部分理解の問題を解決するソフトウェア協調開発に向けて
―議論の内容を構造化して記録するためのモデルの提案―

古宮　誠一

国立情報学研究所 先端ソフトウェア工学・国際研究センター

〒101-8430 東京都千代田区一ツ橋 2-1-2

E-mail:　seiichi_komiya@nii.ac.jp

概要　この論文は，派生開発における部分理解の問題を解決するために，要求定義工程では，開発が本格化する前にプロジェクトの主要メンバーが集まって，顧客との折衝で得られた要求仕様の内容の理解と妥当性確認のための議論を行うことを提案している。そして，この議論で得られた要求仕様に対する理解(解釈)をプロジェクトのメンバー全員で共有することの重要性を強調している。その上で，議論に参加しなかった人でも議事録を読めば容易に理解できるように，議論の内容を構造化して，即時記録するためのモデルを提案している。設計工程では，設計案を検討する際に，どの案にするかの意思決定とその根拠に関する議論をしながら，ソフトウェア開発することを提案している。そして，このときの議論の内容を構造化して，即時記録するためのモデルを提案している。

キーワード　派生開発，部分理解の解決，ソフトウェア協調開発，ソフトウェア設計上の意思決定とその根拠

Toward Software Collaborative Development to Resolve the Problem that Derivative Development Is Conducted from Tunnel Vision:
A Proposal of a Model to Structuralize and Record the Contents of the Discussion

Seiichi Komiya

National Institute of Informatics, Center for Global Research in Advanced Software Science and Engineering,

2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan

E-mail: seiichi_komiya@nii.ac.jp

Abstract: Derivative development has a problem that software development is conducted from tunnel vision. In order to solve the problem, the author proposes that in requirements definition process key members in a software project should get together and conduct the discussion for understanding and validity check of software requirements before software development gets into full swing. And, he advocates strongly that it is important that all of the project members share in understanding (interpretation) of software requirements obtained by the above-mentioned discussion. Furthermore, he proposes a model to structuralize and record the contents of the discussion in real time so that even the project members that had missed the discussion can easily understand the interpretation by reading the minutes. In software design process, he proposes that software design should be conducted based on the discussion of design rational, and proposes a model to structuralize and record the contents of the discussion in real time.

Keyword: derivative development, resolution of software tunnel-visioned development, software collaborative development, design rational

1. はじめに（研究の背景と目的）

派生開発で『部分理解』の状態が生じるのは，要求定義工程と設計工程の２箇所である。そこで，この問題を解決するために，要求定義工程では，開発が本格化する前にプロジェクトの主要メンバーが集まって，顧客との折衝で得られた要求仕様の内容の理解と妥当性確認のための議論を行うことを提案する。そして，この議論で得られた要求仕様に対する理解(解釈)をプロジェクトのメンバー全員で共有することが重要であると強く主張する。その上で，議論に参加しなかった人でも議事録を読めば容易に理解できるように，議論

の内容を整理(構造化)しながら，即時記録するためのモデルを提案する。設計工程では，設計案を検討する際に，担当者が提案した設計案を巡って，関係者がそれぞれの立場から賛否の意見とその根拠を述べたり，別の設計案を提案したりすることによって設計案を決めて行く協調的な設計方式を提案する。このような設計方式を可能にするには，設計案を巡ってネットワーク上で議論できるようなツールが必要である。このツールを使って行う議論において，議論の内容を整理(構造化)しながら，即時記録するためのモデルを提案する。

2. 要求仕様の理解と妥当性確認のためのモデルに求められる条件と既存モデルの評価

2.1. 要求仕様の理解と妥当性確認のためのモデルに求められる条件

議論の内容をリアルタイムで整理(構造化)して記録できるようにするためには，議論の内容を記録するためのモデルが，次の条件を満たしている必要がある．

[C1]議論が発散しないように論点を絞り込むような構造になっていること．

[C2]発言内容の要約を何処に記述するか，記述場所を迷うことがないような構造になっていること．

[C3]次の発言を待たなければ記述できないような記述項目を持たないこと．

また，議論によってソフトウェアの要求仕様の妥当性をチェックできるようにするためには，議論の内容を記録するためのモデルが，次の条件を満たしている必要がある．

[C4]各発言がどのような立場での発言かということが分かるような構造になっていること．

何故なら，開発予定のソフトウェアに求められる要求は，それを求める人の立場によって異なることが少なくないからである．

[C5]当該領域の要求仕様に制約を与える法律の条文やガイドラインの記述項目などとの照合に便利な構造になっていること．

[C6]要求された内容を要求仕様書の何処に記述するかということを IEEE830 目次項目で指定できること．

これは，要求事項相互の無矛盾性チェックや獲得した要求仕様を基に要求仕様書を自動生成する際に利用するためである．

[C7]議論に加わった人全員から賛成意見が得られていることを確認できるような構造になっていること．

議論に加わらなかった人でも要求仕様の内容を理解できるようにするためには，議論の内容を記録するためのモデルが，次の条件を満足している必要がある．

[C8]ノード内に記入された文章の分からない所を質問したり，コメントしたりできること．

[C9]提示された質問やコメントに対して回答できるとともに，回答がなされたことが他の人にも判るような構造になっていること(言い換えれば，議論が閉じていることを視覚的にも確認できること)．

[C10]すべてのノードのノード内記述項目を寄せ集めると，記述項目に遺漏が無く，かつ，記述内容が自己完結していること．

この条件は，『すべてのノードの記述項目を読めば，議論に加わらなかった人でも，議論に参加した人と等価な情報を取得できるような構造になっていること』と言い換えることができる．

2.2. 既存モデルの転用可否の分析

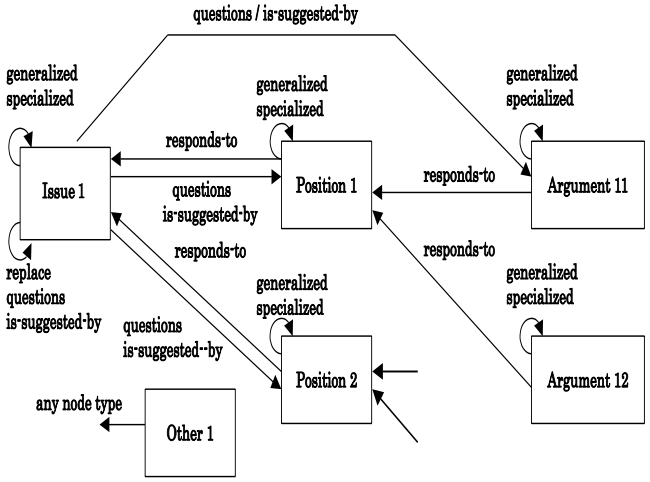


図 1　gIBIS モデル[Conklin(1988)]

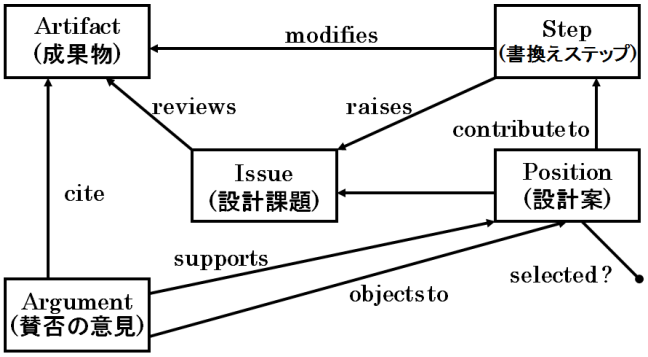
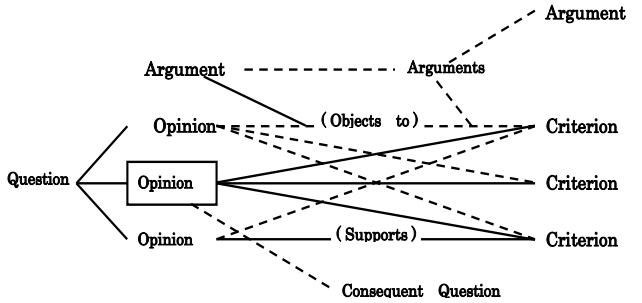


図 2　Potts のモデル[Potts(1988)]

要求仕様の内容理解と妥当性チェックのための議論を，構造化して記録するためのモデルに関する研究事例は未だない．そこで，ソフトウェア設計上の意思決定とその根拠に関する議論を記録するための，既存のモデル(図 1～図 5)の中で，求めるモデルへ転用できるものはないかどうか調査した．しかし，これらの中で，条件[C1]～[C10]のすべてを満足するものは皆無であった．(具体的には，全てのモデルが[C4][C5][C6]を満たさない．その上，図 1 は[C1]を，図 2 と図 3 は[C3]

を，図 4 は[C2]をそれぞれ満たさないことが判った．)
以上の議論から，本論文の目的に適合するモデルをゼロから作成しなければならないことが判明した．



［説明］ 上図は，1 つの Question(課題)に対して複数の Option(解決策)があり，それらの解決案の採否を決定する Criterion(評価基準)が複数考えられるとき，各評価基準はそれぞれの解決案の採用を支援している(Supports)か、否定している(Objects to)かという状況を，Opinion と Criterion との間に張られたリンクのラベルで示している。上図では，Supports は実線で，Objects to は破線で示されている。なお，Supports か Objects to かについて，付帯する議論や条件がある場合には，Arguments(議論)ノードでこれを付記することができる。

図 3　QOC モデル[MacLean(1989)]

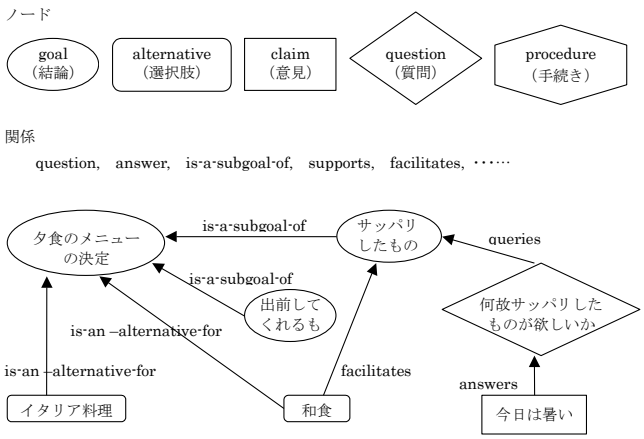


図 4　DRL(SIBYL)による記述例[Lee(1990)]

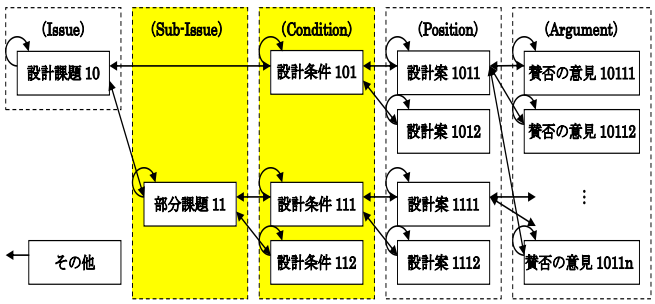


図 5　改良型 IBIS モデル[Komiya(1993)]

3. ノードとノード内記述項目の抽出

3.1. 必要となるノードの抽出

本節では，2.1 節で列挙した，議論を構造化して即時記録するためのモデルに求められる条件の中から，[C1][C2][C8][C9]を採り上げ，ノードを抽出するための条件として使用する．

(1)『要求項目(Requirement)』ノードの設置理由

要求仕様の内容を理解するとともに，その妥当性を議論するためには，議論の対象となる要求項目を記述するためのノードが必要である．このため,『要求項目』というノードを設ける．そして，条件の[C1]と[C2]を満足させるために,『要求項目』ノードには要求事項を１つしか含まない要求項目(= 原子要求)を１つだけ記述するようにする．

(2)『部分要求項目(Sub-requirements)』ノードの設置理由

原子要求のつもりで記述した『要求事項』ノードに，複数の要求事項が含まれていることが判明した場合には，その要求事項を細分化し，原子要求ごとにノードを立てて，それぞれのノードに原子要求の 1 つを記述する必要がある．このため,『部分要求事項』というノードを設け,『要求事項』の直下のノードとして追加できるようにする．原子要求のつもりで記述した『部分要求事項』ノードに，更に複数の要求事項が含まれていることが判明した場合にも，その要求事項を細分化し,原子要求ごとに『部分要求事項』ノードを立てて，それぞれのノードに原子要求を記述する必要がある．

(3)『解釈(Interpretation)』ノードの設置理由

条件の[C1]と[C2]を満足させるためには，提示された原子要求ごとに，原子要求をどのように解釈するかについての様々な提案が寄せられるようにする必要がある．このため,『解釈』というノードを設け,『要求項目』または『部分要求項目』ノードの直下に，このノードを追加できるようにして，その中に解釈を１つだけ記入するようにする．

(4)『問題点(Problem)』ノードの設置理由

条件の[C1]と[C2]を満足させるためには，提案された原子要求に対する解釈に対して，問題点が有るか無いか,問題点があるとすれば何が問題かについて,様々な角度から意見が寄せられるようにする必要がある．このため,『問題点』というノードを設け,『解釈』ノードの直下に，このノードを追加できるようにする．このとき寄せられる問題点の指摘には，1 つの解釈に閉じた指摘と,複数の解釈に絡む指摘の 2 種類がある．

(5)『対策案(Countermeasure)』ノードの設置理由

条件の[C1]と[C2]を満足させるためには，指摘された問題点ごとにそれぞれ，できるだけ多くの対策案が寄せられるようにする必要がある(対策案の中には,顧客に確認するという対策案も含まれる). このため,『対策案』というノードを設け,『問題点』ノードの直下に，このノードを追加できるようにし，その中に対策案を１つだけ記入するようにする．

(6)『賛否の意見(Opinion)』ノードの設置理由

条件の[C1]と[C2]を満足させるためには，提案された対策案について，対策案ごとに賛成か反対かを明ら

かにするとともに，その理由を提示する必要がある．このために，『賛否の意見』というノードを設け，『対策案』ノードの直下に追加できるようにする．しかも，賛否の意見は，対策案ごとに議論の参加者全員から，それぞれ意見が寄せられる必要がある．

(7) 『質問またはコメント(Question)』ノードの設置理由

これまで列挙してきたどのノードにおいても，その中の記述が最初から完璧であることは少なく，それを記述したノードに対して質問またはコメントできるようにする必要がある(条件[C8]に該当)．

このため，『質問またはコメント』というノードを設け，すべてのノードの直下に，これを追加できるようにする．そして，このノードの中に質問またはコメントを１つだけ書くようにする．

(8) 『回答(Answer)』ノードの設置理由

１つの質問またはコメント対して，その回答を記入するためのノードが必要である(条件[C9]に該当)．このため，『回答』というノードを設け，『質問またはコメント』ノードの直下に，追加できるようにする．

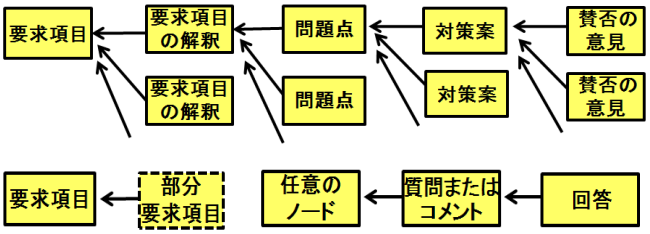


図 6 要求仕様の理解と妥当性確認のための議論を構造化して即時記録するためのモデル

上記の議論により求めるモデルは図 6 の通りである．

3.2.ノードではなく，『立場』というノード内記述項目を設けた理由

開発対象となるシステムに求められる要求は，それを要求する人の立場によって異なってくる．このため，そのノードを記述する人が，どのような立場で記述したかを明らかにする必要がある．立場の種類としては，エンドユーザ(開発したシステムによってサービスを受ける人)，オペレータ(開発したシステムの操作者)，顧客側の管理責任者，開発の担当者，開発者側の管理責任者，法律(このシステムを構築する際に考慮しなければならない法律)，ガイドライン(このシステムを構築する際に考慮しなければならないガイドライン)などを挙げることができる．

もし，各発言がそれぞれどのような立場からの発言であるかを明らかにするために『立場』というノードを設けるとしたら，すべてのノードの前に『立場』というノードを置かなければならなくなる．このため，末端のノードまでの階層が深くなり，議論の結論が出るまでの時間がかかり過ぎる．このため，『立場』というノードを設けず，『立場』というノード内記述項目をすべてのノード内に設けた．

3.3. 記述済みのノード内記述を修正する方法

記述者がノード内に記述して，これで良しとした原稿は修正することができないようにする．その後で，記述済みのノード内記述を修正するには，修正対象と同じ種類のノードを別に立てて，そこに正しい内容を記述することによって行う．このとき，修正後のノードがどのノードの修正版になっているのかが判るようにする(ノード内記述項目にそれを明記する)．図 7 は，『質問またはコメント』の記述が契機となって『ノード X1』の記述内容が『ノード X2』に書き改められた場合を示している．その手順は次の通り．

①『質問またはコメント』ノードを使って，他の人が記述の不備を指摘する．

②『X1』ノードを記述した人が，『回答』ノードを使って『X1』ノードの内容を修正すると回答する．

③『X1』ノードの修正版として『X2』ノードを作成し，その中の修正対象というフィールドにノード X1 と記入することにより，『X2』ノードが，『X1』ノードの修正版であることを明記する．(①→②という手続きを経ずに，『X1』ノードを記述した人が，いきなり③を行うこともできる．)

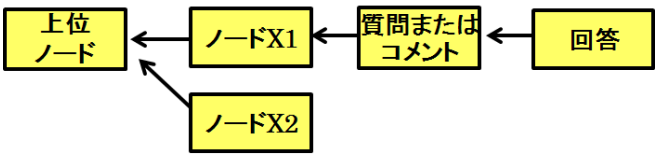


図 7 記述済みノードの記述内容を修正する方法

3.4. 議論の参加者全員から賛成意見が得られていることの確認方法

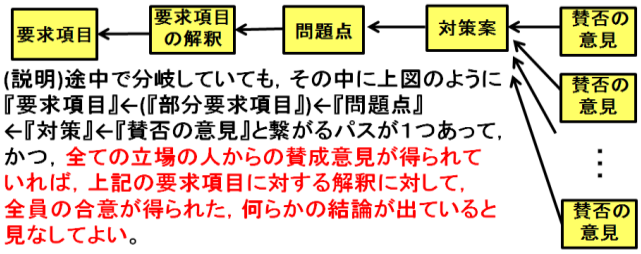


図 8 会議の参加者全員から賛成意見が得られていることの確認方法

条件[C7]により，議論の参加者全員から賛成意見が得られていることを確認できなければならない．そのための方法は図 8 の通りである．

3.5. 各ノード内で必要となる記述項目

条件[C2][C3][C4][C5][C6][C8][C9][C10]を満足させ

るためには，各ノード内では，どのような記述項目が必要かという視点から，各ノード内で必要な記述項目を考察する．

(1)『要求項目』ノード内の記述項目

条件[C2][C3][C4][C5] [C10]を満足させるために，次のような記述項目が必要となる．

①ID： このノードを識別するための ID．システムによって自動的に付与される，R で始まる何桁かの追い番で構成される．

(例) R00001，R00002，R00003，･･･

②修正対象: このノードが他のノードの修正版である場合には，修正対象となるノードの ID を書き，このノードがそのノードの修正版であることを示す．修正版ではない場合には『該当しません』と書く．

③改造対象のシステム名: 既存システムの改造によってシステムを構築する場合に，改造の対象となるシステムの名称を記述する．改造ではない場合には『該当しません』と書く．

④記述者名： このノードをおこし，その中身を記述した人の氏名を記述する．記述者自身が記述する．

⑤立場: 開発対象となるシステムに求められる要求は，それを要求する人の立場によって異なってくる．このため，このノードを記述する人が，どのような立場で記述しているかを複数の中から選ぶ形で明らかにする．

⑥開発対象： 開発しようとしているシステムの名称．

⑦要求内容 : 要求内容(原子要求)を具体的に記述する．

⑧理由または目的：何のためにそのような要求が提示されたのか，その理由または目的を記述する．

(2)『部分要求項目』ノード内の記述項目

『要求項目』または『部分要求項目』ノードの⑦に記述された要求項目に，複数の要求項目が含まれることが判明した場合に，分割して出来た要求項目(= 部分要求項目)の 1 つを記述するためのノードである．

①このノードを識別するための ID．システムによって自動的に付与される，S で始まる何桁かの追い番で構成される．

(例) S00001，S00002，S00003，･･･

②修正対象:『要求項目』ノード内記述項目②と同じ．

③上位ノード:直接このノードに接続する上位ノードの ID.

④記述者名:『要求項目』ノード内記述項目④と同じ．

⑤立場 :『要求項目』ノード内記述項目⑤と同じ．

⑥開発対象:『要求項目』ノード内記述項目⑥と同じ．

⑦要求内容:『要求項目』ノード内記述項目⑦と同じ．

⑧理由または目的 :『要求項目』ノード内記述項目⑧と同じ．

(3)『解釈』ノード内の記述項目

条件[C2][C3][C4][C5][C6] [C10]を満足させるために，次のような記述項目が必要となる．

①ID： このノードを識別するための ID．システムによって自動的に付与される，I で始まる何桁かの追い番で構成される．

(例) I00001，I00002，I00003，･･･

②修正対象 :『要求項目』ノード内記述項目②と同じ．

③上位ノード :『部分要求項目』ノードの③と同じ．

④記述者名 :『要求項目』ノード内記述項目④と同じ．

⑤立場 :『要求項目』ノード内記述項目の⑤と同じ．

⑥開発対象 :『要求項目』ノード内記述項目⑥と同じ．

⑦解釈： その立場から考えた場合の，原子要求に対する解釈を記述する．

⑧解釈の根拠: ⑦のような解釈をした根拠を記述する．

⑨要求仕様書の記述場所： この原子要求を IEEE830 で推奨されている目次項目のどこに記述するべきかを記述する．これによって，同じ目次項目に分類された原子要求同士の照合が可能となり，要求事項の漏れや互いに矛盾する要求事項の有無をチェックすることが可能となる．

(4)『問題点』ノード内の記述項目

条件[C2][C3][C4][C5]を満足させるために，次のような記述項目が必要となる．

①ID： このノードを識別するための ID．システムによって自動的に付与される P で始まる何桁かの追い番で構成される．

(例) P00001，P00002，P00003，･･･

②修正対象 :『要求項目』ノード内記述項目②と同じ．

③上位ノード :『部分要求項目』ノードの③と同じ．

④記述者名 :『要求項目』ノード内記述項目④と同じ．

⑤立場 :『要求項目』ノード内記述項目⑤と同じ．

⑥開発対象 :『要求項目』ノード内記述項目⑥と同じ．

⑦問題点の有無: 上位ノードに記述された解釈に問題があるか無いかの区別を記述する．

⑧問題点の指摘： 問題ありの場合に，問題点(複数あっても，その中の 1 件だけ)の内容を具体的に書く．問題なしの場合には，『該当しません』と書く．

⑨理由または根拠： それを問題点だと見なす，その理由または根拠を記述する．

(5)『対策案』ノード内の記述項目

条件[C2][C3][C4][C5] [C10]を満足させるために，次のような記述項目が必要となる．

①ID： このノードを識別するための ID．システムによって自動的に付与される C で始まる何桁かの追い番で構成される．

(例) C00001，C00002，C00003，･･･

②修正対象 :『要求項目』ノード内記述項目②と同じ．

③上位ノード :『部分要求項目』ノードの③と同じ．

④記述者名:『要求項目』ノード内記述項目④と同じ.
⑤立場:『要求項目』ノード内記述項目⑤と同じ.
⑥開発対象:『要求項目』ノード内記述項目⑥と同じ.
⑦対策の要否:『対策要』または『対策不要』と書く.
⑧理由または根拠:『対策要』または『対策不要』である，その理由または根拠を書く.
⑨対策案の内容:『対策要』の場合に，対策(複数あってもその中の 1 件だけ)の内容を具体的に書く.

(6)『賛否の意見』ノード内の記述項目

条件[C2][C3][C4][C5][C10]を満足させるために，次のような記述項目が必要となる.
①ID:このノードを識別するための ID.システムによって自動的に付与される，O で始まる何桁かの追い番で構成される.
(例) O00001，O00002，O00003，･･･
②修正対象:『要求項目』ノード内記述項目②と同じ.
③上位ノード:『部分要求項目』ノードの③と同じ.
④記述者名:『要求項目』ノード内記述項目④と同じ.
⑤立場:『要求項目』ノード内記述項目⑤と同じ.
⑥開発対象:『要求項目』ノード内記述項目⑥と同じ.
⑦賛否の区別:賛成意見を書くときには『賛成』と書き，反対意見を書くときには『反対』と書く.
⑧賛否の意見:どこがどのように賛成または反対であるか,その理由を含めて意見の内容(複数ある場合には,箇条書きで)を具体的に書く.

(7)『質問またはコメント』ノード内の記述項目

条件[C2][C3][C4][C8]を満足させるために,次のような記述項目が必要となる.
①ID:このノードを識別するための ID.システムによって自動的に付与される，Q で始まる何桁かの追い番で構成される.
(例) Q00001，Q00002，Q00003，･･･
②修正対象:『要求項目』ノード内記述項目②と同じ.
③上位ノード:『部分要求項目』ノードの③と同じ.
④記述者名:『要求項目』ノード内記述項目④と同じ.
⑤質問かコメントかの区別:質問の場合には質問と書き，コメントの場合にはコメントと書く.
⑥質問またはコメントの内容:質問またはコメントの内容を具体的に書く.

(8)『回答』ノード内の記述項目

条件[C2][C3][C4][C9]を満足させるために,次のような記述項目が必要となる.
①ID:このノードを識別するための ID.システムによって自動的に付与される，A で始まる何桁かの追い番で構成される.
(例) A00001，A00002，A00003，･･･
②修正対象:『要求項目』ノード内記述項目②と同じ.
③上位ノード:『部分要求項目』ノードの③と同じ.
④記述者名:『要求項目』ノード内記述項目④と同じ.
⑤立場:『要求項目』ノード内記述項目⑤と同じ.
⑥開発対象:『要求項目』ノード内記述項目⑥と同じ.
⑦回答の内容:上位ノードに書かれた質問またはコメントに対する回答の内容を具体的に書く.

4. ソフトウェア設計の為の議論を構造化して即時記録するためのモデル

4.1 モデルに求められる条件

議論の内容をリアルタイムで整理(構造化)しながら記録できなければならないという点では,『設計のための議論を記録するモデル』も，2.1 節で述べた『要求仕様の理解と妥当性確認のためのモデル』の場合と共通である。従って，このモデルも条件[C1][C2][C3]を満たす必要がある。

また，条件[C7][C8][C9][C10]も,『要求仕様の理解と妥当性確認のためのモデル』の場合と共通なので，このモデルもこれらを満たす必要がある。

一方,条件[C4][C5][C6]は,『要求仕様の理解と妥当性確認のためのモデル』だけに求められる条件なので，このモデルがこれらの条件を満たす必要はない。では,『設計のための議論を記録するモデル』だけに求められる条件は何か？それは条件[C11]である。

[C11] 議論と同時進行で記録した設計案や議論の内容を，再構成しなくても再利用できること。

[C11]は『要求仕様の理解と妥当性確認のためのモデル』に必要な条件ではない。何故なら，このモデルでは，議論の対象となる要求仕様を原子要求に分解して議論することになるので，再利用するには粒度が小さ過ぎるのである。ここでは，この議論全体で 1 つの再利用対象となるので，議論の内容を構造化して記録するためのモデルに求められる条件とはならない。

しかし,『設計のための議論を記録するモデル』にとって[C11]は極めて重要な条件である。このことは，2.2 節で挙げた，改良型 IBIS を除くすべてのモデルが，即時記録を可能にするための条件[C1][C2][C3]を満足させることを放棄することによって条件[C11]を満足させていることからも理解できる。(即時記録が可能であって，かつ，記録した設計案や議論の内容を再構成しなくても再利用できるのは，改良型 IBIS だけである。)

4.2 改良型 IBIS モデルのノード内記述項目

改良型 IBIS モデルは図 5 の通りである。また，図 5 に示した各ノードのノード内記述項目は次の

通りである。

(1) 設計課題ノード
- ① 設計課題 ID：
- ② 修正対象：
- ③ 設定者名：
- ④ 設定履歴：
- ⑤ 設計課題名：
- ⑥ 選定理由：
- ⑦ 課題の内容：
- ⑧ 部分課題へのポインタ：
- ⑨ 適用分野：
- ⑩ 設計条件へのポインタ：
- ⑪ 設計案適用履歴：

(2) 部分課題ノード
- ① 部分課題 ID：
- ② 修正対象：
- ③ 設定者名：
- ④ 設定履歴：
- ⑤ 部分課題名：
- ⑥ 選定理由：
- ⑦ 課題の内容：
- ⑧ 上位課題へのポインタ：
- ⑨ 適用分野：
- ⑩ 設計条件 ID へのポインタ：
- ⑪ 設計案適用履歴：

(3) 設計条件ノード
- ① 設計条件 ID：
- ② 修正対象：
- ③ 設定者名：
- ④ 設定履歴：
- ⑤ 対象設計課題 ID へのポインタ：
- ⑥ 設計案：
- ⑦ 前提条件：
- ⑧ 排他的条件：
- ⑨ 事前検討課題：
- ⑩ 事後検討課題
- ⑪ 同時検討課題

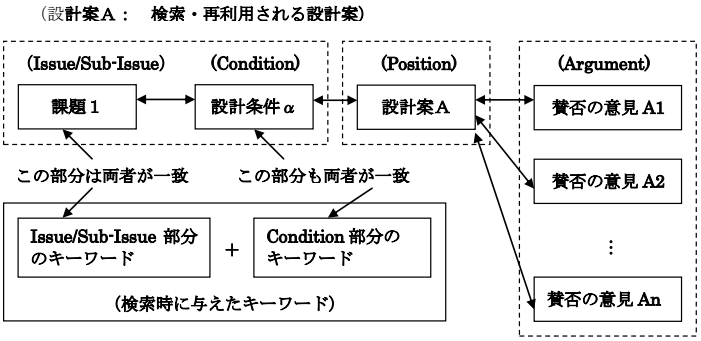
(4) 設計案ノード
- ① 設計案 ID：
- ② 修正対象：
- ③ 設計者名：
- ④ 作成履歴：
- ⑤ 設計条件 ID へのポインタ：
- ⑥ 設計案の内容：
- ⑦ 賛否の意見へのポインタ：
- ⑧ 異条件下の設計案：

(5) 賛否の意見ノード
- ① 賛否の意見 ID：
- ② 修正対象：
- ③ 意見提出者名：
- ④ 提出履歴：
- ⑤ 賛否の区分：
- ⑥ 対象設計案へのポインタ
- ⑦ 意見の内容

(6) 質問またはコメントノード
- ① 質問またはコメントノード ID：
- ② 修正対象：
- ③ 上位ノードへのポインタ：
- ④ 記述者名：
- ⑤ 質問かコメントかの区別：
- ⑥ 質問またはコメントの内容：

(7) 回答ノード
- ① 回答ノード ID：
- ② 修正対象：
- ③ 上位ノードへのポインタ：
- ④ 記述者名：
- ⑤ 回答の内容：

4.3 改良型 IBIS モデルで即時記録された議論の内容を再構成せずに再利用する方法



［説明］検索のために Issue(または Sub-Issue)部分と Condition の部分の両方に対して，それぞれキーワードを与える。このキーワードが，同じ部分に対して登録時に与えられていたキーワードと完全に一致すれば，課題 1(設計課題 1 または部分課題 1)と設計条件 α が検索され，続いて設計案Aとそれに対する賛否の意見が次々に呈示される。

図 9 課題検索用キーワードと設計条件検索用キーワードの両方が一致する場合の再利用

改良型 IBIS モデルを使って，理解し易いように整理(構造化)して即時記録された，設計案を巡る議論の内容を再利用するために与える情報は，設計課題検索用キーワードと設計条件検索用のキーワードである。このときのキーワードの一致状況によって，再利用の方式は次の 2 種類になる。

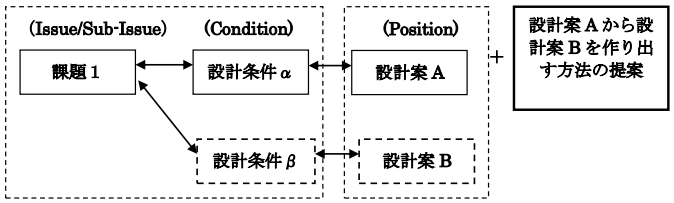
(1) 設計課題検索用キーワードと設計条件検索用キーワードの両方が，それぞれ検索対象にヒットした場合(図 9 の場合)

この場合には，想定する設計課題に対する想定する設計条件の設計案およびその選択を巡る議論がデータベースに収録されていたことになるので，その設計案とその選択を巡る議論を事例として検索し提示する。

(2) 設計課題検索用キーワードだけがヒットした場合(図 10 の場合)

この場合には，想定する設計課題に対する設計条件の異なる設計案およびその選択を巡る議論だけがデータベースに収録されていたことになるので，その設計案とその選択を巡る議論を参考事例として検索し提示するとともに，提示された設計案を，求める設計条件の下での設計案へ改造する方法を事例ベース推論することにより，参考意見としてメッセージの形で提示する。

改良型 IBIS では，(1)と(2)のいずれにおいても，設計案とその選択を巡る議論の内容を収録したデータベースを再構成しなくても再利用できる。



［説明］課題 1(設計課題 1 または部分課題 1)に対する設計条件βの下での設計案 B を検索できなかったとき，設計案 B の代わりに，同一の設計課題 1 に対する設計条件αの下での設計案 A を検索・呈示するとともに，設計案 A を改造して，これから設計条件βの下での設計案 B を作り出す方法を呈示する。

図 10 課題検索用キーワードだけが一致する場合の再利用(事例ベース)

5. おわりに

派生開発で『部分理解』の状態が生じるのは，要求定義工程と設計工程の 2 箇所である。そこで，この問題を解決するために，要求定義工程では，開発が本格化する前にプロジェクトの主要メンバーが集まって，顧客との折衝で得られた要求仕様の内容の理解と妥当性確認のための議論を行うことを提案した。そして，この議論で得られた要求仕様に対する理解(解釈)をプロジェクトのメンバー全員で共有することの重要性を強調した。その上で，議論に参加しなかった人でも議事録を読めば容易に理解できるように，議論の内容を整理(構造化)しながら，即時記録するためのモデルを提案した。設計工程では，設計案を検討する際に，担当者が提案した設計案を巡って，関係者がそれぞれの立場から賛否の意見とその根拠を述べたり，別の設計案を提案したりすることによって設計案を決めて行く協調的な設計方式を提案した。このような設計方式を可能にするには，設計案を巡ってネットワーク上で議論できるようなツールが必要である。このツールを使って行う議論において，議論の内容を整理(構造化)しながら，即時記録するためのモデルを提案した。

今後の課題としては，ツールの開発とそれを使った実験により，提案したモデルの有効性を示すことである．また，特定の顧客を持たないシステムを対象とする『要求仕様の理解と妥当性確認のためのモデル』についても検討する必要がある。

文　献

[1] J. Conklin and M. L. Begeman, “gIBIS: A Hypertext Tool for Exploratory Policy Discussion,” CSCW '88 Proceedings, ACM, pp.140-152, 1988.

[2] Seiichi Komiya, “A Model for Recording Software Design Decisions and Design Rationale, “IEICE transactions on Information and Systems, Vol. E81-D, No. 12, pp. 1350-1363, 1998.

[3]W. Kunz, et al., “Issues as Elements of Information Systems,” No. Working Paper #131,　University of California Berkeley, 1970.

[4] 桑名栄二, “ソフトウェア履歴利用の研究動向, “電子情報通信学会, Vol.77, No.5, pp.531-538, 1994.

[5] J. Lee, “SIBYL: A Qualitative Decision Management System,” in Artificial Intelligence at MIT, P. H. Winston, and S. A. Shellard, EDs. pp. 104-133, MIT Press, 1990.

[6] J. Lee, “SIBYL, “A Tool for Managing Group Decision Rationale,” CSCW '90 Proceedings, ACM pp.79-92, 1990.

[7] J. Lee, “Extending the Potts and Bruns Model for Recording Design Rationale, “Proceedings of the 13th International Conference on Software Engineering, IEEE, pp.114-125, 1991.

[8] A. Maclean, R. M. Young, and T. P. Moran, “Design Rationale: The Argument behind the artifact, “In proceedings of CHI'89 Human Factors in Computing System, pp.247-252, 1989.

[9] A. Maclean, R. M. Young, V. M. E. Bellotti, and T. P. Moran, “Question, Options, and Criteria: Elements of design space analysis, Human Computer Interaction, 1991.

[10] C. Potts and G. Bruns, “Recording the Reasons for Design Decisions, “Proceedings of the 10th International Conference on Software Engineering, IEEE, pp. 418-427, 1988.

[11] C. Potts, “A Generic Model for Representing Design Methods, “Proc. of the 11th International Conference on Software Engineering, IEEE, pp. 217-226, 19889.