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

山﨑 直子　松原 義継　掛下 哲郎

ソフトウェアの理解容易性を計量するためのメトリクスとして理解コストを提案する．理解コストは，認知心理学におけるチャンクや体制化の概念を利用して定義されている．n 個のチャンクで構成される高水準チャンクの理解コストは，$n \leq 7$ において $O(n)$，$n > 7$ において $O(n^2)$ である．理解コストは，特定のプログラミングパラダイムには依存せず，経験的に知られている性質を自然に説明できる．24 種類の C プログラムを用いて，20 名の被験者の理解度に対する評価値を調べた結果，理解コストとの被験者別相関が高いことがわかった．

1 はじめに

ソフトウェア開発において，保守作業の占める割合は 80% にも及ぶことが知られている [10]．大規模なソフトウェアになるほど保守の割合が大きくなる．また，保守作業の約半分が，ソフトウェアの内容を理解してバグの場所や変更箇所を特定することに費されている．つまり，ソフトウェアの開発コストを低減するためには，ソフトウェアの理解容易性を高めることが有効な解決策である．このために，理解容易性の定量的な計量が必要である．

本論文では，これまで感覚的に論じられてきた理解容易性を定量的に求めるためのソフトウェアメトリクスとして理解コストを提案し，その評価を行う[†1]．

ソフトウェアの理解とは，与えられたソフトウェアにおいて，ソフトウェアの仕様および内部構造を理解することである．

理解容易性を計量する際に注意すべきことは，ソフトウェアを理解するのは人間であるということである．すなわち，理解コストは，人間によるソフトウェアの理解過程を踏まえて定義する必要がある．

われわれは，認知心理学で用いられるチャンクや体制化の概念を活用して人間の理解過程をモデル化する．このモデルに基づいてソフトウェアの理解コストを定義する．また，理解コストの正当性を示すために，評価実験を行う．実験では，さまざまなプログラミング経験を持つ被験者に，複数の C プログラムの理解しやすさを評価させる．そして，各被験者の求めた評価値と理解コストの間の相関を求める．実験の結果，提案メトリクスは理解コストを計量する方法として有効なことが確認できた．

理解コストは，認知心理学の研究成果に基づいているため，理論的根拠が明確である．また，チャンクという形でソフトウェア開発におけるさまざまな概念を統一的にモデル化できるため，設計や実装などさまざまなレベルでの計量を行うことができる．さらに，構造化やオブ

A Software Understandability Metrics Based on Cognitive Complexity.

Naoko Yamasaki, 有明工業高等専門学校, Ariake National College of Technology.

Yoshitsugu Matsubara, 佐賀大学理工学部知能情報システム学科, Department of Information Science, Saga University.

Tetsuro Kakeshita, 佐賀大学理工学部知能情報システム学科, Department of Information Science, Saga University.

コンピュータソフトウェア, Vol.16, No.6 (1999), pp.55–67.
[論文] 1998 年 4 月 30 日受付.

†1 まぎらわしくない限り，提案メトリクスの値も理解コストと呼ぶ．

ジェクト指向といったパラダイムの異なるソフトウェアの計量にも対応することが期待できる．

これまでに提案されたメトリクスについて，以下で論じる．LOC (ソースコード行数) [6] [10]や，McCabeの閉路数複雑度 (プログラムの論理的な経路の数) [9]は，ソースコードに基づいて手続きの複雑さを計量するメトリクスである．また，分離度 (クラス内におけるメソッド及びインスタンス変数の関連の程度) [4]，継承木の深さ (継承木の根から葉までの段数) [2]，クラスの応答集合サイズ (クラス内のメソッド数とメソッド内から呼び出されるメソッド数の和) [2]などのオブジェクト指向概念に対応したメトリクスもある．さらに，われわれのメトリクスと同様に認知心理学の概念を利用した CCM [1] [5]も存在する．これらのメトリクスの問題点として，(1) 構造化設計やオブジェクト指向といった特定技術に依存するものが多い，(2) 設計や実装の経験に基づいて知られている性質を説明できないこと，があげられる．(2) であげた性質の例としては，プログラミングスタイルによる理解度の変化などがある (7 章参照)．

以下，2 章では，認知心理学の基本的事項を説明する．3 章では，理解コストを定義する．4 章では，評価実験の概要と理解コストの計算方法について述べる．5 章および 6 章では，評価実験の結果をもとに，理解コストと被験者による評価値間の相関を求め，考察を行う．7 章では，他のメトリクスとの比較を行う．8 章で，まとめと今後の課題を述べる．

2 認知心理学の基本的事項

認知心理学では，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 つに大別できることが知られている [3] [8]．

感覚貯蔵庫は人間が受け取った情報のパターン認識を行う部分であり，最も高速に処理される．感覚貯蔵庫には，非言語的な情報が大量に蓄えられるが，その内容は約 1 秒以内に消失する．そのため，必要な情報は言語的な形に変換されて短期貯蔵庫に送られる．

短期貯蔵庫は人間の理解過程の中心的な役割を担う．チャンクとは，何らかの意味で単一のまとまりを示す情報である．基本チャンクとは，感覚貯蔵庫における認識によって理解できる情報である．体制化とは，複数のチャンクからなるグループを一括理解し，それらをより高水準の単一チャンク (高水準チャンク) にまとめあげる行為を指す．短期貯蔵庫の処理は高速であるが，その記憶容量は約 7 チャンクと少ない．そのため，格納できないチャンクは長期貯蔵庫へ送られる．

長期貯蔵庫は，チャンクを単位とした情報を長期にわたって蓄える．長期貯蔵庫の処理は低速であるが，その記憶容量は事実上無限大である．

つまり，ある情報が短期貯蔵庫で理解されると，対応するチャンクが生成される．短期貯蔵庫で生成されたチャンクは長期貯蔵庫に送られ，長期にわたって記憶される．

もし，何かを理解している過程において，過去に知っている知識 (チャンク) があり，チャンクの体制化を行うよりもその知識を長期貯蔵庫から検索する方が早い場合，体制化しているチャンクとその知識を置きかえることがある．これは，体制化コストより検索コストの方が小さい場合に行われる．

チャンクは，抽象度の異なるさまざまなソフトウェアの概念に柔軟に対応できる．例えば，文，関数，モジュール，インスタンス，クラスなどの概念はすべてチャンクとして扱える．また，文は文字や単語 (トークン) を構成要素とする高水準チャンクである．ここで，文における体制化の例を示す．文においては，文字が最小の単位である．文字の集合は，意味のある 1 つのグループとして理解することで，1 つの単語に体制化される．さらに，単語の集合は，1 つのまとまりとして理解することで，文というさらに高次の高水準チャンクに体制化される．

このように，認知心理学の概念を用いることで，ソフトウェアにおける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，異なるパラダイムにおける概念をすべてチャンクとして同一に扱えるため，パラダイムとの独立性が高くなる．

3 理解コスト

3.1 ソフトウェア理解過程のモデル化

本章では体制化の概念を用いて，ソフトウェアの理解過程をモデル化する．チャンクにはプログラミングの教科書 (例： [7] [12]など) で使用される概念を用いた．

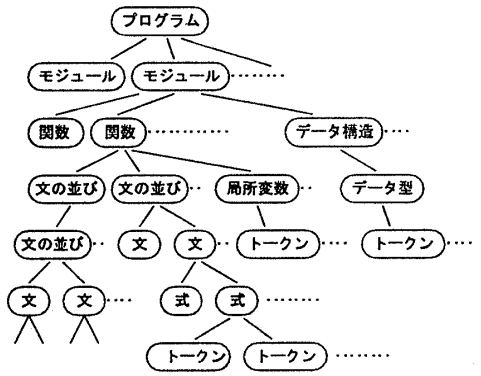

図1 構造化プログラムの理解過程

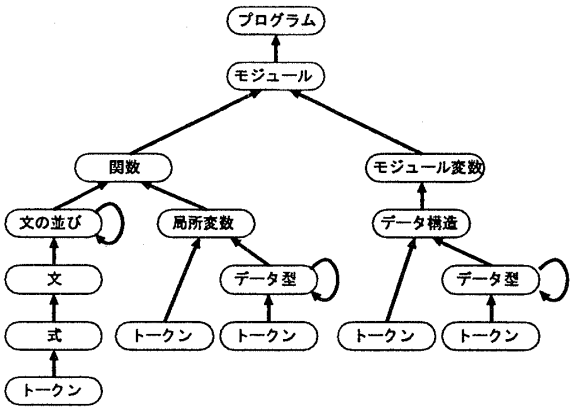

図2 構造化プログラムの理解過程 (簡略記法)

体制化は図1において，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．ここで，最下位レベルのチャンクは基本チャンクである．また，それ以外のチャンクが高水準チャンクに対応している．本論文では，体制化の過程を簡潔に示すために，図2に示す簡略記法を用いる．文の並びとデータ型は自己巡回構造を持つが，これは入れ子構造(複合文や複合データ型)を考慮したためである†2．

以上より，ソフトウェアの理解とは，体制化を繰り返し行うことによって，ソフトウェア全体に対応する単一の高水準チャンクを構成することと定義する．

ソフトウェアの理解過程をモデル化するにあたって，体制化によって生成される高水準チャンクを決定しなければならない．高水準チャンクの決定法により，複数の理解過程が存在する可能性がある．この場合，人間はコストが最小となる理解過程を経験的に選択する．最小コストは，計量対象の本質的な複雑さであると考えられるため，本論文ではこれに対応する理解過程を用いて理解コストを定義する．

3.2 理解コストの定義

複数のチャンクから構成されるグループを体制化する場合，グループに属する各チャンクの理解と，それらのチャンク間のすべての関連の理解が必要となる．チャンク間の関連を理解するためには，双方のチャンクが短期貯蔵庫内に存在する必要がある．短期貯蔵庫の記憶容量は7チャンクと小さいが，その速度は長期貯蔵庫と比較して十分高速である．そこで，われわれは，理解コストを体制化に必要なコストとし，短期貯蔵庫と長期貯蔵庫の間のアクセス回数によって定義する．これは，短期貯蔵庫と長期貯蔵庫を計算機における主記憶と二次記憶に対応づけ，ページフォルト回数を用いて性能評価を行うのと同様の考え方である．

チャンク間の関連は一般に n 項関連であるが，本論文では簡単化のため2項関連について考える．また，理解コストが最大になるのは，すべてのチャンク間に関連が存在する場合である．そこで，この場合についての理解コストを定義する．以上より，前提事項として以下を仮定する．

- 2項関連のみを考える．
- グループを構成するすべてのチャンク間に関連が存在する．
- 短期貯蔵庫の初期状態は空である．
- 基本チャンクはすべて長期貯蔵庫内に格納されている．

このようなモデルは，認知心理学において最も単純なモデル化の1つである．認知心理学では，この基本モデルの詳細構造や振舞い，記憶容量の変化等に関する研究が主に行われており，理解コストの定量化に関する研究はわれわれの知る限り存在しない．

上記の仮定に基づいて，n 個のチャンク $u_1,\cdots,u_n$ からなるグループを体制化して構成された高水準チャンク U の理解コスト $C(U)$ を以下のように定義する．

$$C(U)=\sum_{i=1}^{n}C(u_i)+C_R(n)+1 \qquad (1)$$

式(1)の右辺において，第1項は各構成チャンク u_i の理解コストの合計である．構成チャンクが内部構造を

†2 プログラミング言語によっては，モジュールや関数においても入れ子定義が存在する場合がある．

持つ高水準チャンクである場合，$C(u_i)$ を式 (1) によって再帰的に定義する．基本チャンクである場合には，$C(u_i)=0$ と定義する．第2項の $C_R(n)$ 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．これによって，1度理解された高水準チャンクはそれ以後基本チャンクとして扱えるようになる (例: `typedef` によって定義されたデータ型)．

ここで，関連理解コスト $C_R(n)$ について考察する．

まず，7個以下のチャンクから構成されるグループの理解過程を考える．これらのチャンクは短期貯蔵庫内に納まるため，一括してチャンク間の関連を理解できる．グループは，この過程によって体制化され，より高次の 1 チャンクとなる．このための関連理解コスト $C_R(n)(n \leq 7)$ は，チャンクを長期貯蔵庫から短期貯蔵庫にロードする回数，つまり体制化するグループのチャンク数 n となり，以下の式が導ける．

$$C_R(n) = n \quad (n \leq 7)$$

つぎに，8個以上のチャンクから構成されるグループの理解過程を考える．短期貯蔵庫の容量は 7 チャンクであるため，8個以上のチャンクからなるグループは短期貯蔵庫に一括して格納できない．このグループのチャンク間関連を理解する際には，短期貯蔵庫においてページフォルトが発生する．ページフォルトが発生するとチャンク入れ換え操作が必要になる．以上の考察をもとに，$C_R(n)$ $(n > 7)$ を帰納的に求める．

n 個のチャンクからなるグループを体制化するためには，(1) そのうちの $n-1$ 個のチャンクを選択して相互関連を理解し，(2) 残りの1個のチャンク x と $n-1$ 個の各チャンクの間の関連を理解する必要がある．(1) のコストが $C_R(n-1)$ であることは自明である．以下，(2) のコストを以下の手順で求める．まず，x を短期貯蔵庫にロードする．x と短期貯蔵庫内の6個のチャンクとの間の相互関連は，ページフォルトなしに求められるため，短期貯蔵庫と長期貯蔵庫間のアクセス回数は1である．つぎに，短期貯蔵庫内に存在しない $n-7$ 個の各チャンクと x の関連を理解する．そのためには，$n-7$ 個のチャンクと x を除く短期貯蔵庫内のいずれかのチャンクとを入れ換えることが必要になる．したがって，アクセス回数は $n-7$ である．以上を総合すると，以下の漸化式が導出できる．

$$\begin{aligned} C_R(n) &= C_R(n-1) + (n-6) \\ &\sim O(n^2) \quad (n > 7) \end{aligned}$$

以上の議論により，7個以下のチャンクから構成されるグループの関連理解コストは線形にしか増加しない．しかし，それ以上のチャンクから構成されるグループにおいては，関連理解コストがチャンク数の二乗に比例して増加する．

4 評価実験

3章で理解コストを定義する際に，さまざまな仮定を行った．本章では，これによる影響を検証するために評価実験を行う．

4.1 実験概要

実験では，与えられたプログラムを理解するために必要なコストを評価基準として，定義した理解コストの正当性を示す．

本実験は以下の手順で行う．

(1) 提案メトリクスを用いて，24種類の評価対象プログラムの理解コストを計算する．
(2) 各被験者に5種類のCプログラムを割り当てる．
(3) 被験者は，各プログラムの理解容易性を10段階で相対的に評価する．
(4) 被験者に対し個別に面接する．
(5) 理解コストと被験者による評価値の相関関係を調べる．

実験は20名の被験者に対して行う．各被験者にはソフトウェア工学の知識を仮定しない．20名の内訳は，佐賀大学理工学部知能情報システム学科の教官4名，大学院生4名，学部生4名，有明工業高等専門学校電子情報工学科の教官2名，5年生3名，4年生3名である．被験者のプログラミング経験は1年4ヵ月から20年以上にわたっており，講義を受けた程度からシステム開発経験者までが混在している．また，被験者の年齢も18歳から40歳までと幅広い．

```
/*** ペアの評価値を表示する. ***/
void Couple_Hyouka(void)
{
  int i;       /* 男性の番号 */
  int score;  /* 入力されたペアの評価値 */

  /* 評価値を0とする. */
  score = 0;

  /* 各男性について, ペアの女性との相性値を評価値に加算
する. */
  for(i=0;i<NUM_PAIR;i++)
    score += aishou[i][pair[i]];

  /* 評価値を表示する. */
  printf("解の評価値=%d\n",score);
}
```

図3 安定結婚問題 (パターン1)

4.2 評価対象プログラム

実験で使用する評価対象プログラムは, 8種類のプログラムを基本とする. 基本プログラムの内容は, ゲーム, データ構造の操作, テキスト処理などさまざまな分野から選択している.

まず, 基本プログラムを, 理解しやすいCプログラムを作成するためのガイドライン[†3]に従って作成する. これは文献[11]のガイドラインを簡略化したものである. ガイドラインに含まれる主要な規則を以下に示す.

- アルゴリズムのステップに対応したコメントを加える.
- 各変数, 各関数にはその機能を表すコメントを加える.
- 各変数, 各関数にはその役割を示す名前を付ける.
- 意味的なまとまりを区別するために, 空行やインデントを利用する.

これらの基本プログラムに対して, 以下の3種類のプログラミングスタイルを用いたプログラムをそれぞれ作成する.

【パターン1】 上記のガイドラインに従ったプログラムである. 例を図3に示す.

【パターン2】 パターン1のプログラムからコメントを除去したプログラムである. 例を図4に示す.

```
void Couple_Hyouka(void)
{
  int i;
  int score;

  score = 0;

  for(i=0;i<NUM_PAIR;i++)
    score += aishou[i][pair[i]];

  printf("解の評価値=%d\n",score);
}
```

図4 安定結婚問題 (パターン2)

```
void Couple_Hyouka(void)
{
int i;
int score;
score = 0;
for(i=0;i<NUM_PAIR;i++)
score += aishou[i][pair[i]];
printf("解の評価値=%d\n",score);
}
```

図5 安定結婚問題 (パターン3)

【パターン3】 パターン2のプログラムから, さらにインデント, 空行を除去したプログラムである. 例を図5に示す.

8種類の基本プログラムに対し, 以上の3パターンを適用して作成される24種類のプログラムの中から, 各被験者に異なる5種類のプログラムを割り当てる.

4.3 理解コストの計算

評価対象となるプログラムの理解コストを求めるためには, 基本チャンクと体制化の過程で生成される高水準チャンクを決定する必要がある.

まず, 基本チャンクについて考察する. パターン1のプログラムには, アルゴリズムステップに対応したコメントが付けられている. プログラムはガイドラインに従っているため, 文の並びを体制化して得られる結果とコメントの内容は一致する. よって, 文以下のレベルでの体制化は必要ない. そこで, パターン1の基本チャンクをアルゴリズムステップとする. それに対して, パ

†3 詳細は, http://www.cs.is.saga-u.ac.jp/syllabus/GuideLine/Cguide.html を参照のこと.

ターン2および3のプログラムはコメントがないため，ソースコードからアルゴリズムステップや各変数および関数の機能を理解する必要がある．ただし，すべてのプログラムの関数や変数などには意味のある名前が付けられているため，トークンレベルでの理解が可能である．よって，パターン2, 3ではトークンを基本チャンクとする．

つぎに，体制化によって生成される高水準チャンクについて考察する．プログラムの理解は，基本的に，トークン，式，文，アルゴリズムステップ，関数，プログラム全体の順に，低水準チャンクの集合を高水準チャンクへ体制化することで行う．この時，中間レベルの体制化を省略すると理解コストが増大する．その理由を以下に示す．

高水準チャンクUの構成チャンクを$u_1,\cdots,u_n$とし，各u_iの構成チャンクを$u_{i1},\cdots,u_{im}$とする．式(1)より，

$$\begin{aligned} C(U) &= C_R(n)+\sum_{i=1}^{n}C(u_i)+1 \\ C(u_i) &= C_R(m)+\sum_{j=1}^{m}C(u_{ij})+1 \\ \therefore\ C(U) &= C_R(n)+n\{C_R(m)+1\} \\ &\quad +\sum_{i=1}^{n}\sum_{j=1}^{m}C(u_{ij})+1 \end{aligned}$$

となる．それに対して，u_{ij}の体制化によって直接Uの理解コストを求めると，

$$C(U)=C_R(nm)+\sum_{i=1}^{n}\sum_{j=1}^{m}C(u_{ij})+1$$

となる．ほとんどの場合$nm>7$となるため，各u_iに対する体制化の段階(中間レベル)を設けた方が理解コストは小さくなる．したがって，体制化の過程において，高水準チャンクは中間レベルを省略しない．

ここで，パターン3の体制化の過程について考える．パターン3のプログラムには空行やインデントがないため，アルゴリズムステップを発見するのは困難である．最悪の場合，文のすべての並びについて体制化を試みることになる．これは，かえって理解コストを増大させるので，アルゴリズムステップを高水準チャンクから除外する．

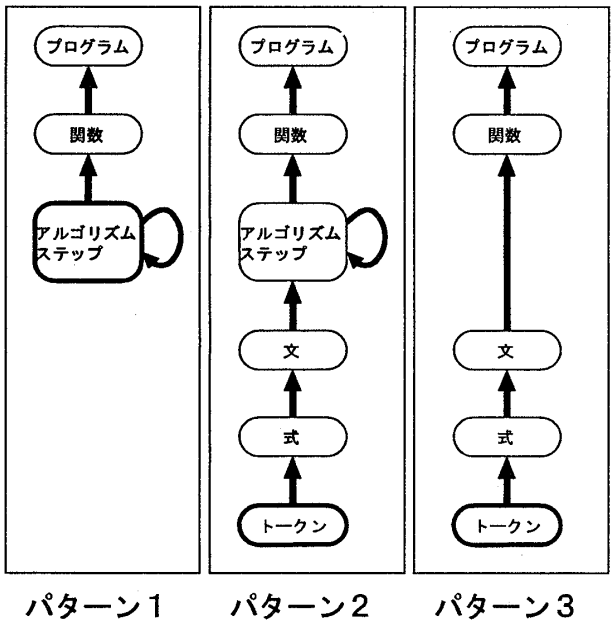

図6 プログラミングスタイル別体制化過程

このように定義すると，基本チャンクと高水準チャンクの関係はプログラミングスタイルによって異なる．プログラミングスタイル別の体制化の過程を図6にまとめる．図6のアルゴリズムステップにおける自己巡回構造は複合文を体制化する際に用いる．例えば，if文やwhile文などの制御文において，複数の文が制御文に属している場合を考える．パターン1および2では，インデントや空白を利用して制御文内のアルゴリズムステップを発見し，それらを体制化した後，制御文全体を体制化する．パターン3では，上記のようにアルゴリズムステップを発見することは困難であるため，制御文を一括して体制化する．

以上のように定義した基本チャンクと高水準チャンクに従って，各プログラムの理解コストを求める．その結果を表1に示す．ここで，A ～ Hは，基本となる8種類のプログラムである．

4.4 被験者のプログラム評価と面接

被験者は，種類やパターンの異なる5つの評価対象プログラムを渡され，各プログラムに対して，10段階で相対的に理解容易性を評価する．プログラムの評価は，きわめて理解が容易なものを0点とし，理解が困難になるにつれて点数を大きくする．理解不能なプログラムは10点とする．ここで，プログラムの理解とは，与えられたプログラムの仕様と内部構造を理解することである．

表 1 評価対象プログラムの理解コスト

	A	B	C	D	E	F	G	H
パターン	簡易電卓	五目並べ	石取りゲーム	安定結婚問題	簡易pack	集合	テーブル	txt2wd
1	415	947	199	231	109	515	530	98
2	1288	5935	435	884	302	1556	2182	150
3	2798	9175	1081	1594	702	3353	4410	384

表 2 評価対象プログラムに対する被験者の評価値 (平均値)

	A	B	C	D	E	F	G	H
パターン	簡易電卓	五目並べ	石取りゲーム	安定結婚問題	簡易pack	集合	テーブル	txt2wd
1	2.20	2.83	1.75	1.33	0.33	1.60	0.75	3.00
2	4.33	6.80	2.83	4.75	4.20	3.25	2.25	3.00
3	4.33	7.33	5.00	6.75	6.80	3.00	7.33	3.00

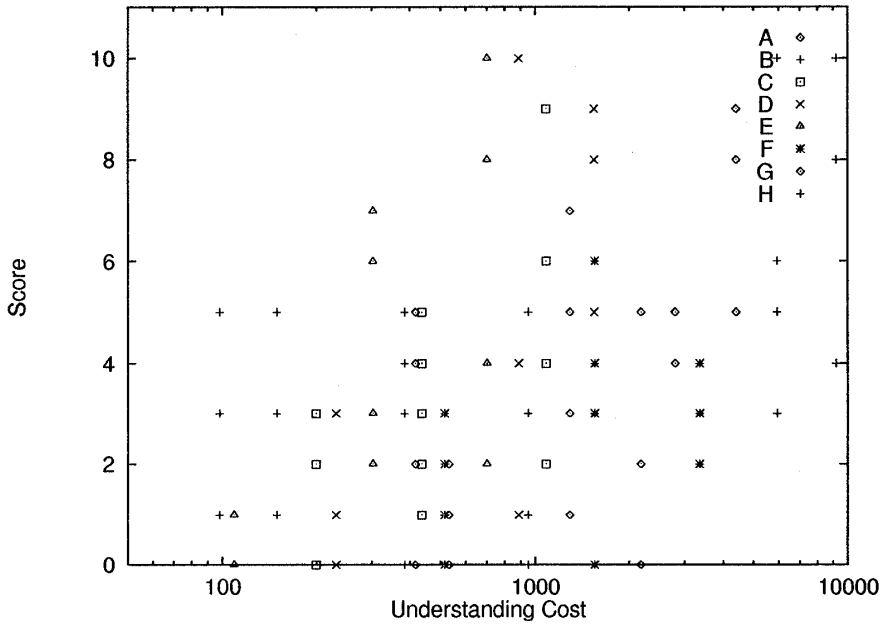

図 7 理解コストと被験者の評価値の分布

そのため，われわれは被験者に対して，与えられたプログラムの仕様だけでなく，内部構造も理解するよう指示している．

また，被験者がプログラムを理解していることを確認するために，個々に面接も行っている．面接では，プログラミング経験などの個人情報や個々の評価対象プログラムに対する理解方法をインタビューし，プログラムの内容に関する質問を行った．以下にプログラム内容に関する質問の例を示す．

(1) 与えられた入力値に対する出力値を求める．

(2) プログラムが採用しているアルゴリズムについて説明を求める．

(3) 関数の機能について説明を求める．

これらの質問に対して被験者が答えられなければ，われわれはそのプログラムに対する再評価を求める予定であったが，そのような被験者はいなかった．

5 実験結果

評価対象プログラムに対する被験者の評価値の平均値を表2に示す．また，理解コストと被験者の評価値の分布を図7に示す．

理解コストと被験者の評価値の間の相関係数は0.44となり，相関が低い．これは，さまざまな知識レベルの被験者が存在することと，評価を相対的に行ったことが原因と思われる．そこで，被験者別に理解コストと評価値との相関を調べる．その結果を図8に示す．半数の被験者の評価値は，理解コストとの相関が高い．

ところで，被験者によるプログラムの評価は10段階の値を用いて相対的に行っているため，それぞれのプログラムに対する評価値の差は被験者によって異なる．そこで，プログラムの理解容易性に対する評価順位を用いて順位相関係数を求めると0.59となる．また，被験者別の順位相関係数分布は図9のようになる．順位相関を用いた場合でも，半数以上の被験者は評価値と理解コストとの相関が高いことがわかる．

このように，単純相関と順位相関ではほぼ同様の結果が得られる．各被験者には5つのプログラムが与えられているため，それらに対する評価値の差には意味があると考えられる．例えば，プログラムa, b, cに対して2, 3, 7の評価を与えた場合，プログラムbはaよりやや理解しづらい程度だが，プログラムcはかなり難しいと判断されている．本論文では，これ以降，単純相関を主に用いて考察を行う．

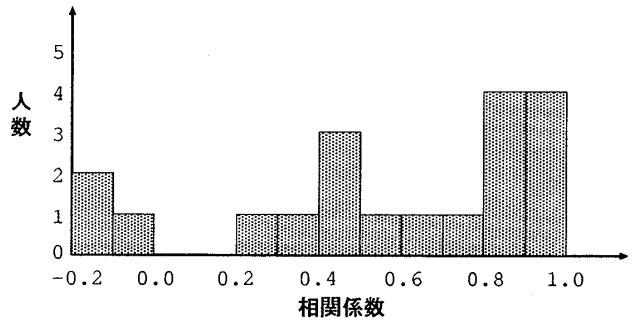

図8 被験者別相関

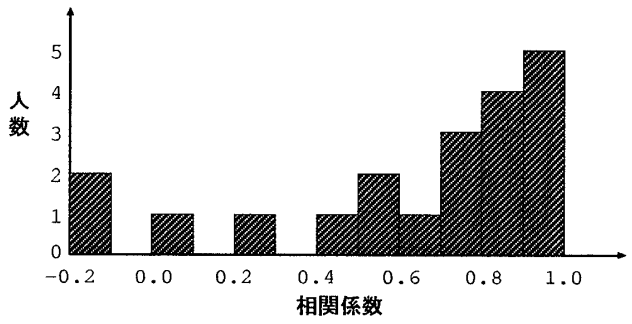

図9 被験者別順位相関

6 考察

6.1 相関係数の再計算

ここで，相関係数の値が低い被験者について考察する．これらの被験者は，簡易電卓(A)，五目並べ(B)，集合(F)，テーブル(G)に対する評価値が低い．つまり，これらのプログラムを容易に理解できると評価している．

そこで，被験者の経歴の確認やインタビューを行った結果，彼らはプログラミングやコンピュータ科学に対する経験が深く，集合，テーブルの知識を持っていることがわかった．また，プログラム中のprintf文などに出現する文字列や関数名，変数名から情報を読み取り，その情報に基づいてプログラム内容を推定していることもわかった．簡易電卓や五目並べも一般によく知られている概念なので，文字列や名前からプログラム内容が推定できたと考えられる．このような場合，上記の4つのプログラムに対しては，高水準チャンクを体制化によってではなく，長期貯蔵庫における前提知識の検索によって求めていると考えられる．この場合の理解コストは，前提知識の検索が体制化よりも低コストで行われることを考慮すると，よりコストの低いパターンの理解コストで近似される．

4つのプログラムに対する理解コストの変更を相関の低い被験者に行い，相関を再計算する．その詳細は付録A章に示す．再計算の結果，図10，11に示すように，被験者の相関係数の9割近くが0.7以上の高い相関を得る．これは相関の低い被験者に対しても，前提知識を考慮すれば評価値の説明が可能になることを意味する．

6.2 基本チャンクの重みづけ

今回の実験では，基本チャンクの種類ごと(コメント，識別子，演算子，括弧など)の重みづけを考慮していない．しかし，直観的には，コメントや識別子と比較

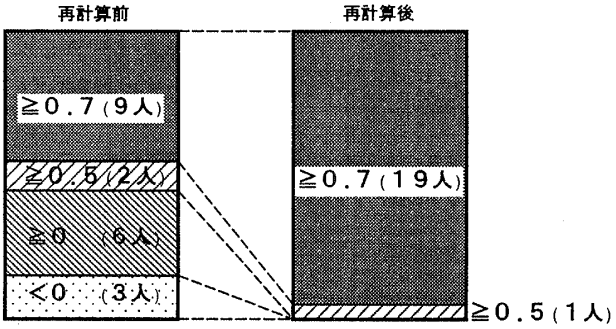

図10 再計算前後の被験者別相関の比較

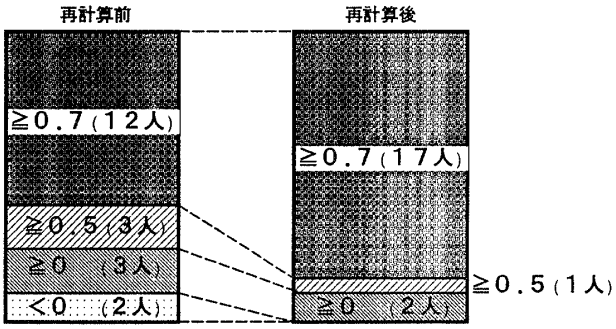

図11 再計算前後の被験者別順位相関の比較

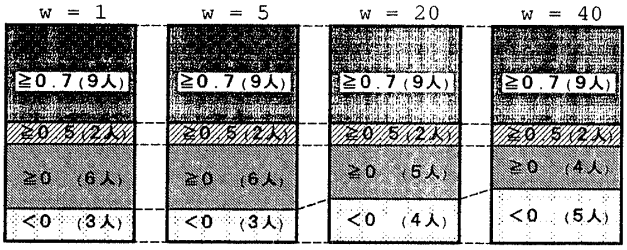

図12 基本チャンクの重みによる被験者別相関の変化

して，括弧やセミコロンなどは理解コストが低いと考えられる．そこで，コメントや識別子，予約語に対する重みをw，その他の重みを1とし，wの値を変化させて相関を調べた．その結果，図12に示すように，基本チャンクの種類ごとの重みによる影響は少ないことがわかった．これは，基本チャンク数がプログラムサイズに対して線形程度しか増加しないのに対し，チャンク間の関連理解コストC_Rは二乗に近い増加を示すためである．また，基本チャンクは，感覚貯蔵庫における認識によって理解された情報であるため理解に要するコストが低い．したがって，理解コストには，基本チャンクの重みよりも関連理解コストのほうが大きく影響する．

6.3 コメントとソースコードの一致確認

パターン1のプログラムでは，各アルゴリズムステップにコメントが付加されている．被験者がパターン1のプログラムを理解するためには，コメントとソースコードが一致していることを確認する作業と，コメントの体制化の作業が必要である．ここで，被験者がプログラムを理解していることは面接で確認している．

一方，パターン1のプログラムの理解コストを計算する際に，われわれはコメントを基本チャンクとし，その体制化コストのみを考慮している．実験の結果，理解コストと評価値との相関が高いことがわかった．したがって，コメントとソースコードが一致していることを確認するためのコストは，体制化に必要なコストよりも十分小さいと考えられる．

6.4 評価実験のまとめ

実験の結果，プログラムの理解方法として，基本チャンクに対応する概念から体制化する方法と，被験者があらかじめ持っている前提知識を利用して理解する方法の2通りがあることがわかった．前提知識を利用した場合，基本チャンクは高水準のものになるため，理解コストが低くなる．このことから，理解度の個人差は基本チャンクの差と考えられる．したがって，各個人がプログラムを理解するために要する労力を理解コストを用いて評価する際には，個人の前提知識を推定することが必要である．また，ある前提知識を想定して理解コストを計算したプログラムに対して，被験者が理解に要する労力を評価することで，個人の概念理解度や適性を調べることもできる†4．その結果は，プログラミング教育等に応用できる．また，ソフトウェア開発の役割分担を決定する際の有用な情報にもなりうる．

7 他メトリクスとの比較

本章では，従来型のメトリクスよりも理解コストのほうが実験結果を明確に説明できることを示す．比較対象として，LOC, McCabe の閉路数複雑度，CCM を用いる．

†4 プライバシーの問題に注意する必要がある．

表3 各メトリクスの評価実験結果

相関	理解コスト	LOC値	閉路数複雑度	CCM値
評価値	0.44	0.08	0.12	0.39
評価値平均	0.64	0.22	0.27	0.57

7.1 LOC・閉路数複雑度との比較

LOC (Lines Of Code) はソースコード行数によって定義されている．また，閉路数複雑度はプログラムの制御フローグラフにおける基本パス数である．これは，次式のように，制御文の数を数えることで求められる．

$$v(G) = \text{制御文の数} + 1$$

これらのメトリクスは，ソースコードに基づいて手続きの複雑さを計量する．しかし，ガイドラインに従って挿入されたコメントや空行，インデントを考慮していないため，同じ内容のプログラムであれば，プログラミングスタイルによらず同一コストになる．ここで，評価対象プログラムに対する被験者の評価値の平均値 (表2) は，パターン1の評価値が最も低く (理解が容易)，パターン3の評価値が最大 (理解が困難) である．したがって，明らかに矛盾が生じる．また，表3は，各メトリクスにおいて，評価対象プログラムのメトリクス値と被験者の評価値および評価値の平均値との相関係数を示したものである．この表より，LOC や閉路数複雑度における相関係数は低い．

以上より，これらのメトリクスは，プログラミングスタイルによる評価値の変動を説明できないことがわかる．

7.2 CCM との比較

CCM (Cognitive Complexity Model) は，理解コストと同様に，認知心理学の概念であるチャンクを活用したメトリクスである．CCM は以下の式で定義される．

$$C_i = R_i + \sum_{j \le N} C_j + \sum_{j \le N} T_j$$

$$R_S = \begin{cases} aS_i & (S_i \le L_{max}) \\ aS_i + b(\dfrac{S_i - L_{max}}{L_{max}}) & (S_i > L_{max}) \end{cases}$$

C_i は，i 番目のチャンクの複雑さを示す．R_i は i 番目のチャンクを理解するコスト，C_j は i 番目のチャンクに属する各構成チャンク (チャンク数 N) の複雑さ，T_j はトレースコストを示す．また，R_S はチャンク数に関するコスト，S_i は i 番目のチャンク数である．R_S は R_i のコストの一部である．R_i に含まれるコストとしては，他に R_C (制御構造を理解するコスト) や R_E (ブーリアン式を理解するコスト) などがある．L_{max} はチャンク数の上限，a, b は実験的に求められた定数である．

CCM と理解コストの相違点を以下に述べる．

- CCM の定義式より，CCM 値はチャンク数に対して線形に増加する．しかし，理解コストは，チャンク数が7を超えると，チャンク数の二乗に比例して増加する．
- CCM では関数間等のトレースコストを考慮しているが，トレース作業はクロスリファレンスツールなどを利用することで自動化できる．よって，理解コストでは，トレースコストを考慮していない．

われわれが実験で使用した評価対象プログラムについて CCM 値[†5]を求める (表4)．その結果，どのプログラムにおいてもパターン2よりもパターン3の CCM 値が低くなっている．これは，CCM 値は総チャンク数にほぼ比例するため，基本チャンクが特定されれば，高水準チャンク数が少ないほどメトリクス値は小さくなるためである．しかし，表2の被験者の評価値平均では，パターン2よりもパターン3のほうが評価値が高いため，CCM 値との間で矛盾が生じる．

つまり，CCM でもプログラミングスタイルによる評価値の変動を説明できない場合がある．それに対して，理解コストでは表1に示すように，これらの問題が発生しない．これは，CCM 値と評価値 (平均値) の相関係数が 0.57 であるのに対し，理解コストと評価値 (平均値) の相関係数が 0.64 と高いことにも反映している (表3)．

7.3 計算コストに関する比較

各種メトリクスに対して，メトリクス値を求めるための計算コストを比較する．各メトリクスの計算量は以下

†5 CCM 値の計算方法は付録Bを参照のこと．

表4 評価対象プログラムに対するCCM値

	A	B	C	D	E	F	G	H
パターン	簡易電卓	五目並べ	石取りゲーム	安定結婚問題	簡易pack	集合	テーブル	txt2wd
1	193.57	416.29	113.14	124.29	55.00	337.41	318.85	67.14
2	577.57	1702.10	293.14	513.36	179.42	847.24	1051.06	116.42
3	562.13	1661.08	284.71	463.86	173.42	818.24	1042.66	114.43

にあげた量に比例する.

Lines of Code (LOC) ソースコード行数

McCabeの閉路数複雑度 分岐数(条件分岐とループの総数)

CCM 基本チャンク,関数,関数呼び出しの総数

理解コスト 基本チャンクと高水準チャンクの総数

LOCにおいて,ソースコード中の任意の1文字は1つの行にしか所属しない.したがって,LOCの計算コストは,ソースコード量(文字数)の線形オーダーである.同様に,閉路数複雑度,CCMにおいて,ソースコード中の任意の1文字は,高々1つの分岐(基本チャンク,関数,関数呼び出しに対しても同様)にしか所属しない.したがって,これらのメトリクスの計算コストもソースコード量の線形オーダーである.

これに対し,理解コストにおいては,ソースコード中の1文字が複数の高水準チャンク(文,アルゴリズムステップ,関数,プログラム等)に所属する場合がある.ここで,理解過程および高水準チャンクには以下の性質がある.

- 理解過程の木構造(図1参照)において,基本チャンクは葉章点に,高水準チャンクは中間章点に対応する.
- すべての高水準チャンクは,2つ以上のチャンクを構成要素として持つ†6.

以上の2つの性質を考えると,高水準チャンク数は基本チャンク数を超えない.よって,理解コストの計算コストもソースコード量の線形オーダーである.したがって,メトリクスの計算に要するコストにおいて,大きな差はない.

†6 そうでない場合は,体制化の必要はない.

7.4 自動計量に関する比較

LOC, McCabeの閉路数複雑度においては,任意のプログラムに対するメトリクス値の自動計量が可能である.CCMでは,まず比例定数a, bの値が定義されていないため,これを与える必要がある.CCMや理解コストを自動的に計量するには,計量対象のプログラムが以下の仮定を満たす必要がある.

- 基本チャンクに対応するプログラム要素(トークン,コメントなど)が明確であること.
- 文,文の並び,関数等が高水準チャンクに対応しており,ひとまとまりの意味を持つこと.

仮定を満たさない例としては,以下の事柄があげられる.

- 複数の目的に使用される変数
- プログラム中の論理的な区切りと対応しない空行
- 複数の機能を持つ関数

4.2章であげたガイドラインは,上記の仮定を満たすために導入された.このようなプログラムに対しては,CCMおよび理解コストの計算は自動化できる.しかし,一般のプログラムではこれらの仮定が守られていることは保証されない.そこで,われわれは与えられたプログラムがガイドラインを守っているか検査するためのソフトウェアツールの構想も持っている.

8 おわりに

本論文では,ソフトウェアの理解度計量のために,認知心理学の成果に基づいて理解コストを提案した.また,理解コストの実験的評価を行った.その結果,理解コストは各被験者の評価に対して高い相関を示した.主要な誤差は被験者の知識レベルに起因する.また,他メトリクスとの比較も行った結果,これらはプログラミングスタイルによる評価値の変動を説明できないが,提案

メトリクスでは説明できることが明らかとなった．

実験で用いたパターン1のプログラムでは，アルゴリズムステップを基本チャンクとみなしている．アルゴリズムステップは設計レベルの概念なので，理解コストはソースコードだけでなく設計に対しても適用できる．また，理解コストによって，複合オブジェクトやモジュールを用いるとソフトウェアの複雑さが下がることも説明できる．

今後の課題としては，(1) オブジェクト指向プログラムに対する理解コストの評価，(2) 知識レベル (基本チャンク) の推定，(3) 変更容易性などを考慮した理解コストの改良，(4) 理解コストを理論的な基礎として各種のソフトウェア構築手法を再評価したり，新たな手法等の構築に役立てること，(5) 理解コスト計量に当たってのさまざまな仮定の除去などがあげられる．

謝辞

評価実験に協力していただいた，佐賀大学理工学部の教官と学生の方々，有明工業高等専門学校電子情報工学科の教官と学生の方々に感謝いたします．

参考文献

[1] Cant, S. N., Henderson-Sellers, B. and Jeffery, D. R. : Application of cognitive complexity metrics to object-oriented programs, *JOOP*, pp. 52–63, July–August, 1994.

[2] Chidamber, S. R. and Kemerer, C. F. : A metrics suite for object oriented design, *IEEE Trans. Softw. Eng.*, Vol. 20, No. 6, pp. 476–493, June, 1994.

[3] 御領謙, 菊地正, 江草浩幸：最新認知心理学への招待, 新心理学ライブラリ 7, サイエンス社, 1993.

[4] 白田光有：オブジェクト指向設計のためのソフトウェア評価尺度の考察, 電子情報通信学会論文誌, Vol. J79-D-I, No. 1, pp. 9–17, 1996.

[5] Henderson-Sellers, B. : *Object-Oriented Metrics: Measures of Complexity*, Prentice Hall, 1996.

[6] 本位田真一 他 編著：オブジェクト指向分析・設計 — 開発現場に見る実践の秘訣 —, 共立出版, pp. 181–196, 1995.

[7] Kernighan, B. W. and Ritchie, D. M. : *The C Programming Language, 2nd ed. Englewood Cliffs, N. J.*, Prentice-Hall, 1978. (邦訳: 石田晴久：プログラミング言語 C 第2版, 共立出版, 1989.)

[8] Loftus, G. R. and Loftus, E. F. : *Human Memory : The Processing of Information*, Lawrece Erlbaum Associates, Inc., 1976. (邦訳: 大村彰道：人間の記憶 — 認知心理学入門 —, 東京大学出版会, 1980.)

[9] McCabe, T. J. : A complexity measure, *IEEE Trans. Software Eng.*, Vol. SE-2, No. 4, pp. 308–320, 1976.

[10] McClure, C. : *The Three Rs of Software Automation: Re-engineering, Repository, Reusability*, Prentice-Hall, 1992. (邦訳: ベスト CASE 研究グループ：ソフトウェア開発と保守の戦略, 共立出版, pp. 1–53, 1993.)

[11] McConnell, S. : *Code Complete: A Practical Handbook of Software Construction*, Microsoft Press, 1993. (邦訳: 石川勝：CODE COMPLETE — 完全なプログラミングを目指して —, アスキー出版局, 1994.)

[12] 内田智史 編著：C言語によるプログラミング [基礎編][応用編], オーム社, 1992.

付録

付録Aでは，6.1における相関係数の再計算例を示す．また，パターン1より高水準な理解過程を導入する．付録Bでは，7.2における CCM 値の計算方法を示す．

A 相関係数の再計算

変更した理解コストを用いて，被験者別に相関係数を再計算した例を表5〜8に示す．

この中ではパターン0が用いられている．パターン0においては，main 関数のアルゴリズムステップとその他の関数を基本チャンクとみなして，関数，プログラム全体の順で体制化を行う．

表5 P_3 の評価値と理解コストの相関

プログラム	評価値	理解コスト	変更した理解コスト	
A3	5	2798	1288	A2
B1	3	947	947	
D2	4	884	884	
F3	2	3353	515	F1
G2	0	2182	354	G0
相関係数		-0.16	0.94	

表6 P_{20} の評価値と理解コストの相関

プログラム	評価値	理解コスト	変更した理解コスト	
A2	1	1288	163	A0
B1	0	947	414	B0
C3	9	1081	1081	
E2	3	302	302	
H1	5	98	98	
相関係数		-0.13	0.69	

表 7 P_{10} の評価値と理解コストの相関

プログラム	評価値	理解コスト	変更した理解コスト	
A3	5	2798	415	A1
D2	10	884	884	
E3	8	702	702	
F2	4	1556	515	F1
G1	0	530	354	G0
相関係数		-0.02	0.92	

表 8 P_1 の評価値と理解コストの相関

プログラム	評価値	理解コスト	変更した理解コスト	
B3	4	9175	947	B1
C2	3	435	435	
D3	8	1594	1594	
F2	0	1556	316	F0
G1	1	530	354	G0
相関係数		0.19	0.96	

パターン 1 のプログラムには，関数自身や関数呼び出しの命令にもコメントが付いており，関数の仕様が書かれている．このコメントを読むだけで関数を理解できる被験者は，関数の中身を体制化によって理解する必要がない．パターン 0 はこのような状況において理解コストを求めるために導入する．

B CCM 値の計算

われわれが実験で使用した評価対象プログラムについて，CCM によるメトリクス値を計量する．基本チャンクや体制化の過程などの条件は，すべて理解コストの評価実験と同じものを使用する．これにより，公平な比較ができる．

以下に，CCM を用いて評価対象となるプログラムの複雑さ (CCM 値) を求める方法を述べる．われわれの実験と条件を同じにするために，つぎの条件を CCM の定義に付加する．

- $R_S = R_i$ とする．簡単化のため R_S 以外の要素 (制御構造の理解コスト R_C など) は考慮しない．
- $a = b = 1$ とする．CCM 値は R_i が $O(S_i)$ であることを考えると，これは高水準チャンクを含むすべてのチャンク数と等しくなる．そのため，比例定数は相関値に影響がない．
- チャンク数の上限 L_{max} は 7 とする．

よって，実験で使用する CCM の式は，以下のように定義できる．

$$C_i = R_i + T_i$$

$$R_i = \begin{cases} S_i & (S_i \leq 7) \\ S_i + (\frac{S_i - 7}{7}) & (S_i > 7) \end{cases}$$

$$T_i = \text{関数の数} + \text{関数呼び出しの数}$$