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

2005－SE－150 (15)
2005／11／29

ソフトウェア規模見積もりと進捗管理手法の提案

上田健之† 海谷治彦† 海尻賢二†

ソフトウェア開発規模見積もりは通常，設計工程後に行われる．ある程度の誤差を許容する前提での上流工程での見積もり手法としてユースケースポイント法が提案されており，その有効性が報告されている．しかしユースケース図も実際の要求仕様書から見ると後の工程と考えられる。そこで要求定義がユースケースではなく自然言語で記述されている場合の上流工程での規模見積もり手法として、次の２つを提案する：①要求を記述する文の数を用いる．②要求記述中の補足的な図表をユースケース記述とみなしてユースケースポイントを計算する．この両者について開発規模との相関を調査した．その結果，すくなくとも情報型の組み込み開発については見積もりが可能であることが判明した．

Proposal of new software size estimation and process control methods

Takeyuki Ueda,† Haruhiko Kaiya † and Kenji KAIJIRI†

In ordinary case, software size estimation will be done after design process. The use case point is proposed in order to estimate effort before design process, and its effectiveness has been reported. But in real development process, estimation in upper requirement level is requested. We propose two estimation methods by using informal requirement documents: 1) the number of statement, 2) the use case point count based on the figure in requirement document. We investigated the effectiveness of these estimations based on the real development data and showed the possibility of estimation for the information type embedded software domain.

1. はじめに

ソフトウェア開発規模の見積りは基本的には詳細設計工程後に行う．設計資料を入力として作成すべきソフトのモジュール数，クラス数，行数等の開発規模を見積もり，過去の類似開発との比較から工数を見積もる．しかし，開発の受託契約は要求分析が終わった段階で要求仕様書に基づいて行われることが多い．その場合，契約は工数見積もりによるコスト分析の裏付けなしにおこなわれることになり，この時点ですでに開発を失敗に導く大きなリスクを発生させている．

そこで，要求仕様書から工数見積もりを行いたいというという需要がある．要求仕様から開発規模を見積もる手法としてユースケースポイント法が提案されている．しかしユースケース図も自然言語による原要求仕様から見ると後の工程の生産物と考えられ，そこからの見積もり手法がもとめられる．

自然言語による要求記述からの規模見積もりは一般に難しい．しかし，問題ドメインを組み込み機器のソフト開発に限れば，”機能追加の繰り返し”という開発の特徴を利用した見積もり手法の存在が期待される．

以上を鑑み，組み込みソフト開発について要求仕様からの規模見積もりに使える指標を提案し，その有効性を検証することを本論文の目的とする．

尚，見積もりの最終的な目的は工数の見積もりであるが，本研究では工数の見積もりではなく規模の見積もりを目的とする．工数は規模に対して開発環境の優劣，要因の錬度，開発対象の難度など各種の環境要因の影響を受け，これらの影響度合いについても実証的な研究が必要となり，研究の難度が高い．そこで，今回はそれらの影響を受けない規模についての実証的な調査を行った．

以下２章では測度と見積もり手法の概要を述べ、３章では非定型要求仕様書に基づく見積もり手法を提案し、４章で実験によりその可能性を示す。５章では関連研究との関係を明らかにし、６章で結論、今後の課題を述べる。

2. 測度と見積もりの概要

本章では測度と見積もりの手法について説明し，合わせて上流工程での見積もりの必要性と，その難しさ

† 信州大学大学院総合工学系研究科
Shinshu University, Graduate School of Science and Technology

について述べる．

2.1 ソフトウェア見積もり

ソフトウェア見積もりは経験データに基づいておこなわれる．つまり，類似する開発での実績データを利用して見積もりを行う．見積もりの対象は一般には開発規模である．工数等の他の effort は見積もった開発規模から経験則によって算出される．

2.2 測　　度

これまでに各種の測度が提案されている．それらを計測する対象より分類する．

2.2.1 ソースコード

ソースコードはソフトウェア開発の最終生産物であるため，これににについての測度は直接見積もりや良形評価の対象となる．主なものを以下に挙げる．

- コード行数（Loc：Line of code）：ソースコードには改行の入れ方に自由度があるため一意な測度にはならず，また同じアルゴリズムをコーディングした場合でもプログラミング言語によって Loc は大きくかわる．これらの欠陥はあるものの計測方法が簡単でありわかりやすいため，今日でもよく使われている．
- 関数数：コードを構成する関数，サブルーチン，メソッド数等のモジュール数を数える物．モジュール分割の自由度の影響を受ける．
- クラス数：オブジェクト指向開発で用いられる．クラス設計の影響を受ける．
- サイクロマティックメトリクス：複雑度を表す指標のひとつ．ソースコードの可能なパスの数を，コードを解析して計算する．同じコードから一意に定まる測度である．保守性，理解用意性との関連が知られており，主に良形判断に利用される．

2.2.2 設計ドキュメント

設計工程完了時点での中間生産物であり，これについての測度はソースコードについての測度を見積もるために使われることが多い．主なものを以下に挙げる．

- FP(Function Point)：本来は LOC に変わるものとして，完成 (もしくは詳細設計完了後) のシステムの規模を見積もる手段として考えられたもの．以下の様な手順で見積もる：
  - 外部仕様書に基づき，内部論理ファイル、外部インターフェイスファイル等の数に重みを掛けて和をとり，調整前 FP(UFP) を求める
  - システムの特性に基づく調整係数を求め，それを UFP に掛けて，FP とする
- Cosmic-FFP：リアルタイムソフトの測定には向いていないという FP の欠点を補い，MIS ソフトやリアルタイムソフトにも適用できるように拡張された FP．ISO により標準化されている．

2.2.3 要求記述

要求分析工程完了時の中間生産物について，今日提案されているメトリクスとして UCP(Use Case Point) がある．これはユースケース図に基づく測度であって，アクタ，ユースケース，シナリオの数および複雑度に基づく測度を足し合わせ，それに環境要因の重みを掛けて算出する．

2.3 上流工程での見積もり

2.3.1 上流工程での見積もりの必要性

ソフトウェアの開発工程は一般に次の 4 つの工程からなる：要求分析、設計、コーディング、テスト．

開発計画を作成する際，コーディング量に対応できるだけの期間と要員を確保しなければ納期の達成も品質の確保もままならないため，正確な開発規模見積もりが必要になる．見積もりは通常は設計工程で作成されるドキュメントを元に行われる．しかし，一般に要員の配置計画の策定には時間がかかるため，設計工程より前に正確な規模を見積もりたいという要求が発生する．

また，開発の受託契約を考える．しばしば発注側で要求分析をすませてその要求仕様書をもとに金額の交渉が行われることがある．この場合，規模の見積もりによるコスト分析の裏付けなしに契約がおこなわれることになり，開発にかかるコストが当初予定と大きくずれる原因の一部になっているものと思われる．

2.3.2 上流工程での見積もりの問題点

そこで，さらに上流工程での見積もりが必要となるのだが，以下のような問題がある．

- ドキュメントの粒度の問題：設計工程で作成する仕様書の目的はコードを書くための入力とすることである．そこで，「コードに落とすことができる程度」の粒度で記載される．それらは構造化分析によって作成された関数仕様書やオブジェクト指向分析によって作成されるクラス図，シーケンス図，状態マトリクスなどであり，具体的にコードを起こすことができる粒度でかつ明確な構造をもって仕様書が作成される．

  しかし，要求分析で作成される仕様書は「要求を矛盾なく記載する」ことを目標に作成され．そのため，仕様の作成者には次工程のインプットとして記載の粒度をあわせるという問題意識はあっても，その目的を実現するための具体的な方法があまりない．その結果として，矛盾なく要求が記述されているものの粒度が揃わない要求記述が生ま

れることになっているものと思われる.

- ドキュメントへの記載内容の自由度の問題：要求記述の記載についてはユースケース記述等，標準化が進みつつあり，進歩的な開発現場においてはそれらの使用も報告されている．しかし，多くの現場では依然として自然言語によって記述がおこなわれているものと思われる．そのため記載内容の自由度が大きくなり，これも仕様書の粒度が揃わない結果の一因になっているものと思われる．
- 要求を実装する手段の多様性の問題：要求を満たす設計にはもともと自由度があり，例えば「ID の検索が可能であること」といったような要求についても，どの検索のアルゴリズムに何をつかうか等の自由度がある．この自由度も要求仕様からの規模見積を難しくしている．

3. 要求分析工程での見積もり手法の提案

本章では組み込みソフト開発の特徴について説明し，特に情報型組み込みソフト開発というドメインで，要求仕様書からの規模見積もりに使うことのできる可能性のある測度を提案する．

3.1 組み込みソフト開発の特徴

組み込み機器のソフト開発は一般の情報システムのソフト開発とは異なる面がある．一般の情報システムでは新規開発の終了後は保守工程に入るというライフサイクルを示す事が多い．組み込み機器のソフト開発では保守工程というものが見当たらず，新規機能の追加が次々と繰り返えされる．

さらに，組み込みソフト開発では個々のモジュールは互いに影響を持ちながらも機能としてある程度独立しており，文献[?)]で説明されるような擦り合わせ開発が行われる．そして製品戦略にしたがってほぼ定期的におこなわれる新機種の開発では，そのたびにいくつかの新機能が同様に擦り合わせられていく．この作業は開発の対象となる装置が製品としての寿命を終えるまで，例えばワープロという装置が世の中からなくなるまでワープロの機能追加による新機種の開発は繰り返えされる．

このサイクルは繰り返し行われるため，新規に追加される機能についての要求分析のためのリソースは再利用される．つまり，要求分析は入れ替えはあるものの大体同じ人員によって行われ，個別にみると差異はあるものの大体おなじ章立てとフォーマットで文書化される．大規模な組み込み開発ではこの作業は組織だって行われるため，リソースが再利用される度合いは高いものと思われる．

さらに，一般の開発では要求がフィックスされないままに開発が行われるという問題がよく聞かれるのだが，携帯電話のように要求分析を通信キャリアで行い，開発をメーカーで行うという分業が行われる場合は要求記述が省略されることはなく，また恣意的な要求記述を行うことに対する抵抗もあるようである．その結果，過去の要求仕様の章立て，フォーマットが再利用され，また内容の統一性も組織的に担保されることになるようである．

3.2 情報型組み込みソフト開発

文献[?)]で説明されるているようにモバイル情報機器の開発は従来の組み込みソフト開発とは事情がことなり，むしろ汎用システムに近い性質を示しつつある．以後，従来の組み込みシステムを「制御型」，モバイル情報機器を「情報型」と呼ぶことにする．制御型とは装置の制御が主となる組み込みソフトであり，リアルタイム制約が厳しくなる傾向があるが開発の規模は一般に小さい．それに対して情報系とは携帯端末のように人間に対する情報の表示・入力を主とする組み込みソフトであり，リアルタイム制約はあまり厳しいものにならないが開発規模は一般に大きくなる．

この分類は絶対的なものではない．例えば情報系の組み込み開発の典型と考えられる携帯端末のソフト開発においても，デバイスドライバ，テレフォニーなどいわゆる”下回り”に限って考えると制御型の開発と考えたほうが妥当である．例えば下回りの開発規模はアプリの開発規模に比して一桁も小さく，全体としてeffort の大多数を占アプリ開発が占めることとなる．そこで，携帯端末のソフト開発は情報型と考えることが妥当である．

3.3 情報型組み込みソフト開発の特徴

以後，情報型組み込みソフト開発という問題領域について考察する．その場合，下記のような傾向が見られることが知られている．

- 機能追加とすり合わせ型開発：繰り返しになるが組み込み開発では新機種の開発が定期的に行われ，それはは従来機種に対する機能追加として行われる．例えばカメラをもつ携帯電話を開発する場合，新機種のためにソフトを全て新規に作り直すということはあまり一般的でなく，従来のカメラを持たない機種に対してカメラ機能を追加する開発が行われる．ただし，カメラ機能の追加はCPU，BUS 等の競合を発生させるため，従来部分のソフトにも”すりあわせ”の為の変更が必要になる．このような開発では，おおまかにいって同じような規模の開発，つまり装置をまるまる開発するほ

ど大きくなく，装置に一つのサービスを追加する程度の新規開発が継続的に行われることになり，また要求仕様書の作成も同じ組織で継続的に作成される．そのためなのか，開発する立場でそれらの仕様書を見ていると，その章立て，記載内容及び粒度が揃っている感じられる．ここで，仕様書の粒度とは個々の記述がコードのどの程度の塊と対応がつくのかということである．

もし，この仮定が正しければ，要求仕様書の個々の文の数は開発規模と相関を持つと考えられる．

- 規格とミドルウェアの存在：先に述べたとおり情報型組み込み開発の場合，アプリケーションを実装するサービスがメールやテレビ電話等，網側でサービスが規定されているものが多く，それらの多くは 3gpp 等の標準化団体によって標準化されている．その結果，それらのサービスを端末側で実装するためのミドルウェアが利用できることが大変多い．その結果，競合や制約への注意はもちろんだが，開発で主に工数を占めるのはミドルウェアをラップするＧＵＩの開発になることが多い．

3.4 情報型組み込みソフト開発の特徴を利用した上流工程での見積もり手法の提案

3.4.1 提案の前提

上で述べたように，情報型組み込みソフト開発の場合，すり合わせ型開発であることから要求仕様書の粒度が揃う傾向があり，また顧客によって要求定義が作成されるため開発の開始時点という最上流で形式と粒度の揃った要求仕様を入手できる．

今日の一般的な開発では要求仕様書は自然言語による要求記述であることが多い．これらは説明を補足する目的の図表がふくまれることがあるが，要求記述の主体は自然言語である．要求記述は誤解のないわかりやすい文章であることが求められるため，短い文で記載することが良しとされる傾向がある．例えば今回の研究では一文の長さの平均は 23 文字であった．そこで，**要求仕様書を構成する文の数**は要求の大きさを良く表す指標として使えるのではないかと考える．

また，規格とミドルウェアの存在により，ミドルウェアをラップするＵＩの開発が主な作業となるモジュールが多い．そこで，ユースケースと開発規模がよく相関するのではないかと期待できる．

先に述べたように，今日の一般的な開発では要求仕様書は自然言語によって記載されており，要所に理解の補助のための図表が挿入されているが，それについて特に形式的なルールはない．このような仕様書からユースケースポイントを用いて規模を見積もるためには，仕様書全体を精査した上で，ユースケースを書き起こすことが必要になる．

しかし，要求仕様書には説明の補足のために画面遷移図，フローチャート，状態遷移図等の図表が使われる．一つのモジュールの要求仕様にはこれらの図が複数用いられることが多い．これらのうち，画面遷移図，フローチャートのようになんらかのシナリオを記述している図表は一つのユースケースを表現していると理解することができる．ここでフローチャートや状態遷移図等は装置内の個々のサブシステムをアクターとするような，装置内部の部分的なユースケースを表現しているものと考えられる．それらの図表は仕様全体の中で理解のポイントになる箇所で用いられているものであることを考えると，仕様書の全体量に対して一定の比率で記載されているものと考えられ，またこれまで述べたとおり仕様書の粒度が揃っていることを考えると，これらをこれらの部分的なユースケースの和は仕様全体から計算したユースケースポイントとよく相関している可能性が考えられる．

3.4.2 指標の提案

以上より，上流工程での見積もりに使用するために下記の二つの指標を提案する．

- 仕様書の文の数
- 1 仕様書中の図表をユースケース記述と見なしたユースケースポイント

以下での議論の便宜のために，これらに名前をつけることにする．前者は，仕様書の中の読点を数えることで求められる量であることから**"。数"**とよぶことにする．また，後者は**部分ユースケースのユースケースポイント**，略して**部分ＵＣＰ**と呼ぶことにする．

4. 有効性の検証

本章では実開発のデータを用いてそれを検証し，その有効性と問題点を考察する．まず，検証に利用した開発の概要を説明する．つぎにその開発でどのようなデータを採取したのかを説明する．その後にデータ間の相関をしめす．さらに，今回提案する指標だけでなく，既存のユースケースポイント法の有効性についてもこのデータで検証する．

4.1 開発の概要

検証に使用した開発について説明する．対象は情報型の組み込み開発で，すり合わせ型開発で数機種を開発してきており，その間に合計で Mstep 規模のコードを開発してきた．情報型組み込み開発では通信機能，デバイスドライバ等のいわゆる"下回り"の開発と，その上のアプリケーションの開発で事情が大きく異な

るのだが，開発工数の多くがアプリケーションの開発にかかることを考え，この研究ではアプリケーションの開発についてのみを調査の対象とする．

この開発では機能毎に要求仕様書が提供され，新規機能が追加されるとその都度新規に要求仕様書が提供され，最終的に40種類程度の要求仕様書がおこされた．これらのうち，約半数を占める下回りについての要求仕様書を除き，また端末の表示メッセージを統一を図るための文言集のように機能を要求していない仕様書についても除くと検証の対象となる仕様書は１８種類になった．そのなかからさらに，仕様書中にユースケース記述とみなせるような補足的な図表がなかったもの，その仕様が機能横断的でありコードを特定することが難しいものを除く７種類を対象とした．しかし，そのなかに端末で使用することのできるスクリプティング言語の言語仕様が含まれており，他の仕様とはけた違いに粒度が細かいため，検証の対象から外し，最終的に６種類の仕様書を検証の対象とした．

4.2 採取したデータ

要求分析工程，設計工程，コーディング工程，テスト工程の各々について以下の生産物について調査し，アプリケーション毎にデータを採取した．

- 要求分析：要求分析の生産物である要求仕様書を調査した．要求仕様書は今回の開発では顧客（キャリア）側で作成し，顧客側での受け入れテストの入力となる．すでに述べたように要求仕様書の記載内容は「～～すること.」,「～～であること.」という，自然言語による要求記述が主たるものとなる．要求仕様書の執筆者はわかりやすい記述であることを目指すため，個々の要求記述は基本的に単文から構成されることが多いため，一つの文が一つの要求を表現していることが多いと考えうる．必要と思われる箇所には補足的にシーケンス図，画面遷移図，クラス図等各種の図表が補われている．

  要求仕様書について，文言の理解を必要とせずに機械的に抽出できる量として下記の指標を計測した．

  - ". "の数：　要求を構成する文の数として"。"の数．つまり要求を構成する文の数である．
  - 部分ユースケースのUCP：　補足的な図表のうち，シーケンス図，画面遷移図等，フローチャート等「シナリオ記述」と考え得る図表（例えばクラス図，画面定義図，ワーデリング表，はシナリオ記述とは考えられないので含まない）はユースケースを記述しているものと考えることができる．これは要求全体を網羅するユースケースではなく，要求のなかの一部についてのユースケースと考えることができる．これらをいいあらわす適切な用語は存在しないため，以下では「部分ユースケース」と呼ぶ．そして，部分ユースケースについてのユースケースポイントを計算して集計する．具体的には下記のように集計した．
    - ＊ 画面遷移図：１個の画面遷移図が１個のユースケースを記載しているものとみなした．
    - ＊ シーケンス図：１個のシーケンス図が１個のユースケースを記載しているものとみなした．
    - ＊ 状態遷移図：１個の状態遷移図が１個のユースケースを記載しているものとみなした．
    - ＊ メニュー定義表：１個のメニューが１個のユースケースを記載しているものとみなした．
    - ＊ フローチャート：１個のフローチャートが１個のユースケースを記載しているものとみなした．

  ユースケースポイントを計算するためのユースケースの大中小と，環境要因についてはすべて同じ値とした．

  尚，仕様書からの図表の抽出は手動でおこなった．仕様書の構造がまったく自由であった場合，図表を自動的に抽出することは不可能ではないが難しいものと思われる．
- 設計：この開発では設計工程では各グループの裁量で各種の生産物が作成されており，UMLに基づいて設計するグループ，フローチャートを作成するグループ，ドキュメントを作成せずにヘッダファイルの注釈から仕様書を生成するグループ等，統一されていなかった．唯一，アプリの網羅的な「画面遷移図」の作成のみが全グループに義務づけらていた．顧客との間でのアプリの仕様の確認はこの画面遷移図をもとに行われるため，この図は網羅的に作成されていた．

  そこで，設計工程では指標として以下を計測した．

  - 定義された画面遷移の数
  - 画面遷移を構成する画面数

  この画面遷移をユースケースの記述と考えると，前者はユースケースの個数を，後者はユースケースの大きさを表すものと考えられる．

ユースケースは本来要求分析工程で作成されるものであるのだが，ここではそのことは問題としない．網羅的なユースケース分析から得られるユースケースポイントの指標としての良否の検証を目的とする．

ここでは二通りの方法でユースケースポイントを計算する．オリジナルな方法では，ユースケースの"大"，"中"，"小"の３通りに対応して重みをつける．この研究ではこの通常の重みつけによる計算のほかに，ユースケースの大きさを画面数に比例させる"連続重み"つけの方法でも計算してみる．そして両者の見積もり指標としての良否を検証する．

- コーディング：コーディング工程ではソースコードに対して次のメトリクスの計測を行った：総行数、実行数、関数の数、複雑度．
- テスト：テスト工程の生産物としてはテスト仕様書，テスト計画書，テスト進捗，バグ数などがあるが，この開発では記録が不十分であり完全なものがとれていたのはバグ数だけであったため，バグ数を計測した．ここで計測したバグ数とは結合テスト，システムテスト，顧客受け入れテスト，フィールドテストでのバグ数の総計である．単体テストはコーディング工程の一部と考え，単体テストでのバグ数は省いてある．

4.3 測度間の相関

上記データをモジュールの単位で集計した結果についての，指標間の相関を表??に記載する．

4.3.1 指標間の相関

表１より読み取れることを述べる．先ず，UCP はコードの指標と大変高い相関を示している．また，連続重みで計算した場合のほうがそうでない場合よりわずかに相関が高いように見えるがその差は小さく，結果として連続重みで計算することは対して意味を持っていないことが示された．部分 UC の UCP もコードの指標と高い相関を示している．。数はコードの指標と相関が有るが，バグ数との相関がない．もともとバグ数は開発対象の難しさ，開発者の経験等の環境要因の影響を受けるために開発ステップ数と単純に比例するものではないと考えられるのだが，UCP がバグ数といくらかの相関を持ち，。数がまったく相関を示さないのは興味深い．

各指標間の回帰分析の結果のうち，総行数に対する。数と部分 UCP の分析結果を表??，表??に記載する．

4.3.2 規模見積もりの残渣

規模見積もりについての回帰分析結果と実測値のグラフを図??、図??に記す。

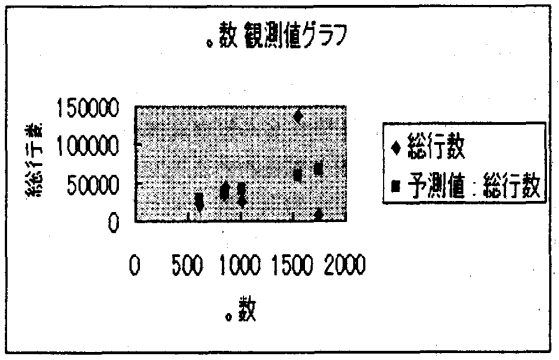

図１ 。数 - 総行数グラフ

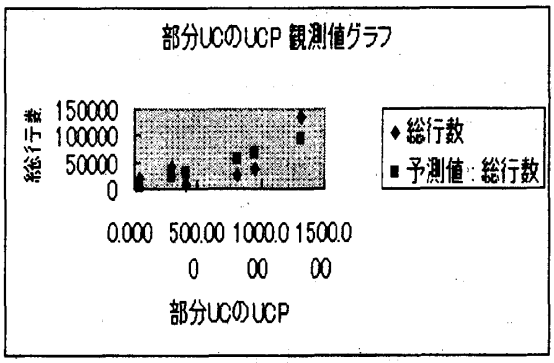

図２ 部分 UCP-総行数グラフ

仕様書の指標に対して極端に総行数の小さいモジュールが残渣の大きな原因となっていることがわかる．このモジュールは要求仕様は要求に良く合う COTS が存在したため実際の開発がほとんど発生しなかったため，開発規模が予想に対して極端に小さくなる結果になった．要求仕様を受け取った段階ではそれを満たす COTS の存在は不明であり，それは規模見積もりのずれのリスクとなることを示している．

しかし，実際の組み込み開発では，開発する機能に対して買い入れることのできるミドルウェアの有無は明確になっていることがほとんどである．したがって，それらの仕様書については規模見積もりから外すことによって正確な見積もりが可能になることが期待される．

4.3.3 バグ数の見積もり

バグ数の見積もりについての回帰分析結果と実測値のグラフを図??、図??に記す。

残渣の原因はすでに述べた COTS の存在だけが原因ではなく，結合テスト前の品質確保作業の優越，デ

表 1 指標間の相関

	要求仕様書		画面遷移図		コード				バグ
	。数	部分 UC の UCP	UCP	連続重み UCP	総行数	実業数	関数数	複雑度	バグ数
。数	1								
部分 UC の UCP	0.4143	1							
UCP	0.4349	0.7883	1						
連続重み UCP	0.3947	0.7339	0.9959	1					
総行数	0.3146	0.7285	0.9572	0.9645	1				
実行数	0.3136	0.7457	0.9598	0.9652	0.9996	1			
関数数	0.3577	0.7938	0.9855	0.9842	0.9893	0.9919	1		
複雑度	0.3281	0.7647	0.9727	0.9757	0.9974	0.9985	0.9969	1	
バグ数	0.02500	0.6744	0.7354	0.7359	0.7929	0.8036	0.8003	0.7942	1

表 2 回帰分析: 。数 - 総行数

	係数	標準誤差	t	P-値	下限 95 %	上限 95 %	下限 95.0 %	上限 95.0 %
切片	8759	58400	0.1500	0.8880	-153400	170900	-153400	170900
。数	33.00	49.79	0.6628	0.5437	-105.2	171.2	-105.2	171.2

表 3 回帰分析: 。数 - 部分 UCP

	係数	標準誤差	t	P-値	下限 95 %	上限 95 %	下限 95.0 %	上限 95.0 %
切片	-689.3	26000	-0.02651	0.9801	-72870	71490	-72870	71490
部分 UC の UCP	71.77	33.74	2.127	0.1006	-21.92	165.4	-21.92	165.4

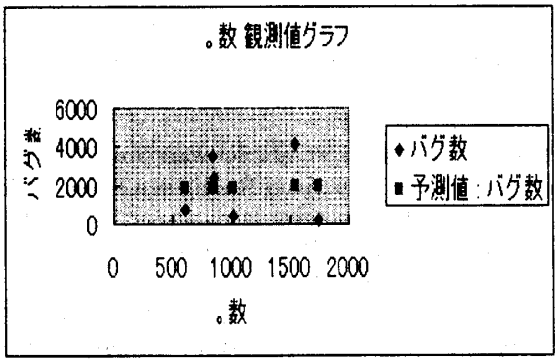

図 3 。数 - バグ数グラフ

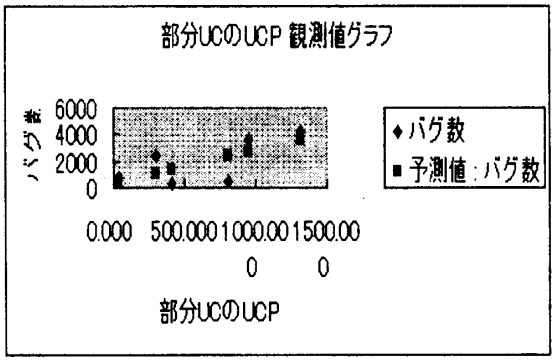

図 4 部分 UCP-バグ数グラフ.bmp

バイスドライバの品質の差異などさまざまな要因の影響を受けていることを反映していると思われる．バグ数の見積もり，つまり結合テスト，総合テスト工程での工数を仕様書から見積もることは，今回提案した指標からは不可能であることを示しているものと考えることが妥当である．

5. 関連研究

見積もりという場合，ソフトウェアの規模を何らかの数値で表現し，その値 (群) に基づいて，コストや effort を見積もる．前者としては，ソース行数 (SLOC) の様な古典的な物から，Function Point[?] や Use Case Point[?),?),?)] の様な，上流でのドキュメントからの見積もりを目指した物がある．いっぽう，後者のための手法としては COCOMOII[?)] がある．また前者の測度には規模ではなく，複雑性，品質，凝集性等をはかる物があるが，本研究では規模見積もりのみを対象としている．

Function Point 法[?),?),?),?)] は国際標準化も進んでいる手法であるが，見積もりの手法がアドホックであり，見積もるためには習熟が必要である．そのためサポートするツールも作られているが，詳細な設計が完了していないと利用する事はできない．ユースケースの普及に伴い，Function Point 法の欠点を補うために，ユースケースに基づいて規模見積もりを行う Use Case Point 法[?),?),?),?),?),?),?),?)] が考案された．しかしながらユースケースにしても最初の段階の要求文書としては敷居の高いものである．ユースケースポイントについても計測の困難さ故にサポートツールが開発されている[?),?)]

オブジェクト指向開発に限定して，クラスを測度として用いようと言う Class Point[?]，UML 図からファンクションポイントを求めようと言う手法[?]，そしてUML 図に対して独自の測度を定義しようとするもの[?]等がある.

これらの手法は基本的には対象とするドキュメント(ソフトウェア) の構文的側面に基づく物である．そのために真にソフトウェアが表現しようとしている性質等を反映していないという指摘がある．そこでドキュメントの含まれる語彙の意味を考慮するという立場から Semantic Metrics[?),?)] がある.

Capers Jones[?)] は非定型の開発ドキュメントのページ数と FP の関係及び，図表の数と FP の関係をあげているが，参考資料であり，根拠は明白ではない.

6. 結論と今後の課題

今回，提案した二つの指標は情報型の組み込みソフト開発の場合は開発規模とよく相関していることが分かった．組み込み型のソフト開発は機能追加の繰り返しであることより，初回の開発時，もしくは前回の開発時の経験値を利用することで，上記の指標を用いた規模見積りが可能であるとの印象を持つ.

今後の課題として，まず COTS として利用することができるミドルウェアが存在する場合の規模見積もり値の修正方法を明確にする必要があると考える。

更に，見積もりの最終目標は工数見積もりであり，設計工程，開発工程，テスト工程までふくめた工数見積もりに利用できるように手法を発展させていきたいと考えている。

参 考 文 献

1) Parastoo Mohagheghi, et. al.: Effort Estimation of Use Cases for Incremental Large-Scale Software Development, ICSE 2005
2) Gennaro Costagliola, et. al.: Class Point: An Approach for the Size Estimation of Object-Oriented Systems, IEEE Trans. on Software Engineering, 31, 1, 2005
3) Yue Chen, et, al.: An Empirical Study of eServices Product UML Sizing Metrics, ISESE 2004
4) Saadi Azzouz, et. al.: A Proposed Measurement Role in the Rational Unified Process and its Implementation with ISO 19761: COSMIC-FFP, SMEF 2004
5) Giovanni Cantone, et. al.: Applying Function Point to Unified Modeling Language: Conversion Model and Pilot Study, METRICS 2004
6) Shinji Kusumoto, et. al.: Estimating Effort by Use Case Points: Method, Tool and Case Study, Metrics 2004
7) Cara Stein, et. al.: Computing Software Metrics from Design Documents, ACMSE 2004
8) 松川、他: ユースケースポイント計測支援ツールの実装とその適用, 情報処理学会 ソフトウェア工学研究会, 2004-SE-144 2004
9) Massimo Carbone, et. al.: Fast&&Serious: a UML based metric for effort estimation, QAOOSE 2002
10) Hyoseob Kim, et. al.: Developing Software Metrics Applicable to UML Models, QAOOSE 2002
11) Bente Anda, et. al.: Estimating Software Development Effort based on Use Cases - Experiences from Industry, UML 2001
12) Letha Etzkorn, et al: Towards a Semantic Metrics Suite for Object-Oriented Design, TOOLS-34 2000
13) John Smith: The Estimation of Effort Based on Use Cases, Rationalte software white paper 1999
14) USC COCOMOII Reference Manual, University of Southern California, 1999
15) Luca Santillo, et. al.: Early Function Points: some Practical experiences of use, ESCOM-ENCRESS 1998
16) Martin Arnold, et. al.: Software Size Measurement and Productivity Rating in a Large-Scale Software Development Department, ICSE 1998
17) Gustav Karner: Resource Estimation for Objectory Projects, 1993
18) A.J. Albrecht, "Measuring Application Development Productivity", Proc. IBM Application Development Symp. 1979
19) 児玉 公信: 実践ファンクションポイント法, 日本能率協会マネジメントセンター 1999
20) Capers Jones 著, 富野監訳: ソフトウェア見積もりのすべて, 共立出版 2001
21) 高田 広章: 組込みソフト産業の実態と開発の課題:組込みシステム開発の要素技術と標準化, 情報処理学会会誌, Vol.46 No.4 2005
22) 奥村 洋: 組込みソフト産業の実態と開発の課題: 日本の組込みシステム開発の特徴と今後の展開, 情報処理学会会誌, Vol.46 No.5 2005