情報処理学会論文誌 Vol.57 No.5 1399–1410 (May 2016)

エンティティの存在従属性に着目した機能規模の測定 ——アジャイルで網羅性の高い業務アプリケーションの見積り手法

井田 明男[1,a)] 金田 重郎[1] 熊谷 聡志[1] 矢野 寛将[1]

受付日 2015年8月15日, 採録日 2016年2月8日

概要：近年では，スコープを細分化して，小さなリリースを繰り返す開発スタイルが広がりをみせているため，見積りの頻度も高くなる傾向にある．ソフトウェアの機能規模の測定方法として国際規格の COSMIC 法がある．この方法は，認知された測定手法であるが，正確な測定のためには，すべての機能プロセスにおけるデータの移動を計測しなければならないため，利用者機能要求が機能プロセスを取り出せるほど詳細でない場合には適用が難しい．それに対して，業務アプリケーションの要求記述は，機能に関する記述の網羅性は概して高くない．なぜならば，要求記述は，何を管理したいかに主眼が置かれ，どのように管理するかについては，あえて捨象されるからである．そうであるなら，要求記述から直接的に機能プロセスを網羅的に抽出することはできないと考えるのが妥当であろう．そこで，本稿では，COSMIC 法をベースに，業務で扱うエンティティの存在従属性に着目した機能規模の測定法を提案する．要求記述から先にエンティティの存在従属グラフを作成し，そこから機能プロセスを抽出して測定を実施する．そのため，利用者機能要件の取りこぼしが少なく，正確な機能規模の測定が行えると期待される．確認のため，宿泊予約サイトの要求記述について，提案手法による測定結果と COSMIC 法による測定結果を比較した結果，それらの間には高い一致性が得られたため，提案手法は有効であると判断する．

キーワード：機能規模の測定，COSMIC 法，存在従属性，存在従属グラフ，生成制約，削除制約

Measuring Software Functional Size Based on Existence Dependency ——Agile and High Inclusion Characteristics Estimation for Business Application

Akio Ida[1,a)] Shigeo Kaneda[1] Satoshi Kumatani[1] Hiromasa Yano[1]

Received: August 15, 2015, Accepted: February 8, 2016

Abstract: In late years the frequency of the estimation tends to rise because we subdivide a development scope, and agile development-style to repeat small release. Method for measurement of the functional size of the software includes the COSMIC method of the international standard. This method is superior measurement technique, however, an application is difficult because we must measure the movement of the data in all functional processes for the accurate measurement when it is not detailed so that a user functional requirement can extract a functional process. In contrast, generally the demand description of the business application is not high in the inclusion characteristics of the description about the function. Therefore, it will be proper to think that we cannot extract a function process from a requirement description directly. Therefore, in this paper, we propose the measurement of the functional size that based on the existence dependency of entities and the COSMIC method. We construct the existence dependency graph of the entities from a requirements description earlier and we extract the functional processes from there and carry out the measurement. Therefore there is little defeat of the user functional requirement and is expected when it is possible for the measurement of an exact functional size. Because as a result of having compared the result of a measurement by the COSMIC method with the result of a measurement by the proposed technique about the requirement description of the room reservation site for inspection, high agreement characteristics were provided between them, we judge the proposed technique is effective.

Keywords: functional size measurement, COSMIC method, existence dependency, existence dependency graph, instance generation constraint, instance deletion constraint

1. はじめに

業務アプリケーション*1開発の現場において，開発対象の機能規模を簡単かつ正確に測定することへの期待が高まっている．なぜならば，これによって開発に要する労力をあらかじめ見積もることができ，ユースケース候補を費用対効果を考慮しながら取捨選択できるようになるからである．そうなれば，利用者と開発者の間で納得感のある開発契約を締結することができるようになる．さらに開発後にも機能的規模を基準にプロジェクトの生産性や欠陥密度などを反省することもでき，以降の開発をより工学的なものに改良していく判断材料にもなる．さらに，近年では，スコープを細分化して，小さなリリースを繰り返すスタイルが広まってきたため，見積りの頻度も高くなる傾向にある [1].

ソフトウェアの規模をどのように測るかについては，かねてから種々の議論が展開されている領域であり，いくつもの測定手法が提案されている．たとえばソースコードの行数で規模を測定するのは確実な方法であるが，実装が終わってからでないと測定できないため，見積りの用途には使えない．一方，あらかじめモデルを作成してソフトウェアの規模を測る方法としては，ファンクション・ポイント（IFPUG：International Function Point Users Group）法 [2]，COSMIC（COmmon Software Measurement International Consortium）法 [3] などが知られている．なかでも COSMIC 法はユースケース・モデルと親和性が高く，ISO で承認され JIS 標準にもなっている測定手法であるため，筆者らは機能規模を測定する必要があるときには COSMIC 法を用いている．

COSMIC 法は認知された測定手法であるが，その精度を保つためには，イベントフローレベルのユースケースの論理的な実現の分析を必要とする．しかしながら，実用的な観点からはもっと早期に，すなわち，ユースケース候補の中から開発対象をユースケースとして決定する時点で機能規模に関する情報が得られることが望まれる．

そこで，本稿では，COSMIC 法をベースに，業務アプリケーション専用として COSMIC 法を機敏かつ簡便に利用する測定手法を提案する．そのために，その業務ドメインで扱うエンティティの存在従属性に着目した．なぜならば，存在従属性はインスタンスのライフサイクルに基づく関係であるため，それらを扱う機能の時間的前後関係も半ば規定するからである．そのことを自然な形で取り入れた提案手法は，要求定義の早い段階で適用でき，測定も簡単かつ正確であり，その作業成果物も機能的規模の測定の用途だけでなく，機能要件の網羅性を確認にも優れたものである．

以下，2 章では，COSMIC 法の概要と課題を説明する．3 章では提案手法について説明する．4 章では，比較のためのケーススタディとして COSMIC 法と提案手法の計測結果を比較する．5 章はまとめである．

2. COSMIC 法

2.1 COSMIC 法の概要

COSMIC 法はソフトウェアの機能規模を測定する手法である．1997 年に Full Function Point（FFP）法（ver1.0）という名前でリアルタイム，技術，システムソフトウェアの機能規模を測定する手法として提案された．その後 2002 年に ISO で承認され，2006 年には JIS X0143 として JIS 標準となった手法である [3]．その特徴として特筆すべきは，COSMIC 法によって測定される機能規模は被測定ソフトウエアの実装形態に依存しないように設計されている [4] ことである．

2.2 COSMIC 法の測定要素

COSMIC 法では，**図 1** に示すように測定対象ソフトウェアについて，システム境界をまたがったデータ移動数と永続化記憶域に読み書きするデータ・グループの移動数の合計値を求めることで利用者の立場から見た機能の規模である機能規模を測定する [4]．システム境界の外側は対等システム（Peer System/Peer Component）と呼ばれる．また，永続化記憶域はファイルやデータベースなどが対応する．ただし，データ移動数といってもその回数をカウントするのではなく，その種類の数を数えることに注意しなければならない．読み書きするインスタンスの件数が増えても，ソフトウェアの行数が増えることはないためである．測定するデータ移動の種類は**表 1** に示す 4 種類である．移動するのは，データ・グループと呼ばれるデータのまとま

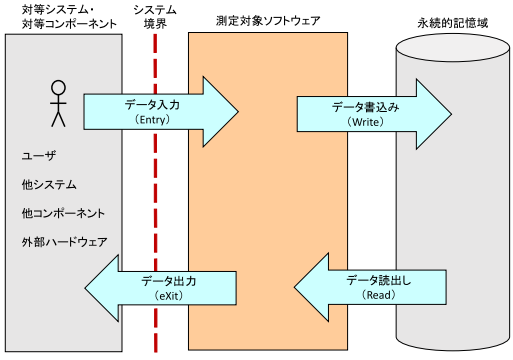

図 1 COSMIC 法の要素間のデータ移動モデル
Fig. 1 Data movement model in the COSMIC method.

*1 取引などに関する事実を効率的に捕捉，蓄積，管理するためのアプリケーションと緩やかに定義する．その特性上，データベース管理システムを核としたアーキテクチャである．

1 同志社大学大学院理工学研究科
Graduate School of Science and Engineering, Doshisha University, Kyotanabe, Kyoto 610–0394, Japan
a) ideaworks@kcn.jp

表 1 COSMIC 法におけるデータ移動の種類
Table 1 The types of data movement in the COSMIC method.

データ移動の種類	説明
エントリ （Entry）	システム境界外からユーザインタフェース（UI）や通信を通じて入力されることに対応するデータ移動
エクジット （eXit ）	UI や通信によりシステム境界外に出力されることに対応するデータ移動
リード （Read）	永続化記憶域の注目オブジェクト（エンティティ）から読み出されるデータ移動
ライト （Write）	永続化記憶域の注目オブジェクトに対して書き込まれるデータ移動

りで，論理データモデルのエンティティに相当する．データ移動の対象となるデータ・グループの種類数を機能プロセスと呼ばれる単位ごとに数えていくことによって，最終的にソフトウェア全体の機能規模を求めることができる．

2.3 COSMIC 法の測定手順

COSMIC 法の測定手順は以下のとおりである [4]．

2.3.1 Step 1：システムの境界と利用者機能要求を定義する

COSMIC 法では，測定対象ソフトウェアとデータの授受を行うシステム境界の外を対等システムと呼ぶ．対等システムは利用者であったり，他システムやハードウェアであってもよいため，ちょうど UML のアクタに対応する概念である．そして，利用者の立場から見たソフトウェアの機能のまとまりを利用者機能要求（Functional User Requirement）と呼ぶ．利用者機能要求は 1 つ以上の機能プロセスで構成され，UML のユースケースにほぼ相当する．このため，COSMIC 法はユースケースモデルと親和性が高い方法であるといえるだろう．

2.3.2 Step 2：利用者機能要求を構成する機能プロセスを明らかにする

機能プロセスとは，1 つのエントリをトリガとして実行される機能のまとまりである．たとえば，システム境界外からの顧客 ID のエントリをトリガとして，顧客エンティティを読み出し，その諸属性をエクジットする機能のまとまりは機能プロセスの例である．これは，ちょうどユースケース記述としてのイベントフローにおける，アクタの入力から，それに対するシステムの応答という対話の 1 往復に相当すると考えられる．ここで，顧客 ID の入力を 1 エントリ，顧客エンティティの読み出しを 1 リード，顧客の諸属性の表示を 1 エクジットとしてカウントするため，この機能プロセスは 3 種類のデータ・グループのデータ移動が観測される．したがって，データ移動に基づく機能規模値は 3CFP ということになる．なお，CFP（COSMIC Function Point）は測定される機能規模の単位である．

2.3.3 Step 3：利用者機能要求ごとの機能規模とソフトウェアの機能規模を求める

利用者機能要求ごとの機能規模は，利用者機能要求を構成するすべての機能プロセスの機能規模（CFP 値）を合計することで求められる．さらに，ソフトウェアの機能規模は，ソフトウェアが提供するすべての利用者機能要求の機能規模を合計することで求められる．

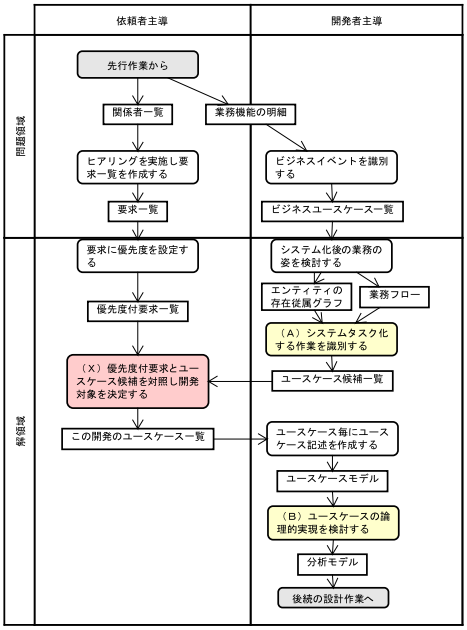

図 2 筆者らの要求定義プロセス
Fig. 2 The requirement process of the authors.

2.4 COSMIC 法適用の課題

筆者らは，COSMIC 法を適用するための課題は 2 つあると考えている．

2.4.1 要求定義プロセス上の測定時期に関する課題

1 つ目の課題は，機能規模を測定可能な要求定義プロセス上の時点に関するものである．**図 2** は，筆者らが検討している要求定義プロセスの部分である．業務システム開発のための要求定義プロセスであり，要求工学実践ガイド [5] を参照しながらまとめつつある．分析の早い時期にエンティティの存在従属グラフを作成して，ドメインについての理解を深めつつ，後続の作業で積極的に活用する意図がある．

筆者らは，機能規模の測定結果を図 2 中の（X）の作業で使いたいと望んでいる．なぜならば，（X）の作業で，費用感も考慮しながら今回の開発対象をユースケースとして決定したいからである．そのためには遅くとも（A）の作業時点で機能規模の測定が必要である．しかしながら，COSMIC 法で正確な測定を行うためには，図 2 中の（B）の作業時点まで待たなければならない．前述のとおり COSMIC 法では，アクタの入力からシステムの応答までを 1 つの機能プロセスとし，その中で行われるデータ移動を計測するためである．

2.4.2 要求記述の特性に関する課題

2 つ目の課題は，要求記述の特性に関するものである．**表 2** は，要求記述の例として一般的なものの 1 つである．一見，ビジネスイベントによって起動される業務プロセス

表 2 受注から出荷までの要求記述の例
Table 2 Example of requirement description from order to shipment.

システム化を検討している業務プロセス
受注プロセスは，顧客から注文が舞い込むと開始される．担当者は，顧客と商品を確認して，注文の内容を受注台帳に書込む．
出荷指示プロセスは，毎日15時になると開始される．担当者は受注台帳から未出荷の受注データを抽出しては，出荷指示済にステータスを更新し，出荷指示書を起票して出荷プロセスに渡す．このとき，出荷コストを抑えるため同一顧客への出荷は複数の受注明細をまとめて行うようにする．
出荷プロセスは，出荷指示書を受取ると開始される．担当者は，出荷指示書に記載された商品をピッキングし，納品書を作成・同封して梱包し，宛先ラベルを作成・貼付して宅配業者に引渡すとともに，出荷指示書のステータスを出荷済に更新して，出荷実績登録プロセスに渡す．ただし，システム化後も商品のピッキングは担当者が手作業で行う．
出荷実績登録プロセスは，出荷済の出荷指示書を受取ると開始される．担当者は，受注台帳の当該受注データのステータスを出荷済に更新するとともに，在庫台帳の当該商品の残高を更新する．

に即して記述されているが，COSMIC 法が要求する機能プロセスの観点からは，いたって粒度が粗く，網羅性も低いものである．たとえば，受注プロセスに関する記述だけをとってみても，顧客を確認する場面において，顧客が登録されていない場合の対処プロセスについては言及されていない．商品についても同様である．このことから，要求記述は，何を管理するかについてはある程度言及されているものの，どのように管理を実現するかについては捨象されていると考えらえる．そのため，これらの課題を克服するべく，筆者らは図 2 中の（A）の作業時点でもある程度正確な機能規模が測定できるような COSMIC 法の利用法を検討することにした．

3. 提案手法

3.1 提案手法の概要

これまで見てきたとおり，COSMIC 法が要求する入力モデルは，ユースケース記述としてのイベントフローレベル以上の詳細さを有する必要がある．すなわち，測定のためにはアクタとシステムの対話の詳細がモデル化されている必要がある．しかし，これでは筆者らが望むタイミングで機能規模の見積りを得ることはできない．

そこで，提案手法では，ユースケース・モデルではなく，業務についての要求記述から存在従属グラフを先に作成し，それを測定のための入力とする．説明を具体的に進めるために，引き続き表 2 に示した業務を例題として用いる．留意すべきは，要求記述は，あくまでもその作成者が業務の一部分だけをプロファイルした結果であるため，ストレート・フォワードに機能規模測定のための機能プロセスに展開することはできないことである．

3.2 提案手法の手順

3.2.1 Step 1：エンティティの存在従属グラフを作成する

エンティティとは，業務で管理すべき重要な概念である．エンティティの存在従属グラフは，主にエンティティ間の存在従属性に着目して作成する有向グラフである．

表 3 日本語要求記述中の品詞とクラス図の要素の対応
Table 3 Class diagram elements in the Japanese requirement description.

日本語要求記述中の品詞	クラス図の要素	例
可算名詞	リソース・クラスまたは属性	顧客，商品，名称など
質量名詞	属性値	数量，金額，色，サイズなど
動作動詞またはそれに由来する名詞	イベント・クラスまたは操作	受注，発注，契約など
状態動詞またはそれに由来する名詞	関連または関連クラス	在庫，提供，履行など

存在従属性の概念は Chen が用いている [6]．あるオブジェクトが，別のオブジェクトの先立つ存在を前提として存在しうるとき，前者のオブジェクトは後者のオブジェクトに存在従属するという．本稿では，文献 [7] の呼称にならって，前者のオブジェクトを**従属クラス**（dependent class）のオブジェクト，後者のオブジェクトを**親クラス**（parent class）のオブジェクトと呼ぶ．この存在従属という概念を用いることによって，たとえば，オークションサイトの入札は，出品に存在従属する．月々のローンの返済は，過去の購買という事象に存在従属する，などと表現できる．

存在従属グラフは UML クラス図の記法を流用しているため，静的な構造図に見えるが，実際はエンティティのインスタンスのライフサイクルに関する依存関係を表現するので筆者らは動的モデルとしても位置づけている．日本語要求記述文からのエンティティの識別については文献 [8] を，存在従属という概念，および存在従属グラフ作成の詳細については，文献 [9] を，それぞれ参照されたいが，ここではその概略を紹介する．

文献 [8] では，存在文が頻出する日本語要求記述を英文に近い行為文単文（動詞が 1 つの文）に変換してからクラス図に表記すべき要素を識別する．**表 3** は，その際に用いる記述中の品詞とクラス図の要素の対応づけの指針を示している．そして，クラスとする段階で帳票や台帳の名称は，たとえば，「受注台帳」は「受注」のように概念としての名称に置き換えたり，「納品書」のように「出荷」のビューにしかすぎないと判断されるものは除外したりする．

さて，クラス図に表記すべき要素が求まると，次にそれらの間の関係を検討していく．**表 4** は，識別されたクラス図の要素を存在従属グラフに組み立てる際のガイドラインを示している．文献 [9] によれば，先に単独で存在可能なクラス群を網羅してから，それらに従属，あるいはそれらを参照するクラスまたは属性を UML クラス図本来のモデル要素に加え，**表 5** に示すモデル要素を用いてグラフに表記していく．表 5 には，UML クラス図と重複する表記が登場するが，その表記の意味は，存在従属グラフとして作成する場合には，表 5 に記載したモデル要素の意味を優先させるものとする．たとえば，誘導可能性のある関連の表記は，存在従属グラフにおいては存在従属性を意味しており，コンポジションの表記は存在従属性でなおかつライフサイクルも一致するという属性的従属性を意味している．

図 3 左は前述のプロセスを経て，表 2 の例題について作

成したエンティティの存在従属グラフである．図 3 中の実線矢印（UML のモデル要素では，誘導可能性のある関連）は存在従属性を表す．矢印の元が従属クラス，矢印の先が親クラスを表す．また，破線矢印（UML のモデル要素では，依存関係）は単なる参照関係を表す．矢印の根元が参照するクラスで，矢印の先が参照されるクラスである．なお，図 3 左では，表 4 のガイドラインから導かれる ID（識別子）および，外部キーを明示的に記載している．従属クラスの ID は，必ず親クラスの ID をその一部に含む複合主キーとなっている．また，時点が帰属するイベント・クラスの場合は，そのクラスの ID にイベントの生成時点を表す時刻の情報を含めていることに注目していただきたい．

属性的従属性も含めた存在従属性と単なる参照関係の違いは，前者が，依存先（親）クラスのインスタンスを参照元（存在従属）クラスのインスタンス（存在従属クラスのインスタンス）が生成される時点でその識別子の一部として設定するため NULL が許容されず，また，設定後はライフサイクル途中で依存先を変更できないのに対して，後者は，参照先（参照される）クラスのインスタンスを参照元（参照する）クラスのインスタンスの非キー属性として保持するため，参照先のインスタンスが存在しない時点においては NULL が許容され，参照先を設定するタイミングは任意の時点でかまわない．また，参照元のインスタンスのライフサイクル中において，参照先のインスタンスを自由に変更することができる点である．

図 3 左は，受注のインスタンスが存在するためには，発注者である顧客のインスタンスがあらかじめ存在していなければならないこと，受注明細インスタンスは，その見出しとなる受注のインスタンスに属性的に従属し，注文対象である商品のインスタンスに存在従属すること，また，出荷明細のインスタンスは，受注明細のインスタンスに存在従属し，かつ，同一顧客への出荷明細をまとめて出荷を構成することを表現している．ここで，出荷明細と出荷の関係には，若干の注意が必要である．このケースでは明細が先に存在し，それらを見出しを付与して束ねる形であるた

表 4　存在従属グラフ作成時のガイドライン
Table 4　Guidelines for existence dependency graph creation.

項		ガイドライン
属性		あるデータ項目をあるクラスの属性とするには，その値がそのクラスのインスタンスに存在従属する場合に限る．ただし，一旦あるクラスの属性とされたデータ項目であっても，構造を持っていたり，データ項目に2回以上の繰り返しが認められる場合は，そのデータ項目を元のクラスから分離する．そして，もとのクラスと分離したデータ項目との間に存在従属性のライフサイクルが一致する形である属性的従属性を定義する．
独立クラス		そのインスタンスが，独立して存在可能なクラスのIDには「必ず」単一属性のIDを与える．
従属クラス		そのインスタンスが，独立して存在できない（存在従属な）クラスのIDには，「決して」単一属性のIDは与えず，「必ず」その存在根拠（前提）となるクラスのIDを含むIDを与える．結果，存在従属なクラスの識別子は複合主キーとなる．
イベント・クラス		そのインスタンスが，（リソースではなく）イベント（時点が帰属するインスタンス）の場合は，上記のルールに加えて，そのクラスのIDに「必ず」時点をあらわす時刻または版（バージョン）のシリアル値を含める．
その他		存在従属でないクラス間の関係は，汎化関係と単なる参照関係のみとする．
	i.	汎化関係の場合：サブクラスのIDは「必ず」スーパークラスと同じIDを共有する．たとえば，「プレミアム会員」のスーパークラスが「会員」クラスで，そのIDが「会員ID」であった場合は，「プレミアム会員」クラスのIDも「会員ID」となる．
	ii.	単なる参照関係の場合：参照元のインスタンスと参照先のインスタンスのそれぞれのライフサイクルは互いに独立している（制約を受けない）．このような場合は，表記では参照元のクラスから参照先のクラスへの依存関係を定義し，参照元は参照先のIDを外部キーとして保持するものとする．

表 5　存在従属グラフに表記する主なモデル要素
Table 5　Extra model elements in the existence dependency graph.

モデル要素	モデル要素の意味	表記
主キー属性	インスタンスを識別ための属性	ステレオタイプ<<pk>>を付与
外部キー属性	インスタンスを外部参照するため属性	ステレオタイプ<<fk>>を付与
主キー属性兼外部キー属性	インスタンスを識別ための属性がインスタンスを外部参照するため属性を兼ねている	ステレオタイプ<<pkfk>>を付与
存在従属性	あるインスタンスとその存在前提となるインスタンスの関係をクラスレベルに一般化したもの	親クラス ←—— 従属クラス
強い存在従属性（属性的従属性）	あるインスタンスとその属性値の関係をクラスレベルに一般化したもの	元のクラス ◆—— 属性的従属クラス
（単なる）参照関係	あるインスタンスが他のインスタンスを参照する関係．参照先の未定や変更があってもよい	参照先クラス <------ 参照元クラス
汎化関係	同じ主キー属性を共有するインスタンスのクラス間における特化したクラスとより一般的なクラスの間の関係	スーパークラス◁—— サブクラス

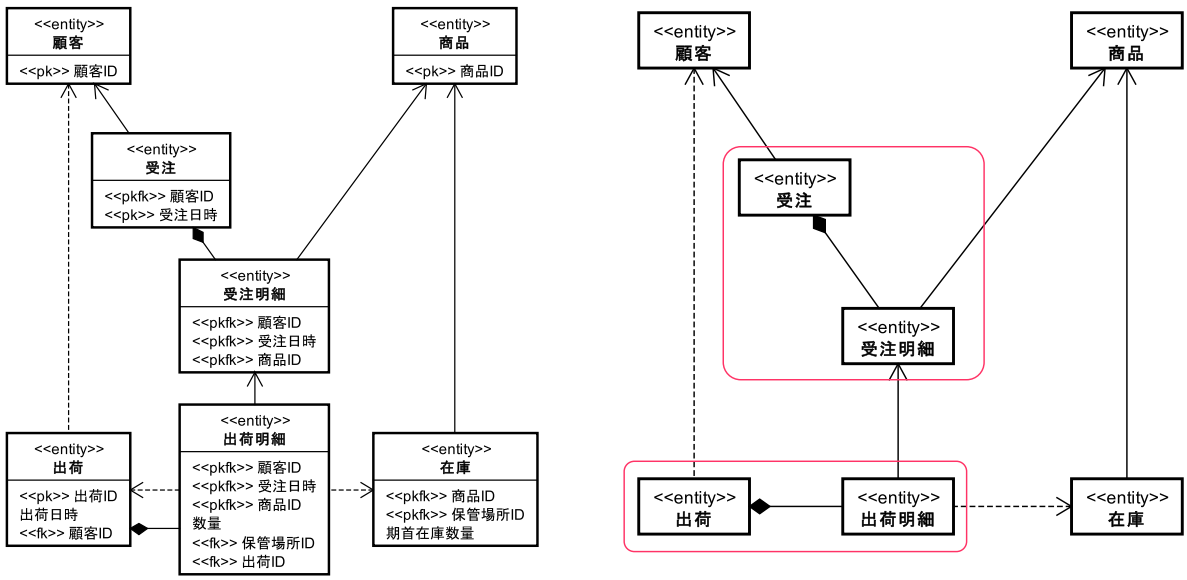

図 3　エンティティの存在従属グラフ（左側の図は識別子の関係を示すために記載）
Fig. 3　Existence dependency graph of domain entities.

め，筆者らは「明細—見出しパターン」と呼んでいる．しかしながら，概念としては出荷という 1 つの管理対象であることには違いがないため，通常の「見出し—明細パターン」と同様な属性的従属性ととらえることも可能である．図 3 左において出荷明細と出荷の間に 2 種類の依存関係が引いてあるのはそのためであるが，前述の理由により属性的従属性を優先してもよいだろう（図 3 右枠囲い部分）．さらに，出荷明細のインスタンスは商品に存在従属する在庫のインスタンスを参照すること，また出荷は出荷先として顧客のインスタンスを参照すること，といった業務上の制約やルールを表現している．なお，以降の測定では，提案手法は，COSMIC 法をベースにしているため，COSMIC 法でその移動をカウントすべきデータ・グループは，提案手法では，エンティティの全属性またはエンティティの一部の属性を摘み集めたものに相当し，COSMIC 法でいう注目オブジェクトは，永続化記憶域に格納される正規化されたデータであると定義されている [4] ため，提案手法ではエンティティそのものに相当すると見なす．そして，COSMIC 法では，エンティティの属性数は測定に影響しないため結局は図 3 右の存在従属グラフを用いることができる．

3.2.2 Step 2：存在従属グラフから利用者機能要件を求める

エンティティの存在従属グラフが得られたならば，それをもとに COSMIC 法でいう利用者機能要件を導出する．利用者機能要件とは，利用者の視点でソフトウェアの機能のまとまりをとらえたものであり，UML のユースケースとそのまま対応する．

図 3 には，明細エンティティを除くと 5 種類のエンティティが登場するため，次の 5 つの「大まかな利用者機能要件」が導かれる．これを提案手法では「ドメイン管理要件」と呼ぶことにする．ドメイン管理要件は原則，存在従属グラフに登場するエンティティの数だけ存在すると考えられる．エンティティが見出しと明細に分れている場合は，それらは概念としては 1 つであると見なす．したがって，例題におけるドメイン管理要件は，1) 利用者を管理する，2) 商品を管理する，3) 在庫を管理する，4) 受注を管理する，5) 出荷を管理する，の 5 つである．

そして，1 つのドメイン管理要件は暗黙裡に 4 つの利用者機能要件を含意する．すなわち，1) 管理対象のインスタンスを新規登録する，2) 管理対象のインスタンスを検索・同定する，3) 管理対象のインスタンスを更新する，そして，4) 管理対象のインスタンスを抹消する，の 4 つである．これらは，CRUD（Create，Refer，Update，Delete）処理と対応し，エンティティのインスタンスを管理していくために欠かせない基本的な機能である．例題の場合は，5 つのドメイン管理要件が識別されため，自動的に 20 の利用者機能要件が導かれる．**表 6** はこれらを一覧にした表である．

表 6 ドメイン管理要件を利用者機能要件に展開した表
Table 6 Expand the domain management requirements to the functional user requirements.

ドメイン管理要件	FUR#	利用者機能要件（≒ユースケース）
顧客を管理する	1.1	顧客を新規登録する
	1.2	顧客を検索・同定する
	1.3	顧客の属性を更新する
	1.4	顧客を抹消する
商品を管理する	2.1	商品を新規登録する
	2.2	商品を検索・同定する
	2.3	商品の属性を更新する
	2.4	商品を抹消する
受注を管理する	3.1	受注を新規登録する
	3.2	受注を検索・同定する
	3.3	受注の属性を更新する
	3.4	受注を抹消する
出荷を管理する	4.1	出荷を新規登録する
	4.2	出荷を検索・同定する
	4.3	出荷の属性を更新する
	4.4	出荷を抹消する
在庫を管理する	5.1	在庫を新規登録する
	5.2	在庫を検索・同定する
	5.3	在庫の属性を更新する
	5.4	在庫を抹消する

3.2.3 Step 3：利用者機能要件を機能プロセスに展開する

COSMIC 法でいう機能プロセスは，システム境界外からのエントリに基づいて実行されるソフトウェアの機能で，利用者機能要件を構成するものであり，必ず複数種類のデータ移動が続けて実行されるものであるとされる [4]．ここでいうデータ移動とは，2.2 節で言及した COSMIC 法の測定対象となる表 1 の 4 種類である．

機能プロセスも存在従属グラフからある程度は導くことが可能である．筆者らは，存在従属グラフからユースケースおよびユースケース記述を逆生成する提案を行っている [10]．この提案によれば，存在従属性は，エンティティのインスタンス間に次のような制約を与える．

生成制約

- 独立エンティティのインスタンスは，制約（前提条件）なしに生成し，新規登録することができる．
- 従属エンティティのインスタンスは，その親となる従属先のインスタンスなしでは，生成・新規登録することができない．したがって，従属先のインスタンスの存在チェックを行う処理（データ移動）が必要である．

削除制約

- あるインスタンスを削除する場合は，そのインスタンスに存在従属するか，あるいはそのインスタンスを参照しているインスタンスの存在チェックを行う処理が必要がある．存在する場合は，削除を中止する．ただし，業務要件によっては，存在従属または，参照しているインスタンスも含めて連鎖的に削除する場合もある．

上記制約から，例題の場合，図 3 が示すとおり，受注は，顧客と商品に存在従属するため，受注を新規登録するには，**表 7** に示す機能プロセスが必要であることが導かれる．受注を新規登録するためには，あらかじめ登録済みの顧客のインスタンスと商品のインスタンス群を新規の受注

のインスタンスを生成して結び付けなければならないためである．

存在従属グラフは，エンティティのインスタンスのライフサイクルに関する制約を表現しているため，存在従属グラフが得られれば，ドメイン管理要件，利用者機能要件，および，利用者機能要件を実現するための機能プロセスも自動的に導かれることは注目に値する．しかしながら，提案手法では，機能プロセスにまで詳細化を行う必要はない．その代わりに，あらかじめ利用者機能要件タイプごとにCFP値を計算した後述の**表8**の値を適用することができる．

3.2.4 Step 4：存在従属グラフから利用者要件ごとのCFP値を求める

提案手法では，存在従属グラフと利用者機能要件の一覧が得られれば，利用者機能要件ごとのCFP値は，表8を用いて求めることができるようにした．表8は，利用者機能要件のタイプ別に存在従属グラフに登場するエンティティの制約を考慮して筆者らがあらかじめCFP値を計算したものである．このように，あらかじめCFP値が計算できるのは，存在従属グラフには3.2.3項で述べた制約が表現されているからである．

ここで表8について若干の説明を行う．表8は，たとえば，図書館で利用者と図書を同定して貸出しを行うのも，通販サイトで会員と商品を同定して販売を行うのも機能規模に影響するデータ移動という観点からはまったく同じであるとの発想に基づいている．表8では，エンティティを単独で存在可能なタイプ，単独で存在できないタイプ，他のエンティティから依存や参照されている場合に分け，それらに対して，新規登録，検索・同定，更新，および削除を行うという抽象的な機能プロセスを想定し，それらにCOSMIC法を忠実に適用してそれぞれのCFP値を計算している．文献[4]には，受注を登録する場合の計測例が掲載されているが，そこでの典型的なデータ移動は次のとおりである．1) 顧客から舞い込んだ注文，すなわち「顧客」と「商品」群に関するデータ・グループの2つのエントリ．「受注」オブジェクトを記述するこれらのエントリのうち，最初のものは機能プロセスを開始するためのデータ移動でもある．2) 指定された「顧客」と「商品」が実在するかを確認するための「顧客」と「商品」に関するデータの2つのリード．3) 登録されたデータを記憶域に移動するための「受注」に関するデータの更新（1つのリードと1つのライト）．そして，4) 受注の合計や各「商品」に対応した倉庫への出荷指示などを含んでいる「受注確認」メッセージを含んだデータの1個以上のエクジット．合計7種類以上のデータ・グループの移動が計測されるとされ，これは，表8のFURタイプ#6のCFP値と一致している．

表8作成時に考慮したことは，手作業において使用しやすい表にすること，および重複計測をできるだけ排除することである．そのために，表4に示した存在従属グラフ作成時のガイドラインの情報を活用している．具体的に

表7 利用者機能要件を機能プロセスに展開した表
Table 7 Expand user functional requirement to the functional processes.

FUR#	利用者機能要件（≒ユースケース）	FP#	機能プロセス
3.1	受注を新規登録する	3.1.1	顧客を同定する
		3.1.2	商品と数量を同定する
		3.1.3	受注を登録する

表8 存在従属性から利用者機能要件タイプごとにあらかじめ計算で求めたCFP値の表
Table 8 CFP values determined by the pre-calculated for each user functional requirements type from the existence dependency.

FURタイプ#	利用者機能要件(FUR)タイプ	Entry	Read	Write	eXit	CFP値
1	あるエンティティのインスタンスを検索・同定する	同定するためのIDまたはキーワードを入力する	そのエンティティの該当するインスタンスを検索する	なし	同定に成功したインスタンスの諸属性を表示する 同定に失敗したメッセージを表示する	4
2	親を持たない独立クラスのあるエンティティのインスタンスを新規登録する	新規登録するインスタンスの諸属性値を入力する	重複登録を避けるためエンティティのインスタンスを検索する	新規インスタンスを書き込む	登録に成功したインスタンスの諸属性を表示する 登録に失敗したメッセージを表示する	5
3	親を持たない独立クラスのあるエンティティのインスタンスの属性値を更新する	更新対象のインスタンスを同定するために機能プロセスタイプ#1を実行する．そのためのCFP値は4				8
		属性値を更新するインスタンスの諸属性値を入力する	なし	属性値変更後のインスタンスを書き込む	更新に成功したインスタンスの諸属性を表示する 更新に失敗したメッセージを表示する	
4	あるエンティティのインスタンスを削除する．ただし，そのエンティティに存在従属したり，そのエンティティを参照するエンティティはグラフ上に存在しない	削除対象のインスタンスを同定するために機能プロセスタイプ#1を実行する．そのためのCFP値は4				8
		削除の実行を入力する	なし	同定されたインスタンスを削除する	削除に成功したインスタンスの諸属性を表示する 削除に失敗したメッセージを表示する	
5	あるエンティティのインスタンスを削除する．ただし，そのエンティティに存在従属したり，そのエンティティを参照するn種類のエンティティがグラフ上に存在する	削除対象のインスタンスを同定するために機能プロセスタイプ#1を実行する．そのためのCFP値は4				n+8
		削除対象のインスタンスに存在従属，または削除対象のインスタンスを参照しているインスタンスを同定するために，削除対象のエンティティに存在従属，または削除対象のエンティティを参照しているエンティティの種類数nだけReadする．そのためのCFP値はn				
		削除の実行を入力する	なし	同定されたインスタンスを削除する	削除に成功したインスタンスの諸属性を表示する 削除に失敗したメッセージを表示する	
6	n種類の親を持つ，ある存在従属エンティティのインスタンスを新規登録する	親エンティティのインスタンスを同定するために，種類数nだけエンティティをReadする．そのためのCFP値はn				n+5
		自分自身を新規登録するために機能プロセスタイプ#2を実行する．そのためのCFP値は5				
7	n種類の親を持つ，ある存在従属エンティティのインスタンスの属性値を更新する	親エンティティのインスタンスを同定するために，種類数nだけエンティティをReadする．そのためのCFP値はn				n+8
		自分自身の属性値を更新するために機能プロセスタイプ#3を実行する．そのためのCFP値は8				

は，利用者機能要件のタイプでは，存在従属グラフ上のあるエンティティからみて，そのエンティティが直接依存するエンティティの種類数と，そのエンティティを直接参照するエンティティの種類数だけを問題とし，芋づる式に参照をたどってエンティティ数をカウントしないようにしている．これは，表の使いやすさと重複計測の回避のためである．そして，上位のエンティティのドメイン管理要件を実現するためにカウントされるであろうデータ・グループの移動は重複してカウントしないようにしている．たとえば，再び表 7 の利用者機能要件である「受注を新規登録する」に注目する．もしも，ここで表 7 に記載された個々の機能プロセスに対して誤って表 8 を適用した場合は，4 + 4 + 7 = 15 という過大な CFP 値を得ることになってしまうが，表 8 は「機能プロセス」ではなく「利用者機能要件」に対して適用するように作成されていることに注意されたい．この例では表 8 の FUR タイプ#6（n = 2）に該当するが，そこでは n 種類の親クラスのインスタンスを検索・同定するために，あらためてそのためのデータをエントリすべくカウントするようにはなっていない．もしもそうした場合は，少なくとも CFP 値は n だけ余分に増加してしまう（エクジットについても同様）．しかしながら，実際に登録するインスタンスは n 種類の親クラスのインスタンスは既知であるとしなければならない．よって，これらのエントリをカウントしないようにしている．

表 8 の使い方は以下のとおりである．

1. Step 3 で明らかにした利用者機能要件に着目する．
2. 利用者機能要件が扱うエンティティの特性を Step 1 で作成した存在従属グラフから確認する．このとき，対象となるエンティティが，存在従属グラフ上で，何種類の直接の依存先と何種類の直接の被依存エンティティを持つかを確認しておく．
3. 表 8 を参照しながら，着目している利用者機能要件のタイプに応じた CFP を求める．

たとえば，例題の表 6 について実施した場合，「顧客を新規登録する」利用者機能要件は，表 8 の FP タイプ#6 に該当し，受注の n（存在従属している親エンティティの数）は 0 であるため，その CFP 値は 5 である．また，「顧客を抹消する」は，表 8 の FP タイプ#5 に該当し，顧客に直接依存するエンティティの数は 2（受注と出荷）であるため，その CFP 値は 10 である．この作業を，表 6 のすべての利用者機能要件について適用すると**表 9** を得る．表 9 からは，図 3 の存在従属グラフを扱うためのアプリケーションの機能規模は 139CFP であることも分かる．

表 9 利用者機能要件ごとの CFP 値
Table 9 CFP values for each user functional requirements.

ドメイン管理要件	管理対象の親クラスの数 n	FUR#	利用者機能要件（≒ユースケース）	CFP値
顧客を管理する	0	1.1	顧客を新規登録する	5
		1.2	顧客を検索・同定する	4
		1.3	顧客の属性を更新する	8
		1.4	顧客を抹消する	10
商品を管理する	0	2.1	商品を新規登録する	5
		2.2	商品を検索・同定する	4
		2.3	商品の属性を更新する	8
		2.4	商品を抹消する	10
受注を管理する	2	3.1	受注を新規登録する	7
		3.2	受注を検索・同定する	4
		3.3	受注の属性を更新する	10
		3.4	受注を抹消する	9
出荷を管理する	1	4.1	出荷を新規登録する	6
		4.2	出荷を検索・同定する	4
		4.3	出荷の属性を更新する	9
		4.4	出荷を抹消する	8
在庫を管理する	1	5.1	在庫を新規登録する	6
		5.2	在庫を検索・同定する	4
		5.3	在庫の属性を更新する	9
		5.4	在庫を抹消する	9
			合計	139

4. 比較のためのケーススタディ

この章では，ある程度の大きさと複雑さを有する業務アプリケーションに対して提案手法を適用して機能規模を測定した結果と，COSMIC 法を本来の手順で適用して機能規模を測定した結果を比較する．COSMIC 法のためには，要求記述がよりイベントフローに近い情報を持っている方が適しているため，測定対象には，ビジネスホテル宿泊予約サイトの事例を用いた．この事例ではインターネットから日本国内の提携ホテルの宿泊プランの検索と予約を行うことができる．**図 4** に宿泊予約サイトのやや詳細な要求記述を示す．

4.1 提案手法による測定

本節では，図 4 の要求記述に対して，3.2 節で述べた提案手法の手順を適用する．具体的には，1) 存在従属グラフの作成，2) ドメイン管理要件の識別，3) 利用者機能要件の識別，4) 表引きによる利用者管理要件ごとの CFP 値の取得の順で測定を進める．**図 5** に要求記述から作成したエンティティの存在従属グラフを示す．複雑さを回避するため図 5 では，表 4 で示した存在従属グラフ作成時のガイドラインから導かれる識別子や外部キーは表記していない．そして，**表 10** に存在従属グラフから導出したドメイン管理要件，利用者機能要件および，表 8 を参照することによって求めた機能プロセスごとの CFP 値を示す．

表 10 の CFP 値は，利用者機能要件からただちに図 5 の存在従属グラフと表 8 を参照して求めたものである．提案

このシステムは，インターネットから日本国内のビジネスホテルの宿泊プランの検索と宿泊プランの予約を行うことができる．宿泊プランの検索は会員でなくても可能であるが，予約は会員でないとできない．利用者が宿泊を希望する地区（都道府県，エリア），利用期間（チェックイン日付，および泊数）を指定すると，指定した地区に所在する提携ビジネスホテルの一覧が表示される．利用者が一覧に表示されたホテルを選択すると，指定されたホテルの指定された宿泊期間に予約可能な宿泊プランの一覧が表示される．宿泊プランとは，部屋グレード，部屋ベッドタイプ，喫煙可否，食事オプションの組み合わせに名前を付けたものである．利用者が一覧から希望する宿泊プランを選択すると，会員の認証が求められる（認証済みでない場合）．会員はあらかじめ登録した会員IDとパスワードを入力する（この際，会員登録が済んでいない場合は，新規登録を行うこともできる．その場合，利用者は氏名，住所，メールアドレス，パスワードを登録する）．認証が成功すると，選択された宿泊プランの詳細，チェックイン日，泊数，チェックイン日からチェックアウト日にかけての日毎の宿泊料金，そして宿泊料金の合計金額が表示される．会員が表示内容を確認し，予約を実行すると，宿泊プランが引当てられ（ただし，具体的な部屋の割り当てまでは行われない），予約番号が表示されるとともに，予約番号と予約内容が記載された電子メールが会員の登録済みアドレス宛に届く．

図 4 サンプルプロジェクトの要求記述
Fig. 4 Requirement description of the booking site.

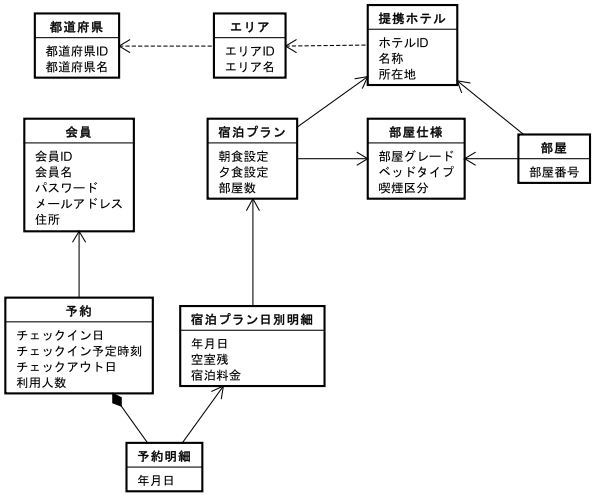

図 5 エンティティの存在従属グラフ
Fig. 5 Existence dependency graph of domain entities.

表 10 提案手法で求めた CFP 値
Table 10 CFP value obtained in the proposed method.

ドメイン管理要件	管理対象の親クラスの数n	利用者機能要件	提案手法で求めたCFP値
会員を管理する	0	会員を新規登録する	5
		会員を検索・同定する	4
		会員を更新する	8
		会員を抹消する	9
予約を管理する	2	予約を新規登録する	7
		予約を検索・同定する	4
		予約を更新する	8
		予約を抹消する	8
宿泊プラン日別明細を管理する	1	宿泊プラン日別明細を新規登録する	6
		宿泊プラン日別明細を検索・同定する	4
		宿泊プラン日別明細を更新する	9
		宿泊プラン日別明細を抹消する	9
宿泊プランを管理する	2	宿泊プランを新規登録する	7
		宿泊プランを検索・同定する	4
		宿泊プランを更新する	8
		宿泊プランを抹消する	9
部屋仕様を管理する	0	部屋仕様を新規登録する	5
		部屋仕様を検索・同定する	4
		部屋仕様を更新する	8
		部屋仕様を抹消する	9
部屋を管理する	2	部屋を新規登録する	7
		部屋を検索・同定する	4
		部屋を更新する	8
		部屋を抹消する	8
提携ホテルを管理する	1	提携ホテルを新規登録する	6
		提携ホテルを検索・同定する	4
		提携ホテルを更新する	9
		提携ホテルを抹消する	10
エリアを管理する	1	エリアを新規登録する	6
		エリアを検索・同定する	4
		エリアを更新する	9
		エリアを抹消する	9
都道府県を管理する	0	都道府県を新規登録する	5
		都道府県を検索・同定する	4
		都道府県を更新する	8
		都道府県を抹消する	9
		要求記述に直接関係するCFP値合計	53
		CFP値合計	245

手法では，このサイトのソフトウェアの機能規模は 53 と測定された．なお，表 10 では，図 4 の記述からだけでは直接的に求めることができない利用者機能要件についてはグレイアウトしてある．ここで，機能規模の測定とは直接的には関係ないが，要求記述からエンティティの存在従属グラフをまず最初に作成することは，ドメイン管理要件を明確にし，それらの CRUD の必要性から利用者機能要件がほぼ求まるため，要求定義プロセスにおける機能要求の網羅性確認のために非常に有効な方法であると思われる．

4.2 COSMIC 法による測定

今度は，図 4 の要求記述に対して，2.2 節で説明した COSMIC 法の手順を適用する．COSMIC 法では，最初に利用者機能要件を明確にする必要がある．利用者機能要件は，UML のユースケースに相当するとされる [11] ため，

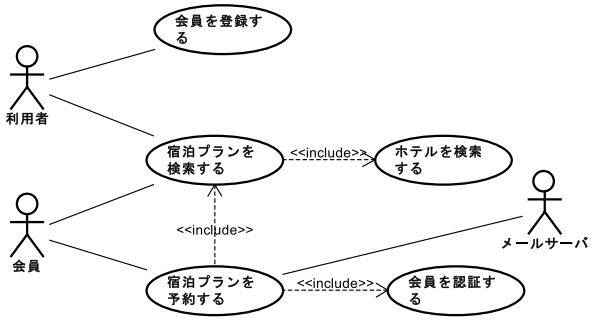

図 6 宿泊予約サイトのユースケース図
Fig. 6 Use case diagram of the hotel reservation site.

最初にユースケース図を作成した（**図 6**）．

次に，ユースケースごとにイベントフローを作成し，ユースケースを論理的に実現するための機能プロセスを洗い出す．これは，結構労力の必要な作業である．もちろん，開発が決定したユースケースに対してこの作業を行うのは必要であるが，作り込みを行うかどうかが定まっていないユースケース候補に対してこの作業を行うのは避けたいところである．しかしながら，COSMIC 法で機能規模の測定を行うのであればそうはいかない．

表 11 は，そのようにして識別した，各々の機能プロセスと機能プロセスが移動するデータ・グループの種類を数えたものである．利用者機能要件欄には，図 6 で明らかにしたユースケース名を記入した．機能プロセス欄には，ユースケースごとに作成したイベントフローで明らかにした，アクタの入力からそれに対するシステムの応答までの対話の 1 往復を記入してある．そして，エントリ欄からエクジット欄の間のデータ・グループは図 5 の存在従属グラフに記載されているエンティティをそのまま使用した．エクジット欄で X(eXit) が複数マークされている機能プロセスは，アクタへの通常の応答以外にもエラーメッセージの出力とメール送信があることを示している．そして，CFP 値の欄には，その行に含まれる E，R，W，X の個数を単純にカウントした値が記入してある．

4.3 計測結果に関する考察

4.3.1 ケーススタディの計測結果について

さて，表 10 と表 11 では，異なるプロセスで CFP 値の合計を求めているにもかかわらず，提案手法が 53 で，COSMIC 法は 52 ときわめて近い値になっている．また，**表 12** は，表 10 と表 11 の値を，ドメイン管理要件ごとに集計して比較したものであるが，こちらもきわめて近い値になっている．これらの値が世界標準の COSMIC 法の値であることの価値は大きい．業界では CFP を基準に，これを介して，人時値への換算事例や実装環境ごとのソースコード行数（LOC）への換算実績データが蓄積されつつあるからである．ただし，COSMIC 法だけを用いた測定では，機能プロセスに近視眼的にならないように注意しなけ

表 11 COSMIC 法で求めた CFP 値
Table 11 CFP values obtained in the COSMIC method.

利用者機能要件	機能プロセス	トリガ・エントリ	会員	都道府県	エリア	提携ホテル	宿泊プラン	部屋仕様	宿泊プランの日別明細	予約	予約明細	エクジット	CFP値
会員を登録する	アクターがメニューから会員登録を選ぶと，会員登録フォームが表示される	E										X	2
	アクターが会員属性を入力すると，当該会員が新規登録され，登録完了メッセージが表示される	E	R W									X X	5
会員を認証する	アクターが会員IDとパスワードを入力すると，認証の成否が表示される	E	R									X X	4
ホテルを検索する	アクターが都道府県とエリアを指定すると，当該エリアに存在する提携ホテルの一覧が表示される	E		R	R	R						X X	6
宿泊プランを検索する	アクターが，宿泊を希望する地区（都道府県，エリア），利用期間（チェックイン日付，および泊数）を指定すると，システムは「ホテルを検索する」を実行する	E	ホテルを検索するでカウント済みの6										7
	アクターがホテルを選択すると，指定されたホテルの指定された宿泊期間に予約可能な宿泊プランの一覧が表示される	E				R	R	R	R			X X	7
宿泊プランを予約する	アクターが宿泊プランを選択すると，システムは「会員を認証する」を実行する	E	会員を認証するでカウント済みの4										5
	認証が成功すると，選択された宿泊プランの詳細，チェックイン日，泊数，チェックイン日からチェックアウト日にかけての日毎の宿泊料金，そして宿泊料金の合計金額が表示される	E				R	R	R	R			X X	7
	アクターが予約を実行すると，宿泊プランが引当てられ（ただし，具体的な部屋の割り当てまでは行われない），予約番号が表示されるとともに，予約番号と予約内容が記載された電子メールが会員の登録済みアドレス宛に届く．	E							R W	W	W	X X X	8
											CFP値合計		51

表 12 COSMIC 法で求めた CFP 値
Table 12 CFP values obtained in the COSMIC method.

ドメイン管理要件	提案手法で求めたCFP値	COSMIC法で求めたCFP値
会員を管理する	9	11
予約を管理する	44	40
CFP値合計	53	51

ればならない．なぜならば，表 10 中でグレイアウトしてある利用者機能要件は，要求記述からは直接読み取りにくいものの必要な要件のはずである．提案手法では，先に存在従属グラフに変換し，利用者機能要件はそこから導出するため，このような利用者管理要件も取りこぼさないが，COSMIC 法では取りこぼしてしまう可能性がある．これは，高速道路の建設に例えれば，本線の見積りに気を奪われ，パーキングやインターチェンジの存在を見落とすようなものである．

4.3.2 重複計測について

利用者機能要件には，取りこぼしのリスクとは反対に，それらの機能規模を重複計測するリスクがある．提案手法では，3.2.4 項で説明したような対策を行ってはいるものの重複計測を回避することはできない．なぜならば，存在従属グラフに登場する個々のエンティティについて CFP を求めていく際に，どうしても依存性の直上・直下のエンティティが含まれるデータ・グループの移動を重複してカウントするためである．

回避策としては，次のようなアルゴリズムが考えられる．すなわち，1) 存在従属グラフに登場するすべてのエンティティについて，親のないエンティティから開始して，その依存関係を再帰的下降的にたどりつつ，それぞれを CRUD するための利用者機能要件をすべて生成する．その際，利用者要件には，親のないエンティティを 0 としたレベル番号を付与する．親のないエンティティが複数存在する場合には，それぞれについて同様に利用者要件を生成する．再帰関連が現れた場合は 1 回だけの再帰で処理は打ち切るようにする．2) 生成が終了したならば，利用者要件名で整列し，重複する利用者要件を除去する．3) 今度は，存在従属グラフの末端側に位置するエンティティを CRUD する利用者機能要件（付与されたレベル番号が大きい利用者機能要件）からレベル番号の降順に，参照先のエンティティの移動を除外したデータ移動を計測する．

4.3.3 エントリのバリエーションについて

提案手法では，エンティティごとに基本となる 4 つの利用者機能要件（新規登録，検索，更新，削除）を想定し，それを基に CFP 値の単価を設定している．しかしながら，新規登録，検索，更新について，1 つのエンティティに対して様々なバリエーションが存在する業務アプリケーションも多いはずである．たとえば，新規登録においては，画面上での 1 件ずつエントリするほかに，表計算ソフトなどで複数件数のデータを用意し，それを一括エントリするような機能が実装されることが多い．

最後に，このようなエントリのバリエーションが COSMIC 法で計測する機能規模に影響するかについて検討する．結論から先にいえば，COSMIC 法では，いっさい影響されない．なぜならば，COSMIC 法による機能規模測定は，ソフトウェアの「規模」に影響するかもしれない機能性のあらゆる側面を測ろうとしているわけではないからである [4]．したがって，COSMIC 法では，ただ愚直に移動

するデータ・グループ種類数をカウントするのみであり，エントリの方法や実現手段方法の違い，あるいは 1 データ移動あたりのデータ属性の数の影響はこの測定法ではとらえられない．

5. まとめ

本稿では，業務アプリケーションの機能規模測定のためにエンティティの存在従属性に着目したグラフと表 8 を導入することによって COSMIC 法を機敏かつ簡便に利用する手法を提案した．COSMIC 法の概算法を開発対象やプロジェクトの事情に合うように提案した事例はすでにいくつも存在している（文献 [12], [13], [14] など）が，エンティティの存在従属性に由来する制約を，機能プロセスへの展開ルールとして位置づけ，扱うエンティティの存在従属グラフ上の位置に着目することによって機能規模の測定に結び付けたところに本提案手法のオリジナリティがある．

渡辺はそのブログ [15] の中で，業務アプリケーションというものが「テーブル構造から導出可能な多彩な影」でしかない，と発言しているが，まさにそのとおりであろう．エンティティの構造が決まれば，アプリケーションのユースケースはそれに沿って大部分は自動的に決まってしまうものであること [10] を提案手法の検討中にも改めて確認した．

本稿の提案手法が要求定義プロセスの早期の段階で簡単に規模を見積もれる手法として幅広く利用され，システム開発の依頼者と開発の技術者の間で互いに納得感のある開発請負契約を締結したり，あるいは機能規模を基準にプロジェクトの生産性や欠陥密度など検討したりする材料に使われるようになることを願っている．しかしながら，提案手法には，特に表 8 などはまだまだ改良の余地が残されていると思われる．また，存在従属グラフを入力とした自動計測が前提であれば，重複計測を効率的に排除するアルゴリズムも実装可能であろう．それらは現場で提案手法を実際に運用する中で，改良していきたい．また，運用の中で CFP 値と実際のコスト（人時値や実装言語別の LOC 値）の関係も明らかにしていきたい．

参考文献

[1] Cohn, M.: アジャイルな見積りと計画づくり—価値あるソフトウェアを育てる概念と技法，毎日コミュニケーションズ (2009).

[2] Albrecht, A.J. and Gaffney, Jr. J.E.: Software function, source lines of code and development effort prediction: A software science validation, *IEEE Trans. Softw. Eng.*, Vol.SE-9, pp.639–648 (1983).

[3] The Common Software Measurement International Consortium: COSMIC 機能規模測定法，Version 3.0 手法概要編，日本ファンクションポイントユーザ会 (2007).

[4] The Common Software Measurement International Consortium: COSMIC 機能規模測定法，Version 3.0 測定マニュアル，日本ファンクションポイントユーザ会 (2007).

[5] 情報サービス産業協会 REBOK 企画 WG：要求工学実践ガイド：REBOK シリーズ 2，近代科学社 (2014).

[6] Chen, P.: The Entity Relationship Approach to logical Database Design, *QED Information Science*, Wellesley (1979).

[7] Snoeck, M. and Dedene, G.: Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types, *IEEE Trans. Softw. Eng.*, Vol.24, No.4, pp.233–251 (1998).

[8] 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).

[9] 井田明男，金田重郎，熊谷聡志，藤本明莉：存在従属性に着目した論理要件ロバストなドメインモデルの作成—ドメインクラス図をユビキタス言語として用いるために，情報処理学会論文誌，Vol.56, No.5, pp.1340–1350 (2015).

[10] 金田重郎，井田明男，矢野寛将：存在従属に基づくユースケースの逆生成，電子情報通信学会，知能ソフトウェア研究会，信学技報，Vol.115, No.54, KBSE2015-3, pp.13–18 (2015).

[11] 藤井　拓：ビジネスアプリ開発者のための機能規模測定手法 COSMIC 法入門，オージス総研・オブジェクトの広場 (2010)，入手先 ⟨https://www.ogis-ri.co.jp/otc/hiroba/technical/IntroCOSMIC/IntroCOSMICPart1Jun2010.html⟩.

[12] 駒木和彦：COSMIC 法向けの早期・概算見積もり—不完全な要件定義への適用，プロジェクトマネジメント学会研究発表大会予稿集 2010（春季），pp.188–192 (2010).

[13] 駒木和彦：CRUD 図で CFP を測定する—より簡単で確実な COSMIC 法による機能規模測定，プロジェクトマネジメント学会研究発表大会予稿集 2011（春季），pp.379–384 (2011).

[14] 木村めぐみ，藤井　拓：COSMIC 概算法による機能規模測定のコスト削減の検討結果，情報処理学会研究報告，Vol.2012-SE-175, No.11 (2012).

[15] 渡辺幸三：設計者の発言—アプリの類型における処理テーブルの形式化 (2015)，入手先 ⟨http://watanabek.cocolog-nifty.com/⟩.

井田 明男 （正会員）

1959 年生．1984 年同志社大学文学部文化学科（心理学専攻）卒業．銀行員，大手メーカ系 SE を経て，1989 年 4 月株式会社オージス総研入社．業務系および技術系のシステム開発やオブジェクト指向と UML を用いた開発のトレーニングやコンサルティングに従事．2012 年に独立．現在，有限会社井田代表取締役，同志社大学理工学部講師．要求デリング，構造モデリング，コード自動生成の研究に従事．特種情報処理技術者．電子情報通信学会会員．

金田 重郎 （正会員）

1951 年生．1974 年京都大学工学部電気第二学科卒業，1976 年 3 月京都大学大学院工学研究科電子工学専攻修士課程修了．同 4 月日本電信電話公社・武蔵野電気通信研究所入所．大型汎用電子計算機の実用化，ならびに，誤り検出訂正符号の研究に従事．1997 年 4 月同志社大学大学院総合政策科学研究科教授・同工学部教授．現在は，同理工学研究科・情報工学専攻教授．要求分析・センシング応用技術の研究に従事．工学博士（京都大学），技術士（情報処理部門）．IEEE 会員．

熊谷 聡志

1990 年生．2014 年 3 月同志社大学理工学部インテリジェント情報工学科卒業．2014 年 4 月同志社大学大学院理工学研究科情報工学専攻博士前期課程入学．要求モデリング手法の研究に従事．

矢野 寛将

1991 年生．2014 年 3 月同志社大学理工学部インテリジェント情報工学科卒業．2014 年 4 月同志社大学大学院理工学研究科情報工学専攻博士前期課程入学．要求モデリング手法の研究に従事．