社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
KBSE2011-62 (2012-01)

受動的コードレビューのためのフローチャート自動生成手法

廣瀬　義実† 　服部　　峻† 　久保村千明†† 　亀田　弘之†

† 東京工科大学コンピュータサイエンス学部　〒 192–0982 東京都八王子市片倉町 1404–1
†† 山野美容芸術短期大学美容総合学科　〒 192–0375 東京都八王子市鑓水 530
E-mail: †c010847552@st.teu.ac.jp, {hattori,kameda}@cs.teu.ac.jp, ††ckubomura@yamano.ac.jp

あらまし　今日，ソフトウェア開発においてソースコードの品質管理は，システム開発が大規模になるにつれ重要な課題となっている．その中で，ソースコード品質の最低保証のためにコードレビューは一つの有効な手段である．しかしながら，従来のコードレビュー手法では，永遠とレビューし続けてしまう危険性があり，非効率的，非生産的といった批判も受けている．そこで，本稿では，レビュアが時間を指定すると，その指定された時間内で効率的な受動的コードレビューの手順を自動的に生成しソースコードを流し見することで理解に繋がるコードレビュー支援システムを提案する．そのために，ソースコードの読解を支援するフローチャート図の自動生成手法と，今後の課題となるソースコードごとの設計に合わせたモデル図の選定によるモデル図を併用したレビューも合わせて提案する．
キーワード　コードレビュー，フローチャート図，ソフトウェア工学，モデル図．

Automatic Flowchart Generation for Passive Code Review

Yoshizane HIROSE†, Shun HATTORI†, Chiaki KUBOMURA††, and Hiroyuki KAMEDA†

† School of Computer Science, Tokyo University of Technology
Katakura-machi 1404–1, Hachioji, Tokyo 192–0982, Japan
†† The General Department of Aesthetics, Yamano College of Aesthetics
Yarimizu 530, Hachioji, Tokyo 192–0375, Japan
E-mail: †c010847552@st.teu.ac.jp, {hattori,kameda}@cs.teu.ac.jp, ††ckubomura@yamano.ac.jp

Abstract　Today, it is getting more important to manage the quality of source code in more large-scale software development. In such system development, code review is one of effective means to guarantee the minimum quality of source code. However, the traditional techniques of code review are often criticized as inefficient or unproductive, because they have the risk of endless review. Therefore, this paper proposes a novel support system for code review to generate an effective procedure of passive code review within a reviewer-given period of time. And also we propose a method to generate a flowchart of source code to help its reviewers understand it. As our future work, we describe a technique of code review with model diagrams selected based on each design of source code.
Key words　Code review, Flowchart, Software engineering, Model diagram.

1. はじめに

我々の日常生活は ICT 技術に支えられており，ICT 技術なしの社会はもはや考えられない．とりわけ，ソフトウェア技術は ICT の中枢そのものであり，ソフトウェアは今後とも高い信頼性・安全性を確保しつつも大規模化・高度化・高機能化への要求は増えることはあっても減ることはないものと考えられる．このような状況のもと，ソフトウェア工学に対する期待はますます高まっている．

ソフトウェア工学それ自体の歴史はコンピュータ発明以後であるため，未だ歴史は浅いがソフトウェア開発プロセスの明確化，ソフトウェア開発手法の提案など多くの重要な研究成果を上げている．その一つにコードレビュープロセスの意義の明確化および，当該プロセス遂行手法の提案などがある [1]．例えば，各プロセス（設計段階なのかコーディング段階なのかなど）に応じたレビュー観点の整理，レビューに使用すべきワークシートの開発など様々なものが提案され現場に適用され，今日でもなお試行錯誤され不断の研究・開発がなされている．

今日では，GDB や DDD のような単純なデバッギングツールだけではなく，Eclipse のような統合開発ツールや Review-Board のようなコードレビュープロセス管理ツールですらオープンソースソフトウェアとして公開され利用されている．

このように一見ソフトウェア開発の開発環境は充実しているように見える．しかしながら，Eclipse などはやはり開発を念頭に置いたツールであり，GDB や ReviewBoard はレビュープロセスをマネジメントするためのツールであり，コードレビューに重きを置いたものではない．

このような状況に鑑み，本研究ではソフトウェア開発プロセスにおいて極めて重要であるにも関わらずしばしば軽視されがちな「コードレビュー」に着目し，コードレビューのためのツールを提案，開発しその基本的有効性を確認した．

2. 提 案 手 法

本章では，コードレビュー手法の中で主要だと考えられる能動的コードレビュー手法について，問題点の抽出，問題分析を行い，新たに受動的コードレビューを提案する．

2.1 能動的コードレビューからの問題抽出

コードレビューを行う際，多くのレビュアは，テキストエディタやソースコードを紙媒体に印刷しコードレビューを行う傾向がある．これらは一挙にレビュアの意志により自在にコードレビューする範囲を定め，ソースコードを読み解く方法（以下，これを “能動的コードレビュー” と記す）である．

この能動的コードレビュー方法は，各レビュアの予備知識やプログラミングの経験に大きく依存するものであり，ソースコードの全体の構成を見通した上でならば高いフィードバックが期待される．しかし，そのためにはエンジニアとしての高い教養と経験が必要となり，能動的コードレビューは中級者向けの手法と言えるだろう．しかし，エンジニアとしての高い教養があることで能動的コードレビューで高いフィードバックが必ず得られるかというとそうではない．実際の開発現場では，開発プログラマーが説明の場を設けソースコードに含まれるロジックを解説するといったことがなされている．能動的コードレビューは局所的な解析に他ならず，ソフトウェアの高機能化および多機能化に伴ってあまり有効な手法とは必ずしも言えない．

以上の要因を分析するに一行ずつソースコードを読み解く従来の能動的コードレビュー手法では，全体を見通したソースコード理解を行うには難がある．このような知見のもと，本研究では，コードレビューにおいて受動的視聴 [2] によって全体を見通す手法（以下，これを “受動的コードレビュー” と記す）を提案する．

2.2 受動的コードレビュー

受動的コードレビューとは，レビュアが時間を指定するとその指定された時間内で効率的にコードレビューするための手順をレビュー支援システムが自動的に生成し，それを流し見することでソースコードの理解に繋がるレビュー手法である．

レビューシステムはレビュアが現在ソースコードのどの位置を見ているといった情報をより正確に監視することが可能となり，レビュアが求めているソースコードに対しての補足的な情報をより適時，適切に提示することができる．また，レビュアにとって限られた時間でのコードレビューとなることから効率的な新しいレビュー方法が必要となる．

本研究では，受動的コードレビューの効率的なコードレビュー手法としてモデル図による “ソースコード全体を見通すための可視化”，“ソースコードのレビュー範囲に合わせたロジックの可視化” を支援する方法を提案する．

ソースコードをモデル図に起こし，レビュアの理解を深める方法は，従来の能動的コードレビューでもソースコード解析ツールによって併用されていた手法である．しかし，従来の解析ツールによって得られる情報では，アルゴリズム，外部システムなどの複雑に交わったロジックから情報を抽出するまでには至っていなく，高い成果が得られているとは考え辛い．

そこで，受動的コードレビューでは，図 1 のようにソースコードレビューとモデル図による可視化を並行して行い，モデル図にて捨象された情報をコードレビューにより補い，かつ，レビューしている範囲に適したモデル図を提供する．これにより限られた時間内で高いレビュー成果が期待できる．

以下，受動的コードレビューを支援するためのモデル図として手始めにフローチャート図 [3] による支援システムを作成し，本提案手法の有用性を検証する．

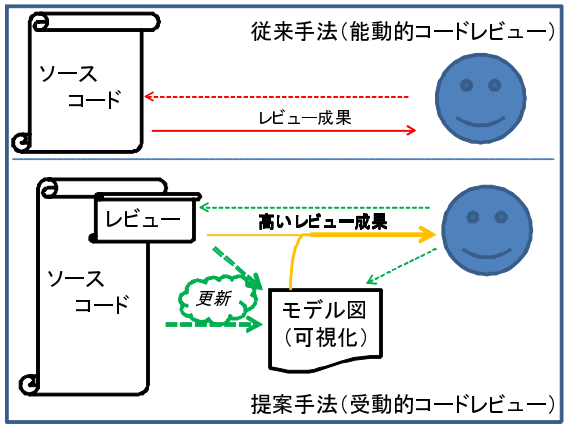

図 1 受動的コードレビュー

3. 受動的コードレビューのためのフローチャート自動生成システム

本研究では，受動的コードレビューを支援するフローチャート図の有用性の確認を目的としている．だが，フローチャート図に限定したローカルな要求に限定せず，多種類のモデル図に共通する概念の要求を定めた．

以下本章では，従来の解析ツールでのモデル図生成と開発者自身が記すモデル図との差異の検討，提案する受動的コードレビューのためのモデル図の要求定義，フローチャート図に追加的に設ける機能の選定について述べる．

3.1 モデル図生成における問題点とその考察

解析ツールから作られたモデル図での問題点を開発者自身が記すモデル図と比較した場合，開発者自身が記すモデル図では用途の状況を想定した最低限の記述が可能である．しかし，解析ツールにて生成されるモデル図は，多数のユーザを対象とするシステムであり，ユーザ間での誤認識を防ぐためにモデル図の共通性や規則を厳密に守る必要がある．このことから解析ツールからモデル図を生成する場合，抽象化と詳細情報が混合

したモデル図となる．これでは本来のモデル図として抽象的な表記法という利点が損なわれてしまう．

一方，本研究で提案する受動的コードレビューのためのモデル図は，コードレビューと併用することのみを想定している．また，モデル図にて抽象化された情報についてもソースコードと併用して見ることで誤認識するといった間違いを防げるものであり，モデル図での抽象的な表記法も可能である．

また，受動的コードレビューシステムはレビュアが見ているソースコードの範囲を監視することとなり，システム側からレビュー範囲に適したモデル図を提供し理解を促すことができる．以上を踏まえ，次のような要素を含んだモデル図を提案する．

- レビュー範囲に適したモデル図の更新
- ソースコードから補完可能なモデル図の記述法

また本研究では，受動的コードレビューにおいてソースコードとモデル図を並列して閲覧する際に有用である次の要素を含んだモデル図の提案も同時に行う．

- ソースコードとモデル図の部分的な対応付け
- レビュー範囲の切り替えなどのシステム側の動作の明示
- モデル図からのレビュー範囲の操作
- コードレビューに割り当てる時間の設定

以上のことを踏まえ本研究では次の機能をフローチャート図に実装し，検証する．

3.2 レビュー範囲に適したモデル図の更新

コードレビュー実行時，レビュー範囲に適応したモデル図を表示することでレビュアの理解に繋がるようにする．図 2 のようにソースコードすべてを表示しているフローチャート図から現在レビューしている範囲をフローチャート図内に記すことで，レビュアはフローチャート図とソースコードの対応関係を簡易的に読み取ることができる．

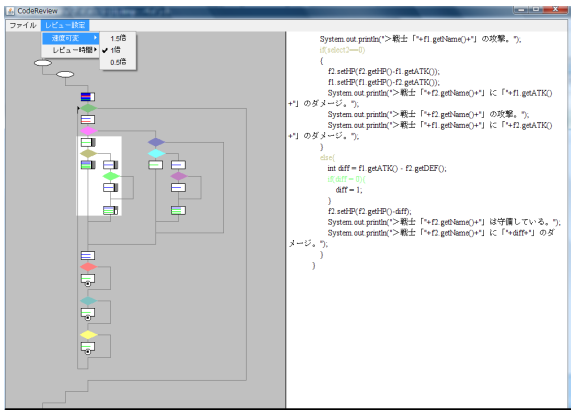

図 2　受動的コードレビュー実装画面

3.3 ソースコードから補完可能なモデル図の記述法

受動的コードレビューにおいてモデル図から読み取れる情報はより抽象的な情報であり，ソースコードからこの抽象化された情報を補完することでモデル図にてより広く情報が読み取れるようにする．フローチャート図において，図 3 のように処理記号が有するソースコード部分の内容を入力，出力，処理全般の三つのカテゴリに分類し，図 4 のようにそれぞれを赤のライン，緑のライン，青のラインとして表示することで，ソースコード全体の見通しが良くなることが期待される．尚，これらは，予め入力処理として `readLine`，出力処理として `System.out.print` や `System.out.println` といった命令コードのリストを用意することでカテゴリの分類を行っている．

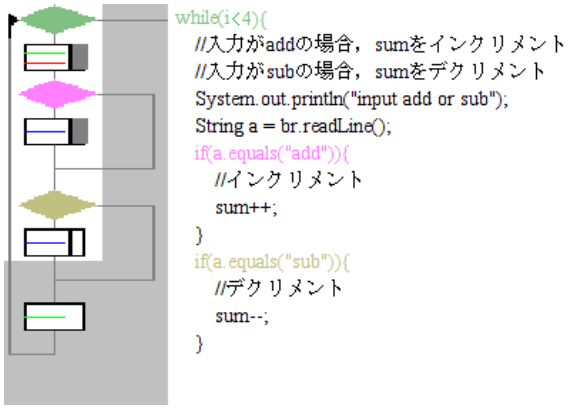

図 3　受動的コードレビューの拡大画面

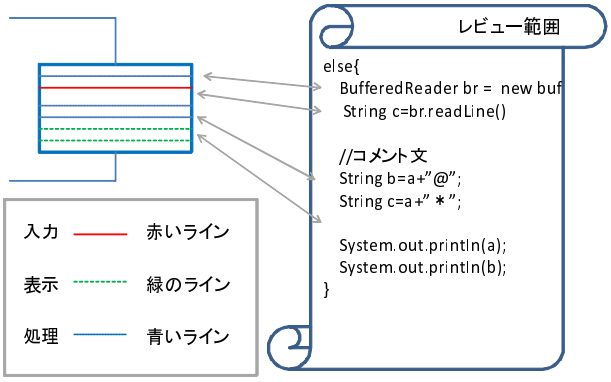

図 4　ソースコードから補完可能なモデル図の記述法

3.4 ソースコードとモデル図の部分的な対応付け

コードレビュー実行時，レビュアは対象のソースコードの一部と，フローチャート図との双方の情報を閲覧することとなり，画面の情報を部分的に強調した表示が必要となる．そこで図 3 のように `for` 文や `while` 文，`if` 文といった制御構文の命令に対して図 5 のようにフローチャート図とソースコードとで同じ表示色を割り当てて，情報の補完を助ける働きを成している．

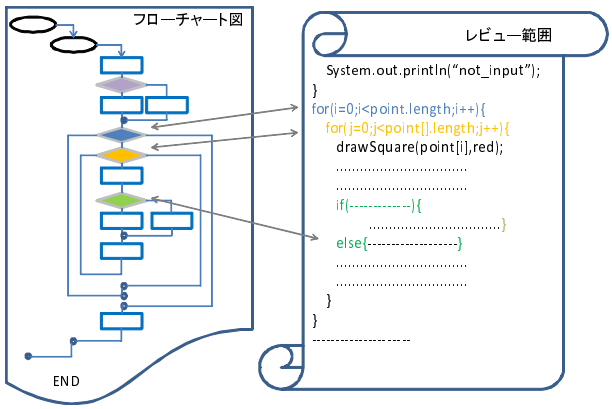

図 5　ソースコードとモデル図の部分的な対応付け

3.5　レビュー範囲の切り替えなどのシステム側の動作の明示

受動的コードレビュー実行時に，レビュー範囲の切り替えなどのシステム側の突然の動作に対して何らかのタイミングを明示する必要がある．そこで，フローチャート図内に，現在システム側がレビューしている情報をどの程度辿り，どのタイミングで切り替えるといった情報を示唆するようにこれを設ける．

図 3 が実際の動作画面である．処理記号の右部，左の黒色のラインが読み込んだソースコードを表し，灰色のラインが現在辿っているレビュー状態を表している．また，図 6 のように黒色のラインの終端に灰色のラインが辿り着くと新たにレビュー範囲を切り替え，灰色のラインは初期化され，新たにレビュー範囲をゲージとして表す．

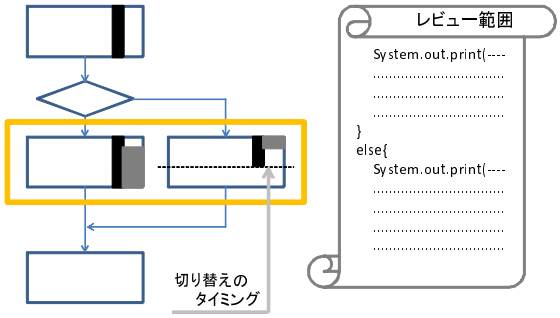

図 6　レビュー範囲の切り替えなどのシステム側の動作の明示

3.6　モデル図からのレビュー範囲の操作

コードレビューとモデル図を併用する際，モデル図にてレビュー範囲を操作することで，レビュアはより自在にソースコードのレビュー範囲を選定することができる．本システムでは図 7 のように各フローチャート記号をクリックすることで対応するソースコードをレビュー範囲に指定し表示する能動的な機能も用意した．

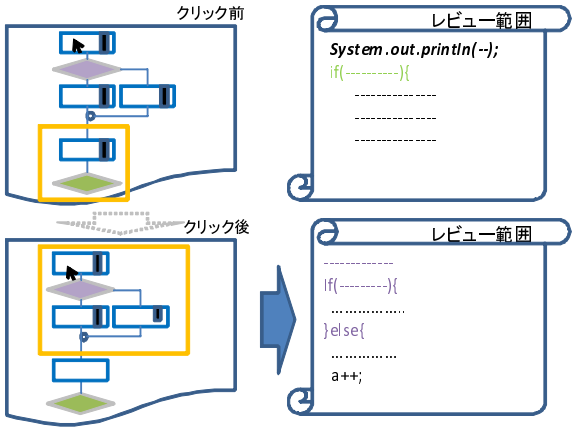

図 7　モデル図からのレビュー範囲の操作

3.7　コードレビューに割り当てる時間の設定

受動的コードレビューにおいて，レビュアはレビュー時間などを予めシステムに設定することで，タイムマネージメントなどのコードレビューが抱える問題に対して，終了時間を明確にすることが可能となる．図 8 のように，基準となるレビュー時間から相対的に設定する機能と，予め用意された時間リストから選択してレビュー時間とする機能の二つを用意した．

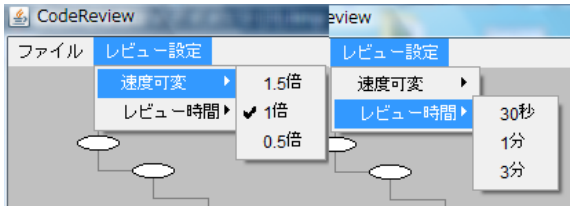

図 8　コードレビューに割り当てる時間の設定

4.　評価と考察

本章では，受動的コードレビュー支援システムを東京工科大学コンピュータサイエンス学部の学生 17 人に使用アンケートに協力して頂き，その集計結果を基に行った本システムの評価と検討，考察について述べる．

4.1　実装した機能の評価結果と検討

受動的コードレビューに設けた各機能を評価するにあたり，評価基準としてレビュアの主観的な評価を元にこれらの機能の有用性を検証する必要がある．アンケート方法は，レビュアに受動的フローチャートレビューシステムを使用してもらい，5 段評価（1 ：不要，2 ：やや不要，3 ：やや役に立った，4 ：役に立った，5 ：良く役に立った）で各機能ごとに評価してもらった．結果を図 9〜図 14 に示す．グラフの縦軸は各評価の集計数 [人] を示し，横軸に評価値を示す．また，表 1 に各機能ごとの評価値の平均値を記す．

表 1　各機能に対する評価の平均値

実装機能	平均値
レビュー範囲に適したモデル図の更新	4.00
ソースコードから補完可能なモデル図の記述法	3.41
ソースコードとモデル図の部分的な対応付け	4.35
レビュー範囲の切り替えなどのシステム側の動作の明示	3.29
モデル図からのレビュー範囲の操作	4.24
コードレビューに割り当てる時間の設定	3.12

それぞれの評価の平均値を見比べた場合，“ソースコードとモデル図の部分的な対応付け”，“モデル図からのレビュー範囲の操作” がそれぞれ高い値を示している．これらはフローチャート図とのソースコードとレビュー範囲を照らし合わせ理解するのに，有用な機能であったと考えられる．これに対して，“ソースコードから補完可能なモデル図の表記法” に関しそれ程の有用性が見られたとは考え辛い．その理由にそれぞれの画面を同時に見る場合にレビュアによる部分的な補完は負担がかかる方法であったと考えられ，このような評価になったと考えられる．これについて，今回の使用では三つのカテゴリ分類の色が何を表しているか十分に身に付いていない状態であったが，何度も使っているうちに，本システムへのリテラシーが高まり有用性は増加するものと考えられる．

モデル図からのレビュー範囲の照らし合わせに関連した物の中に “レビュー範囲に適したモデル図の更新” も含まれる．今回機能としてフローチャート図に設けるには適当ではなかったのではないかと考えられる．自由記入欄にてレビュー範囲に対応してフローチャート図にも拡大縮小が必要というようなコメントが見られた．

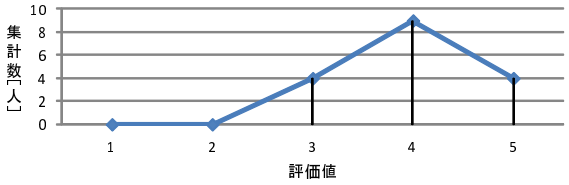

図 9　“レビュー範囲に適したモデル図の更新” の集計結果

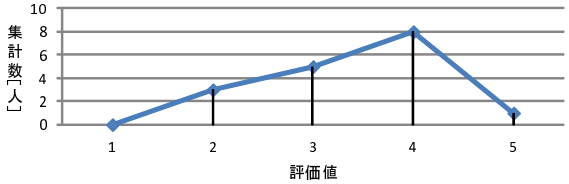

図 10　“ソースコードから補完可能なモデル図の記述法” の集計結果

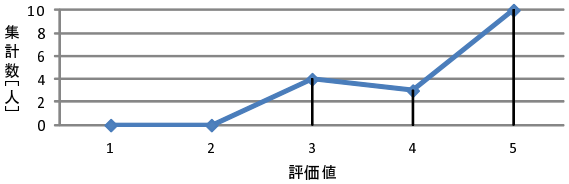

図 11　“ソースコードとモデル図の部分的な対応付け” の集計結果

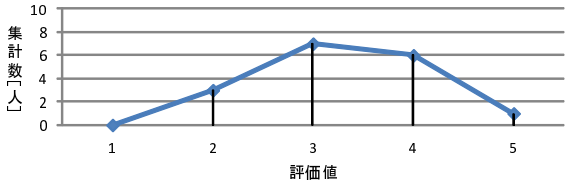

図 12　“レビュー範囲の切り替えなどのシステム側の動作の明示” の集計結果

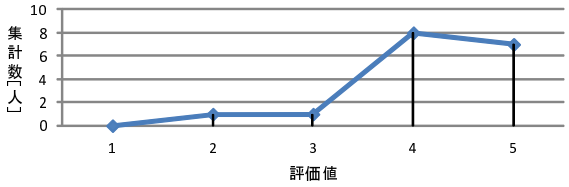

図 13　“モデル図からのレビュー範囲の操作” の集計結果

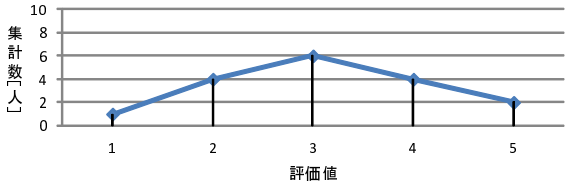

図 14　“コードレビューに割り当てる時間の設定” の集計結果

また，評価が低いものから見ていくと “コードレビューに割り当てる時間の設定” と “レビュー範囲の切り替えなどのシステム側の動作の明示” という順で平均値が低いことが分かる．これに関し，双方とも直接的にソースコードのレビュー範囲に関与するものでないという理由が考えられる．

新しく提案した 6 つの機能をフローチャート図にて実装することで新しいモデル図の有効性評価を行ったが，フローチャート図に実装する機能として向き不向きがあったことは否めない．そこで今後，フローチャート図に追加する機能をレビュアが自由に選択することができる機能も実装する必要があると考える．

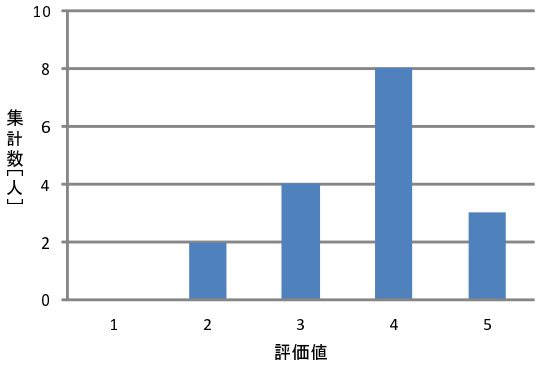

図 15　フローチャート図の有用性に関する集計結果

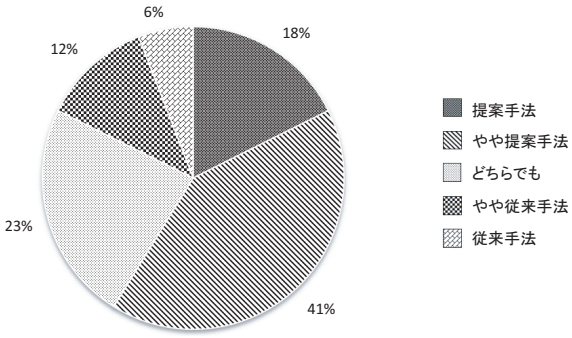

図 16　コードレビューにおけるモデル図併用の評価集計結果

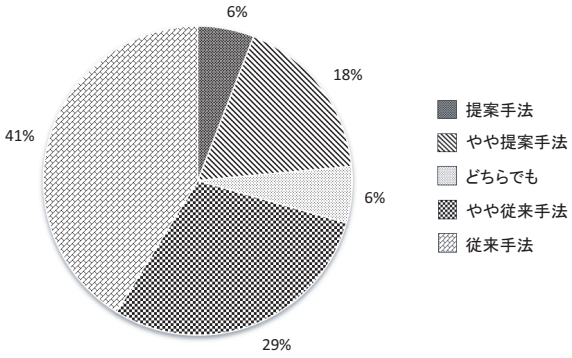

図 17　受動的コードレビューと能動的コードレビューの比較集計結果

4.2　コードレビューにおけるモデル図の併用の有効性

受動的コードレビューシステムにおけるフローチャート図の有効性に関して図 15 のような結果が得られている．この集計結果から，大多数の被験者は 3 以上の評価であり，平均値としても 3.71 であった．本研究にてフローチャート図に実装した機能の総合的な評価であったと考えられる．また，コメントを参照のもとモデル図をフローチャート図に限っても多用な用途から，“拡大縮小など” の状況に応じて複数の表示形式を有するフローチャート図の設計も十分に期待できると考えられる．

また，図 16 に示すようにモデル図の併用によるコードレビューは，高い評価を得ていることから，モデル図を用いることでのコードレビュー支援は可能であると考えられる．

4.3 受動的コードレビューと能動的コードレビューについて

図 17 が本システムを使用してもらった上での受動的コードレビューと能動的コードレビューとでコードレビュー手法を比較した集計結果である．

現状のモデル図による受動的コードレビューの有効性はあまり高いものではないが，役に立ったといった評価も含まれること，またモデル図による支援が有効であることから，今後モデル図の併用をより適切なものにすることで十分に有用性があると考えられる手法だと考えられる．

5. 今後の課題

本稿ではフローチャート図に限ったモデル図によるコードレビュー支援であった．本章では，ソースコードの情報に限らず開発現場の情報をモデル化することにも有用性があることを示し，また，より実用性のあるモデル図を併用したレビューについても提案する．

5.1 ログ記録からのソースコード編集軌跡の明示

ソースコードをレビューする際に既知のソースコードに関しては，その箇所に対するレビューの優先度は低くする必要がある．従ってソースコードの更新に際して，そのログを検知することで図 18 のようにソースコードの変更に伴った背景や部分的処理の特定が可能となり，ログ記録からシステムの変更点を明示することが可能となる．

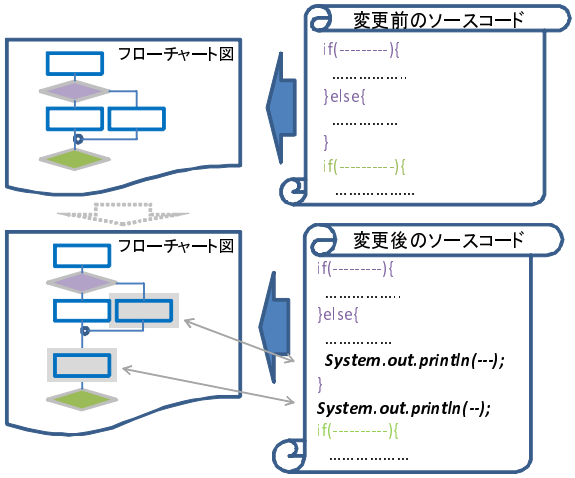

図 18 ソースコード変更に伴った差分明示モデル

5.2 モデル図を併用したレビュー

本稿では，受動的コードレビューを支援するモデル図としてフローチャート図に限定したものであった．しかし，本来プログラムの記述方法は多様であり，フローチャート図では表現が難しいものもあると考えられる．そこで，モデル図を併用したレビューはソースコードの静的解析のもと，よりソースコードの構造を表すことできるモデル図の潤沢化が必要であり，その中から受動的コードレビューにおいてより最適なレビュー手順とモデル図を選定する必要がある．本節ではクラス図とシーケンス図での受動的コードレビューにおけるモデル図の素案を提案する．

5.2.1 クラス図

プログラムの構造化においてクラス図での表現は必要であると考えられる．また，何らかのデータ `XXX` を取得する `getXXX` や，何らかのデータ `YYY` を設定する `setYYY` といったコーディング時において慣習化された関数（メソッド）の命名規則などに関してもより簡易的に明示することができ，受動的コードレビューにおいて，文字列として認識する必要性が無い（低い）箇所などを飛ばしてレビューさせることも可能であると考える．

5.2.2 シーケンス図

プログラムに含まれるデータの生成や破棄，更新などのアルゴリズムに着目しそれらのロジックやデザインパータンといった設計を表したい場合などには，シーケンス図を用いることが有用であると考えられる．受動的コードレビューにおいて，レビュアが見ているレビュー範囲と対応付けることで，ソースコード内に含まれる複数のロジックを適宜明示することでレビュアの理解を促すことが可能であると考えられる．

データの生成や破棄，更新などのアルゴリズムに着目しそれらをシーケンス図として抽象化して表すことでレビュアにプログラムに含まれる設計を明示することができると考えられる．

6. ま と め

本稿で我々は，ソフトウェア開発現場において肥大化するソースコードの新たなレビュー方法として，ソースコードから自動生成したモデル図を併用してレビューを行う “受動的コードレビュー” を提案した．限られた時間での高いコードレビュー効果を念頭に置いたレビュー手法を提案し，その基本的な考え方，要件及び基本的構成について述べた．現在，フローチャート図を用いてコードレビューを補助する基本設計は定まっていることから，今後これらの基本設計を他のモデル図にも適用することで，開発者からのヒアリングを必要としない大規模コードレビューが受動的コードレビューで可能になるかもしれない．

今後は，モデル図の併用レビューにおいて多数のモデル図を併用し受動的コードレビューの有用性を高める必要がある．当面の目標はクラス図による構造化の表現，シーケンスによるデザインパターンの抽出などを受動的コードレビューという手法の上でどのように設計するべきかを検討することがするべき課題だと考えられる．

謝　　辞

本研究は科学研究費助成事業（学術研究助成基金助成金）若手研究（B）（研究代表者：服部峻，課題番号：23700129）「ウェブから時空間依存データを抽出するウェブセンサに関する研究」の助成を受けたものである．ここに記して謝意を表す．

文　　献

[1] 上野 秀剛, 中道 上, 井垣 宏, 門田 暁人, 中村 匡秀, 松本 健一: “プログラマの視線を用いたレビュープロセスの分析,” 電子情報通信学会信学技報, Vol.105, No.12, pp.21–26 (2005).

[2] 蓬莱 博哉, 灘本 明代, 田中 克己: “トークショーメタファーによる複数 Web ページの受動的視聴,” 情報処理学会データベースシステム研究会報告, Vol.2003, No.5, pp.155–162 (2003).

[3] “情報処理用流れ図・プログラム網図・システム資源図記号,” 日本工業標準調査会, JISX0121 (1986).