システム基盤設計工程の高速化へのチャレンジ

Challenge to Accelerate the Process of System Frameworks Design

川辺　拓郎

要旨

システム開発プロジェクトの成功には、開発体制や開発規模に着目した「マネジメント」が重要であることは言うまでもないが、システム基盤設計の重要性は見逃されることが多い。システム基盤設計と言ってもその範囲は広いが、特にシステム全体のアーキテクチャ設計を行う方式設計が鍵となる。これからのシステム開発は品質保証とともに開発期間の短縮も求められる潮流にあるが、NRI（野村総合研究所）は、開発期間短縮の実現を基盤設計開発の中核である「方式設計」から考える取組みを「The 1/2」と称して実施してきている。

本稿では、「The 1/2」の活動で見出した方式設計時間短縮を更に改善強化するために行った研究内容をご紹介する。

キーワード：プロジェクトマネジメント、PMBOK、ソフトウェアエンジニアリング、エンタープライズアーキテクチャ、システム基盤、システムアーキテクチャ、方式設計、DSM

It is a well-known fact that management which focuses on a development organization or development scale is critical for success of a system development project. In reality, however, we tend to overlook the importance of designing system frameworks. Although the concept of designing of system frameworks is wide-ranging, the key to success is " system design " which carries out architecture design of the entire system. In terms of future system development, shortening a development period and guaranteeing the quality are required.

Nomura Research Institute, Ltd.(NRI) has been addressing challenges with " The 1/2 " which enables to realize shortening a development period by taking into consideration the system design, the core of basic design development.

Keywords : Project Management, PMBOK, Software Engineering, Enterprise Architecture, System Framework, System Architecture, System Design, DSM

１．システム開発の生産性とは

システム開発プロジェクトの生産性向上をシステム基盤設計の側面から考えるにあたり、情報システム構築の根幹であるソフトウェア開発の生産性をいまいちど振り返り、建築やハードウェア製造の世界と対比してみたい。

（1）プロジェクトマネジメント

システム開発プロジェクトを成功させる上では、最近話題にあがることの多い、ＰＭＢＯＫ（Project Management Body Of Knowledge）の知識領域（表１）をベースにしたプロジェクトマネジメントが重要であることは言うまでもない。プロジェクトであるから、Ｑ（品質）、Ｃ（コスト）、Ｄ（期限）の死守は絶対使命であるが、この三要素は生産性と密接な関係をもっている。

表１　PMBOKの知識領域

総合マネジメント	プロジェクト計画の策定、プロジェクト計画の実施、変更管理
スコープマネジメント	プロジェクトの立ち上げ、スコープ計画、スコープ定義、プロジェクト成果物の検収、スコープ変更管理
タイムマネジメント	作業定義、作業順序設定、作業所要時間の見積、スケジュール作成、スケジュール管理
コストマネジメント	資源計画、コスト見積、予算設定、コスト管理
品質マネジメント	品質計画、品質保証、品質管理
組織マネジメント	プロジェクトの組織計画、要員の調達、チームの育成
コミュニケーションマネジメント	コミュニケーション計画、情報の配布、進捗報告、プロジェクト完了手続
リスクマネジメント	リスクの特定、リスクの定量化、対応策の策定、リスク管理
調達マネジメント	調達計画、引合計画、発注先選定、契約管理

（2）ソフトウェアエンジニアリング

生産性向上が「効率」と「品質」の両面を指すこともあって、イメージとしてはわかるが具体的には何をすべきであるかがわかりにくい面もある。ソフトウェア開発においては、属人性の排除（標準化）や開発効率をあげるための取組みとして「ソフトウェアエンジニアリング」への期待がある。

ソフトウェアエンジニアリングは、1968年のＮＡＴＯ（北大西洋条約機構）の科学委員会による「ソフトウェア危機」の提起を発端に、提唱された概念であるとされ、ソフトウェア工学国際会議（ICSE：International Conference on Software Engineering）の活動を中身の具体化起源として、現状では、図１に示すように体系立てられていると言える。

ソフトウェアエンジニアリングは、世界的規模で取り組まれていることは事実でありながら、ハードウェア（製造）業界の「カイゼン」に代表される効率化の例をソフトウェア開発ではあまり多く聞かない。ハードウェア開発にせよソフトウェア開発にせよ、「設計」と「製造」という分類は共通的なものであるにも関わらず、このような状況になっているのは、ソフトウェア開発では、ハードウェア

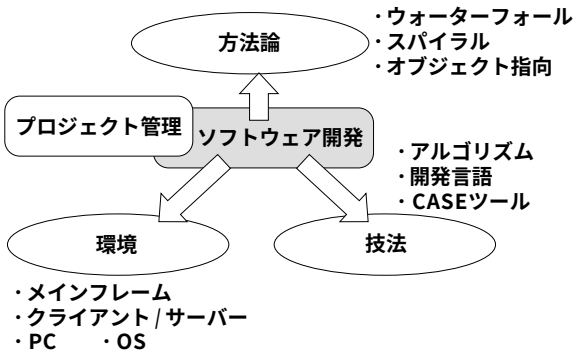
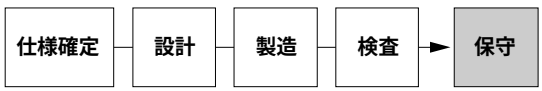


図1　ソフトウェアエンジニアリングの姿

（出所）Mint（経営情報研究会）、『図解でわかるソフトウェア開発のすべて』

開発にない難しさがあることによる。

ハードウェア業界の設計は、製造段階に入っても修正されることのないよう、製造上の曖昧さが排除された質であると言えるが、図2に示すように、ソフトウェア開発ステップにはハードウェア開発との違いがある。



仕様確定：使う側がコンピューターで何ができるのかは知らない
開発側は業務には精通していない
お互いが共有できる知識は乖離しており、知識交換して共有部分を広げることが必要
設計：機械設計は図面によって何をどう作るかがわかるがソフトウェア開発の「図面」はまちまち
製造：ソフトウェアには色々な作り方がある
検査：思いがけない見落としがあったりするので完全な検査が難しい
保守：不具合や機能追加が必要となるが中身の理解は後任には難しい状況であったりする

図2　ソフトウェアの開発ステップ

（出所）Mint（経営情報研究会）、『図解でわかるソフトウェア開発のすべて』

（3）視座、視点の変更

視点をソフトウェア開発からシステム開発プロジェクトに移すと、そこにはソフトウェア開発以外に、機器調達、コンピューター稼動設備等のファシリティの領域、実際のシステム運用や稼動後のメンテナンスを支えるための運用設計等の領域等、ソフトウェアエンジニアリングでカバーしきれていない世界がある。故に、システム開発プロジェクトの生産性をソフトウェア開発だけで考えていくことは非常に難しいものとなっている。

企業活動におけるIT（情報技術）の活用が大前提となる時世において、システム開発がメインフレーム主体の時代からオープン化へ変遷する中で、多くの資産が形成されていることは、新規プロジェクトをいかに成功させるかという悩みに加えて、システム開発投資判断、実施時期判断をどのように行なうのかという課題に直面することにもなる。つまり、その企業に必要な情報システム全体を捉えた実現効果を見ていくことの必要性の現れである。このことを表わす象徴的なものの一つが、日本政府も導入予定のEA（Enterprise Architecture）であろう。これは企業全体の情報システムのあるべき姿を色々な視座（関係者）の立場で定めてから、個々のシステム開発に取り組み、その結果を企業全体のあるべき姿にフィードバックするためのプロセス、成果物、組織の指針となるものである。

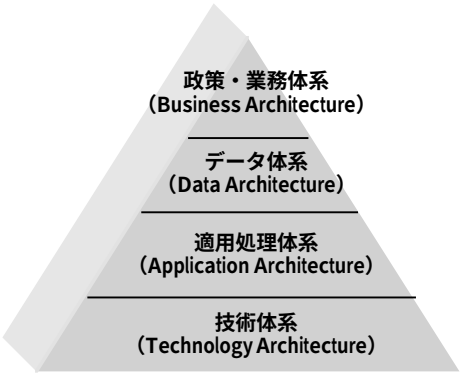


図3　エンタープライズアーキテクチャの体系例

（出所）経済産業省 ITアソシエイト協議会資料 「政府調達のためのIT専門家について～ITアソシエイト協議会中間報告～」

（4）生産性評価指標

システム開発プロジェクトの生産性向上には、ソフトウェア開発とそれ以外のさまざまな活動の両方の取組みが必要となるわけであるが、生産性は何によって評価できるのかも考えておかなければならない。

最もわかりやすいものとしては、アプリケーション開発規模算出とその妥当性評価で利用されることの多い、FP、COCOMOと呼ばれるものである。これは、ある決まりによって、プログラム開発規模算出を誰でも行なえるようにするものであり、目標となる規模を定め、実績との乖離が評価しやすいものとなっている。しかし、前述したように、プロジェクトとしての生産性を見ていく場合には、プログラム開発以外の活動が多く存在し、それらを何らかの共通的な手法で見積り、実績を評価するレベルまでは達していないのが実情である。このような中ではどうしても、「成果物」と「それにかかる時間」という見方をせざるを得ないが、時間については実行する人の技術や経験等による個人差が出てしまう。

このような状況ではあるが、手掛けてきたプロジェクトの実績をベースに、いくつかの指標を作成していくのが今のところは現実的な取組みであろう。NRIにおいてもPMO（Program Management Office：プロジェクトマネージャを支援する組織）に相当する品質監理組織において、過去プロジェクトからの多くの実績が蓄積、利用されている。

（5）アーキテクチャ設計の重要性

生産性を考えるキーワードの一つは、明らかに「時間」である。システム開発プロジェクトにおいて、システム基盤が重要であり、システム基盤の中でもアーキテクチャ設計が鍵を握るということは、時間短縮にどのような意味を持ってくるのかを考えてみる。

アーキテクチャという言葉のそもそもの意味は、建築家を指すものである。建築家は、施主の依頼を具体的な構造物設計や構築に落とし込んでいくための全体設計を遂行し、構築におけるさまざまな役割を持つ職種を支援しながら完成まで活動する役割を担う。

我々が家を建てる場合、建築に関する知識や技術に精通し、部材調達や実際の組立て方法等が手の内になっていれば、工務店に直接

工事発注することもあろうが、大半は、自分たちの希望を建築士や設計士に伝え、作成された図面をもとにコストや期間を勘案しながら進めている。目的とする家の構築を着手するにあたっては、ニーズを構造物として具体化するまでの設計を専門家であるアーキテクトに委ねているのである。これを情報システム開発に置き換えてみると、目標とするシステムの構成や構造を明確にし、アプリケーションが効率よく、信頼性を保ちながら稼動するための基礎や共通的な部分が明確に定義されない限り、構築には着手できないことは自明である。NRIでは、この重要な設計を「方式設計」と呼んでいるが、これはまさにアーキテクチャ設計にあたるものである。

開発プロジェクトには、図４に示すように、実に多くの「人」が関わっており、各々が自身の成果物を完成させるために活動しているが、根底は人と人とのコミュニケーションである。このことは、時間短縮を考える上で、単に作業を見直すこと以外に、交わされる「情報」に着目することの重要性も示唆している。

２．大規模システム開発の実際

システム開発プロジェクトにおけるシステム基盤の活動と時間と情報に着目して生産性を考えていくことが重要であるとしたが、その実態をNRIの例を基に整理してみたい。NRIにおいては、常時三桁の数の開発プロジェクトが進行しているが、開発プロジェクトにおけるシステム基盤領域の重要性は、次のように認識され、実行されている。

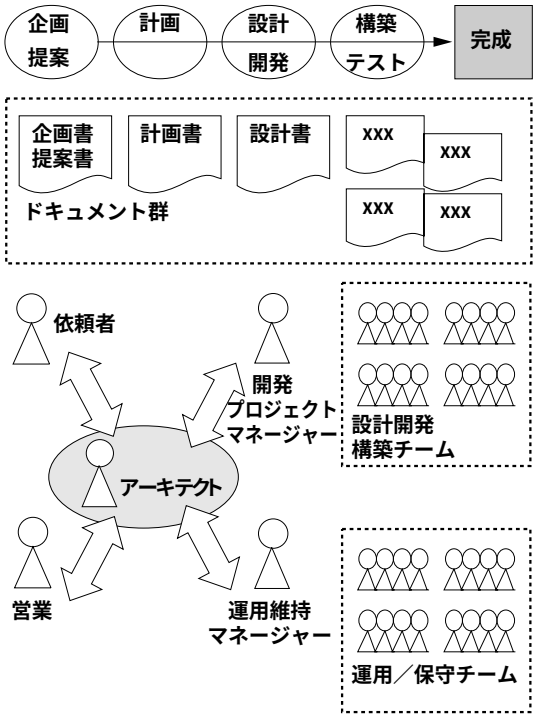


図４　システム開発プロジェクト上の役割

（１）プロジェクト遂行環境

プロジェクトはQ、C、Dの基本三要素との闘いであり、規模や新技術採用等が難易度を高くする。NRIでは、プロジェクトマネージャを孤立させず、全社をあげて支援する制度、標準、組織体制が、図５のような形で整備、運用されている。

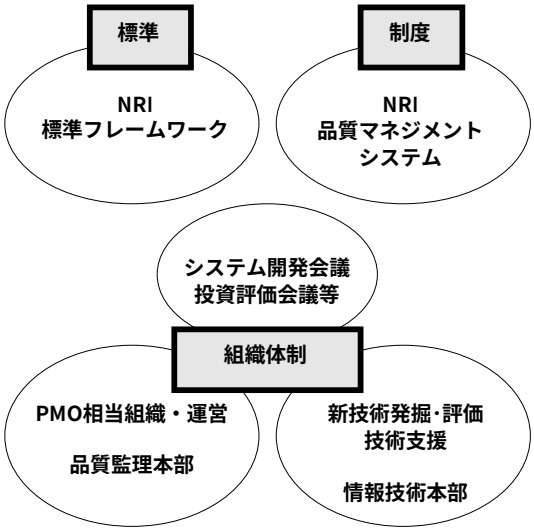


図５　NRIの大規模開発プロジェクト遂行構図

（２）システム基盤設計の明確化

システム基盤は捉えにくい性質であることを鑑み、NRIでは図６に示すような、アプリケーションを支える下地としてのシステム基盤の定義を行なっている。

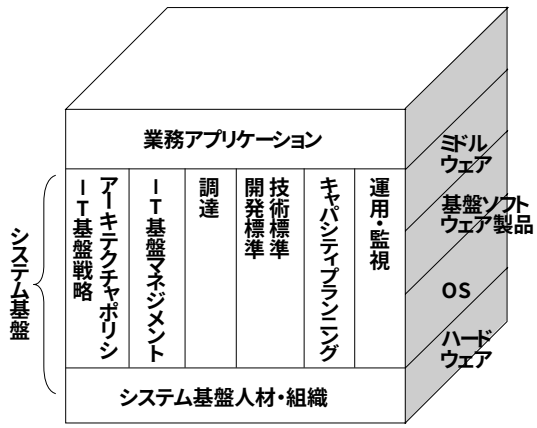


図６　NRIのシステム基盤の捉え方

なお、プロジェクト工程に準えた場合の定義を「NRI標準フレームワーク」として、図７のように捉え、標準形としている。

（３）システム基盤職種と人材教育

図６、図７に示すように、システム基盤として必要な活動は多岐に渡るため、一般的にアプリケーションエンジニアと呼ぶ人材に求められる設計・開発・テストに比べて、より専門性が要求される仕事である。プロジェクトの対象となるシステムが採用する技術、製品種類、性能、信頼性のレベルが広範囲、高度になると、プロジェクト体制上のシステム基盤を担当する要員体制も相応の構えを必要とする。技術進歩の早さや採用製品選択肢が

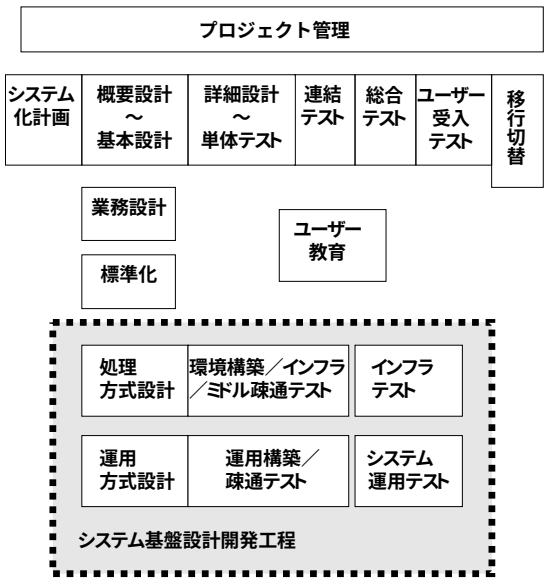


図７　NRI標準フレームワークでのシステム基盤設計

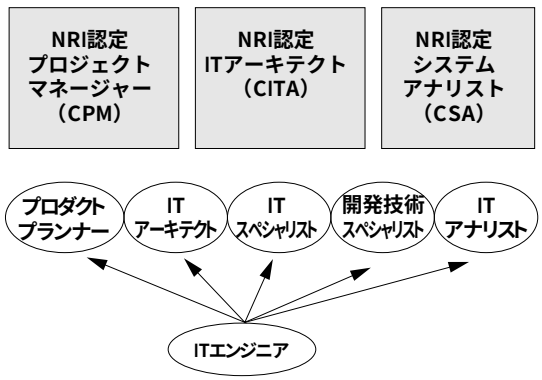


図８　NRIのテクニカルエンジニア職種分類

より多くなることを見据えると、システム基盤人材の育成が益々重要なものになる。

このような点を踏まえて、NRIのシステム系専門職は、アプリケーションエンジニア（AE）とテクニカルエンジニア（TE）に分かれ、全社教育体系も整備されているが、情報技術本部に所属するTE職種にあたる人材は図８に示すようなキャリアパス設定を基本形とした育成やローテーションを受けている。

ＮＲＩ社内では、CITA（認定ITアーキテクト）と呼ばれる認定制度があり難易度の高いプロジェクトには認定者の参画を必須としている。

３．処理方式設計の意義

システム基盤設計開発において鍵を握り、プロジェクト全体のアーキテクチャ設計でもある方式設計に課せられた使命は大きい。それだけにその活動はいくつかの特性を持っている。

（１）先行実施

インフラ規定を速やかに行ない、アプリケーション開発を円滑に開始できるようにするために、方式設計は全てに先行する必要がある。

（２）要件の確実な落とし込み

最終的には、開発したプログラムが目的の処理を実行し、その集合体としてのシステム運用を成立させるための機器選定、ソフトウェア選定、稼動環境構築がなされればよいが、その全ての拠り所は「要件」である。従来型のウオーターフォール開発であれ、RADやスパイラルという開発スタイルであれ、ビジネス要件、業務要件、処理要件、システム基

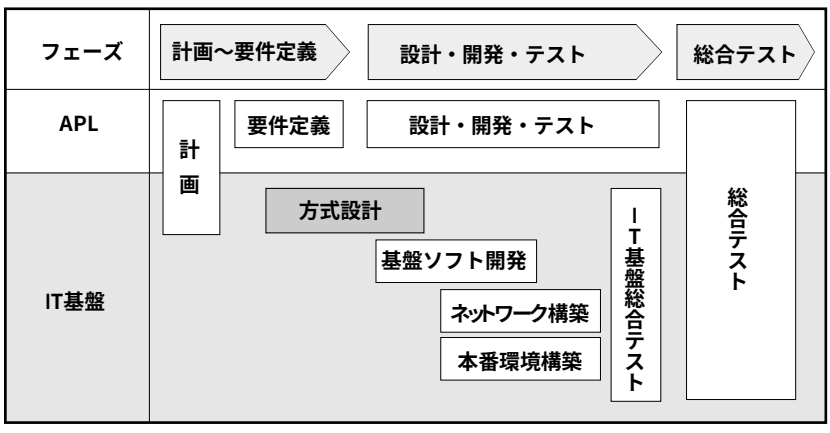


図９　システム基盤設計における方式設計の位置

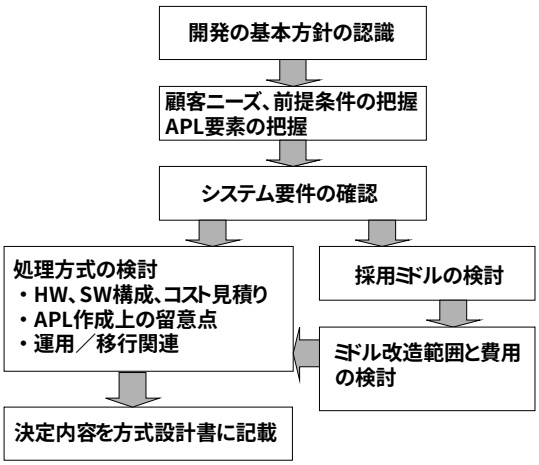


図10　方式設計の活動概要

盤要件といったような落とし込みは、確実に行われる必要がある。

（3）設計成果物のデリバリ

方式設計では、最終的に表２に示す内容でのドキュメンテーションを行なう。設計仕様を定めることに加えて、システム全体の詳細な機器発注、環境構築計画、運用環境構築やその実行体制についてもプロジェクト計画時点の内容との整合性を確保しつつ、詳細化することが求められる。つまり、個別開発や運用というモノ作り自体に必要な仕様の確定とともに、プロジェクト全体を円滑に推進していくための方向性を示し、プロジェクト・マネージャーのナビゲーションとなる役目も担っている。

表２　方式設計の内容

分類	内　容
構成	システム構成の規定
処理方式	メッセージ、データ処理の規定
接続方式	接続に関する規定
運用方式	運用に関する規定
性能	性能確保に関する規定
信頼性	障害等への対応の規定
拡張性	拡張への対応の規定
開発／テスト	開発テストの規定
維持／保守	稼動後維持／保守の規定
機器調達	初期購入機器類の決定
環境構築	環境構築日程の詳細決定
実施体制	インフラ構築活動体制の詳細決定

（4）処理方式設計の実行局面

処理方式設計は前述のようにプロジェクトの成功を占う重要な活動であるが、図11に示すような実施局面がある。

一つは提案段階であり、もう一つはプロジェクトが正式に開始した後の設計段階である。

①提案開始地点の方式設計

お客様へのシステムご提案段階では、いわゆる「見積り」が行なわれるが、この時点でのお客様の要件自体は必ずしも定まった状況にない。見積りを行なうためには、可能な限り要件を拾い上げ、想像して実システム構成に落としていくことによる、機器類、導入ソフトウェア製品、開発ソフトウェア規模等を積み上げていかなければならない。これは、短時間での方式設計活動である。

②プロジェクト正式開始地点

お客様へのご提案が承認された後に、プロジェクト計画策定と承認が得られれば、名実ともにプロジェクトは正式スタートとなる。

この場合のシステム基盤設計は、一般的なプロジェクト開始後の設計活動である。

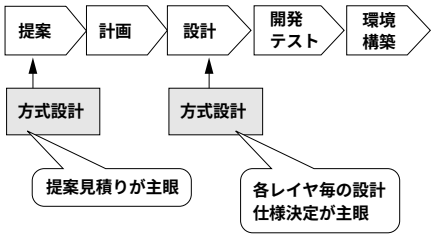


図11　方式設計の実施局面

（5）処理方式設計の特性

方式設計はアーキテクト足る人材により遂行される点での難しさを持つが、これ以外には求められる時間の早さと新技術の発掘や見極め活動を必要とする特性を持つ。

①遂行時間

システム基盤設計がアプリケーション設計に先行しなければいけないことも然りであるし、提案段階で行なう方式設計には週単位あるいは日単位の実行スピードが要求される。

②新技術／製品の見極め

プロジェクトによっては、新技術や新製品を採用する場合があるが、プロジェクト開始後にそれらの調査や評価を行なう場合もあろうが、常日頃からアンテナを張り、プロトタイプによる評価をしておくことも重要である。

4．方式設計時間短縮の期待効果

プロジェクトの命運を握る方式設計について、その時間短縮がどのような効果を生むのかを考える。

（1）開始基点としての捉え方

①上流工程の重要性

NRIのプロジェクト実行実績では、計画承認後のプロジェクト全体工程の時間比率は、図12のように表現できることが多い。これは、最初の設計の良し悪しがプロジェクト成功の命運を握っていることを意味する。つまり、全体時間配分上の上流にあたる工程自身の時間短縮意義は大きいものと考える。

②プロジェクト基点の見方

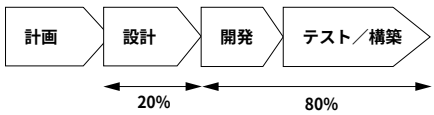


図12　プロジェクト全体の工程比率

システム開発プロジェクトの開始点はどのように見るべきであろうか。一般的な開発プロジェクトでは、図13で言えば、プロジェクト計画承認直後の時点となるが、実際のプロジェクトは、提案開始時点で始まっていると考えるべきではないだろうか。

お客様へのシステムご提案段階では、いわゆる「見積り」が行なわれるが、この時点でのお客様の要件自体は必ずしも定まっていないために、想像力を発揮させながらの短時間

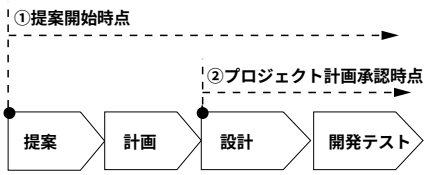


図13　二つのプロジェクト開始基点

方式設計が必要である。一方、プロジェクト計画承認後の段階では、一早くアプリケーション設計や運用設計に着手するための設計が主眼となる。

③プロジェクト離陸速度向上

図14に示すように、プロジェクトは提案段階から事実上の開始されているという認識に立った場合、目標とするシステム基盤の設計は、その充実度は高くないかもしれないが、提案段階において、見積りをする上では何らかの設計がなされているとみるべきである。

つまり、提案段階の見積りのための方式設計ではあっても、プロジェクト計画正式承認後に行なう設計工程としての方式設計の開始地点は、提案段階の方式設計を踏まえれば、ゼロからの出発ではないと言えるのではないだろうか。

提案段階の方式設計をより短時間で行なえるようにし、システム開発プロジェクトの開始タイミングを従来よりも早期に行なえるようにし、プロジェクト正式開始以降の設計設計段階での方式設計の開始にあたり、提案段階の検討内容が確実に引き継がれるようにすることによって、プロジェクト離陸までの時間と離陸速度は向上できるものと考える。

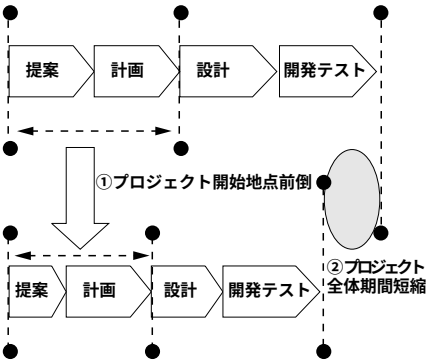


図14　方式設計時間短縮の期待効果

（2）プロジェクト遂行総合力拡大

NRI全体で見ると、個別プロジェクトでの方式設計時間短縮は、方式設計担当要員全体の遂行力拡大となり、全社ベースで見れば、

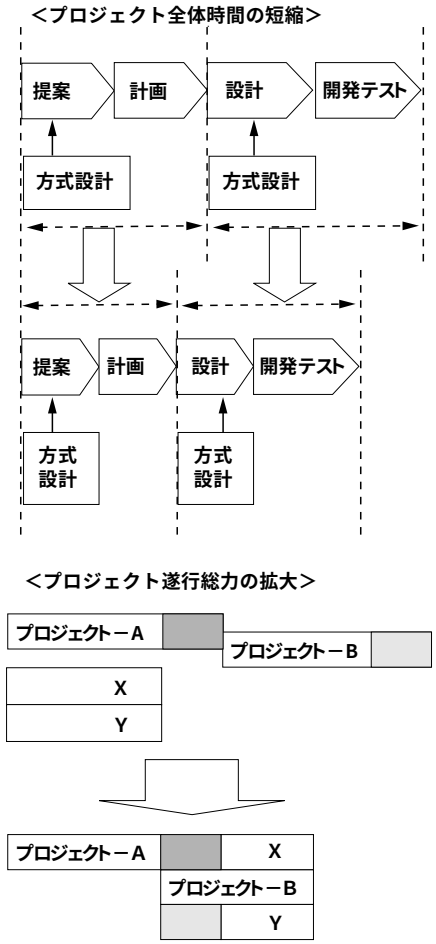


図15　方式設計時間短縮の意義

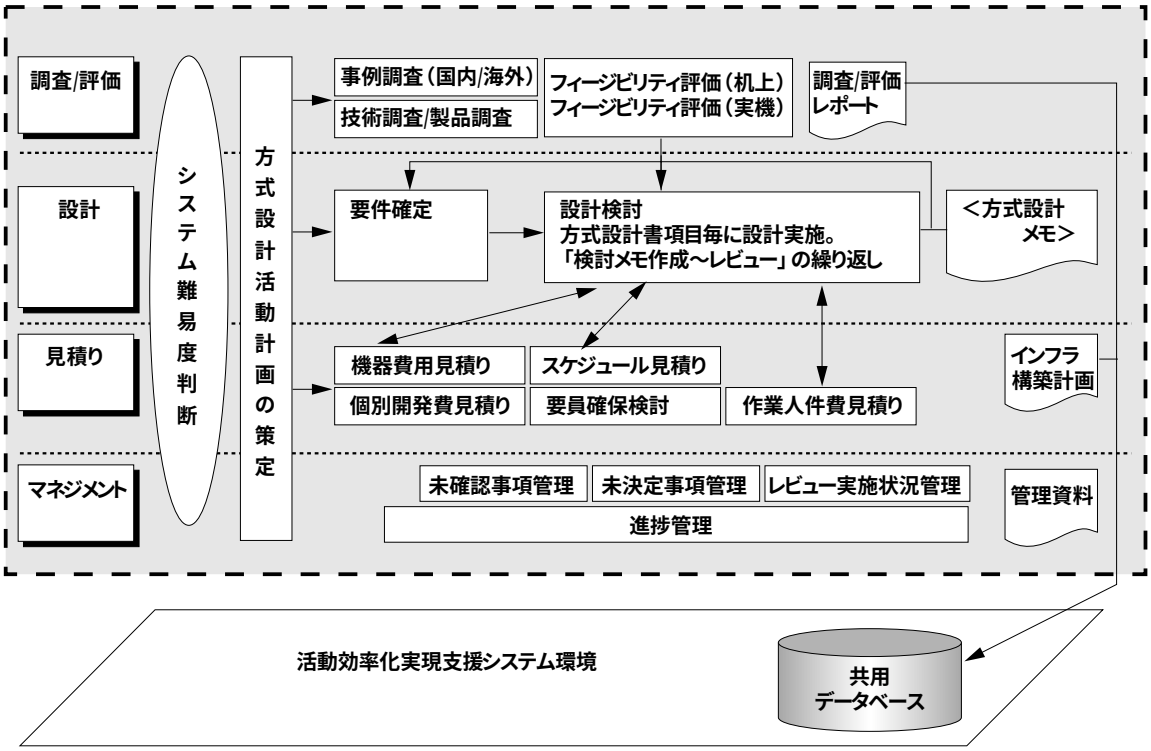


図16　「The 1/2」活動の概要図

より多くのプロジェクト実行が可能となりうる下地ができあがる。

5．設計高速化研究の位置付け

NRIでは、「The 1/2」と称する、属人性排除を目的とした工程最適化の取組みを2002年度より実施してきている。本稿で紹介する研究は、属人性排除を「可視化」というキーワードで、もう一歩推し進めることによるプロセス最適化を目標としている。

（1）方式設計高速化の意義

システム基盤設計の領域が広範囲で多くの

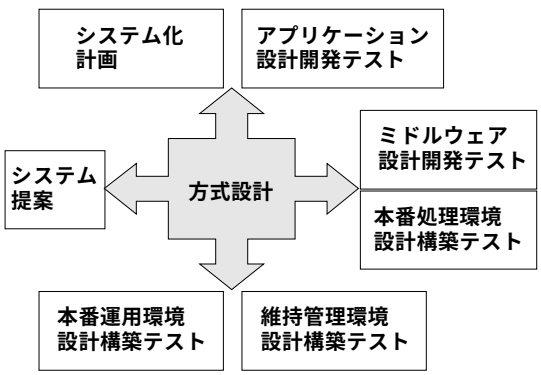


図17　方式設計と関連タスク

システム基盤人材を必要とすることは前述した通りであるが、全ての根幹は方式設計の内容に大きく依存するため、方式設計工程自体の高速化の意義は大きいと考えている。

（2）可視化の重要性

NRIの経験則では、方式設計工程におけるプロセス（タスク）数は500程度であることがわかっているが、そのやり方には個人差がある。例えば、家を建てる場合には、施主の希望を設計士が検討し、図面化していくが、成果物である図面を完成させるまでのプロセス（タスク）は、おそらく設計士個人ごとに異なるものであり、他人に説明することは難しい。設計された通りにモノを作ることとは異なる性質の仕事であり、知識と経験に左右される職人がなせる技となっているからである。

つまり、「設計」という基本的なプロセス（タスク）の形を描くことはできても、個々の要件に則した設計の進め方には、知識と知恵が必要であるため、全ての設計手順のパターンを明文化、図式化するのは難しいのである。

方式設計という職人的な技である見えない世界を前にして考え込んでも答えは出ない。一見、手の付けどころがない問題への取組みに見えるが、本稿で紹介する研究においては、「検討は可視化から始まる」という前提を置いている。

（3）方式設計高速化の仮説

設計活動上の定型的、形式的な部分を機械任せにすることは可能かもしれないが、それだけで時間短縮を実現することは難しい。繰返し述べているように、「人」の知識と経験に基づく知恵が要求される性質である点に光を当てるべき問題と考えている。いくら可視化ができたとしても最終的には「人」が行なうという点で何らかの具体策が欲しくなる。本稿で紹介する研究にあたっては、次のような高速化実現仮説を立てている。

①可視化対象の明確化

可視化の対象は、プロセス（タスク）に限らず、プロセス（タスク）に纏わる「情報」も含めることが重要である。

②可視化内容の理解

方式設計実行者は、プロセス（タスク）とプロセス（タスク）相互の関係とともに、「情報」についてもその種類、利用局面、性質、情報間の相互関係を理解することが重要である。

③「情報」利用環境の整備

この工程で必要とする情報にはさまざまなものがあり、必要な時に十分な内容が利用できなければ、情報自体の探索や必要内容に足るものに到達するまでの時間が非効率な結果に終わる。

6．MIT-DSMの活用

方式設計工程の可視化が実現し得る方法論、

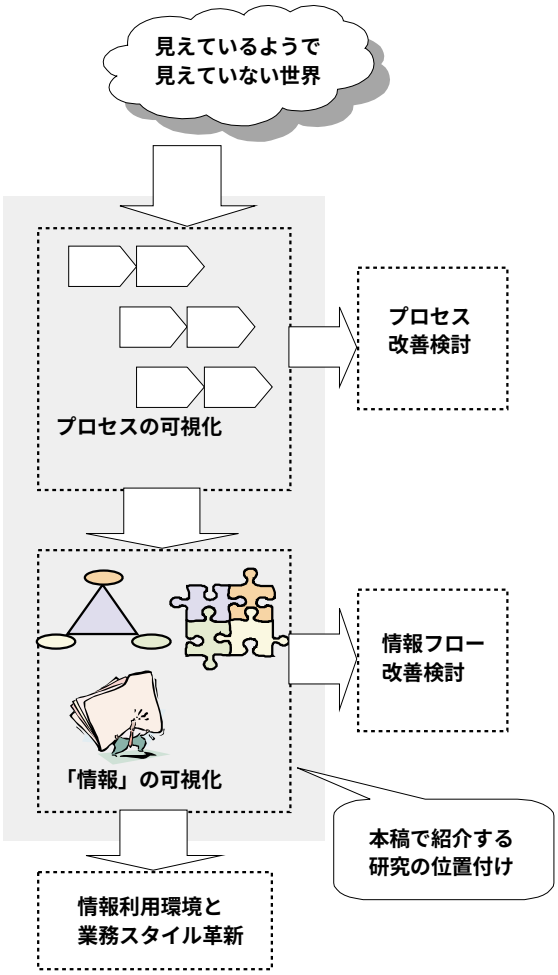


図18　方式設計可視化の意義

手法ないしはツールを探し求めて、今回の研究で利用したものがMIT-DSMである。

（1）MIT-DSMの誕生

時間浪費要因の一つは、「戻り」や「反復」の存在である。これは、必要なものと必要ないが不可抗力となって現れるものに二分できるが、必要か必要でないのかをどのように判断すればよいのであろうか。まずは、誰がみても理解できる、現状の真の姿を表現できることを可能にしなければならない。

本研究では、MIT（米国マサチューセッツ工科大学）で研究され、欧米の製造業で実導入ないしは導入評価が行われている、DSM（Design Structure Matrix/Dependency Structure Matrix）と呼ばれる方法論、手法に着目した。

DSMの基本理論は、1967年にカリフォルニア州立大学サクラメント校の名誉教授であった、Donald V. Steward, PhDによって考案された。

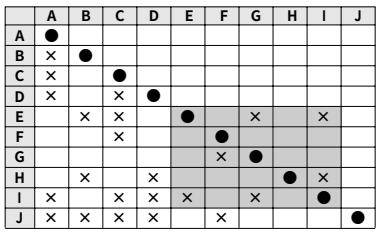
注：Donald V. Steward, PhDは後述するDSMツール製品開発企業に一つである、Problematics LLC.（本社：カリフォルニア州ナパ）の社長である

1981年に著書である、「Systems Analysis and Management：Structure, Strategy and Design」が出版され、その内容を基に、NASA（米国国家航空宇宙局）のJim Rogersが、「DeMaid」と呼ばれる、NASAの開発プロジェクトにおける可視化適用プログラムコードを開発し、その取組みが、Boeing、Lockheed-Martinといった航空機製造企業に紹介されたことが注目を浴びる契機になったとされている。

その後、1992年にMITがDonald V. Steward, PhDを招聘し、Professor Steven D. Eppingerと学生による応用研究が開始され、

以降、数回のDSMワークショップが開催されて現在に至っている。

MIT-DSMでは、図19のように相互の依存関係をマトリックス上に表現することが可能である。



対角線よりも下にある印は、下流工程作業のために必要な情報が上流工程から流れてくるものであることを示している。
対角線よりも上にある印は、下流工程からの情報が上流工程に流れることを示している。

図19　MIT-DSMによる状態表現例

従来型のフロー表記に比べるとマトリックス表現の明瞭さは明らかである。

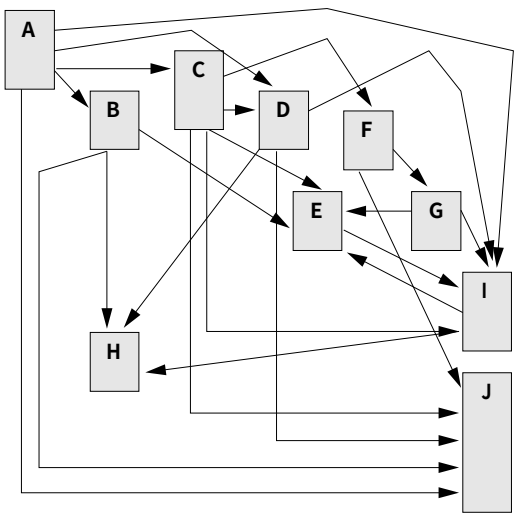


図20　従来フロー表記での状態表現例

（2）DSMの改善検討コンセプト

マトリックス上の表現において、対角線の上にある印は、「戻り」が発生していることを意味しているが、印が対角線よりも下に存在する状態に遷移することができれば、「戻り」や「反復」は解消されたことになる。

前述の図19に示したマトリックスは、改善検討によって、以下のような姿になる。

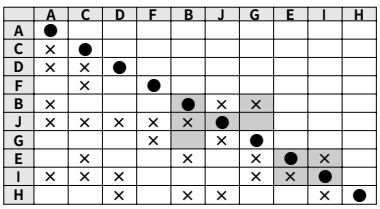


図21　図19の改善検討後マトリックス

これを前述の図20のフロー図相当の表記にしてみると、図22のに改善されていることがわかる。

この例では、印のすべてを対角線よりも下にすることはできてはいないが、フロー図を

見てもわかるように、検討前に比べてかなり見やすい状態になっており、DSMの活用のポイントは、次のようにまとめられる。

―自身から情報を出さないものは、最後の段階に持ってくる（図中のH）

―並行稼動できるものは極力そのように配置を変える（図中のD、F）

―反復関係が避けられない場合、それらを可能な限り集める
（図中のB、G、J及びE、I）

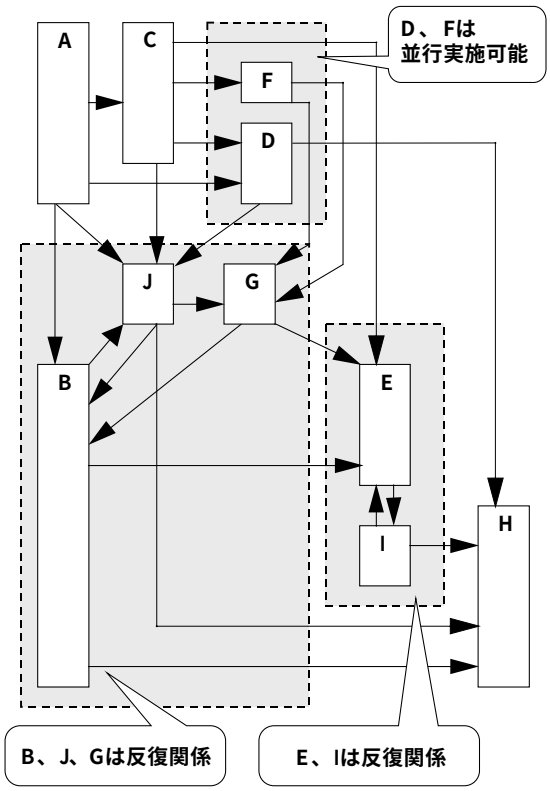


図22　図20の改善後フロー表記

（3）Partitioning

行と列を見た場合、どの実行に必要な情報を誰からも必要としないタスクと自身の出力する情報が誰にも作用しないタスクをマトリックス上から一旦外すことによって、検討範囲を絞り込んでいくための方法である。

その手順は以下のようになる。

①マトリックスの行に着目

どのタスクからもインプットのないタスク（行で見た場合スペースとなっているもの）を見つける。例ではタスクＦが該当

a	A	B	C	D	E	F	G
A			X	X			
B							X
C	X	X				X	X
D		X					
E			X			X	
F							
G				X		X	

図23　インプットのないタスクの扱い
（出所）The Design Structure Matrix　Home Page
http://www.dsmweb.org/Tutorial/tutorial.htm

②マトリックスの列に着目

・タスクFを上端に移動

・どのタスクへもアウトプットしないタスク（列で見た場合にスペースとなっているもの）を見つける。例では、タスクＥが該当

b	F	A	B	C	D	E	G
F							
A				X	X		
B							X
C	X	X	X				X
D			X				
E	X			X			
G	X				X		

図24　アウトプットのないタスクの扱い
（出所）The Design Structure Matrix　Home Page
http://www.dsmweb.org/Tutorial/tutorial.htm

③検討対象の絞り込み

・タスクＥを最右端に移動

・A、B、C、D、Gへの検討絞り込み

c	F	A	B	C	D	G	E
F							
A				X	X		
B						X	
C	X	X	X			X	
D			X				
G	X				X		
E	X			X			

図25　Partitioningによる絞り込み
（出所）The Design Structure Matrix Home Page
http://www.dsmweb.org/Tutorial/tutorial.htm

④Partitioningの完了

マトリックスは次のような反復ブロックを含む状態になる。

	F	B	D	G	C	A	E
F							
B				X			
D		X					
G	X		X				
C	X	X	X			X	
A			X		X		
E	X				X		

図26　Partitioning完了状態
（出所）http://www.dsmweb.org/
Tutorial/partitioning_path.htm

（4）Tearing

Tearingは、Partitioningの結果によって、現れるブロック内の回路状態を分析、数値化した上で、どこから順序変更を行なうかを考えるための方法である。

以下に、ループ回路の状態分析に使用する語句と値の例を示す。

Partitioningの結果、以下のような１、６、５、10、12という大きな回路の中に、Shunt（短絡路）と呼ぶ小さな回路があるものとする。

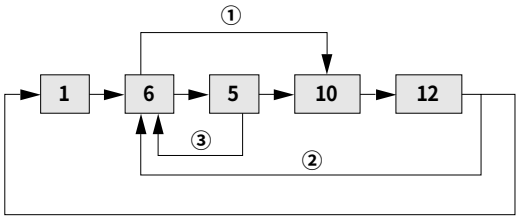


図27　Tearing説明用例題図

このループブロックに①、②、③で示す短絡路が存在する場合、タスク関係を次の項目によって分析する。

－NS（Number of Shunts）：
タスク間に存在する短絡経路の数

－NP（Number of Paths）：
当該タスクが途中地点となっている短絡路の数

－WS（the Weighted passes Statistics）：
全体ループ上のタスク総数長に対して、個々のタスクが含む短絡路長が占める比率

－B（Number of Beginning of Shunts）
当該タスクが基点となっている短絡路の数

－E（Number of End of Shunts）
当該タスクが終点となっている短絡路の数

この時の各値は表３のようになる。

WS＝Σ｛(ループブロック長－短絡路長)／ループブロック長｝
１：②（5－1)／5＋③（5－3)／5＝6/5＝1.2
６：短絡路はないので　0
５：①（5－1)／5＝4／5＝0.8
10：③（5－3)／5＝2／5＝0.4
12：③（5－3)／5＝2／5＝0.4

表３　Tearing用分析表

	短絡路毎の基点(B)と終点(E)			B	E	NS	NP	WS
	①	②	③					
1		↓	↓	0	0	2	2	1.2
6	B	E	E	1	2	1	0	0
5	↓		B	1	0	2	1	0.8
10	E			0	1	1	1	0.4
12		B		1	0	2	1	0.4

ループ構造をシンプルにするためのタスク順序変更を行なう際の着眼点がこの表からわかる。

WSが０のタスクは、そのタスク自身がループ構造内において、他のタスクよりも短絡路終着点になっていることを示す。

この例では、タスク６のWSが０であり、タスク６に着目して全体順序を入れ替えると、以下のような流れに変更できる。

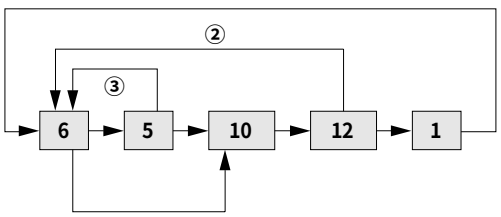


図28　Tearing実行後の例題図

タスク６を全体ループ中の通過点とするのではなく、全体ループの基点に変更することと、短絡路の終点も全体ループの基点になることによって、大きなループとその中の短絡路の流れがきれいになる。

７．設計高速化研究成果

今回の研究は、一般的に類を見ない、システム基盤の上流設計に焦点を当てたものであったが、MIT-DSMの利用効果も含めて、NRIにおける生産性向上施策を広く考えていく上でも大いに意味のある活動となった。

（１）可視化手段としてのDSM

①マトリックス表現による直感的理解

ガントチャート、PMBOK-WBS表でのタスク表現ではわからない、タスク相互の関係や反復箇所や範囲を視覚的に表現することができる。

DSMによる可視化によって、以下のようなタスク数があることを再確認することができた。

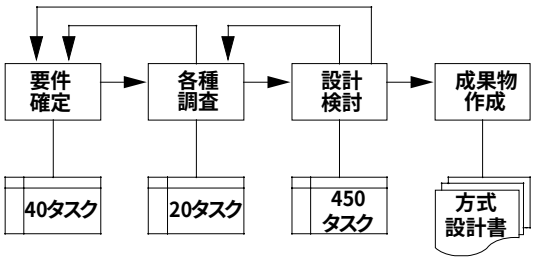


図29　DSM表現の対象となる方式設計工程のプロセス概要及び実行タスク数

②コンピューター利用解析が可能

マトリックス表現から、手順、プロセスの遂行や内容の性質や重みを数値化し、それを

コンピューター利用により解析することができる。

③方式設計工程に限定しない用途

プロジェクト計画立案と検証への応用が可能であり、表現させるデータによっては、ソフトウェア設計や組織体制検討への応用も可能である。

a Component-Based DSM
ソフトウェアコンポーネント設計に応用できる。

b Team-Based DSM
複雑な業務フローから成る世界での個人職能、担当組織の改善検討に応用できる

（2）全社プロジェクト遂行人材拡大

可視化されたプロセス（タスク）と情報の内容、相互関係を方式設計熟練者だけでなく、方式設計熟練候補となる若手人材に理解させることによって、若手人材がある程度は独力で設計を進めることが可能となる結果、熟練者が複数のプロジェクトに力を割くことが可能と成り得る。

これまで、NRI社内での方式設計プロセス説明資料は、方式設計書目次を参考にした大筋の流れを示すレベルに留まっていたが、今回の研究過程において、タスクとタスク間の情報の流れをDSM上に表現しつつ、従来のフローイメージで示すことができた。方式設計経験の少ない人材においては、活動タスクを丸暗記するのではなく、個々の意味や前後関係を理解するための有効な学習資料にすることが可能となる。

NRIでは、本研究過程で作成されたいくつかの資料を社内人材教育メニューに取り込むべく、講座化を検討中である。

（3）方式設計時間短縮目処

今回の研究内容から、方式設計時間の短縮は、概ね次のように実現しうる感触を得ている。

表４　方式設計時間短縮目処（想定）

プロセス名	従来	今後	備考
要件確定	30	20	3割効率化
各種調査	20	10	5割効率化
設計検討	40	30	2割効率化
成果物作成	10	10	

表中の数字は全体を100とした場合の割合

研究を行なうにあたって、「戻り」や「反復」の姿をイメージレベルで捉えていたが、ＤＳＭに表現することによって、それらの箇所や内容を具体的にすることができた。又、方式設計工程上のタスクは約500程度と見ているが、今後継続して行なう予定の研究において、DSMツール製品を利用することによって、現状から３割程度の効率化は図れるものと推定している。特に要件確定と各種調査の効率化が大きいとしている理由は次の通り

である。

①要件確定の効率化

当然のことであるが、要件確定対象内容は、最終成果物である方式設計書へ密接に関係するものであり、今回の研究によって、要件確定で着目すべき事項が工程上で関与しあう状況を表現できる内容を実行者が理解することで、要件確定精度を工程全体の視点で高めることができる。結果的に要件確定自体の手戻り頻度を下げることができる。

②各種調査の効率化

調査は技術調査、製品調査、システム自体の現行調査等、広範囲であるが、求める情報の所在を見つけ、自身の思考にマッチする質をもった情報ソースにたどり着くまでの時間短縮を実現することが重要である。今回の研究成果から、求められる情報の種類、性質、期待される利用方法、情報としての鮮度保持レベルといったことが整理できるので、物理的に情報が格納されている環境を作るレベルではなく、実際に使える状態の情報を作るための下地ができあがる。これまでの人づてに手繰り寄せていた情報へのアクセスがシステム的な機能装備との相乗効果で格段に効率化できる。

（4）提案段階の方式設計

今回の研究で対象とした方式設計は、正式なプロジェクト開始後の設計段階で行なう部分を対象にした「守り」の設計であったが、提案段階の方式設計は「攻め」の活動となるべきものであり、当社の今後の事業拡大の面でも、相応の創意工夫を要するものとして、今回の研究成果を活かすことを意識している。

（5）方式設計のパターン化

家を建てる場合に、同じ建坪であっても採用しうる間取りはさまざまであり、屋根、外壁、内壁の素材や屋内の機器についてもその選択肢は実に多いものとなる。システム基盤設計においてもこれと同様のことがありえる。ハードウェア機器選択やソフトウェア製品選択の幅が広いことや目的とするシステムに必要な処理のパターンに応じて、システム構成のバリエーションも多彩となる。

この点については、DSM上で完全に表現することは難しいと考えるが、パターン化はアーキテクチャ設計上も必須であることに間違いはなく、今回の研究成果で得られた内容を基に、パターン化を決める要素の洗い上げとそれを実現する製品や技術の選択肢を整理していく活動へ繋げられるようにしたい。

（6）活動体制のあり方

現状の方式設計の遂行体制は、優秀なチーフアーキテクトといえるリーダーと実行メンバから構成されることが多いために１人１プロジェクトという、著名な建築家に注文が殺到するような事態となっている。結果として、

NRI全社で見ると、方式設計を要するプロジェクトが方式設計担当者アサイン待ちとなってしまう恐れがある。チーフアーキテクトの育成を強化することに加えて、方式設計活動内容をより専門化することによって、１人１プロジェクトから１人複数プロジェクト実行体制の実現性検討も必要になるが、この検討には、DSMが表現できるデータタイプの一つである、「チームベース」での応用検討の価値が出てくることになる。

（7）設計環境改善

方式設計個々のタスク遂行時に必要な各種情報の属性や利用用途、別プロジェクトで活用できるような情報加工上のポイントをより具体的に検討する上でDSMで表現する「情報の流れ」の内容が参考になる。

情報を対象とするという点では、多くの企業で導入や実例が出てきている、EIP（Enterprise Information Portal）や、KB（Knowledge Database）の構築に類するものとなるであろうが、DSMで表現しえたタスクと情報項目、内容を基に、「いつ」、「どこで」、「誰が」、「どのような情報を」、「どのように利用したいか」といった点を具体化していくことによって、方式設計工程における貴重なノウハウ等が管理、活用されることに繋げられる。

この取組みは、情報技術本部の一部で開始されているところであるが、留意しなければならないのは、環境や仕組みの整備だけでは意味がないという点である。人は情報活用には非常に興味を持つが、他に有益な情報を提供することには積極的でない場合も多い。情報提供のインセンティブやモチベーションを保つための施策の検討があわせて必要になる。更に、情報は生ものであるため、鮮度をいかに保つかということが重要となる。従って、情報そのものの意味や性質や利用価値を理解した上で蓄積、廃棄、更新することを専門的に行なう人材配置の検討も必要になる。

８．おわりに

DSMは製造業の多くで、その効果をあげているが、ソフトウェア開発プロジェクトへの適用例はほとんどない状況と言える。本研究内容を振り返ると、DSMという方法論やツール利用があるとはいえ、可視化ということ自体は特に目新しいものではない。

しかし、ソフトウェア開発プロジェクトの大半が人手による活動である以上、「必要な情報」に着目して仕事を組み立てていくことの重要性は当分変わることはない。

マニュアルになっているものを暗記し、それを使うだけではアーキテクチャ設計ができるようになるものではないことは自明であるが、今回の研究成果は、ソフトウェアアーキテクト足る人材に求められる資質や仕事への取組み姿勢を改めて考えさせられる面もあった。今回の研究過程で作成された以下の資料

類が、実際のシステム開発プロジェクトで利用されはじめていることは嬉しい限りであるが、アーキテクトの育成という課題に向けても今回の研究成果を活かし、NRI自身のためだけではなく、NRIのお客様にとってもシステム基盤設計開発において有効なものとなるよう、今後も研究活動を継続していきたいと考えている。

－システム難易度判定表
－システム基盤要件整理／チェック表
－方式設計工程ＷＢＳ
－システム基盤見積り項目表

参考文献

[1] Project Management Institute, " A Guide to the Project Management Body of Knowledge（PMBOK Guide）2000 Edition ", Project Management Inst Pubns, 2000

[2] Mint（経営情報研究会）「図解でわかるソフトウェア開発のすべて」日本実業出版社、2000

[3] マーク・スウェル・ローラ・スウエル著／倉骨彰訳「職業としてのソフトウェアアーキテクト」ピアソン・エデュケーション、2002

[4] 経済産業省 ITアソシエイト協議会資料「政府調達のためのIT専門家について　～ITアソシエイト協議会中間報告～」
http://www.meti.go.jp/feedback/downloadfiles/i21227mj.pdf

[5] Donald V. Steward, " Systems Analysis and Management: Structure, Strategy and Design ", McGraw-Hill, 1981

[6] James L. Rogers, " Knowledge-Based Tool for Decomposing Complex Design Problems. ", ASCE Journal of Computing in Civil Engineering, Volume 4, No. 4, October 1990

[7] The Design Structure Matrix Home Page
http://www.dsmweb.org/index.html

[8] Problematics Home Page
http://www.problematics.com