Vol.2014-IS-130 No.4
2014/12/8

施主中心の開発プロセスに関する一考察

雑賀充宏† 児玉公信†

企業基幹系システムにおいて，ベンダ依存の問題が指摘されている．情報システム部門が主導的に情報システムの構築ため，情報システムサイクルの所有者である CIO，情報システム部門を，建設プロセスのメタファーにより，「施主」と呼び，施主中心の開発プロセスを設計した．業務の視点から情報システムを管理階層と業務相より分割及びに階層化アーキテクチャによる分割で施主の要求単位をした．実装方法に依存しない要求設計のプロセスを定義し，経営目標（原要求）から業務設計，機能設計を行えるプロセスにした．要求を資源の状態の変換とし，それを表現する方法として，施主が使用できるように UML2.0 の使用ルールを定めた．体制は，構想，設計，施工とチームを分割し，ベンダである施工チームは，役割を生産性と品質の確保とし施工に専念できる体制と定義した．実際のプロジェクトに適用した．その結果，いままで進まなかった情報システム構想が立案でき，施主中心で要求を設計することができ，情報システムサイクルを運営できている．

A Discussion on the Owner Driven Development Process

MITSUHIRO SAIGA† KIMINOBU KODAMA†

1. はじめに

企業の基幹系システムの構築において，ベンダ丸投げ，過剰依存で開発トラブルや訴訟が増えているという[1]．また，ユーザ主体開発で発注者責任を果たしてプロジェクトを進めている事例がある[2]．従業員 1000 人以上の大企業では，IT 戦略策定と戦略実行の分担が進み，従来の業務を情報子会社やベンダに移管する動きがある[3]一方で，情報子会社を親会社に統合する例[4]もある．情報システム部門が情報システム構築及び運営で果たすべき役割に関する議論は，まさに今日的な課題となっている．

筆者は，企業の基幹系システム再構築業務において，情報システム部門に対するアドバイザの役割を負っている．その業務の中で，上述の課題に対応するために，施主中心の開発プロセスを提案し，情報システムの設計と実装，さらに運営を支援している．

本報告では，施主中心の開発プロセスの目的と目標，具体的なプロセスの定義とその適用結果について述べる．

施主中心のプロセスでは，まず企業情報システムの中長期的な発展過程を組織学習のプロセスと見なす情報システムサイクル[7]の所有者を，建築プロセスのメタファーによって「施主」と呼ぶことにした．同様に，ソフトウェアを実装するベンダを,「施工者」と呼ぶことにした.「ユーザ」は，事業部門の利用者および顧客を指すものとした．建築のメタファーを採用したのは，現在の都市計画おいて，既存のインフラストラクチャを活かしつつ，住民との合意を形成しながら，街区の再開発を漸進的に行うことが，情報システムも，既存のシステム基盤を生かしつつも，サブシステム単位で再構築を行うことに類似しているという理由からである[7]．施主中心のプロセスの要点は，情報システムの全体構造を維持しながら，業務目的の視点からサブシステムを切り出し，実装方法に依存せず要求を設計することである．以下では，このプロセスの規定の妥当性と，それを実践するための施策について経験的に論じる．

2. 研究の概要

2.1 研究の背景

児玉[5]は，変化するビジネス環境に柔軟に対応して，経営層や事業部門の期待・ニーズに応えた情報システムを再構築する方法を提唱している．それは，システムアプローチによる企業情報システムの構造を再構築するもので，現状（as-is）の問題を分析して原因除去を目指すのではなく，目標状態（to-be）を構想し，その実現を図る点に特徴がある。この方法を早期アーキテクティング法と呼んでいる。目標状態の策定は，会社の経営目標（原要求）に対して行うべき業務プロセス（仕事）を明らかにする「分析」と，仕事をより速く，正確に，かつ効率的に可能とする機能の「設計」と，それを具体的に人工物として提供する「実装」という局面に分離される．原要求から機能を設計するところで，施主は責任を持つ．また，実装物を業務環境に適用して効果を生むことに責任を持つ．

今回，実際の再構築プロジェクトを進めるにあたり，早期アーキテクティング法に基づく開発プロセスを明確に規定することにより，施主主導がどのように実現するのかを考察する．

†株式会社情報システム総研
Information Systems Institute, LTD.

2.2 研究設問

本研究のテーマは，施主中心の開発プロセスによって，施主が中心となり，目標状態を達成しうるかを問うことである．

2.3 研究の方法

筆者は，次の設計課題として開発プロセスを設計した．

① 施主は原要求から機能設計まで行えるか．
② 施主は機能設計による実装物を獲得できるか
③ 施主は実装物を業務環境に適用して効果を生むか
④ 施主は上記のサイクルを継続的に循環できるか

実際に適用した結果を観測することで本開発プロセスの有効性と課題を評価する．

3. 施主中心の状態と開発プロセス

3.1 施主とは誰か

情報システムは，JIS の規定で「情報処理システムと，これに関連する人的資源，技術的資源，財的資源などの組織上の資源とからなり，情報を提供し配布するもの」[6]と定義されている．これは，情報システムは，ソフトウェアだけでなく，人の活動も含めたシステムであることを示している．児玉[7]は，情報システムの構築プロセスが，継続的に循環するサイクルを図 1 の情報システムサイクルで再定義した．その内部には，ソフトウェアなどの実装物を生成するソフトウェアプロセスを包含する．情報システムは，都市計画に類似しており，継続的に時間をかけて行われる漸進的なプロセスである．

施主とは，情報システムサイクルの所有者になる．具体的には CIO，であり CIO の代理としての情報システムメンバーである．事業部門が情報システムサイクルを担当することもありえる．

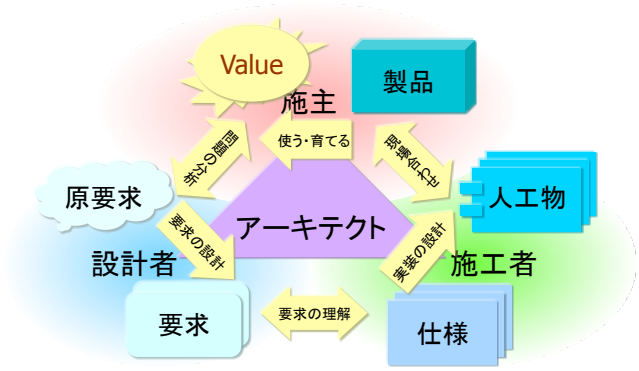

図 1 情報システムサイクル

3.2 施主中心の必要性

施主中心が必要な理由の一つは，開発ベンダとのトラブル防止である．いわゆる丸投げで，開発ベンダが人工物の生成以外の役割を担う場合，要求設計，要求管理の責務が不明瞭になり，トラブルに発展するケースがある．

もう一つの理由は，情報システムの効果の有効性の問題である．経営目標を実現するよりよい情報システムを構築するためである．

情報システムが扱う問題は，ill-defined でソフトな問題であり，構造化され明確に定義できる well-defined でハードな問題とは対照的に複数の問題がお互いに絡み合い，それに対する完全な答えはなく，局所的な答えは多数あっても，それがまた新しい問題を生む可能性を持っている[8]．さらに事業の多くは多様的で，複雑であり，事業内容は変化している．また，業務運用は従来から方法に慣れ，例外的な対応は暗黙知化している．このような性質の情報システムの開発は，ベンダでは担えず，本来的に施主が中心であるべきである．

3.3 開発プロセスの必要性

施主は，情報システム，ソフトウェアの開発に関する専門知識を十分もっているとは限らない．基幹系の再構築の機会がない場合，業務の全体像を把握していないこともある．このような背景の中，施主中心なるための施策の一つが開発のプロセス化である．成果物，アクティビティ，体制を定義して，施主中心の活動ができるようにする．

4. 開発プロセス策定のための先行調査

施主中心の開発プロセスを設計するにあたって，ユーザ企業が中心的に情報システムの運営を行うためのアプローチがある.開発プロセスを設計するために参考になるかを検討した．

4.1 開発標準フレームを定める

ユーザ企業が開発標準を定める場合，その枠組みを特定の開発手法，ソフトウェアフレームワーク，ベンダ提供プロダクト，開発プロセス管理ツールに依ることがある．実績のある枠組みが導入しやすいと考えるためである．しかし，施主中心の視点からすると，実装技術に依存した開発プロセスは，要求設計を制約するという問題がある．また，目標状態を実現する最適な技術やプロダクトが市場に登場したときに，既存の開発標準と合致しない可能性があることも問題である．

実装技術に依らない開発標準のあり方として，JIS X 0170:2013(ISO/IEC 15288:2008)「システムライフサイクルプロセス」がある．施主は，情報システムサイクルのプロセスとして JIS の枠組みを参考にすることができる．

4.2 ベンダマネジメント

ベンダとの間のトラブルを防止するために，契約，手順，ベンダ評価などの管理面を強化しようという考え方である．「共通フレーム 2013」がある．ベンダとの協力体制のあり方も，プロセスの枠組みの中にいれておかなければならない．

4.3 IT 投資

松島[9]は，IT 投資は，「経営目標を達成するという目的のために，必要となる機能を開発し，利用者に提供するという手段的活動の遂行が、目標の実現を支援するという文

脈が適合する.」という．目標実現の手段との位置づけで，IT 投資の承認が得られる開発プロセスでなければならない．

4.4 ユーザ中心設計

ユーザ中心設計（User-centered design , UCD）は，利用者であるユーザの使用性品質をためのものである．施主中心とユーザ中心とは別の概念である．

5. 情報システムアーキテクチャ

5.1 ビジネス視点での情報システムアーキテクチャ

施主中心の情報システム開発を行うには，規模と複雑性への対応方針を定めなければならない．複雑性を増大させる要因は，前述の ill-define なシステムが問題を創発すること，業務自体が多様なバリエーションを持っていること，ソフトウェアの実装上の問題に起因すること，コード・データのあり方に起因することなどが考えられる．アーキテクチャは，開発規模と複雑性への取り組みから始まっている[]．適切なアーキテクチャを実現することによって，問題領域を，施主が取扱いやすいサイズと複雑性に分割していく．ビジネスの視点から分割していくのが施主は取扱いやすい．テクノロジーの視点から，実行環境を規定するアプローチは，基盤プロダクトの陳腐化問題とアプリケーションに複雑性が残ることから，十分とはいえない．

5.2 管理階層と業務相

業務の視点での，情報システムの構造化として，児玉[5]は，情報システムのアーキテクチャを策定した(図 2)．企業システムは,いくつかの管理階層に自己組織化されており，それぞれ特有の管理の主体と管理対象のライサイクルを持つ．さらに、知識，計画，実績，報告の業務相を持つ．このアーキテクチャは，実装技術に依存しない．新しいビジネス領域ができたときは，業務の視点からアーキテクチャを更新できる．

管理階層と業務相で分割された要素（ドメイン）を開発のプロジェクト単位とできる．ドメインは，既存のサブシステムを生かしてもよいし，パッケージを導入してもよいし個別に開発してもよい．導入や破棄のサイクルが異なってもよい．

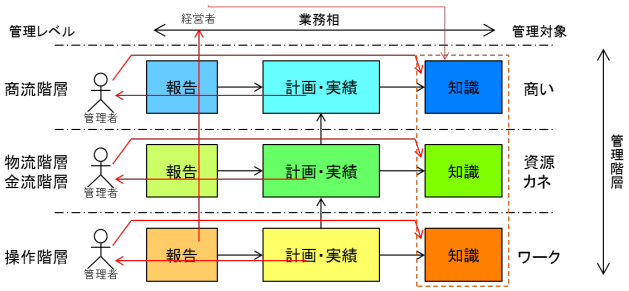

図 2　管理階層と業務層

5.3 ドメインの階層化アーキテクチャ

機能開発する場合の構造として，さらに児玉[7]は，管理階層と業務相よって分割された要素内を階層化アーキテクチャより，ユーザインタフェース(UI)層，業務層，ドメイン層，永続層と分割した(図 3).

機能要求をまとめる単位とできる．ユーザインタフェースは，画面・外部インタフェース，業務はユースケース，ドメインは，ドメイン概念モデリングして要求設計する．

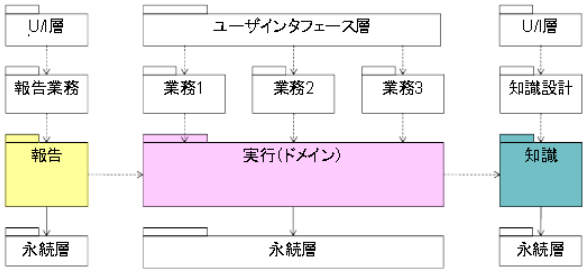

図 3　階層化アーキテクチャ

6. 開発プロセスの提案

情報システムアーキテクチャによって，情報システムの構造と施主の取扱う単位を定めることができた．次に各要求を獲得するための方法とアーキテクチャを遵守する開発プロセスを提案する．

6.1 プロセスと成果物

プロセスとは JIS では,「インプットをアウトプットに変換する，相互に関連又は相互に作用する一連のアクティビティ」と定義されている．プロセスは資源をある状態から別の状態への一連の変換と言える.施主は,開発において，原要求から実装物まで変換される成果物の状態と過程を明示的に把握することで，開発を管理することができる．

施主中心の開発プロセスとするには，次の要件を満たすようにする．

①各成果物の終わりの状態が定義されている
②各成果物の記述方法が明確になっている
③標準に基づいている

標準に基づいたプロセスと成果物は，異なるベンダに対しても適用でき，施主自身の開発プロセス策定を効率化とその品質を担保できる．プロセスの枠組みは，JIS X 0170:2013 システムライフサイクルのテクニカルプロセスをベースに，ISO を参照した．具体的な設計書の表記は，UML2.0 を採用する．UML2.0 は，国際的にも標準な表記法であり，情報システムアーキテクチャ，業務要求，ソフトウェアの構造も統一的に記述することができるからである．

6.2 プロセスと成果物の対応例

プロセスと成果物の対応を表 1 に示す. UML の記法については、多くの文献があり，参考にすることができるが，施主チームとしての統一性を保つため，記述のガイドをまとめた.

表 1 開発プロセスと主な成果物

プロセス	主な成果物
利害関係者要求事項定義	業務構想書 要求関連図 業務シナリオ 論理アーキテクチャ図 概念モデル
要求事項分析	状態遷移図 業務フロー図 ユースケース図 ユースケース記述 概念モデル 画面モック 外部インタフェース仕様
方式設計	オブジェクト図 方式設計書 実装モデル 業務フロー制御
実装	物理アーキテクチャ図 シーケンス図 コード プロダクト
検証	テストシナリオ テストケース テストデータ

6.3 利害関係者要求事項定義プロセス

管理階層と業務相の考え方に基づき，企業の活動に応じて設計したのか，論理アーキテクチャ図である．これは，ビジネス構造と情報システム構造を定義するものであるため，この段階から早期に作成する．開発対象となる事業部門のドメインの概念モデルも作成する．ビジネス上の構造が含まれているため，論理アーキテクチャ図と概念モデルで経営層も含む利害関係者と目標状態を共有することができる．

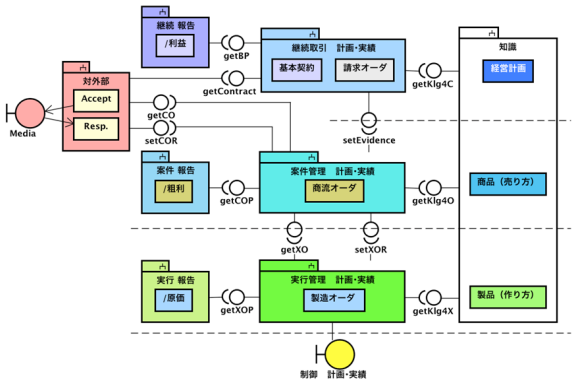

図 4 論理アーキテクチャ図例

6.4 要求事項分析

要求事項分析では，業務設計と機能設計を行う.

6.4.1 業務設計

(1) 状態遷移図

業務設計では，最初に概念モデル上の資源の状態遷移図を作成する．資源は，商品，勘定，設備などである．これらの状態・場所を遷移させていくことが業務を表現することになる.

(2) 業務フロー図

業務フロー図は，資源の状態を遷移させるアクションとして記述する．アクタは，論理アーキテクチャ図で示した管理階層とする．管理階層は，固有の管理の主体，管理対象，管理のサイクルを持つからである．実際の担当者は，アクタの人的資源への割当であるので, 運用設計で決める．アクタ間で交換するメッセージを明記する．これが，管理階層間のインタフェースになる.

このとき，概念モデルを同時に精緻化していく．概念モデルを実装モデルに転換し，それを忠実にドメイン層に実装するためである.

6.4.2 機能設計

業務を支援する機能要求として設計する．施主は機能要求を保守・維持できるようにする．実装のための一時的な要求にはしない．追加・変更される機能要求が，実装物の中に埋没すると，仕様のブラックボックスになり施主中心を阻害する.

(1) 原ユースケース

機能設計は，業務設計でのアクションを支援する機能として業務フローと対応してユースケースを抽出し，内容を記述する．最初の段階では，機能設計は発見的な活動であるため，機能を抽出していくことに注力する.

(2) 設計されたユースケース

原ユースケースを，概念モデルの言葉を使って表記する．ユースケースの記述粒度，ユースケース間の関係性の整合をとる．設計されたユースケースにする理由は，開発，テストに展開しやすい形にすることと，保守性を高めるためである.

(3) ユースケースによる要求追跡性の確保

ユースケースは，要求・設計・実装の追跡性を保つ基点となる．ユースケース記述にユースケースシナリオを記述し，それを実装単位，機能テスト単位とする.

(4) 要求に対する品質追跡性

ユースケースシナリオとテストシナリオを対応させる．このことで，要求に対して機能テストの実施度を把握できる．ユースケース全体を通して，テストシナリオの世界観を定めておく．登場する顧客，仕入先，組織，担当者，商品・サービスなどを決め，各ユースケースに登場させる．実業務に近いテストシナリオにすることで，テストシナリオ自体を維持することができる.

6.4.3 ユーザインタフェースの要求設計

ユーザインタフェースは，ユースケースを呼び出すものとして要求設計する．一つの UI ら複数のユースケースを呼び出すこともあり，一つのユースケースが複数の UI から呼び出されることもある． 画面の場合は，モックアップ用のツールなどで使用性の確認を中心に要求設計する.また，インタフェース仕様として, Mapping, Translation, Validation の 3 つのルール[1]を定義する．

6.5 方式設計プロセス

情報システムアーキテクチャにおいて，制御した複雑性が実装の段階で復活しないように制御しなければならない．そのために方式設計で遵守すべきことがある．施主と共同で設計チームが方式設計を行う．

6.5.1 外部複雑性と内部複雑性のドメインからの分離

複雑性には，情報システムの外部と連携するときの外部複雑性と、情報システム内部で処理条件を規定する内部複雑性がある．外部複雑性の例としては，注文伝文，伝票，報告書などがある．取引先や業務によって多くのバリエーションがあると外部複雑性が高いという．外部複雑性は，ドメインの外部に出すことで，その複雑性を制御する．外部化した外部複雑性の詳細な設計・実装は，ベンダに任せてもよい.

内部複雑性は，契約条件の一種とみなしルール化，知識化することにより制御する．

6.5.2 サブシステム間の疎結合性確保

論理アーキテクチャ図は，ビジネスの関心・興味を反映した分割になっている．実装では，各ドメイン間は，疎結合にしておかなければならない．各ドメインは，管理のライフサイクルも異なるため，ドメイン位に改修，廃棄，新規追加ができるようにする．そのためには，疎結合性を保持する必要がある．

6.5.3 インフェース設計

インタフェース設計出は、各ドメイン間の情報の流れを決める．共通通信基盤を構築し，管理階層と分離する．

6.6 試作

利害関係者要求設計プロセス，要求事項分析プロセス，方式設計プロセスでは，試作を用いることも有効である．実際に動くソフトウェアを見ながら検討すると多くの気づきが得られる．また，方式的な検証も行える．実装プロセスに移るときに，実装は，試作を発展させるか，別の方法で開発するかを判断する．

6.7 実装プロセス

実装段階の詳細の開発プロセスは，ベンダからの提案により，施主と協議して決定する．施主中心で進める場合の留意点について考察する．

6.7.1 開発フレームとプラクティス

施主が，実装手段と切り離された要求を実装の前に持っていれば, 実装の手法は，ウォータフォール，アジャイル，パッケージ導入，SaaS 利用などでもよい．

6.7.2 プロジェクトの資源割当管理

施主中心であるためには，プロジェクトの資源の割当状況と把握し，問題が発生した場合の是正措置の妥当性を判断できるようにしておく.プロジェクトの資源には, 要員，時間，環境，データ，情報などがある．

資源の割当状況を把握するには，WBS（Work Breakdown Structure）を活用する．WBS は，「作業分割構成」などの訳で，作業であるとされていたが，最近，成果物の構成要素として本来の定義であるモノであることが明確化さてきた[15]．これらを作成する作業時間を測定して，ポイント化する．構成要素単位のポンイトにより，作業工数，生産性を評価する．要員の配置計画も規模と生産性から換算できるようにしておく．

7. 開発プロセスを推進する体制

開発プロセスを推進する体制は，施主チーム，設計チーム，施工チームの 3 チーム編成とする．設計チームと施工チームが複数になってもよい．設計チームと施工チームを分離したのは, 設計チームと施工チームの責務を明確にし，それぞれ，その専門性を発揮して，生産性を向上させるためと，施主が各チームの可視性を高めプロジェクト主導しやすくするためである．

7.1 体制と役割

7.1.1 施主チーム

施主チームは，情報システムサイクルを運営し，要求提示し，人工物を獲得する役割を持つ．以下のチームからなる．

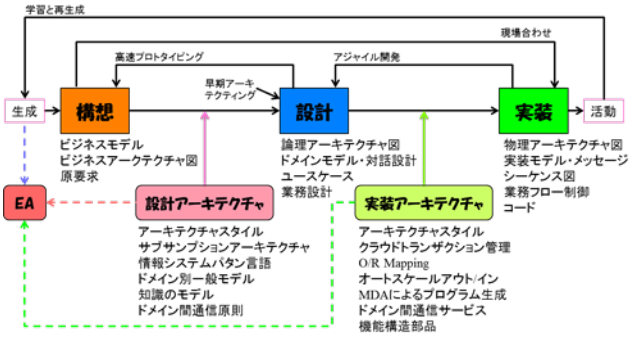

図 5 構想・設計・実装の構成

(1) CIO チーム

CIO は, 情報システム戦略を立案, IT 情報投資を決定し，それを実現することに責任を持つ役員のことである．この役割を担うために，CIO のスタッフとなる CIO チームを編成する．情報システムイクル全体を運営する役割を担う．情報システム全体のアーキクテクィングを行う情報システムアーキテクトも CIO チームの一員である．

プロジェクトを立ち上げ前に，プロジェクト憲章を作成し，プロジェクト推進者を任命する．

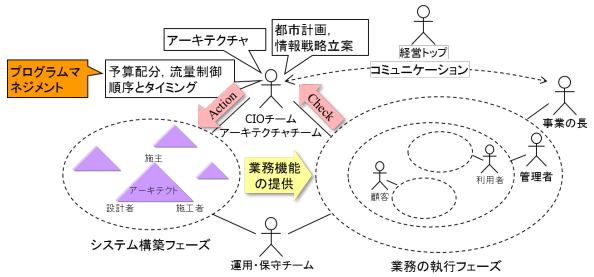

図 6　情報システムサイクル運営

(2)　PMO（プログラム・マネジメント・オフィス）

情報システムのプロジェクト全体の推進を行う．各プロジェクトを立ち上げ，監視コントロールする．プロジェクト全体の費用管理を行う．情報共有環境の整備，構成管理の整備を行う．本開発プロセスの改定の責任も持つ．

(3)　プロジェクトの施主

情報システムサイクルの所有者も施主であり，各プロジェクトの所有者も施主である．プロジェクトの施主は，プロジェクトの計画，実行，終結を行う．施主は事業部門の代表者と協議して，原要求まとめ設計チームに伝え，設計チームが設計され業務設計，機能設計を作成する．施主は，開発のスコープ，優先順位を決め，施工チームに開発をオーダする．施工チームからのフィードバックで，優先順位の調整を行うことがある．

7.1.2 設計チーム

施主チームの作成した原業務フロー図，原ユースケースを設計された業務フロー図，ユースケース記述に変換する．モデラが中心的なメンバである．設計チームでは，外部の専門家の協力を得てもよい．

7.1.3 施工チーム

(1)　実装チーム

設計された要求を実装する．ベンダがその役割を担うことが多い．生産性と品質に責任を持つ．作業工程やタスクごとの予実，品質を可視化する．作業の予定と実績の差異によるタスク実施計画の是正や，より生産性と品質を高めるための作業調整（オプティマイズ）は，施工チーム内で行う．

(2)　結合チーム

ドメイン間の結合テストを各ドメインの実装チームと共同して行う．

7.2 会議体

(1)　施主ミーティング

CIO チーム，PMO，プロジェクト施主により会議体を持ち，プロジェクトチーム間の問題について解決を図る．

(2)　アーキテクチャ委員会

情報システムアーキテクチャを遵守することを目的する委員会である．新しい方式やサブシステムの追加・変更があるときに随時開催する．メンバは，PMO，アーキテクト，プロジェクトの施主である．必要に応じて技術的に詳しい設計者または施工者が参加することがある．サブシステム間の凝集度，結合度及び情報の流れ，インタフェースの妥当性を確認し，追加・変更する方式が情報システム全体のアーキテクチャを遵守しているか確認する．ここで確認された方式により，設計・実装が行われる．

(3)　モデラミーティング

ドメインの概念モデリングに更新画あるときは、アーキテクト，設計チーム，施主にレビューする．概念モデルが実装と解離しないようにする．施工チームの判断によりモデルと異なる実装がされることは,避けなければならない．そのため，要求の変更に伴う要因においても，設計チームのフィードバック要因においても，モデルを変更する場合は，この場で，関係者合意の上決定する．

8.　適用結果と今後の課題

本開発プロセスは，一つの基幹系構築プロジェクトに適用している．プロジェクトは進行途中であり，中間段階での評価になる．本開発プロセスで設計課題を解決したかを評価する．

8.1　施主は原要求から機能設計まで行えるか.

8.1.1 実施した状況

本開発プロセス以前は，施主は，ビジネス状況の変化から基幹系システムの再構築の必要性を認識し，検討を数年にかけて行っていたが，結論がでていない状況であった．本開発プロセスを活用することで施主は、中心となり次のことを実現した．

・基幹系再構築の全体業務構想書を作成した
・全体ロードマップ及び IT 投資計画を策定した
・役員会の承認を得て，基幹系構築プログラムを開始した
・事業部門と業務構想書を共有した.
・状態遷移図，業務フロー図を作成し機能設計した
・ユースケース記述を作成し機能設計をした
・画面モックツールを使用して画面要求を作成した
・設計チームが施主の原要求に基づき，設計されたユースケースを設計した

施主は，本開発プロセスにより，目標状態を表記した．UML2.0 の記法を初めての施主においても、十分理解して，活用できている．

8.1.2 課題事項

(1)　ドメイン責務の設計

ある業務の責務を管理階層の上位側と下位側のどちらに持たせるかで議論があった．実組織において，顧客対応や業務効率の観点から他組織の本来業務を代替しているケースがあるためである．このような場合は，本来業務を行う管理階層からの代替する管理階層へ情報をキャッシュと

して複写して対応する．企業の業務全体のビジネス構造の設計と情報システムアーキテクチャは連動する．本来業務と代替業務の責務設計が必要である．

(2) 内部複雑性の取扱い

施主が中心となり，ビジネスルールとして要求を設計している．

(3) 外部複雑性の取扱い

外部インタフェースの定義は，相手先の関係があり，調査・決定・調整に時間を有している．施主が定義しやすい手法の設計が必要となっている．

(4) 論理アーキテクチャ，概念モデルの作成・更新の課題

外部の専門家により，論理アーキテクチャ，概念モデルは作成された．施主チームは，その内容について十分な理解があり，利害関係者に説明できる．しかし，全体の整合を保ちながら,自ら更新できるところまでは達していない．モデラミーティングなどを通じて，専門家からスキルを伝達する．

8.2 施主は機能設計による実装物を獲得できるか.

8.2.1 実施した状況

施主は，自ら中心となり次のことを実現した．

・プロジェクト憲章とプロジェクト計画書を作成した
・施工チームに要求を 2 週間単位に実装物を獲得している
・生産性の指標を基に，開発見通しを立て，施工要員の配置の計画を施工者とともに実施している．

8.2.2 課題事項

(1) 施工者間での対応の違い

本開発プロセスを理解し，積極的に協力しているベンダにおいては，開発プロセスが適応できている．複数あるベンダの中には,作業計画の提示が十分でないケースがある．

(2) 非機能要求の標準的な記法

非機能要求は，UML2.0 のような標準的な記法がないため，個別の書式になっている．

8.3 施主は実装物を業務環境に適用して効果を生むか

現在，業務環境に適用していないので評価はできない．

8.4 施主は上記のサイクルを継続的に循環できるか

8.4.1 実施した状況

施主が中心となり次のことを実現した．

・開発プロセスに基づいたチーム編成を行った
・そのチーム編成を運営している
・新たに参画するメンバに対して，開発プロセスの理解を促進している．

8.4.2 課題事項

(1) PMO の役割

複数の役割をまとめる PMO の役割で議論があった．計画立案，環境整備，問題解決・調整，事務局的な役割が混在している為である。CIO チームを支援する位置づけを明確した．

(2) 情報システムの有効性評価

施主自身の情報システムの有効性を評価方法が未整備である．評価方法として“EEE”（「適正 efficacy」「効力 effectiveness」「能率 efficiency」）による評価が提唱されている[16]．これらを参考とした評価指標と評価プロセスの策定を進める．

9. おわりに

施主中心の開発プロセスの系統化を試みた．情報システムの構築をビジネス視点からスタートし，施主の本来的責務を考慮し構想から実装まで一貫した形で進めための開発プロセスである．まだ，スタートした段階であり，今後，事例を蓄積し，施主中心の開発プロセスの向上を進めていく．

参考文献

1) 木内里美, ここがおかしい日本の IT 産業, IPA FORUM 2012, 情報処理推進機構，入手先 <www.ipa.go.jp/files/000008457.pdf>（参照 2014.11.10）.
2) ITmedia エンタープライズ，“ユーザー主体”の姿勢が開発のスピードと質を高める，入手先 <www.itmedia.co.jp/im/articles/1108/10/news104.html>（参照 2014.11.10）.
3) 日本情報システム・ユーザー協会，第 20 回 企業 IT 統合調査 2014（13 年度調査），2014.
4) 日経 ITpro，グローバル経営に向けシステム子会社を吸収，2014/03/14，入手先 <http://itpro.nikkeibp.co.jp/article/COLUMN/20140318/544323/?ST=system>（参照 2014.11.10）.
5) 児玉公信，企業情報システムのための早期アーキテクティングの一方法，情報処理学会研究会報告，2012-IS-120, Vol.3, 1-8.
6) JIS X 0001:1994 情報処理用語-基本用語.
7) 児玉公信，情報システムサイクルと原要求の記述，日本情報経営学会誌，28(2), 77-87, 2007.
8) Marshall, C., Enterprise Modeling with UML, Addison-Wesley, 1999. 児玉公信（監訳），企業情報システムの一般モデル–UML によるビジネスの設計，ピアソンエデュケーション，2001.
9) 松島桂樹，IT 投資マネジメントの変革，白桃書房，2013.
10) Zachman, J. A., A framework for information systems architecture, IBM SYSTEMS JOURNAL, 1987
11) 児玉公信，計画・実行システムの一般モデル–生産管理システムと金融業務システムの共通性，情報処理学会研究会報告，2009-IS-109, Vol.4, 1-7.
12) Fowler, M., Analysis Patterns, Add-son-Wesley, 1997. 堀内 一（監訳），アナリシスパターン，ピアソンエデュケーション，1998.
13) 児玉公信，システム間インタフェース試論–バリエーションの源泉，情報処理学会研究会報告，2010-DD-78(6), 1-6.
14) Schwaber, K. and Sutherland, J., Scrum Guide, available from http://www.scrumguides.org/ (accessed 2014-11-10).（邦訳）スクラムガイド，入手先 <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-JA.pdf>（参照 2014.11.10）.
15) Snyder, C., A User's Manual to the PMBOK Guide, Wiley, 2010. 清水計雄，亀井邦裕（訳），PMBOK ガイド・マニュアル，鹿島出版会，2014.
16) 児玉公信，情報システムの有効性評価のガイドラインについて（中間報告），情報処理学会研究会報告，2011-IS-117, Vol.13, 1-5.