連載　プロマネの現場から
第 64 回　大規模システムの「複雑さ」を考える

蒼海憲治（大手 SI 企業・金融系プロジェクトマネージャ）

最近、大規模システムの開発が難しいとはどういうことなのだろうか、ということを少し考え直しています。中小規模のシステム開発と違って、大規模システムでは、意思決定に時間がかかり、コミュニケーションロスが大きく、生産性が著しく悪化する、ということを常日頃体感しています。じゃあ、その構造はどうなっているのか、と自問自答してみると、１５年余り前の１９９０年代後半に電気学会の情報システム技術委員会における巨大システム調査専門委員会（高橋勝委員長）において、「複雑さ」に焦点を当てて、大規模システムの難しさの構造を明らかにした研究結果と、その当時、私自身が参画していた大規模プロジェクトのプロマネをされていた岡村正司さんの著作に立ち返って考えていることに気づきます。

前者については、芳賀正憲氏のメルマガ『連載 情報システムの本質に迫る　第１１回 複雑さ、コミュニケーション、能力開発 あるいは プロジェクト管理の基礎』（＊１）において、芳賀氏が既に的確に整理し紹介済みのことなのですが、まさに≪情報システムに関し、複雑さの観点からプロジェクト管理の本質の解明≫（＊１）されたものであることもあり、改めて、この成果を辿り直してみようと思います。

そもそも、情報システムが大規模化・複雑化してきている背景は、以下のプロジェクトの要請にある、と指摘されています。

≪プロジェクトの目的とするところが、

①部門間での情報のリアルタイム共有化
②業務の全社統合化・部門間の業務の統合化
③部分最適から全体最適へ
④業務の商品単位から顧客単位への革新
⑤企業間の有機的連携・情報共有≫（＊２）

等にあること。

≪これらの目的はすべて各サブシステム間を密結合とし、
結果として巨大な集中化システムを構築する恐れがある。≫（＊２）

システムが大規模化・複雑化するのは、時代の要請ですが、その結果、大規模システムと中小規模のシステムの実績を整理すると、開発規模によって、生産性が大きく異なるという事象があります。この事象をもたらす要因として、大きく２つの仮説を立てています。

１つは、システム開発の規模が大きくなると、システムの複雑さが急激に増大することであり、もう１つは、プロジェクトの組織が大きくなると、組織の効率が急激に低下すること、にあります。

（仮説１）システム開発規模が大きくなるにつれ、システムの「複雑さ」は急激に増大する

この構造を表すものとして、「複雑さ」モデルを検討します。
「人間には処理能力（こなしうる仕事の量（複雑さ））」と「理解能力（制御し得る複雑さ）」の二つの面があります。
また、「規模」を示すと考えられる要因としては、データ量、データエントリー数（属性の数）、仕様量（画面数、帳票数）、ホストコンピュータ数（サーバ数、ノード数、ディスク数）、クライアント数（トランザクション数）、稼働時間を挙げます。
ただし、一見沢山の変数があるように見えますが、大本にあるのが、「データエントリー数」です。データエントリー数が増加すると、項目定義量、入出力の画面数、帳票数、バッチ本数、ＤＢ数に直接影響します。そして、入出力の画面数、帳票数、バッチ本数の増減は、機能仕様量⇒プログラムモジュール数⇒プログラムコード数、に影響し、また、ＤＢ数の増減は、状態を管理すべきエンティティ数⇒内状態を共通で管理すべきエンティティ数⇒エンティティの状態の総組合わせ数に影響します。さらに、それらの変数の増減が、状態管理仕様量及びドキュメント量に影響し、「複雑さ」度合いを規定することになる、というモデルを提示されています。

「状態の総組合わせ数」とは、例外処理・チェック処理の数を含みますが、項目数の増大とともに、分岐・判断条件が増大します。その上で、「開発対象システムの複雑さ」を表す方法として、横軸に「データ項目数・状態管理エンティティ数」、縦軸に「複雑さ」を採ります。
「データ項目数・状態管理エンティティ数」の増加と、「複雑さ」度合いの増加を見て驚くのは、項目数の増加に対して、等比級数的に増えていくことです。
参考値となりますが、規模が２倍になると、「複雑さ」度合は２０倍となり、規模が４倍になると、「複雑さ」度合は４００倍となるという、驚くべき結果が示されています。数千項目以内のシステム開発プロジェクトであれば、個別チーム任せでも、プロジ

ェクトを推進することができるかもしれません。しかし、項目数が１万項目以上となると、データの標準化・管理を一元的に行うチーム・組織が必要となります。

（仮説２）開発プロジェクトの組織が巨大化するにつれ、効率が著しく低下する

この構造を表すものとして、「組織パワー」モデルを検討します。
組織のパワーとは、単純に、要員数に、個々人の能力を掛け合わせたものとはなりません。
「組織パワー」モデルとは、プロジェクトの「組織パワー」は、「個人の処理能力の総和」から「人と人とのコミュニケーションに要するパワーロス」を引いたものになるというものです。
また、個々人の処理能力は、要員の理解可能の範囲内に収まる場合と理解可能限界を超える場合とで、処理できる仕事量が大きく異なります。
理解可能限界を超えた途端、個々人の処理能力は、急激に悪化します。

つまり、仕事量が要員の理解可能限界を超えた場合、要員は、①自ら発想することを止め、②指示待ち人間となります。
その結果として、①コミュニケーション負荷が著しく増大し、②仕事の品質が著しく低下します。この２つの前提条件を「組織パワー」モデルは置いています。

要員が理解可能範囲にある場合は、「個人の処理能力の総和」から通常の「人と人とのコミュニケーションに要するパワーロス」を差し引いて算出します。一方、要員の理解可能限界を超えた場合、「個人の処理能力の総和」から、規模によって幾何級数的に増大する「人と人とのコミュニケーションに要するパワーロス」を差し引いて算出します。

プロジェクトが、要員の理解可能限界を超えた場合、問題解決能力失調症というべき、プロジェクトの「課題は常に２０人という多くのＳＥを集めないと解決できない」という事態が出現します。そして、「組織パワー」モデルは、この生産性悪化の現象を説明するモデルとなっています。

この問題を解決し、大規模プロジェクトを実施する方法が、疎結合による「サブシステム分割」にあります。
サブシステムとは、ユーザに提供する業務機能を指し、特定の機能が特定のデータベースを更新し、独立した運用が可能なシステムのことです。

サブシステム内のデータベースは、サブシステム内部の機能からしか、登録・更新・照会がされないこと。他のサブシステムのデータベースを、直接、更新していない点が、サブシステムの独立性の観点から重要になります。

開発側からのサブシステム分割する理由には以下のものがあります。

１．開発の生産性向上

開発規模の増加による開発生産性の低下を防ぐため。サブシステム分割により、中小規模のシステムに変える。

２．保守の生産性向上

後に述べる機能別開発ｖｓ商品別開発の方針選択のように、サブシステムの分割の仕方によって、運用システムを含めた総コストを削減できる。

３．コミュニケーション負荷の軽減

疎結合による「サブシステム」の場合、他のチームとの調整案件が少なくなり、比較的独立したスケジュールで開発を進められるようになる。

４．要員への要求スキルの低減

「サブシステム」毎に、要求スキルを限定することによって、複数のスキルを保有する　要員の調達を回避することができるようになる。

つまり、上手にサブシステム分割することにより、これらのメリットを享受することができます。そのことによって初めて、大規模システムを円滑に進めることができるようになります。

では、いかにして分割すればよいのでしょうか？

岡村正司さんの『２５のセオリーで学ぶシステム設計の必修スキル―基盤システム、性能・品質、運用から業務プロセスまで 』（＊３）に、ＤＯＡ（データ中心設計）準拠の開発標準を採用した場合の事例が紹介されています。

ＤＯＡによる開発においては、ＤＦＤ（データフロー・ダイヤグラム）や業務階層図を用いて、サブシステム分割を表現します。

超大型プロジェクトでは、平均すると階層は５段階程度になり、実装される最下位レベルの機能数は数千、個々の機能の規模は平均で１５００～２０００ステップになる、といいます。

全体で数百万ステップのプロジェクトが、数十万ステップのサブシステムのプロジェクトとなります。

また、その際のサブシステムの分割の方針が大切になります。

１．モジュール結合度とモジュール強度

サブシステム分割は、プログラムのモジュール分割のイメージに対比することができます。

モジュール分割の方法は多様ですが、良いモジュール設計とは、他のモジュールが変更された場合でも、仕様変更の影響を受けにくいこと・変更する必要がないこと、そのモジュールを部品として再利用できることにあります。

つまり、モジュールの独立性が高いものが優れたモジュールであるといえます。

このモジュール分割の評価基準として、モジュール強度とモジュール結合度があります。　モジュール結合度には、悪いものから良いものの順に、

①内容結合、②共通結合、③外部結合、④制御結合、⑤スタンプ結合、⑥データ結合があります。

また、モジュール強度には、悪いものから良いものの順に、

①暗号的（偶然的）強度、②論理的強度、③時間的強度、④手順的強度、⑤連絡的強度、⑥情報的強度、⑦機能的強度があります。

良いモジュール設計とは、他モジュールの仕様変更の影響を受けにくくする疎結合を実現していることなので、モジュール結合度が弱く、モジュール強度が強くある必要があります。

つまり、なるべく少ないデータの受け渡しでつながる「⑥データ結合」と、すべての要素が１つの機能を実行する「⑦機能的強度」であることが望ましい。

サブシステム分割にあたっても、モジュール結合度とモジュール強度の考え方は援用すべきです。

２．ＣＲＵＤマトリックスによる機能単位の特定

ＤＯＡに基づく開発標準を採用した場合、業務分析にはＤＦＤ（データフロー・ダイヤグラム）、データ分析にはＥＲＤ（エンティティ・リレーションシップ・ダイヤグラム）が用いられます。

サブシステム分割にあたっては、このＤＦＤの機能名と、ＥＲＤのエンティティ名を基にしたＣＲＵＤ（作成・参照・更新・削除）マトリックスを策定します。

縦軸に、業務機能を展開し、横軸には、エンティティ名を展開します。

このマトリックスの中に、エンティティの発生するＣ（作成・発生）を、記述する。そして、Ｃで集められた集合体を括り出して、サブシステムの候補として抽出する。

データベースの実装設計等を通して、サブシステムの分割は何度か見直されることになる。その際、考慮すべきことは、エンティティは特定のサブシステムで発生し、消滅するということ。サブシステムに、エンティティの管理責任を持たせる「エンティティのオーナー制度」を設けることが重要になります。

「エンティティのオーナー制度」によって、モジュール結合度は、「⑥データ結合」となり、サブシステムの独立性を確保することができるようになります。

３．機能別開発か、商品別開発か

歴史的に見ても、１９７０年代以来、機能別開発という「④業務の商品単位から顧客単位への革新」がなされることによって、システム全体の「複雑さ」は増してきています。

≪すべての商品機能を整理し、その機能単位に開発を行い、全商品の機能をそのなかに実装、商品横断の機能を共通プログラムとして開発した。≫（＊３）

その結果、開発量は削減できた、といいます。当初は、開発量の削減は、保守作業の効率化、保守工数削減となるという目論見がありました。

しかし、実際はそうでなかったといいます。

≪開発量は削減できたが、保守が困難になった。

仕様を追加したときに、その影響範囲を調べて変更部分を検出し、その後のテストを実行するのが大変だった。数本のプログラムを修正しただけなのに、全商品に与える影響を調査してテストする羽目に陥ったこともあった。特定の人しか保守できないシステムになった≫（＊３）

と、指摘されています。

大規模プロジェクトの成否は、工程別、サブシステム毎に分割し、中小規模のプロジェクトの集合体にすることができるか、にかかっています。ただし、どんなに細分化しても、すべての機能を均等に小さく分割することはできません。

特に、システムの「企画プロセス」から、サブシステム分割するまでの工程、及び、個別チームに先だってプロジェクト計画やＷＢＳ等を策定する「管理プロセス」は重要になります。つまり、下流工程よりも、上流工程に、スキルの高い要員をアサインすること。また、サブシステム分割後も残る難度の高いサブシステムやタスクに優秀層をアサイン、確保し続けることができるかが、プロジェクトの成否を握っています。

また、サブシステム分割で、大規模プロジェクトの複雑さがすべての解決されるわけではありません。

≪大型プロジェクト特有の会議の多さ、仕様確定の困難さ、
　テスト計画立案作業量の巨大さとテストの難しさなど≫（＊３）
が残っており、これらは大規模システム開発の難しさの大きな部分を占めています。

（仮説２）のプロジェクトの「組織パワー」モデルで説明したように、組織のパワーは、大規模になるにつれて、「個人の処理能力の総和」から差し引かれる「人と人とのコミュニケーションに要するパワーロス」が大きくなります。サブシステム分割により、要員の理解可能限界を超えないようにプロジェクトを作った場合でも、分割したサブシステム間のコミュニケーションロスは発生します。このコミュニケーションロスをいかに小さくするかということも重要な問題です。そのため、ステークホルダを含めて、コミュニケーションプロセスの計画や管理を徹底することで、コミュニケーションロスを減らす工夫も必要になります。

最後に、新システムのイメージを構築できる人、業務に精通している人がいてこそ、ビジネスを的確に反映した概念モデルを作ることができます。そのような要員の確保と育成こそ、大規模システムの構築プロジェクトにおいては、最優先課題の一つだと考えています。

（＊１）『連載 情報システムの本質に迫る　第１１回 複雑さ、コミュニケーション、能力開発 あるいは プロジェクト管理の基礎』情報システム学会メールマガジン　２００８年４月

（＊２）『講演・研究会　巨大情報システムの将来を探る（平成１０年４月１６日）』「巨大システムの”複雑さ”を考える」　電気学会の情報システム技術委員会・巨大システム調査専門委員会

（＊３）岡村正司『２５のセオリーで学ぶシステム設計の必修スキル―基盤システム、性能・品質、運用から業務プロセスまで 』日経ＢＰ社　２００６年刊