Java におけるソフトウェア開発支援ツールとしてのソースコード表現方式の提案

外山　雄一　　越田　一郎
東京工科大学工学部

1. はじめに

近年ソフトウェア開発において，オブジェクト指向開発が，非常に普及している．しかし，複雑なシステムになるほど，構造が把握しづらく，開発の作業に影響を与えているように思われる．

日々の打ち合わせにおいて，開発しているソフトウェアのある処理について報告する際に，全体の構造やクラス関係に関しては理解できるが，実際の処理の中身やアルゴリズムなどは理解しづらい，あるいは，理解するのに多少時間を必要とするといった状況が頻繁に起こっている．この場合，未熟な開発者であった場合，開発した処理に開発者自身が混乱してしまうということもある．開発者自身が開発した処理をうまく整理できていないのである．

また，報告書を作成する際に，ある処理におけるソースコードを添付する場合がある．オブジェクト指向で開発している場合，クラス単位でソースコードを添付するのが一般的であるが，そのクラスのソースコードの中には，報告したい処理とは関係の無い部分まで含まれている．その分，ドキュメント量を増やしてしまう．

これらの状況を解決するためにソースコードの表記方式を提案する．

2. 表現方法の提案

現在，ソフトウェア開発の分野で広く知られているものに UML がある．UML は，開発者間だけではなく，ユーザとのコミュニケーションを図るためにも用いられているため，一般的な表記法を使用している．しかし，開発者同士では，ソースコードを直接示しながら議論を行う方が，より簡潔に済むであろう．そこで，ユーザと開発者という視点から離れ，開発者同士という視点から新たな表記方法を提案する．

ここで取り上げる表記方法とは，主にソースコードに対する提示方式である．より少ない表記方式で，的確に問題としている箇所を提示できる方法を考える．また，この表記法を SCD(Source Code Diagram)と呼ぶ.

A Diagram Representation for A Software Development Supporting with Java
Yuichi Toyama, Ichiro Koshida
School of Engineering
Tokyo University of Technology

3. 一般的な流れ

オブジェクト指向で開発を行う多くの場合には，複数のオブジェクトが持っている処理を組み合わせて，ある一つの処理を実行するような形になるであろう．実際に，その処理を説明する際には，その処理に関わるクラスのソースコードを提示し，呼ばれている処理をそのクラスソースコード中から探すといった作業を行う．その際に，それぞれのクラスのソースコードを繰り返し見比べ，その処理について把握することとなる．

しかしながら，クラスのソースコードには，問題にしている処理とは関係のない処理も含まれており，これらの記述が妨げとなって，一つの処理を理解するのに時間がかかることがある．また，ドキュメント量を増やしてしまうこともある．

4. SCD

まず問題にしている処理を明確にし，その処理名を題名にする．

4.1 ソースコードの記述

問題にしている処理のソースコードを示すためには，次のように行う．

まず，水平線で 3 つの区分に区切られた長方形で表し，最上部の区画にはクラス名，中間部の区画には問題にしている処理で必要な宣言，下部の区画には実際に使用しているメソッドのソースコードを記述する．

中間部の区画における処理に必要な宣言とは，下部の区画で記述されるメソッドにおいて使用されるフィールドの宣言や，それらのフィールドの初期化部分にあたるコンストラクタなどを記述する．問題としている処理に必要の無い宣言などについては，記述する必要は無い．

4.2 他のクラスの処理との関連

オブジェクト指向では，他のクラスの処理を呼び出すことが多々ある．この部分をどれだけ簡潔に表

現できるかどうかが重要な部分となる．SCD では，次のように表記する．

他のクラスを呼び出している部分を下線で示し，この呼ばれたクラスに対しては，すべてイタリック体で表記する．その直下に，入れ子のようにして呼ばれたクラスに対するソースコードを同様にして記述する．さらに，他のクラスの処理が呼び出されている場合には，この入れ子の状態を繰り返す．この表現方式により，一つの図でその処理における各クラス間の関係を見ることができる．さらに，呼び出された他クラスの処理の説明を長方形の内側に“ // ”(ダブルスラッシュ)を頭記号として記述する．

また，名前を取得するだけなどの処理のように単純な処理の呼び出しや，説明上必要でないと判断した場合には，入れ子を省略してもよい．

処理によっては，非常に複雑で 1 ページでは収まりきらない場合が考えられる．その場合，切れた部分を波線で表し，次のページに渡って表記されていることを示す．

4.3 問題の処理におけるクラス関係

最後に，ソースコードの表記だけでは，クラス関係がわかりづらいことを考慮し，UML で定義されているクラス図を付加することによって，この問題を解決した．

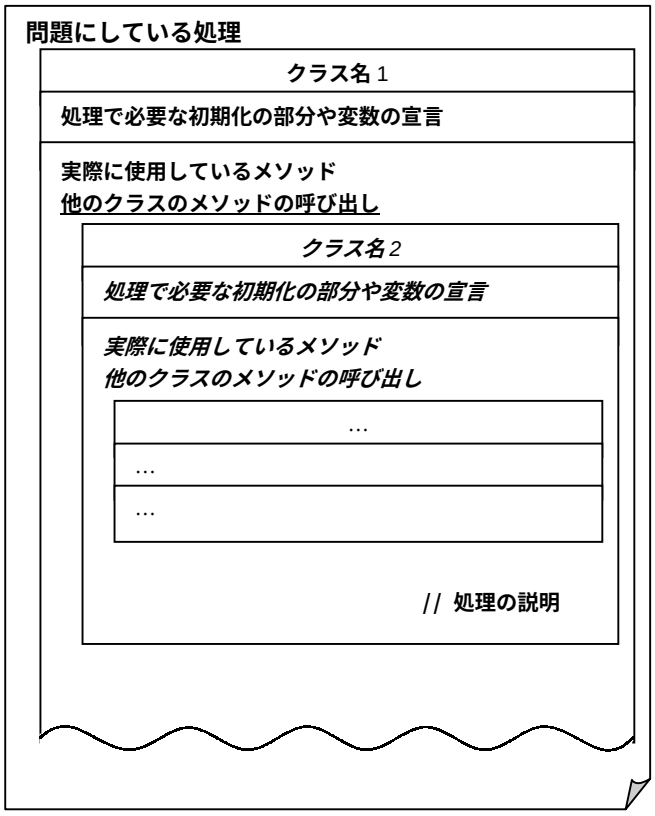

図 1 ソースコードの表記

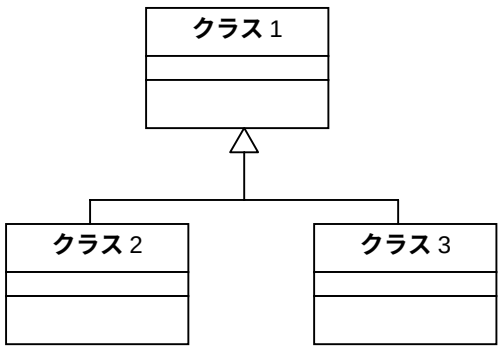

図 2 クラス関係図

5. 考察

問題点としては，まず，他のオブジェクト指向言語への対応である．現在は，Java を対象としているが，他の言語に対して，適用が可能かどうかを検討する必要がある．

次に，SCD を作成する際の手間が非常にかかるという問題である．SCD を自動的に生成するためのツールを開発することによって，この問題を解決できると考えている．

また，この表記の用途は，報告書を作成する時だけにとどまらない．開発者にとってシステムにおける個々の処理を常に細かい部分まで把握していることは，非常に重要なことである．この表記法を使用し，開発しているシステムを整理しておくことによって，開発者は常に個々の処理を細かい部分まで把握できることに加え，問題点の早期発見と改良，さらに，時代の変化に伴う急速な変化にも対応し得るものであると考える．

6. まとめ

オブジェクト指向開発における開発者間のコミュニケーションツールとしての SCD を提案した．

この SCD により次の利点が得られる．

- 必要な部分だけのソースコードを提示することにより，ドキュメント量を最小限に抑えることができる．
- 一つの図で，他のクラスの処理もまとめて参照することができる．
- 個々の処理ごとに分けられているため，クライアント側から見た処理と開発者側から見た処理との照合が取れやすい．

現在，この SCD を実際のシステムに適用し，検証を続けている．