A-026

# プログラム理解のための変数凝集度・結合度の視覚的表現

## Visualization of Variable Cohesion and Connectivity for Understanding of Program

西本 一平 †　　戸田 真志 ‡
Ippei NISHIMOTO　Masashi TODA

## 1 はじめに

プログラムの品質を図るにはいくつかの指標がある．その中の一つとして広く知られているものが，モジュールの凝集度や結合度といった概念である．凝集度に関して，従来の構造化プログラミングのパラダイムでは，機能的凝集度が考慮されているプログラムが最も望ましいものであるとされていた．しかしその後普及した抽象データ型やオブジェクト指向といった概念では，それに加えてさらにデータに関する凝集度が考慮されるべきであるとしている．データに関する凝集度が高い状態とは，各モジュールが必要最低限のデータのみを保有している状態を指す．凝集度が低い，すなわちモジュール内のデータ数が多い場合，モジュールの複雑度が上がり保守が困難となる．また結合度が低い状態とは，各データが必要最低限のモジュールによってのみ参照・更新されている状態を指す．結合度が高い，すなわち特定のデータが複数のモジュールから操作されている場合，モジュール間に依存性が発生し，独立性が低下する．

このような，機能やデータに関する凝集度，及び結合度を考慮してプログラミングを行うことは，より高品質なソフトウェアの開発につながる．

中村ら [1] は，プログラムの理解支援に必要な 5 つの要素とは「プログラムに関する知識，抽象化，資格化，理解の方法に関する知識，対話である」と定義している．そしてプログラムを抽象化した結果を人間が理解するためには，それらを視覚的に具体化することが必要であると述べている．そこで本研究ではユーザが作成したソースコードを解析し，凝集度や結合度の傾向・分布がどのようになっているかをユーザに視覚的に提示するシステムの開発を行う．

プログラムのソースコードを可視化することで情報提示や習得支援を行うという研究は既に数多く存在している．中村ら [2] は，プログラムの理解支援を行うシステムである Program Explorer(PE) を開発している．これはプログラムの実行状況やデザインパターンに関する情報を視覚化するものである．静的な要素と動的な要素を両方組み込むことで，特定のタスクに関する静的情報の切り出しを可能としている点が特徴である．また岩本ら [3] はオブジェクト指向を題材に，アニメーションを用いた動的コード可視化システムを提案している．これらのシステムではプログラムの動作状況などの把握は可能であるが，凝集度や結合度といった観点からプログラムの構造を可視化することはできない．本システムは凝集度と結合度に注目し，それがプログラム全体でどのような傾向・分布を示すかを明らかにするものである．

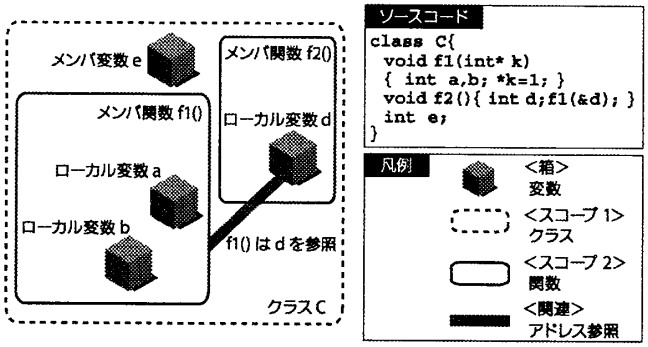


図 1　可視化の例

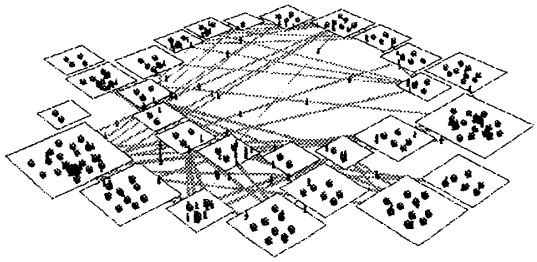

図 2　システム動作イメージ

## 2 システム概要

今回開発するシステムはユーザの作成した C/C++ ソースコードを解析し，その結果得られた構造情報を視覚的にユーザに提示するものである．これによりソースコードの凝集度・結合度がどの程度であるかを示し，また局所的に凝集度・結合度が保たれていない個所があれば，それをユーザに認識させる．

ターゲットは小規模以上のシステム設計の経験を持ち，構造の改善に目を向けられる程度のプログラミング・設計スキルを持ったユーザとする．

図 1 に示すように，このシステムではソースコード中で定義された変数は該当するスコープ内に＜箱＞として表示される．これにより各モジュール内でどれだけ変数が定義されているかをユーザに提示することができる．また変数のアドレスを外部から直接参照するモジュールには＜関連＞が張られる．具体的には始点から終点，すなわち参照元モジュールから参照先変数に向かう物体及びその軌跡によって＜関連＞が表現される．これによりソースコード中に結合度の高い変数がどの程度あり，どう分布しているかをユーザに提示する．＜関連＞は 2 要素の方向つき関係を示すものであり，表現の役割としては矢印とほぼ同様である．しかし矢印という静的表現は＜関連＞の数自体やその分布の把握が容易という長所が

† 公立はこだて未来大学大学院
‡ 公立はこだて未来大学

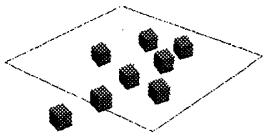

図3　初期状態

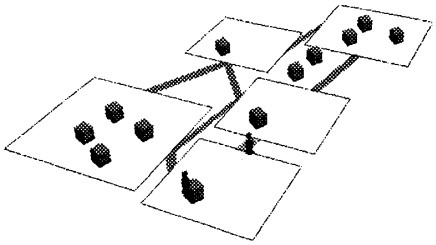

図4　機能に関する凝集度を高めた状態

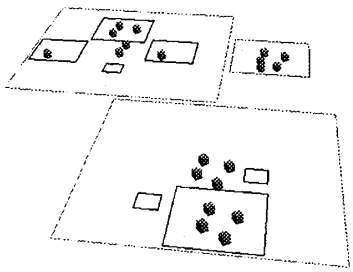

図5　データに関する凝集度を高めた状態

ある一方で，数が増すにつれて方向が瞬時に把握できなくなるという欠点を持つ．このシステムでは移動物体という動的表現と，その軌跡という静的表現を併用することで両者の長所を取り入れている．

システムの動作イメージ図を図2に示す．

## 3　実験

本システムのソースコードから1000行程度の規模のモジュールを抜き出し，凝集度や結合度を変えた3つの実装について，それぞれを可視化した．

図3は全ての処理を一つの関数に記述したソースコードを可視化したものである．図を見ると，一つの関数に7つの<箱>が密集してしていることがわかる．すなわち凝集度が低く複雑であるという傾向が読み取れる．

図4はこのプログラムを機能ごとに関数分割したソースコードを可視化したものである．関数分割したことで，モジュール内の変数の平均数は2にまで減少している．ここから，モジュール一つ一つの凝集度が初期状態よりも向上している傾向が読み取れる．また<関連>が7本と数多く現れていることから，モジュール間の依存性が高いという傾向が読み取れる．この依存性は関数を分割したことで，複数の関数が共通のデータを参照する必要が生じたために現れたものである．

図5はデータに関連する操作をまとめ，クラス化したソースコードを可視化したものである．関連は1本となり，図4で生じていた依存性の問題が解決していることがわかる．これら一連の視覚表現を通して見ると，本システムの狙いである，凝集度や結合度が与える視覚表現の違いが達成できていると考えられる．

その他，数百行から数万行まで様々な規模のソースコードを可視化した．この結果，本システムは極端に小規模なプログラムには有効に機能しないことがわかった．凝集度や結合度を考慮してモジュール分割を行うには，分割に要するオーバヘッド分のコードが必要となる．小規模なプログラムでは全体のうち，それらのコードが占める割合が大きくなってしまい，一関数だけに全処理を記述したようなソースコードがかえって良い結果を示してしまう．

また数万行を超えるような大規模なプログラムでは，その規模に伴って視覚表現も複雑化してしまう．複雑度がある程度に達した場合，凝集度や結合度がもたらす可視化表現の複雑さが，プログラムの規模による複雑さに埋もれてしまうという欠点がある．しかしプログラムの規模が大きい場合，仕様レベルでの構造は明らかになっている場合が多いものの，より実装寄りのレベルでプログラムがどのような構造になっているかを把握することは一般に困難である．集団開発を行っているような場合は特にこの傾向が顕著であり，自分以外の人間の書いたソースコード，あるいは膨大なプログラム全体のソースコードの構造を把握し，評価するのは困難である．このシステムでは全体の構造を高い一覧性で視覚化することが可能であるため，凝集度や結合度の傾向・分布や局所的な構造の欠点などを容易に把握することができる．

## 4　まとめ

本研究ではソースコードの可視化により，ソースコードの構造把握を支援するシステムを提案・実装した．

また開発したシステムを用いていくつかのソースコードの可視化を行い，システムの視覚表現を確認する実験を行った．その結果，ソースコードのデータやモジュールの構造の把握が容易に可能であること，凝集度や結合度が感覚的に提示されていること，極端に小規模なプログラムには向かないこと，などの考察が得られた．

今後の展開としては，まずは視覚表現にさらなる改善を加える必要がある．例としては，関係のあるスコープ同士の距離がなるべく均一化されるよう，配置の最適化を行うという点が挙げられる．また時間経過に応じて変化する何らかの動的な要素をアニメーションとして視覚表現するという展開も考えている．

## 参考文献

[1] 中村 宏明，安田 和，大平 剛，三ツ井 欽一，“オブジェクト指向ソフトウェア開発におけるプログラム理解支援”，情報処理学会研究報告書，プログラミング，No.15，1994

[2] 中村 祐一，“オブジェクトの視覚化を利用したプログラム理解支援”，電子情報通信学会論文誌，Vol.J79-D-I，No.10，pp738-744，1996

[3] 岩本 裕明，酒井 三四郎，“アニメーションを用いたオブジェクト指向言語プログラムの学習支援”，教育システム情報学会論文集，pp137-138，2002