Vol.2011-SE-171 No.28
2011/3/15

COSMIC 法によるリッチ UI アプリの機能規模測定プラクティス

藤井拓[†]　椿野禎三[†]　林和彦[†]　木村めぐみ[†]

本論文では，シンクライアントとリッチクライアントに基づくビジネスアプリケーションソフトウェアの機能規模測定を COSMIC 法により行うためのプラクティスを提案する．提案したプラクティスを用いて 3 件のリッチクライアントアプリと 4 件のシンクライアントアプリの規模測定を行い，開発労力との相関を調べた．その結果，リッチクライアント 1 件のデータが他のデータから大きく外れたものの，2 件のリッチクライアントアプリと 4 件のシンクライアントアプリの機能規模と開発労力は比較的良い相関を示すことが分かった．

Practices Of The COSMIC Method for Measuring Rich UI Applications

Taku Fujii[†], Teizo Tsubakino[†], Kazuhiko Hayashi[†] and Megumi Kimura[†]

Practices for the COSMIC method to measure functional size of business application software based on rich client and thin client are proposed in this paper. Proposed practices were applied to three rich client application and four thin client applications, and correlation between obtained functional size and development effort is examined. As the results, one rich client application shows very different correlation from other applications, remaining two rich client applications and four thin client applications show good correlation between functional size and development effort.

1. はじめに

90 年代後半に Java が登場し,その後 J2EE (Java 2 Enterprise Edition)が提供されるようになって，ビジネスアプリケーションの分野では新規開発されたり，再構築されるシステムにおいて JSP (Java Server Page)に代表されるようなマークアップ言語で UI (User Interface)を実装する方式が普及していった．本論文では，このようなマークアップ言語を主に用いて実装した UI をシンクライアントと呼ぶ．このようなシンクライアントは，スケーラビリティーが良く，アプリの配布コストが低いシステムを開発するためには有効だったものの，その反面操作性が良いシステムを作ることが困難だった．

このようなシンクライアントの問題点を克服するために，近年注目されているのが JavaScript のようなスクリプト言語で UI を実装するリッチクライアントである．本論文では，JavaScript のようなスクリプト言語に加えて C#のようなコンパイラ言語で実装された 3 層アーキテクチャの UI をリッチクライアントと呼ぶ．リッチクライアントを使うことで，操作性が良く，配布コストが低いソフトウェアが実現できる可能性がある．しかし，リッチクライアントアプリはシンクライアントと比べて画面あたりの機能数が多いため，シンクライアントの同様に開発労力の見積もりが行えないのではないかという懸念がある．また，開発言語や開発環境の違いによりシンクライアントとリッチクライアントとでは開発生産性，ひいては開発コストがかなり異なるのではないかという点も懸念される．

本論文では，まずシンクライアントアプリとリッチクライアントアプリの特徴を説明する．次に機能規模測定手法 COSMIC 法の概要を説明し，本論文で提案する COSMIC 法によるシンクライアントアプリとリッチクライアントアプリの機能規模測定プラクティスを説明する．さらに，提案した機能規模測定プラクティスを 7 件のビジネスアプリに適用し，求めた機能規模の測定結果と開発労力との相関を示す．最後に，本論文のまとめと今後の課題を論じる．

[†]株式会社オージス総研　技術部ソフトウェア工学センター
Software Engineering Center, IT R&D Dept., OGIS-RI Co., Ltd.

2. シンクライアントアプリとリッチクライアントアプリの特徴

2.1 シンクライアントアプリの特徴

シンクライアントを実装するためのマークアップ言語は，なんらかの仕組みで HTML に変換され，その HTML がサーバからクライアントに送信されることで Web ブラウザ上で UI が表示される．また，シンクライアントを実装するためにはカスタムタグのような画面部品を使用することができるが，そのような画面部品はユーザと対話的にデータを操作するような機能を持たない，比較的単純なものが多い．

シンクライアントは HTML の表現力や画面部品の機能という点で制限される．そのため，画面毎のユーザ操作数が比較的少数になるように設計されることが多い．その結果として，ユースケースによって表現されるような１つの業務は複数の画面のシーケンスを用いて実行され，各画面は少数のイベントフロー（必要なユーザ操作とシステムの応答）に対応するように作成されることが多い．また，もっとも一般的なシステムの利用パターンである基本フローを中心に画面シーケンスが作成されることが多い．

2.2 リッチクライアントアプリの特徴

リッチクライアントには，Web ブラウザ上で動作するものとクライアントの OS の GUI ライブラリ上で動作するものがある．前者は JavaScript など Web ブラウザ上で動作するスクリプト言語で実装されるものであり，後者は C#などの言語で GUI ライブラリを用いて実装される．これらの両者とシンクライアントの大きな違いは，クライアント側にあるプログラムコードにより UI を表示したり，UI に対する操作で発生するイベントに対するアクションを実行するという点である．このような動作形態により，リッチクライアントはシンクライアントよりも１画面上で複雑な表示やアクションを行うことができる．また，プログラムコードにより複雑な表示や複数のアクションを受け付ける画面部品としてモジュール化することが可能になる．

１画面上で複雑な表示や複数のアクションに対応することができる点や，より複雑な表示やアクションを集積した画面部品を利用することで，リッチクライアントでは以下のような機能の提供が可能になる．

- 複数の検索機能を１つの画面に集約し，その画面の検索結果を選択して複数の異なる機能を実行する
- 複数のデータをグリッドなどに表示し，それらのデータの編集（追加、更新、削除）を自由に編集する
- １つの業務（ユースケース）を実行するための画面から，必要に応じて他の業務（ユースケース）を実行するための画面を呼び出すことができる
- メニューの選択や画面上でのマウスのドラッグ操作を行うなど，複数の異なる操作から同じユースケースを開始する

これらの機能により，１つの画面からユースケースの分岐が発生したり，１つのユースケースから他のユースケースの呼び出しが発生したり，ユースケースの開始点を複数設けたりすることが可能になる．その結果，イベントフローに沿って順次的に画面を使い，ユースケースと画面の対応が比較的取りやすいシンクライアントの場合と異なり，リッチクライアントの場合は画面とユースケースの対応が非常に取りにくくなる．

3. 機能規模測定手法 COSMIC法とその測定プロセス

3.1 機能規模測定手法 COSMIC法

COSMIC法1), 2)は，ISOで認められた４つの機能規模測定手法の１つであり，ソフトウェアの外界（または外部システム）との間のデータ移動の数と，ソフトウェアと永続ストレージの間のデータ移動数の合計値を求めることで機能規模を定量化することを特徴とする測定法である．図1はCOSMIC法の規模測定を行う４種類のデータ移動を示したものである．また，これら４種類のデータ移動に対する説明をまとめたものが表 1である．

COSMIC 法では，永続ストレージに保管される ER (Entity Relationship)モデルのエンティティに相当する

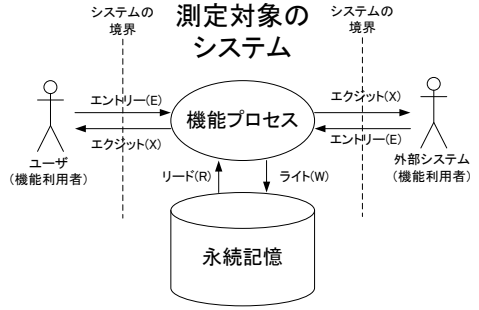

図 1　COSMIC 法で測定するデータ移動

表 1　COSMIC 法で測定する４種類のデータ移動

データ移動の種類	説明
エントリー(E)	システムの境界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(X)	UI や通信によりシステムの境界に出力されることに対応するデータ移動
リード(R)	永続ストレージから読みだされるデータ
ライト（W）	永続ストレージに対して書き込まれるデータ

データのまとまりの 1 つの移動を 1CFP という単位で測定する 3). さらに，ER モデルのエンティティに準ずるオブジェクトを”注目オブジェクト”と呼び，実際にデータ移動を行うデータのまとまりを”データグループ”と呼ぶ.

COSMIC法の機能規模測定の正式な手順はCOSMIC法の測定マニュアル1)に記しているので参照して頂きたい. この正式な測定手順を簡略化すると以下のようになる.

A) 利用者機能要件の識別
B) 注目オブジェクトの識別と ER 図の作成
C) 利用者機能要件からの機能プロセスの識別
D) 機能プロセス毎のデータ移動の識別
E) 機能規模の集計

利用者機能要件(FUR : Functional User Requirements)とは，ユーザの視点でソフトウェアの機能要求を捉えたものであり，ビジネスアプリケーションの場合は”ユースケース”に対応する. COSMIC 法で機能規模を測定する際には，これらの利用者機能要件をまず要求や既存システムなどから識別する. さらに，利用者機能要件から利用者機能要件中に存在する注目オブジェクトを識別し，それらの注目オブジェクトに対する ER 図を作成する.

次に，利用者機能要件を構成する機能プロセスを識別する. 機能プロセスとは，ソフトウェア機能の利用者（機能利用者）がその利用目的を果たすためにソフトウェアと行う一連の相互作用を表す. 機能プロセスとなるためには，機能利用者により機能プロセスをトリガーするエントリーのデータ移動に加えて 1 つ以上のデータ移動を持たねばならない.

さらに，機能プロセス毎に先に述べた 4 種類のデータ移動を識別し，識別されたデータ移動の数により機能プロセスの機能規模を求める. これらの機能プロセス毎の機能規模を利用者機能要件に集計し，さらにソフトウェア全体の利用者機能要件に渡り集計することでソフトウェア全体の機能規模を求めることができる.

3.2 シンクライアントアプリの機能規模測定プラクティス

2.1 節で説明したように，シンクライアントの場合には画面とユースケースの対応付けをしやすいことが多い. そのため，機能規模の測定 A)と C)のステップは以下のように進めることができる.

1) ユースケースを利用者機能要件に設定する（ステップ A に相当）
2) 利用者機能要件（ユースケース）を満たすための画面を識別する（ステップ C のサブステップ）
3) 画面に関係する機能プロセスを識別する（ステップ C のサブステップ）

ここで 3)において識別される機能プロセスは，以下の 2 種類であることが多い.

i) 画面を初期表示するための機能プロセス
ii) 画面上のボタンなどの操作によって開始される機能プロセス

i)の例としては，画面上のドロップダウンリストなどの選択項目をデータベースから読みだすためのデータ移動を含むような機能プロセスが挙げられる. それに対して，ii)は更新ボタンなどのボタンの押下をきっかけにしてデータベースに対する書き込みを行うような機能プロセスを挙げることができる.

次のステップ D)で機能プロセスを構成するデータ移動を識別する際は，文献 3)に示されたガイドラインを参考にして機能規模を概ね測定できる. このガイドラインに記述されていないケースで筆者らが悩んだのは「先行する画面で読み込んだメモリ上のデータグループを後続する画面で再び読み込むというデータ移動を考えるべきか」という点であった. 例えば，ログインユーザの ID や氏名などの情報は論理的にはログイン後の画面ではデータベースから読み込む必要がないと思われるが，そのようなメモリ上のオブジェクトについてデータ移動を数えるべきかどうかという問題である.

この点に関して COSMIC 法では明確な指針が設定されていないので，筆者らは以下のようなプラクティスを設定することにした.

- アプリ共通のデータグループ
  - ログインユーザの情報などすべての画面で共通に使われるデータグループはメモリ上に存在し、個別の画面でそのデータグループを読み込むデータ移動は発生しないと考える
- 1 つの画面シーケンス共通のデータグループ
  - 1 つの画面遷移シーケンスの先行する画面で一度読み込んだデータグループは、後続する画面でそのデータグループを読み込むデータ移動は発生しないと考える

以上のようなプラクティスを用いることで，シンクライアントの場合の機能規模は比較的容易に測定することができる.

3.3 リッチクライアントアプリの機能規模測定プラクティス

2.2 節で述べたように，リッチクライアントアプリの場合は機能規模測定の情報源となる要求が画面定義中心になされることが多い. そのため，ユースケースと画面の対

応付けが困難な場合が多い．さらに，画面遷移の経路上に共通の画面が登場するようなケースも多いが，それらの共通画面が異なる画面遷移の経路上に展開されているために共通画面であることに気付きにくいこともある．これらの事情で，リッチクライアントの場合は利用者機能要件からトップダウン的に機能プロセスを識別するのが難しいことがある．

そのため，筆者らは以下のような手順で機能規模を測定することにした．

A)　画面定義書から画面の使用順序をマインドマップ等で展開した図を作成する
B)　A)の中で共通の画面を特定し、共通画面グループとする
C)　A)で展開した画面中で共通画面以外の画面を、それらの画面を使うユーザ機能の種類別のシーケンスにグループ化する
D)　B)と C)で作成した各グループ中の各画面に対する機能プロセスを識別する

B)では，複数の画面から共通に呼び出されるような画面を識別し，それらの画面に付随する機能プロセスを「利用者機能要件」に準ずるものとして独立させている．COSMIC 法では，ユーザの視点から見て判別できないような内部機能の場合には内部機能間の共通性があってもその共通部分を一本化せずに機能規模を測定することを推奨している．しかし，筆者らはユーザの視点から見ても同一の機能を提供している画面（外部機能）の場合(B の場合）は COSMIC 法の推奨からは外れている場合と判断し，それらの画面を括りだして機能規模を測定を行うことにした．つまり，このようなユーザの視点から見て同一の機能を提供している画面は呼び出されている先に展開せずに共通画面として 1 回だけ機能規模を測定することにしたのである．

C)は，共通画面を除いた画面をその画面を使うユーザ機能別にグループ分けするステップであり，共通画面以外の「利用者機能要件」に準ずる機能グループに分類するためのものである．このグループ分けの際には，要求の大分類や中分類となるべく整合するように同じ注目オブジェクトを対象とするユーザ機能は 1 つのグループにまとめていくようにした．例えば、「売り上げを登録する」というユーザ機能と「売り上げを訂正する」というユーザ機能は「売り上げ」というユーザ機能のグループにまとめるということである．このようにまとめることで，機能グループを実際の開発作業と対応づけしやすくなり，機能グループ毎の開発生産性のばらつきを求めるのに役立つ．

4.　機能規模測定プラクティスの適用結果

本論文で提案した機能規模測定プラクティスをシンクライアントアプリ 4 件とリッチクライアントアプリ 3 件に適用した．表 2に適用した 7 件のアプリのプロフィールを示す．これらの 7 件のうち，B,Dには画面以外にバッチ処理が含まれている．A1，A2 は画面の大半はJSPで実装されていたが，一部JavaScriptを用いている．また，Dはアプリ機能一部をGはアプリ機能の大半をオフショアで開発した．さらに，Gは JavaScript を初めて実開発で使うメンバーで構成されていたのに対して，それ以外のプロジェクトではほとんどのメンバーがUI実装言語を実開発で使った経験を持っていた．

表 2　機能規模測定プラクティスを適用したプロジェクトのプロフィール

プロジェクト ID	開発種別	処理種別	UI 種別	UI 実装言語	UI実装言語を実開発で使った経験	オフショア開発部分の有無
A1	新規	オンライン	シン	JSP+JavaScript	あり	無
B	新規	オンライン +バッチ	リッチ	ActionScript	あり	無
C	新規	オンライン	シン	JSP	あり	無
A2	改修	オンライン	シン	JSP+JavaScript	あり	無
D	新規	オンライン ＋バッチ	シン	JSP	あり	一部あり
G	新規	オンライン	リッチ	JavaScript	なし	大部分
H	新規	オンライン	リッチ	C#	あり	無

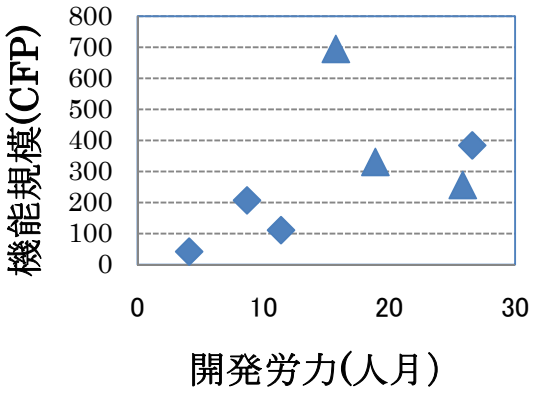

図 2　機能規模と開発労力の相関

図 2は，これらのアプリの機能規模と開発労力を散布図にしたものである．この散布図において，シンクライアントのデータ

はひし形のマーカ，リッチクライアントアプリのデータは三角形のマーカで示されている．

図 2を見ると，Gの機能規模 694CFP，開発労力 15.76 人月の点のばらつきが大きいもののそれ以外の点は比較的相関が高い可能性があることが分かる．

そこで，図 2の散布図からGのデータを外してみたものが図 3である．

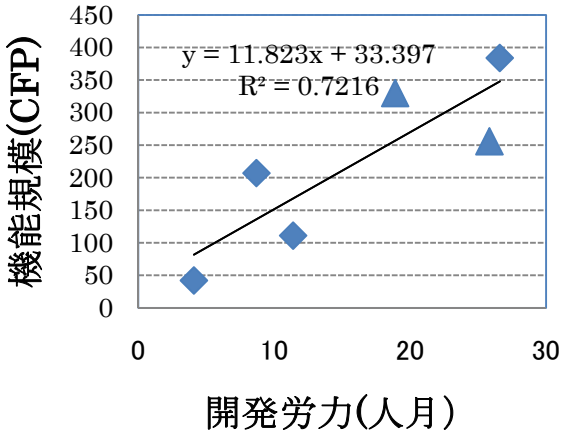

図 3 機能規模と開発労力の相関

図 3を見ると，シンクライアントアプリもリッチクライアントアプリも含めて機能規模と開発労力が比較的良い相関を示していることが分かる．

これらの結果より，G はなんらかの理由により全体的な機能規模と開発労力の相関からの外れ点である可能性が高いと考えられる．G については，開発メンバーのヒアリングから，開発の際に既存システムのデータベースのスキーマやチェックロジックを再利用したことが分かっている．この再利用による開発生産性の著しい向上により，G は散布図上で外れ点になったのではないかと考えられる．

図 3 より，本論文で提案した機能規模測定プラクティスにより求めたシンクライアントアプリの機能規模と労力の相関はリッチクライアントの機能規模と労力の相関とほぼ重なる傾向を示していると考えられる．言い換えれば，開発労力との相関という点では本論文で提案した機能規模測定プラクティスで求めたリッチクライアントの機能規模はシンクライアントの機能規模と同等な尺度になっていることが期待できる．

5. まとめ

本論文では，機能規模測定手法 COSMIC 法を用いてシンクライアントアプリとリッチクライアントアプリの機能規模測定を行うためのプラクティスを提案した．リッチクライアントアプリの機能規模においては，利用者機能要件から機能プロセスへの分解をトップダウンで行わず，画面の実行順序に基づいて画面を整理し，共通画面をくくりだした後に画面をユーザ機能毎にグループ化するという測定プラクティスを提案した．また，本論文で提案した機能規模測定方法をシンクライアントアプリ 4 件とリッチクライアントアプリ 3 件に適用し，機能規模の測定値を開発労力の相関を求めた．その結果，リッチクライアントアプリ 1 件のデータは全体的な機能規模と開発労力に対する相関から外れたものの，それ以外の 6 件のデータは比較的良い相関を示した．この結果より，本論文で提案した機能規模測定プラクティスはシンクライアント，リッチクライアントを問わず規模の定量化に有効なものだと期待できる．

今後は，本論文で提案した機能規模測定のプラクティスを適用して求めたシンクライアントアプリとリッチクライアントアプリの機能規模のデータをさらに集めてそれらが統計的に同じ分布であることを確かめたいと考えている．

謝辞
本論文で紹介したプロジェクトの測定結果を得るのに協力して頂いた（株）オージス総研の技術部ソフトウェア工学センターのメンバーとプロジェクトメンバーにこの場を借りて感謝の意を表します．

参考文献

1) COSMIC法ver3.0 測定マニュアル, http://www.jfpug.gr.jp/cosmic/CFFP-index.html
2) Abran A., Desharnais J.M., Maya M., St-Pierre D.,Bourque P., Design of a functional size measurement for Real-Time Software, UQAM, Research report No 13-23,1998.
3) COSMIC-FFPによる業務アプリケーションソフトウェア規模測定の指針, http://www.jfpug.gr.jp/cosmic/CFFP-index.html