Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

ディペンダブルシステムのための木構造を用いた合意形成データベースの提案と実装

大城信康[†1]　河野真治[†1]
玉城将士[†1]　永山辰巳[†2]

ディペンダブルシステムの構築を目指す DEOS プロジェクトでは対象システムの要求を維持するデータベースである D-ADD が重要である．D-ADD はシステムに障害が発生した際に要求・説明責任を果たすため機能の 1 つとして合意形成支援がある．この合意形成支援は合意状況の時間的な関係をみることができると考えられている．本論文では, 要求・説明責任を果たすために木構造による合意形成支援を構成する方法を提案する．また, Web アプリケーションとして実装を行い, 実際に例題を記述して, その有効性を確認する．

Nobuyasu OSHIRO,[†1] Shinji KONO,[†1] Shoshi TAMAKI[†2]
and Tatsumi NAGAYAMA [†2]

D-ADD is a Knowledge Database in the DEOS project, which aimed at building a Dependable Systems. It is the database which maintains requests and responsibilities of the target system. D-ADD has a consensus building tool to fulfill the responsibilities when the system fails. This consensus building tool shows the temporal relationship of agreement statuses. In this paper, we propose a knowledge structure of consensus building tool. It is a tree structure of requirements and responses. We implement a Web application using GraphDB and show some examples.

1. 研究の目的と背景

IT システムが巨大化していくにつれ, 障害発生事例が社会に与える影響もより大きな物となる．それに伴い, IT システムにおけるディペンダビリティへの注目が増している．

そこで, DEOS プロジェクトは IT システムにおける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している．DEOS プロジェクトは 2006 年に独立行政法人科学技術機構 (JST) は CREST プログラムの 1 つとして始まったプロジェクトである.[1)]．DEOS プロジェクトは, 変化し続ける目的や環境の中でシステムを適切に対応させ, 継続的にユーザが求めるサービスを提供することができるシステムの構築法を開発することを目標としている.[2)] DEOS プロジェクトではそれらの技術体系を「オープンシステムディペンダビリティ」として定義し, それを DEOS プロセスとしてまとめた (**図 1**)．DEOS プロセスには変化対応サイクルと障害対応サイクルの 2 つのサイクルがある．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ためのプロセスである．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である．DEOS プロセスは企画や設計という上流からシステム運用までの対象システムのライフサイクル全体に係わる．対象システムの運用は, 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としたあらゆる議論に関する合意をベースに行われる．

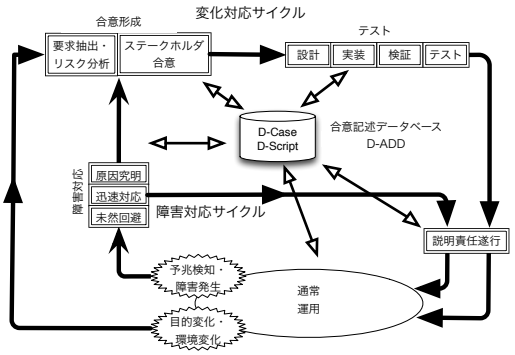

図 1　DEOS プロセス

†1 琉球大学理工学研究科情報工学専攻
Interdisciplinary Information Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
†2 株式会社 Symphony

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

DEOS プロセスにおいて, 全てのデータを保持する D-ADD(DEOS Agreement Description Database) と呼ばれるデータベースがある. D-ADD で扱うデータにはステークホルダ間で合意された契約書等も含まれる. そのため D-ADD に基本ツールとして合意形成支援ツールが必要である.

本研究では D-ADD で行われる合意形成のモデルを提案しツールのプロトタイプの実装を行う. また, 実際に例題を入力してみて D-ADD における合意形成支援のあり方についての追求を行う.

2. D-ADD

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.[1] D-ADD の概略を**図 2** に示す. 上位層は D-ADD における基本ツールである. ここでは DEOS プロセスとのやり取りを支援する. 2 つめの層は D-ADD が扱うデータのモデルを指す. 今回提案するモデル含め, 上位層で扱うデータはここで定義される. 下位層は D-ADD で扱うデータベースである. D-ADD は様々なデータを扱うため数種類のデータベースを利用する予定である.

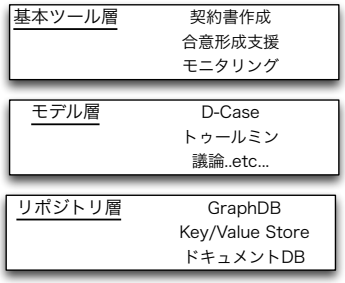

図 2 D-ADD 概略

図 2 において D-Case は DEOS プロセスにおける共通言語として設計されているものである[3]. トゥールミンモデル[4] は 1 つの主張のモデルである.

2.1 説明責任と合意形成

D-ADD はシステムに障害が発生した際, 説明責任を果たさなければならない. 説明責任とはなぜその障害が発生したのか, 次からはその障害を起こさせない, もしくはしっかりと対応できることを示すことである. そして説明責任を果たすためにはまず, なぜそのようなシステムになったのかということを説明できなければならないと考えられた. そのためには D-ADD に入るデータはプロジェクトに関わる人, ステークホルダの合意を得たデータにすべきである. そこで D-ADD 自身に合意形成を支援する機能が必要となってくる. D-ADD はその合意形成支援を Web アプリケーションにより行う.

3. 提案するモデル

ある事柄に対して合意を取る場合, 議論が行われる. そこで, 合意形成支援を行うため, 議論のモデルから考えた.

3.1 モデルの概要

提案する議論のモデルは, 合意形成を**主張・関係・ユーザ**の要素から構成される木と考える. 合意を取りたい**主張**があり, その内容を深めて議論していくことで**ユーザ**に合意するよう説得していく. 議論を深めるには, **主張**から更に踏み込んだ内容の**主張**が複数派生させていくことになる. このとき, **主張**に対してどのように踏み込んだかという**関係**も発生する. よって, **主張**から複数の**関係**と**主張**が派生し, その**主張**からさらに複数の**関係**と**主張**が派生することにより, 木構造が構成される. これは, 木構造には閉路が含まれないため, 循環論法を生じさせないという狙いがある.

ユーザ・主張・関係は以下の用に定義される.

- ユーザ
  - 合意形成の参加者をユーザという.
- 主張
  - ユーザが作成した合意をとりたい, 議論すべき内容を含むものを主張という.
- 関係
  - ユーザと主張, もしくは互いにことなる主張と主張がどのように踏み込んだかを示すのを**関係**という.

このモデルにおいて**ユーザ・主張**はノード, **関係**はエッジとして扱われる.

3.2 合意状況の計算

このモデルにおいて主張を合意させるには, 合意要求を出している全員から合意を受けなければならない. しかし, その主張に子となる主張 (**図 6** での主張 2,3) がある場合, 関係次第では子となる主張の合意状態も影響を与えることにした. 自身の主張の合意を通すための各関係は以下のとおりとなる.

- 反論
  - 反論となる主張が合意されると親の主張は合意できない. 反論には別の反論を用意して相手を納得させることで合意を進める.
- 質問
  - 質問となる主張は別の主張により答えが得られたら合意をする. 質問の関係で繋げられた主張を合意させなければ親の主張の合意は通らない.
- 提案
  - 提案となる主張はどの状態であろうと親の合意状況に影響は与えない.

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

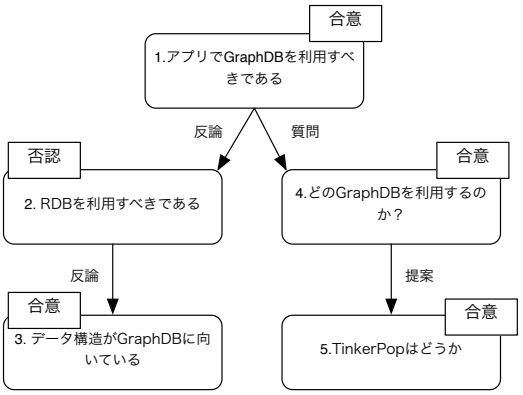

図 3 主張 1 の合意が取られた状態

上記のように 3 種類の関係にそって主張の合意状態は計算される．実際にどのように主張が立てられて合意がなされていくのか簡単なシナリオを記述して示したものが**図 3** となる．シナリオの議論の目的はアプリで使用するデータベースを決めることである．

図 3 の説明を行う．四角が主張を, 矢印が関係をそれぞれ表す．まず最初に主張 1「アプリで GraphDB を利用すべきである」が立てられる．次にその主張に対して反論となる主張 2「RDB を利用すべきである」が立てられる．この時, 最初に立てた主張は自身の合意を取るためには反論となる主張を否認させなければならない．そこで, 反論に対して反論を用意し, 主張 3「データ構造が GraphDB に向いている」を立て主張 2 を否認する．主張 2 の作成を行った人は主張 3 で納得したため主張 3 に合意を行う．それにより主張 2 は否認されることとなる．次に質問となる主張 4「どの GraphDB を利用すべきか？」が立てられる．これに対しては提案となる主張 5「TinkerPop はどうか」を立てることで応える．質問者はその答えに納得し主張 5 に対して合意を行い, 他に反論や質問も無いため主張 1 に対しても合意する．これで主張 1 に対して反論の関係にある主張は否認にし, 質問となる主張の合意を取ることができたため, 主張 1 の合意はなされた．以上が提案するモデルによる合意を取るまでの簡単な流れである．

3.3 主張のモデル

また, それぞれ個々の主張に対してもモデルを考えた．そこで今回は, トゥールミンモデルと呼ばれるモデルを適用する．トゥールミンモデルは 1 つの主張に以下の 5 つの情報も必要であるとするモデルである．

- データ (Data)
- 根拠 (Warrant)
- 裏付け (Backing)
- 論駁 (Rebuttal)
- 限定詞 (Qualifier)

主張を後押しする客観的な資料となるデータがあり,

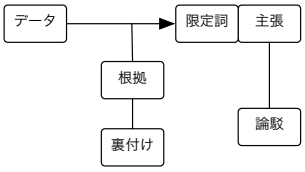

図 4 トゥールミンモデル概略図

根拠はデータがなぜ主張を後押しするのかを示す．裏付けは根拠が正しいことを示し, 論駁には主張が成り立たなくなる例外の条件が入る．限定詞を使うことで主張が完全であるかないかを表すことができる．これらの情報をもった主張がトゥールミンモデルとなる．トゥールミンモデルの概略図を示したのが**図 4** となる．

もちろん D-ADD における主張のモデルはトゥールミンモデルだけでなく, 別のモデルを用いても良い．D-ADD による合意形成で扱う主張のモデルは熟考していく必要がある．

4. 実　　装

作成するアプリケーションのユースケース図を**図 5** に示す．

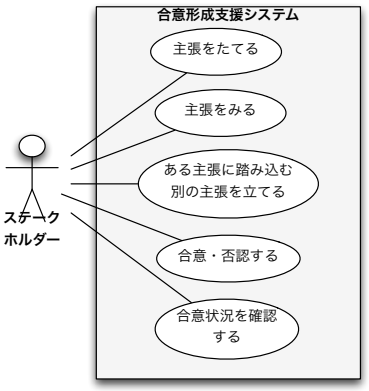

図 5 ユースケース図

まず, ユーザ (ステークホルダ) は主張を立てることができる．この時, 立てた主張は自分以外のユーザに対して合意要求を出さなければならない．ユーザは自分に対して合意要求を出している主張を見ることができ, その合意要求に対して合意・否認・保留のどれかを選択することができる．また, 既にある主張に対して関係をはる主張を立てることができる．ここでいう関係は反論・質問・提案のどれかである．

上記の機能を備えた合意形成支援 Web アプリケーションを作成する．また, 今回は合意がとられている様子がみられるよう, リアルタイムでデータが更新さ

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

れていくものを作る.

4.1 モデルの実装に使用するデータベース

提案するモデルは木構造で議論を深めていく. このことを念頭に, モデルの実装に使用するデータベースについて考えた. 木構造は閉路を持たないグラフである. グラフのデータの扱いが得意なデータベースとしては GraphDB があげられる. よって, GraphDB を使用することにした.

GraphDB はデータの情報をノードとエッジで持ち, ノードとエッジはそれぞれプロパティを持つことができる. ノード同士はエッジで繋ぐこともでき, トラバースと呼ばれる操作でノード間を渡り歩き情報を引き出すことができる. また, エッジには関係 (ラベル) が定義され, トラバースは渡り歩くエッジの関係を指定することで行える.

GraphDB は各ノード, 各エッジが自身に繋がっているエッジ, ノードの情報を保持しているため次のノードへと渡り歩くことが容易である. 仮に, RDB でグラフを表そうとすると, まずノードの情報を引き, 次に index を引いてエッジの情報をとってきて次のノードの情報をとるという手間がかかる. GraphDB を用いることでその手間のなくすことを狙いとする.

4.2 GraphDB によるモデルの表現

GraphDB を用いて提案したモデルを表現する. 提案するモデルの**ユーザ**と**主張**がノードで, **関係**がエッジにあたる. 各主張とユーザとの関係を示したものが図 6 となる. 四角がノードを, 矢印がエッジをそれぞれ表している. 主張 2,3 からユーザへのエッジは省略しているが, 各主張ノードからはそれぞれ作者と合意要求の関係となるエッジがユーザノードへと繋げられる.

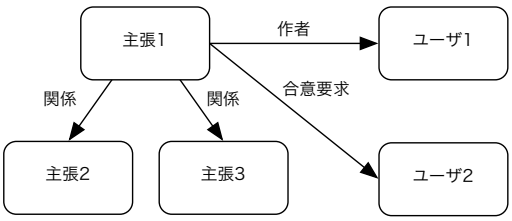

図 6 主張ノードとユーザノードの繋がり

今回使用した GraphDB は TinkerPop になる. TinkerPop で実際にノードとエッジを作成するコードの例を示す.

```
1 Graph g = new TinkerGraph();
2 Vertex claim = g.addVertex(ID);
3 Edge relation = g.addEdge(ID,From,To,Label);
4 claim.setProperty(PropertyName,PropertyValue);
```

1 行目はグラフの作成を行っている. 引数にパスをいれた場合, グラフのデータが指定されたパスに保存される. 2 行目ではノードの作成を行なっている. 設定したい ID を引数にいれる. 3 行目では関係 (エッジ) の作成を行なっている. 設定したい ID を引数にいれる. 引数の From と To はそれぞれノードであり, From から To 向きへエッジが作成される. Label には関係の名前がはいる. 4 行目ではプロパティの作成を行っている. ノード・エッジともに同様のメソッドでプロパティの作成ができる.

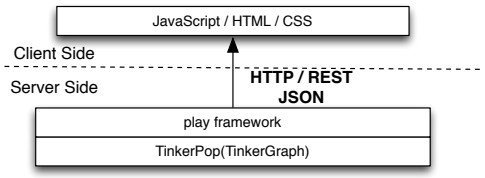

図 7 Web アプリ実装概要

このようにノードとエッジ, それとプロパティの作成を行い実装していく.

4.3 実装内容

サーバサイドの実装は Java を使用した. GraphDB は TinkerPop 使用し, play framework により REST API を提供する. クライアントサイドは JavaScript/HTML/CSS を用いる. クライントサイドはサーバサイドが提供する REST API を非同期に叩いて主に JSON データを受け取り, ブラウザへと表示をおこなう. 実装の概略を**図 7** に示す.

4.4 各ノード, エッジのプロパティ

ユーザノードは特にプロパティを持たないが, 主張ノードは以下のプロパティを持つ.

- status： 主張の合意状態を表す. passed, failed, unknown の状態がある
- title： 主張のタイトル
- content： 主張の内容
- toulmin： トゥールミンモデルにおける主張以外の 5 つの要素が入る
- timestamp： 作成された時間を示すタイムスタンプ

主張のそれぞれの合意状態は, passed は合意されている, failed は否認された, unknown は passed でも failed でも無いことを示す.

主張同士を繋ぐエッジはプロパティを持たないが, ユーザへと伸びる合意要求のエッジだけは以下のプロパティを持つ.

- status： ユーザの主張に対する合意状態を表す. agreed, denied, pend, unknown の状態がある.

agreed は合意, denied は否認, pend は保留, unknown は最初の状態をそれぞれ示す.

4.5 合意状態の計算

基本, 各主張は自身に繋がっている合意要求の関係にあるエッジの status が全て agreed か failed となる

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

ことで合意か否認にされたとみなされる．しかし，今回のモデルの特徴の 1 つとしてエッジの関係により合意状態の計算が変わってくるものがある．これにより，ユーザへと伸びる合意要求のエッジの他に，子となる主張の合意状態とエッジの関係も見る必要がでてくる．主張が合意状態となるために以下の 2 つもチェックする必要がある．

- 反論の関係で繋がっている子の主張は全て否認の状態になってなければならない．
- 質問の関係で繋がっている子の主張は全て合意の状態になってなければならない．

提案の関係は，合意状態に影響を及ぼさないため考慮する必要はない．

4.6 時系列ごとにみられる議論と合意状態

合意形成を行う最中に，一度否認された主張が合意されたり，合意されていた主張が別の主張の登場で否認されるといったことも起きることが考えられる．しかし，最新の合意状態をみるだけではその合意の流れを後から見ることができない．だが，これらの情報を得られるとすると，説明責任を果たすことの手助けになるのではないか考えられる．そこで，合意状態の時系列ごとの変化を見られるようにした．具体的にはタイムスタンプをノードのプロパティとして追加し，更に主張同士の繋がりでできる木構造のデータのコピーを行うことにした．コピーされた木は最新の木に対して prev の関係となるエッジで繋げる．これにより，prev エッジを辿ることで前の合意状態を見ることができるようになった (**図 8**)．

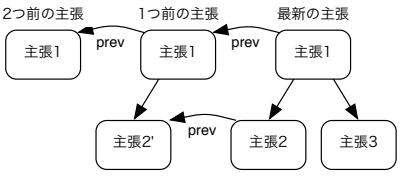

図 8 時系列毎の合意状態の保存

5. 合意形成支援 Web アプリケーション

合意形成支援 Web アプリケーションの使い方を簡単に説明する．まず初めはログイン画面よりログインを行う．ログインが行われるとホーム画面へと移る．この時，ユーザが参加している議論があればその議論のタイトルが出力される (**図 9**)．

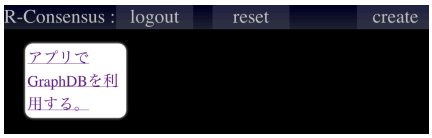

図 9 ホーム画面

図 10 主張の作成，合意要求相手の選択画面

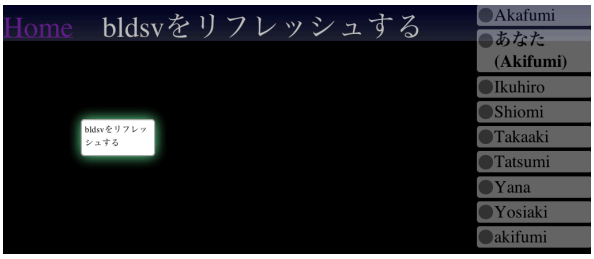

図 11 合意形成画面

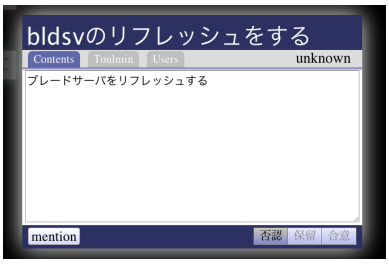

図 12 主張の内容表示と合意用状態の選択画面

新しく議論を作成したいときはホーム画面 (図 9) 右上の create より主張の作成を行う．主張のタイトルや名前，必要ならばトゥールミンモデルの情報をいれる．最後に合意要求を出す相手を選び save ボタンを押す．**図 10** は bldsv(ブレードサーバ) をリフレッシュするという主張を立て Akifumi というユーザに合意要求を送る操作である．これで議論の初めの主張が作成された．

5.1 議論の画面

次に作成された主張をクリックすることで合意形成を行う画面へと移る (**図 11**)．

図 11 の画面左側にある四角は主張 (ノード) を表す．主張をクリックすると合意状態や内容といった詳しい情報がみることができる．また，ここで主張に対する合意・否認・保留の選択も行える (**図 12**)．

また，図 12 の画面より mention を選ぶことで踏み込んだ主張の作成が行える．**図 13** は反論の主張を作成する画面である．

主張が作成されるとノードが表れ関係の名前がかかれたエッジで主張同士が繋がる (**図 14**)．

この様に関係のある主張でつながれていくことで木

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

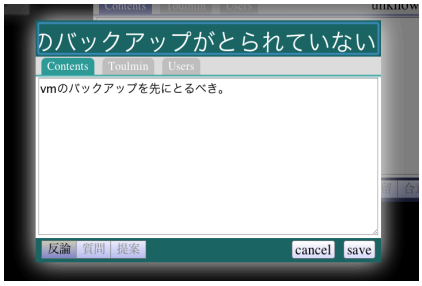

図 13 主張に踏み込んだ主張の作成画面 (反論の作成)

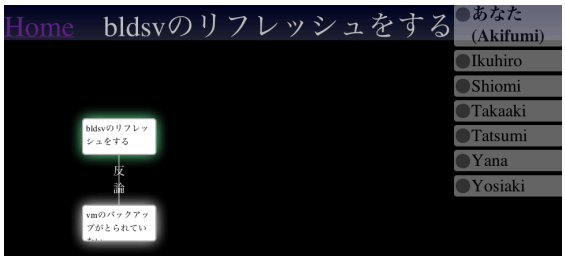

図 14 反論の関係にある主張が出現する

構造の議論が作られていき合意形成支援を行う Web アプリケーションとなっている.

6. 合意形成支援 Web アプリケーションの考察

作成した Web アプリケーションに実際にデータを入力する. また, 同じデータをプロジェクト管理ソフトである redmine に入力する場合も想定する. redmine は, チケットと呼ばれる単位で作業の記録をとる. チケットにはさらに子チケットや関連するチケットといったものを登録ができる. コメント機能もあり作業の報告や相談をそこで行うこともできる. Web アプリケーションと redmine での表示についての考察を行う.

6.1 データの入力

今回入力するのはブレードサーバ (以下 bldsv) のリフレッシュ作業に関する作業ログである. 実際に行う予定だった作業は以下の通りである.

(1) bldsv 上で稼働している VM のバックアップをとり停止させる
(2) bldsv のリフレッシュを行う
(3) VM をバックアップから戻し再稼働させる

当初はこの 3 つの作業だけを予定していた. しかし実際に行った作業は以下の内容となる.

(1) bldsv 上で稼働している VM のバックアップをとり停止させる
(2) bldsv のリフレッシュを行う
(3) VM をバックアップから戻すことに失敗する
(4) VM を戻せない原因の追求を行う
(5) 原因が bldsv 上の VM を落としたため DNS サーバも落ち, アドレスが引けないことだと分かる
(6) DNS サーバを使用しないですむ方法を考える
(7) /etc/hosts に直接書き込むことで DNS 無しでもドメインを解決できかもしれないと案があがる
(8) /etc/hosts に必要なサーバの情報を書き込む
(9) VM が bldsv 上に戻り無事起動する
(10) 次回からの解決方法を探る
(11) 学外にセカンダリとなる DNS サーバを置くことで解決するのではないかと案があがる
(12) 学外にセカンダリ DNS サーバをおく

ブレードサーバをリフレッシュするという話が学外にセカンダリ DNS サーバを置くという話にまで広がっている. redmine でチケットを作成しこの作業を行なわれた場合を考える. まず, 「サーバをリフレッシュする」というチケットがあるだろう. しかし, そのチケットの中で子もしくは関連チケットとして「学外にセカンダリ DNS サーバを置く」というチケットがでてくる. これは初見の人にとっては繋がりがよく分からずに疑問が浮かぶことだろう. なぜそのようなチケットがあるのかを知るためには子チケットや関連するチケットのコメントを読む必要がでてくるかもしれない. だが, 今回私達が作成した Web アプリケーションでは redmine のチケットは 1 つの主張ノードにあたる. そのため, 各関係がエッジで繋がっているため関連の確認が行いやすくなっている. 実際にデータを入力して出来た主張の木が**図 15** となる.

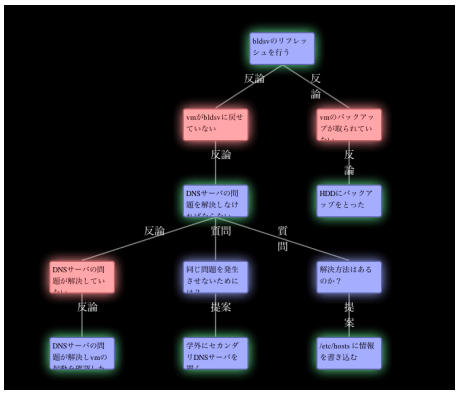

図 15 合意形成を表す木

各主張の合意状態は主張をクリックすることでみられるが, 色でも分かるようにした. 青色が合意, 赤色が否認を表す.

7. 時系列に合意状態を見ることの有用性

最後に, 時系列に合意状態をみることによる有用性について考察を行う.

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

今回, 時系列的な合意状態を保存した主な理由に次の場面を想定したからである.

(1) 最初に立てられた主張の合意がとられる.
(2) 一度合意はされたが否認に直され, 別の主張が合意される.
(3) やはり最初の主張の方で合意を行いたいと思い最初の合意状態に戻される.

これら一連の流れを**図 16**, **図 17**, **図 18** に示す.

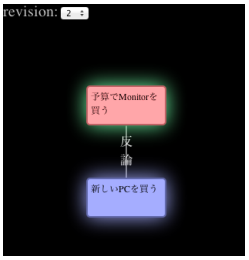

図 16 時系列的な合意状態の確認その 1

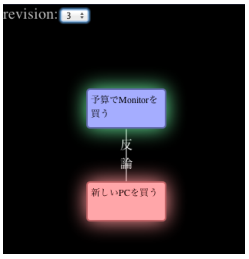

図 17 時系列的な合意状態の確認その 2

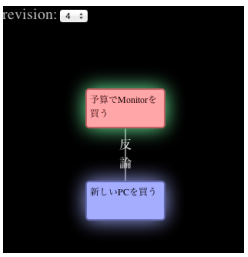

図 18 時系列的な合意状態の確認その 3

最終的な合意は図 18 となる. ここで問題になるのが, 図 17 の状態である. もし, 時系列的な表示ができなければ, 図 18 の状態しか確認することができない. しかし実際にこの議論は, 一度は別の主張が合意されたという流れを持つ. 時系列的な表示を行うことで, 合意がなされるまでの新たな情報を得ることができる. これにより, なぜこの主張が合意されたのか, なぜ別の主張が合意されなかったのかといった説明要求に応えるための情報を増やすことができると期待できる.

8. ま と め

議論のモデルを考え Web アプリケーションとして合意形成支援ツールの実装を行い D-ADD における合意形成支援についての追求を行った. 木構造による議論のモデルの提案と GraphDB を用いた実装を行った. 作成した Web アプリケーションに実際にデータを入力し, redmine に入力する場合と比較した. redmine で入力した場合に比べ, 各主張同士の関係が見やすいことを確認した. 時系列的に流れを見ることで, 合意までの流れがより分かりやすくなり, 合意全体の理解を支援できる可能性もみることができた.

8.1 今後の課題

本実装では主張同士の関係が反論・質問・提案だけであったが, 更に増やすことが可能である. 関係を増やす際は, 反論・質問・提案が与える合意状態への影響を元に作成する. 例えば質問の関係を元にした回答という関係をつくれば, 合意されなければならない回答というエッジを作ることができる. どのような名前のエッジを用意したらユーザがより分かりやすいかは考えるべきである.

今回実装した Web アプリケーションでは, 大量のデータの入力は想定していない. そのため, 大量に主張が作成され議論の木が膨大になった際の対処方法を考えなければならない. また, 内容が同じノード同士の検出も行なっていない. その点はユーザに任せている. しかし, D-ADD はそれら複数のデータ間で整合性を取らなければならない. また, 一度合意形成がなされたデータでも変更される可能性がある. この時, 変更された際にどこまで影響が及ぶのかといったことも検出するのが理想である. 複数のデータ間での整合性と変更したことで影響を与える範囲の検出方法も考えていかなければならない.

9. 謝　　辞

本研究は, JST/CREST 研究領域「実用化を目指した組み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」D-ADD 研究チームの助成を受けて行われた. 適切な助言と支援をいただきました D-ADD 研究チームの永山辰巳氏, 横手靖彦氏に感謝します.

参 考 文 献

1) Tatsumi Nagayama, Yasuhiko Yokote: 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 (2012).
2) DEOS 研究開発センター：DEOS プロジェクト「実用化を目指した組み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」(2012).
3) 株式会社ダイテックホールディング：D-Case 入門, ISBN: 978-4-8629-3079-8 (2012).
4) Stephen E. Toulmin: The Uses of Argument,

Vol.2013-ARC-205 No.10
Vol.2013-OS-125 No.10
2013/4/26

ISBN: 978-0-5215-3483-3, Cambridge University Press, July 7 (2003).
5) Renzo Angles, Claudio Gutierrez: Survey of Graph Database Models, Technical Report Number TR/DCC-2005-10 Computer Science Department Universidad de Chile.
6) Clare Churcher: Beginning Database Design: From Novice to Professional, Apress (2007/1/15).
7) : TinkerPop, http://www.tinkerpop.com/.
8) : play framework, http://www.playframework.com/.
9) Phan Minh Dung: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games (1993).