システムズエンジニアリングの上流設計プロセスについて

株式会社コギトマキナ 代表取締役　鈴木 尚志

1 はじめに

1.1 本稿の背景と目的

近年、様々な複数のシステムが相互に結合された新しいサービスや価値が生まれている。これらのシステムの開発を成功させるためには、これまで行ってきた単独システム開発とは異なる開発視点が要求されることも多い。

システムズエンジニアリングは、対象とするシステム（System of Interest）全体を見渡すことにより、システムの開発や運用などを成功に導くための工学である。学問としてのシステムズエンジニアリング自体は、新しいものではないが、近年の複雑であったり大規模であったりするようなシステムの成功のためには重要であると近年注目を浴びている。

一方、これらのシステムの成功には、組み込まれたソフトウェアが大きく寄与していることも多く、そのため、ソフトウェアエンジニアがシステムズエンジニアとして活動することを期待されることも多くなってきている。

そこで本稿では、ソフトウェア・エンジニアリング領域で活動される諸氏を主な対象とし、まずはシステムズエンジニアリングプロセスのうち上流の「システム設計活動」を紹介する。

また、最近は、便利さのため「モデル」を用いるシステムズエンジニアリングであるモデルベースシステムズエンジニアリング（以下MBSE）に脚光が当たっている。本稿後半ではモデル化言語SysMLを用いたMBSEについても触れる。

2 システムズエンジニアリングにおけるシステム設計活動とは

2.1 システム設計活動の目的

システム設計活動の目的は様々な利害関係者のシステム把握に関する合意である。

主な合意の内容は、以下の２点に関するスコーピングであり、詳細な実装方法ではない。

1. そのシステムが解決する課題（What）
2. 課題に関する解法（How）

この目的を達成するために、対象システムのスコーピングを定義した最終成果物と合意を目的としたものも含む、議論を効率よく行うための資料としての補助的成果物という２種類の成果物が存在する。

2.2 システム設計活動の概要

システム設計活動は、大きく２つの活動、要求分析とシステムアーキテクチャの構築から構成されている。

成果物であるシステムアーキテクチャは、システムの企画時から、開発、運用、そして破棄に至るまで、すべてのライフサイクルにおける活動の基盤である。

2.2.1 合意について

システム開発に当たっては様々な議論が行われる。これらの議論の目的や内容は多分野にわたる（Multi-discipline）ことも多い。議論の例を挙げれば、要求や機能の追加や変更、新規テクノロジーやCOTS品の採用や既存システムとの共存、開発期間の短縮を含む様々なコスト削減、運用時や廃棄時の環境への省影響性、クリティカル性能などへの対応、相互運用性などである。

また最近ではとくに、自動車業界におけるISO26262のような機能安全規格のトレーサビリティや障害対応性などの安全設計、またセキュリティへの考慮も多くなっている。

これらの議論は、ある課題についての複数の解法のトレードオフを考慮しなければならない部分で紛糾することが多い。そのための、トレードオフスタディについての資料を作成することもシステムズエンジニアリングにおける重要なミッションの一つである。

2.2.2 システム設計活動を適用するタイミング

また、よく誤解されているところだが、システムズエンジニアリングの上流プロセスは、開発初期時にのみ実行されるものではない。

システムのライフサイクルは、概念検討から始まって、

開発、運用、利用、そして廃棄に至る。

しかし、現実として、システムの開発は、完全に上流で要求を固め、アーキテクチャを一度だけ構築するという、いわゆる理想的なV字プロセスではない。事実、システムのライフサイクル後期の製造や運用、保守の際に発見され、考慮された事項が、設計や開発、すなわちライフサイクル上流の成果物に反映されることも多い。システムズエンジニアリング各活動は、「どのライフサイクルにおいても「手戻り」が発生する可能性はあり、そして、システム成功のためには「早期の迅速な手戻り」は重要である」という思想を根底に持つ。

3 システム設計活動の詳細

ここではシステム設計活動の中心である「要求分析」「システムアーキテクチャの構築」という２つの活動について紹介する。なお、この基本的な２つの活動はウォーターフォール的に行われるわけではなく、開発ライフサイクルを通じ、何度も実行されるイテレーティブな活動であることをあらためて記しておく。

3.1 要求分析

システム設計活動は、通常、そのシステムを使用する、あるいは購入する顧客の要求を獲得することから開始される。

しかしシステムの使用者や操作者のようなユーザが、開発者がすぐに開発可能なような完全な要求を定義することは不可能である。また、提示された要求が、ほかの利害関係者の提示した要求と不整合を起こすことや、開発者自身が開発者制約と言われる要求を入力することも多い。

要求分析とは、つまり、このような様々な利害関係者の要求を、設計者にとって十分な品質を持った技術的仕様へと変換する作業である。

要求分析の作業は、２つの側面を持つ。１つはシステムをブラックボックスとして定義することで、対象システムとシステム外部との間のシステム境界を明確にするという側面。もう１つが、対象システム内部を明確に記述するというホワイトボックス的側面である。

3.1.1 要求の種類と領域

システムへの要求は多岐にわたる。主な要求の種類として

目的を実現するために持つべき機能
目的を実現するために持つべき性能
システムと外界との外部インターフェース
動作環境
目的を実現するために持つべきリソース
システムに対する物理的要求（容積、サイズ、重量など）
設計（設計基準など）
プロセスと手法（製造プロセスや品質管理など）

などが存在している。

その一方で、要求分析の技術領域もとても広範である。これらは様々なシステムズエンジニアリング標準においても様々な定義がある。

そこで、以下にシステムズエンジニアリングの世界標準の一つであるIEEE1220のシステムズエンジニアリングプロセスにおける要求分析タスク領域の例を挙げて要求にかかわる領域を簡単に紹介する。

IEEE1220における要求分析タスクには以下の15の領域がある。

1．顧客の期待
2．プロジェクトと会社の制約
3．外部制約
4．運用シナリオ
5．効果指標（MoE）
6．システム境界
7．インターフェース
8．利用環境
9．ライフサイクルプロセス概念
10．機能要求
11．性能要求
12．運用モード
13．技術的性能指標
（Technical Performance Measures）
14．物理特性
15．ヒューマンファクタ

そして上記15領域の要求分析タスクを入力として、「要求ベースライン統合」タスクが実行されることとなっている。

3.1.2 要求分析の手法

要求の種類も多岐にわたり、考慮すべき領域も広範であるため、現実には要求分析作業の多くが属人的なものである。効率的に要求を抽出するためには、経験的に様々な手法がある。ここではその一部を例示する。

3.1.2.1 ブラックボックス的な要求分析手法

ブラックボックス的な要求分析手法は、主にシステム境界を明確にするという側面のために行うことが多い。そのためにはそのシステムの使用方法（ユースケース）の文書化や、そのシステムのコンテキストやミッションプ

ロファイルなどを記述した運用コンセプト文書の作成などを行うことによって要求を洗い出す。

これらの文書作成や要求の洗い出しのためにはユーザや近似システムの経験者などの参加が基本的に必須である。

3.1.2.2　ホワイトボックス的要求分析手法

ホワイトボックス的な要求分析手法は、対象システム自体を明確にするという側面のために行われることが多い。そのために、洗い出しを中心としたブラックボックス的要求分析手法とは異なり、明確に「システムの機能」について、様々な観点から考慮することが多くなる。

ブラックボックス的要求分析において、ユースケースなどで洗い出された機能をより明確にすることを機能分析と呼ぶこともある。ここで言うユースケースはあくまで、その「システムの使い方」であり、「ユースケースが機能を使用する」と考えれば理解しやすい。そして、この機能分析が、要求分析とシステム設計を接続するための軸となる活動であることに注意されたい。機能分析を通じて、システムは何を行うべきであり、行うべきでないか、という判断が決定され、明確化される。

機能を明確にするためには、従来からFFBD(Function Flow Block Diagram)のような機能フロー図や、状態モード分析、規定外の条件への対応を規定する範囲外動作分析などを行うことも多い。

一方、属人性を可能な限り廃するために、検証方法を先に定義することによって要求のヌケモレなどを検出することもある。そのため、最近では形式的仕様記述や構文解析による要求抽出などの形式的方法が用いられることもある。これもホワイトボックス的要求分析の一つである。

上記の活動などから得られた要求を統合したものがシステム仕様書である。作成の際には、各要求の妥当性、明確性、完全性、無矛盾性、単一性、検証可能性、変更容易性、追跡可能性、重要度などの要求そのものの品質についても意識しなければならない。

3.2　システムアーキテクチャの構築

システムズエンジニアリングにおいて「システムアーキテクチャの構築」は、トップダウン・アプローチである。

「システムアーキテクチャの構築」活動は、目的としている機能を規定のパフォーマンス制約の制限内で提供可能にするようなシステムの物理アーキテクチャの開発に重点を置いている。

ここでは要求分析の成果物であるシステム仕様書を入力とし、成果物としてシステムアーキテクチャ仕様書と補足文書を作成する。ここで言うシステムアーキテクチャは、デバイスやチップレベルのハードウェア設計や、物理的なプラントや社会インフラ開発などを対象としたものであり、明らかにソフトウェアアーキテクチャと言われるものよりも抽象度が高いものを指す。

また、システムアーキテクチャでは対象となるシステムと、関係するシステムとの相互作用も定義する。

アーキテクチャの定義は様々であるが、システムズエンジニアリングのためのシステムアーキテクチャでは、システムの機能やサブシステム構造のみならず、システムの安全性、信頼性、可用性、保守性、セキュリティといったシステムを横断するような重要な関心事についての定義は必須である。

3.2.1　観点と要素の割り当て

システムアーキテクチャの構築活動は、システムの目的を達成できるように独立したアーキテクチャ要素同士を関連付ける活動である。システムアーキテクチャには関心事ごとの様々な側面(ビュー)がある。その側面を理解するための観点(ビューポイント)がある。アーキテクチャ要素の種類も、観点ごとに存在し、例えば要求、機能、振る舞い、構造、フロー、プロパティなどと多岐にわたる。

アーキテクチャ要素同士の関連付けは、同種のもの同士とは限らないことに注意が必要である。異種のアーキテクチャ要素同士を割り当てること(アロケーション)によって、異なったアーキテクチャの側面間の整合性が構築されることになる。

3.2.2　システムアーキテクチャの構築活動の手法・手順

「システムアーキテクチャの構築」は、アーキテクチャ分析とアーキテクチャ設計という２つの大きな活動から構成される。

3.2.2.1　アーキテクチャ分析の活動

アーキテクチャ分析はトレードオフスタディと呼ばれることもある活動である。この活動によって、システムアーキテクチャ設計のコンセプトが構築される。

なお、現実には完全なアーキテクチャ分析が実行されることは少なく、既存アーキテクチャを使用するときにはその一部が省略されることも多い。

主要なシステム機能を明確化する

アーキテクチャ分析の活動は、基本的にシステム機能分析を中心として行われる。そして、概念的にはシステムの主要なシステム機能は、副次的な機能サブセットや論理的なシステム構成要素で構成されているため、これらを洗い出す必要がある。

機能サブセット発見の手がかりとして以下のような例が考えられる。これらが最初に考慮すべき大粒度の論理的なアーキテクチャ要素の候補となる。

- 機能や構造が密接に凝集している
- 単一アーキテクチャコンポーネントによって実現される
- 既存コンポーネント（HW/SW）の再利用によって実現される
- システム内で再利用される場合がある
- 特定の設計制約に対応している

これらを用いて、ブラックボックスを、システムの目的を実現できるよう、論理的なアーキテクチャ要素へ、あるいは、物理的なアーキテクチャ要素へ分解することになる。

ところで、通常、システムの目的を実現可能な解の候補として、複数のアーキテクチャ要素を想定可能であることが多い。これらの要素は、ハードウェアやソフトウェア要素であったり、あるいはその両方を含む大粒度のものであったりする。

そこで、その候補の中から合理的に最適な要素の選択を行うために、選択の基準に従った選択を行うが、通常は、相対的な重要性に基づいて重み付けされた選択の基準（有効性の指標（Measures of Effectiveness: MoEs）など）を定義し、それに基づいて選択を行うことが多い。

これらに基づき、主要な要求の集大成として、システムアーキテクチャ設計コンセプトが生成される。

3.2.2.2 アーキテクチャ設計の活動

この活動は、ブラックボックスとして定義されていたシステムを、必要な関心事が割り当てられたサブシステムに分解し、最終的にはシステムアーキテクチャ構成を得るためのものである。

この活動は、アーキテクチャ分析から得られたシステムアーキテクチャ設計コンセプトに基づいて、システムを論理サブシステム、物理サブシステムに分解することから開始する。

この活動は、以下のようなワークフローで実行されることが多い。

システムレベルの操作をサブシステムに割り当てる

このサブシステムは、前述のアーキテクチャ分析から得られたもの、あるいは既存アーキテクチャの要素である。

品質要求や安全性要求など、要求分析活動で捕捉されたシステムレベルで解くべき課題について、様々な割り当てを行うことになる。

割り当て活動は、反復プロセスであり、通常はドメインエキスパートと協力して行う。

一つの操作を単一に割り当てることができない場合は、操作をサブ操作に分解し、複数のサブシステムに割り当てる場合もある。

フォールトトレランス要求を満たすためにアーキテクチャの冗長性が必要な場合などは、同じ操作を異なったサブシステムに割り当てる場合もある。

複数のサブシステムにまたがった割り当てには、インターフェースが存在することになる。このタイミングで個々の通信チャネルへの負荷見積もりを行うことができる。

品質要求を関連パーツに割り当て、トレーサビリティを確立する

システムの機能要求を満たすようなシステム操作がシステムサブシステムに割り当てられた後、品質要求（安全性のための設計制約や性能要求など）が関連するアーキテクチャ要素を割り当てる、これによって機能要求と品質要求のトレーサビリティが確立される。

機能サブセットをサブシステムに割り当てる

前述の、従来からシステム機能を明確にするために、機能フロー図や、状態モード分析などが使用され、機能サブセットへの分解が行われてきた。

ここで、システム機能を実現できるように機能サブセット（機能フロー図における機能ブロックに相当）をサブシステムに割り当てる。

サブシステム間インターフェースを定義する

機能サブセットのフローをもとに、サブシステム間のトポロジーとインターフェースを明確にする。

インターフェースは、ソフトウェア的なサービスインターフェースだけでなく、物理的なものも定義する。

その結果、システム内部のサブシステム構造に関するアーキテクチャが構築されることになる。

各サブシステムについて分析する

サブシステム間インターフェースを定義した結果として生じる各サブシステムの状態遷移の定義を行う。

システムアーキテクチャを検証する

システムアーキテクチャの正確性や完全性の検証は、まず、機能要求の側面の検証を行い、その後に品質要求の検証を行う。

また、安全性要求の観点から、故障モードとその影響の解析（Failure Modes Effects Analysis: FMEA）やミッション・クリティカリティ分析などを行う。

成果物を作成する

システムズエンジニアリングにおいて利害関係者の合意を得るための文書の作成は必須である。記述される範囲と内容はプロジェクトの特性によって異なるが、通常、システムアーキテクチャ仕様書や補足文書としての要求

文書（HWやSW要求仕様書など）が作成される。

特筆すべき要求文書としてICD：インターフェース管理分書（Interface Control Document）がある。

ICDには要素間の物理的、機能的なインターフェースのみでなく、それぞれの要素を担当する組織間の作業や責任所掌のインターフェースなども記述されることもある。通常ICDは単独の文書ではなく、外観図、インターフェース項目表、電気的情報、CADモデルなど多くの図や表を含む複数の文書から構成されるパッケージ構成となっている。

4 モデルベースシステムズエンジニアリング（MBSE）によるシステム設計活動について

MBSEは自然言語記述だけでなく、便利さのためにモデル記述言語による記述を併用するシステムズエンジニアリングである。本節ではモデル記述言語SysMLを用いたMBSEによるシステム設計活動について簡単に述べる。

4.1 なぜモデル化言語も併用するのか？

これまで、基本的なシステムズエンジニアリングにおけるシステム設計活動について解説してきたが、これらの活動の主目的はMBSEにおいても同様である。

しかし、これまでのシステムズエンジニアリングには２つの「不便」があった。

一つは自然言語そのものの記述力を原因とする不便である。

例えば、自然言語でシステムズエンジニアリング領域で多用される階層構造や並行動作を同時に記述することは困難である。また、入り組んだ情報の俯瞰や、複数の原因と結果を持つ因果関係の説明をすることも難しい。SysMLはシステムズエンジニアリングのために策定された準形式的図式言語であり、これらの不便を解消しやすい。

もう一つの不便は、記述内容の一貫性の維持という問題である。

自然言語で記述された文書間をまたがるような情報の完全性や一貫性の維持は困難であるし、文書に記載された情報が変更されたときのシステムのライフサイクルへの影響の見通しも困難である。

これらについてもモデリングツールを併用しながらSysMLを使用することでこれらの不便を解消できる。

4.2 SysMLのダイアグラム

MBSEにおいては要求、構造、振る舞い、パラメトリック制約という４つの観点を4 Pillarsと呼んでいる。これを表現するためのSysML図はそれぞれ要求モデル、構造モデル、振る舞い/機能モデル、制約モデルと呼ばれる。

以下、それぞれにごく簡単に紹介する。

要求モデル：要求図

要求図は要求の分類や、要求同士の包含、派生、因果などの関係をビジュアル化できる。

要求図の主なモデル要素は要求を表現した要求ブロック、依存関係などである。

機能安全規格準拠のためのトレーサビリティ表現に最近よく使用されている。

構造モデル：ブロック定義図
（BDD: Block Definition Diagram）

ブロック定義図は、システムの基本構造要素（ブロック）、及びそれらの全体-部分関係や依存関係などを示す。

主なモデル要素は、構造単位であるシステムブロック、コンポジット関連など。

システムブロックは、ハードウェア、ソフトウェアなどに限らず、関心事一般に使用できる。

全体-部分の階層構造の表現にとくに多く使用される。

構造モデル：内部ブロック図
（IBD: Internal Block Diagram）

内部ブロック図は、BDDで定義された全体を表すブロックを構成する部分（パート）をどう結合して全体システムをどう実現するか、すなわちサブシステム同士がシステム内部でどう内部接続されているかを示す。

主なモデル要素は、パート、ポート、コネクタなど。

パートは、コネクタを介して他のポートと接続される。

ポートは、パート間の相互作用ポイントである。サービス呼び出しや、入出力するデータや物理的実体（液体、固体、気体、エネルギーなど）など、相互作用の種類がポートごとに決まっている。

このような明確に定義されたインターフェースを使用するポートをシステム要素として指定することで、再利用可能モジュールとしてのブロックの設計が容易になる。

振る舞い/機能モデル：シーケンス図

シーケンス図は、ユースケース図やアクティビティ図で示された要求に対し、アクターやブロックがどのような順序で相互作用しながら振る舞うかをビジュアル化する。

主なモデル要素はライフライン、メッセージ。

ユースケースのシナリオを詳細化するためによく使用されている。

振る舞い/機能モデル：アクティビティ図

アクティビティ図は、機能の実行フローをコネクタによって結合された複数のアクションに分解することで、ユースケースのワークフロー、ビジネスプロセス、アル

ゴリズムなどを記述する。

主なモデル要素はアクション、コントロールフロー、決定ノード、スイムレーンなど。

アクティビティ図は、単純なアクティビティも、条件付き分岐や並行性を使用した、複雑なアクティビティも記述できる。

アクションはグループ化され、個々のサブシステムに割り当てることができる。この場合、個々の責任範囲を示すスイムレーンを用い、アクティビティ図を分割する。

振る舞い/機能モデル：ステートマシン図

ステートマシン図は、ブロックの状態ベースの振る舞いを記述する。

主なモデル要素は状態、遷移、イベント、アクションなど。

振る舞いの階層構造や並行性を記述可能である。

アクティビティ図（機能フロー）とシーケンス図（環境との相互作用）の両方の情報を集約し、それをブロックにおけるイベント駆動の振る舞いとして定義できる。

ステートマシン図は、遷移線によって結合された状態によって構成されている。イベントは、ある状態から別の状態への遷移のトリガーとなる。アクションは、状態遷移時や状態への入場時、退場時に実行される。

振る舞い/機能モデル：ユースケース図

ユースケース図は、システムのユーザとシステム自体との間の相互作用を記述することで、システムの機能要求を捕捉する。

主なモデル要素は、ユースケース、アクター、システム境界など。

対象システムのスコーピングに使用される。

ユースケース図においてシステムはブラックボックスとして扱う。

制約モデル：パラメトリック図

パラメトリック図は、システムプロパティ間のパラメトリック関係を視覚化するための、特殊なタイプの内部ブロック図である。

主なモデル要素は制約ブロックやコネクタなどである。

技術品質測定やトレードオフスタディーのためによく使用される。

4.3 SysMLを用いたシステム設計活動

ここでは、MBSEで用いられるSysMLの特徴的な使用方法について簡単に述べる。

4.3.1 要求分析時

前述のように、要求分析の作業は２つの側面を持つ。MBSEでは、それぞれの側面を表現するため、それに適した図を用いる。

システムをブラックボックス的に定義する

MBSEでは対象システムとシステム外部との間のシステム境界を明確にするためにユースケース図を使用し、対象システムのスコーピングを行う。

システムをホワイトボックス的に定義する

システム機能を明確にするため、シーケンス図、アクティビティ図、ステートマシン図などの振る舞い図を使用し、システムレベルユースケースの振る舞いを定義する。これは機能分析にも使用可能である。

4.3.2 システムアーキテクチャの構築時

システムアーキテクチャの構築活動においては、モデルを用いた様々な割り当てを行う。グラフィカルなモデル要素同士を様々な方法でリンクすることによって表現する。最も視覚的な方法は《allocate》ステレオタイプのついた破線矢印で２つのモデル要素を結ぶことである。以下にMBSEで使用することの多い割り当ての例を挙げる。

要求の割り当てについては、要求ブロックを割り当ての単位とする。要求図は要求ブロックと設計要素である論理ブロックや物理ブロックとの実現関係を示すことが可能である。

また、要求が満たされたことを検証するためのテストケースとの検証関係なども表現可能である。

振る舞いと構造の割り当てについては、振る舞いモデルの要素（アクティビティ図のモデル要素であるアクティビティやアクション）を構造モデルのサブシステム要素（ブロック定義図のブロック）に割り当てることで機能割り当てを表現する。割り当てられたモデル要素ブロックやパートは、振る舞いを実行する責務があるという意味となる。

また構造要素間のフローはアクティビティ図でアクティビティ間を接続するオブジェクトフローや内部ブロック図におけるアイテムフローで表現される。

構造同士の割り当ては、論理-物理割り当てと言われる、論理アーキテクチャの論理ブロック階層と物理アーキテクチャの物理ブロック階層を対応付けるときによく使用される。同様にHW-SW割り当てにも使用される。（ソフトウェアのより詳細な割り当てにはUMLの配置図を使うことのほうが多い）

プロパティの割り当ては、とくに性能や物理プロパティのような値プロパティ（システムの重量や部品コストなど）をブロックのプロパティに割り当てる。

パラメトリック図を使い詳細化することも多い。

システムレベルの操作をアーキテクチャ構造の要素へ割り当てたい場合、もし、一つの操作を単一ブロックに割り当てることができない場合は、操作を分解する必要がある。

その際は、ユースケースレベルのホワイトボックスアクティビティ図を作成し、アクションを、それぞれ分解階層のブロックを表すスイムレーン内に分配する。

ユースケースレベルのホワイトボックスアクティビティ図は、開発イテレーションによるアーキテクチャ分解を反映し、再帰的に、より詳細になる場合がある。

そして、アクティビティ図におけるスイムレーンをまたがったリンクはインターフェースに相当するため、ホワイトボックスアクティビティ図を使用して、個々の通信チャネルへの負荷について最初の見積もりを行うこともできる。

一方、システムレベルの一つの操作を異なった複数のブロックに割り当てなくてはならない場合もある（フォールトトレランス要求を満たすためのアーキテクチャ冗長性など）。

4.3.3　アーキテクチャ分析の活動時

システムアーキテクチャ設計のコンセプトを構築するためには主要なシステム機能の明確化を行わねばならない。

MBSEにおいて主要なシステム機能は、ユースケース図で捕捉し、機能サブセットへの分解は前述のようにアクティビティ図を用い、操作をBDDで分解したアーキテクチャ構造の要素であるシステムブロックへ割り当てる。この割り当ては論理ブロックに対しても、物理ブロックに対しても行うことができる。

この作業は、非常に容易な上、理解性にも富んでいる。

また、その際、もし複数のアーキテクチャ要素が割り当て選択の候補に上がっている場合、選択の基準である有効性の指標（MoE）を定義、計算する際に、パラメトリック図を用いると便利である。

重み付け目標計算の結果、決定したソリューションをもとにしたシステムアーキテクチャ設計コンセプトには、ここまで使用してきたSysMLの各モデルが含まれる。

4.3.4　アーキテクチャ設計の活動時

この活動ではアーキテクチャ分析から得られ、SysMLモデルで記述されたシステムアーキテクチャ設計コンセプトに基づいて、システムを論理サブシステム、物理サブシステムに分解し、システムアーキテクチャ仕様書を作成する。

システムレベルの操作をサブシステムに割り当てる

ユースケースレベルのホワイトボックスアクティビティ図を作成し、BDDを用いて分解したシステムブロックごとに、アクティビティ図のスイムレーンを作成、サブアクションを各スイムレーンに割り当てることで実行の責務を割り当てる。

要求分析活動で捕捉された、品質要求や安全性要求などのシステムレベルで解くべき課題について、様々な割り当てを行うことになる。

割り当て活動は、反復プロセスであり、通常はドメインエキスパートと協力して行う。

一つの操作を単一ブロックに割り当てることができない場合は、操作をサブ操作に分解し、複数のサブシステムに割り当てる場合もある。

フォールトトレランス要求を満たすためにアーキテクチャの冗長性が必要な場合などは、同じ操作を異なったサブシステムに割り当てる場合もある。

複数のサブシステムにまたがった割り当てには、インターフェースが存在することになる。このタイミングで個々の通信チャネルへの負荷見積もりを行うこともできる。

品質要求を関連パーツに割り当て、トレーサビリティを確立する

システムの機能要求を満たすようなアクションがシステムサブシステムブロックに割り当てられた後、要求図上で品質要求を表す要求ブロックとサブシステムブロックを《satisfy》依存関係で割り当てる、これによって機能要求と品質要求のトレーサビリティが確立される。

そして、アクティビティ図におけるスイムレーンをまたがったリンクはインターフェースに相当するため、パラメトリック図を用い、サブシステム間通信の負荷についての見積もりを行う。一度、性能評価のモデルを構築しておけば、将来、振る舞いや構造、あるいは操作などの変更依頼への対応が容易になる。

機能サブセットをサブシステムに割り当てる

従来の機能フロー図における機能ブロックはアクティビティ図におけるアクション相当、状態モード分析は、階層化されたステートマシン図相当とみなせる。

サブシステム間インターフェースを定義する

IBDを用い、サブシステムパート間のトポロジーとインターフェースを明確にする。

ポートを使用することで、サービスインターフェースだけでなく、物理的な入出力も定義可能となる。

その結果、システム内部のサブシステム構造に関するアーキテクチャが構築される。

各サブシステムについての分析

ステートマシン図を用い、サブシステム間インターフェースから通知されるイベントをトリガーとした各サブシステムブロックの状態遷移の定義を行う。

システムアーキテクチャの検証

様々な抽象度におけるブロックのステートマシンを構築することで、検証項目のヌケモレを回避できる可能性が向上する。

成果物の作成

MBSEにおいても従来のシステムズエンジニアリング同様、通常、システムアーキテクチャ仕様書や補足文書としての要求文書（HWやSW要求仕様書など）が作成される。

ただし、これら文書中には理解性、明瞭性などの観点からSysML図が含まれている。

通常、SysMLモデリングツールを使用してSysML図を記述した場合は、後工程で特定要求のトレーサビリティや詳細の確認、トレードオフスタディの再検証などの際に構築したモデルが必要になることも多いため、ツールで記述した全モデルレポジトリのデータを補助的成果物として扱うことも多い。

5 おわりに

本稿では、従来のシステムズエンジニアリングにおける上流工程である要求分析活動とシステムアーキテクチャの構築活動について紹介した。また、近年話題に上ることの多いSysMLを利用したMBSEの上流工程で、実際にモデルをどのように使用しているかについても簡単に述べた。

今回紹介したのは、要求を起点とした古典的システム開発、言い換えればQCDを守って成功裏に開発するためにアーキテクチャを構築するためのシステムズエンジニアリングのごく一部にすぎない。

基本的にMBSEを含むシステムズエンジニアリングは、主にそのシステムの目的を達成するためのシステム機能を入力とし、様々な観点を持つアーキテクチャを構築することが「上流工程」となる。

また、MBSEにおいては観点ごとのビルディングブロック（要求ブロック、システムブロック、アクティビティなど）を用いた図式表現を行ったり、再帰的なシステム-サブシステム構造を多用したりすることで、効果的なアーキテクチャ表現が可能となる。

ところで、最近の巨大で複雑なシステム開発プログラム/プロジェクトにおいて、そもそも多岐にわたる膨大な要求にどう対応すべきか不明となり、十分な成功を収められないことが多くなってきた。

そして、巨大で複雑なシステムは個々人の要求を満たすという目的より、ビジネス戦略を成功裏に遂行するという目的が重要と考えられる。

この目的達成のためには、ビジネスプロジェクト開始時からビジネス戦略だけでなく、ビジネス戦略遂行の仕掛けとしてのシステムアーキテクチャを同時にマネジメントしなければならない。これをアーキテクチャファーストの考え方という。

そして、アーキテクチャファーストに基づき、ビジネス、運用、情報、アプリケーション、制御などといった異なる機能ドメインからの視点を持つシステムアーキテクチャを構築するためのアーキテクチャがアーキテクチャフレームワークである。

現在IoTに対する関心がますます高まってきている。ここで最も懸念されているのは、異なるハードウェアやソフトウェアであっても、IoTサービスやデバイス間の連携が可能な相互運用性と言われている。そしてアーキテクチャ不在で、プロプライエタリなセンサデータ形式とサービスインターフェースだけではIoT業界は協調不能なサイロを作り続けるだけなのではないか、という声も聞こえてくる。

一方、Industrie 4.0においても、2030年の実現に向け、サプライチェーン、生産の効率化にデジタルモデルを活用することの重要性を強調しているが、このリファレンスアーキテクチャもアーキテクチャフレームワークそのものである。

これらの新潮流の考え方は、明らかにアーキテクチャファーストであり、強くアーキテクチャフレームワークを意識している。これら領域での事業を計画されている諸氏は、ぜひMBSEを含むシステムズエンジニアリングを再確認していただければと思う。