2007－SE－158 (7)
2007／11／2

ソフトウェア構成管理をベースとした再利用開発における開発規模・工数・費用見積り法

神戸英利[†] 永松博子[††] 三井浩康[††] 小泉寿男[††]
[†]三菱電機株式会社 [††]東京電機大学

再利用を伴う新規のソフトウェア開発に当たっては、最初に，作業工数や期間，必要人員ならびにそのスキル，開発費用等の見積りを実施し，開発者や費用等のリソースを確保して，開発プロジェクトを実行する必要がある．本稿では，既存ソフトウェアのモジュール間関係を可視化できる構成管理ツールを活用した見積り支援システムを提案する．本システムでは，開発範囲の見極め，洗い出したソフトウェアモジュールを基にした開発工数の予測集計による見積りを支援する．さらに本稿では，本手法を組み込みソフトウェア再利用開発に用いた評価について述べる．

An Estimation Method of the Development scale, workload and cost-volume-profit in the Reuse-based Software Development

HIDETRODHI KAMBE[†] HIROKO NAGAMATSU[††]
HIROYASU MITSUI[††] HISAO KOIZUMI[††]
[†]Mitsubishi Electric corporation [††]Tokyo Denki University

To develop a new piece of reuse-based software, the first task is to estimate the workload, the time, the human resource availability and their skill level, the reusable software resources, and the cost that is necessary for development. Consequently, the correct amounts of resources, such as development members and costs, must be prepared; then the development project can proceed. In this paper, we propose an estimate support method using a software organization management tool that can visualize the relationships among the existing software modules, and estimate data linked to each module. The method allows investigation of the existing software to select reusable modules and gathers estimate data to produce an estimate for a development workload while taking into consideration of the complexity of every software module found. Furthermore, this paper describes an evaluation when the proposed method is used in a reuse-based software development project for embedded software system.

1. はじめに

昨今のソフトウェア開発においては，できるだけ既存ソフトウェアの再利用を促進して，開発コストや稼動までの期間を最小限にするアプローチをする．そのため事前の見積り作業による事前の見極めが重要となる．また複数の開発関係者が関わる開発では，管理者単独で見積りを実施することはなく，当該プロジェクトの規模や過去に開発を分担した関連会社を含め，見積り要員をアサインして見積り作業を分担させ，それを集計して全体の見積りを行う．会社間にまたがる共同見積りが必要となるケースもある．

しかしながら，現状における多くの見積り作業は手作業で行われ，さらに定量的データを基にした見積り方法が確立していないため，見積り実施者の経験と勘によって行われている．また，主たる開発モジュールが修正された場合に，その影響を受け修正開発が必要となる関連モジュールの見積りが抜けると見積りの精度は低くなる．

さらに，途中で仕様変更が発生した場合などは，その都度見積りをやり直す必要があるが，更新作業が追いつかなくなると，これらの管理情報と開発実態が一致しなくなる．この不整合に気がつかず開発を継続すると，開発途上で費用や期間，必要リソースの確保に問題が浮上して、その解決と調整に困難を来たすことがある．これらは見積り時に開発対象のソフトウェアの中身を詳細に調査する時間が足りないため，担当者が見込みで見積りを行なうことに起因する．

筆者らは，既存ソフトウェアのモジュール間関係を可視化できる構成管理ツールを開発し，効果的な再利用を実現する手法を研究してきた[3][5]．このシステムは，開発対象のソフトウェアモジュール，または関連したソフトウェアモジュールを集めた機能ブロックに対して，関数の参照関係をもとにインターフェイスを持つソフトウェアモジュール，機能ブロックを抽出する機能を持つ．これにより，改修作業が発生する可能

性がある範囲を自動的に抽出して，開発の抜けや漏れを阻止することが可能となった．しかしながら，これまでの研究では，この方法に見積り関連モジュールの抽出を組合わせた見積り作業の支援機能は含まれておらず，見積り作業は従来の方法で行っていた．

本稿では，既存ソフトウェアのモジュール間関係を可視化できる構成管理ツールを活用し，ソフトウェアモジュールの情報に基づく開発工数の予測集計，および見積りを実現する見積り支援システムを提案する．本手法では，ソフトウェアモジュールのソースファイルと見積り用の管理情報を対応させてセットにし，モジュール間依存度をビジュアルに表示して，再利用時に修正対象のモジュールをもれなく抽出して，開発見積りを効率的により正確に行うものである．

本方式における見積り用管理情報には，担当企業，担当者，ステップ数，作業単価，技術的難易度等の見積りに必要な情報を含め，開発対象のソースファイルと対応付けて集計と保存ができるようにする．これにより既存モジュールの仕様を変更して再利用する場合，依存関係のあるモジュールを構成管理ツールで抽出することで，見積りが必要な範囲と見積りデータを開発者ならびに管理者が共有して有効に利用できる．

本稿では，2 章に本方式の提案について述べ，3章で本システムの評価，4章で考察を述べる．

2. 本方式の提案

2.1 見積り方式

本方式においては，開発のときに得られた見積り情報を，対応するソフトウェアモジュールと 1 対 1 のセットで保存する．このセットを機能ブロックで示される階層構造と連携して管理し，修正対象のソフトウェアモジュール間の連携関係を見積り情報として利用する．すなわち，直接開発対象とするソフトウェアと，これに影響を受けて修正作業が発生する関連ソフトウェアの洗い出しを支援し，開発者の開発作業範囲の絞込み，抽出作業の効率化と抜けを防止する．さらに入力した見積り情報の自動集計とリアルタイムの更新情報を管理者や見積りを行う開発者に提供して，作業の軽減を狙う．本手法では，具体的に以下の 3 つの機能によって実現する．

(1) ブロック図の利用
(2) ソフトウエアモジュールベースの見積り
(3) 連携見積り

2.1.1 ブロック図の利用

開発するソフトウェアモジュールと見積りの対応づけには，ソフトウェアのブロック図表示および関連表示機能[5]を利用する．この機能は、ソフトウェア間の関連情報を用いて、ブロック図上で矢印を表示したりブロック図の色を変化させて、ソフトウェア間、および機能ブロック間の関連性を図示するものである.これにより，開発対象のソフトウェアとそれに影響を受けて修正が必要になる関連ソフトウェア像の全体をビジュアルに確認しながら見積ることができる．

本手法では，関連解析で得られたブロック情報と見積りデータを関連付けて管理する．見積りデータは，見積りファイルと呼ぶ別ファイルで管理する．見積りデータは，ブロック情報と 1 対 1 で対応する．ブロック図と見積り設定データの関係を示した例を図 1 に示す．図において左側が関連解析で得られるブロック図の情報で，右側がそれと対応する見積りファイル内の見積り情報である．図において，Application x のブロックが Function x→Sub-func1,2 → SourceA-D で構成

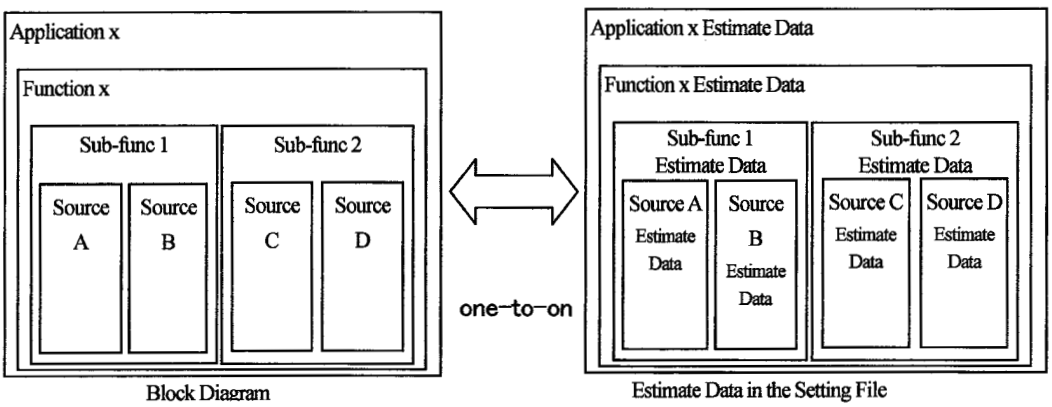

図 1　ソフトウェア構成情報に対応した見積りデータ

されている場合，それぞれのブロックに対応して見積り情報が見積りファイルに保存される．上位ブロックでは，下位階層の見積りデータが集計されて表示される．例えば，図において　Sub-func1 ブロックの見積りデータは下位の Source A，B の２つを集計したものとなり，Function x のブロックでは，Sub-func１と２を集計したデータが表示される．

2.1.2 ソフトウエアモジュールベースの見積り

本手法では開発するソフトウエアモジュール（ソースファイル）を見積り処理の基本データとする．見積りデータをそれぞれのソフトウェアモジュールのステップ数や参照関数の数などの定量的データをもとに補正を行って，見積り値を算出する．これにより，見積りデータの算出根拠を明らかにする．通常，見積り者の想定が見積りに含まれていると他人にはその見積りの算出根拠が分からない．そのため，複数人で見積りデータを共有し，確認し合うことや次のプロジェクトで見積りデータを再利用することが出来ない．見積り根拠を明らかにすることで，過去の見積りデータを参考に現時点の見積りが可能となる．

2.1.3 連携見積り

開発規模が大きい場合に，管理者単独でなく，サブリーダ以下のプロジェクトメンバーに階層的，複数に分散して見積りを実施することになる．ソフトウェアモジュール単位の見積りデータをブロック図と関連させて１つにまとめて管理し，そのデータをネットワーク等を使って共有することで，管理者ならびに見積り担当者は、現在の作業の状況と，最新の見積りデータをリアルタイムに確認することが可能となり、見積り者が離れたロケーションの開発部門にいる場合でも，共同の見積りを実施できるようになる．

このようにブロック図情報が入った関連解析データと，見積りファイルデータを見積り関係者で共有することで，見積りデータのやり取りに関する関係者間の連絡を簡素化し，管理者は上位ブロックで常に集計値を参照できるとともに，それぞれの見積り内容を直接確認して妥当性を吟味できるようになる．

2.1.4　見積り作業の流れ

見積りの流れを図２に示す．図２は，図中左側のソフトウェアＡを再利用し，ソフトウェアＢを開発する際の見積り作業の手順を示したものである．

(1)　開発内容の特定と開発環境の準備

最初に再利用予定のソフトウェアの開発環境一式を準備する．図２で示すように，既存のプロトタイプＡのソフトウェア環境をコピーして，プロトタイプＢのベースとする．また,今回開発するソフトウェアの仕様を入手する．

(2)　見積りメンバーの設定と分担の設定

管理者は，仕様からおよその規模を想定して，見積りを実施するために必要なメンバーを設定し，これらのメンバーと相談して見積り範囲の分担，期限等を決める．

(3)　追加開発と機能改修部の特定および見積り

最初にプロトタイプソフトウエアＡの関連解析を実施して，関連解析データを作成する．次に仕様からか改修開発が必要なブロックの特定と，追加が必要なブロックの追加作業を実施する．特定したソフトウェアモジュールもしくはブロック上で，開発予定のステッ

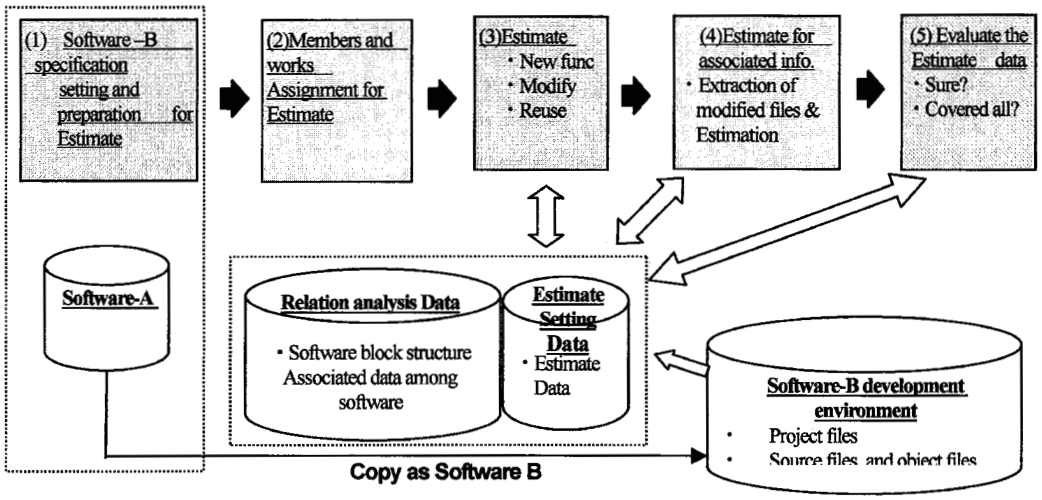

図２ ：見積りフロー

プ数，開発担当者情報，人数，開発効率，単価情報，ブロック難易度など見積りに必要なデータを入力する．見積りデータを保存することにより，関連解析データの機能ブロック，ソフトウェアモジュールの情報と１対１で対応した見積りデータが見積りファイルとして生成される．

(4) 連改修部の改修要否と規模の調査および見積り

次に，直接の開発対象となったブロックまたはモジュールそれぞれにつき，これを改修すると影響を受ける開発/修正対象のソフトウェアモジュール間の連携関係を関連表示機能を利用してブロック図上で確認し，改修が必要な場合は，見積り情報を入力する．全ての関連部のチェックと見積りデータ入力が終了したら，自ブロックの改修見積りと，関連部改修見積りならびに双方を合計した当該ブロックに関わる作業範囲全体の見積りを確認する． また,見積り入力画面には開発担当の情報を入力することができる．開発担当欄は３つに分割されており，開発グループ名，会社名，担当者名等に分類して入力することができる．上位ブロックで見たときには，下位層で入力した情報が集計されまとめて確認することができる．

(5) 見積り集計データの確認と査定

開発/修正対象で洗い出されたブロック，ソースファイルに見積りデータを入力したあと，上位の機能ブロックで見積りデータを表示すると，確認したブロックの下位に含まれる全ての見積りデータを集計した情報をみることができる．最上位のブロックで見積りを確認すると，開発するソフトウエア全体の集計値を確認できる．見積り作業の進捗はブロック図上で確認し，それぞれ分担した見積りが完了した時点で，管理者及び見積り者相互に見積り内容を確認し，妥当性を評価して見積り作業を完了する．

3. 本システムの評価

3.1 実施した見積りプロセス

3.1.1 見積りの条件

このプロトタイプは，約 5000 ファイルから構成される電子機器ソフトウェアの中から３つのアプリケーションの機能拡張に関する見積りを関連情報を利用して行った．取りまとめとサブリーダ３名の計４名で分担して見積り作業を実施した．

・管理者（とりまとめ：場所１）
－サブリーダ１（場所１）アプリケーション１
－サブリーダ２（場所１）アプリケーション２
－サブリーダ３（場所２）アプリケーション３

関連解析データと見積りファイルをネットワークで共有して，場所１，場所２の２箇所に２人づつに別れた状況で見積りを行った．４名はそれぞれＣ言語のプログラミング経験１０年以上のベテランプログラマである．

3.1.2 見積りプロセス

最初に今回の仕様を入手し，再利用する開発環境を準備した．まず，関連解析を一度実行し関連解析データを準備した．構成管理ツールの機能を使い，保守部品センターの管理ソフトウェア機能ブロックをブロック図上で追加し，主要な関数レベルの関連付けを行い概算の見積りデータを入力した．次に既存ブロックで機能修正をするブロックにつき，修正量の見積りと，影響を受けて修正が発生する関連モジュールの見積りを実施した．改修対象の，ある既存ブロックを中心に関連表示した例を図３に示す．図において，見積り時にチェックが必要な範囲を矢印表示している．

指定ブロックから見積り用画面を表示させた例を図４に示す．図において，左側はブロック図表示で指定したブロックから，ダイアログを呼び出して開発規模の見積り画面を呼び出す場面，右側は呼び出された見積り画面を示す．見積り画面のそれぞれの項目の内容を図５に示す．見積り者はこの画面上で必要な見積りデータを入力する．

機能の難易度，開発担当や単価，投入予定の人数等は，見積り時にそれぞれの見積り担当が入力した．

見積りデータを保存することで,後に見積り画面上で復元する事が可能である．基準データとして保存した見積りデータを選択すると，過去と現在の見積りデータ間の比較が可能となり，仕様変更などの影響の確認が容易になった．

作業を見積る際には，図５の改修見込みのチェックボックスにチェックを入れて，改修割合を指定すると基

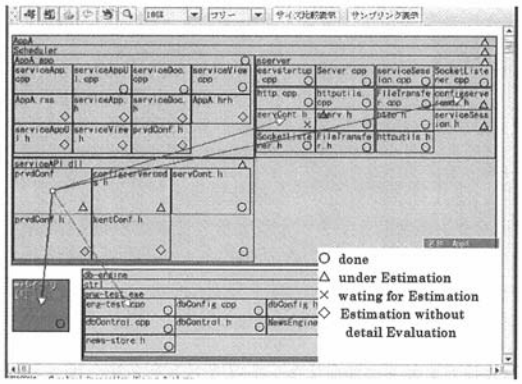

図３: 関連先ソフトウェアの表示例

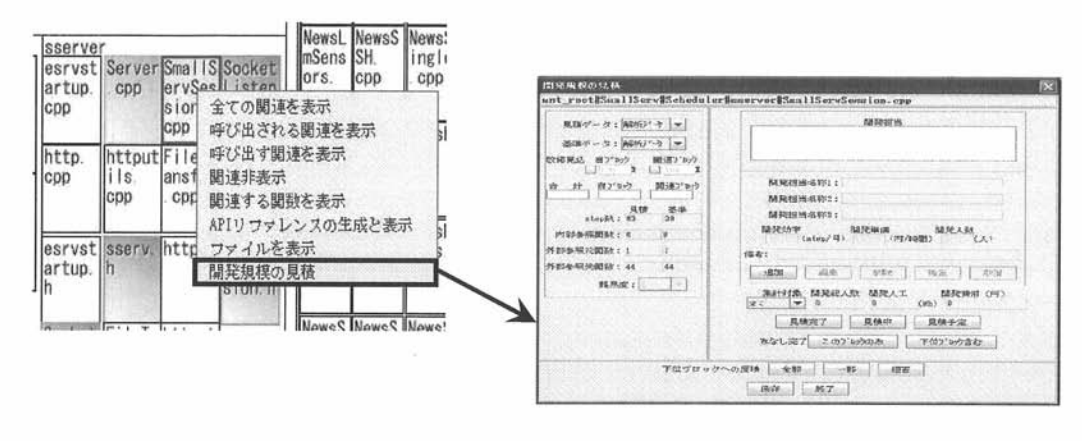

図4：ブロック図から**見積り入力画面への展開**

準データのステップ数に指定した割合分を掛けた値が見積り値として表示される．値を直接入れた場合は，入れた値が優先される．また，この時点で見積りステータス（画面右下）の「見積り予定」，または「見積り中」を指定すると，自ブロックの下位層ならびに，関連モジュールに見積りステータスのマークが表示され，見積り状況が確認できる．マークの表示される状況が図４ブロック図上に示されている．

マークは，○，△，×，◇で示され，それぞれ，

○　　：見積り完了

△　　：見積り中

×　：見積り未着手

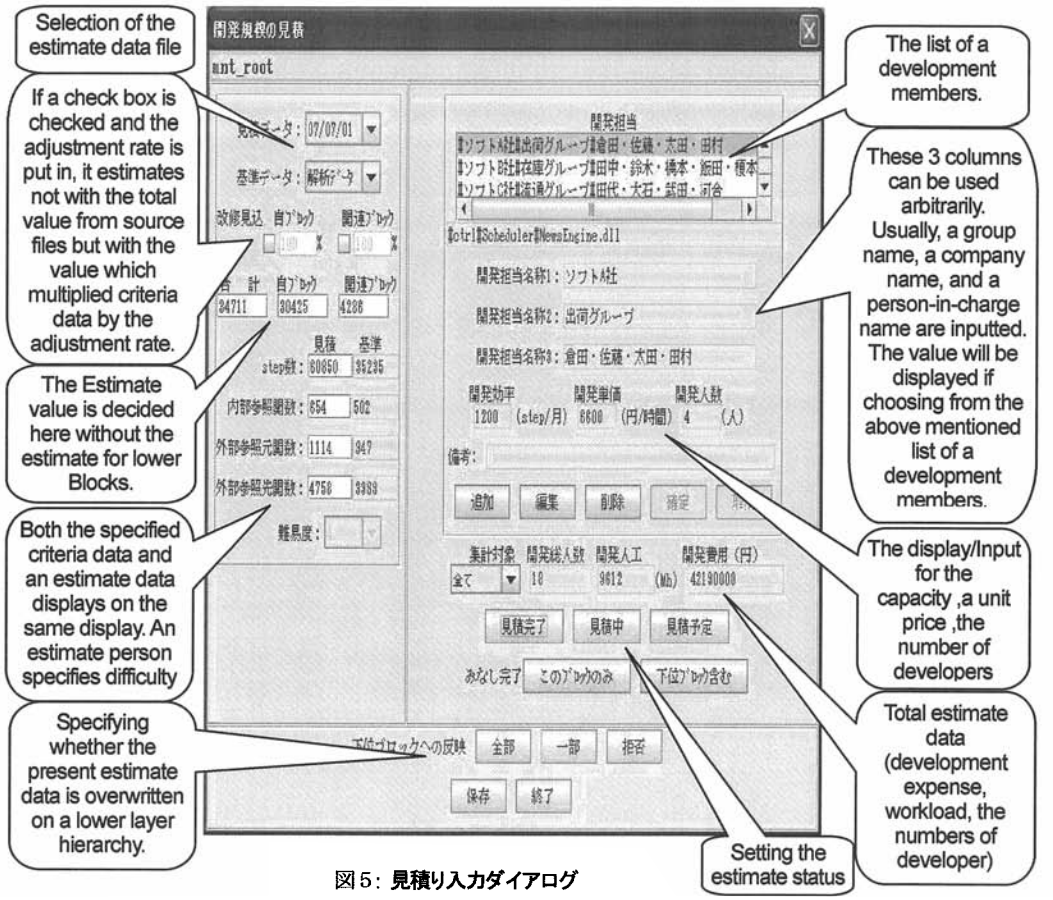

図5：**見積り入力ダイアログ**

◇　：見なし完了

見なし完了は，詳細設計まで進まないと，下位のソースファイルの構成まで確定しないケースがあるため，指定したブロック上で見積りデータを確定させるためのものである．この場合，指定したブロック配下の情報は集計せず，当該ブロックで入力したデータを用いて見積もる．今回，保守部品センター管理ブロックは，詳細設計をする前に見積りを実施したため，この機能を使い，下位ブロックは作成（設計）しなかった．
今回は，見積り着手で関連する全部ロックに×が付き，見積り作業において，作業中は△，作業完了で○をつける手順をそれぞれのサブリーダが実施した．最上位で見ていると，全ての下位層の見積りが終わるとマークが○になり，集計されているデータが有効となる．下位層の見積りの進捗に合わせて，データが逐次更新されるので，見積り作業に対するフォローが非常に簡易で済んでいる．見積り作業の結果と評価

上記見積り作業を４名で実施し，各パートの見積り作業が終了した時点で，集計値を確認した結果，82個（約28Kステップ）のソフトウエアファイルが開発対象として抽出され，開発工数7690時間の作業量，15人の必要延べ人員が算出された．これは，一人当たり200時間/月稼動で算出して，38.5人月の作業量で，15人均等に作業分担すると，2.6ヶ月（38.5/15）の開発期間となった．ロケーションが２ヵ所に分かれた状況で見積り作業を実施したが，見積りファイルとの共有により見積りは問題なく実施することができた．

次に，見積り作業に要した時間に対する評価を行った．指定したブロックを基点にして，下位ブロックと関連先を調査する作業について，従来方式と本方式との比較を行った．今回関連調査の結果，改修の要否を確認する必要があるファイルは205検出された．関連付けの確認の基となる約 1450 の関数に対し調査分析を要した．

従来方式の場合，利用予定の関数名をキーに，ソフトウェアが格納されているディレクトリ内を検索して指定した関数名が含まれる全てのファイルを抽出する．そのファイルの機能を調査しながら改修の要否と，改修が見込まれる場合は改修量を算出して行った．この作業を確認が必要と思われる関数全てに対して行った．10の関数に対して評価したところ，ひとつの関数につき，定義ファイルを特定するまでに平均しておよそ 8 分の時間を要した．これから類推すると，全1450関数の確認には，8分x1450＝11,600分（約193時間）が延べ作業時間として必要と推測される．これは今回の見積りメンバー　４人で均等に分担したとして，一人当たり37.5時間必要で，およそ６日＝約１週間の作業となる．

次に本提案の関連表示を使用した場合は，まず関連解析を実施し，その結果を見積りする者が利用した．関連解析には約30分かかった．関連解析データから関数名でファイル検索してブロック図表示し，入出力の関係から改修要否を判断するまでに平均１分半であった．全 1450 関数の確認には　３０分＋１.５分x1450＝2,205 分（約 37.5 時間）　必要であった．ここで，見積り者は計算機処理となる関連解析中には機械についている必要がないので，必要時間としては37時間となる．管理者１人＋見積り作業者３人で分担し約１日で結果が出せた．この手法で見積もる場合，見積り時の根拠（条件）がデータとして見積りファイルに残っているため，更新時にも論理的に確認して修正がしやすい．修正の規模が広範囲に渡るほど，確認すべき項目が増えるほど，関連ファイルの調査作業の軽減に効果があることが分かった．

4. 考察

(1) 的確な見積りの容易性

再利用をベースとしたソフトウェア開発において，開発対象となる機能は，どのような方法を用いても特定し，見積もることが可能である．しかし，その修正により周囲で影響を受けるモジュールを初期段階から絞り込み，見積りに含めることは一般には難しい．

本提案方式による見積りでは，開発対象になる機能部分を指定した際に，そこから関連がある機能（ブロック，ソースファイル）が自動的に抽出できるため，修正が発生しそうか否かのチェックが最初から可能となる．また，見積りをすべき範囲が示され，この作業状況を合わせて自動的に管理できるため，作業抜けが起こらず，見積りの中身に集中できる利点がある．
特に複数のスタッフで見積り作業を分担したり，その開発部隊のロケーションが離れている場合でも，見積りファイルをネットワーク等を利用して共有することで，全員で作業状況の共有が可能となる．そのため，途中で見積り条件が変わって再見積りになった場合も，状況をリアルタイムに確認できることで，作業の進捗管理を効率よく実施することが可能である．さらに，再利用時に過去の見積り情報が入っていない状況でも，見積りが必要な部分のみデータを入力すれば，当該開発において必要な見積り範囲の集計が可能であるから，見積り者に対する負担も軽くなる．管理者は，それぞれの見積り状況をブロック図上で自分の都合のつくタイミングで確認できるとともに，算出される値の妥当

性を中心にチェック出来るので，見積りの確度を高める上で効果があると考えられる．

(2) **複数担当による見積り**

複数のスタッフで見積りをした場合や，外注先に見積り依頼をした場合に，ソフトウェア情報に基く見積りとなるため，見積もられた範囲を最後にレビューすることで，見積り自体の査定が可能となる．特に，意味のない作業の上乗せや単価の上乗せ，作業時間の追加などが直感的に確認できることから，不確定要素分を必要以上に水増しすること等を抑制して妥当な見積りが可能となる．

(3) **ブロック難易度の設定**

現在，機能ブロックにより異なる難易度の違いを見積り時に共通の基準で考慮する方法がない．現状では見積り者（=判定者）の主観により決められ，ばらつきが出るため，判定のためのガイドラインもしくは設定方法の検討が必要である．自動の設定方法は，再利用を中心に考える場合ソフトウェアがすでに存在するため，その情報から自動的に算出できると良いが，現状適当な評価パラメータが見つかっていない．ファンクションポイント[8]のような点数をつける評価法も再利用中心の開発で応用ができないか検討が必要である．

(4) **総合的な考察**

サービスシステムのソフトウエア開発の規模は，今後増大してその種類も増えてくる．その背景として，再利用をベースとしたソフトウエア開発がますます進行してゆく．これらのソフトウエア開発における工数，開発期間，費用，人員確保の見積りを的確に行うことは，開発プロジェクトを成功させるために最も重要な事項の一つである．本稿で提案した手法は，基本単位であるモジュールに対応した管理情報に見積り情報を記憶させておき，修正して再利用する全てのモジュールの見積り情報を自動的に集計して見積り値を出すことを基本としている．さらに本手法では，モジュール間の相互依存性をビジュアルに表示することにより，再利用に伴って必要なモジュールをもれなく抽出する機能を持たせている．

従来は見積り対象のモジュールを手動で見つけ出してそれらの集計値を出していた．しかしながら，ソフトウエアの規模の増大とともに，モジュールの数が数千，数万に及んできた段階では人手に頼ることは不可能といえる．

このように，基本データをもとに見積り値の自動集計が可能になると，従来管理者のみが見積もり作業を行っていたものが，サブリーダや開発担当者と共同して相互調整を行いつつ見積り作業ができるようになる．一方，このことは見積りに関する意見の不一致を関係者間で招く可能性にもつながるが，見積り自体が定量的データに基いて行えるようになっている以上，各種の意見不一致は，プロジェクト管理の新たな課題として解決されるべき問題と考える．

5. まとめ

本稿では，関連解析情報に基づく関連表示を利用したソフトウェア再利用開発における共同見積り手法を提案した．本見積り手法を，電器製品の保守サービスシステムにおける保守部品管理サブシステムのソフトウエアプロトタイプ開発の見積り作業に適用して評価を行い，その有効性を確認した．今後は，本研究を継続し，次の課題に取り組んでいく予定である．

(1) 見積り対象のソフトウェアやブロック図の難易度をいかに決めていくかについての検討

(2) 本方式の見積り対象の増加による評価の実施

(3)今回の見積りは２つのサイトにまたがって行い，その集計を行ったが，今後はサイト数を３つ以上に増やし,共同見積りの効果と問題点についての評価を行う．

参考文献

[1]ソフトウェア・エンジニアリング・センター：“http://sec.ipa.go.jp/index.php”

[2]2006 年度組込みソフトウェア産業実態調査報告:“http://sec.ipa.go.jp/download/200606es.php”

[3]鷲澤暢亮，神戸英利，小泉寿男：構成管理をベースとしたソフトウェアの再利用法，平成 18 年電気関係学会関西支部連合大会，G12-4，p.G289 (2006)

[4]小池誠，並木美太郎，岩澤京子：分散環境における再利用性の高いオブジェクト作成を支援する共同開発環境の研究，情報処理学会 ソフトウェア工学会，Vol.1999，No.122-6 pp41-48 (1999-3)

[5]H.Kambe, H.Nagamatsu, H.Mitsui, H.Koizumi :A Visualizing Method of Relations between Modules to Support Re-use of Embedded Softwares, IPSJ SIG Technical Report, 2007-SE-156,PP.71-78)
神戸英利，永松博子，三井浩康，小泉寿男：組込みソフトウェアの再利用を支援するモジュール間相互関連表示法 (情学技報 2007-SE-156,PP.71-78)

[6]Saeid Nooshabadi, Jim Garside : Modernization of Teaching in Embedded System Design – An International Collaborative Project, IEEE Trans. on Education, VOL.49, NO.2 (2006)

[7]Bernard Coulange：Software Reuse Springer, usa (1997)

[8] 鶴保征城，富野寿訳，Capers Jones 著：ソフトウェア開発の定量化手法第 2 版，構造計画研究所発行

(1998)
Applied Software Measurement-Assuring Productivity and Quality Second Edition by Capers Jones.
[9] Rogen s. Pressman : Software Engineering –A Practitions Approach Sixth Edition, McGraw Hill International Edition,2005.
[1 0]Paul Clemens, Linda Northrop : Software Product Lines:Peactice and Patterns Addison-Wesley 2002