社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

既存システムのユースケース図を利用したステークホルダと非機能要求の獲得法

海谷　治彦† 　長田　晃† 　海尻　賢二†

† 信州大学 工学部 情報工学科　〒 380-8553 長野市 若里 4-17-1
E-mail: †{kaiya,csada,kaijiri}@cs.shinshu-u.ac.jp

あらまし　同一分野の既存システムのユースケース図の比較を通して，利害関係者や機能に関する品質要求等を識別する手法を提案する．本手法では，最初にユースケース図間の差異を特徴付けている変数を識別する．次に，そのような変数の変化を意図的に想定することで，影響を受ける存在を識別し，ステークホルダとする．加えて，その変化をステークホルダがどのように評価しているかをもとに非機能要求を識別する．
キーワード　非機能要求，GQM, ステークホルダ，ユースケース図

Requirements Elicitation Method about Stakeholders' Preferences and NFR by Using Use Case Diagrams

Haruhiko KAIYA†, Akira OSADA†, and Kenji KAIJIRI†

† Faculty of Engineering, Shinshu University　4-17-1 Wakasato, Nagano City, 380-8553 Japan
E-mail: †{kaiya,csada,kaijiri}@cs.shinshu-u.ac.jp

Abstract　We present a method to identify stakeholders and their preferences about non-functional requirements (NFR) by using use case diagrams of existing systems. We focus on the changes about NFR because such changes help stakeholders to identify their preferences. Comparing different use case diagrams of the same domain helps us to find the changes that can occur. In order to illustrate and evaluate our method, we applied our method to an application domain of the Mail User Agent (MUA) system.
Key words　Non-Functional Requirements (NFR), Goal Question Metrics (GQM), Stakeholders, Use Case Diagrams

1. Introduction

あるシステムの要求獲得を行う際，類似した既存システムの内容が役立つ．本稿では，そのような要求獲得において，複数の既存システムを比較することで，識別していなかったステークホルダ，非機能要求 (NFR) そしてステークホルダの好感度 (preference) を識別する手法を提案する．

ステークホルダとはシステム導入によって利害を得る者を指すため [1]，ユーザーやアクターだけではない [2]．ステークホルダとその好感度を識別することは重要である．なぜなら，それらは非機能要求や品質要求に大きく関係しているからである．例えば，同じ機能を提供するにしても，それを「安全に」提供するのか，「迅速に」提供するのかの違いはステークホルダが期待する利害によって決まってくる．

一般に，ある非機能要求や品質がステークホルダにとって利益のあるものか否かが即座に判断しにくい場合がある．しかし，その非機能や品質を変化させると，ステークホルダは自分の利害を識別しやすいと思われる．このような変化は Goal-Question-Metrics (GQM) [3] における Metrics を識別し，それを変化させることで，より系統的に扱うことができる．

GQM での Metrics の識別は，その名の通り，Goal, Question, Metrics の順にトップダウンに行う．しかし，実際に Goal から Metrics を識別するのは経験上，それほど容易ではない．そこで，本研究では，類似した既存システムの比較を通して Metrics に相当するものを識別することにする．

類似システムの表現にはユースケース図を利用する．第一の理由はソフトウェア要求記述で最も標準的に表現形式だからである．第二の理由はユースケース図にはアクターの概念があり，これはステークホルダの第一次近似となるからである．第三の理由は既存システムのユースケース図を記述するのに役立つ手法がリバースエンジニアリングの分野でいくつか提案されているからである [4], [5]．

次節では，本手法に利用する用語や概念を説明したあと，その手順を紹介する．そして，3. 節において，その適用例を通し

て本手法の評価を述べる．最後にまとめと今後の展望を述べる．

2. 獲　得　法

2.1 用語と概念

獲得法の手順紹介の前に，本方法で用いる用語と概念をここで整理しておく．

- ユースケース図とユースケースは UML での定義に順ずる．
- システムインタラクションとユーザーゴールは UML の解説書 [6] に紹介がある．これらは，ユースケースにおいて機能を表現するには二つの異なった様式である．ユースケースをシステムインタラクション様式で記述する場合，システムとアクターとの相互作用が記述される．一方，ユーザーゴール様式で記述する場合，そのユースケースの機能によって達成されるゴールが記述される．本手法では既存システムのユースケースを利用するため，システムインタラクション様式でユースケースを記述する．すでにシステムは存在し，場合によってはそれを実行することが可能なため，この様式でユースケースを表現することは容易である．
- アクターやユースケースの類似性は主観的に判断するしかない．しかし，アクターやユースケースの名前，および仕様に出現する語彙をもとにその類似性を判断することができる．
- アプリケーションの類似性も原則として主観的に判断するしかない．しかし，アクターやユースケースの類似性や類似した数をもとに，ある程度，アプリケーション間の類似性を判断することができる．また，店頭やカタログ，ウエブサイト等にある既存の分類も参考になる．類似したアプリケーションの集合を本稿ではアプリケーションドメインと呼ぶことにする．
- あるユースケースの周囲とはそのユースケースに extend や include の関係で接続されている他のユースケースと，関与しているアクターの集合である．
- ユースケース周囲の差異とは類似ユースケースの周囲同士の差異である．図 1 に例を示す．システム P と Q が類似アプリケーションとして，それぞれに類似するユースケースが存在する．P のユースケースの周囲は A, B, C, X, Y で，Q のユースケースの周囲は D, E, C, X, Y のため，差異は A, B, D, E となる．

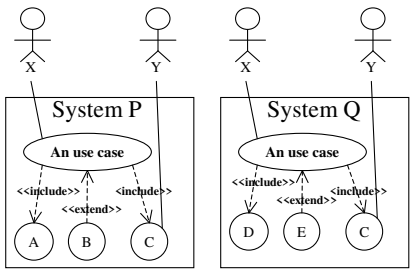

図 1　例: ユースケース周囲の差異

- **NFR** 分類とは非機能要求 (NFR) のカタログである．現在，ISO 標準であるソフトウェア品質特性 [7] と NFR 型 [8] をカタログとして利用している．NFR 分類は類似したユースケースの差異の理由を推測するために利用する．
- 変数は類似したユースケースやそれらの差異を特徴付けるために導入する．変数は名前を持ち型付けされている．例えば，列車制御システムに含まれるあるユースケースは「列車のスピード」という変数によって特徴づけられるため，その変数に vSpeed という名前を付け，自然数 (nat) で型付けする．

型は変数などのような範囲をどのように変化しうるかを示している．型付けされた変数の変化は本稿では以下のように表現する．

- C 言語や Java 等の代入のような形式．
- 以下に示すように古い値と変化後の値のペア．

$$var1 : old_value \rightarrow new_value$$

- 古い値が必要ない場合は以下のように表現する．

$$var1 : * \rightarrow new_value$$

- 単に文章で説明．例えば，

$var1$: 増加
$var2$: x は $var2$ のメンバーとなる.

現状では以下のような型を利用しており，それぞれの型は以下に示すような変化をすると想定している．

- nat: 自然数．以下の二種類の変化を想定．

$var++$ は var の増加，$var--$ は var の減少．

- $boolean$: 真偽値．
- set: 集合．列挙された値．$var2$ が型 $\{x, y, z, ...\}$ である場合，以下のように変化を表現．

$var2 = x$ は「$var2$ が 値 x になること」．

- $pset$: パワーセットを示す型．$var3$ が人の部分集合となる場合，「$var3 : pset\ of$ 人」と書き，以下にあるような変化を想定している．

$var3 = \{\}$ は「$var3$ が空集合となる」こと，
$var3 \ni Jone$ は「$Jone$ が $var3$ のメンバーとなること」．
尚，集合や自然数に適用される演算子 ($\cup, \subset$ or $+, -, \times$ 等) は適宜利用する．

- 変数のインバリアントは変数間の相関関係を表現するのに利用する．現在は以下の表現しか用いていない．

$x \sim y$: x が増減すれば y も増減する．

- ステークホルダ **(Stakeholder)** はシステムに利害関係のある者や物．
- 好感度 **(Preference)** はステークホルダの利益や被害を示す度数．現状では「嬉しい (利益がある)」と「困る (被害がある)」の区別のみをしている．

2.2 手　　順

前述の用語や概念を用いて獲得手法の手順を説明する．手順の入力は既存アプリケーションやシステムのマニュアルやヘルプ等の文書である．

Step 1: 複数の既存システムのユースケース図を記述するか得るかする．システムは同一もしくは類似アプリケーションドメインに属さなければならない．ユースケースはシステムイン

タラクション様式で表現されていなければならない．

Step 2: 上記の複数のユースケース図の中から，同一もしくは類似のユースケースを見つける．そのようなユースケースは記述したユースケース図全てに含まれる必要はない．

Step 3 Step 2 におけるそれぞれのユースケースについて，ユースケース周囲の差異を識別する．

Step 4 ユースケースの特性と差異を特徴付ける変数を識別する．変数の変化を考慮してその変数の型を決める．

Step 5 図 2 は本手順で利用・識別される要素間の関係をクラス図で表現したものである (いわゆるメタモデル)．この図をもとにステークホルダと好感度を識別する．

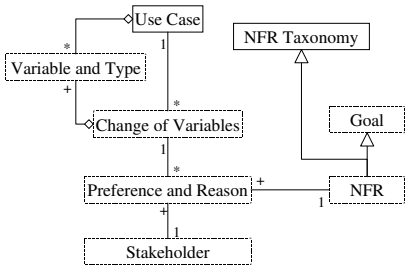

図 2　ステークホルダ，好感度そして NFR 間の関係を示すクラス図

図 2 中の実線で書かれたクラスのインスタンスはこの手順前に用意されるべき要素を示しており，破線で描かれたクラスのインスタンスがこの手順を通して作成されることになる．

(1)「変数の変化 (Change of variables)」のインスタンスを識別する．変数は Step 4 で型付けされているため，変化は系統的に識別できる．

(2)「変数の変化 (Change of variables)」に関係する「ステークホルダ (Stakeholder)」と「好感度とその理由 (Preference and Reason)」のインスタンスを識別する．直観的にいいかえれば，ある変数が変化することにより，あるユースケースの非機能要求が変わるため，ステークホルダの好感度がある理由で変化することを識別することになる．

これは主観的に識別するしかないが，ユースケース図や関係する変数を実際のステークホルダ候補に提示することにより，識別することが可能である．

(3)「好感度 (Preference)」を NFR として一般化する．この際に NFR 分類を参考にする．

図 2 に示すように，それぞれのユースケースは複数の「変数の変化 (change of variables)」が対応するため，Step 5 は，それぞれの変化について，繰り返し適用することになる．

Step 6: 図 2 のインスタンスであるオブジェクト図は複雑になるため，かわりに以下のような行列でインスタンスを表現する．

この行列は「変数の変化 (change of variables)」の個々のインスタンス毎に記述され，個々のステークホルダのある NFR に対する好感度をまとめている．

a change	NFR_1	NFR_2	...
$Stakeholder_1$	$pref_{11}$	$pref_{12}$	...
$Stakeholder_2$	$pref_{21}$	$pref_{22}$	...
⋮	⋮	⋮	⋮

行列のそれぞれのセルには「好感度とその理由 (Preference and reasons)」のインスタンスが入り，「注目している変化があった場合，ある $stakeholder_i$ が NFR_j を好きか否か 」を表現している．ステークホルダが当該の NFR に関心がなければセルは空でもよい．この表の具体例は次節で説明する．

結果として，アクターとはならないステークホルダ，初期には識別されていなかった NFR，そしてアクターの NFR に対する好感度を識別することができる．

3. 適　用　例

提案する獲得法の説明し評価するために，本節では Mail User Agent (MUA) システム (いわゆるメーラー) のドメインへの適用例を紹介する．MUA はそれほど大きなシステムではないが，現実的かつ今日最も重要なアプリケーションの 1 つであり，数多くの MUA が存在する．

ここでは以下に示す 4 つの MUA を獲得法への入力とした: Outlook Express, The RAND MH System [9], AL-Mail [10], Mutt [11]. 表 1 にそれぞれのシステムの概要を示す．

図 3 に MUA システム一般のユースケース図を示す．無論，本節で扱う 4 つの MUA システムもこの図に示す機能を有している．

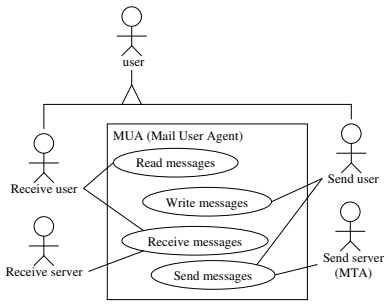

図 3　MUA システムの一般的なユースケース図

3.1　それぞれのシステムのユースケース図: Step 1

マニュアルおよびヘルプを参考にそれぞれのシステムのユースケース図を記述した．MH, Mutt および ALMail は著者ら自身が普段利用しているため，その経験から知識が補完された．システムとのインタラクションの側面からユースケース図を記述するため，この作業はそれほど難しいものではなかった．これは Step 1 の作業内容である．

これら 4 つのシステムは異なるプラットフォームやユーザー向けに設計されているため，ユースケースは多様性に富んでいた．これらの多様性は本節の以降にて部分的に参照する．それぞれのシステムのユースケース図におけるアクターとユースケースの数を表 2 に示す．

3.2　Step 2 から 6

Step 2 では，これら 4 つのユースケース図をもとに，類似したユースケースを可能な限り多く識別しなければならないが，紙面の都合，以降では 2 つのユースケース 'Receive messages', 'Read messages' を扱う．獲得法の中心は Step 3 から 6 であり，それらは各ユースケースの差異に適用される．

表 1　それぞれのシステムの概要

System Name	Version	Release	Main Platform	Abbreviation used in this Paper	User Interface Type	Note
Outlook Express	6.00	Oct. 2002	Windows	OE	GUI	Japanese Extension
RAND MH System	6.8.3	1993	UNIX	MH	Console, Commands	Japanese Extension
AL-Mail	1.13	Jun. 2002	Windows	ALMail	GUI	Domestic System
Mutt	1.5.4	Mar. 2003	UNIX	Mutt	Console, Interactive	Japanese Extension

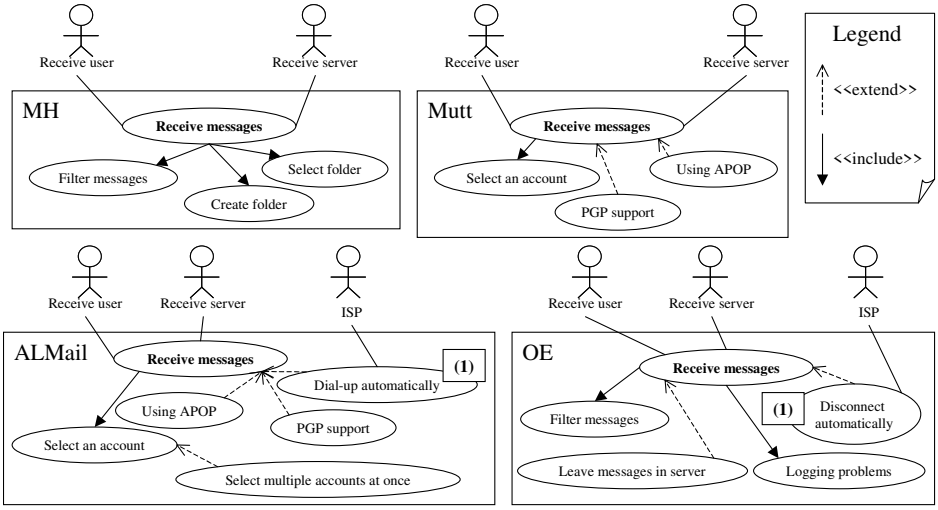

図 4　'Receive messages' 周囲のユースケースとアクター

表 2　それぞれのシステムの規模の概要

System	Number of use cases	Number of actors
OE	81	11
MH	38	6
ALMail	72	8
Mutt	53	7

3.2.1　ユースケース 'Receive messages'

図 4 は 4 つそれぞれのユースケース図におけるユースケース 'Receive messages' の周囲を示している．図を簡素化するため extend と include の関係は図中の凡例 (Legend) に示すように略記している．図 4 においても多数の差異を識別できるが，ここでは図中で (1) とマークされた差異を取り上げる．

(1) 回線の自動接続・切断

ここでの差異はメール受信の際，通信回線を自動接続したり自動切断したりする機能を特徴付けている．

Step 3: 周囲におけるユースケースの違いを識別．

ユースケース *'dial-up automatically'* は ALMail にあり，*'disconnect automatically'* は OE にあるが，他には該当するものはない．

step 4: 差異を特徴付ける変数を識別 (表 3)．

表 3　(1) における変数

Type	Name	意味とインバリアント
boolean	$vIsAuto$	自動的に接続・切断をするか否か
nat	$vNumCon$	接続回数
nat	$vNumPas$	パスワードが回線に流れる数．$vNumCon \sim vNumPas$

Step 5 と 6:

この差異については三種類の「変数の変化 'Change of variables'」に注目した．それぞれの変化について図 5, 6, 7 に示すような行列を記述し，ステークホルダと彼らの好感度を識別した．行列内のセルが狭いため，内容はセルの外部に記述してある．セル内の 'yes' は好感度が良い (嬉しい) ことを示し，'no' は悪い (困る) ことを示す．行列の左上のセルに「変数の変化」のインスタンスが埋め込まれるが，UML の表記にあわせるため，2. 1 節で紹介した表現形式とは若干異なる．このような行列を書くことで，総計 3 種類のステークホルダと，総計 3 種類の NFR を識別することができた (図中で太線で記述された部分)．

図 6 と 7 では二つの変数が変化を 1 つの変化として扱っている．この例に示すように複数の変数の変化を組にして扱ってよい．

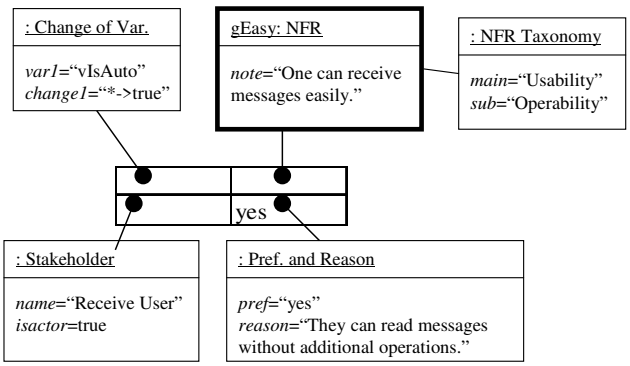

図 5　差異 (1) における変数 vIsAuto の変化について

3.2.2　ユースケース 'Read messages'

図 8 も 4 つのそれぞれのユースケース図におけるユースケー

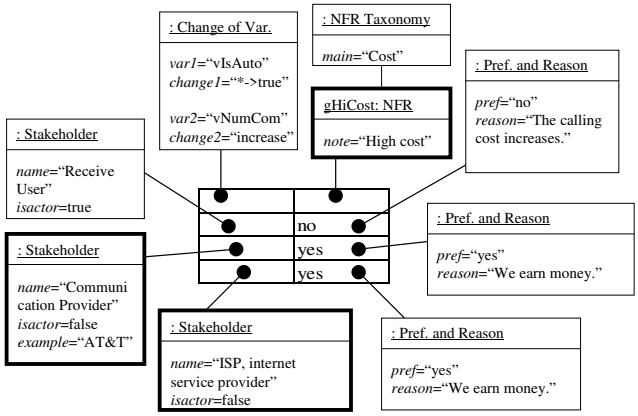

図 6　差異 (1) における vIsAuto と vNumCom の変化について

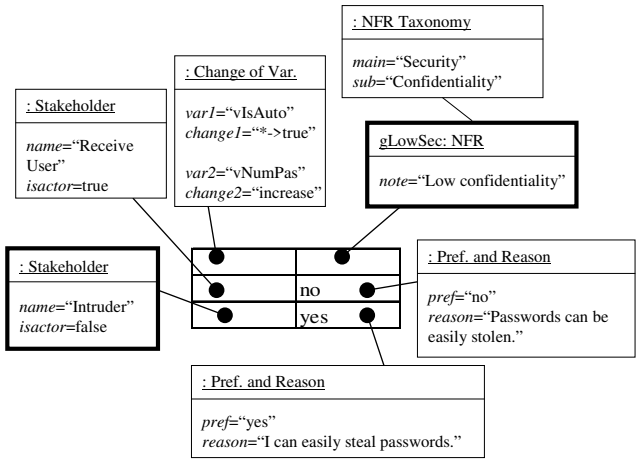

図 7　差異 (1) の vIsAuto と vNumPas の変化について

ス 'Read messages' の周囲を示している．ここでは (2) とマークされて差異をとりあげる．

(2) 外部プログラムの呼び出し

ここでの差異はメールにプログラムやハイパーリンク等を添付することで，メールからプログラムを実行する機能を特徴付けている．

Step 3: 周囲におけるユースケースの違いを識別．

ユースケース *'invoke external application'* は Mutt と ALMail にあり，ユースケース *'execute active contents'* は OE にある．OE にはユースケース *'control execution of active contents'* もある．ALMail にあるユースケース *'allow hyperlink(HTML)'* は同様の機能を提供する．

Step 4: 差異を特徴付ける変数を識別 (表 4)．

表 4　(2) における変数

Type	Name	意味とインバリアント
pset	$vApp$	アプリケーションの集合
boolean	$vAutoEx$	自動実行を許可するか否か

Step 5 と 6: この差異については三種類の「変数の変化 'Change of variables'」に注目した．それぞれの変化について図 9 と 10 に示すような行列を記述し，ステークホルダと彼らの好感度を識別した．このような行列を書くことで，総計 1 種類のステークホルダと，総計 4 種類の NFR を識別することができた (図中で太線で記述された部分)．

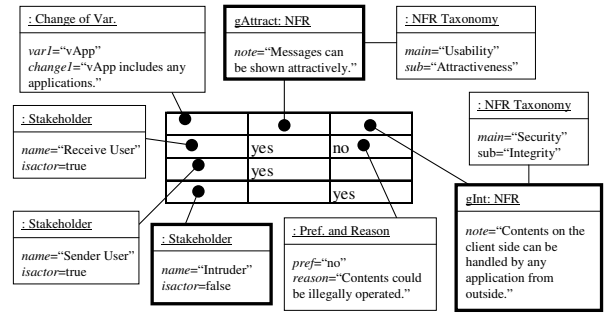

図 9　差異 (2) における変数 vApp の変化について

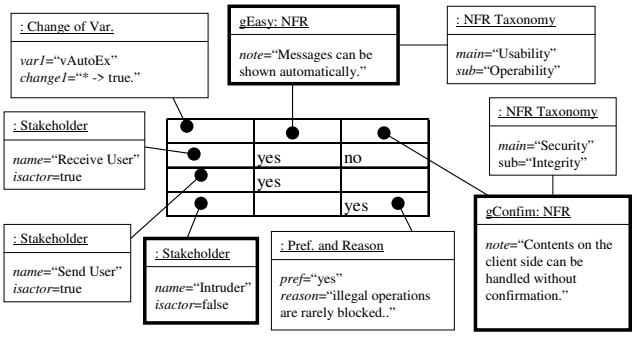

図 10　差異 (2) における変数 vAutoEx の変化について

3.3 評　　価

提案する獲得手法を利用してみることでいくつかの知見を得た．まず，通常の GQM を用いてはゴールを発見するのがそれほど容易でない．ここ手法では類似したユースケースの差異に着目することで，そのユースケースのゴールを識別しやすい．加えて，事前に用意した NFR 分類をもとに容易に一般化することができた．また，悪意のあるステークホルダ，例えば侵入者や競合者等の存在と好感度も変数の変化に着目することで発見しやすかった．

NFR は通常，複数の機能に横断的に関連している [12]．単にユースケース図を記述するだけでは，このような性質を識別することが難しい．しかし，本手法では変数を導入することで，より横断的な性質を識別しやすくなっている．例えば，(1) の $vIsAuto$ と (2) の $vAutoEx$ は同様の性質を指している．この点を考慮すれば一貫性のある設計を行うことが可能だと思われる．

4. おわりに

本稿では，認識されていないステークホルダと NFR，そしてある NFR の観点からの彼らの好感度を複数のユースケース図の比較を通して識別する手法を提案した．

WinWin アプローチ [13] では，ステークホルダ間でゴール間のトレードオフを模索しなければならない．我々の手法は変数を通して NFR に相当するゴールを特徴付けているため，こ

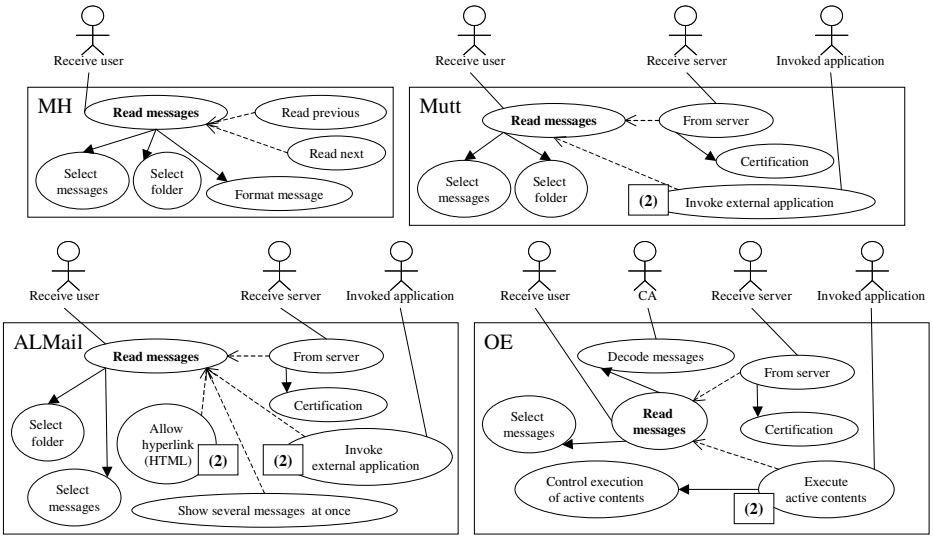

図 8　‘Read messages’ 周囲におけるユースケースとアクター

のようなトレードオフ模索に貢献するものと思われる．DDP 法 [14] では要求とリスクをなるだけ沢山識別しておくことが重要となる．我々の手法は予期していなかった状況を変数の変化によって意図的に作り出すため，数多くの要求やリスクの発見に貢献する．我々の手法はステークホルダ間の衝突を明確化する AGORA 法 [15] と組み合わせ使うことも有効だと思われる．

ここでの手法はパッケージ指向要求獲得法 PAORE [16] を基礎としている．PAORE では NFR やステークホルダ，そしてその好感度を扱っていなかった．また，表現としてユースケース図ではなく単純な機能分解を示す階層構造を用いていたため，アクターを自然に扱うことができなかった．結果として，ここでの手法は PAORE の弱点の一部を克服したことになる．

我々の手法を実適用するためには支援ツールは必須である．少なくともユースケース図のデータベースとそれをもとにしたユースケース図の比較機構は必須である．

現状ではユースケース内の内部構造については扱ってはいない．例えば，個々のユースケースをシナリオ等で記述するこにより，より詳細な差異を識別することが可能となるが，手法を適用するための手間も大きくなるため，この点のトレードオフを検討する必要がある．現状ではユースケース図の構造的特徴のみを扱うつもりである．

本手法では開発過程に関連する開発者等のステークホルダを扱いにくい．この点を扱うためには，開発プロセス自体のモデル化と比較が必要だと思われる．

GQM や品質標準を利用した要求獲得技法の研究はほとんど見られない .[17] では，GQM, QFD そして要求仕様書を品質管理に利用する試みが報告されている．品質標準を用いてソフトウェアパッケージの選択を支援する研究も見られる [18]．

文　献

[1] Linda A. Macaulay. *Requirements Engineering*. Applied Computing. Springer, 1996.

[2] Ian Alexander and Suzanne Robertson. Understanding Project Sociology by Modeling Stakeholders. *Software*, Vol. 21, No. 1, pp. 23–27, Jan./Feb. 2004.

[3] Victor R. Basili and David M. Weiss. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 6, pp. 728–738, Nov. 1984.

[4] Tarja Systa. Understanding the Behavior of Java Programs. In *Seventh Working Conference on Reverse Engineering*, pp. 214–223, 2000.

[5] Martin Glinz. A Lightweight Approach to Consistency of Scenarios and Class Models. In *4th International Conference on Requirements Engineering*, pp. 49–58, 2000.

[6] Martin Fowler and Kendall Scott. *UML Distilled, Applying the Standard Object Modeling Language*. Addison-Wesley, 1st edition, 1997.

[7] International Standard ISO/IEC 9126-1. Software engineering - Product quality - Part 1: Quality model, 2001.

[8] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[9] The RAND MH Message Handling System UCI version 6.8.3. `http://www.ics.uci.edu/%7Emh/`.

[10] AL-Mail32. `http://www.almail.com/`. Japanese page only.

[11] The Mutt E-Mail Client. `http://www.mutt.org/`.

[12] Motoshi Saeki and Haruhiko Kaiya. Transformation Based Approach for Weaving Use Case Models in Aspect-Oriented Requirements Analysis. In *The 4th AOSD Modeling With UML Workshop*, Oct. 2003. Joint workshop of UML 2003, `http://www.cs.iit.edu/~oaldawud/AOM/index.htm`.

[13] Barry Boehm, Paul Grunbacher, and Robert O. Briggs. Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software*, Vol. 18, No. 3, pp. 46–55, May/Jun. 2001.

[14] Steven L. Cornford, Martin S. Feather, John C. Kelly, Timothy W. Larson, Burton Sigal, and James D. Kiper. Design and Development Assessment. In *Proceedings of the Tenth International Workshop on Software Specification and Design (IWSSD'00)*, pp. 105–114, 2000.

[15] Haruhiko Kaiya, Hisayuki Horai, and Motoshi Saeki. AGORA: Attributed Goal-Oriented Requirements Analysis Method. In *IEEE Joint International Requirements Engineering Conference, RE'02*, pp. 13–22, Sep. 2002.

[16] Junzo Kato, Motoshi Saeki, Atsushi Ohnishi, Morio Nagata, Haruhiko Kaiya, Seiichi Komiya, Shuichiro Yamamoto, Hisayuki Horai, and Kenji Watahiki. PAORE: Package Oriented Requirements Elicitation. In *Proceedings of 10th Asia-Pacific Software Engineering Conference (APSEC 2003)*, pp. 17–26, Dec. 2003.

[17] Stanislaw Szejko. Requirements Driven Quality Control. In *COMPSAC'2002*, pp. 125–130, 2002.

[18] Xavier Franch and Juan Pablo Carvallo. Using Quality Models in Software Package Selection. *Software*, Vol. 20, No. 1, pp. 34–33, Jan./Feb. 2003.