ソフトウェア設計支援ツール Perseus を用いたソフトウェア設計の理解過程の分析

本末 光[†]　　掛下 哲郎[†]

本研究の目的は，ソフトウェア設計の理解容易性を定量的に評価することである．そのために，ソフトウェア設計支援ツール Perseus を用いて被験者によるプログラムのリバースエンジニアリングを実施する．これにより，ソフトウェア設計を理解する過程が Perseus の生成したログに記録される．記録されたログを分析することで，アルゴリズムステップ，ルーチン，モジュール等のソフトウェア要素の理解に要する時間を計測する．先の研究で，我々はソフトウェアの理解容易性を計量するために理解コストの概念を提案した．ログを分析することで得た理解所要時間と理解コストの値を比較することで，理解コストの理論を検証できる．また，理解所要時間または理解コストに基づいて，理解が難しいソフトウェア要素を特定し，リファクタリング等を用いて設計を改善した場合の効果を定量的に評価できる．また，学生・技術者の理解度評価を行うこともできる．本論文では，ソフトウェア理解容易性メトリクスに関する我々の研究構想および研究計画について述べる．

Analyzing Understanding Process of Software Design using Software Design Support Tool Perseus

Hikaru Motosue[‡]　and Tetsuro Kakeshita[‡]

This research aims at quantitative evaluation of software design understandability. We perform reverse engineering experiment using software design support tool Perseus. The design understanding process of the given program is recorded as a log generated by Perseus. The time for understanding software components, such as algorithm step, routine and module, can be calculated by analyzing the log. We have proposed the notion of understanding cost in our previous research. The theory of understanding cost can be validated by comparing the calculated understanding time and the understanding cost. The understanding time or cost can be used to figure out hard-to-understand software components in a quantitative manner. It can also be used to evaluate the effect of design improvement such as refactoring. Furthermore understanding level of a student or IT professional can be evaluated. This paper describes the vision and plan of our research on software understandability metrics.

1. はじめに

ソフトウェアの大規模化に伴い，ソフトウェアの開発にかかるコストは年々増加している [9]．このため，ソフトウェアを開発する企業などではこのコストを如何にして削減するかが重要な課題となっている．一方，ソフトウェアを開発するコストの半分は保守工程で発生することが知られている．これは，既存のソースコードや設計ドキュメントを理解するためにコストがかかるためである．ソフトウェア理解に要するコストを定量的に計量することは，ソフトウェア開発全体のコストを下げる上で効果があると考えられる．

本研究ではソフトウェア設計に着目し，理解容易性を客観的に評価する．理解容易性の重要性は広く認識されているが，定性的に議論されるケースが多いため，改善が難しい状況にある．これを定量的に評価するため，ソフトウェア設計を理解する過程のログを取得し，それを分析する．本研究では，我々が開発したソフトウェア設計支援ツール Perseus を用い，被験者によるプログラムのリバースエンジニアリングを実施する．これにより，ソフトウェア設計を理解・再現する過程を Perseus の生成したログに記録する．ログにはアルゴリズムステップやルーチン，モジュールの設計過程が記録されている．また，設計過程には各操作に対応して時間が記録されている．この時間を設計過程とともに分析することで，設計を理解するために要した時間を算出できる．これにより，ソフトウェアの設計要素ごとに理解所要時間を求める．

先の研究で，我々はソフトウェアの理解容易性を計量するため，理解コスト [5][6]の概念を提案した．理解コストは認知心理学の概念に基づいて定義されており，木構造の理解過程モデルに対してコスト式を再帰的に適用することで，理解容易性を定量的に算出できる．プログラムのリバースエンジニアリングにより生成されたログを分析することで得たソフトウェア設計の要素ごとの理解所要時間と，同じプログラムの理解コストの値を比較することで，理解コストの理論を検証できる．また，理解所要時間または理解コストに基づいて，ソフトウェアの理解容易性をソフトウェア要素ごとに計量し，理解が難しい部分を特定する．これにより，ソフトウェアに対する理解のコスト分布を分析することで，理解が難しいソフトウェア要素の傾向を分析する．

また，特定のソフトウェア工学技術が適用されたプログラムと，適用されていないプログラムの間で理解所要時間等を分析することで，その技術の適用効果を定量的に評価できる．また，技術適用時と非適用時では理解過程がどのようにちがうのかの比較も行い，理解所要時間に及ぼす影響についても考察できる．

一方，理解所要時間や理解コストの分析を個人ごとに行うことにより，個人の知識・能力を分析することができると考えられる．個人の能力を分析することができれ

† 佐賀大学 工学系研究科 知能情報システム学専攻，
‡ Department of Information Science, Saga University, Japan

ⓒ2009 Information Processing Society of Japan

ば，ソフトウェア開発への効率的な人的リソースの活用や効果的な教育を施せる．

本稿の資料構成について述べる．2 節では，我々が開発したソフトウェア設計支援ツール Perseus について述べる．本研究では設計木構造の構築に要した操作情報が必要である．これをデータとして取得するため，Perseus はユーザー操作をログに記録する機能を提供する．3 節では，プログラムの理解に必要なコストを求める理論として我々が提案した理解コストについて紹介する．本研究では求められる設計の理解に要する時間との比較対象として用いる．4 節では，1 つのプログラムの設計に対する理解過程の分析について説明する．分析はプログラムをリバースエンジニアリングする過程の記録を用いる．過程を記録するために Perseus のログ機能を使用する．また，プログラム設計の理解に要した時間や設計木構造の理解コストとの対応について説明する．5 節では，プログラム設計の理解過程に関するデータを複数収集し，収集したデータを系統的な処理によって行える分析について説明する．理解コストと理解所要時間を比較することで理解コストの妥当性を検証する．また，ソフトウェアの理解容易性の定量的な評価，ソフトウェア工学技術の評価，個人や組織における知識・能力の把握を行う際に有用である．

2. ソフトウェア設計支援ツール Perseus

2.1 設計木構造の編集

ソフトウェア設計支援ツール Perseus[2]は，系統的なソフトウェア設計を支援するためのソフトウェアであり，ソフトウェア設計の編集機能，設計レビュー機能，および設計ガイドラインに関する自動チェック機能などを提供している．現在，佐賀大学・知能情報システム学科で開講している専門必修科目「ソフトウェア工学」においても，モジュール設計演習および Jackson 法による構造化プログラミング演習を行う際に Perseus を活用している．

Perseus はソフトウェア構造を木構造（設計木構造）で表現している．設計木構造の各ノードは，モジュール，ルーチン（手続き，関数），データ構造（単純データ構造，構造体，配列・リスト），制御構造（単純操作，条件分岐，ループ）等のソフトウェア設計要素に対応しており，これらを組み合わせてソフトウェア設計を表現する．これにより，モジュール設計やデータ構造設計，アルゴリズム構築など，様々なレベルの設計に対応している．

設計木構造を構築するため，表 1 の操作を提供している．選択しているノードの変更は設計木構造で編集を行うノードを選択する．この操作から，編集するノードを選択する．設計テキストの変更は選択したノードの設計情報を編集する．また，Perseus で定義している設計部分木の追加することで，新たな設計要素を追加し，設計を行う

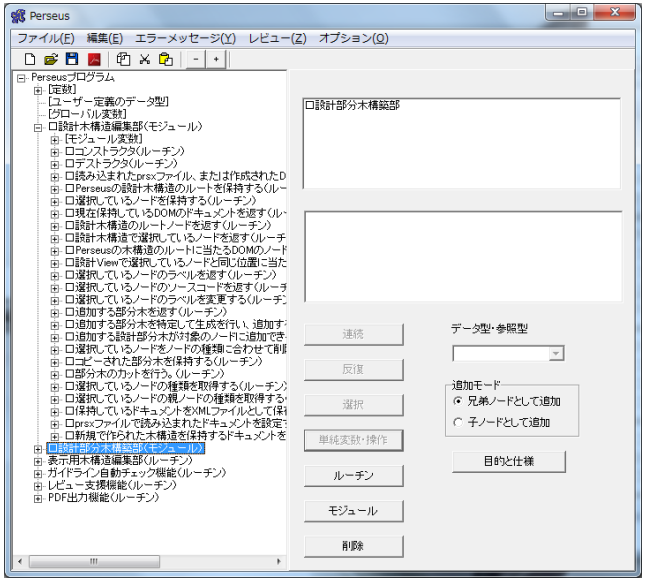

図 1 Perseus のユーザインターフェース

操作の種類	引数 1	引数 2
選択しているノードの変更	ノードの位置情報	
設計テキストの変更	変更後テキスト	
設計部分木の追加	部分木の種類	追加モード
選択しているノードのコピー・切り取り・削除		
選択しているノードの貼り付け	追加モード	
選択しているノードのデータ型・参照型の変更	変更後の内容	
選択しているノードの展開・縮約		

表 1 Perseus における設計木構造の編集機能と操作情報

ⓒ2009 Information Processing Society of Japan

ことができる．定義している設計部分木はモジュールやルーチン，データ構造である構造体，配列/リスト，共有体，単純要素，制御構造である if/else 選択，for/while 反復，switch 選択，単純操作である．この部分木の追加によって，ユーザー定義のデータ型やグローバル変数，モジュール偏すなどのデータ構造やルーチンの制御構造を木構造で設計できる．これを追加モードの切り替えによって部分木の追加方法を変えることができる．さらに構築した設計木構造はコピーや貼り付け，切り取りといった設計木構造の高水準操作により，編集が可能である．構築したデータ構造設計では，各「単純要素」ノードに対し，個別にデータ型・参照型を定義できる．

2.2 ログ出力機能

Perseus は操作をログとして記録している．記録する操作はユーザーレベルである．このため，表 1 のような操作がログとして記録される [7]．このとき，操作に応じたパラメータも一緒に記録される．また，操作を行ったときの環境も記録している．この情報もログとして追加される．操作ログは操作命令を受け取るとすぐに発行され，ログとして追加される．このログ機能により記録する情報は以下の通りである．

- 操作の種類
- 操作された時間
- 選択しているノードの木構造のフルパス
- 選択しているノードの所属しているモジュール ID
- 選択しているノードの所属しているルーチン ID
- 選択しているノードの種類
- 選択しているノードのアルゴリズムステップ
- 操作に必要なパラメータ

また，操作ログの例として図 2 を挙げる．ここではノードの選択，設計テキストの変更，設計部分木の追加を記録している．保持されている情報は左から順に，操作の種類番号，操作名，操作時間，木構造のフルパス，モジュール ID，ルーチン ID，選択しているノードの種類，アルゴリズムステップで，残りは操作に対応するパラメータである．パラメータについては表 1 を参照する．

511	設計部分木を追加	Tue Feb 07 19:03:23 2012	プログラム-ルーチン-[操作部]-入力された文字列について以下の処理を繰り返す。(for/while反復構造)-単純操作(単純操作)	0	1	単純操作	2	56	81	
501	ノードの選択	Tue Feb 07 19:03:30 2012	プログラム-ルーチン-[操作部]-入力された文字列について以下の処理を繰り返す。(for/while反復構造)-〜ならば以下の処理を行う。(if/else選択構造)-単純操作(単純操作)	0	1	単純操作	2-1-1	0:3:3:1:0:		
502	設計テキストの変更	Tue Feb 07 19:04:25 2012	プログラム-ルーチン-[操作部]-入力された文字列について以下の処理を繰り返す。(for/while反復構造)-表示している文字位置が0ならば以下の処理を行う。(if/else選択構造)	0	1	if/else選択構造	2-1	表示している文字位置が0ならば以下の処理を行う。		
501	ノードの選択	Tue Feb 07 19:04:31 2012	プログラム-ルーチン-[操作部]-入力された文字列について以下の処理を繰り返す。(for/while反復構造)-表示している文字位置が0ならば以下の処理を行う。(if/else選択構造)	0	1	if/else選択構造	2-1	0:3:3:1:0:0:		
502	設計テキストの変更	Tue Feb 07 19:04:39 2012	プログラム-ルーチン-[操作部]-入力された文字列について以下の処理を繰り返す。(for/while反復構造)-表示している文字位置が0ならば以下の処理を行う。(if/else選択構造)-文字列を表示する(単純操作)	0	1	単純操作	2-1-1	文字列を表示する。		

図 2　ログの例

設計木構造による設計木の構築は基本的に以下の手順で行われる．

1. 設計部分木の追加
2. 追加した部分木もしくはノードを選択する(自動で選択されるノードもある)
3. 設計テキストを書き換える

このため，操作ログには追加，選択，テキストの変更の順番で記録される．このため，図 2 で構築された木構造は図 3 のようになる．

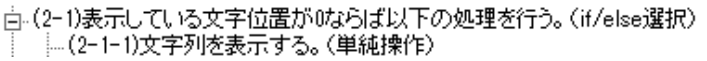

図 3　操作ログに対応する設計

このログと設計との関係を説明する．最初の操作では「if/else 選択」の設計部分木を追加している．これは子ノードに「単純操作」を持つ「if/else 選択」ノードを根とする設計部分木を，図 3 の設計木構造の(2-1)のノードに追加している．次に選択しているノードの変更を行い，追加したノード(ここでは「if/else 選択」ノード)が選択状態になる．3 つ目で設計テキストの変更を行っている．ここで「表示している文字位置が 0 ならば以下の処理を行う.」という文字列をノードの設計テキストに書き込んでいる．これで図 3 の(2-1)のノードが作成完了となる．4 つ目の操作では図 3 中の(2-1)のノードを選択している操作である．(2-1-1)と(2-1)は設計部分木が同じである．5 つ目の操作で(2-1)の設計テキストを「文字列を表示する.」

ⓒ2009 Information Processing Society of Japan

3. 理解コスト

3.1 認知心理学とソフトウェアの対応

理解コストは認知心理学に基づくソフトウェアの理解容易性の計量法として提案された[5]．理解コストはコスト式と理解過程モデルから構成される．

認知心理学では，何らかの意味で単一のまとまりを示す情報をチャンクと呼ぶ．人間の記憶装置は感覚貯蔵庫，短期貯蔵庫，長期貯蔵庫の 3 つによってモデル化される[11]．感覚貯蔵庫は，パターンマッチングを用いて単純な情報（例：文字，単語，顔）を認識する．感覚貯蔵庫における認識によって理解できるチャンクを基本チャンクという．一方，複数のチャンクから構成される，より複雑なチャンク（例：文，段落）を高水準チャンクと呼ぶ．チャンク間の関連を理解し，高水準チャンクを認識する過程を体制化と呼ぶ．短期貯蔵庫はチャンクの体制化を行うためのものである．短期貯蔵庫の記憶容量は，人間の場合 7±2 チャンクと少ない．そこで，短期貯蔵庫に格納できないチャンクは長期貯蔵庫で保持される．もし，ある情報を理解している過程において，既に知っている知識（チャンク）があった場合，チャンクの体制化を行うよりも早く知識（チャンク）を検索できるならば，チャンクの体制化コストを知識の検索コストに置き換えることも可能である．

チャンクは抽象度の異なる様々なソフトウェアの概念に柔軟に対応できる．アルゴリズム，データ構造，ルーチン，モジュール等がこれに該当する．また，文は文字や単語を構成要素とする高水準チャンクである．体制化の過程はソフトウェアの理解過程に対応する．文における体制化は，文における文字を最小単位とし，文字の集合である単語へ体制化でき，さらに単語の集合を 1 つのグループとして理解することで文という高水準チャンクに体制化できる．以上より，ソフトウェアの要素の抽象度に依存しない汎用性の高いメトリクスを定義できる．また，パラダイムにおける概念を全てチャンクとして同一として扱えるのでパラダイムの独立性が高くなる．

3.2 理解過程モデル

人間の理解過程はソフトウェアのパラダイムによって異なる．このため，人間によるソフトウェアの理解過程をモデル化した理解過程モデルはパラダイムごとに定義する必要がある．今回は Perseus が対応している構造化プログラミングと静的な OOP における理解過程モデルを示す．

構造化プログラムの理解過程モデルは図 4 の木構造で表される．理解過程モデルにおいて，体制化は下位レベルのチャンクの集合を 1 つ上のレベルのチャンクとして理解することに対応している．また，最下位レベルのチャンクは基本チャンクであり，それ以外のチャンクは高水準チャンクに対応している．構造化プログラムの構成チャンクとしてはトークン，文，モジュールプログラムなどが挙げられる．

基本チャンクは実行分野変数宣言を構成するトークン等に対応している．アルゴリズムステップはこれを構成する文の体制化により，1 つの高水準チャンクとして理解される．入れ子の文に対しては，入れ子の内側から外側へ体制化を繰り返し行うことによって理解される．入れ子のアルゴリズムステップも同様である．

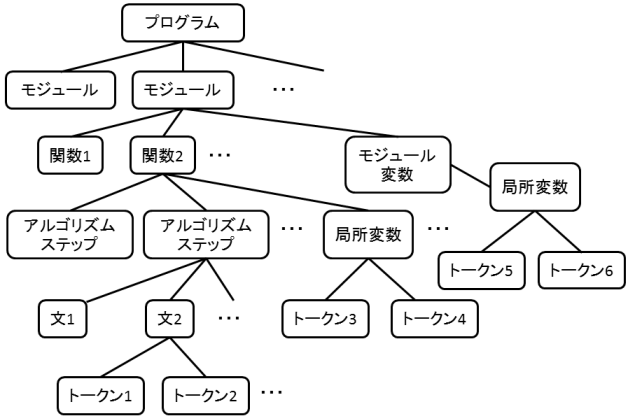

図 4　構造化プログラムの理解過程モデル

オブジェクト指向プログラム（OOP）[6]の場合には，理解過程モデルには OOP の構成要素が含まれる．このうち，メソッドの理解過程モデルを図 5 に示す．メソッドの理解過程モデルは，構造化プログラムの理解過程のうち，関数の理解過程モデルと同一である．

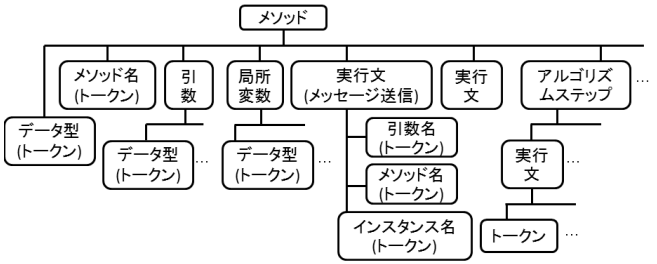

図 5　メソッドの理解過程モデル

ⓒ2009 Information Processing Society of Japan

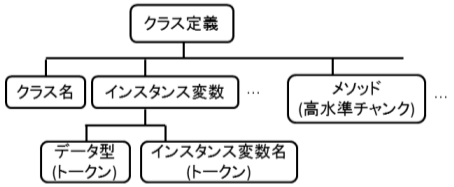

図 6　クラスの理解過程モデル

クラスの理解過程モデルを図 6 に示す．クラスはインスタンス変数，インスタンスの状態を変更するメソッド，クラス名によって定義される．メソッドの理解過程モデルに従い，さらに体制化すればクラスを理解できる．

3.3 コスト式

コスト式はソフトウェアの理解容易性の度合いを求める計算式である．理解過程モデルを元にコスト式で求められた理解コストはソフトウェアの理解に要するコストを示す．従って，ソフトウェアの理解コストが低いほど，ソフトウェアの理解容易性は高い．

3.1 節で説明した認知心理学に基づいてコスト式を定義する．この際に，前提条件として以下を仮定する．

1. チャンク間の関連としては 2 項関連のみを考える．
2. グループを構成する全てのチャンク間に関連が存在する．
3. 短期貯蔵庫の初期状態は空である．
4. 体制化された高水準チャンクは全て長期貯蔵庫内に格納される．

このとき, n 個の構成チャンクu_1~u_nからなるグループを体制化して得られた高水準チャンク U の理解コスト *C(U)*は以下のように定義される．

$$C(U) = C_R(\mathrm{n}) + \sum_{i=1}^{n} C(u_i) + 1$$
$$C_R(\mathrm{n}) = \mathrm{n} \qquad (\mathrm{n} \leqq 7)$$
$$C_R(\mathrm{n}) = C_R(\mathrm{n}\text{-}1) + \mathrm{n} - 6 \qquad (\mathrm{n} > 7)$$

*C(U)*の右辺の第一項 $C_R(\mathrm{n})$はグループ内のチャンク間の関連に対する理解コスト（関連理解コスト）である．第二項は各構成チャンクu_iの理解コストの合計である．u_iが基本チャンクであれば，$C(u_i)=0$ である．各u_iの体制化が完了していれば $C(u_i)$は定数となる．体制化が完了していなければ，自身を高水準チャンクと考えて再帰的に体制化する必要がある．第 3 項は高水準チャンク *U* を長期貯蔵庫に格納するためのコストである．

$C_R(\mathrm{n})$ の計算式は，高水準チャンクを構成するチャンク数に応じて定義される．これは短期貯蔵庫の記憶容量を反映したものである．短期貯蔵庫に全てのチャンクが収まる場合は，一括してチャンク間の関連を理解できる．よってコストは貯蔵庫へのアクセス回数 n で求められる．一方，チャンク数が 7 より大きい場合，チャンクを一括して理解できないため，短期貯蔵庫内に存在しない各チャンクと連携を理解する必要がある．n＞7 の場合は，これを考慮している．

4. 1 つのプログラムのリバースエンジニアリングに対する分析

2 節で説明した Perseus のログ収集機能を用いて，プログラムの設計の理解に要する時間を定量的に求める．4.1 節では収集するログや被験者の前提について述べる．また，Perseus の設計プロセスと理解コストの理論の対応を示す．

4.1 リバースエンジニアリングを行う際の前提条件

ソフトウェア設計の理解過程を分析するために，被験者にプログラムを割り当て，Perseus を用いてリバースエンジニアリングを実施させる．これにより，Perseus の設計木構造を用いてプログラムの設計を理解する過程を記録し，プログラムを構成する各設計要素の理解に要した時間を計測できる．

ログの収集に当たり，いくつか想定すべき条件がある．1 つは被験者についてである．これは被験者にはプログラムをリバースエンジニアリングしてもらうための予備知識が必要である．まず始めにプログラムの知識である．プログラムをリバースエンジニアリングすると言うことは，プログラムで使用されている言語を理解している必要がある．使用される言語には C++や Java が想定されている．また，2.1 節で説明し Perseus を用いてリバースエンジニアリングするため，被験者には Perseus で設計が行える必要がある．このため，被験者の前提は以下のものが挙げられる．

- リバースエンジニアリングで使用するプログラムの言語を理解している．
- Perseus で使用されている設計概念を理解している．

また，リバースエンジニアリングを行う対象プログラムについても前提を設ける．これは，系統的に設計されていないプログラムを用いた場合，設計を理解するために余分な労力を必要とするためである．そのため，条件として C++ Programming Guidelines [12]に準拠したものを想定する．これはプログラミングスタイルや 1 ルーチン 1 機能といったプログラム構築を行ううえで基本的なガイドラインを示している．なお，こうした前提を満たさないプログラムを対象とした理解容易性の計量は今後の課題とし，本論文の検討対象には含めない．

ⓒ2009 Information Processing Society of Japan

4.2 理解過程モデルと Perseus の設計木構造の対応

理解コストの計量において重要なのが理解過程モデルである．理解過程モデルは構造と振る舞いごとに定義されるもので，人間の理解の過程を表す．このため，プログラムの設計要素を理解過程モデルから Perseus の設計要素の対応を表 2 に示す．

Perseus の設計要素	構造化プログラムの要素
プログラム	プログラム
モジュール	モジュール
ルーチン	関数
モジュール変数	モジュール変数
ローカル変数	局所変数
if/else 選択，for/while 反復，switch 選択	アルゴリズムステップ
単純操作	文
構造体，配列/リスト，共有体，要素	トークン

表 2　設計木構造と構造化プログラムの対応

また，OOP の理解過程モデルとの対応を表 3 に示す．

Perseus の設計要素	OOP の要素
プログラム	
モジュール	クラス定義，クラス名
ルーチン	メソッド，メソッド名
モジュール変数	インスタンス変数(名)
ローカル変数	局所変数
引数	引数
if/else 選択，for/while 反復，switch 選択	実行文
単純操作	アルゴリズムステップ
構造体，配列/リスト，共有体，要素	トークン全般

表 3　設計木構造とオブジェクト指向プログラムの対応

この表により，構造化プログラムやオブジェクト指向プログラムをリバースエンジニアリングして得た設計木構造と理解過程モデルを対応付ける．

4.3 理解コストとプログラム設計の理解に要した時間の対応

理解コストは 3 節で説明した理解コストの理論を適用することで求められる．この理解コストとプログラム設計の理解に要した時間の対応を図る．

Perseus を用いたプログラムのリバースエンジニアリングの過程で生成されるログは設計過程を記録している．Perseus の設計木構造は設計要素ごとに分かれて構成されているため，設計木構造の各設計要素に対応している部分の構築にかかった時間を測ることで，対応する設計要素の理解にかかる時間が求められる．

Perseus で構築する設計木構造は大きく分けて 3 つのパターンにより構築される．

1 つは①ノードの追加，②ノードの選択，③設計テキストの変更，という手順で行われる設計である．これは設計要素を追加し，編集するノードを選んで設計テキストを書き換えるという形で行われる．このため，この手順の直前の操作の時間と③の時間の差が，このノードの構築にかかった時間ということになる．これをノードごとに適用することで，1 つ 1 つのノードの理解コストが求められ，部分木全体の構築にかかった時間も算出できる．この算出された時間はリバースエンジニアリングでプログラム設計の理解過程に基づいているため，プログラムの設計要素ごとの理解にかかった時間を求めることができる．

2 つ目は①ノードの選択，② 設計テキストの変更，という手順で行われる設計である．これは既に追加されたノードに対し，設計テキストの変更を行うという操作である．これは①の直前の操作時間から②の操作時間の差を求めることで理解にかかった時間を求める．

3 つ目は①ノードの選択，② 部分木のコピー，③ ノードの選択，④ 貼り付け　という手順である．これはコピーするノードを選び，コピーしたノードを貼り付けるという操作である．これはコピーされる設計部分木が構築され，これを他のノードに移すということから構築された部分木にかかる理解所要時間と上記の手順①の直前の操作から④までの時間の差を足し合わせたものが，この部分木，もしくはノードの理解所要時間となる．

上記 3 つの理解所要時間の算出は設計がうまくできたものを仮定している．これ以外で，設計の間違いに気付き，修正を行った場合などは考慮されていない．これらの場合については今後検討する．

5. 理解所要時間と理解コストの応用と活用事例

4 節で求めたソフトウェア設計における理解所要時間を求めた．それらを用いて，様々な観点から分析を行う．ここでは，そのいくつかの応用と活用方法について説明する．

5.1 理解コストの妥当性

本研究ではプログラム設計の理解過程を分析するため，2 節で説明した Perseus を使用する．Perseus の設計木構造は 3 節で説明した理解過程モデルの木構造と同様の構造

ⓒ2009 Information Processing Society of Japan

を持っているため, Perseus で記録した設計ログによって部分的にプログラムの理解に要した時間が求められる．これはモジュールやルーチンといった理解過程モデルの要素と対応して設計を部分的に時間が求められる．これは対応している設計の理解するために必要な時間として扱える．これによって定量的に求められたプログラム設計の理解に要する時間を理解コストと比較することにより，理解コストの理論の妥当性を検証する．

Perseus の設計の部分木における理解所要時間と各理解過程モデルに対応する部分的な理解コストの対応は 4.3 節で説明した. もし, 理解コストの理論が妥当であれば，この 2 つの値を比較したとき，相関関係を持つはずである．また，相関関係が薄い場合，何故相関関係が薄いのか，部分的に比較し分析することで理解過程モデルにおいて考慮されていない要因について考察する. これを基に理解コストの理論を修正する.

5.2 理解所要時間による設計の良し悪し

理解所要時間はプログラムの理解にかかった時間を指す．ここではプログラム設計の理解に要した時間として扱う．

操作ログから個別のノードの理解にかかった時間を計量できる．例えば，各モジュールやルーチンにおけるアルゴリズムとデータ構造の理解所要時間を求めることができる．これにより，部分木全体の理解所要時間と，その内訳を求めることができる．

例えば，A というルーチンがあったとすれば，これらの計算でデータ構造（引数，戻り値，ローカル変数）にどの程度のコストがかかり，アルゴリズムにどの程度のコストがかかったかが分かる．A のデータ構造にかかった理解所要時間は 40 秒，内訳で引数に 10 秒，戻り値に 10 秒，ローカル変数に 20 秒だとする．アルゴリズムは全部で 100 秒だとすれば，このルーチンの場合，アルゴリズムの理解に多くの時間がかかっていることになる．

また，ノードの数で理解所要時間を割り，ノード 1 つ辺りどれくらいのコストがかかっているのかを測る．これを各モジュールやルーチンごとに行い，どの部分木が比較的理解しにくいのか，理解しやすい設計なのかを計量できると思われる．ノード 1 つ辺りで理解所要時間が多い所は，どこにコストが集中しているのかも分析する．コストが集中するノードの構成にはどのようなところが多いか，ノードの種類ごとやデータの種類ごとに集めて分布などを作成して分析する．分析した結果，理解コストが多くかかる構造にはどのような傾向があるのか，少ない部分はどのような構造に多いのかを調査する．

全ての被験者からこれらのデータを収集し，理解所要時間をそれぞれ求める．プログラムにおいて分かりやすい設計というものは誰でも分かりやすいというものが原則である．このため，被験者によって理解所要時間に大きく差が出るケースは少ないと考えられる．よってリバースエンジニアリングの対象となった既存のプログラムの理解所要時間の平均を求めることで，対象のソフトウェア設計における理解コストを求めることが可能と予想される．

5.3 ソフトウェア工学上の技術の効果とそれらの比較

ソフトウェア工学上，リファクタリングはソフトウェア保守において重要な役割を持つ．リファクタリングにはいくつかの技法が知られているが，Perseus の理解コストを分析することでこれらの適用効果を定量的に把握する．例えば，クラスの分割を行った場合と行わなかった場合との理解コストを比較する．これにより，どのくらいの効果が出たのか，効果が無かった場合，どのような原因が考えられるかなどを分析する. 他にもメソッドの抽出やダウンキャストのカプセル化などの効果を調べる. また，5.2 節で述べたような理解しやすい設計になっているかなどを調べ，どのようなモジュールやルーチンに対し，効果が大きかった・小さかったのかを調べ，効果的なリファクタリングの適用法を考察する．

この分析項目はリバースエンジニアリングの対象となっているプログラムにリファクタリングを適用したプログラムが必要になる．また，既存のプログラム自体が小さいものもあるため，リファクタリングの種類によってはデータが収集できない可能性がある．このため，プログラムに適用するリファクタリング技術は検討する必要がある．

5.4 個人の知識・能力の分析

リバースエンジニアリング求められた理解所要時間からその人の能力を把握する．

個人を対象とした分析として行うと以下のような活用方法が考えられる．ログから設計要素ごとに理解所要時間を求めることができるので, どの部分に多く時間を要し，どの部分に時間がかからなかったのかを分析することできる．時間が余りかからなかった部分を得意とし，時間が多くかかった部分を不得意として扱うことができる．これにより，個人の得意・不得意を把握することができる．この分析に基づくことで，ある部分が得意な人にはそれに該当するような仕事を割り当てるなど，人的リソースを効率的に行うことで効果的な人員配置を行うことができる．逆に苦手なところを把握することで，効果的な教育プログラムを組んだり，仕事を依頼する場合にサポートを付けたりすることができる．

一方，集団に対する分析を行う．企業のような集団に対して分析を行うと，その会社全体における能力分布を得ることができる．これにより，各個人がどの程度の能力であるかを集団の中で評価することができる．また，能力分布からは集団全体を評価することができる. ある集団を分析したとき，どの部分に強く，どの部分に弱いのか，客観的に知ることができる. これを利用することで様々な企画を検討できる. 例えば，弱い部分を補うため，セミナーや勉強会を開くことを検討することが考えられる．ま

ⓒ2009 Information Processing Society of Japan

た，情報システム調達等の中で企業を選定する状況を考えると，能力の高い企業や仕事内容に強い企業を選択する基準にも成りえる．

6. 関連研究

本研究は実際のデータから改善を検討するエンピリカルソフトウェア工学 [3]の分野に属する．また，この分野においては，質的なアプローチ [10]と量的なアプローチ [8]が存在する．本論文が対象とするソフトウェア設計の理解過程の把握は質的アプローチに該当するが，理解所要時間のように定量的な計量は量的アプローチに該当する．このうち，我々と同じようにログを使って分析を行うものとして PKK(Problem-Product-Knowledge)法 4)が知られている．

PKK はソフトウェア設計プロセスを分析する手法である．ソフトウェア設計を実際に行った際，その設計において取捨される選択を迫られるはずである．これをソフトウェア設計の工程としてメモに残す．これを後から分析することでソフトウェア設計プロセスを定性的に分析するというものである．

本研究と異なるのは，分析する対象が異なる点がまず挙げられる．本研究は，ソフトウェア設計の理解過程を対象としているのに対し，PKK 法ではソフトウェア設計過程そのものを対象としている．また，データ収集においても大きな違いがある．Perseus は設計を行った際，自動的に操作ログを収集できるが，PKK 法では設計過程をメモで残すという形でデータを収集している．そのため，メモの取り忘れや，メモの書き方の指導にかかるコストなど，個人の力量や管理の違いによって収集できるデータが一定ではない．

ソフトウェアの理解にかかるコストの計量として知られているのが，Chidamber のメトリクス [1]である．Chidamber のメトリクスはクラスレベルでの理解にかかるコストを定義するが，インスタンスを考慮していない．このため，我々はインスタンスを考慮した静的な OOP の理解過程モデルを構築した 6)．これにより，インスタンス数の違いに対する理解容易性の変動についても説明できる．

7. 結び

本稿では Perseus を用いたソフトウェア設計の理解過程の分析の構想について説明した．プログラムをリバースエンジニアリングする過程を Perseus によって記録することで，リバースエンジニアリングにおけるプログラム設計の理解所要時間を求めることができる．これは Perseus の設計木構造と理解過程モデル，理解コストとプログラム設計に理解に必要な時間をそれぞれ対応されることで可能となる．

また，この結果を用いて様々な分析が期待できる．理解コストと理解所要時間を様々なプログラムで比較することで，理解コストの理論の検証を行うことができる．加えてプログラムの理解容易性を定量的に計測することができる．一方，リファクタリングなどのソフトウェア工学上の技術の効果を定量的に分析できる．集団においては能力分布を作成できるなどの活用方法がある．

本稿で説明した理解コストにおける理解過程モデルと Perseus の設計木構造の対応は検証できる範囲のごく一部でしかなく，動的な OOP については検証できない．また，4.3 節で説明した以外のパターンが含まれている場合，理解所要時間を正確に求めることはできない．これは編集の対象となるノードが変更されると起きるため，ノードの作成をやり直しや訂正を行うと，正確な時間を算出することは困難である．これらの課題について今後検討し，ログから理解所要時間が求める．

参考文献

1) CHIDAMBER S. R.：IEEE Trans.Softw.Eng. 20(6), 476-493, 1994
2) T. Kakeshita, T. Fujisaki：Perseus: An educational support tool for systematic software design and algorithm construction, Proc. CSEE&T，2006
3) 井上 克郎，松本 健一，鶴保 征城：実証的ソフトウェア工学環境への取り組み，情報処理学会誌，45(7)，722-728，2004
4) 中島 毅，田村 直樹，寺島 美昭：ソフトウェア設計プロセスに対する質的研究の技法の提案，情報処理学会論文誌，52(12)，3699-3714，2011
5) 山崎 直子：認知心理学的アプローチに基づくソフトウェア理解度計量法，コンピュータソフトウェア，16(6)，571-583，1999
6) 山崎 直子，掛下 哲郎：静的なオブジェクト指向プログラムに対するインスタンスを考慮した理解コストの計量法，コンピュータソフトウェア，20(4)，331-344，2003
7) 本末 光，掛下 哲郎：MVC アーキテクチャを用いたソフトウェア設計支援ツール Perseus の再設計，電気関係学会発表九州支部連合大会，2011
8) 本田 勝久：量的研究デザインの方法，中部地区英語教育学会，千葉大学，2011
9) 河村 一樹：ソフトウェア工学入門，近代科学社(2003)
10) 西條 剛央：ライブ講義・質的研究とは何か (SCQRM ベーシック編)，新曜社，2007
11) 御領 他，最新認知心理学への招待 -心の働きと仕組みを探る-，サイエンス社，1993
12) 掛下 哲郎：C++ Programming Guidelines,
http://www.cs.is.saga-u.ac.jp/syllabus/GuideLine/Cguide.html

ⓒ2009 Information Processing Society of Japan