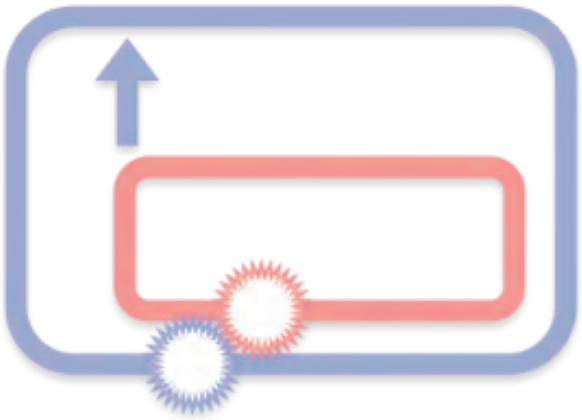
合意記述データベース

オープンシステムディペンダビリティとD-Caseを繋ぐリポジトリ



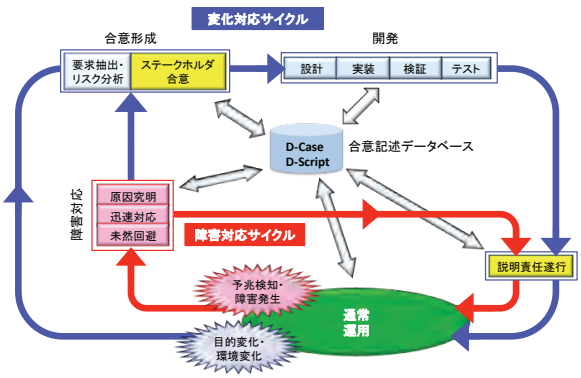
永山辰巳（株式会社Symphony）
横手靖彦（慶應義塾大学村井純研究室）

DEOSプロジェクト
2013年11月20日

1. はじめに

本ドキュメントではD-Caseを保存し、その変更を記録するDEOS合意記述データベース(以下、D-ADDと呼称)に関して、図1に示すDEOSプロセス[1]を統治する観点から述べる。D-ADDはステークホルダ合意と対象システムに存在するプログラム・コード、及び対象システムの運用状態との間の一貫性を常に保つための機構を提供すると同時に、いつでも必要な時点でステークホルダの説明責任遂行を可能とする機構を提供する。以下、まず、DEOSプロセスとD-ADDとの関係を議論する。次に、D-ADDアーキテクチャに関して述べた後、DEOSプロセスにおける説明責任遂行に関して、営業放送システムを事例に述べる。本ドキュメントで述べるD-ADDは、利用者が体験できる実装が用意されている。該実装の概略を述べた後、実ビジネスとD-ADDとの関連、及びD-ADDによるソフトウェア開発プロセスの革新可能性を考察して、本ドキュメントをまとめる。

図1:DEOSプロセス



2. DEOSプロセスとD-ADD

DEOSプロセスを構成している各(サブ)プロセスからD-ADDはアクセスされる。合意形成プロセスにおいてD-ADDは、合意された議論の過程の記録、関連議論の検索、過去の事例の検索、議論の構造化、等の合意形成の確実な実行を支援する。D-ADDが記録している合意形成の過程は、D-Caseのエビデンスノード[2]に関連付けられることで、対応するゴールが成立することの論拠を提供する。例えば、ステークホルダ要求、企画書、仕様書、契約、等のドキュメント、およびD-Caseが構造化文書リポジトリに記録される。複数のオーナーシップを有する複合システムの場合にはD-ADD同士が情報を交換する。

開発プロセスにおいてD-ADDは前プロセスでの合意項目の確実な実行を支援する。開発時に生成される、あるいは利用されるドキュメントやプログラム、コード等を前記リポジトリに記録する。障害対応プロセスにおいてD-ADDは障害原因の特定や、対応、事前の回避、等に必要な情報を提示する。

説明責任遂行プロセスにおいてD-ADDは新サービスの立ち上げ、改修、および障害対応に関してステークホルダが説明責任を遂行するに十分な情報(障害要因候補、ステークホルダ合意関連情報、エビデンス、等々)を入手できるように支援する。D-ADDは通常運用状態においても対象システムとのやり取りを通じて、対象システムのオープンシステムディペンダビリティが担保されている事を確認し、状況に応じて障害対応サイクル、あるいは変化対応サイクルが起動される。この様にD-ADDはDEOSプロセスを遂行する上で必須の構成要素である。

2.1. D-ADDアーキテクチャ

DEOSプロセスを構成する各サブプロセスから利用されるD-ADDのアーキテクチャを、(1) 各サブプロセスを支援するツール群、(2) ツール群が必要とするモデルを処理する中核部、(3) モデル情報をデータベースに記録する永続化部の３要素から説明する。

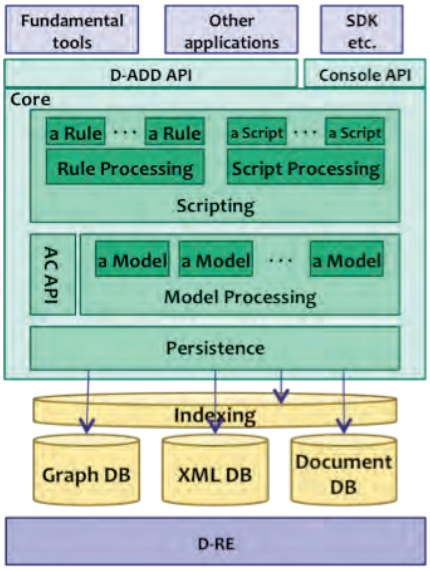
(1) 基本ツール群

図2にD-ADDのアーキテクチャ図を示す。D-ADDでは特にDEOSプロセスの遂行を支援しD-ADDとやり取りするアプリケーションを基本ツール（Fundamental tools）として３アプリケーションを用意した。

- 合意形成支援ツール
- 説明責任支援ツール
- モニタリングルーツ

合意形成支援ツールはDEOSプロセスの合意形成プロセスで利用されることを想定し、合意形成プロセスの遂行を円滑にする。説明責任支援ツールはDEOSプロセスの説明責任遂行プロセス及び障害対応プロセスで利用されることを想定し、新規サービスの開始のみならず、対象システムでの障害の未然回避処理が行われた場合や、対象システムに障害が発生した場合に、D-ADDに記録されている情報をもとにステークホルダの説明責任遂行を支援する。モニタリングツールはDEOSプロセスの通用運用状態で利用されることを想定し、D-Caseモニタリングノードと関連し、対象システムをモニタリングするために必要な支援をする。

図2：D-ADDアーキテクチャ



(2) モデル処理を中心とする中核部

APIを利用してD-ADDにアクセスする。D-ADDの主な機能は中核（Core）部に定義され、スクリプティング（Scripting）部、モデル処理（Model Processing）部、永続化（Persistence）部から構成される。スクリプティング部はルール処理（Rule Processing）部とスクリプト処理（Script Processing）部から構成される。ルールは下位層のモデル処理（Model Processing）部でのモデル情報の変化を記述し、対応する必要なアクションをスクリプトとしてスクリプト処理部が処理する。ルール処理を含む実装の概要は「実装概要」で述べる。

モデル処理部は、対象ドキュメントのデータ型毎に定義されたモデルを処理する。モデル処理では、基本データモデル、合意形成モデル、トゥールミンモデル、会議モデル、D-Caseモデル等の基本ツール群で必要とされるモデルを定義しているが拡張可能である。基本ツール群で扱われるデータには、D-Caseとその付帯情報、仕様書、設計書、運用計画書、会議録、ログ、等の各種ドキュメントがある。ドキュメントには全ての変更履歴が残されている。更にステークホルダ間で合意された対象システムに係るパラメータであるサービスレベル変動許容範囲を調べることで、対象システムが通常運用状態にあることを検証

するためのD-Script、通常運用状態から外れた場合の対応アクションを記したD-Script、等のドキュメントも対象になる。これらは、グラフ構造を基底モデルとする合意グラフとして永続化インターフェースを利用してD-ADDに記録される。ここでは、各モデルに対応したデータに対する操作群が図3に示した合意グラフ演算として定義され、各種ドキュメント間の関連性、合意とステークホルダとの関係、仕様書と対象システムの運用状態との関係、等々のオープンシステムディペンダビリティを担保する為に設定された対象システムの安全基準の充足状況を確認可能にする機能を提供する。

(3) 永続化部

永続化部は下位層の混成データベースに対する抽象化層であり、モデル処理部におけるモデル情報を下位層データベースに記録する。対象とするデータの性質が非構造的であり、関係性が重要であり、時間特性を備え、高速に多種大量のデータ処理が必要である事を考慮し、グラフデータベース、ドキュメントデータベース、そして、Key/Value Storeという３種類のリポジトリを扱う。

図4にD-ADDアーキテクチャをArchiMate®形式[3]で示した。ここではDEOSプロセスを構成する各サブプロセスからD-ADDが利用されるため、基本ツール群、中核部、永続化部が提供するインターフェース（記号－○）を中心に描いた。

2.2. D-CaseとD-ADD

D-Caseモデルは D-Case記述様式である木構造を頂点とエッジとしたグラフ構造で記録している。合意記述支援ツールでは「実装概要」で詳述する合意形成手法を実装している。図5を用いてD-ADD内部でD-Caseをどのように扱っているかを説明する。D-Caseストラテジノードを対象に議論を展開した場合、合意ノードとしてはD-Caseストラテジノード、記述ノード、および主張ノードがリンクする。また、ゴールを対象に議論を展開しても良いし、複数のD-Case間の関係に関して議論を展開しても良い。各々モデルに従って対応ノードが繋がっていく。D-Caseの分解パターンを合意形成支援ツールに応用することで、複数の対象のD-Caseを効率的に扱うことが出来る。例えば、システム分解パターンはサブシステム毎のD-Caseを扱うので関係する文書グラフもサブシステム毎に合意を構成することが可能である。 さらに図5に示すように、D-Caseエビデンスノードに対して

図3：モデル処理と合意グラフ

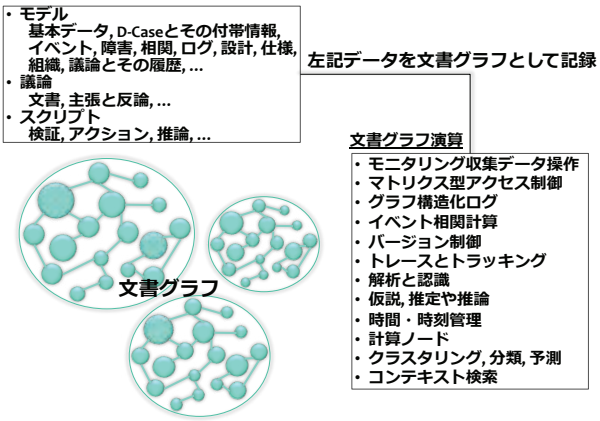
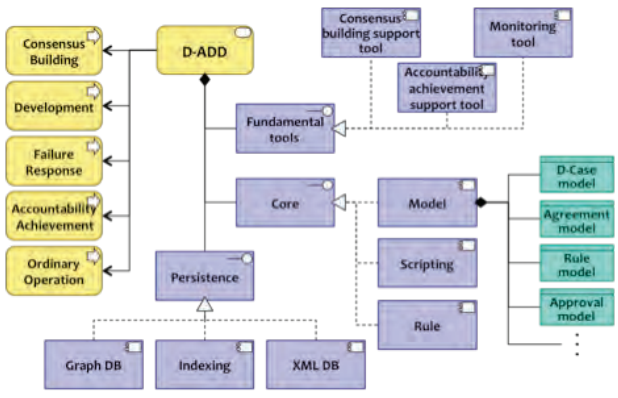


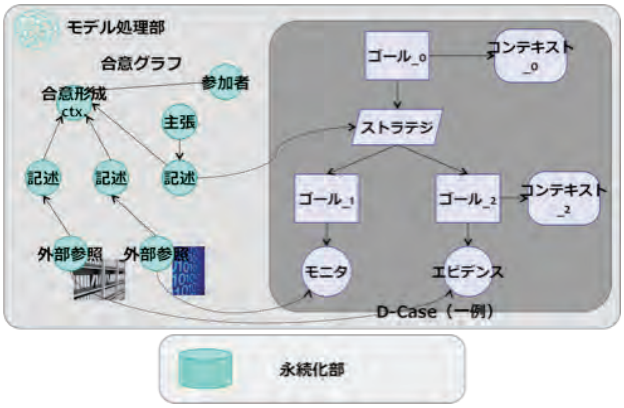
図4：ArchiMate®形式によるD-ADDアーキテクチャ



関係するドキュメントを関連付ければ、ゴール_2を成立させるエビデンスとなる。D-Caseモニタノードに関係するスクリプトを関連付ければ、ゴール_1を成立させるモニタノードとなる。ここでも、それらのドキュメントが用いられた理由・根拠は上記議論の過程としてD-ADD永続化部に記録される。

D-Caseはある時点での対象システムのディペンダビリティに係るステークホルダ間の合意、すなわち議論の結果を表しており、静的である。オープンシステムにおいては性格上D-Caseは動的になる。通常運用状態からの逸脱が起き、変化対応サイクルにてD-Caseが修正されれば迅速にD-ADDに登録され対象システムは再構成され通常運用状態に復帰しないと行けない。さらに、何が議論されたかの経緯をも提示できる必要がある。D-ADDはそれらを可能にする機構を提供している。

図5：D-Caseと合意グラフ



3. D-ADDが支援するDEOSプロセス

DEOSプロセスでは対象システムに障害が発生した場合、ステークホルダの説明責任遂行は必須である。ステークホルダ（例えばサービス提供者）は、サービス利用者や他のステークホルダに対して、障害の状況や原因、復旧計画や障害による損失などを説明しなければならない。

D-ADDはステークホルダが説明責任を成し遂げるために必要な情報の管理を行うが、それは広報部が記者会見の原稿を書くための機能ではない。いつでも必要に応じて、障害対応の状況と、それを裏付けるエビデンスを提供できることであり、それがD-ADDが行う説明責任遂行のための情報の管理である。D-ADDが支援する説明責任遂行の範囲を、DEOSプロセスにおける障害対応サイクルに沿って、障害の未然回避から、迅速対応、原因究明を行い、これらを経て対応のための新たな変化対応サイクルが起動され、ステークホルダによる新たな合意が形成されて、（実際に起きた障害の再発防止策を含む機能を）リリースするまでと定義する。以下に、D-ADDで管理する説明責任遂行の流れをまとめる。

3.1. D-ADD支援による説明責任遂行

D-Caseモニタリングノードに記述されたD-Scriptによって未然回避したフォルトを含めて、D-ADDはすべての障害を記録している。障害が発生した場合には、D-Scriptが自動的に起動され迅速対応を行う。もし自動的に迅速対応ができなかった場合は、障害情報をD-ADDに登録し、人による迅速対応の作業を開始する。原因究明フェーズでは、D-Caseを起点に、D-ADDに保存されているエビデンス情報（D-Caseのエビデンスノードやコンテクストノードに紐付いている重要な文書や合意記述や監視ログ）を調べ、フォルト（誤りの原因）を特定する。その作業記録もまたエビデンスとしてD-ADDに保存する。原因究明を行った

者は、特定した欠陥を元に再発防止のための障害対応策を議論し、合意を形成する。DEOSプロセスの障害対応サイクルはここで完了し、変化対応サイクルへと移行する。

障害対応策に基づき、ステークホルダが要求を抽出するフェーズに入る。障害対応策として決定した改修や機能追加は、原因究明で判明した欠陥の対応のみに限定されるが、要求抽出フェーズではシステムを継続的に発展させるという目的のもと、障害対応策もより良い方向へ発展することが期待される。その結果、新しい機能の追加が議論され、合意されることもあるだろう。D-Caseは障害対応を含む新しい要求に従ってアップデートされる。対象システムは改修作業、テストを経てリリースとなる。

ここで、1 つの説明責任遂行のために必要な準備が完了する。D-ADDは、このような過程でリリースされた新しい機能が、上述の障害に対応した機能であることを、時間を経た後もトレース可能なリポジトリとして設計される。

3.2. D-ADD支援による合意形成：営業放送システムを具体例に

D-ADDの基本機能の一つであるD-Case管理機能は、合意形成のプロセスに深く係わる。そのためD-ADDは、単なるD-Caseの保存や管理だけではなく、DEOSプロセスのサブプロセスである合意形成プロセス全体を支援できる必要がある。

合意形成支援におけるD-ADDの役割を見出すため、オープンシステムディペンダビリティを求められている放送局の基幹システム「営業放送システム」を実例として、D-Caseの合意形成手法を研究した。その結果から、D-ADDに必要な機能を見出した。

ここではその研究を要約し、合意形成手法と、その手法がD-ADDによって可能となることをD-ADDの合意形成支援の面で説明する。

(1) 営業放送システムとは

営放システムは、民間放送局固有の基幹業務システムである。民間放送局は、収入のほとんどをスポンサーとの契約に基づいたCMを放送することで得ている。

営放システムは、複数の部署と多くのユーザが関わる巨大なシステムである。主に、編成部、営業部、放送部の3つの部署の流れ作業をシステム化したものである。編成部が作成した放送スケジュールに基づいて、番組とCMを管理する。各機能は部署毎の業務に特化しているため、行程と行程の間の複雑な情報連携が必要となる。

また、放送事業は、電波法と放送法によって規律されており、近年、相次ぐ放送事故やCM間引き問題により、総務省は放送局に対し再発防止を求めた行政指導を行っている。[4]法律や放送基準などの変更による外部からの変化に柔軟に対応する必要があり、24時間365日稼動し続けることが求められる営放システムは、高いオープンシステムディペンダビリティを求められている。

(2) 営業放送システムのD-Caseにおける合意形成の課題

営放システムのD-Caseは、上記のようなシステムの特性を反映するため、巨大化し、複雑化し、専門性が高くなる。一人で記述することは不可能であり、必然的に、専門知識を有した各部署のメンバーがステークホルダとして必要となる。D-Caseの詳細な部分については各部署にて記述し、それら部分部分を統合し、最終的にトップゴールを達成しているかについて、確認し、議論し、合意しなければならない。また、

変化対応によりD-Caseを修正する時には、修正箇所は1つだとしても他部署の責任範囲への影響を見極めなければならない。これらの作業を人間だけの力で行うことは難しい。

このような事実から、巨大で複雑なD-Caseの合意を形成するための課題を、次の4つにまとめた。

A. 多人数でのD-Caseの操作の必要性

B. D-Caseの整合性の確保

C. D-Case修正時の影響範囲の明確化

D. ステークホルダの責任範囲の明確化

多数の部署が絡む故のこれら課題を解決するために、どのような合意形成の手法が有効か、次に示す。

(3) 合意形成のためのV-モデル手法

巨大で複雑で1人のステークホルダの管理が不可能なD-Caseの場合、適切な合意形成は組織構造によってもたらされる。

組織はツリー構造を持っている。組織の権限と責任は、ツリー構造を通じて下位組織へと委譲される。一方、D-Caseもツリー構造を持っており、ディペンダビリティの達成を記述したノードは、上から下へと詳細化される。組織の委譲レベルをD-Caseのツリー構造に反映させ、組織のトップダウンによる計画フェーズと、ボトムアップによる検証フェーズの2段階で合意を形成する手法を提示する。

A. 合意形成の場

合意は、ある目的のため、会議中に、ある事象に対して、様々な利害を持った複数の人々によって為される。そのような合意形成の必要条件を、猪原氏の『合意形成学』[5]より言葉を借りて「場」と呼ぶ。

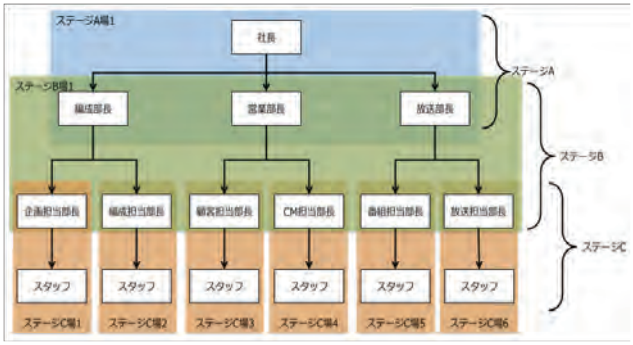
場には、合意しようとする内容を理解し、責任を持てる者が集まるべきである。経営会議には企業トップや役員が出席し、新製品のマニュアルについての会議では製品の開発部署や販売部署、運用部署などが出席するだろう。場は企業の組織構造と密接に結びついている。

この研究では、権限の委譲がなされる2つの組織レベルを同じ合意形成の場に設定することで、上位で合意したことを、下位の合意形成の場に正確に伝えることができると考えた。図6は放送局の組織構造元に、合意形成の場を設定したものである。2つの組織レベルを含む階層をステージと呼ぶ。

社長と3部長の2階層を含むステージAは、D-Caseのトップゴールを決めシステム全体の開発、運用、議論に影響する前提ノードを明らかにすることが目的のため、場は1つとした。ステージBは、3部長と各業務の担当部長の2階層を含み、D-Case全体の大ざっぱな構造を決めることが目的である。部署間の調整を図る必要があるため場は1つとした。ステージCは、担当部長と実際に作業を行う作業者の2階層を含む。D-Caseを完成させることが目的である。証拠ノードをつけられるまで詳細に分解する必要があるため、業務毎に場を設定した。

このステージと場の設定によって、巨大で複雑なD-Caseの体系的な構築と合意を可能とする。次に、その手順をV-モデルライフサイクルを用いて説明する。

図6：ステージと場



B. 合意形成のV-モデル

この手法は、D-Caseの構築と合意形成の行程を大きく2つに分割して進める。計画の合意フェーズと検証の合意フェーズである。その全体像を図7にまとめた。

1) 計画の合意フェーズ

計画の合意フェーズの目的は、証拠ノードを除いたD-Caseを定義し、合意することである。組織が持つトップダウンによる意志決定を権限委譲モデルを用いて、D-Caseは上から順に生成する。

ステージAでは、トップゴールと全体に関わる前提ノードを決定する。次にトップゴールをどのように達成するかの戦略を設定し、さらに戦略ノードを明確にするためにサブゴールの提示を行う。

ステージBでは、ステージAで合意された制約の範囲内で、全体の計画をたてる。実際の業務を理解しているステージBのステークホルダは、ステージAが提示した方向性を元に、部署間の調整を行いながら、さらにゴールを分解し、合意する。

ステージCでは、ステージBで合意された制約の範囲内で、D-Caseを業務毎に詳細化する。1～6の各場で別々に記述した部分的なD-Caseは、全体のD-Caseに配置されるためには整合性のチェックを行う必要がある。

2) 検証の合意フェーズ

検証の合意フェーズの目的は、すべてのリーフゴールに証拠を付加し、各場の責任範囲のゴールが達成していることを確認し、合意することである。このフェーズでは、計画の合意フェーズとは逆に、ボトムアップで組織の下から検証を行い、下位レベルの検証結果を受けて、上位レベルがゴール達成を合意する流れである。

(4) D-ADDによる合意形成支援

先に提示した4つの課題解決の検証のため、この合意形成手法を、D-ADDのアプリケーションツールとして実現している。

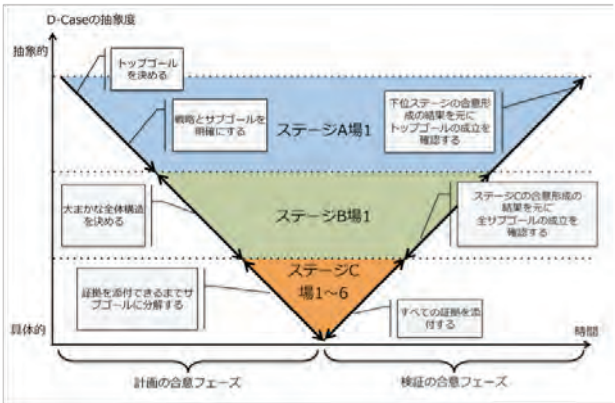
A. 多人数でのD-Caseの操作の必要性に対しては、D-ADDとD-Case編集ツールを連携する。

B. D-Caseの整合性の確保の課題に対しては、D-ADDの整合性支援機能の辞書機能およびAgda[6]を利用する。

C. D-Case修正時の影響範囲見積の課題に対しては、D-ADDの整合性支援機能の1つである索引機能およびAgdaを利用する。

D. ステークホルダの責任範囲の明確化は、D-ADDの組織・人・権限支援機能を利用する。また、合意形成の状況を表示する機能を提供し、変化していくD-Caseの合意形成の状況を、D-ADDによって、組織・人という観点から可視化する。

図7：V-モデル図



4. 実装概要

D-ADDの実装はDEOS体験版として利用することができる。その実装の詳細を記すにはスペースが足りないので、ここではその概要を述べる。実装にはJava言語[7]とScala言語[8]を用いた。Java言語は主に基本ツール群と永続化部の記述に、Scala言語は主に中核部の記述に用いた。

ルース処理部が対象とするルールは図8に記載のクラス図に示すオブジェクトとして実装した。ルールはモデルの状態が変化したときに発火する（ルールが成立し対応する処理が実行される）。ルールにはモデル処理部のモデル情報への参照を記述できる。また、ルールにはモデル状態の取得を拡張するためのプラグインが利用できる。例えば、統計解析のためのプラグインを利用することができる。

モデル処理部が対象とするモデルには、合意形成支援のためのAgreementモデル、D-Caseを記録するためのD-Caseモデル、モニタリング対象のTargetモデル、ルールモデル、ルールやスクリプトをD-ADDに動的に配備するためのDeploymentモデル、等が定義されている。一例として図9にD-Caseモデルのクラス図を示す。

永続化部では、説明責任遂行で重要な役割を果たす合意グラフは、グラフフレームワークを利用して記録される。今回はグラフフレームワークとしてオープンソースのTinkerPop[9]を採用し、グラフデータベースとしてはオープンソースのNeo4j[10]を採用し、Play!フレームワーク[11]を用いてRESTfulアクセスできるように実装した。Play!ではグラフデータベースとの接続が未対応だったので、Play!からTinkerPop経由でNeo4jにアクセスする機能を実装した。Play!はプラグインの仕組み（モジュール）を備えているので、モデル、および上記グラフデータベースとの接続部をPlay!のモジュールとして実装した。また、基本ツール群もPlay!アプリケーションとして実装した。

図8：ルールクラス図

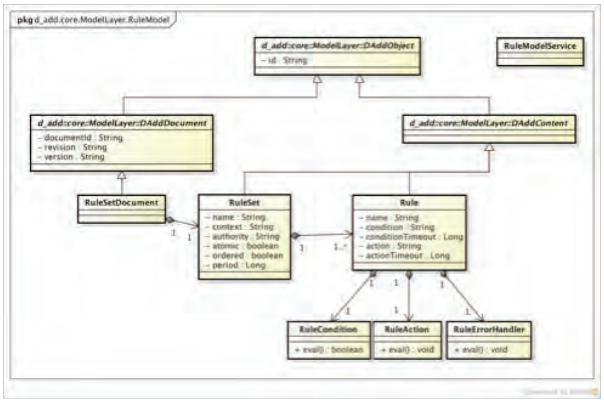
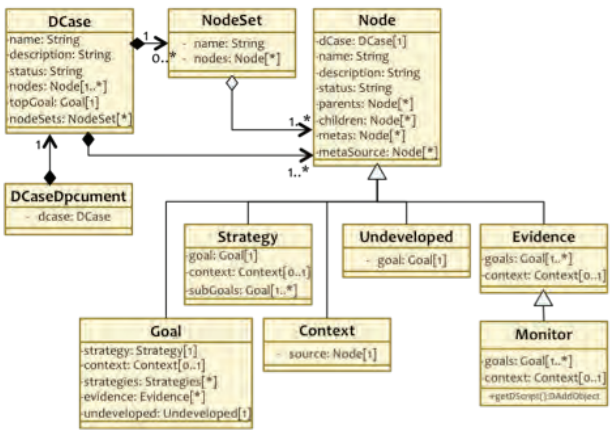


図9：D-Caseモデルのクラス図



5. 実ビジネスにおけるD-ADDの利用

巨大ITシステムでは、一般的に保守的に設計開発されることが多い。このためややもすると、ウォーターフォールで経験済みの技術を使って、納期通り、工期通りに製造されることが選択される。

結果的に品質が向上していれば、正しい選択である。ところが、DEOSの調査結果では、システム事故はシステムの巨大化、複雑化によって減少することはなく、むしろ重篤なシステムダウンが社会的な脅威となってきている。

D-ADDは、合意記述データベースとして研究開発が進められたが、その理由の一つは設計品質の向上のためである。巨大ITシステム開発では、設計書に記載されない重要な要件情報、設計情報が開発期間が長ければ長いほど増える傾向にあり、またその発生時期も不定期であるが故、関係者間での周知徹底、設計情報への反映が疎かになる。これだけが原因ではないが、たった一つの設計齟齬でも下流工程に与えるインパクトは無視できない。

要件の整理、定義から設計開発まで数年、関係者の数は数百人を超えるというようなケースでは、仮にウォーターフォールであろうがアジャイルであろうが、発生した新たな設計情報は、整理統合されてこれまでの設計情報へ正しく反映されなければならないが、こうした仕様記述についての具体的な施策は、発注者側と受注者側の双方の努力がなければ目的を達成することが難しいのだが、ここに具体的な方法論が久しく現れていない。例えPMBOKなどのプロジェクト管理技術が導入されたにしても、設計情報の継続的なインテグレーションは難しいのである。実ビジネスにおけるD-ADDの利活用では、この本質的な課題に迫る設計情報の継続的なインテグレーションを目指して、合意形成のあり方、全体情報との再統合の方法について、D-Case技術をさらに活用して実践的な開発を進める段階に入ることを検討している。

6. まとめ

DEOSプロジェクトにおいて、本格的にD-ADDの研究開発が始まったのは2012年からであり、D-ADDはプロジェクト最後の構成要素でありプロジェクト成果のインテグレーションの役割が期待されている。対象システムのオープンシステムディペンダビリティを担保するにはDEOSプロセスを効果的に運用可能する支援環境が必要である。DEOSプロセスは対象システムのライフサイクル全般に係るのみならず、複数の対象システムが連携する環境をも対象にしており、D-ADDはそれに向けて最適化されている。

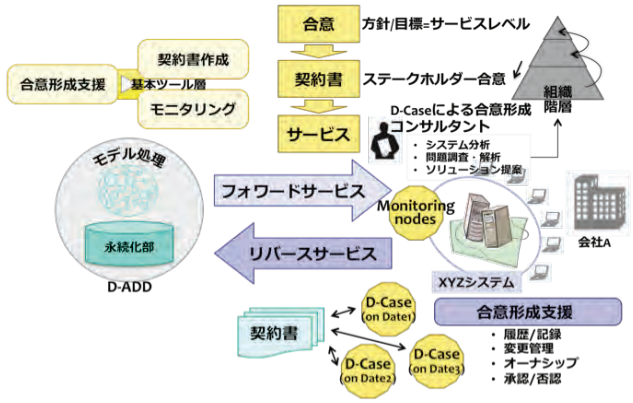
まず、D-ADDの適用領域をアプリケーションサーバと定め、そこでのD-ADDに最低限必要な機能としてのグラフデータベースに関して検証した。グラフデータベースを選んだ理由は、第一義的にはD-Caseのインテグレーションデータベースである必要があり、D-Caseはまさにグラフ構造を有しているからである。次に、合意形成支援ツールを軸とし、D-ADDに必要な機能を整理しプロトタイプを開発した。我々は（現在はまだ一部であるが）D-ADDの研究・開発そのものにもD-ADDプロトタイプを利用している。現在では、D-ADDと主たる格納情報であるD-Caseとの接続を軸としD-ADDの研究・開発を進めている。領域の研究成果であるD-Case Weaverを改造し、構造化文書リポジトリに格納された文書情報との関連性を保持しつつD-Caseを格納するというプロトタイプシステムを開発した。本稿でも述べている説明責任遂行支援を軸に、「営業放送システム」を対象システムとするシナリオを用いてプロトタイプを開発し、障害時の機能について定義した。

さらにD-Caseを格納するD-ADDとしてオンライン性能を検証するために、D-Case編集ツールの一つであるAssureIt[12]と接続し、複数同時ユーザ、複数のD-Caseの接続についても継続的な研究開発を行っている。D-ADDの内部機構の一つであるD-ADDエンジンは、D-Scriptの性能要求の一部を担当して、D-

Script連携を容易にするための機能も実装した。

D-ADDは、横浜国立大学 倉光チームと共同で、D-ADD、D-Case、D-Scriptが連動して動く領域成果のインテグレーション環境の構築を行っている。同環境にて、木下チームの「利用者指向ディペンダビリティの研究」と共同で統合デモシナリオの検討、 慶應義塾大学 河野チームの「耐攻撃性を強化した高度にセキュアなOSの創出」の実験データを用いたデモ、さらにはDEOS協会設立に向けたDEOSプロセスの産業界への優位性デモ開発をDEOS研究開発センターと共同で行う予定である。そのような最終目標をめざし、今後は数度のリファクタリングを経て、DEOSプロセスの「体験版」としての実装を進めている。成果は、クラウド上（例えばAmazon Web Services）に展開し公開する予定である。

図10：D-ADDと関係ビジネス



参考文献

[1] Mario Tokoro, Editor, “Open Systems Dependability – Dependability Engineering for Ever-Changing Systems, ” ISBN: 978-1-4665-7751-0, CRC Press, 2013.

[2] 松野裕, 山本修一郎 (2013)『実践D-Case ディペンダビリティケースを活用しよう!』, アセットマネジメント, 愛知, ISBN: 987-4-86293-091-0.

[3] http://www.opengroup.org/subjectareas/enterprise/archimate

[4] 日本民間放送連盟編 (2007)『放送ハンドブック改訂版』, 日経BP社, 東京, 666pp

[5] 猪原健弘編著 (2011)『合意形成学』, 勁草書房, 東京, 282pp

[6] Yoshiki Kinoshita, Makoto Takeyama, Makoto Hirai, Yoshifumi Yuasa and Hiroyuki Kido, “Assurance Case Description using D-Case in Agda”, D-Case in Verification and Validation, Hyogo, Japan, National Institute of Advanced Industrial Science and technology, 2013, ch.1, pp.1-18.

[7] http://www.java.com/ja/

[8] http://www.scala-lang.org/

[9] http://tinkerpop.com/

[10] http://neo4j.org/

[11] http://www.playframework.org/

[12] https://github.com/assureit

合意記述データベース

オープンシステムディペンダビリティと D-Caseを繋ぐリポジトリ

d-add.org

「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」(DEOSプロジェクト)は科学技術振興機構(JST)の戦略的創造研究推進事業CRESTの研究領域のひとつです。

