社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－SE－144 (1)
2004／3／18

ソフトウェア協調開発コミュニティにおける要求抽出手法とリリース計画の実現

島影 正俊　櫨山 淳雄

東京学芸大学大学院教育学研究科

〒184-8501 東京都小金井市貫井北町 4-1-1

E-mail: {m021110, hazeyama}@u-gakugei.ac.jp

あらまし　筆者らはより実践的なソフトウェア工学教育を大学の学生に提供するため，ソフトウェア協調開発コミュニティを開発し，試行実験を行った．その結果，ソフトウェア要求者は機能要求だけでなくユーザインターフェースに対する変更要求を頻繁に行っていた．そこで本論文ではソフトウェア協調開発コミュニティにおけるユーザインターフェース指向の要求抽出手法を提案し，その手法を支援するシステムを開発した．このユーザインターフェース指向の要求抽出手法は，機能要求仕様とユーザインターフェース仕様と密に連携させている．加えて，不特定多数の要求者と開発者の機能要求優先度を考慮したリリース計画の作成について述べる．

Implementation of Requirement Elicitation Method and Release Planning in Collaborative Software Development Community

Masatoshi SHIMAKAGE　Atsuo HAZEYAMA

Tokyo Gakugei University, Graduate School of Education

4-1-1, Nukuikitamachi, Koganei-shi, Tokyo, 184-8501 Japan

E-mail: {m021110, hazeyama}@u-gakugei.ac.jp

Abstract. We developed a collaborative software development community-ware whose goal is to provide practical tasks for the students in software engineering education in a university and to provide software for education purpose. We performed a pilot experimentation for the community-ware. The results show that software requesters made software change requests for not only functional aspects but also user interface. This paper proposes a user interface oriented requirement elicitation process. The process is that software requesters present functional requests and user interface design which met the requests. Then based on the requests and the user interface, developers and software requesters decide priority for each requirement in a collaborative manner.

1.はじめに

近年の情報化の流れに伴い，ソフトウェア開発業界では，顧客に対して高品質なソフトウェアを提供するために高い情報処理能力を有する人材が必要になってきている．そのため，大学の情報系学部・学科ではモデリング教育，プログラミング教育，アルゴリズム教育などが行われているが，より実践的なソフトウェア設計・開発を行う機会は多いとはいえないのが現状である[4]．ましてや，情報系学部・学科所属の学生は与えられた開発課題ではなく，実際に利用したいと考えている人からのソフトウェア要求を要求仕様書にまとめ，分析，設計，開発，テスト，保守を行うという，より実践的な開発を行える機会は少ない．

そこで我々はインターネットを介してソフトウェア開発要求を有している人（以下，要求者と呼ぶ）と大学の情報系学科所属の学生（以下，開発者と呼ぶ）と

の間でソフトウェアを協調開発するコミュニティモデルを構築し[3]，そのモデルに基づいたコミュニティウェアを実装し，試行実験を行った[6]．

その試行実験の結果，以下のような問題点が明らかになった．

1. 要求者は機能要求だけでなく，開発されたソフトウェアのユーザインターフェース（以下，UIと表記）に関心をもっており，ソフトウェアのリリースの際，UIに対する変更要求が頻発した．
2. ソフトウェアの機能要求に対して不特定多数の要求者がそれぞれ異なった優先度を持っており，要求者，開発者双方が同意できるリリース計画の作成が困難であった．

要求者と開発者が要求仕様を決定する過程では，主に機能要求についての議論が中心となり，UIについての議論が不十分なまま次のフェーズに進むことが多かった．一般的に「*ユーザにとってユーザインターフェースこそがシステム*」であると言われているので，要求者は想定していたものとは異なるUIを持つソフトウェアを受け入れることができないと考えられる．そこで，要求者と開発者が協調して，機能要求だけでなくUIを含めた要求仕様の決定を行える機構が必要である．

また，一般のソフトウェア開発において，顧客サイドは企業にソフトウェア開発を依頼する際，様々な要求をするにしても事前に顧客側で要求優先度の調整がなされる．しかしながら，我々が対象とするソフトウェア協調開発コミュニティでは，あるソフトウェアに対して要求者は不特定多数である可能性があり，各々の要求者がそれぞれ異なった優先度を持っている．そのため，コミュニティ上で要求者の優先度調整を行い，その優先度に基づいたリリース計画作成支援を行う必要がある．

以上の背景より本研究では，ソフトウェア協調開発コミュニティにおいて，要求者と開発者間で機能要求と同時に明確なUI仕様の合意を得られるような要求抽出手法と，不特定多数の要求者と開発者の双方が同意できる機能要求優先度決定機構によるリリース計画作成の提案と，それらの機構を実現したコミュニティウェアを実装することを目的とする．

2. ソフトウェア協調開発コミュニティ

要求抽出手法とリリース計画について述べる前に，前提となるソフトウェア協調開発コミュニティモデルについて説明する．

ソフトウェア協調開発コミュニティは学習者である開発者と不特定多数の要求者から構成される．開発者はインターネットを介して要求者から要求仕様を抽出する．そして，その要求仕様をもとにソフトウェアを分析，設計，実装し，再びインターネットを介して要求者に公開する．このコミュニティは，要求者にとって真に要求するソフトウェアを安価で提供し，また，開発者にとっては要求定義，適用・保守を含めたソフトウェア開発プロセス全体の体験的学習を目的としている．

我々が想定しているソフトウェア協調開発コミュニティモデルは[2]で述べられている内容を踏まえ，以下に示すような流れになる．また，図1にそのモデル図を示す．

1. 要求者と開発者は機能要求を決定する．
2. 要求者と開発者は非機能要求を決定する．
3. 要求者と開発者はリリース計画を決定する．
4. 開発者は3で決定されたリリース計画に従い，ソフトウェアを実装する．
5. 要求者はリリースされたソフトウェアのテストを行う．
   - 5.1. テストの結果，要求者が満足すれば次に優先度の高いグループを実装したソフトウェアをリリースする．
   - 5.2. テストの結果，要求者が不満を持ったらその不満を解消するまで修正を行う．
6. リリース計画に含まれる機能要求がすべて実装されるまで4から5を繰り返す．

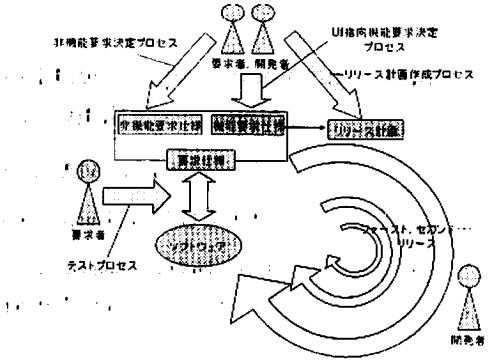

図1 ソフトウェア協調開発コミュニティモデル

3. UI指向機能要求抽出手法

前述の問題点1の解決方法として，我々は機能要求仕様にUIに関する仕様を関連付けるというアプローチを採用する．つまり，機能要求とUI要求を密に連携させた要求仕様を作成することである．これにより機能要求に関する議論を行う際には，必然的にUIに関する議論も行われるため，要求者と開発者間で機能要求と同時に明確なUI仕様の合意を得ることが可能になると考えている．

このUI指向機能要求抽出手法は，ソフトウェア協調開発コミュニティモデルでいうと機能要求仕様を作成するプロセスに相当する．そこで本研究では，このプロセスをUI指向機能要求決定プロセスと呼ぶことに

し，要求イメージ作成，機能要求抽出，UI 作成，機能要求洗練の 4 ステップを定義した．要求イメージ作成で，要求者のイメージしているソフトウェア要求をコミュニティに表出させる．機能要求抽出では，漠然としたイメージを機能要求として切り出す．UI 作成では，ソフトウェアの画面設計を行い，機能要求洗練で，機能要求と UI 要求を組み合わせた要求仕様を作成する．以降で，ステップごとに具体例を交えた詳細な説明を行う．

3.1. 要求イメージ作成

要求イメージ作成は UI 指向機能要求決定プロセスの最初のステップである．

本研究において要求イメージとは「ソフトウェアに行ってもらうサービスの漠然としたイメージの内容」とする．つまり要求者が望むシステムの特徴と考えることができる．この要求イメージは，構造化されていない自然言語による文章という形で表現される．そのため抽象的であったり，要求に対して不足があったりすることが予想される．しかしながら，この段階においてそのような曖昧性が存在してもかまわない．なぜなら，本ステップの目的は，「要求者がどのようなソフトウェア要求を有しているか」，をコミュニティに表出することであり，曖昧性の排除は以降のプロセスで行うからである．

例として，試行実験での要求者からの最初の要求記述は以下の通りであった．なお，提案しているソフトウェアは学校の座席表作成システムである．

「クラス名簿から簡単に座席表が作成できるシステム．名前順，男女別または男女混合座席，ランダム，くじ引きが選択でき，画面表示で確認してマニュアル操作で入れ替えもできてそれを印刷もできる．」

3.2. 機能要求抽出

このステップの目的は前述の要求イメージを洗練することである．このステップにおいて，開発するソフトウェアの機能要求を抽出する．本研究では，機能要求の抽出を UML[7]におけるユースケースの抽出と捉える．機能要求をユースケースとして捉えることで，開発者にとっては，今後 UML を用いて設計を行う際に，その作業をシームレスに行うことができると考えられる．

このステップは，要求者，開発者が協調して行うことを想定している．理想的には要求者が要求イメージや，議論の内容から機能要求を抽出することが望ましいが，そのようなスキルが備わっているとは考えにくい．したがって実際には，開発者がソフトウェアのユースケースを抽出し，抽出したユースケースに対して要求者が確認を取る，といったプロセスになるであろう．

前述の要求イメージの例から機能要求の抽出例として以下のようなものが考えられる．

- クラス名簿を取り込む
- 座席表を作成する
- 座席表を印刷する

3.3. UI 作成

このステップでは，以下に示した 2 つの側面から開発するソフトウェアの UI 仕様を定義する．

- コンポーネント構造の定義
- システムインタラクションの定義

コンポーネント構造の定義ではソフトウェアがどのようなコンポーネントから構成されているか（静的側面）を定義する．言い換えるとコンポーネント構造は開発するソフトウェアの画面設計図に相当する．

またシステムインタラクションの定義は，ソフトウェアの利用者がコンポーネント構造で定義したコンポーネントに対して起こしたイベントに対して，そのソフトウェアがどのように振舞うのか（動的側面）を定義することである．本研究では，この振る舞いのことをリアクションと呼ぶことにする．

例えば，要求者が「右上のプッシュボタンを押したときに座席表が生成される」といったようなイメージを持っていたとする．この場合，コンポーネント構造の定義時には画面の右上の箇所に PushButton を定義することになり，同時にシステムインタラクションの定義では，この配置された PushButton に対して「押す」というイベントが起こった際に「座席表を生成する」というリアクションを定義することになる．

3.4. 機能要求洗練

このステップでは，3.2 節で抽出した機能要求と 3.3 節で作成した UI を組み合わせ，抽出された機能要求をさらに洗練する．その目的のために，抽出された各々のユースケースに対してユースケース記述を作成する．ユースケース記述とは，ユースケースに対するシナリオと，それに付随する情報のことを指す．ユースケース記述のテンプレートは UML で特に決められていないが，本研究では，概要，アクター，事前条件ユースケース，基本系列，例外系列の 5 つを採用した．事前条件ユースケースとはこのユースケースを行う前に実行されていなければならないユースケースを示し，この情報は，後のリリース計画で利用される．基本系列はこのユースケースにおける，ユーザとシステム間の対話を表す一連の手順であり，本研究では UI 作成で定義したシステムインタラクションを組み合わせて作成する．こうすることで，機能要求と UI 要求が密に連携した要求仕様を作成することができ，結果として機能要求と同時に明確な UI 要求を決定することができる．

例として，「座席表を作成する」というユースケースの基本系列として以下のようなシナリオが想定されたとする．

1.アクターは座席作成の種類を選択する
2.アクターは座席作成を実行する
3.システムは座席表を出力する

機能要求洗練のステップの目的は，この基本系列のシナリオを UI 作成時に定義した内容を利用して作成することである．この場合，アクターが主語である 1,2 の文の目的語をアクターが操作するコンポーネントとして捉えることができ，動詞に関してもコンポーネントに対するイベントと捉えることが可能である．そして，3 のシステムが主語の文は，2 のイベントが起こった際システムの反応と捉えることができる．

UI 作成のコンポーネント構造の定義時に ComboBox と PushButton が定義されていたとする．また，システムインタラクションの定義時に PushButton は前述したリアクションが定義されているとする．

まず，座席表を作成するためにアクターが操作しなければならないコンポーネントを選択する．ここでは最初に ComboBox を選択する．次に選択されたコンポーネントに対してアクターが起こすイベントを選択する．ここでは「選択する」というイベントを選択する．もし ComboBox の「選択する」というイベントに対してリアクションが定義されていれば，その内容を踏まえて，基本系列が作成されるが，この例だと定義されていないので，以下のような基本系列になる．

1.アクターは ComboBox を選択する．

続いてアクターが操作するコンポーネントが存在する場合，先ほどと同じようにコンポーネントとイベントを選択していく．例の続きとして，PushButton を選択し，そのイベントとして「押す」を選択したとする．この PushButton のシステムリアクションの「押す」というイベントには「座席表を作成する」というリアクションが定義されているので，最終的には以下の基本系列が作成される．

1.アクターは ComboBox を選択する
2.アクターは PushButton を押す
3.システムは座席表を作成する

このままでは，ComboBox，PushButton という言葉が基本系列に入り込んでしまい，日本語としてわかりにくい．そこで，それぞれのコンポーネントには別名として「座席作成の種類」，「座席作成」と定義することで，先ほど想定したシナリオが作成できる．

4. リリース計画作成

前述の問題点 2 の解決方法として，本研究では機能要求優先度決定に Cost-Value Approach[1]の概念を拡張するというアプローチを採用した．この Cost-Value Approach は，Karlsson らによって考案された機能要求優先度決定手法であり，AHP(Analytic Hierarchy Process)[5]を利用して，要求者は価値という評価基準で，開発者は開発コストという評価基準で機能要求の優先度を決定し，要求者にとって価値が高く，開発者にとって開発コストの低い機能要求を全体の優先度の高い機能要求とするものである．

この Cost-Value Approach に要求者の不特定多数性を考慮することで，不特定多数の要求者と開発者が同意できる機能要求優先度決定が行えることができる．それに加え，本研究では Cost-Value Approach では考慮されなかった機能要求の依存関係を考慮する．Cost-Value Approach では純粋に AHP の計算結果のみで機能要求の優先度を決定しており，機能要求の依存関係を考慮していなかった．つまり，R2 という機能要求は，R1 という機能要求に基づいて実装されなければならない場合が想定されるが，AHP の計算結果のみでリリース計画を作成すると，R1 よりも R2 を先に実装するような計画がなされる可能性がある．

以上を踏まえ本研究ではリリース計画作成プロセスとして提案する．このリリース計画作成プロセスは機能要求一対比較，機能要求見積り，機能要求優先度決定，グルーピングの 4 ステップから構成されている．機能要求一対比較では，要求者それぞれの機能要求優先度を決定し，機能要求見積りでは開発者による見積りから開発者側の優先度を決定する．機能要求優先度決定では，不特定多数の要求者と開発者全体の機能要求優先度を算出する．最後のグルーピングで，機能要求優先度と依存関係を考慮したリリース計画を作成する．

4.1. 機能要求一対比較

このステップでは，UI 指向機能要求決定プロセスで抽出された機能要求（ユースケース）に対して，要求者は各々，ソフトウェアの価値という観点で機能要求の組を表 1 で示した重要性尺度を利用してそれぞれ一対比較する．

表 1 重要性尺度とその定義

重要性尺度	定義
1	同じくらい重要
3	少し重要
5	かなり重要
7	非常に重要
9	極めて重要

例えば，以下の 5 つの機能要求が定義されたとする．

・座席表を画面上に表示する(以下 R1)
・座席表を印刷する(以下 R2)
・クラス名簿を取り込む(以下 R3)
・決定した座席表を手動で入れ替えする(以下 R4)
・座席のレイアウトを設定する(以下 R5)

そして，要求者 A,B,C（以下，単に A,B,C）の 3 人が存在したとして，要求者 A が，機能要求 R1 からみて機能要求 R2 は「少し重要」であると感じた場合，表 1

よりその重要性尺度は3となる．この場合，R2からみてR1の重要度は3の逆数である1/3となる．

この一対比較をすべての機能要求の組で行う．その結果を表形式にしたものが表2であり，これを一対比較行列と呼ぶ．この場合，要求者は3人いるので一対比較行列は3つできることになる．

表2 要求者Aの一対比較行列

	R1	R2	R3	R4	R5
R1	1	3	1	3	5
R2	1/3	1	1/4	3	2
R3	1	4	1	6	5
R4	1/3	1/3	1/6	1	3
R5	1/5	1/2	1/5	1/3	1

全ての要求者の一対比較行列が作成されたならば，AHPにより要求者それぞれにとっての優先度を計算することができる(表3)．以降，要求者iの機能要求jの優先度をRRP_{ij}と表記する．

表3 要求者にとっての機能要求優先度(RRP_{ij})

	R1	R2	R3	R4	R5
A	0.3187	0.1379	0.3962	0.0906	0.0566
B	0.3049	0.1362	0.2793	0.0692	0.2104
C	0.3502	0.2116	0.2822	0.0655	0.0904

4.2. 機能要求見積り

このステップは，開発者が機能要求をどのくらいの期間で実現できるかという見積り日数を決定する．この見積り日数はその機能要求を設計，実装，テストを行うために要する日数とする．そして本研究では，この見積り日数を機能要求に対する開発者側の優先度を計算するために用いる．Cost-Value Approachに倣い，機能要求に設定された見積り日数が少ないほど，開発者にとってその機能要求の優先度が高いとする．

- ET_j：機能要求jの見積り日数
- TET：$1/ET_j$の総和

上記のように変数を定義したとき，開発者側の機能要求jの優先度DRP_jを以下の式で算出する．

$$DRP_j = \frac{1}{ET_j} \Big/ \frac{1}{TET} \quad \cdots (1)$$

例として開発者が機能要求R1からR5に対して表4のような見積りを入力したとする．

このとき，開発者側の優先度は式(1)を利用すると表5のようになる．

表4 機能要求の見積もり(日)

	R1	R2	R3	R4	R5
ET_j	10	5	10	15	10

表5 開発者側優先度

	R1	R2	R3	R4	R5
DRP_j	0.1765	0.3529	0.1765	0.1176	0.1765

4.3. 機能要求優先度決定

このステップでは，要求者が機能要求一対比較時に，開発者が機能要求見積り時に設定したそれぞれの値から，要求者と開発者が同意できる機能要求の全体的な優先度を決定する．そのためにまず，要求者側として機能要求優先度を一つにまとめなければならない．そのために，どの要求者の結果を重要視するかを決定しなければならない．そこで本研究では，このソフトウェア協調開発に対する貢献度が高い要求者の結果を重要視することにした．この貢献度を評価基準として採用することにより，開発により多く携わった要求者の意見をより採用できる形となり，ソフトウェア協調開発コミュニティにおける要求調整手法として適当であると考える．

- TC：議論における要求者全体の発言総数
- OC_i：議論における要求者iの発言数
- TR：機能要求の総数
- OR_i：要求者iが提案した機能要求数
- cw：議論と機能要求のどちらを重要視するかの重み．但し，$0 < cw < 1$

上記のように変数を定義したとき，要求者iの貢献度であるCV_iを以下の式で算出する．

$$CV_i = cw\left(\frac{OC_i}{TC}\right) + (1-cw)\left(\frac{OR_i}{TR}\right) \quad \cdots (2)$$

但し，$\sum CV_i = 1$

式(2)を用いて要求者の貢献度を算出したら，それをAHPの評価基準の重みに設定し，AHPを利用して優先度を計算することで，要求者全体の優先度が算出される．要求者全体の機能要求jの優先度をRRP_jは，式として以下のように表すことができる．

$$RRP_j = \sum_i (CV_i * RRP_{ij}) \quad \cdots (3)$$

これにより，要求者，開発者それぞれにとって各機

能要求の優先度が決定することになる．後は，機能要求ごとに優先度を足し合わせ，総和が1になるように正規化する．

以上のことを具体例を用いて説明する．コミュニティウェア上で要求者A,B,Cが表6のような活動をしたとする．

表6 要求者の活動

要求者	議論における発言数 (OC)	機能要求作成数 (OR)
A	27	4
B	36	1
C	7	0

すると，TRは5，TCは70となる．なおcwを0.5と定め，式(2)に当てはめると以下のようにそれぞれの貢献度が求まる．

$CV_A = 0.5*(27/70)+0.5*(4/5) = 0.5926$

$CV_B = 0.5*(36/70)+0.5*(1/5) = 0.3571$

$CV_C = 0.5*(7/70)+0.5*(0/5) = 0.05$

この貢献度と表3の一対比較結果を式(3)に代入することで，要求者全体の機能要求優先度が算出される(表7参照)．

表7 要求者の機能要求優先度

R1	R2	R3	R4	R5
0.3150	0.1409	0.3487	0.0817	0.1948

表5，表7より要求者，開発者それぞれにとって各機能要求の優先度が決定した．後は，機能要求ごとに優先度を足し合わせ，総和が1になるように正規化する．以下表8に最終結果を示す．この最終結果が不特定多数の要求者にとって価値が高く，開発者にとって開発コストが低いという観点での機能要求優先度である．

表8 最終的な優先度

順位	機能要求	優先度
1	クラス名簿を取り込む(R3)	0.2524
2	座席表を印刷する(R2)	0.2373
3	座席表を画面上に表示する(R1)	0.2361
4	座席のレイアウトを設定する(R5)	0.1784
5	決定した座席表を入れ替える(R4)	0.0958

4.4. グルーピング

このステップでは，前節で述べた機能要求優先度の情報をもとに，機能要求をいくつかのグループに分割する．この分割されたグループが漸進的開発プロセスの1サイクルで実現されるべき機能要求の単位である．したがって優先度の高い機能要求から順番に1サイクルの日数で実現できる範囲内でグルーピングしていけばよいことになる．

しかしながら，前述したようにAHPでの優先度決定プロセスでは，機能要求の依存関係を考慮していない．そこで本研究では，ユースケース記述における事前条件ユースケースの情報を利用する．具体的には，あるユースケースがユースケース記述における事前ユースケースが含まれるグループよりも優先度が低いグループに決定された場合，そのユースケースは事前ユースケースと同じグループ，もしくはより優先度の低いグループになるように調節する．

開発者は1サイクルの日数をシステムに入力すれば，システムはこのグルーピングを自動で行う．そして，システムによって出力されたグルーピングを吟味し，1サイクルの日数を調節することが可能である．

以下にシステムが行うグルーピングの具体的な流れを示す．

1. まず，1サイクルの日数（ここでは21日）を超えないように機能要求を優先度の高い順にグルーピングする（表9）．

表9 グルーピング1

グループ	機能要求	日数
1	クラス名簿を取り込む(R3)	10
	座席表を印刷する(R2)	5
2	座席表を画面上に表示する(R1)	10
3	座席のレイアウトを変更する(R5)	10
4	決定した座席表を入れ替える(R4)	15

2. グループ1に属している「座席表を印刷する」は事前ユースケースとして「座席表を画面上に表示する」が指定してあるとすると，システムは「座席表を印刷する」を「座席表を画面上に表示する」と同じグループにする(表10)．

表10 グルーピング2

グループ	機能要求	日数
1	クラス名簿を取り込む(R3)	10
2	座席表を印刷する(R2)	5
	座席表を画面上に表示する(R1)	10
3	座席のレイアウトを変更する(R5)	10
4	決定した座席表を入れ替える(R4)	15

今回の例では，これで条件を満たしたので表10の結果が出力されるが，もし，グループに属している機能要求の実現見積もりの和が1サイクルの日数を超えてしまった場合は，そのグループの中で一番優先度の低い機能要求を一段下のグループに移動する．

また，今回は機能要求が必須機能的なものが多かったため優先度とは異なるグルーピングがなされたが，定義されている機能要求がオプション的なものの場合，優先度情報と，機能要求間の依存関係の両方を踏

まえたグルーピングがなされると考えられる．
リリース計画が終了すると，開発者はリリース計画に従い実装を開始する．

これにより，複数の要求者と開発者の要望を踏まえつつ，実際のソフトウェア開発においても無理が生じないリリース計画を作成することができる．

5. 協調開発コミュニティウェアの開発

本節では，我々が今まで述べてきた要求抽出手法とリリース計画作成手法を実現したソフトウェア協調開発コミュニティウェアについて述べる．このコミュニティウェアは，UI 作成部分を担当する Java アプリケーションと，それ以外を担当する Web アプリケーションとで構成されており，両アプリケーションとも Java 言語で実装した．

以降の節で実装したコミュニティウェアを利用したソフトウェア協調開発の流れを説明していく．

5.1. UI 指向機能要求決定プロセスの実現

要求者はまず，Web アプリケーション上で要求イメージを記述する（要求イメージ作成）．この要求イメージの記述なしでは，ソフトウェアの協調開発は始まらない．この要求イメージは，あらかじめ他の要求者によって記述されたものに追記することが可能である．

次に要求者と開発者は記述された要求イメージからユースケースを抽出する（機能要求抽出）．記述された要求イメージを見ながらユースケースを抜き出してコミュニティウェアに登録していく．この機能要求抽出は要求者，開発者のどちらでも行うことが可能であるが，要求者が登録することにより，その要求者の貢献度が向上する．

次に要求者と開発者は開発するソフトウェアの画面設計を行う(UI 作成)．図 2 は UI 作成を支援する UI エディタである．

中央のフレームがソフトウェアのウィンドウを表しており，そのフレームに対してテキストエリアやチェックボックスなどのコンポーネントを配置するように作成していく(コンポーネント構造)．

配置が完了したら，次に各コンポーネントのシステムインタラクションを定義する．定義したいコンポーネントを選択状態にして，右下のプロパティウィンドウ内のシステムリアクションを選択する．するとシステムリアクション編集ウィンドウが表示される．このウィンドウには，選択されたコンポーネントごとに起こすことができるイベントの一覧が左側に表示されているので，定義したいイベントの右側にそのリアクションを記述する．図 2 の例だと，「押す」というイベントに対して，「座席表を作成する」というリアクションを定義している．また，コンポーネントの別名もプロパティウィンドウで編集することが可能である．

UI 作成が終了したら，そのデータをサーバー側に送信することになる．

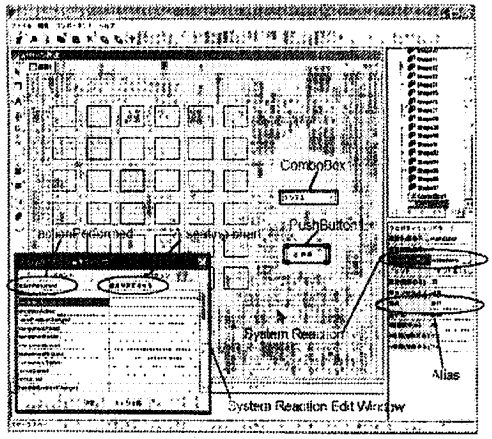

図 2 UI エディタ

最後に抽出したユースケースに対してユースケース記述を作成する（機能要求洗練）．ユースケース記述作成は Web アプリケーション上で行うことになる．本論文では特に基本系列作成の手順について説明する．

まず，ユースケースを実現するために操作するコンポーネントを選択する(図 3 参照)．図 3 の例だと，ComboBox1 というコンポーネントを選択している．

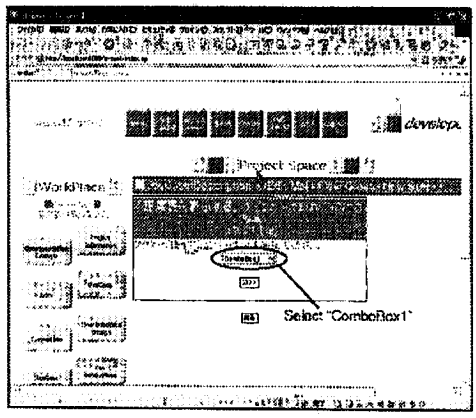

図 3 コンポーネント選択

次に ComboBox に対して起こすことができるイベントの一覧が表示されるので，その中からユースケースを実現するために起こす必要のあるイベントを選択する．するとシステムが 3.4 節で述べたように基本系列を作成する．

5.2. リリース計画作成プロセスの実現

要求者は機能要求仕様が固まったならば，それらの一対比較を行い，一対比較行列を作成する(機能要求一対比較)．図 4 にその画面を示す．

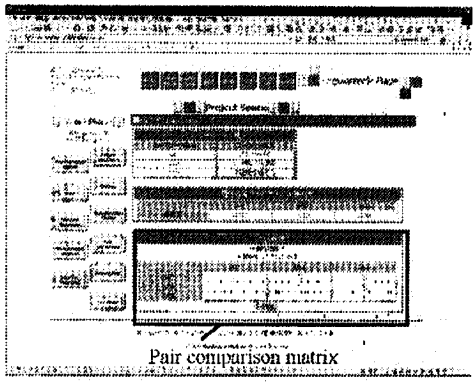

図4 機能要求一対比較

一対比較行列の要素部分をクリックすると重要性尺度の定義の選択画面が表示されるので，該当するものを選択することになる．一対比較をするごとにそのときの優先度がAHPにより計算され，画面上に表示される．このとき，AHPの一貫性尺度(CI)が0.15を超えると一般的に一対比較に矛盾が生じているといわれているので，CIが0.15を越えた場合にシステムはその旨を要求者に警告する．

同時に開発者は機能要求の見積りを入力する(機能要求見積もり)．

要求者による一対比較と開発者による見積りの入力が終了したならば，最後に開発者はリリース計画作成ボタンを押す．すると，システムは入力された情報をもとにリリース計画を画面上に出力する(機能要求優先度決定及びグルーピング)．

開発者は出力されたリリース計画(図5参照)にそってソフトウェアの分析，設計，実装を進めていくことになる．

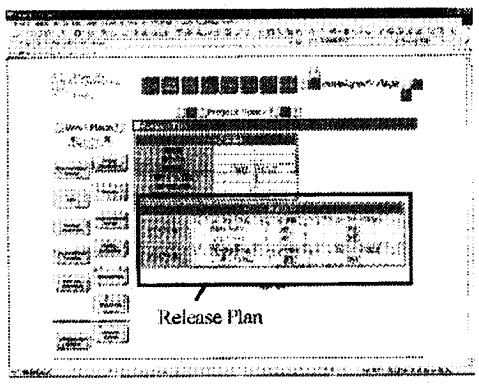

図5 リリース計画出力画面

6. まとめ

本研究では，協調ソフトウェア開発コミュニティにおいて，不特定多数の要求者と開発者間で，機能要求だけでなく，明確なUIの仕様の合意を得られるような要求抽出手法，そして複数の要求者と開発者の間での優先度を調整し，その優先度をもとにしたリリース計画作成手法の提案を行い，そのプロセスを実現するコミュニティウェアの開発を行った．

今後の課題として，本研究の適用，評価である．このUI指向開発プロセスと，協調ソフトウェア開発コミュニティウェアを実際に適用，評価して，その有効性を示さなければならない．

謝　辞

本研究の一部は，平成14年度文部科学省科学研究費補助金（基盤研究(C)(2)課題番号 14580209）の支援を受けている．ここに記して謝意を表す．

参　考　文　献

[1] J. Karlsson, and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements," IEEE Software, Vol.14, No.5, pp.67-74, September/October 1997.

[2] 古宮誠一，加藤潤三，永田守男，大西淳，佐伯元司，山本修一郎，蓬莱尚幸，"インタビューによる要求抽出作業を誘導するシステムの実現方法，" 情報処理学会研究報告ソフトウェア工学，No.121，pp.99-106，November 1998.

[3] 櫨山淳雄，石出勉，島影正俊，"小中学校と大学による協調ソフトウェア開発コミュニティウェア e-duet の提案，" 教育システム情報学会第26回全国大会，pp.201-202，August，2001.

[4] 松本吉弘，"IT時代に対応する大学のソフトウェアエンジニアリング教育，" 情報処理学会誌，Vol.42，No.1，pp.99-100，2001.

[5] T. L. Saaty, "The Analytic Hierarchy Process," International Symposium of the Analytic Hierarchy Process 1999, August 1999.

[6] M. Shimakage, A. Hazeyama, and T. Ishide, "Collaborative Software Development Communityware and Its Preliminary Application," Proceedings of the First International Conference on Knowledge Economy and Development of Science and Technology, 2003.

[7] http://www.omg.org/uml/