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.

概念モデルからユースケースを抽出するための Traverser 拡張

中谷多哉子† 佐藤　雄朗†† 紫合　治†††

† 放送大学
†† 合同会社なのはなシステム
††† 東京電機大学

あらまし　ソフトウェアシステムが我々の日常生活に欠かせないものとなっている．したがって，要求工学においても，UX（User eXperience) を顧慮した要求抽出を議論する必要がある．**Traverser** は、ソフトウェアシステムの UX 要件を抽出するための手法として開発された．**Traverser** ツールは，クラス図で書かれた概念モデルのクラス間の関連および継承を辿りながら，web システムへの訪問者がサイトのページを渡り歩く経路を抽出する．この経路を本稿ではトラバースと呼ぶ．本研究では，このツールを拡張し，トラバースからユースケース記述の概要を自動的に抽出する機能を追加した．本稿では，概念モデルからユースケース記述を生成する仕組みを論ずると共に，カーシェアリングシステムを事例として用い，**Traverser** ツールの有効性を議論する．事例研究を行った結果，このツールが UX に関する要求を抽出するために貢献できる機能を提供していることを確認した．
キーワード　要求抽出支援ツール，概念モデル，ユースケース記述，UX

Extending Traverser for defining use cases from a conceptual model

Takako NAKATANI†, Kazuaki SATO††, and Osamu SHIGO†††

† The Open University of Japan
†† Nanohana Systems LLC.
††† Tokyo Denki University

Abstract　Software systems have become indispensable to our daily lives. Thus, we have to consider the importance of UX (User eXperience) in requirements engineering. **Traverser** is developed as a method and tool to extract UX requirements for software systems. **Traverser** tool extracts routes across pages in a web system by tracing relationships and inheritance between classes in a conceptual model. We call this route “traverse.” In our research, we added a function to the tool to automatically extract a summary of use case descriptions from the traverse. In this paper, we introduce the mechanism for generating use case descriptions from a conceptual model and also discuss the effectiveness of **Traverser** tool for defining UX requirements by an experimental case study of a car sharing system. As a result, we can conclude that the tool can provide effective functionality to engineers.
Key words　requirements elicitation tool, conceptual model, use case description, UX

1. ま え が き

要求は機能要求と非機能要求に分けられる．機能要求の定義には，ユースケース記述を用いることが多い．ユースケース記述は，アクターとシステムとの相互作用を表すために提案されたものである [1]．しかし，web システムのように，web サイトへの訪問者が，様々なリンクを辿りながら様々な目的を達成するためにシステムの機能を呼び出すといった，一連の機能の呼び出し列を表現するには適さない．なぜならば，ユースケース記述には，アクターとシステムの相互作用を，目的を達成するための処理手順として表現するが，web システムの訪問者の活動は処理手順というよりは，探索的なプロセスだからである．

では，探索的に行われるアクターの行動を要求として定義するためにはどうすればよいのか．訪問者の活動を，要求として定義するためには，訪問者にどのような活動が許されているかを明らかにする必要がある．そこで我々は，訪問者に許可された活動を明示するために，**Traverser** と名付けた手法とツールを開発した [2]．**Traverser** ツールは，UML のクラス図におけるロール名を，アクターのアクセス権限を定義するために流用している．これによって，アクターのアクセス権限を考慮しながら，かつ，クラス間の関連や継承構造を追跡しながら，アクターがどのような情報にアクセスできるかをオブジェクト列と

して生成する機能を実現できた．**Traverser** では，アクターのアクセス権限を付与したクラス図を概念モデルと言う．また，アクターがアクセスできるオブジェクト列を，トラバースと呼ぶ．トラバースは，web システムへの訪問者がサイトのページを渡り歩く経路と見なすこともできる．我々は，トラバースを訪問者のアクセス経路に関する要求して定義することで，目的の情報に効率的に訪問者を誘導することが可能となり，UX の向上に貢献できると考えている．

しかし，トラバースから仕様書に定義すべき要求を抽出するためには，いくつかの課題を解決しなければならない．第 1 に，アクターのアクセス権限を記述するための方法を定義しなければならない．第 2 に，ツールが概念モデル内のクラス間の関連を辿って情報を収集し，正当なトラバースを生成できなければならない．第 3 に，生成したトラバースを構成する機能から，ユースケース記述を生成する必要がある．これによって，UX のためのトラバースと，従来の機能定義の手法とを接続することができるようになり，実際のシステム開発のための機能要求が抽出できることになる．これまで著者らは，第 1，第 2 の問題を解決してきた [2]．本稿では，第 3 の課題解決の方策を述べる．

本稿の構成は以下の通りである．2. 節では，関連研究を紹介する．3. 節では，Traverser ツールの概要を紹介すると共に，アクセス権限の表記法とトラバースの可視化の方法を示し，ユースケース記述との接続について論ずる．4. 節では，Traverser ツールを UX を向上させるための要求抽出に適用した結果，様々な要求が抽出できただけでなく，概念モデルの誤りを発見し，修正できた事例を示す．また，実際にツールが出力したユースケース記述の概要を示すことで，第 3 の課題を解決できたことを示す．5. 節で本稿をまとめる．

2. 関 連 研 究

要求抽出に適用される手法には，ペルソナ分析 [3], [4] やシナリオ分析 [5]，ユーザストーリ [6]，そしてユースケース [1] などがある．T. Rietz らによる “Ladder Bot” [7] と名付けられた要求抽出ツールもある．これは一種のチャットボットで，様々な質問をしながら，要求を定義し，同時に要求者に要求の確認をするものである．このツールは，自然言語による対話が用いられているという点で，インタビュー支援の一つとみなすことができる．これらの手法が抽出する要求の源泉は，現実世界で行われるインタビューであったり，図面や書類といったものが多く，要求の根拠をツールを用いて辿ることは困難である．ただし，自然言語を用いるこれらの手法は，要求者が要求の妥当性を確認しやすいという点で，成果物の後ろ向き追跡可能性を補っている．自然言語による曖昧性を許容できるならば，要求仕様書の要求に自然言語を用いることは現実的である．**Traverser** ツールが出力するトラバースの表記にも，自然言語を用いている．

L. Rosenfeld らが提唱した設計規律は， 情報アーキテクチャと呼ばれている [8]．この規律によると，サービスや製品は，情報によって作られた場であると見なすことができ，したがって，この情報環境は，情報を見つけやすいように整えるべきであるとされている．先に紹介したペルソナ分析やユースケースは，いずれもサービスの使い方やサービスの種類を定義するための手法であるが，どのような情報の場を構成すべきかという視点はない．web システムのように，アクターが自由にサイト内を歩き回る場合，特に，アクターがシステム内で迷わないようにする配慮が必要である．我々は，「情報を見つけやすいように整える」ことは，情報を構造化すること，すなわち概念モデルを利用者が歩き回る場として構成することであると考えている．

M. Bates は，“Berrypicking” モデルを提案し，web システムのための使用性を向上させる目標を示している [9]．“Berrypicking” モデルとは，web システム内でのアクターの行動を森の中のイチゴ摘み行動というメタファで表したものである．しかし，アクターがサイト内を歩き回る過程では，アクターのアクセス権限にも配慮しなければならない [10]．我々が開発している **Traverser** ツールで取り扱う概念モデルは，UML のクラス図にアクター毎のアクセス権限を付与したものとなっている．

既存のサービスを更新したり，新しいサービスを作るとき，プロセスマイニングを適用する手法もある [11]．我々は訪問者の “Berrypicking” プロセスをプロセスマイニングして抽出するのではなく，開発の初期に作成した概念モデルに基づいて，自動的にアクターの探索プロセスを抽出する方法を選択した．

Traverser で用いる概念モデルは UML のクラス図を拡張したものであるが，関連に意味を持たせたオントロジーから要求を抽出する研究には長い歴史がある [12], [13], [14], [15]．後藤らはクラス図からシナリオを抽出し，アクターのイベントリストを生成する研究を行った [16]. 我々は，アクターのアクセス権限をクラス図に付与することで，セキュリティに考慮した要求抽出を行うことを試みている．

概念モデルを作成するのは，依然として技術者の課題であるとの主張がある [17]．A. Augusto はビジネスプロセスから概念モデルを生成することを試みている [18]．丸山らは，概念モデルを詳細化するためのツールを開発しており [19]，金田らは，クラス図を作成するためのガイドラインを開発した [20]. 妥当なクラス図を作成するための研究はこれからも行われるであろう．しかし，我々の研究では，クラス図が作れた後に，それをシステム開発にどう活かすかを議論すべきであるとの立場を取る．我々が開発した Traverser ツールは，作成した概念モデルの不備を発見することにも効果的であることを確認した．これは 4. 節で紹介する．

3. Traverser ツール

この節では，**Traverser** ツールの概要を紹介すると共に，アクセス権限の表記法とトラバースの可視化の方法を概観し，ユースケース記述との接続について論ずる．

3.1 Traverser 概要

Traverser は，UML のクラス図にアクターのアクセス権限を付与することで，アクターの “berrypicking” プロセスを抽出する手法である．この手法のツールは，以下の機能を持っている．

- クラス図の描画
- アクターのアクセス権限の定義

- トラバースの抽出
- トラバースに対する二種類の可視化
- トラバースから細粒度の機能要求をユースケース記述の概要を出力

クラス図の描画については，一般的なツールと同様であるので本稿では触れない．ツールを用いて描いた概念モデルを図 1 に示す．

3.2 アクターのアクセス権限の定義

アクターのアクセス権限は，オブジェクトの生成 (create)，参照 (read)，更新 (update)，削除（delete) に関する権限である．以降，これらの権限を CRUD 権限という．UML のクラス図では，クラス X とクラス Y の間に関連が定義されているとき，関連の両端には，ロール名と多重度を表記する．**Traverser** では，トラバースの生成に不要なロール名の代わりに，一方のクラスのインスタンスから，他方のクラスのインスタンスに対して，アクター別の CRUD 権限を定義する．**Traverser** ツールが，この権限情報を参照しながら，アクター毎のトラバースを生成するための情報を収集するためには，以下の条件が満たされる必要がある．

- 生成するトラバースの主体であるアクターの種別が明らかになっていること．

たとえば，一般ユーザなのか，登録会員なのか，管理者なのか，サポート担当オペレータなのか，などである．それぞれのアクターのグループによって，許可されている権限は異なるはずである．そこで，**Traverser** ツールでは，権限に基づくアクターの継承構造をアクターモデルとして定義する．

- *Actor*: *Actor* とは，web システムへの訪問者であり，概念モデルでは，クラス *Class* として定義されていること．

$$Actor \subset Class$$

- アクター毎の CRUD 権限が関連に定義されていること．

UML のクラス図でクラス間の関係を表す表記には，集約（aggregation）や合成（composition）のようにアクセスする方向が明示されているものと，関連（association）のようにアクセスの方向を明示しないものとがある．クラス X がクラス Y を集約/合成する場合は，関連の Y 端のみに，関連の場合は両端に，アクター毎のアクセス権限を書く．ＵＭＬのクラス図に用いられているロール名をアクセス権限として流用することによって，クラス図のエディタを大きく改変することなくツールを開発できる．権限 *Permission* は，create, read, update, delete を要素とする集合 *CRUD* を用いて，アクター毎に宣言する．

$$CRUD = \{create, read, update, delete\}$$

この *CRUD* 集合を用いて，権限 *Permission* を以下のように定義する．

Permission:

$$Permission \subset Actor \times 2^{CRUD}$$
$$perm \in 2^{Permission}$$
$$perm = (actor, crud)$$
$$actor \in Actor$$
$$crud \subset CRUD$$

すなわち， *actor* には *CRUD* の部分集合である *crud* に定義された権限が許可される．具体的には，アクター d*Person* に *r* 権限が与えられるとき，以下のように表記する．

$$Person\{\mathrm{r}\}$$

図 2 にアクセス権限を定義するためのダイアログボックスを示した．アクター間には，権限に基づく継承関係があるため，親クラスのアクターに R 権限が与えられると，自動的に子クラス達にも R 権限が与えられる．クラス図の関連の両端に各アクターの権限を表記することは，クラス図の紙面の制約によって選択できない．ツールの表記では省略表記を使っているが，関連を選べば，権限を確認したり編集することが可能である．

3.3 トラバースの抽出

トラバースを概念モデルから抽出するとき，ツールが参照できるのは，クラス間の関係 *Relation* が持つ 5 つの属性だけである．ここでは，関連クラス（assocClass）を持つ場合も考慮している．

$$rel \in Relation$$
$$rel = \{from, to, perms, mult, assocClass\}$$
$$from, to \in Class,$$
$$perms \in 2^{Permission}$$
$$mult \in Mult$$

Mult は UML のクラス図における多重度である．

$$assocClass \in Class \cup \{\bot\}$$

ここで，$2^{Permission}$ は，*Permission* のべき集合を表す. *rel* はクラス *Relation* のインスタンスであり，始点，終点，権限，多重度，関連クラスという５つの属性を持つ. これによって，ツールは，from 端から to 端（または to 端から from 端）に進めるか否かを行き先の端に定義されているアクターの権限に基づいて決定することが可能となる．したがって，トラバースを抽出するときは，以下の情報が必要である．

- トラバースするアクター
- トラバースの始点となるクラス
- トラバースするアクターに許可される CRUD 権限

始点の情報は，関係を指定することでツールに渡すことができる．アクターと権限は，トラバースを抽出するときのデータとして，ツールに入力する．

3.4 トラバースに対する二種類の可視化

関連研究で触れたように，様々な読者に理解してもらえる情報は自然言語で記述した方がよい．そこで，トラバースの可視化には，自然言語を用いることにした．図 3 に生成したトラバースの例を示す．この記述が UX を考慮するための要求記述の例であり，アクターに許可されるアクセス経路である．ただし，“クラス名.obj” は，そのクラスのインスタンスを表し，イタリックはクラス名を表す．トラバースは，クラス間の関係を辿りながらアクターがアクセスできるオブジェクトを探索していくことで得られる．

トラバースのもう一つの可視化の方法として図式表現を用いた．これによって，オブジェクトを探索しながらトラバースが構成される過程を視覚的に示せるようにした．図 4 に *Member* が *SearchVehicleKey* からトラバースを始め, *ActualLendingRecord*（貸し出し履歴）を得たときのトラバースを，アクターのスコープとして表現した例を示す．**Traverser** ツールでは，自然言語

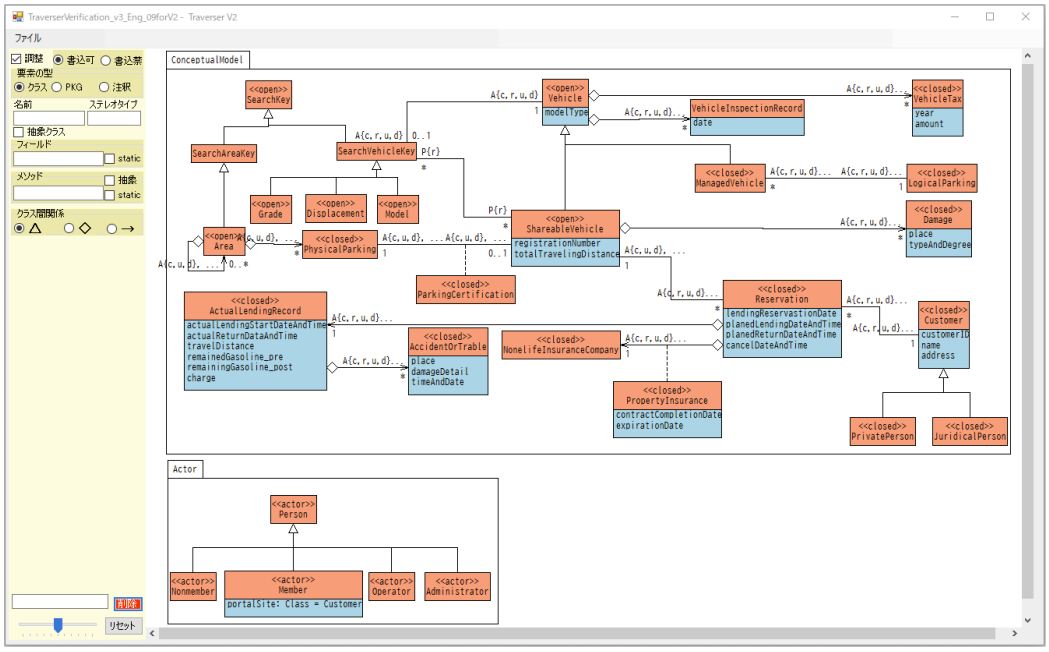

図 1　カーシェアリングシステムの概念モデル (部分).

Fig. 1　The conceptual model of a car-sharing system (part).

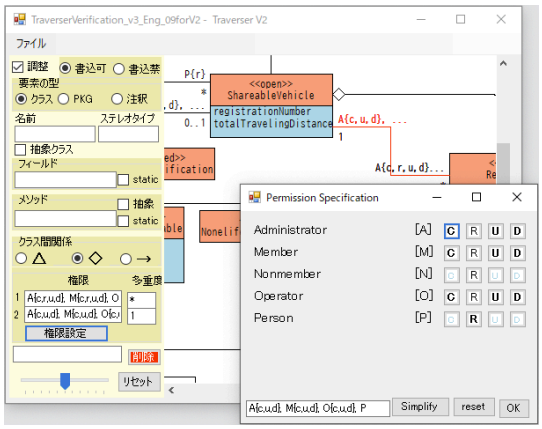

図 2　アクターのアクセス権限を定義するためのダイアログボックス.

Fig. 2　A dialog box to define the permissions for an actor.

アクター *Member* が *SearchVehicleKey* またはそこから継承を辿った *Model* または *Grade* または *Displacement* のいずれかのオブジェクト をアクセスしているときに、参照できるのは、*SearchVehicleKey.obj* から *ShareableVehicle.obj*、次に *ShareableVehicle.obj* から *Reservation.obj*、次に *Reservation.obj* から *ActualLendingRecord.obj* である。

図 3　自然言語によって表現されたトラバースの例：UX を考慮した要求記述

Fig. 3　An example of traverses in Japanese.

で表現したトラバースと共に，アクターが情報を探索する経路が，概念モデル上で順次視覚的に表示される．トラバースはアクターとアクターの権限に依存して抽出されるため，アクターと権限を指定する必要があることは既に述べた．図 4 には，アクターとアクターの権限を指定するためのダイアログボックスと，自然言語によって表現されたトラバースも示した．ここで

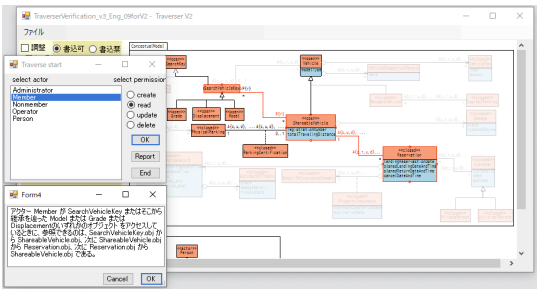

図 4　*Member* が *SearchVehicleKey* からトラバースを始めたときの，読み込み権限に着目したスコープ

Fig. 4　The scope of *Member*'s traverse with a *read* right from *SearchVehicleKey*

ハイライトされている部分が，トラバースに相当するアクター *Member* のスコープである．

UX に関する要求の確認をするとき，他のユーザにしかアクセスが許可されていないクラスは不要である．**Traverser** ツールは，アクター毎のスコープをハイライトすることで，レビューアに不要な情報を見えなくする．**Traverser** ツールは，ハイライトされた部分を取り出して，新たな概念モデルとして保存する機能も有している．図 5 に，*Member* のスコープとして取り出された概念モデルを示した．カーシェアリングシステムの場合，車というリソースの管理に必要なデータやオブジェクトもあるが，*Member* の UX を議論するときは，会員から何が参照可能であり，何ができるのかに焦点を絞って，会員に提供できる情報の確認をできるようにした方が良いであろう．

3.5　ユースケースへの接続

トラバースは，アクターと，アクターに許可された活動のペアの順序付き集合で表現される．アクターに許可された活動とは，CRUD を指す．また，順序付き集合の個々の要素の順序

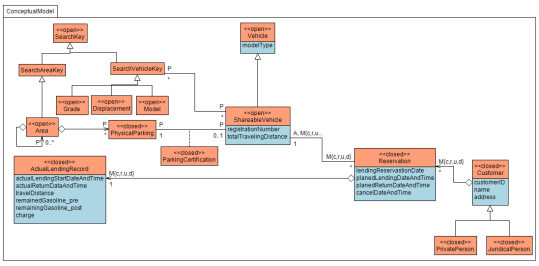

図 5　*Member* のスコープとして取り出された概念モデル

Fig. 5　The conceptual model of *Member*'s scope.

表 1　**Traverser** ツールが出力したユースケースの概要記述の例

Table 1　Examples of use case descriptions extracted from traverses

名前	*Customer* のアクセス
アクター	生成 または 参照 または 変更 または 削除を許された *Administrator*
事前条件	アクターは *Reservation* を保持していること
事後条件	アクターは *Customer* を新たに保持していること
名前	*Reservation* のアクセス
アクター	生成 または 参照 または 変更 または 削除を許された *Member*
事前条件	アクターは *Customer* を保持していること
事後条件	アクターは *Reservation* を新たに保持していること

は，たとえば，i 番目の行動をした後には，i+1 番目の行動が可能になることを意味している．したがって，順序付き集合の要素を参照することで，アクターに許可された活動の事前条件と事後条件を抽出することが可能となる．以上により，**Traverser** ツールは，順序付き集合で表現されたトラバースから，ユースケース記述のアクター，事前条件，事後条件を出力できるのである．その例を表 1 に示す．ここでは，「カーシェアサービスの予約情報から，予約者である会員顧客の情報を得る」機能要求と，「カーシェアサービスの会員が，自分の予約情報を得る」機能要求に関するユースケース記述である．

4.　事例による評価

この節では，Traverser ツールを事例に適用した結果，様々な要求が抽出できただけでなく，概念モデルの誤りを発見し，修正できた事例を示す．

4.1　事 例 概 要

カーシェアシステムのリソース管理とカーシェアの予約管理サービスに関する部分に着目し，概念モデルを構築した．この事例評価には，次の 2 名の技術者が参画した．

- UML モデラー：UML の技術支援を行っている技術者
- ドメイン専門化：カーシェアシステム開発を 10 年以上関わってきた技術者である．したがって，ドメインに関する知識と，システム開発に関する知識を有している．

ツールの有効性は，以下の手順で評価を行った．

(i)　ドメイン専門家の話を聞いて UML モデラーが概念モデルを構築する．（数日）

(ii)　我々がドメイン専門家に **Traverser** ツールの意図，使い方，および出力の見方を解説する．（約 30 分）

(iii)　ドメイン専門家が，**Traverser** ツールを用いてアクターの権限に基づくアクターモデルを作成し，クラス間の関係の端にアクターのアクセス権限を定義する．

(iv)　出力されたトラバースの内容に基づいて，ドメイン専門家が概念モデルの問題点をまとめる．（(iii) と合わせて約 8 時間）

(v)　ここまでで構築した概念モデルをドメイン専門家と UML モデラーがレビューをして，モデルの誤りを修正する．(約 2 時間）

(vi)　ドメイン専門家が，出力されたトラバースの内容に基づいて **Traverser** ツールの評価を行う．

Traverser ツールが出力したユースケースの概要記述は，48 個であった．ただし，一つのユースケース記述には，オブジェクトの生成，参照，変更，削除の操作が含まれている．ドメイン専門家に依頼した評価項目は以下の通りである．

- トラバースの二種類の可視化は概念モデルの理解に有効か．
- 抽出された要求は，これまでの仕様書に明記されていたか．もし，明記されていなかったとしたら，どのようにそれらの要求を定義していたか．
- **Traverser** ツールを今後の開発で使いたいか．

4.2　評 価 結 果

以下に，ドメイン専門家からの得た評価結果を示す．

- トラバースの二種類の可視化は概念モデルの理解に有効か．

アクター毎のスコープが図式表現されることで，誰に何が見えるのかを直観的に知ることができた．それによって，権限の定義の誤りを発見出来た．アクセス権限をアクター毎に確認していく必要があるため，アクターのスコープを絞り込めんで参照できる機能は役に立つ．このようなスコープが参照できるのであれば，自然言語で表現されたトラバースは不要かもしれない．しかし，他のツールで読み込んで仕様書を作ることを意図するのであれば，xml で出力できるようにしてもらいたい．ユースケース記述の概要も同様である．

- 抽出された要求は，これまでの仕様書に明記されていたか．もし，明記されていなかったとしたら，どのようにそれらの要求を定義していたか．

新規ビジネスのために開発では，利用者インタフェースのモックアップを作りながら発注者と共に開発を進めることが多い．しかし，利用者のシステム内での活動プロセスについて検討したことはない．そのため，保守コストが増大している．誰が，どの状況で，どのオブジェクトを参照するのかを把握できなくなっているシステムも少なからずある．アジャイル開発では，分かっているところから作ることが多いかも知れないが，把握している範囲を概念モデルに表すことで，システム開発の優先順位付けや開発方針を検討することも可能になるであろう．

- **Traverser** ツールを今後の開発で使いたいか．誰でも使えるかという点では，ツールを直観的に使えるようにする必要がある．たとえばメニュー構成など，ツールの UX を検討してもらいたい．しかし，アクター毎のスコープモデルや，ユースケースの概要記述など，**Traverser** ツールが出力する情報には満足している．ソフトウェアの開発者であれば，取り敢えずの概念モデルは書けるであろう．その後，トラバースを参照しながら直せるのではないか．このツールを今後の開発で使ってみたい．

評価の結果，**Traverser** ツールの有効性を確認することができた．しかし，被験者は一人である．より公正で客観的な評価結果を得るためには，他の技術者にも評価を依頼する必要がある．ツールは，現在も開発を続けている．出力データのＸＭＬ化や使用性の向上などの課題がある．特に，アクター毎に絞り込んだ概念モデルを保存できることで，アクター毎の UX の評価が可能となると考えているが，具体的に，どのように UX を評価するのかは今後の課題である．

5. ま と め

本稿では，web システムの UX を向上させるために，概念モデルにアクター毎のアクセス権限を付与し，自動的にトラバースを抽出するツールを開発した．事例を用いて評価を行った結果，ツールの有効性を確認した．ただし，評価者が１名である点など，より公平な評価を行う必要がある．今後は，他のツールとの相互接続性を満足させるために，出力ファイルの形式を検討すると共に，トラバースが UX の改善にどのように貢献できるかを検証する予定である．

謝辞　本研究を進めるにあたり，多くのヒントを頂いた情報処理学会ソフトウェア工学研究会の要求工学ワーキンググループの研究者の方々に感謝いたします．この研究は，本研究は科研費（19K11905）の助成を受けたものです．

文　献

[1] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, Object-Oriented Software Engineering, Addison-Wesley, 1992.

[2] T. Nakatani, H. Goto, T. Nakamura, and O. Shigo, “A method to generate traverse paths for eliciting missing requirements,” Proceedings of the Australasian Computer Science Week Multiconference, pp.54:1–54:10, ACSW 2019, ACM, New York, NY, USA, 2019. http://doi.acm.org/10.1145/3290688.3290697

[3] M. Aoyama, “Persona-and-scenario based requirements engineering for software embedded in digital consumer products,” the IEEE International Requirements Engineering Conference, pp.85–94, IEEE Computer Society, 2005.

[4] A. Cooper, “The origin of personas,” `https://www.cooper.com/journal/2003/08/the_origin_of_personas`, 2003.

[5] J.M. Carroll, Making Use: Scenario-Based Design of Human-Computer Interactions, MIT Press, 2000.

[6] D. Benyon and C. Macaulay, “A scenario-based design method for human-centered interaction design,” pp.211–235, John Wiley & Suns, 2004.

[7] T. Rietz and A. Maedche, “Ladderbot: A requirements self-elicitation system,” 2019 IEEE 27th International Requirements Engineering Conference (RE), pp.357–362, 2019.

[8] L. Rosenfeld, P. Morville, and J. Arango, Information Architecture, 4th Edition: For the Web and Beyond, O’Reilly, 2015.

[9] M.J. Bates, “The design of browsing and berrypicking techniques for the online search interface,” Online Review, vol.13, no.5, pp.407–424, 1989.

[10] H. Villamizar, M. Kalinowski, A. Garcia, and D. Mendez, “An efficient approach for reviewing security-related aspects in agile requirements specifications of web applications,” Requirements Engineering, vol.25, pp.439–468, Dec. 2020.

[11] W. v. d.Aalst, “Service mining: Using process mining to discover, check, and improve service behavior,” IEEE Transactions on Services Computing, vol.6, no.4, pp.525–535, 2013.

[12] R. Studer, V.R. Benjamins, and D. Fensel, “Knowledge engineering: Principles and methods,” Data & Knowledge Engineering, vol.25, no.1–2, pp.161–197, 1998.

[13] E. Insfrán, O. Pastor, and R. Wieringa, “Requirements engineering-based conceptual modeling,” Requirements Engineering, vol.7, no.2, pp.61–72, 2002.

[14] C. Rolland, “From conceptual modeling to requirements engineering,” Proceedings of the 25th International Conference on Conceptual Modeling, pp.5–11, ER’06, Springer-Verlag, Berlin, Heidelberg, 2006.

[15] S. Nalchigar, E. Yu, and R. Ramani, “A conceptual modeling framework for business analytics,” Conceptual Modeling, eds. by I. Comyn-Wattiau, K. Tanaka, I.-Y. Song, S. Yamamoto, and M. Saeki, pp.35–49, Springer International Publishing, Cham, 2016.

[16] K. Goto, S. Ogata, J. Shirogane, T. Nakatani, and Y. Fukazawa, “Support of scenario creation by generating event lists from conceptual models,” 2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp.376–383, Springer, 2015.

[17] K. Rosenthal, S. Strecker, and O. Pastor, “Modeling difficulties in data modeling,” Conceptual Modeling, eds. by G. Dobbie, U. Frank, G. Kappel, S.W. Liddle, and H.C. Mayr, pp.501–511, Springer International Publishing, Cham, 2020.

[18] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and G. Bruno, “Automated discovery of structured process models: Discover structured vs. discover and structure,” Conceptual Modeling, eds. by I. Comyn-Wattiau, K. Tanaka, I.-Y. Song, S. Yamamoto, and M. Saeki, pp.313–329, Springer International Publishing, Cham, 2016.

[19] M. Maruyama, S. Ogata, K. Okano, and M. Kayama, “Support tool for refining conceptual model in collaborative learning,” The proceeding of the Joint Conference on Knowledge-Based Software Engineering in Corfu (JCKBSE’18), pp.147–157, Springer, 2018.

[20] 金田重郎，井田明男，酒井孝真，熊谷聡志，“日本語仕様文からの概念モデリングガイドライン――行為文と関数従属性に基づくクラス図の作成――,” 電子情報通信学会論文誌 D，vol.J98-D，no.7，pp.1068–1082，2015.