Vol.2009-SE-165 No.6
2009/7/2

フォーマルメソッド：分散開発への応用例

宮永 照二†
独立行政法人　産業技術総合研究所

アブストラクト
フォーマルメソッド[10)11)12)13)14)15)16)17)19)]は高信頼ソフトウェア[1)2)3)4)5)6)7)]を開発するための方法論として有効であるが、多くの場合、信頼性を得るよりも高いコストを要求する。そのため、フォーマルメソッドの適用においては、適用範囲と適用時の効果を明確にする必要がある。
本稿では、アプリケーション開発に対して、フォーマルメソッドを適用することにより、効果的な適用方法について分析する。

Formal Method: Application on Distributed Development

Shoji Miyanaga
National Institute of Advanced Industrial Science and Technology

Abstract
Formal Method is effective to develop the highly dependable system, but in many cases is need higher cost than obtaining of dependability. Therefore, to apply the formal method, it is need to clarify the applied domain and effectiveness when applied.
In this paper, we analyse the executive way to apply the formal method for application development.

1.はじめに

1.1背景

ソフトウェアの高信頼性を得るための方法としてフォーマルメソッドが注目を浴びている。フォーマルメソッドは、高信頼性ソフトウェアを開発する方法として、産業界でも実績を得ているが、高い信頼性を得られる反面、高いコストを代償とする問題があり、フォーマルメソッド導入の課題となっている。フォーマルメソッド導入に際しては、システムを記述するという問題だけでなく、それを適用する人材や、適用時のコストや、メンテナンス性などを考慮して決定しなければならない。

本稿では、ソフトウェア開発に形式手法を適用するための、実用的な方法を模索するために、1つのアプリケーション開発への適用について分析した。

1.2動機

コンピュータシステムが社会基盤として定着している。それに伴い、ソフトウェアの開発組織・開発形態もさまざまな形で運営されている。最近ではアジャイル開発などの迅速な開発モデル[18)20)]が流行している。フォーマルメソッドは堅牢なシステムを提供するには有効であるが、その分多くのコストを要するため、安易な導入はアジャイル開発のメリットを損ないかねない。そのような現実の開発モデルへの適用を促進するためには、現実の開発形態にあわせたフォーマルメソッドの適用方法を考える必要がある。UML[20)]による仕様記述をそのままモデル検査する方法の研究はその例である。しかし、実際にはUMLを利用しているケースは30%程度と報告されており、その中でもUMLで要求仕様を満たす完全な記述をしているケースはさらに低くなると考えられる。詳細な仕様が文書化されていなかったり、メモ的なテキストで与えられていたりと、多様な運用の仕方をする。フォーマルメソッドの適用対象を分析し、より現実的な適用方法を模索する必要がある。

1.3構成

本稿は次の構成による。2章では、分析対象のアプリケーションを概観する。3章では、対象アプリケーションを開発し、対象アプリケーションの設計に対して設計モデル検査を適用する例を示す。4章では、本対象アプリケーションへのモデル検査適用について議論する。5章では、フォーマルメソッドの実用的な適用についての関連研究について述べる。

2.分析対象アプリケーション

2.1本分析の位置づけ

アプリケーションの規模が大きくなり、複数の要員による開発になるに従い、アプリケーションは複数に並行して開発されるようになる。そのよ

うな開発工程では、モジュールの結合時に不具合が発生することが多く、モジュール単体では安定動作するが、結合時に予期しない状態になり、致命的な不具合に至るケースもある。このようなケースは分散開発だけではなく、コンポーネントウェア等のソフトウェア部品の組合わせによりシステムを構築する場合の問題とアナロジーがある。そこで本稿では、１つのアプリケーションをいくつかのモジュールに分割しての開発にフォーマルメソッドを適用することにより、コンポーネントウェアの開発方法をエミュレートする。

実用的なフォーマルメソッド適用の方法を模索することを目的として、フォーマルメソッドの適用実験について説明する。フォーマルメソッドと一括りに言っても、その適用方法は広く、形式仕様記述からソフトウェア自動テストまで、幅広い工程に対してその方法が研究されている。

本稿では、1 つのアプリケーションを対象とし、アプリケーションの構造やモジュールの並行開発を考慮した際に適用可能な適切な方法として、モデル検査の応用例を示す。

2.2分析対象アプリケーションの概要

フォーマルメソッドをアプリケーション開発に対して適切な適用方法を模索するために、そのアプリケーションを設計から分析し、既存のツール等をフォーマルメソッドを適用しやすい点を洗い出すことにした。

分析対象アプリケーションの以下のように想定した。

表．分析対象アプリケーション仕様上の特徴

No.	性質	説明
1.	GUI アプリケーション	GUI 画面をインタフェースとするアプリケーション
2.	スタンドアローン	スタンドアローンで動作するアプリケーション
3.	３D グラフィックス	内部計算処理のグラフィックスによるプレゼンテーション
4.	計算プログラム	GUI 操作により計算実行制御が可能なアプリケーション
5.	マルチスレッド	計算はスレッド化され、メインプログラムと並列に動作する

本アプリケーションは、計算により、システムをシミュレーションするプログラムである。ユーザは GUI 画面を介して計算を実行する。ユーザが GUI から計算開始を指示すると、アプリケーションは計算プログラムを実行し、計算を行う。アプリケーションは、計算プログラムの実行結果を逐次受け、リアルタイムに計算状況を表示する。ユーザは GUI から計算中のプログラムの中断、再開、停止等を実行制御可能である。

また、アプリケーションの構造として、MVC アーキテクチャによるレイヤー分割された構造を採用し、以下の性質を考慮した。

表．分析対象アプリケーション設計上の特徴

No.	性質	説明
1.	MVC アーキテクチャ	モデル・ビュー・コントローラ・アーキテクチをの採用
2.	スレッディング	マルチスレッドによる並列プログラム
3.	イベント駆動型	GUI によりイベント駆動型アプリケーションを操作
4.	汎用ライブラリ	汎用のアプリケーションフレームワークの利用（Qt）

3.対象アプリケーションの構築

3.1通常の開発例

フォーマルメソッドの実用性を示すために、まずアプリケーションをフォーマルメソッドを使用しない方法で構築することを考える。アプリケーションの構築には、普及型のツールを使用する。実用的なフレームワークとして Qt を採用し、プログラムを開発する。Qt は極めて多くの API を提供しており、網羅的に API の仕様を把握することすら困難である。その反面、そのような限定的な API の理解をもって API を利用する場合を想定することができる。

一般的なアプリケーションの操作方法として、マウスによるメニュー操作やクリックやドラッグなどをサポートする。

MVC アーキテクチャでは、GUI はビュー、計算プログラムや計算対象はモデル、ユーザによる GUI からの指示を得てモデルを操作するのは、コントローラである。アプリケーションの設計時に、MVC アーキテクチャの構造にモジュールを適切に分割する。Qt は GUI アプリケーションのフレームワークを提供しており、MVC アーキテクチャを適用したプログラム作成を可能としている。Qt が提供するクラスを使用してアプリケーション開発に適用する際に、下図のような一般的なプログラムの開

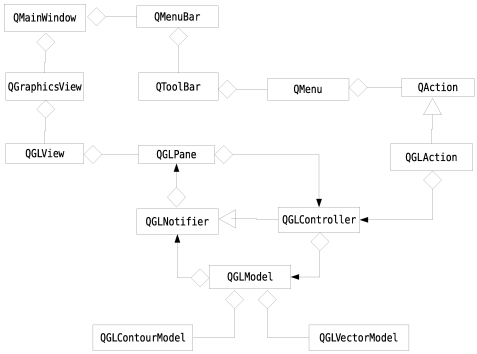
発モデルを想定している。

表．タスク一覧

No.	タスク
1.	ソフトウェアへの要求を獲得する
2.	要求を実現するための仕様を抽出する
3.	仕様を設計におとしこむ
4.	設計からコードを生成する
5.	コードを実行し仕様の正しさを確認する

3.2アプリケーションのクラスの静的構造

以下はアプリケーションのクラスの静的構造を示している。



図．アプリケーションの静的構造

この静的構造に関して簡単に説明する。Qt が提供するアプリケーションフレームワークは、通常のメニューバーをもつアプリケーションもつアプリケーションを作成するときに、QMainWindow クラスを継承する。メニューは QMenu により作成しメニューアイテムは QAction により作成する。QAction クラスは単独でもメニューアイテムとなりうるが、本アプリケーションでは QAction クラスを派生し、コントローラ部と接続可能にする。

グラフィックスの表示は QGraphicsView クラスにより行うが、OpenGL による描画をサポートするために、QGraphicsView クラスを継承した QGLView クラスを提供している。QGLView クラスは有効な描画処理をもたないため、本アプリケーションでは QGLView クラスを継承した OpenGL 描画可能な描画ペインのクラス QGLPane クラスを作成する。

モデルは QGLModel クラスにより実現する。QGLModel クラスはモデルの構造とモデルに対する操作およびモデルを表示するためのメソッドを提供する。

コントローラ部は QGLController クラスにより実現する。QGLController は QGLNotifier クラスを実装する。QGLNotifier クラスは、QGLModel クラスからの要求を受けるためのインタフェースとして作用する。QGLController クラスが、ビューからモデルの方向への操作に対して、QGLNotifier クラスを実装することにより、モデルからビューの方向への呼び出しを可能にする。

4.フォーマルメソッドの適用

4.1フォーマルメソッドの適用に関する考察

フォーマルメソッドは開発上流で適用するのが理想的とされる。しかしフォーマルメソッドを形式的に仕様を記述しようとすると、B メソッドや Z 記法等の形式仕様記述による仕様記述方法をマスターしなければならないなど、導入への障害が大きい。フォーマルメソッドのような有効であるが、コストのかかる手法の適用に関しては、以下のような要件が必要と分析されている。

1．標準との親和性
2．将来の標準候補との親和性
3．新しいことを覚えなくてもよいこと

標準との親和性の点でみると、前述の形式仕様記述使用しない方法では、UML による仕様記述をモデル検査する方法が研究されている。ただし、前述のように UML の実際の適用割合は低い。

もう 1 つのフォーマルメソッド適用時の問題として、要求から設計仕様と設計モデル検査用のモデルの両方を作成しなければならないこと、さらに設計モデル検査用のモデルは設計仕様と検査用性質を的確に反映しなければならないことがある。このコストは現実的に形式手法を適用する際の夫も大きな障害となる。モデルからプログラムのコードを生成するなどの方法が考えられる。

適用可能な方法として、標準との親和性および純標準との親和性の点では、UML による設計は妥当であり、補足事項として表などによる仕様追記も標準的に利用されているといえる。一方、設計モデルの様々な側面について完全な検証を行うのにはコストがかかる。そこで、特に重要な性質についてモデル検査をおこなうのが一般的である。

フォーマルメソッドを通常の開発フェーズに当てはめた場合を以下に示している。

No.	タスク
1.	要求は何なのかということを厳密に定義すること
2.	厳密に得られた要求を数学的・論理的に誤

	りのないように記述する
3.	誤りの無い仕様からプログラムを作成するための設計を作成する
4.	作成されたソフトウェアにたいして、ソフトウェアが仕様どおり正しく動いているかどうかプログラムを検証する

検証という作業を ISO の標準にのっとって記述すると極めて多くの作業が発生するのだが、ここでは、大枠として記載している。フォーマルメソッドを導入することにより、これらの新たな工数が発生することがわかる。フォーマルメソッドは非常にコストのかかる方法であるが、近年実用化にむけての活発な研究がなされており、ツールの提供や開発環境の提供などがなされている。本稿では、誤りの無いソフトウェアを作成するために設計に対してモデル検査を適用する。

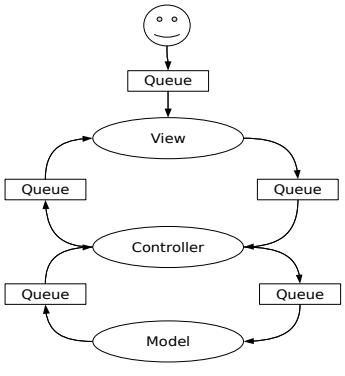
4.2アーキテクチャからみた考察

MVC アーキテクチャを採用するが、このアーキテクチャではユーザ、ビュー、コントローラ、モデルがそれぞれ、下表のチャンネルによって通信することを想定する。

表．レイヤー間のチャンネルの種類

No.	送信者	受信者
1	ユーザ	ビュー
2	ビュー	コントローラ
3	コントローラ	ビュー
4	ビュー	モデル
5	モデル	ビュー

このようなビューに基づくと、各ビュー間のメッセージの流れを以下のように表せる。



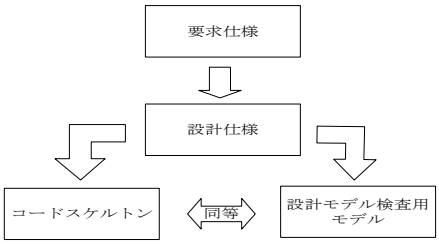
図．メッセージの流れ

このようなキューイングの構造はモジュールの分散開発を可能にする。ただし、モジュールが単体では正常に動作することが保障されていても、モジュールを結合した場合の複合状態により、予期しない不具合を発生させる可能性は残る。ここが重要な設計モデル検査で対象となる。

4.3モデル検査導入に関する考察

フォーマルメソッドのような効果はあるがコストがかかる手法を導入するための要件として、新しいことを覚えなくてもよいことがあげられるが、現実問題として、モデル検査導入の阻害要因としてはモデル検査可能なモデルを記述することである。さらにモデル検査用のモデルが正しく設計を反映していないと、モデル検査事態が無意味になることも起因している。

そこで、以下のようにコードスケルトンを生成しモデル検査用モデルとプログラムとの対応付けを与えることによりモデル検査法を適用することを考える。



図．モデル検査の概要

本稿では、設計モデル検査用モデルに対して、設計モデル検査用モデルとそれと等価なコードスケルトンを生成する。

4.4検査内容と検査方法に関する考察

プログラムのもつ性質は極めて豊富であり、フォーマルメソッドを適用する際には、どの性質を検証モデルを作成して検査するかを決めることが重要である。そこで前項で述べたように各モジュールを組み合わせた場合の複合的な状態で不具合が起きないことをモデル検査により検証することが考えられる。

一方、このような複合的な状態に対して、状態爆発と呼ばれる組合わせの数が膨大になる問題が指摘されているが、このような問題に対して、あらかじめ、処理上の制約を与えることにより、動作を検証することが考えられる。以下は、そのような制約を与える例である。

マウスイベント処理中には、メニューアイテムからの処理を起動しない

Qt を含めたリッチな GUI を提供するフレーム

ワークでは、ショートカットキーによりメニューを実行することが可能となっている。しかしこれはユーザが不用意にキーを押すことにより想定外の状態に到達してしまう可能性がある。このようなことのために、到達可能な状態に制約を与える方法を採用することは有効である。

4.5モデル設計用モデルとコードスケルトン

モデル設計用モデルとコードスケルトン間を対応づけることにより、モデル検査用モデルとプログラムとで、異なるものを表現してしまう問題の解決をはかる。

以下にその例を示す。

表．モデル例

No.	オブジェクト	説明
1	ユーザ	イベントを発行するユーザ
2	ビュー	GUI 画面
3	キュー	ユーザからビューへのキュー

コード例を以下に示す。

リスト．Promela コード例

```
mtype = { none, mousedown, mouseup,
mousemove, menustart, menustop, menupause,
menuresume };
chan queue = [N] of { mtype };

proctype userAction() {
   mtype m;
   do
      ::m=mousedown; queue!m
      ::m=mouseup; queue!m
      ::m=mousemove; queue!m
      ::m=menustart; queue!m
      ::m=menustop; queue!m
      ::m=menupause; queue!m
      ::m=menuresume; queue!m
   od
}

proctype viewAction() {
   mtype m;
   do
      ::queue?m → printf(“msg=%d\n”, m)
   od
}
```

リスト．C コード例

```
typedef enum { none=0, mousedown, mouseup,
mousemove, menustart, menustop, menupause,
menuresume } mtype;
mtype queue[N];

void userAction() {
   mtype m;
   while (1) {
      m = random_mtype();
      push_queue(queue, m);
   }
}

void viewAction() {
   mtype m;
   while (1) {
      m = pull_queue(m);
      printf(“msg=%d\n”, m);
   }
}
```

この例では、userAction()がユーザが起こしうるイベントをランダムに発生させ、キューに送る処理を担い、viewAction()がユーザからのイベントを受け取って何らかの処理をする例を示している。このコードスケルトンでは、変更するべきは 、mtype の列挙型定数と、イベントを受け取ったときの viewAction()の処理である。上記例では printf()文として例示している。

4.6複合状態の検査

組合せ状態に対する検査は、起こらない状態に到達しないことを言明する論理式を検査することにより行う。いま、マウスイベント処理中(mousedown, mouseup, mousemove)は、ショートカットキーによるコマンド(menustart, menustop, menupause, menuresume)を実行できないという制約があるとき、上記リストは以下のように変更することができる。

リスト．Promela コード例 2

```
mtype viewstate=none;
mtype ctrlstate=none;
chan queue = [N] of { mtype };
chan v2c = [N] of { mtype };
. . .
proctype viewAction() {
   mtype m;
   do
      ::queue?m → if
            ::(m==menustart && ctrlstate!
=mousedown)  →  viewstate = m; v2c!m
                  fi
   . . .
   od
}

proctype ctrlAction() {
   mtype m;
   do
      ::v2c?m → ctrlstate = m
   od
}
```

ビューの処理状態を保存する変数を追加する。そして特定の処理状態でしかメッセージを受け付けないという処理を記述することで複合的なイベント時の制約を与えるモデルを記述できる。さらに、この制約に該当する以下の論理式を検査することにより動作をモデルの正しさを検査すること

ができる。

リスト．Promela 論理式

検査用論理式：
[] ! p
マクロ：
#define p (ctrlstate==mousedown &&
viewstate==menustart)

これにより、マウスダウン処理中に開始メニューを受け付けないことを示すことができる。

5.議論

5.1フォーマルメソッドの適用方法に関する分析

本稿では、ソフトウェアのすべての開発についてフォーマルメソッドの特にモデル検査方法について実験した。導入のコストが高いと言われるフォーマルメソッドについて、適用を容易にするための分析した。その結果として以下のことが言える。

１．モジュール化された個々の処理に対してのモデル検査とモジュールを統合してのモデル検査の両方で効果的である。
２．モデル検査用モデルと C 言語のソースコードに対していくらかの記述制約をあたえることによって、性質を検証可能かつ互換性のあるモデルとコードを生成可能である。
３．モジュールの組合せに対する検査は、モジュール間の状態の組合せに対して制約を記述することで実現できること。さらにそれを論理式を検査することにより検証可能なこと。

本稿では、フォーマルメソッド適用時の障害となる、モデル作成時の問題に対して、コードスケルトンとモデル検査用モデルとの互換性と、複合状態に対する制約により分散開発への適用について述べた。

5.2議論

アプリケーション全体にフォーマルメソッドを適用する例として、MVC アーキテクチャのそれぞれのレイヤーに対して、モデル検査を適用する方法を試行した。特に、コントロール・レイヤーに対して、モデル・レイヤーとビュー・レイヤーの状態の複合状態を管理するモデルを生成し、検査した。

フォーマルメソッドの適用を阻害する工数増大を抑止するために、設計モデル検査用のモデルと等価なコードスケルトンを利用することにより、モデル検査されたモデルに対してプログラムの妥当性を保証する方法を示した。本稿で説明したモデル検査用モデルと等価なコードスケルトンを生成し、必要な性質を検証する方法は、要求仕様から設計仕様を作成することと、設計仕様から設計モデル検査用モデルを生成する方法に対して、検証モデルと実装の乖離を防ぐのに有効と考える。

5.3高信頼モジュールのカタログ化について

リッチなユーザインタフェースをもつアプリケーションは単独でも多くの内部状態をもつが、並行処理と関連づけられるときには、さらに多くの内部状態が存在し、管理を困難にする。さらに、そのようなアプリケーションの変更は新たな不具合の混入を招く。そこで、信頼性の指標として、モデル検査を適用し、安全性が確認された API についてそれをカタログ化し、正しいソフトウェアを開発することを可能にするなどの、モジュール選択を支援するなどの方法をとることにより、フォーマルメソッドの導入を支援する方法も考えられる。

6.関連研究

フォーマルメソッドの適用例に関する研究は野田らによる先行研究がある。野田ら[22]はフォーマルメソッドがどのような開発に適用されているかを分析している。

フォーマルメソッドは適用するのにコストがかかるが、同時に利用するユーザに高いスキルを要求する。そのためにフォーマルメソッドに習熟した高度 IT 技術者の育成が必要であり、トップエスイープロジェクト[10]に代表されるソフトウェア技術者への教育研究が盛んである。

アプリケーションやシステムの開発に UML 等のダイアグラムを利用する方法が利用されているが、UML で記述された設計に対してモデル検査する研究が行われている。

また、高信頼モジュールのようなソフトウェアの実行プログラム以外の管理情報を保存する方法としては、井上ら[23]によるソフトウェアタグに関する先行研究がある。ソフトウェアタグはソフトウェアの開発組織やソフトウェアの設計や変更などのプロファイルを保存し、再利用を支援する。

参考文献

1)組込みソフトウェア開発力強化推進委員会. 組込みソフトウェア開発におけるプロジェクトマネジメント導入の勧め. ISBN4-7981-0951-7. 翔泳社. 2005.
2)組込みソフトウェア開発力強化推進委員会. 組込みソフトウェア開発における品質向上の勧め [設計モデリング編]. ISBN4-86147-011-0. 翔泳社. 2006.
3)組込みソフトウェア開発力強化推進委員会. 組込みソフトウェア開発における品質向上の勧め [ユーザビリティ編]. ISBN4-7981-1190-2. 翔泳社. 2006.
4)組込みソフトウェア開発力強化推進委員会. 組込みスキル標準 ETSS 概説書［2006 年度版］. ISBN4-7981-1191-0. 翔泳社. 2006.

5)IPA/SEC. ソフトウェア開発データ白書 2006. ISBN4-8222-6200-6. 日経 BP 社. 2006.
6)IPA/SEC. プロセス改善ナビゲーションガイド〜ベストプラクティス編〜. ISBN 978-4-274-50175-3. オーム社. 2008.
7)IPA/SEC. 組込みシステムの安全性向上の勧め(機能安全編). ISBN4-274-50113-2. オーム社. 2006.
8)ソフトウェア工学研究会研究報告. IPSJ SIGSE 2009-03. 情報処理学会. 2009.
9)ソフトウェア工学研究会研究報告. IPSJ SIGSE 2008-12. 情報処理学会. 2008.
10)本位田 真一,粂野 文洋,田原 康之,鷲崎 弘宣. 「トップエスイー：サイエンスによる知的ものづくり教育」. IPSJ Magazine Vol.48 No.11 Nov.2007. 日本情報処理学会. 2007.
11)中島 震. ソフトウェア工学の道具としての形式手法 Formal Methods as Software Engineering Tools. ISSN 1346-5597. 2007.
12)中島 震. 先端ソフトウェアツール小特集. コンピュータソフトウェア Vol.24 No.2 Apr.2007. 日本コンピュータ科学会. 2007.
13)来間 啓伸. Bメソッドと支援ツール. コンピュータソフトウェア Vol.24 No.2 Apr.2007. 日本コンピュータ科学会. 2007.
14)佐原 伸, 荒木 啓二郎. オブジェクト指向形式仕様記述言語 VDM++支援ツール VDMTools. コンピュータソフトウェア Vol.24 No.2 Apr.2007. 日本コンピュータ科学会. 2007.
15)来間 啓伸, Burkhart Wolf, David Basin, 中島 震. 使用記述言語 Z と証明環境 Isabele/HOL-Z. コンピュータソフトウェア Vol.24 No.2 Apr.2007. 日本コンピュータ科学会. 2007.
16)二木 厚吉, 緒方 和博, 中村 正樹. CafeOBJ 入門(1) 形式手法と CafeOBJ. コンピュータソフトウェア Vol.25 No.2 Apr.2008. 日本コンピュータ科学会. 2008.
17)中村 正樹, 二木 厚吉, 緒方 和博. CafeOBJ 入門(2) 構文と意味. コンピュータソフトウェア Vol.25 No.2 Apr.2008. 日本コンピュータ科学会. 2008.
18)鈴木 正人. ソフトウェア工学 プロセス・開発方法論・UML. サイエンス社. 2003.
19)中島 震. SPIN モデル検査. 近代科学社. ISBN978-4-7649-0353-1. 2008.
20)玉井 哲雄. S ソフトウェア工学の基礎. 岩波書店. ISBN 4-00-005608-5. 2006.
21)H.E.エリクソン,M.ペンカー, UML ガイドブック, エスアィビー・アクセス, 2000.
22)向剣文、野田夏子. A Survey on Formal Specification in Industry. IPSJ SIG Technical Report. 情報処理学会. 2008.
23)田中昌弘、石尾隆、井上克郎. プログラム理解のための付加注釈 DocumentTag の提案. IPSJ SIG Technical Report. 情報処理学会. 2008.