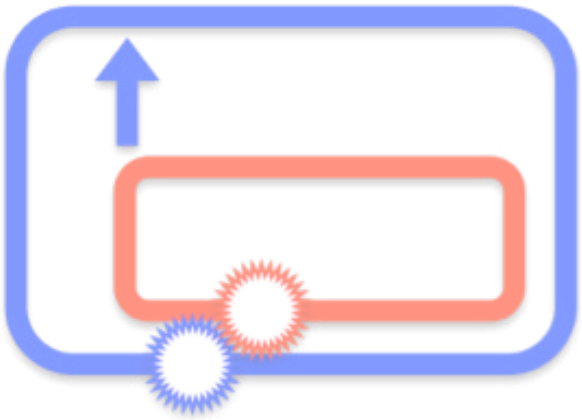
合意記述データベース

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5月8日

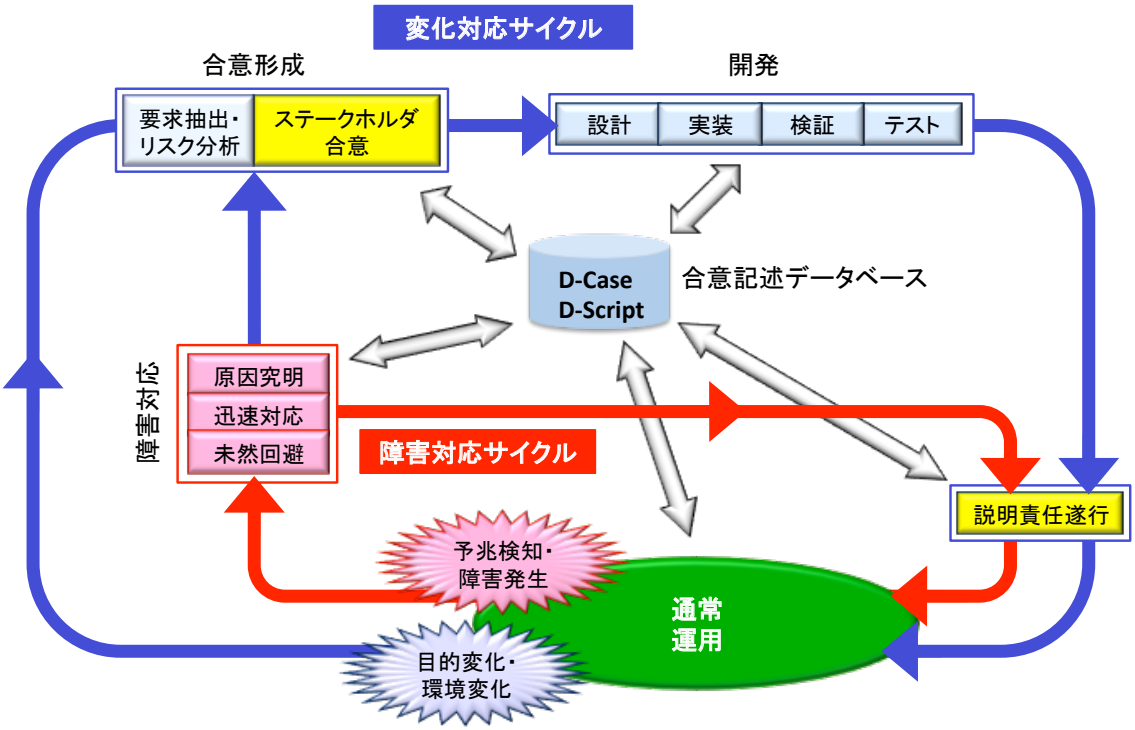
はじめに

ITシステムのディペンダビリティが注目されてから久しいが、依然として障害発生事例には事欠かない。システムは巨大化し、社会インフラとしても非常に重要な役割を担っているので、一度事故が起こると全国、国民すべて、場合によっては世界中にその影響を及ぼす事になる。銀行システム、通信システム、鉄道、そしてクラウドシステムで深刻な障害を起こしており、巨額の賠償金に発展するケースも珍しくない。もし、大きな事故が起これば責任の追及が行われ、過誤が認定されれば厳しい罰則を受けることをステークホルダは覚悟しなければいけない。

そこで、ITシステムのディペンダビリティを担保する技術体系をまとめ、制度化、さらには事業化を目指すべく2006年に独立行政法人科学技術振興機構（JST）はCRESTプログラムの1つとしてDEOSプロジェクトをキックオフした。その様な技術体系を一言で言えば、システムの安全基準を定義して、それを逸脱するようなことがあれば、異常事態に陥る前に回避するか、異常事態に陥ってもすぐさま復旧させることが出来る技術体系であり、我々はオープンシステムディペンダビリティとして定義した。

次に、我々はオープンシステムディペンダビリティをプロセスの観点から整理し、DEOSプロセスとしてまとめた。図１に示すDEOSプロセスは複数のプロセスから構成される２つのサイクルからなっている。変化対応サイクルは上流プロセスにおけ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為に必要とされるプロセスとして、合意形成、開発、説明責任遂行の３プロセスを定義している。障害対応サイクルは対象システムの運用時に必要とされるプロセスとして、障害対応と説明責任遂行の２プロ

図１：DEOSプロセス



セスを定義している。普段は両サイクルとも対象システムは通常運用状態にある。DEOSプロセスの詳細は文献[1]に述べられているが、その特徴を一言で表現するなら、企画や設計という上流から運用までの対象システムに関するライフサイクル全体に係るオープンシステムディペンダビリティを担保する為に、対象システムに係る人と物の行動を律するプロセスであり、対象システムに関わる人々（ステークホルダ）からの対象システムに対する要求に関する合意を始め、あらゆる議論に関する合意をベースに、対象システムを通常運用状態に維持すると同時に、ステークホルダの説明責任遂行を可能とする。

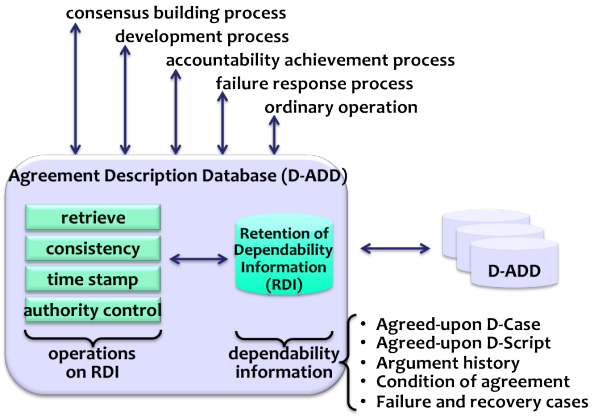
各プロセス間での共通言語としてD-Caseが設計されている[2]。また、各プロセス間の共通のリポジトリとしてD-ADD（DEOS Agreement Description Database）を導入した。DEOSプロセスは対象システムのライフサイクル全般に係るプロセスであり、ステークホルダの合意が対象システムの通常運用状態を定義している。D-ADDはステークホルダ合意と対象システムに存在するプログラム・コード、及び対象システムの運用状態との間の一貫性を常に保つための機構を提供すると同時に、いつでも必要な時点でステークホルダの説明責任遂行を可能とする機構をも提供する。

以下、本ドキュメントではDEOSプロセスとD-ADDとの関係を詳細に議論する。次に、営業放送システムのシナリオ事例をベースとしてD-ADDの実装に関して述べる。最後に、D-ADDの実ビジネスにおける利用、及びD-ADDとソフトウェア開発プロセス革新について考察し、本稿をまとめる。

DEOSプロセスとD-ADD

DEOSプロセスを構成している各プロセスからD-ADDはアクセスされると同時にD-ADD同士も情報を交換する（図２）。変化対応サイクルを構成するプロセス群からはD-ADDの提供する基本ツール群を用いて対象システムとやり取りする。障害対応サイクルを構成するプロセス群からも同様にD-ADDとのやり取りを通じて、各々のプロセスを遂行している。具体的には、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を保証する。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。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

図２：DEOSプロセスとD-ADD



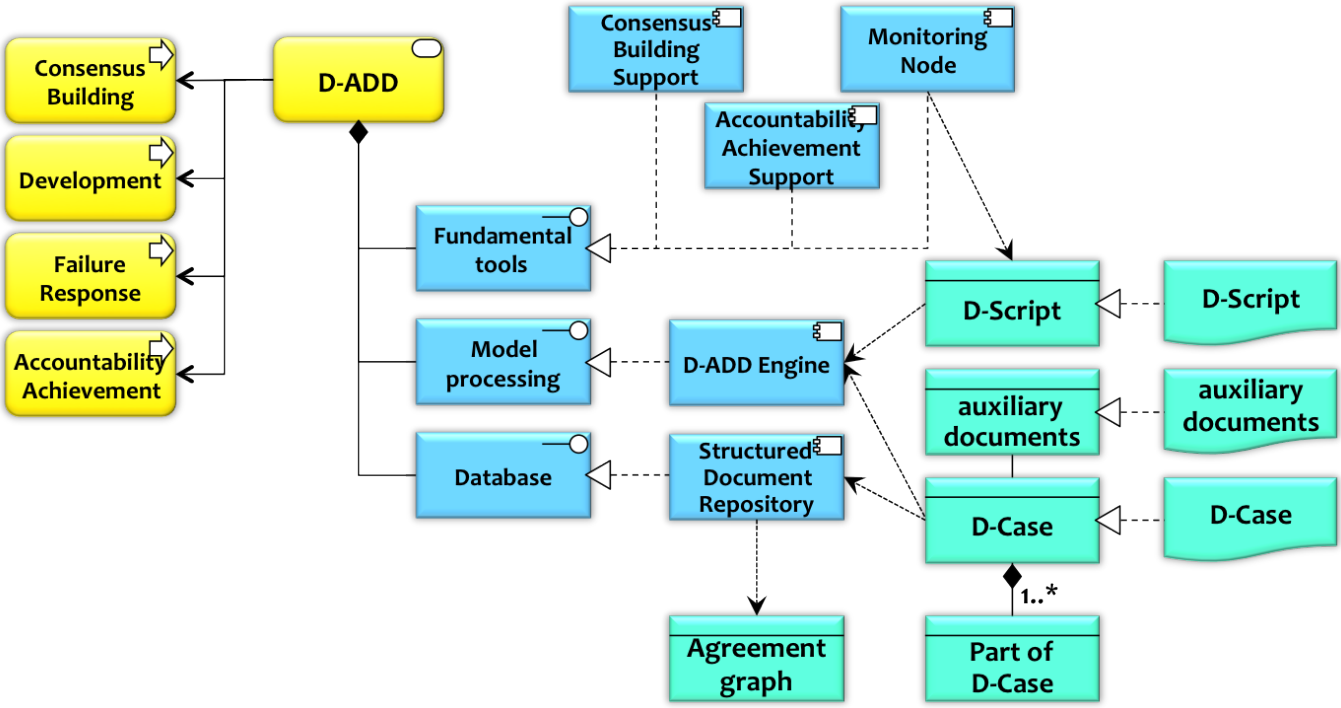
示する。説明責任遂行プロセスにおいてD-ADDは新サービスの立ち上げ、改修、および障害対応に関してステークホルダが説明責任を遂行するに十分な情報を入手できるように支援する。D-ADDは通常運用状態においても対象システムとのやり取りを通じて、対象システムのオープンシステムディペンダビリティが担保されている事を確認し、状況に応じて障害対応サイクル、あるいは変化対応サイクルを起動する。この様にD-ADDはDEOSプロセスを遂行する上で必須の構成要素である。

D-ADDアーキテクチャ

図３にD-ADDアーキアーキテクチャをArchiMate®形式で図示した。D-ADDの提供するサービスは基本ツール群（図中ではFundamental tools）、モデル処理（Model processing）、データベース（Database）の３種のインターフェースから構成される。

基本ツール群は次の３個のコンポーネントを利用して実現される。合意形成支援ツール（Consensus Building Support）はDEOSプロセスにおける合意形成プロセスの遂行を円滑にする。ステークホルダ間の議論過程において合意されたドキュメント情報を、上記モデル処理インターフェースを利用して処理し、D-ADDに記録する。説明責任遂行支援ツール（Accountability Achievement Support）は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

図３：D-ADDアーキテクチャ

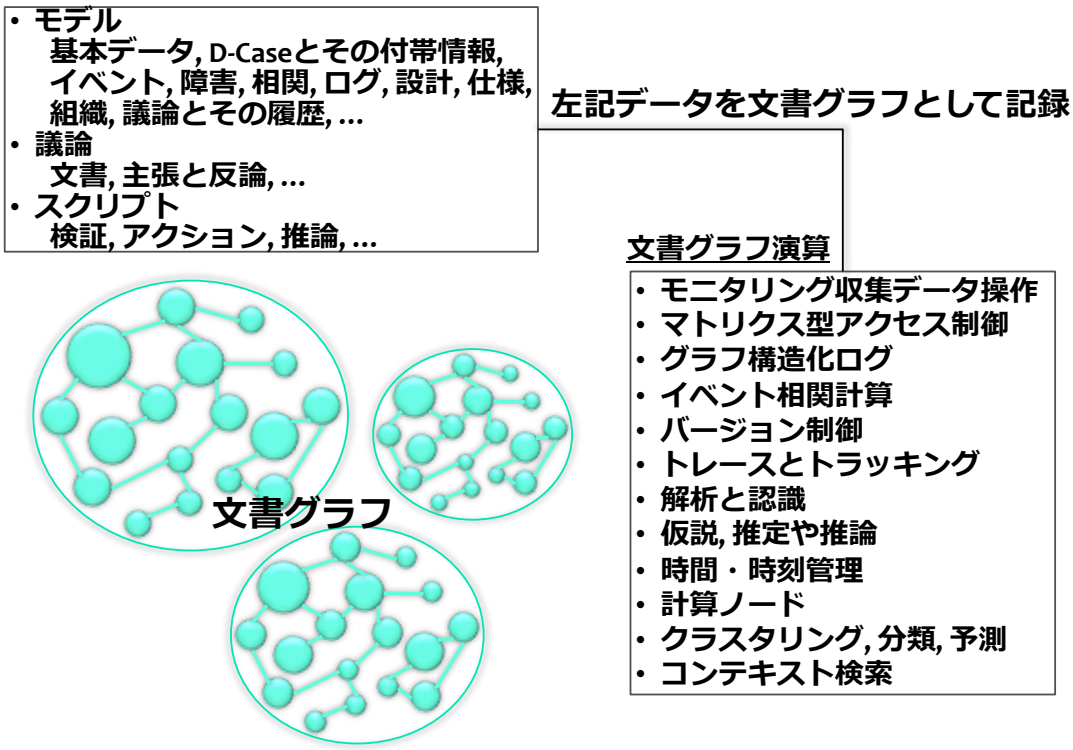


を支援する。モニタリングツールはD-Caseモニタノード[2]と関連し、対象システムをモニタリングするために必要な支援をする。

モデル処理はD-ADDエンジン（D-ADD engine）コンポーネントを利用して実現される。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する。基本ツール群で扱われるデータ群には、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In-Operation Range[1]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スクリプト、通常運用状態から外れた場合のアクションを記したスクリプト、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文書グラフとしてデータベース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 4 に示した文書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リアルタイムで確認可能にする機能を提供する。

データベースは構造化文書リポジトリ（Structured Document Repository）コンポーネントを利用して実現される。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

図 4 :D-ADDにおけるモデル処理



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。他のインターフェース群からはこれらのリポジトリには透過的にアクセスできる。

D-CaseとD-ADD

D-ADDモデル層のD-Caseモデルは文献[2]に記載のD-Case記述様式である木構造を頂点とエッジとした文書グラフとして記録している。上記合意記述支援ツールではD-Case記述に対してどの部分を議論の対象にするか、自由に設定出来る。図５を用いて説明する。例えば、D-Caseストラテジノードを対象に議論を展開しても良い。この場合、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、一部のサブ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。文献[2]に記載の議論分解パターンを合意形成支援ツールに応用することで、対象の複数の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リポジトリ層に記録される。

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を表しており、議論の結果として静的である。オープンシステムにおいては性格上D-Caseはダイナミックになる。通常運用状態からの逸脱が起きればD-Caseは迅速に修正されなければいけないし、何が議論されたか、その経緯も提示できる必要がある。D-ADDはそれを可能にする機構を提供している。

図５：D-CaseとD-ADD

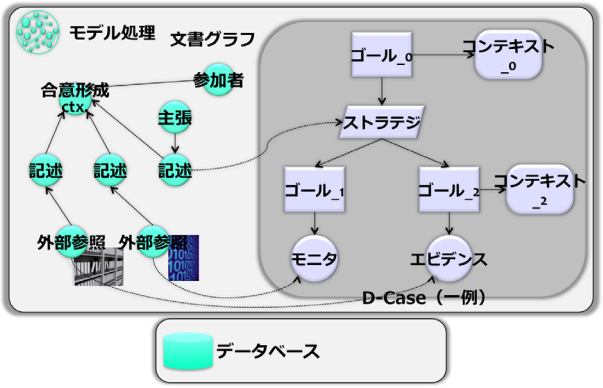
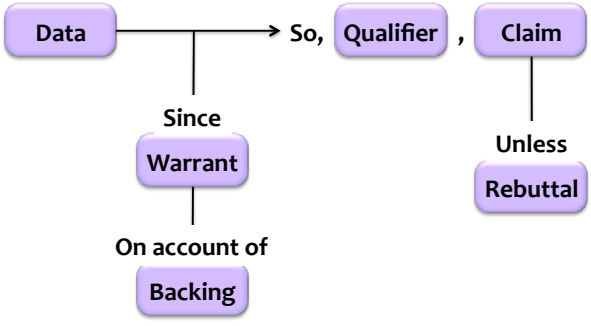


図６：トゥールミンモデルの一例



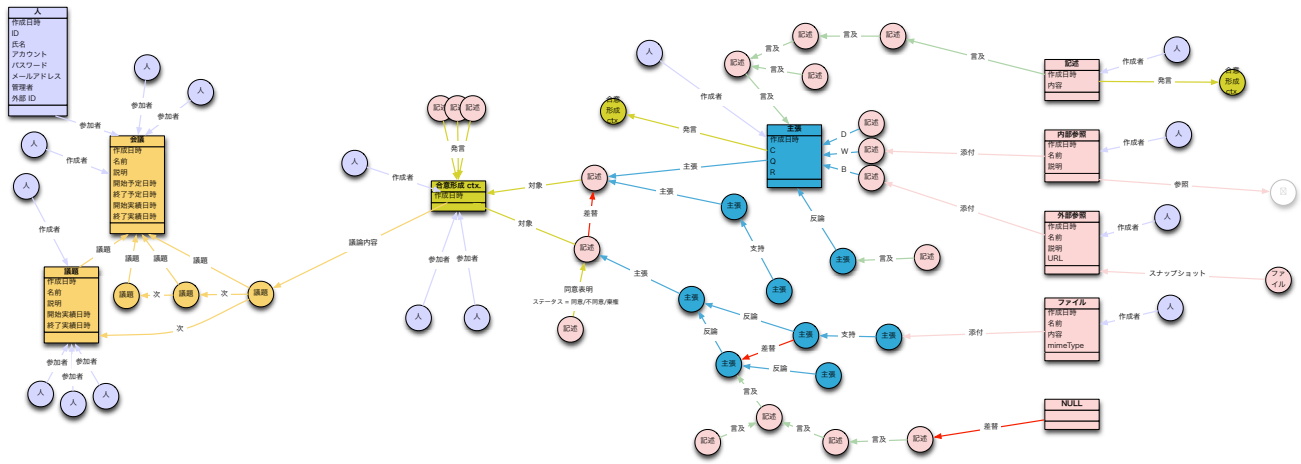
合意形成支援とD-ADD

合意形成支援ツールはDEOSプロセスにおける合意形成プロセスで利用されることを想定している。ここではトゥールミンの議論モデルを採用するが、他の手法を用いても良い。例えば、OMGの規定するSACM[4]を用いても良い。図６にトゥールミンモデルの一例を示す[3]。議論に当たって発言者は自らの発言内容が図６で示す構造を持つように発言する。議論の結論である主張（Claim）と、それを裏付けるデータ（Data）、そして主張に対するデータの根拠（Warrant）を考える。根拠には裏付け（Backing）が必要である。また、主張である結論が完全で無い場合もあるので限定（Qualifier）することが出来る。議論ではこの主張が当てはまらないことを主張する論駁（Rebuttal）も考える。実際には図６の構造が複合的に複数の発言者に関連し複数存在し議論を構成する。

合意形成モデルでは上記トゥールミンモデルでの主張、限定詞、論駁を頂点とし、支持や反論をエッジとする文書グラフとして構造化文書リポジトリ層に記録する。また、必要な付帯モデルとして発言モデルと会議モデルを導入した。前者は発言間の関係を記録し、後者はトゥールミンモデルに則って会議を進める際に必要な複雑な議論構造の関係を上記リポジトリに記録する。

図７に合意形成に係るモデルの全体像を示す。まず、合意形成の場として「合意形成コンテキスト（ctx.）」を定義する。複数の「参加者」と複数の「記述」が「発言」として合意形成コンテキストに関連付けられる。一方、今回は会議の形態で合意を形成する場合を想定する。「会議」モデルが定義され、複数の参加者と「議題」が関連付けられる。複数の議題は「議論内容」として合意形成コンテキストと関連付けられる。上記議論モデルにしたがって、記述は「主張」と関連付けられ、記述間には「言及」の関係が存在する。主張間には「論駁」（図７では反論と記載）の関係、あるいは「支持」の関係が存在する。記述も主張も「差替」することで置き換える事が出来る。また、記述や主張は「添付」として「内部参照」「外部参照」あるいは「ファイル」を参照できる。記述間の

図７：合意形成に係わるモデル



関係には「同意表明」があり、そのステータスとして「同意」「不同意」「棄権」を定義する。合意形成支援ツールは合意形成の過程を図7のモデルにしたがって必要な関連情報を構造化文書リポジトリ層に記録していく。

説明責任遂行支援とD-ADD

D-ADDに記録したD-Caseおよび合意形成の結果は、説明責任遂行プロセスで使用される。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。社内あるいは社外のステークホルダに向けて情報を発信するとき、例えばそれが社会的に大きな影響を及ぼすような障害であれば、企業を代表する社長・役員などによる記者会見が行われるかもしれない。その場合には広報部が説明のための原稿を用意するであろう。もっと軽微な、一般利用者向けネット接続が5分途切れたというような障害の場合には、サービス利用者向けのウェブサイトにお詫びのリリースを掲載するだけかもしれない。このように、障害の規模や内容、サービスの種類、説明の相手によって、果たすべき説明の方式、そのタイミング、説明内容は異なる。

D-ADDは説明責任を成し遂げるための管理を行うが、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、それを裏付ける証拠を提供できること、それがD-ADDが行う説明責任遂行の管理である。D-ADDでは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だけではなく、障害の未然回避から、迅速対応、原因究明を経て、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。

D-Caseとモニタノードに記述されたD-Scriptによって未然回避した誤りを含めて、D-ADDはすべての障害の記録を行う。障害が発生した場合には、D-Scriptを自動的に起動し迅速対応を行う。もし自動で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、欠陥（誤りの原因）を特定する。その作業記録もまた証拠として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、一つの説明責任遂行が完了する。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。

実装

本節ではD-ADDの機能群の内、D-Caseの格納（合意記述を含む）と説明責任遂行の管理について、WEBアプリケーションとしてシステムを構築したので、その実装に関して述べる。本実装では、放送局の基幹業務システムである「営業放送システム（以下、営放システム）」を例にとり、営放システムで障害が発生した場合のシナリオに基づいて開発を行った。まず営放システムと放送局における障害について概略を説明する。次に障害発生から原因究明までの本実装の動きを説明し、最後に本実装で利用した技術をまとめる。

シナリオ：営業放送システム

営放システムは大きく3つのサブシステムに分割できる。放送スケジュールを作成し管理する編成システム、スポンサーやCM素材を管理する営業システム、番組を管理するとともに最終的に完成した1日分の放送情報を放送機器に送信する役目を負う放送システムである。放送スケジュールが完成すると、営業がスポンサーを獲得しCM素材をスケジュールに沿って配置する。番組の制作が終わるとその素材を配置し、最終的に1日分の放送データを作成して放送機器に送信するまでが、営放システムのフローである。

放送局における一番の障害は放送事故である。放送機器の故障による事故、操作ミスなどによる人為事故、内容に問題がある事故など、その内容は多岐にわたる。昨今ではモラルの低下に伴う倫理的に問題のある番組が流れてしまったというような事故が多発している。そのような事態を防ぐために、放送局では放送2日前にす

べての番組およびCMを視認チェックするという作業が行われている。また放送局にとってその収入源であるCMは、スポンサーとの契約に基づき放送される時間や回数などが決まっており、正しい時間に流れなかったなどの事故が起こるとスポンサーへの補償が発生する。そのためCMに関しては営放システムでも厳格に管理される。

実装概要

本実装では上記放送局の業務や発生しうる障害情報をもとに、D-Caseのトップゴールを「放送事故を起こさない」とした。起こりうる障害をリスクとしてリストアップし、それらの障害を防止あるいは未然回避するD-Caseを作成し、営放システムはそのD-Caseに基づいてモニタノードによって監視されていることと想定した。このD-Caseは、『D-Case　Weaver』で記述した。合意形成を経て保存すると、D-ADDの構造化文書リポジトリへ格納する。構造化文書リポジトリでは、D-Caseは1つのXMLファイルとして扱いバージョン管理を行う。

構造化文書リポジトリにD-CaseのXMLファイルを格納すると、記述されたノードを解析し、D-Caseのツリー構造を文書グラフへ展開する。文書グラフでは、D-Caseのノードのうち、コンテクストノード、エビデンスノード、モニタノードの3種類を抽出し、それらをノードとエッジとして格納するように定義した。

この文書グラフ内では、D-CaseやD-Caseのノードに関連する文書、合意記述結果、監視ログなどはノード（節点）として扱われ、各々はエッジ（枝）により結合される。例えば、あるD-Caseについての合意形成の結果を見たい場合には、該当するD-Caseのノードに紐付いている合意記述結果のノードを見つけて閲覧する。コンテキストノードに記述される要件定義書や設計書などもファイルをそのままノードに添付することができるので、文書グラフに表示されたコンテクストノードを閲覧し、そこから添付した要件定義書ファイルを見つけることができる。このように D-Caseを中心に、文書間の繋がりを辿る形で、必要な情報に行き着けるように設計し実装した。

またD-Caseの合意形成は、『D-Case Weaver』上から可能とした。合意したいノードを選択し「合意開始」ボタンを押すことで、合意形成支援ツールを起動する。合意したいノードがD-Caseのトップノードの場合には、D-Case全体が合意対象となる。木構造の一部の場合には、選んだノード以下のすべてのノードが合意対象となる。合意形成支援ツールでの議論や合意の結果もまた構造化文書リポジトリに格納され、文書グラフ上では合意対象となったノードと紐付けられる。D-ADDはこの様に、D-Caseとそれに関連付く文書、合意記述、監視ログの構造化文書リポジトリへの格納と、それらの関係を表現する文書グラフへ展開する。

ここまでは営放システムを元にしたプロトタイプの前提である。「それでも障害が発生した」というところからプロトタイプのシナリオはスタートする。録画番組を放送中に、番組の最後の方で番組が途切れていきなりCMが放送された、という放送事故（障害）が起こったと想定した。またこのシナリオではD-Scriptによる未然回避はできなかったとした。

次に説明責任遂行管理の機能について説明する。D-ADDが管理すべき説明責任遂行の流れ

は前述した通りであるが、本実装で実現した範囲はその一部である。障害発生から人間の作業による迅速対応を経て原因究明を行い、障害の原因を特定するまでとした。

今回のプロトタイプでは、障害は人間が発見するシナリオとした。放送機器を管理する部署の担当者が、放送中の映像を確認していたとき、番組が途切れたという事実を発見した。その担当者は、放送機器が正常に動作しているかを確かめ（正常に動作していた）、またCMへの影響を確認した（CMは時間通りに正しく放送されており影響はなかった）。同時に上司に報告した。放送局内に設置された放送事故対策委員会が報告を受け、障害対応を行うよう指示を出した。

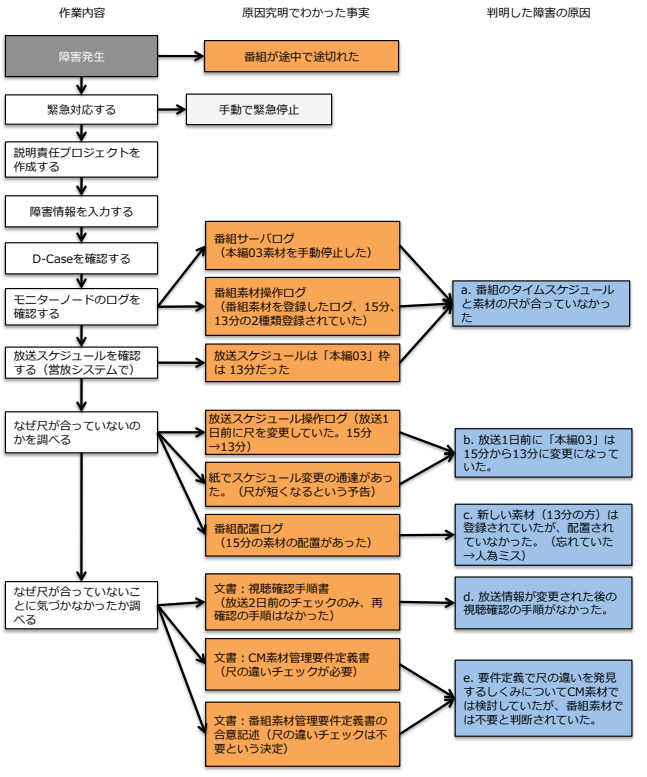
営放システムの運用担当者はD-ADDのツールの一つである説明責任支援ツールにアクセスし、説明責任遂行プロジェクトを立ち上げる。説明責任遂行プロジェクトでは、まず、そこで障害対応にあたるユーザを設定する。次に障害情報の入力を行う。障害情報は、システムの監視によって検知する場合と人間が検知する場合の両方を想定し、あらかじめ対象システムや業務に合った障害内容の組み合わせを定義できるようになっている。ユーザが入力する場合は選択肢を選択することで設定できる。

ユーザが障害情報を入力すると、D-ADDはその情報を元にD-ADD内に格納した情報を調査し、初期調査結果としてユーザに提示する。例えば、障害情報にて「サブシステムである編成システムが停止している」という項目を選択している場合には、実際に編成システムのログをD-ADDが分析し、それが停止していることを示すエビデンスをユーザに提供する。これは事前に障害情報の内容に基づいて設定されたD-Scriptによって自動で行われる。

このD-ADDによる初期調査によって、障害の状況を正確に判断した後、障害対応にあたる担当者は原因究明の作業を開始する。原因究明は、直接的な原因を発見しただけでは終われない。次に、それがなぜ起こったのかを調べ、さらになぜその障害に気づけなかったのか、と障害の欠陥が入り混んだところまで特定できなければ、再発防止策を立てることができないからである。

D-ADDは、人間が「なぜ」という問いを繰り返しながら、複数の問いに対してそれぞれ有効なエビデンスを発見できる手助けを行う。実際の操作としては、原因究明の担当者が、「デスクトップ」と呼ぶ画面にて、自分の問い（今、何を調査しているのか）とそれに関連するエビデンス群と、

図８：原因究明の作業ステップ



そこから導き出した結果を整理できるようになっている。前頁図８は作業のステップをまとめたものである。

今回起こった放送事故では、まず「なぜ番組が途切れたのか」を調べる。ユーザがその作業を始めると、D-ADDは事故が起こった日時付近の放送スケジュールや番組に関連するログを提供する。ユーザはログを解析し「放送スケジュールと番組素材の尺が合っていなかった」という事実を発見する。これは直接的な原因に過ぎないので、次に「なぜ尺が合っていないのか」を調査する。ログを分析した結果、放送1日前に放送スケジュールが変更され3分短くなっていたことが判明する。それは紙媒体で告知され、3分短くした番組が制作されていたにも関わらず、その正しい番組素材の配置が行われていなかったという事実を発見する。そして最後に「なぜ尺が合っていないのか気づかなかったのか」という根本の原因を調べる。ここでは、CM管理の要件定義書（CMの尺の違いをチェックする仕組みは必須であるという記述）、過去の合意形成の結果の記述（番組では尺チェック機能は不要であるという合意がなされていた事実）や、放送2日前の視聴確認手順書（一度確認した後に放送情報が変更されても再確認する手順がなかった事実）などを調べる。それらが障害となる欠陥の特定である。

「テスクトップ」では、こうして調べた情報が、ノードとエッジを用いて表現される。「なぜ尺が合っていないのか」という疑問を入力して表示し、それに対してそのユーザがエビデンスとして採用したログや関連する文書が紐付き、それらの分析結果として「放送スケジュールが放送1日前に変更されたが、正しい素材を配置し忘れた」という事実を記述できるようになっている。この原因究明の結果もまた、文書グラフとしてD-ADDに格納される。

D-ADDでは、原因究明の調査を複数の作業者が個別に行えることと定義し、最終的に、各人が原因のエビデンスとして選び出したD-Caseやログや導き出した結果などをマージして比較することができるように実装した。また、各担当者がデスクトップでどのような操作をして、どのようなD-Caseや文書を閲覧し、どれを原因のエビデンスとして採用したか、ということについて、D-ADDはすべてのログを格納する。そのログもまた説明責任遂行のエビデンスとなる。

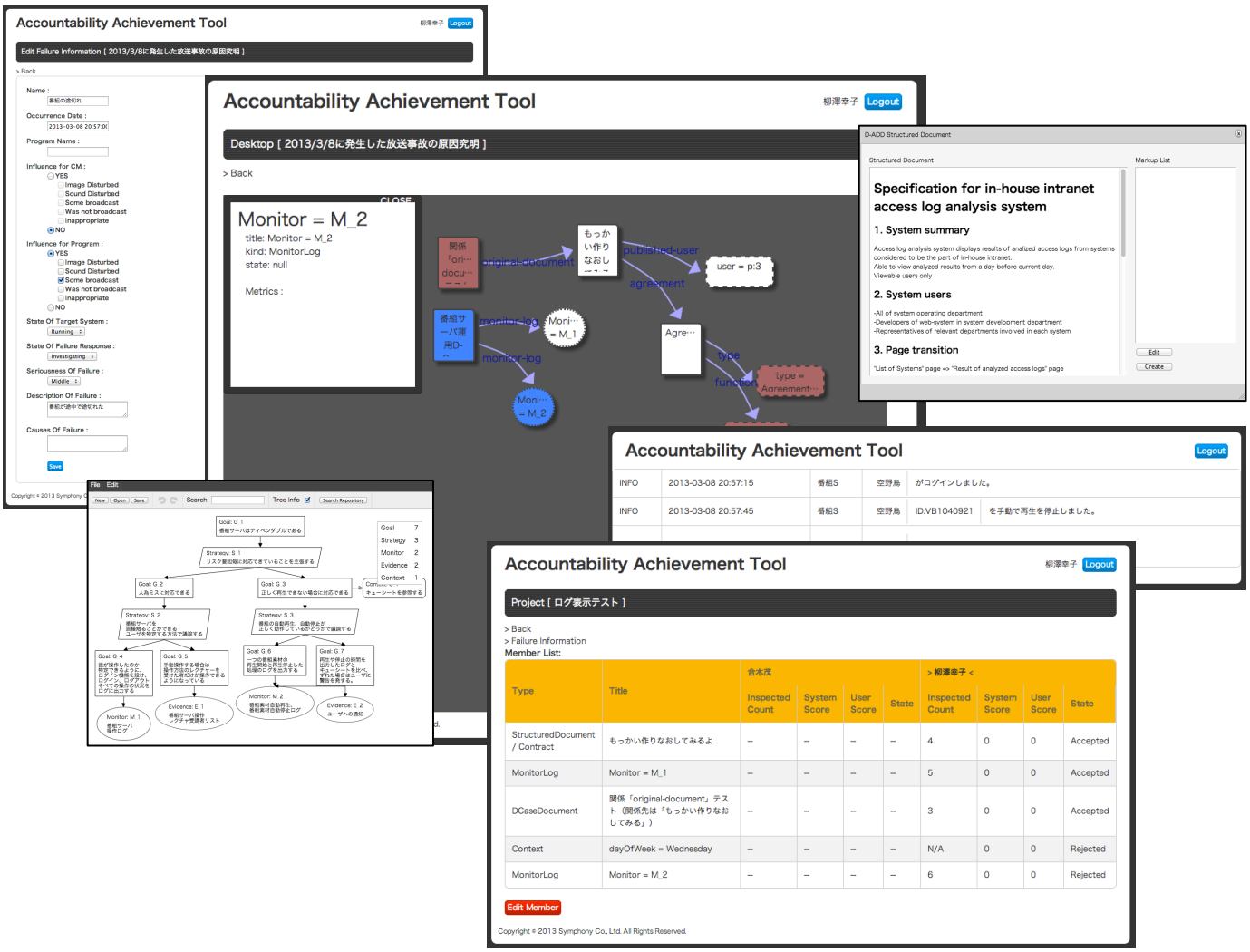
実装技術

最後に、本実装を実現した技術について述べる。説明責任遂行で重要な役割を果たす文書グラフやデスクトップは、グラフフレームワークにより提供され、文書グラフやデスクトップはグラフフレームワークによりグラフデータベースに記録される。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[5]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[6]を採用した。また、アプリケーションサーバにはPlay!フレームワーク[7]を採用した。Play!に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、モデル群、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、合意形成支援ツールおよび説明責任遂行支援

ツールはPlay!アプリケーションとして実装した。

図9には当該プロトタイプの実行イメージを示した。文書グラフを用いた、「デスクトップ」を表示している。各ノードにカーソルを合わせるとノードの情報がポップアップする。それがD-Caseであれば、「Inspect」することによってD-CaseWeaverが起動しD-Caseを表示する。

図９：実行イメージ



Type	Title	倉木茂 Inspected Count	System Score	User Score	State	> 柳澤幸子 < Inspected Count	System Score	User Score	State
StructuredDocument / Contract	もっかい作りなおしてみるよ	–	–	–	–	4	0	0	Accepted
MonitorLog	Monitor = M_1	–	–	–	–	5	0	0	Accepted
DCaseDocument	関係「original-document」テスト（関係先は「もっかい作りなおしてみる」）	–	–	–	–	3	0	0	Accepted
Context	dayOfWeek = Wednesday	–	–	–	–	N/A	0	0	Rejected
MonitorLog	Monitor = M_2	–	–	–	–	6	0	0	Rejected

実ビジネスにおけるD-ADDの利用

D-ADDの適用領域は多岐に亘る。例えばアプリケーションサーバ領域ではDEOSプロセスが規定しているようなステークホルダの合意事項を対象システムの通常運用状態にまで連携させた実装は存在しない。世界初の試みである。

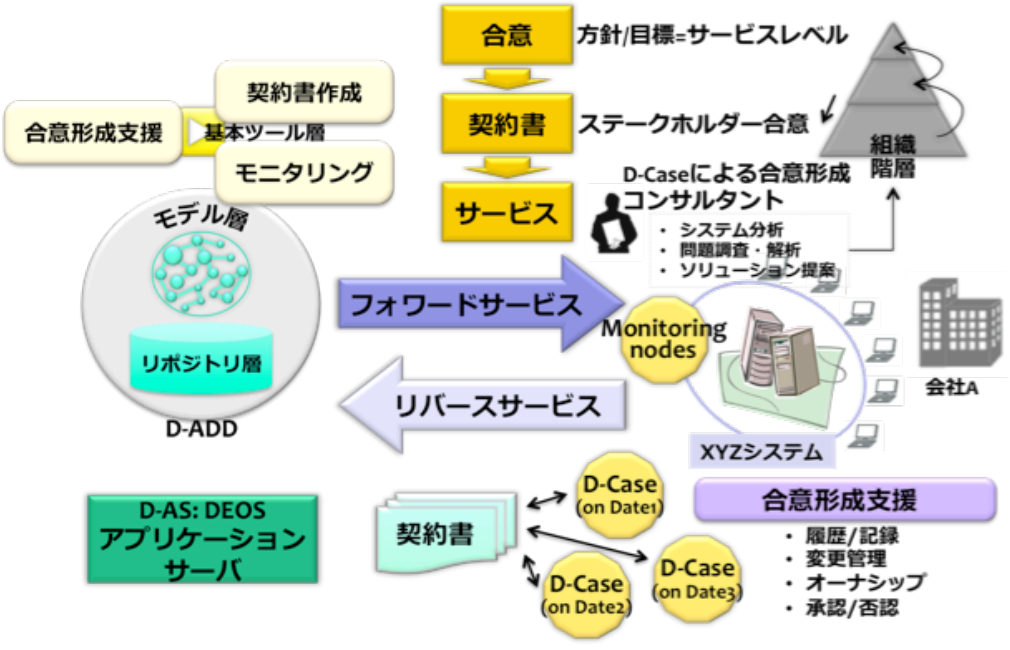
D-ADDと関係ビジネスを図１０ではフォワードモデルとリバースモデルから整理する。前者は、新規開発案件への適用であり、DEOSプロセスの全てを対象システムに作り込む事が可能である。D-ADD機能をSaaSとして提供するビジネスや、D-Caseによる合意形成支援ビジネス、基本ツール群を中心にしたアプリケーションビジネス、等が考えられる。

TOGAF[8]ではエンタープライズ連続体を支援するためのアーキテクチャリポジトリを規定しており、アーキテクチャ開発手法によるプロセスの成果物を格納する。D-ADDはこのアーキテクチャリポジトリの実装として最適である。一方、後者は、DEOSプロセスの既存のシステムへの適用であり、その際の支援サービスや、オープンシステムディペンダビリティを担保する際の既存システムの弱点を診断するサービス、既存システムに関するD-Case記述開発ビジネス、等が考えられる。

また、ITIL[9]はITサービスマネジメントのベストプラクティスをまとめたフレームワークである。ここには、単なるデータを情報（Information）、知識（Knowledge）、そして知恵（Wisdom）へと昇華させる戦略をITILサービスライフサイクルのフレームワークで整理している。D-ADDモデル層にITILフレームワークを追加することで、ITILにおけるベストプラクティスをDEOSプロセスの運用で活用することができる。

アプリケーションサーバ領域はJava技術を中心に、商用ソフトウェアとオープンソース（OSS）がしのぎを削っている。しかし、DEOSプロジェクトにおいて提案されているオープンシステムディペンダビリティを担保する為の技術体系に則

図１０：ビジネスモデル



ったアプリケーションサーバは存在しない。そこには多くの潜在的ビジネス機会が存在していると考える。D-ADDによりDEOSプロセスを適切に運用することで社会インフラとしてますます重要性を高めるITシステムのオープンシステムディペンダビリティを違った次元に高めることができる。

D-ADDとソフトウェア開発プロセス革新へのアプローチ

大規模システム開発にD-ADDを応用した場合に、ソフト開発プロセスがどのような変貌を遂げるか、幾つかの想定を元に説明を試みる。

DEOSプロセスを展開した大規模ITシステムの開発において、D-Caseによるアシュアランスケースの記述、ならびにD-ADDによる合意形成支援と記述データベースは、従来方法による設計品質や製造品質を向上させることにある。金融システムや医療システムに代表される高信頼性を求められる大規模ITシステムは、特に顧客要件や性能要件について、多くの関係者により十分な時間をさいて議論を行うため、一般的には開発プロセスとしてウォーターフォール型が好まれる傾向がある。

しかし、業務要件が非常に複雑で発注者側でも将来に柔軟性を持たせたい場合（リリース時期が数年先であれば、市場や業務の変更までを視野に入れて）には、反復開発型を試みることもある。反復開発時の初期には、アジャイル開発が使われることもある。前者の場合は、膨大なドキュメントが生まれることであり、その発生期間も長く、関係者の数も多いために、ドキュメントで示されたシステムの特性が、からずしも一様な仕様に確定していかない傾向がある。一方で後者は、期間を短くした反復性にて、かつ関係者も限られるために、仕様が明確になる確率が高くなる。もともと仕様の抜けや漏れが発生することを前提として反復開発により完成度を高める方法論である故、開発中のコミュニケーションの記録がどこまで精度が高いかで開発プロセスそのものが問われる。近年、新しい業務要件に対するITシステム開発では、運用と開発との間のコミュニケーションを密に行い、システムをつくりあげていくDevOps（デブオプス）と呼ばれる方法論も注目されている。

いずれの方法論を用いても、ソフトウエア開発プロセスにおいては、対象システムの設計品質や製造品質が向上できるかが目標であり、DEOSプロセス、D-Case、D-ADDが、それらの向上目標に対して、寄与できるかどうか、または、いかに高信頼性を担保できるかである。

要件定義、設計ドキュメントと合意形成

システムの基本骨格、性能を決めるのは要件定義と設計ドキュメントの二つの資料である。これらは、D-ADDの格納情報である。そして、この二つの資料のさらに上流には、原要求という発注者側トップの要求があるわけであるから、開発システムを対象としたD-Caseは、これらの格納情報から検討され開発システムの信頼性を向上させる指針として合意の対象となる。つまり、D-ADDは、原要求、要件定義、設計ドキュメントから検討されるD-Caseを相互に格納しつつ、複雑に絡み合ったそれらの資料群を巧妙に合意形成支援と記述を積み重ねることになる。D-ADDのユーザとは、発注者側と受注者側のプロジェクトマネージャクラスを筆頭に、全開発関係者がなり得るが、個々の合意形成支援とその記述においては、その合意部分の関係者が適切に選ばれているかといった、組織上の課題があり、それらの検証が常に適切であるという事を保証するには、D-ADDには、実行組織上の認証機能が必要であることが理解できる。

D-ADDを用いた対象システムに関する議論は、その議論がどのようなリスクを孕んでいるか、齟齬なく収束に向かっているのか、システム

の部位に対して、複雑に絡み合う仕様の記述になっていないかなどを知ることによって、合理的で妥当な合意に繋がっていくと判断できることが望ましい。少なくともこれらの「希望」は、実際の大規模システム開発では、適切な記録として開発業務が進行することは希であり、各社アプローチや管理基準があるかもしれないが、系統だった手法として公開された管理技術はない。

D-ADDは、D-Case、各種ドキュメント類、それらを対象とした合意記述をXMLリポジトリとして設計し、製造されるプログラムの構成管理と対になる管理技術を検討している。

合意形成と仕様の変化

では、D-Case、D-ADDによる合意形成と記述により開発が実行され、複雑な構成管理が提供されると開発プロセスはどのような変化を受けるだろうか？ DEOSプロセスの重要な視点に、環境的な変更が仕様に与える影響を、ステークホルダの合意形成を実行し対応を図るという考えがある。仕様変更に伴うDEOSプロセスの実行時は、システムエンジニアリングの現場において、問題解決を図る場合の現実的な適応力を高める必要がある。特に反復開発やアジャイル開発では、仕様の変更は発展的解決の糸口になるケースも見られる。昨今のソフトウエア開発の現場では、エンジニア同士が、twitterやSNSを使って仕様議論や実装議論をすることは希ではなく、むしろ積極的に用いてコミュニケーションを活発にする動きがある。こうした、ツールを用いたコミュニケーション能力の拡大は、ここ数年に起こり広まった事象であり、プロジェクト管理の面からも有形無形のメリットをもたらしている。当然ながらプロジェクトの実行状況に関する情報がリアルタイムに観測出来るようになる点であり、特に人材管理の側面で効果が高い。ある企業の大規模ITシステム開発のための専用コミュニケーションツールを開発し運用した経験では、年間を通じた開発記録とその議論の記録は、開発者同士の間で非常に多くの実装アイディアが記載されており、貴重な知識の宝庫となり得た。そして、今回のD-ADDプロトタイプ開発では、約10数名のエンジニアが実装技術アイディアの提案、ならびに実装方法における議論で、D-ADDによる合意形成支援が、プログラム開発における創造的思考と発想支援環境に近いことも解ってきた。

D-ADDの合意形成支援と記述様式は、仕様の変化に対して、意見の調整や整合を実現し、DEOSプロセスの実行と共に、高信頼性開発手法を提供することにある。高信頼性開発手法が、創造的思考と発想支援環境であるかどうかは、議論の余地があるが、ソフトウエア工学からみたDEOSプロセスの応用については、D-ADDの研究開発プロセスそのものを合意形成支援と記述の対象として進めることで検証することにしている。

革新へのアプローチ

D-ADDは、対象となっているシステムのアシュアランスケースを記述したD-Caseを格納し、要件定義、設計情報が更新される度に、関係者の合意を促し、齟齬が生まれ放置されることを防ぐ役割である。将来的にD-ADDが達成すべきで

はないかと考える機能として、原要求から、要件定義、設計と進行する開発業務において、受注者と発注者側の相互理解が深まっていくことが重要との観点から、D-Caseを中心とした各種ドキュメントは、適切な論理構成をもって構築されることが望ましいと考える。

これまでの大規模ITシステムの開発プロセスでは、分割開発される故プロセスの部分部分では関連する細部の論理性チェックと構成管理が行われてきた。しかしながら、D-ADDにより、アシュアランスケースに代表される業務レベルでの達成目標を中心に、開発におけるすべての資料群の構成管理と合意形成がなされれば、システム全体の信頼性は大きく向上されるであろう。そしてそれらの合意記述がソフトウエア開発時の知識の構造になっていくものであると考えている。つまり、D-ADDは、開発関係者全員によって、合理的で妥当な合意に繋がっていくと判断できる環境がもたらされることで、従来の開発プロセスからの脱却をもたらし、オープンシステムディペンダビリティを担保する開発プロセスがスタートするのである。

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4月からであり、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も期待されている。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に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。

2012年度第一四半期では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。第二四半期では上記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を利用している。第三四半期ではD-ADDと主たる格納情報であるD-Caseとの接続を軸とした。領域の研究成果であるD-CaseWeaver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。第四四半期では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。

2013年度は引き続き営業放送システムをシナリオとして用いてD-ADDを整理し、D-Scriptとの接続など、DEOSプロセスを周回させるための機能を完成させる予定である。

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] D-Case入門, ISBN: 978-4-8629-3079-8, 株式会社ダイテックホールディング, 2012.

[3] Stephen E. Toulmin, “The Uses of Argument; Updated Version,” ISBN: 978-0-5215-3483-3, Cambridge University Press, July 7, 2003.

[4] Object Management Group, “Structured Assurance Case Metamodel (SACM), FTF - Convenience Document Beta 2,” OMG Document Number: ptc/2010-06-06, URL: http://www.omg.org/spec/SACM, June 2012.

[5] http://tinkerpop.com

[6] http://neo4j.org

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

[8] The Open Group, “TOGAF® Version 9.1,” ISBN: 978-9-0875-3679-4, 2011.

[9] The IT Service Management Forum, “ITIL® Foundation Handbook,” ISBN: 978-0-1133-1349-5, The Stationary Office, 2012. (available on online: http://www.tsoshop.co.uk)

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。

