オペレーショナル・モデリングにおける非機能要件の効果的検証方法

山本 久好* 榊原 彰**

An Effective Inspection Method of the Non-Functional Requirements in Operational Modeling

Hisayoshi Yamamoto* Akira Sakakibara**

IBM Global Services Methodのアーキテクチャ・ドメインに従いオペレーショナル・モデルを作成する際には, 実に様々な考慮が必要である. 考慮する項目はダイアグラムやその他の記述に反映されるが, モデル上での非機能要件の検証に関しては, 要件記述を確かめる以外に方法らしき方法が考案されていない. 本論文ではこうした点を補完すべく, 非機能要件を関心事に分離して確かめる検証方法の提案を行う. また, 併せてこれらの方法を実現するモデリング・ツールの実装に関する要件を提示するものである.

Based on the architecture domains of the IBM Global Services Method, there are many considerations in creating an operational model. The points to be considered are usually expressed in terms of diagrams and other descriptions, but as far as verification of non-functional requirements is concerned, there are hardly any proven means, outside of checking requirements descriptions. In order to complement such means, this paper gives a proposal on a method of verification by dividing non-functional requirements into categories. At the same time, it presents the requirements for implementing a modeling tool which supports the proposed method.

Key Words & Phrases：オペレーショナル・モデル, 非機能要件, 関心事の分離, モデリング・ツール
Operational Model, Non-Functional Requirements, Separation of Concerns, Modeling Tool

1. はじめに

IBM開発手法の体系であるIBM Global Services Method（以下, GSMethod）におけるアーキテクチャ・ドメインでは, システム・アーキテクチャが, 機能要件を反映させるファンクショナル・アスペクトと, 非機能要件を反映させるオペレーショナル・アスペクトの協調で構成されると捉えている. システムの非機能要件・運用基盤的要件を反映させるオペレーショナル・モデルは, 稼働環境, すなわち静的側面を表現するノード・リレーションシップ・ダイアグラムと, 機能要件の具象であるコンポーネントをノード上に配置するための,「配置ユニット」の協調動作を表現するトランザクション・ウォークスルー・ダイアグラム, およびそれらを補完する記述から成る. また, これらのモデルはConceptual Operational Model（COM）, Specification Operational Model（SOM）, Physical Operational Model（POM）の3つのモデルにわたって推敲される.

GSMethodのアーキテクチャに関する考え方を定義している, いわばアーキテクチャのメタモデルであるArchitecture Description Standardでは, オペレーショナル・モデルを作成するためのダイアグラム表記法を規定しており, 3種のモデルは, 明確な意味定義が行われている図形で構成されるダイアグラムにより, 設計内容の標準化が図られている.

2. 関心事の分離

ところが, オペレーショナル・モデル, 特にノード・リレーションシップ・ダイアグラムは, システムの稼働環境を精緻に表そうとするため, 規模が膨らむと複雑に

提出日：2003年8月29日
* hisayosi@jp.ibm.com **aquila@jp.ibm.com

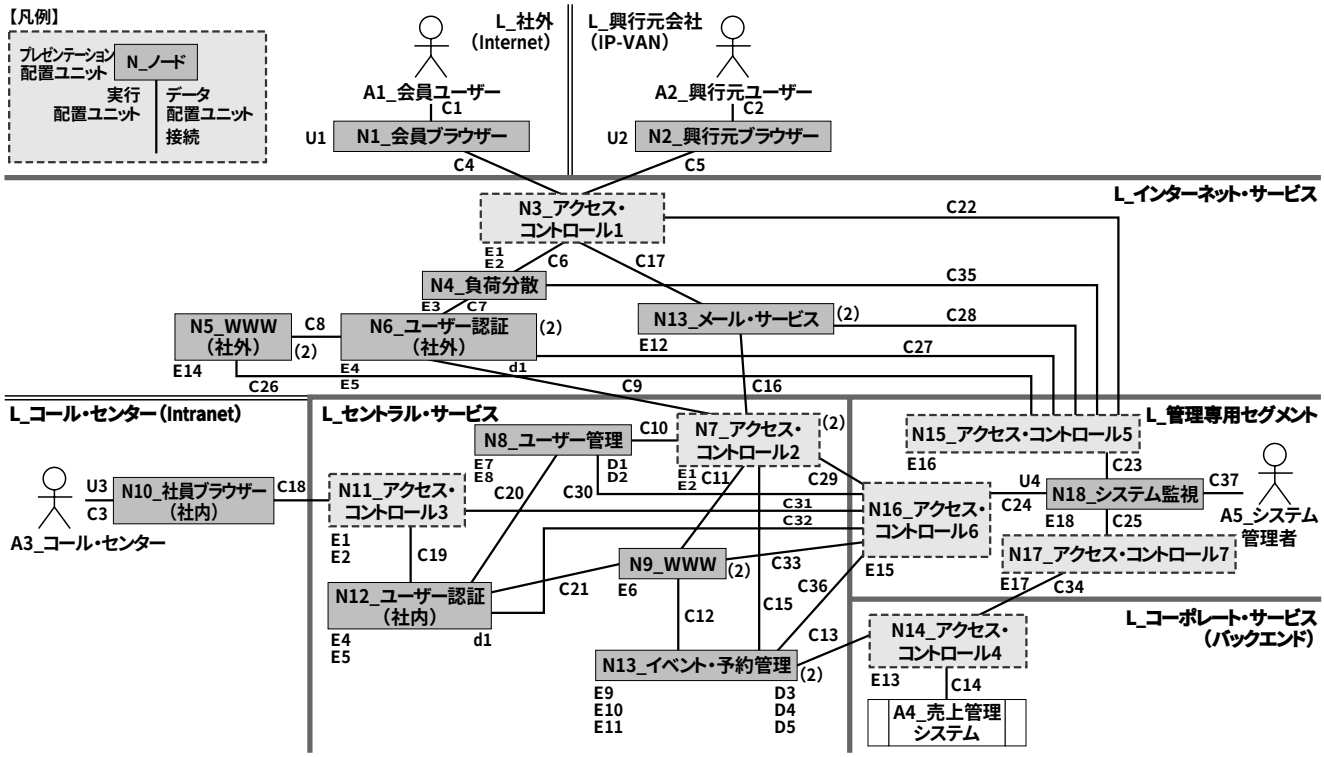
なりがちである. ノードの数が増加し, ノード間を接続するコネクションがそれ以上に増加するためである.

我々はこの状況を打破するため, システムを横断する関心事によって, ノード・リレーションシップ・ダイアグラムを分離するという方策を考えた. この発想は, 昨今注目を浴びているアスペクト指向(Aspect-Oriented)のアナロジーをオペレーショナル・モデリングに持ち込むものに他ならない.

現在脚光を浴びているのは, 主にAspectJなどに代表されるアスペクト指向プログラミング技術[1]のみである. しかしアスペクト指向の考え方そのものはプログラミングにとどまらずソフトウェア開発全般, システムズ・エンジニアリング全般に広く受け入れられるものであると考えられる[2]. 我々が導入したアスペクト指向の概念を以下にあげる.

(1) 横断的な関心事(crosscutting concern):アスペクト指向の根幹をなす基本概念である. 複数の対象(プログラミングであればモジュール)に横断的に存在する関心事を言い, これらに対する変更の柔軟性を確保することがアスペクト指向の大命題となっている. 現実的には横断的な関心事の多くは, 運用的な関心事や容量的・速度的な関心事など, 非機能要件を取り込むものである.

(2) アスペクト(aspect): 横断的な関心事を切り出してモジュール化するためのメカニズムを言い, 今回のアプローチのようにプログラミングではない文脈においては, 完全にこの意図を反映する



プレゼンテーション配置ユニット
U1:イベント検索, チケット予約(社外)
U2:イベント登録(社外)
U3:予約販売登録, 予約修正/削除(社内)

実行配置ユニット
E1:ルーティング
E2:パケット・フィルタリング
E3:負荷分散
E4:リバース・プロキシー
E5:ユーザー認可
E6:HTTP(ページ表示)
E7:ユーザー管理, 認証機能含
E8:ACLレプリカ(d1)作成
E9:イベント・予約管理
E10:マスタ管理
E11:売上管理連携
E12:メール送信
E13:ルーティング(バッチ用)
E14: HTTP(社外トップ・ページ, 静的コンテンツ)
E15:ルーティング(セントラル・サービス監視用)
E16:ルーティング(インターネット・サービス監視用)
E17:ルーティング(コーポレート・サービス監視用)
E18:システム監視

データ配置ユニット
D1:ACL(ユーザー認可), d1:一部コピー
D2:LDAP(ユーザー認証)
D3:マスタ・データ
D4:イベント・予約データ
D5:アプリケーション・ログ

接続
C1:会員がブラウザーにアクセス
C2:興行元ユーザーがブラウザーにアクセス
C3:コール・センターがブラウザーにアクセス
C4:インターネット・サービスにアクセス:社外から
C6:インターネット・サービスにアクセス:興行元から
C6:負荷分散ノードにアクセス
C7:ユーザー認証:社外から
C8:トップ・ページにアクセス:社外から
C9:セントラル・サービスにアクセス:社外から
C10:ユーザー管理にアクセス:社外から
C11:Webページにアクセス:社外から
C12:イベント・予約管理にアクセス
C13:コーポレート・サービスにアクセス
C14:売上管理システムへアクセス
C15:インターネット・サービスにアクセス(メール送信):社内から
C16:メール・サーバーにアクセス:社内から
C17:メールの送信:社内から
C18:セントラル・サービスにアクセス:コール・センターから
C19:ユーザー認証:社内から
C20:ユーザー管理にアクセス:社内から
C21:Webページにアクセス:社内から
C22:インターネット・サービス監視
C23:インターネット・サービスにアクセス:管理専用セグメントから
C24:コーポレート・サービスへアクセス:管理専用セグメントから
C25:セントラル・サービスへアクセス:管理専用セグメントから
C26:WWW(社外)監視
C27:ユーザー認証(社外)監視
C28:メール・サービス監視
C29:セントラル・サービス監視(社外アクセス)
C30:ユーザー管理監視
C31:セントラル・サービス監視(社内アクセス)
C32:ユーザー認証(社内)監視
C33:WWW監視
C34:コーポレート・サービス監視
C35:負荷分散監視
C36:イベント・予約管理監視
C37:システム管理者がシステム監視にアクセス

図1. 全ての関心事を表現したモデル

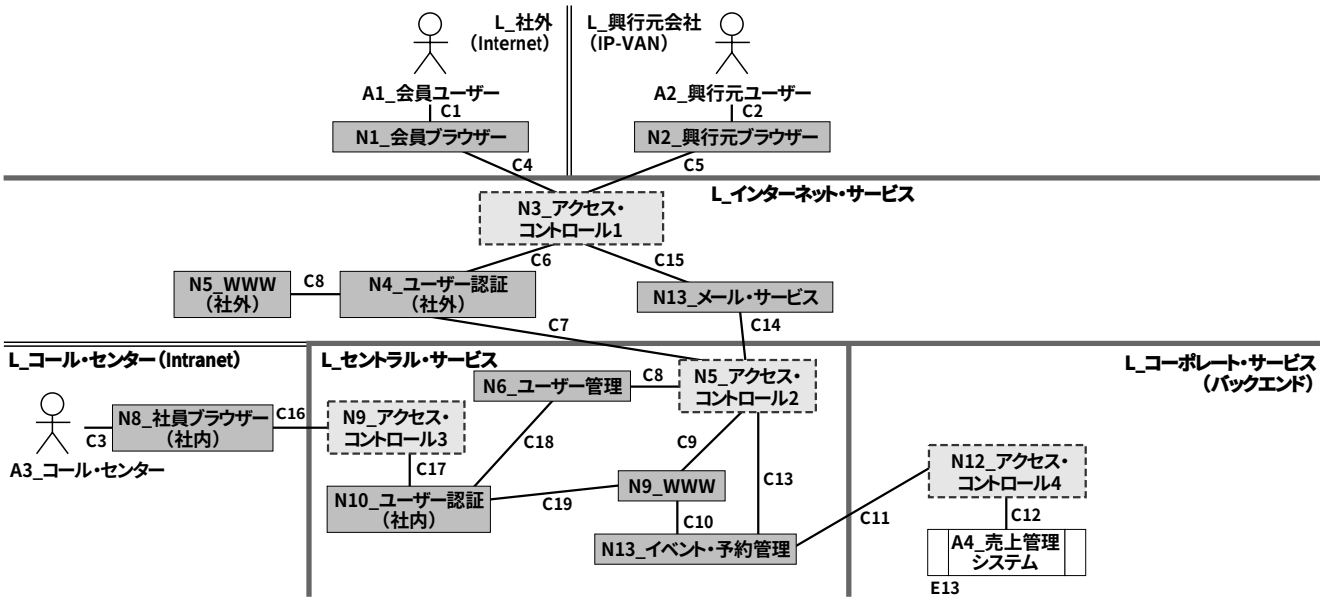


図2. イニシャル・ビュー

ことは難しい. そこで, 今回は「関心事を切り出す」事にこの概念を利用し, 「ビュー」と呼ぶことにした.

(3) 合成(composition)またはウィーブ(weave): 横断的な関心事に対応した複数のアスペクトによってモジュール化された対象をまとめ上げることを言う. この概念によって, 対象内に渾然一体となっていた関心事を切り出すだけではなく, また整然と組み上げることが可能になる. ※1

これらの概念を使用して我々がとったアプローチは, オペレーショナル・モデルにおいて非機能要件が適切に反映されているかの検証である. オペレーショナル・モデルにおいて, 非機能要件は, 前述のダイアグラム類に表記されることが多くはなく, 基本的にダイアグラムを補完する記述書に記載されることが大半である. だが, 非機能要件を反映するためのノード追加, コネクション変更などは確実に存在すると思われ, ダイアグラム上でこれらの要件の取り込みが行われていることの確認が必要と考えるに至った. そこで, ノード・リレーションシップ・ダイアグラムに対して, 関心事の分離という概念を適用することにしたというわけである.

3. オペレーショナル・モデルにおける関心事

オペレーショナル・モデルで全ての関心事(全ての非機能要件のモデル化)を1つのモデル上で表現しようとすると, **図1**のように非常に煩雑になってしまい, 判読性を著しく欠いてしまう. そこで, オペレーショナル・モデルにおける関心事を展開し, 非機能要件の特性毎に「関心事を切り出す」ビューとして整理したのが**表1**である. まず(**図2**参照)で, COMにゾーニングおよびそれにまつわるアクセス・コントロールを配置する. このイニシャル・ビューで表現されたモデルが, 機能要件を実現する運用基盤を表現することになる. このビューを起点に, 各関心事のビューではそれぞれのビューでの関心事を検証する.

例えば, 運用ビューの場合, **図3**のように, 運用管理という関心事のみに関係するコネクションや配置ユニットだけが記載されることで, 図1の全ての関心事を表現したモデルと比較し, 非常にシンプルになっている. これにより, モデルにおいて運用管理という非機能要件の特性に関し, 適切に反映されているかという検証がやりやすくなるのである.

注意しなければならないのは, オペレーショナル・モデルのレベルにより, 関心事のビューとして使えないものもあることだ. 例えば, 性能ビューで関心事としている「パフォーマンス」や「キャパシティー」は, 要件として「オンラインのレスポンス・タイムは6秒以内」, 「10万行の注文レコードのバッチ処理が10分以内で完了」といった目標値を定義している可能性がある. ところがプラットフォーム, ハードウェア, ソフトウェアが明らかになっていないSOMでは, 期待される目標を満たすアーキテクチャか否かに関する有効性の高い判断は不可能である. これを踏まえて**表2**にオペレーショナル・モデルのレベルと利用可能なビューを整理してみた.

※1 アスペクト指向プログラミング言語によっては, これらをポイントカット(pointcut)やジョインポイント(join point)という技術で支援するが, 今回はそこまでの具象技術を持ち込んではいない.

表1. 関心事を切り出すビュー

ビュー名	関心事となる非機能要件の特性	関心事とモデルでの検証内容
イニシャル・ビュー	機能要件を実現する運用基盤	(1)機能要件を実現するのに必要な運用基盤とゾーニング，およびアクセス・コントロールが配置（全ての検証ビューの基となるビュー）
運用ビュー	運用管理（サービス監視，システム管理，ソフトウェア配布）	(1)サービス監視の対象とその配置場所，アラート（警告），監視状況の伝送経路の確保 (2)バックアップ取得対象データの配置場所とバックアップ保存先と伝送経路の確保 (3)コンテンツ更新の対象，その配置場所，および伝送経路の確保 (4)構成情報ファイルの配置場所，および更新方法の検討
障害対策・信頼性ビュー	障害対策 信頼性	(1)重要業務サービスに関わるノードの二重化などの必要性（障害発生箇所，影響度，発生確率）検討 (2)Single Point of Failureとなりうるノードやコンポーネントの発見 (3)障害発生の発報伝送経路の確保 (4)障害時のリカバリーに必要な経路の確保 (5)障害回復対応時の代替フロー
スケーラビリティー・ビュー	スケーラビリティー	(1)業務処理量の増加に対し，アーキテクチャを変更せず，ノード追加，ノード交換などの対応可能性の確認
性能ビュー	パフォーマンス キャパシティー	(1)トラフィックが集中するノードの存在 (2)業務処理量の拡大に伴う処理能力の強化が必要なノードの存在
セキュリティー・ビュー	セキュリティー	(1)守るべき資源は何か，配置はどうか ・データ（エンティティー，ログ，アプリケーション，コンテンツ） ・メッセージ・フロー（コネクションと各ノードの協調） ・構成要素（ノード） (2)脅威の発生箇所（構成要素） (3)脅威への対抗策の配置箇所の検討 ・侵入検知と警告発報先への経路 ・暗号化/復号化の実施箇所 ・ユーザー認証の実施箇所，ユーザー管理（ユーザー登録/変更/抹消）の仕組み ・セッション管理の仕組み ・認可の実施箇所，認可管理（登録/変更/抹消）の仕組み

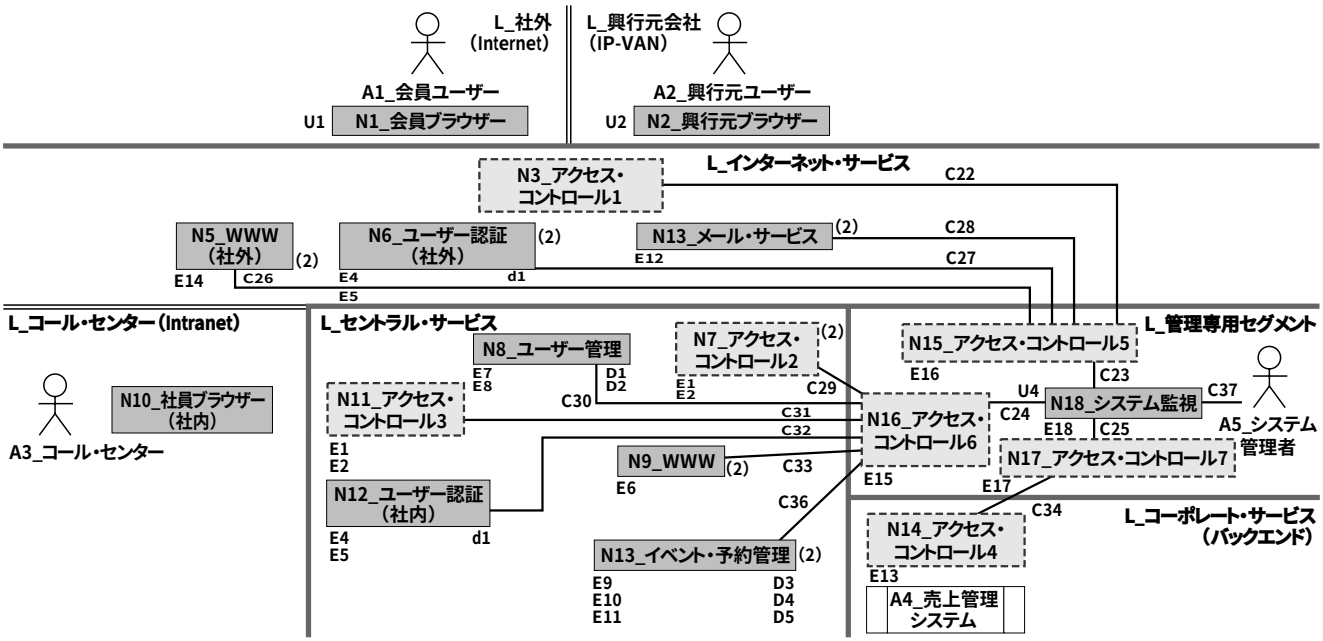


図3. 運用ビュー

表2. オペレーショナル/モデルと関心事の関係

	関心事				
オペレーショナル・モデル	運用管理	障害対策 信頼性	スケーラビリティー	パフォーマンス キャパシティー	セキュリティー
SOM	○	○			○
POM	○	○	○	○	○

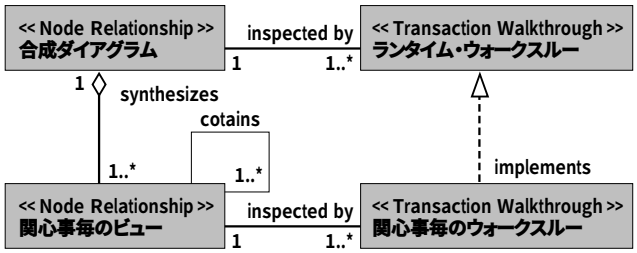


図4. 関心事の分離／合成のメタモデル

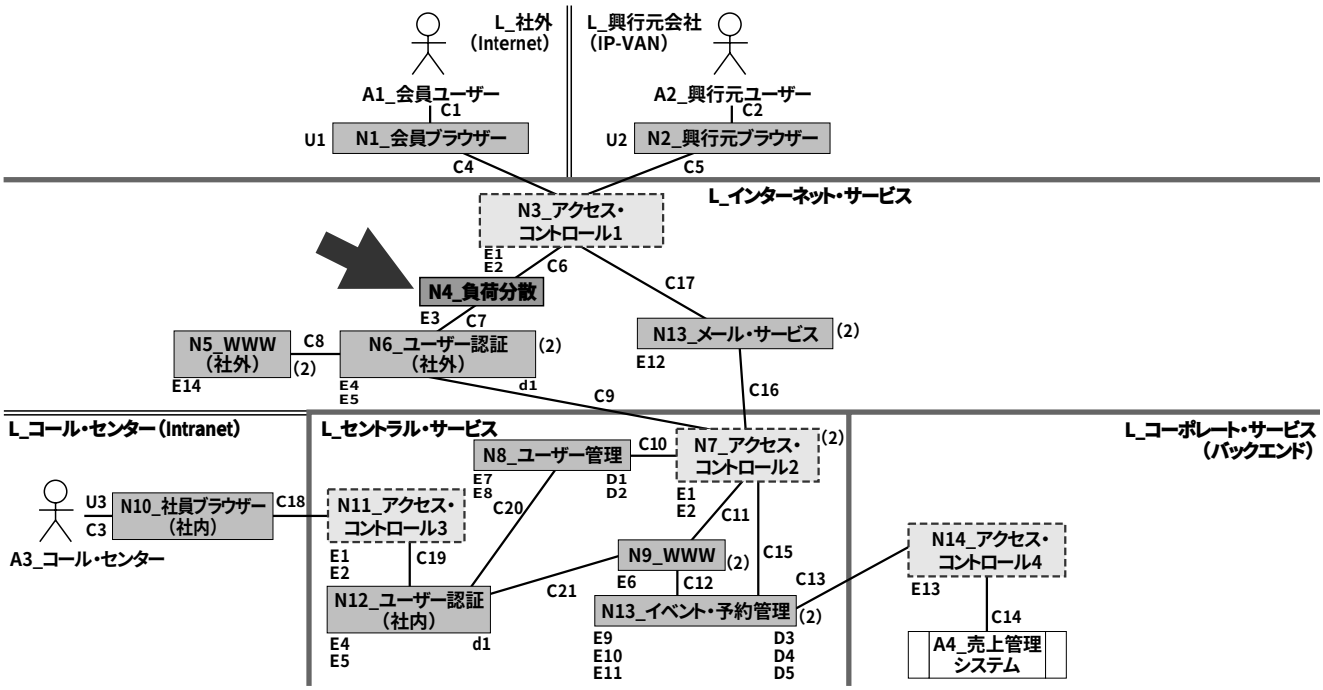


図5. 信頼性ビュー

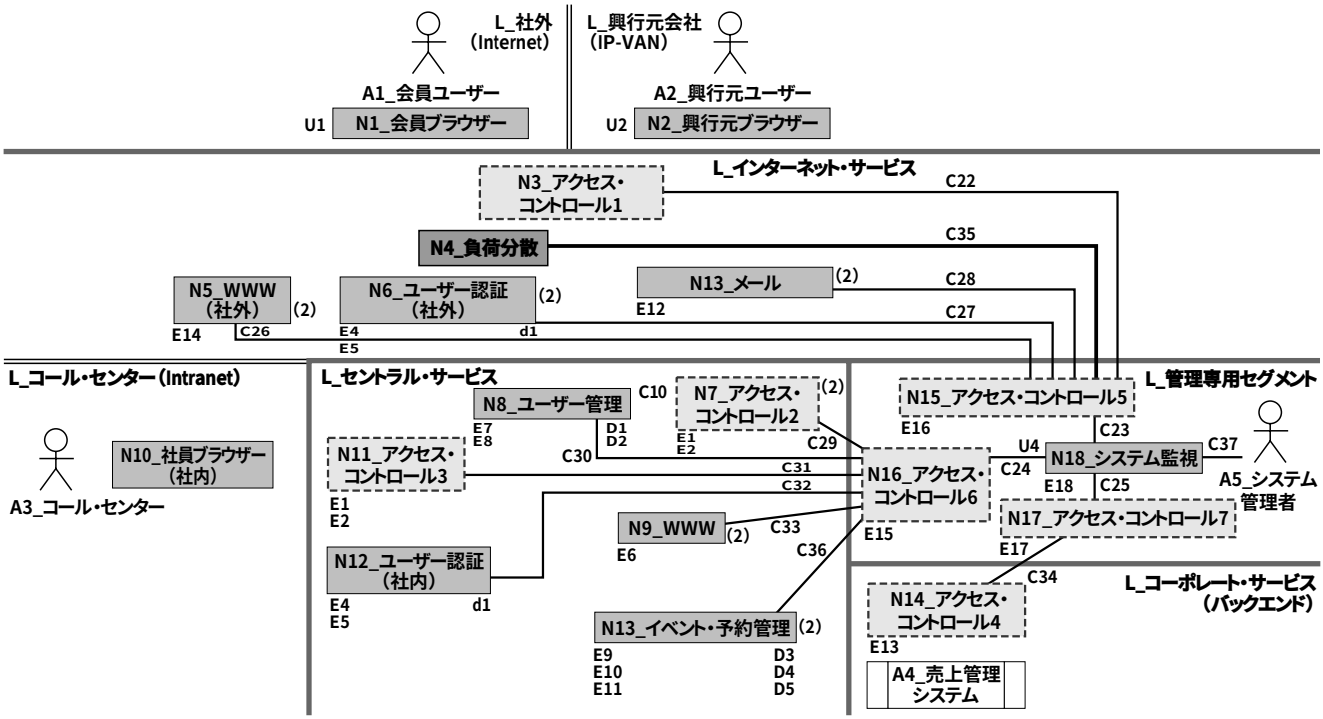


図6. 運用ビュー（負荷分散ノード追加）

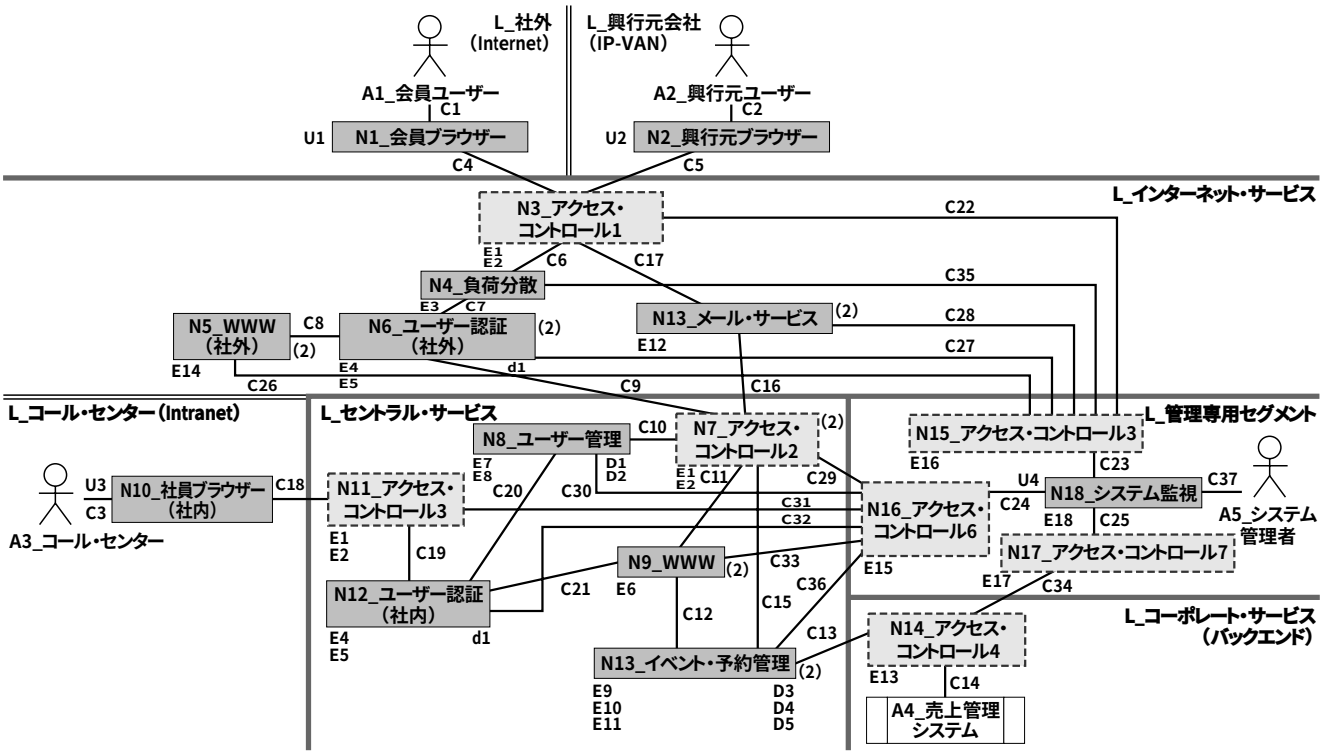


図7. 非機能要件を反映したモデル

このようにそれぞれの関心事に分けたビューを利用し, 前述のトランザクション・ウォークスルーを実施する.

4. 関心事の合成

一旦分離して検証を行ったそれぞれの関心事は, また合成して全体の整合性を確認する. これをメタモデルとして簡単に表したのが**図4**である.

各関心事のビューは合成を行うため合成ダイアグラムとは集約関係にあり, 合成ダイアグラムで指定されるウォークスルー・ダイアグラムは, 実際には各関心事のウォークスルーとして実現される. また, 各関心事は, 切り出された関心事を内包する関係にあるため, 再帰的な構造をとる可能性もある. なお, 各ステレオタイプ表記(<< >>で囲んだ名称)はオペレーショナル・モデルで用いるダイアグラムの種別を表している.

例えば, 障害対策・信頼性ビューで, 信頼性の要件からノード『N6_ユーザー認証(社外)』の負荷分散ノードを利用した信頼性向上策の適用を行う必要が出てきた場合, **図5**のように『N4_負荷分散』ノードが追加されることになる.

また, 図3の運用ビューは, 追加された負荷分散ノードに対するサービス監視, ログ収集, 構成ファイルの配布などの新たな要件が加わり, **図6**のようになる.

全てのビューを合成すると**図7**のようになり, これがシステムの全体像を表すモデルである.

5. モデリング・ツールの要件

分離したビューを合成し, 全て目視によるチェックで対応するということは非現実的である. チェックするだけではなく, 不整合が認識されればその内容を識別して修正し, 他のビューへの反映も行わなければならず, これは作業の反復を意味するからだ.

そこで我々は, モデリング・ツールADSG for e-business Modelo(以下, Modelo)[3]に, この反復作業を支援する機能要件をまとめた.

Modeloは, ツール・リポジトリであるADSG for e-business Dictioと連携し, モデル情報の取得・格納を行うことが可能となっている. 既にER図, DFD, UML(クラス図とシーケンス図のみ), オペレーショナル・モデルの各プラグインが提供済みである.

現在のバージョンでは, 1つのモデルの中で, 作成する各ダイアグラムをタブで切り替えて複数ダイアグラムを同時に開くことができる. この機能を利用して, 分離したビューを全て同じモデル内にまとめることができると考えた. これに加えて, 合成の機能を追加する必要がある. 合成に際しては, 以下の機能要件を満たさなければならない.

(a) ダイアグラム合成

オブジェクト表示のコンテナをレイヤーでコントロールすることにより, 異なる関心事のビューを1つのダイアグラム上にあるように見せる.

(b) ダイアグラム構成要素の解析

ダイアグラムに含まれる構成要素を解析し, ロケーション, ノード, コネクションの関係を合成ダイアグラムと各ビュー双方 に関して実施する.

(c) 不整合の発見

追加されたノードやコネクション, 変更されたコネクションなどを検知し, 合成ダイアグラム上に表示. 修正を促す.

(d) 各ビューに対する修正の反映

不整合部分の修正を合成ダイアグラム上で行う場合に, 修正箇所の識別した後に修正内容を各ビューに反映する. これは自動で反映されるよりも, 何らかのヒントを表示し, モデル作成者に操作させる方が使い勝手が良いであろう.

これらの機能を満たせば, オペレーショナル・モデル作成と検証の反復がツール上であまり労力を要せずに行うことが可能となる.

6. まとめ

GSMethodにおけるオペレーショナル・モデルは, 従来単純に絵として以上の意味を持たなかったシステム構成表記を, 再利用可能なレベルのモデルにまで引き上げることに成功している. 表記法を標準化されただけではなく, セマンティクスが与えられたことによってモデルとしての意味を持ち, ステークホルダ間にぶれのない共通のコンセンサスを確保することができたということである.

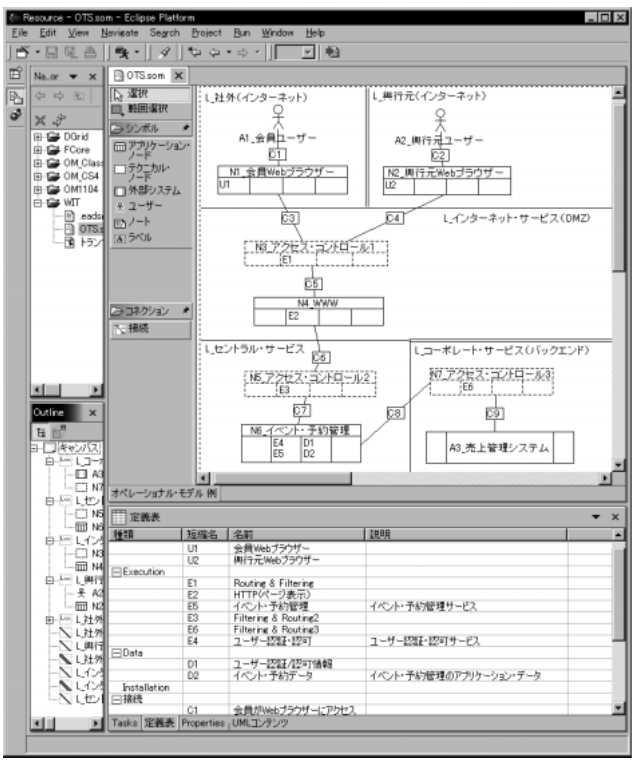


図8. Modeloで作成中のオペレーショナル・モデルのスクリーン・ショット

しかし, プログラミングにも機能横断的な関心事があるように, オペレーショナルな観点でも関心事は多岐に及ぶものである. また, ビジュアルなモデルを作成する際に, 最も難しい点は, 非機能要件を可視化することであると言える.

今回のアプローチでは, 非機能要件を中心とした関心事を分離することによって, これらの検証を容易にし, 十分に可視化できない部分でもある程度効果的(機械的)に, 品質保証活動の一端を担うことを可能とした. 慣れは必要であろうが, 表1に示した各ビューの類別は, 関心事の分離のテンプレートとしても使用可能なものである.

7. おわりに

このアプローチを試すこととなったきっかけは, ノード・リレーションシップ・ダイアグラムを描いていた際に煩雑になっていくモデルを整理したくて, アスペクト指向のコンセプトを導入してはどうかと考えたところにあった. 厳密に言うと, アスペクト指向というからには, ノード間にまたがる横断的な関心事も分離すべきであろう. この実現は今後の課題であるが, イメージとして例をあげるならば, フェイルオーバーの際に関連する各ノードで行われるリカバリー&リランの処理が1つのまとまったアスペクトとなるようなコンセプトである. しかし残念ながらまだそのレベルにまでは考察の整理が及ばなかった. 今後は関心事の分離をより細かい粒度でかつ名前の参照で行えるような明示的で柔軟な分離方法を考えていく必要がある. 関心事に名前をつけて参照でき, かつそれらの構造がわかる形態を保ちつつ合成できるような仕組みが必要になる. さらなるメタモデルの洗練が必須であろう.

また, こうした名前による参照や関心事の構造の明確化に関しては, ツール上でビジュアルにアノテーション(セマンティクス情報の付加)追記が可能となることが望ましい. 1つのヒントとしては, Chungら[4]の提唱する, 非機能要件のゴール指向分割グラフ(Softgoal Independent Graph)と, 関心事の切り出しをマッピングする術を探し出すことが考えられる.

多数のシステム基盤を設計してきた我々IBMのエンジニアは, オペレーショナル・モデルという有用な道具を手に入れた. 今後はこの効能を最大限活用すべく様々な知恵を絞っていくことが我々の大きな義務となる.

今回のアプローチが, そうした活動を促進する一環となれば良いと考えている. 日頃, 非機能要件の対応に悩んでいるオペレーショナル・モデラ諸氏にご賛同を得られれば幸いである.

謝辞

オペレーショナル・モデリングを現実的な技術として実践し, 今もなお洗練を続けているIBM UKのIan Charstersは, 著者らの種々雑多な質問に丁寧に応対してくれ, 今回の検討の機会を提供してくれた. 感謝の念に堪えない. また, 日本アイ・ビー・エムのITAコミュニティにおいてオペレーショナル・モデリング技術をリードし, 適切な助言を与えていただいている日本アイ・ビー・エム システムズ・エンジニアリングの吉田幸彦ICP Sr. Cons. ITAにもこの場を借りて深謝するものである.

参考文献

［1］ G. Kiczales, J. Lamping, A. Mendhekar, C.Maeda, C. V. Lopez, J. Loingtier, and J. Irwin, Aspect-Oriented Programming, Proceeding of ECOOP, vol. 1241 of LNCS, pp. 220-242, 1997

［2］ 岸知二, 野田夏子, アスペクト指向ソフトウェア開発, 情報処理学会2003 OOシンポジウム予稿集, pp.247-250, 近代科学社, 2002年

［3］ 榊原彰, ツールによるアセット再利用支援に関する考察, PROVISION Summer 2003 No. 38, pp.70-79, 2003年

［4］ L. Chung, et al., Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, 2000年

日本アイ・ビー・エム株式会社
技術・テクニカル・マネジメント・
Techスキル開発
ITアーキテクト

山本 久好 Hisayoshi Yamamoto

［プロフィール］
1983年に日本アイ・ビー・エムに入社・野洲工場でメインフレームの製造, 製品技術支援を経て, 1994年より, SEとして多数のプロジェクトに参画してきた.
現在は, プロジェクトでアーキテクチャ設計支援の他, 開発方法論に基づいたITシステムの非機能要件に関する研修コースの開発, 展開も手掛けている.

日本アイ・ビー・エム株式会社
技術・ソフトウェア・エンジニアリング
シニア・コンサルティングITアーキテクト

榊原 彰 Akira Sakakibara

［プロフィール］
1986年に日本アイ・ビー・エムに入社・以来, SEとして多数のプロジェクトに参画してきた・現在は, 開発方法論や開発支援ツールおよびアーキテクチャ設計技術などの開発および社内外への展開を手掛けている.
情報処理学会, プロジェクトマネジメント学会, IEEE, ACMの各正会員・日科技連SPC研究委員・WWISAメンバー.