機能実装 ・マスタ管理機能実装 ・原料倉庫連携機能実装完了 ・バーコードラベル印刷実装 ・既存システム連携実装	UC：（20） MD：50	・販売システム改修 ・検査システム改修	・移行計画着手 ・第二次リリースフィードバック完了
第八 （2007/9）	・マスタ管理機能 ・設備計画機能実装 ・既存システム連携実装完了	UC：（11） MD：35	・既存システム改修作業完了	・移行計画完了 ・マスタ整備作業着手
第九 （2007/10）	・要望対応 ・運用保守機能実装	UC：（3） MD：7	・既存システム監視系機能搭載	・マスタ整備作業

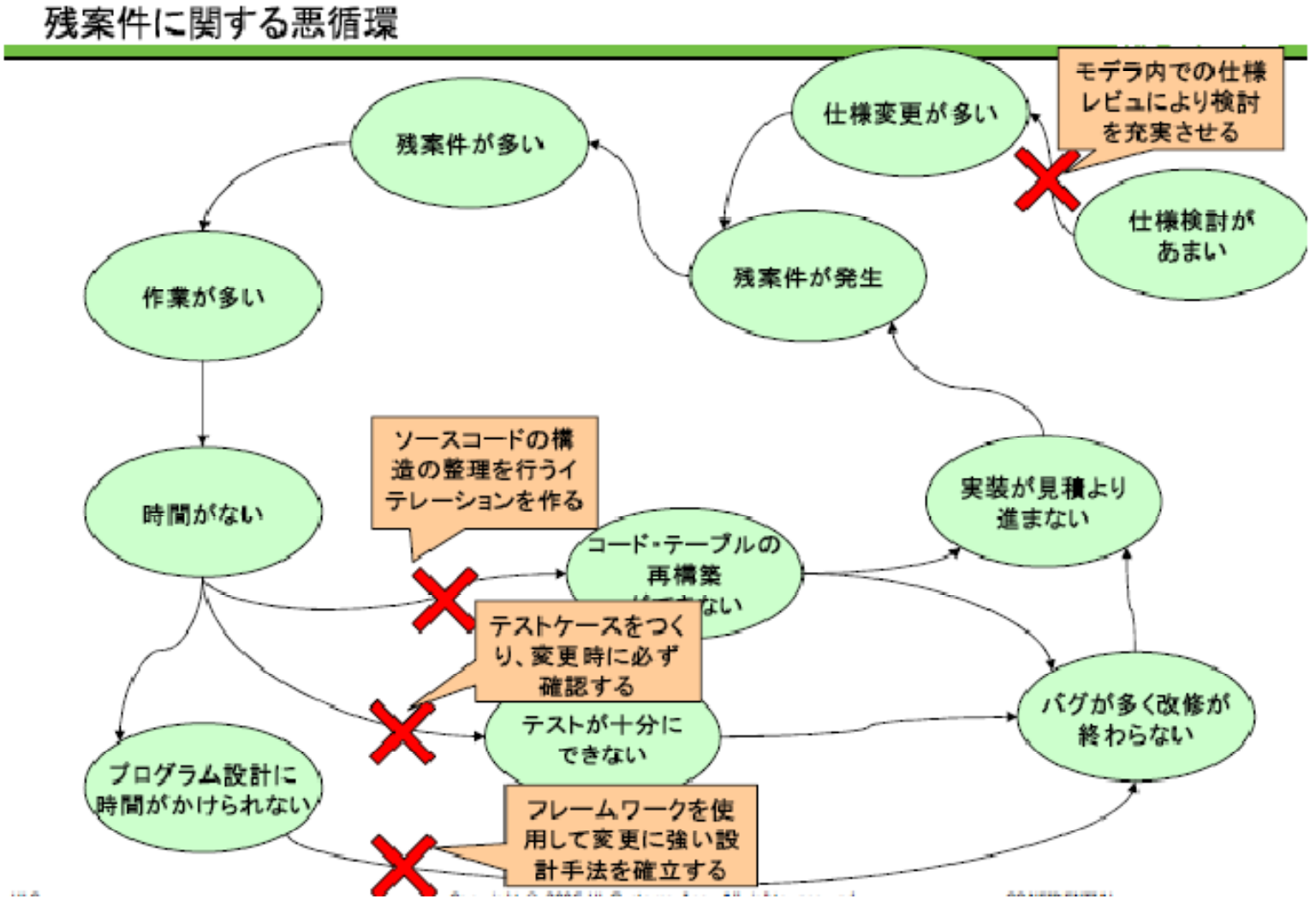
プロジェクト管理
イテレーション内のスケジュール

モデラー（設計担当）による詳細設計作業はエンジニア（実装担当）による実装作業着手よりも半月先行させて実施した



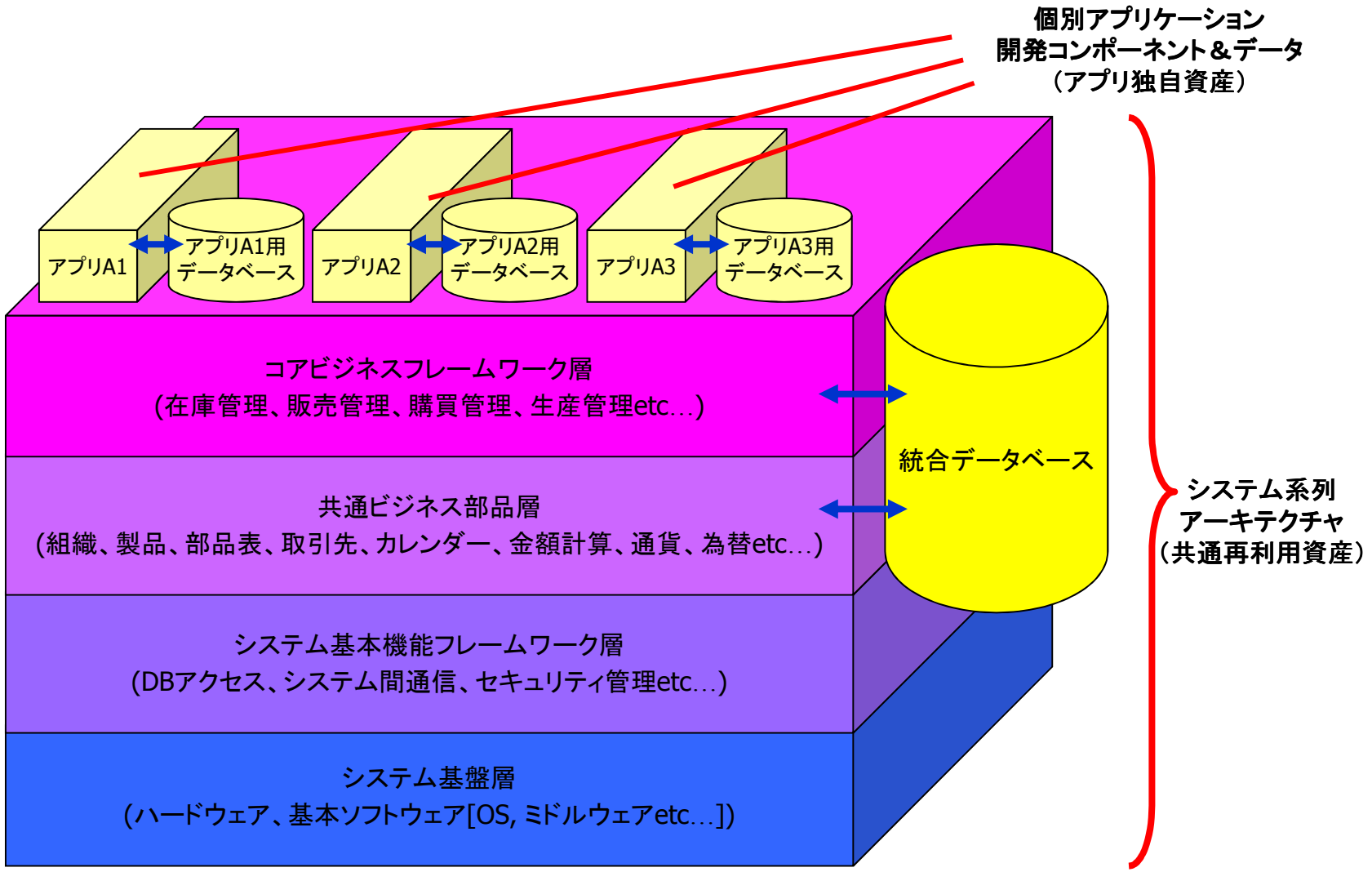
プロジェクト管理
イテレーション反省会の例

イテーション後の反省会で問題点に対する解決策を都度検討し、対策を講じた



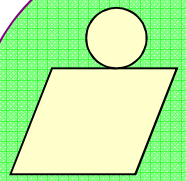
アーキテクチャ構築
アーキテクチャの考え方

各開発者が個別アプリケーションの実装に集中できるように、共通基盤となるアーキテクチャを整備した



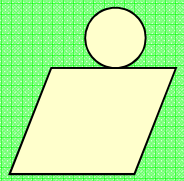
アーキテクチャ構築
アジャイル開発におけるチーム役割分担例

アーキテクトという役割を明確にし、責任をもって作業を遂行するように働きかけた



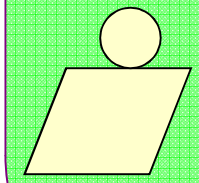
業務検討チーム

- 現場部門の代表としてプロジェクトに参加し、業務のあるべき姿とそれを実現するために必要な要求に関する知識や情報を提供する
- 現場部門と開発チームの間のコミュニケーション窓口として様々な調整や意思決定を行う
- 要求の優先順位を決定して、開発チームに伝える
- イテレーション計画の承認と受入テストを実施する



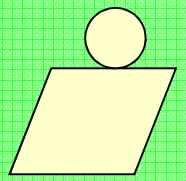
プロジェクトマネジャー

- 開発チームのマネジメントを行う
- ユーザと共同でプロジェクトに関する重大な意思決定を担う
- イテレーション計画を顧客、モデラーと共に立案する
- プロジェクトの成果に関する責任を負う



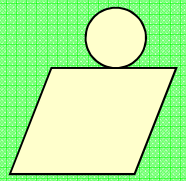
モデラー

- 要求モデリングの主体者として要求を可視化して定義し関係者が共通の認識を得られるようにする
- イテレーション計画を顧客、プロジェクトマネジャーと共に立案する
- 基本設計をアーキテクトと共に実施する
- ユーザの受入テストの結果に責任を負う



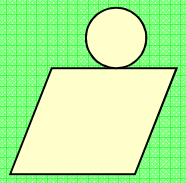
アーキテクト

- 設計、実装に関する重要な意思決定を行う
- 基本設計をモデラーと共に実施する
- イテレーション毎の作業工数を見積もる
- エンジニアへの作業割当、コーチングを行う
- システムアーキテクチャに関する責任を負う



エンジニア

- 設計、プログラミング、テストを実施する
- 単体テスト、結合テストの結果に責任を負う



コンサルタント

- プロジェクトの各作業を支援する
- 開発プロセスの継続的な改善を支援する
- 外部の視点からプロジェクト状況を客観的に評価してアドバイスを行う

まとめ：アジャイル開発を成功させるにはポイントを押さえる必要がある

UL Systems, Inc.

アジャイル開発がうまくいかない理由	解決のポイント
担当者の要望によって改修を重ねているうちに、システムの目的が不明確になってしまう	【システム企画】 システム企画段階で目的を明確にする 実開発に入る前のシステムの企画段階において、業務分析や課題の把握、システム化の効果についてステークホルダーで共有しておく
システムの効果がはっきりしない	
複数部門が関係する要求の調整に時間がかかり、イテレーションでの開発が進まない	
要求のモレが原因の改善要望が多く改修に時間がとられ、新しい機能の開発ができない	【ビジネスモデリング】 要求を見える化し共有する システム要求の把握には、文書ではなくモデルとして描くことで、短い時間の中でもあいまいさを省いてドメインの全体像を共有することができる
データ構造の大きな変更が相次ぎ、リファクタリングのために多くの時間がさかれている	
高いスキルのメンバだけでプロジェクトを構成できない	【プロジェクト管理】 開発チームを成長させ要求を適切に管理する 顧客との間およびチーム内におけるコミュニケーションを通じて、要求のトリアージを実施すると共に、PDCAサイクルを短く回してチームを成長させる
スコープや要求が変わってしまうため、進捗管理がうまくできず、プロジェクトの状況がわからない	
実装担当が実装する箇所が多く、イテレーション期間内に予定していた機能の開発が終わらない	【アーキテクチャ構築】 変更に強いアーキテクチャを整備する 変更に対する短時間での対応や、コードの品質を一定に保つため、プロジェクトの土台となる共通アーキテクチャを整備する
変更が生じる度に手を入れなければならないコードが多く、開発が進まない	

CONFIDENTIAL