社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－SE－163 (13)
2009／3／18

組込みシステム設計のための UML拡張とレビュー支援ツール

石田 直樹†, 上田 賀一†
†茨城大学

組込みソフトウェア開発では，デバイスの排他制御や多重割込みの制御といった非常に複雑な処理が必要とされる一方で，高い品質と信頼性が要求されている．それらを獲得するためには，開発の上流工程における設計レビューが重要な役割を果たす．しかし，レビューの内容がレビューアの知識や経験に依存してしまうなどといった問題点がある．本研究では，特に割込み処理に関してさらに詳細なレビューを行えるようにUMLの記法を拡張した．また，それを用いて新たなレビューの対象となりうる項目を生成するように，本研究室にて開発されてきた組込みシステム向けの設計レビュー支援ツールを拡張した．そして，適用例を用いて評価を行うことで提案した記法と開発したツールの有用性を確認した．

An Extension of UML and A Review Support Tool for Embedded Software Design

Naoki ISHIDA†, Yoshikazu UEDA†
†Ibaraki University

The embedded software development requires a high quality and a reliability, because the embedded software has very complex processing such as exclusive control of devices and multiple interrupts. The design review in upper process of development is very important to acquire these properties. There is a problem that the content of the review depends on the knowledge and experience of reviewer. In this study, we extended notations of UML to enable a more detailed review about the interrupt processing. Moreover, we extended the design review support tool, that has been developed in our laboratory, to generate new review items by using extended notations. And, we confirmed the usability of proposed notations and developed tool through the example.

1 はじめに

今日の組込みシステム開発では，携帯電話のネットワーク接続や音声・画像処理などに見られるように，非常に複雑で高性能な機能が求められている．それにもかかわらず，実際の開発現場においては，開発期間の短縮や予算の縮減によってシビアな状況下での開発を余儀なくされており，品質の低下や納期の遅れといった重大な問題を引き起こす結果となっている [3].

組込みソフトウェア開発にはソフトウェアとハードウェアの両方の知識が求められるだけではなく，リアルタイム処理，割込みの制御といった複雑な処理も要求されるため開発が困難である．その一方で，組込みソフトウェアには高い品質と信頼性が求められており，開発の困難さは増すばかりである．高い品質と信頼性を獲得するためには，開発の実装工程よりも設計工程の方が極めて重要である．また，設計工程において高い品質を確保するためには設計レビューを行うことが必要である．

こういった背景から本研究室にて開発されてきた組込みシステム向けの設計レビュー支援ツール [1, 2] がある．このツールは標準のUMLを組込みソフトウェア向けに拡張した記法を用いて記述されたUMLモデルを入力とし，レビューの対象となる点を検出してチェックリストを生成する．本研究では，このツールを用いて割込み処理に関してさらに詳細なレビューが行えるように，UMLの記法及びツールの拡張を行う．記法の拡張にはUMLに拡張メカニズムとして標準で用意されている制約を用いた．

本研究で提案する記法を用いることで，設計段階から割込み処理の仕様を明確にすることができ，開

発者はそれを一意に解釈できるようになる．また，本ツールを用いることで，レビューアの経験や知識，能力に依存することなく一定レベルのレビューが可能となり，近年の組込みソフトウェア開発が抱える問題を解決することができる．

2 関連研究

本章では，本研究のベースとなる研究 [2] の概要と関連研究について説明する．

2.1 組込みソフトウェア設計のためのレビュー支援ツールの開発

この研究では，開発の上流工程である設計段階を対象として標準の UML を組込みソフトウェア向けに拡張した記法の提案を行った．また，その拡張した記法の利点を活かして組込みソフトウェア設計のレビュー支援ツールを開発した．組込みソフトウェア向けに UML を拡張する際には，UML に拡張メカニズムとして標準に用意されているステレオタイプと制約を用いており，ツールではそれら定義された記法がどのように記述されているかを調べることでレビューの対象となる問題点を検出している．

2.2 UML 設計に対するモデル検査のための検証パターンの提案と評価

UML のモデル検査における検査パターンの研究として，UML 設計に対するモデル検査のための検証パターンの提案と評価 [5] がある．この研究では，変換先の仕様記述言語として PROMELA を用い，SPIN を使用してモデル検査を行うことを目的としており，そのためのモデルから仕様記述言語への変換手法とそのパターンを提案している．

この研究は本研究と目的は同じであるが，本研究では UML モデル情報である XMI ファイルを用いてレビューの対象となりうる項目を検出するため，別方面からのアプローチであると言える．

2.3 Constraint-Based Software Specifications and Verification Using UML

SpecTRM と呼ばれる制約ベースの仕様を UML で表現し，それによって制約ベースのソフトウェアの仕様を決定し，さらに検証を行うという研究に，Constraint-Based Software Specifications and Verification Using UML[6] がある．この研究では決められた制約のセットを UML のクラス図やシーケンス図，OCL を用いて表現し，それらを組合わせてソフトウェアの設計とその検証を行うことを目的としている．

この研究における検証にはフォールトツリーを用いて行う手法があり，本ツールで行うよりもその手法で行ったほうが効果的であるため，本研究との相性はあまり良くないと言える．この研究も目的は本研究と同じではあるが別方面からのアプローチであると言える．

3 UML の組込みシステム向け拡張

3.1 拡張に使用する UML のバージョン

今回使用する UML のバージョンは 2.1 である．UML2.0 以降では組込み向けの拡張がなされており，例えば時間制約は UML のタイミング図で表現することも可能である．しかし，タイミング図を使用するといくつかの問題が発生するため [1]，今回はタイミング図は用いない．タイミング図への対応は今後の課題とする．

3.2 組込みシステム向け拡張

一般的なソフトウェア開発と異なり，組込みソフトウェア開発では以下のような処理が重要視される．

- 処理の 7 割を占めるとも言われるエラー処理
- 時間制約といったリアルタイム処理
- リソースの排他制御
- 割込み・多重割込みの制御

また，デバイスドライバには読み込み/書き出し処理といった最低限必要なインタフェースも決まっている．

以上のような組込みソフトウェア開発の特徴を考慮し，UML を用いた開発を行う上でステレオタイプと制約によって特別な意味を持たせ，明確に定義すべきであると考えたものは以下の通りである．各処理に対して定義した記法をまとめたものを表 1 に示す．

- エラー処理
- 割込み処理
- 時間制約
- 同期/非同期
- ドライバインタフェース

本研究においてさらに記法を拡張した割込み処理に関して説明する．割込み処理はステレオタイプで定義した割込み処理 (≪InterruptHandling:Interrupt≫) と制約で定義した割込み制約 ({InterruptHandling:type=*type*, level=*level*, time=*time*[*unit*]}) から成る．割込み処理は，クラス図のメソッドなどに付加することで割込み処理を行うメソッドであることを明確に表すことができ，割込み制約は，割込

表1　各処理に対して定義したステレオタイプ/制約

分類	処理	定義したステレオタイプ/制約	主な使用先
エラー処理	例外処理	≪ ErrorHandling:Exception ≫	クラス図，シーケンス図，ステートマシン図，アクティビティ図
	タイムアウト処理	≪ ErrorHandling:Timeout ≫	
割込み処理	割込み処理	≪ InterruptHandling:Interrupt ≫	
	割込み制約	{ InterruptHandling:type=*type*, level=*level*, time=*time*[*unit*] }	シーケンス図，ステートマシン図，アクティビティ図
排他制御	排他制御の開始	{ ExclusiveControlAcquire:*object* }	
	排他制御の解除	{ ExclusiveControlRelease:*object* }	
時間制約	時間制約	{ TimeConstraint:time=*time*[*unit*] }	
同期/非同期	同期処理	≪ SyncAsync:Synchronous ≫	クラス図，シーケンス図，ステートマシン図，アクティビティ図
	非同期処理	≪ SyncAsync:Asynchronous ≫	
ドライバインタフェース	初期化処理	≪ Interface:Initialize ≫	
	読み込み処理	≪ Interface:Read ≫	
	書き出し処理	≪ Interface:Write ≫	
	停止処理	≪ Interface:Stop ≫	

みの詳細情報（割込みの種類：*type*，割込みのレベル：*level*，割込み処理の処理時間：*time*[*unit*]）といった付加的な情報を表現できる．例えば割込みレベル 5，処理時間 500msec のタイマ割込みは，{InterruptHandling:type=Timer，level=5，time=500[msec]} と記述する．

4　組込みシステム向け設計レビュー支援ツール

本章では，本研究で拡張した部分も含めツールの説明をする．前章で示したステレオタイプと制約を用いて開発された，組込みソフトウェア向けの設計レビュー支援ツールがある [1, 2]．そのツールは入力された UML モデル情報（XMI）を解析することで重大な問題点とレビューの対象となる問題点を検出する．ツールは必ずしも設計ミスやバグを出力するわけではなく，あくまでもレビュー支援のための情報の出力である．また，対象となる範囲は静的なレビューのみである．

4.1　レビュー項目

本ツールで検出するレビュー項目は，重大な問題点の検出とレビューの対象となる問題点の検出から成る．

4.1.1　重大な問題点の検出

UML にはシステムの色々な側面を記述できるように様々なダイアグラムが用意されているが，各ダイアグラムは互いに完全に独立しているわけではない [2]．そのため，このレビュー項目では主にダイアグラム間の整合性，つまり，必ず修正しなければならないような重大な設計ミスがないかを検出する．このレビュー項目はツールに正しいモデル情報が入力されたかを調べるものであるため，3.2 節で示した記法は使用せず，モデル間の一般的な整合性検証を行う．以下に本ツールに実装されているレビュー項目を示す．

表2　ダイアグラム別レビュー項目数

ダイアグラム	レビュー項目数
クラス図	17
シーケンス図	16
ステートマシン図	13
アクティビティ図	2

- クラス図とシーケンス図の整合性
- クラス図とアクティビティ図の整合性
- シーケンス図とステートマシン図の整合性

4.1.2　レビューの対象となる問題点の検出

重大な問題点は必ず修正しなければならない問題点であったのに対し，レビューの対象となる問題点はレビューの支援を目的としているため必ずしも修正する必要はない．なぜなら，例えば実行速度を重視するのか，再利用性を重視するのかによって設計方針が異なり，速度を重視して意図的にそのような設計にした，ということも考えられるため，本ツールが出力する結果が必ずしも間違いであるとは言い切れない．

しかし，実装工程になってから致命的な設計ミス

表3　レビューに使用するステレオタイプ/制約

レビュー項目	使用するステレオタイプ/制約
割込み・多重割込みの発生	≪ InterruptHandling:Interrupt ≫ { InterruptHandling:type=*type*, level=*level*, time=*time*[*unit*] } { ExclusiveControlAcquire:*object* } { ExclusiveControlRelease:*object* }
時間制約・タイムアウト処理	≪ ErrorHandling:Timeout ≫ { TimeConstraint:time=*time*[msec] } { ExclusiveControlAcquire:*object* } { ExclusiveControlRelease:*object* }
エラー処理	≪ ErrorHandling:Exception ≫ ≪ ErrorHandling:Timeout ≫
ドライバインタフェース	≪ Interface:Initialize ≫ ≪ Interface:Read ≫ ≪ Interface:Write ≫ ≪ Interface:Stop ≫ { ExclusiveControlAcquire:*object* } { ExclusiveControlRelease:*object* }

が発見されないためにも，レビューの場においてなぜそのような設計にしたのかといった議論を行うことでメンバ間で知識を共有することが大切である．そのため，必ずしも間違いとは言えないが，レビュー項目として議論すべき項目であるとも言える．

前章で示したステレオタイプと制約を用いて設計すると，次のようなレビューが可能となる．以下に本ツールに実装されているレビュー項目の一部を示す．また，本ツールに実装されているダイアグラム別のレビュー項目数を表 2 に示し，各レビュー項目が問題点を検出するために使用するステレオタイプ/制約を表 3 に示す．

- ダイアグラムの粒度
- 割込み・多重割込みの発生
- 時間制約・タイムアウト処理
- エラー処理
- 排他制御
- 初期化処理
- 読み込み処理/書き出し処理
- 停止処理
- キャンセル処理

本研究で拡張した割込み/多重割込みの発生に関するレビューについて説明する．本研究では割込み処理に関してさらに深いレビューを行えるように，前章で定義した割込み制約を用いて以下の項目について調べ，チェックリストを生成する．

- 多重割込みが発生しうるか，発生しうるなら

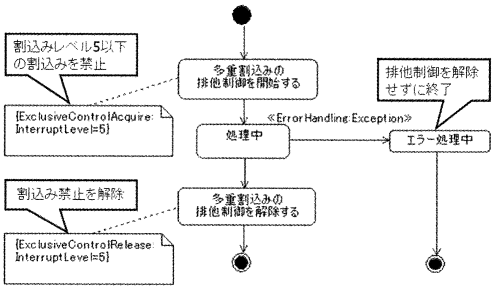

図1　排他制御の例

通常処理および割込みレベルが低い方の処理は最大でどの程度遅延する可能性があるか．

4.1.3　問題点の検出方法

問題点を検出する例として，レビュー項目「排他制御を解除せずに終了していないか」について説明する．図 1 は，多重割込みの排他制御（割込みレベル 5 以下の割込みを禁止）のステートマシン図である．通常の場合は正常に排他制御を開始/解除しているが，排他制御を開始後に例外が発生すると排他制御を解除せずに終了してしまっていることがわかる．組込みソフトウェアで重要となる処理をステレオタイプと制約で定義したため，ツール側で意味解析を行うことでどのような処理を行っているかを判断でき，問題点を検出することができる．また，ステレオタイプ/制約が複数指定されていたとしても，各レビュー項目は独立しているため問題はない．

4.2　レビュー項目の深刻度

レビュー自体の効率化を図るために，前節で示した各レビュー項目に対して以下に示す 3 段階から成る深刻度が設定されている．

- Advice(アドバイス)
  間違いではないがこうした方がモデルが良くなるというレビュー項目に対して付加される．
- Warning(警告)
  必ずしも間違いとは言えないがレビューの対象となるレビュー項目に対して付加される．
- FatalError(重大なエラー)
  主に 4.1.1 節の重大な問題点に対して付加される．

各レビュー項目に対して深刻度を定義することによって，実際のレビューを効率的に進めることができるだけではなく，現在の設計にはどのような問題が発生しているかも一目でわかるようになるといった利点もある．

4.3 本ツールの流れ

本ツールは以下の手順で行われる．

(1) モデル情報である XMI ファイルや対応関係を記述した CSV ファイルを入力として読み込む．
(2) そこから重大な問題点やレビューの対象となる問題点を検出する．
(3) その検出した問題点を出力する．出力形式は汎用性を考慮して XML 形式とした．その出力した XML ファイルと XSLT ファイルを用いて HTML 形式のチェックリストに変換・表示する．

4.3.1 入力データ

本ツールは他の UML モデリングツールから出力した XMI ファイルを入力として読み込む．使用する XMI のバージョンは 2.1，XMI で表現される UML モデルのバージョンは 2.1 である．通常，それらのモデリングツールを用いて記述すると，ダイアグラム間の各要素は ID 情報として関連付けられるため，ダイアグラム間の関連付けにはその ID 情報を用いる．

4.3.2 出力データ

本ツールは，4.1.1 節で述べた重大な問題点や，4.1.2 節で述べたレビューの対象となる問題点を検出し，HTML 形式のチェックリストにまとめて出力する．このチェックリストはダイアグラムの画像にハイパーリンクされており，クリックすることによって各要素別のチェック内容を簡単に確認できる．また，そのチェックリストをツール上で編集することができ，それを PDF 形式に変換して配布することも可能である．

5 適用例

Sparx Systems 社の UML モデリングツール Enterprise Architect を用いて作成した各ダイアグラムを XMI 形式で出力し，ツールの動作検証と出力結果の考察を行った．作成したダイアグラムは，クラス図，シーケンス図，アクティビティ図，ステートマシン図である．

5.1 入力モデル

エレベータシステムを適用例とし，以下のような条件を想定した．

- 最上階にエレベータが停止中
- 1F の通行人がフロア側のボタンを押下
- その直後に 2F のフロア側にいる通行人もボタンを押下（割込みの発生）

また，地震発生時のエレベータシステムの動作（以下，地震時管制運転）を以下のように想定した．

- センサーが P 波を感知
- 地震時管制運転に切り替える（割込みの発生）
- 最寄りの階に停止し，一定時間（60 秒間）扉を開放
- 扉を閉め，それまでに一定以上の揺れを感知していれば運転を休止，感知していなければ通常運転に復帰

作成したクラス図を図 2 に，通常運転時と地震時管制運転時のシーケンス図をそれぞれ図 3 と図 4 に示す．この他にもステートマシン図等のダイアグラムも作成したが，省略する．

5.2 実行結果

作成したダイアグラムの XMI ファイルを本ツールに入力し，実行した．その出力されたレビュー項目の一部抜粋を図 5 に示す．

5.3 考察

図 5 の実行結果について考察する．

- 1 行目
  シーケンス図で使用されているメッセージがクラス図で定義されていないことがわかる．シーケンス図とクラス図間の整合性が取れていないことがわかる．
- 2・3 行目
  クラス図で定義されているメソッドがシーケンス図において使用されていないことがわかる．使用すべきところがあるかどうか，あるいはこのメソッドが必要かどうかのレビューの対象になるといえる．
- 4 行目
  クラス図で定義されたメソッドの可視性に関するアドバイスである．
- 5 行目
  シーケンス図において初期化処理が呼び出されていないことがわかる．初期化処理を適当な箇所で行うべきである．

表 4 に示すように今回の適用例で実際に検出できたレビュー項目の総数は 77 個であり，意図的に設計して検出されたほぼ全てが納得のいく出力であった．この表からこの設計では粒度が大きく無駄が多いことが分かり，まだ実装には移行できそうにない

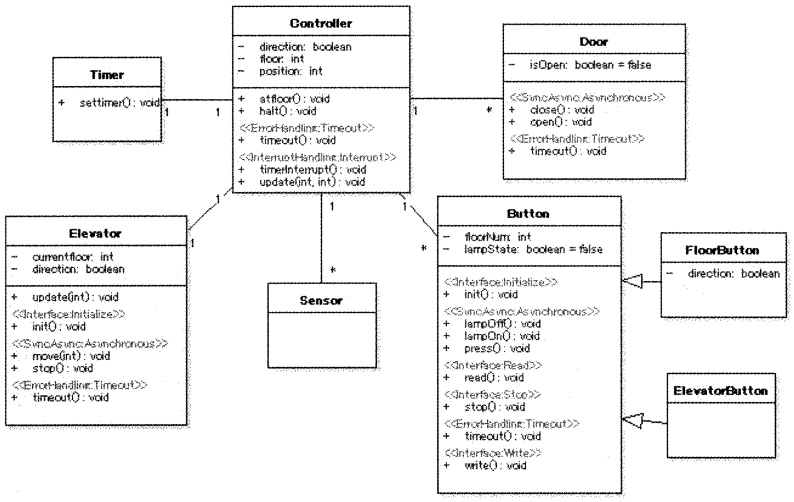

図 2　エレベータシステムのクラス図 (一部抜粋)

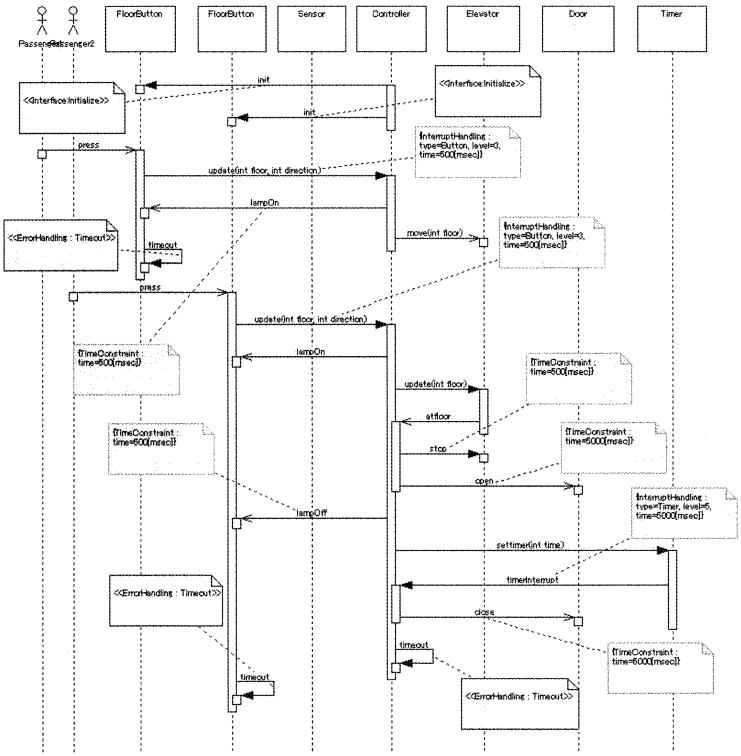

図 3　エレベータシステムのシーケンス図 (通常運転)

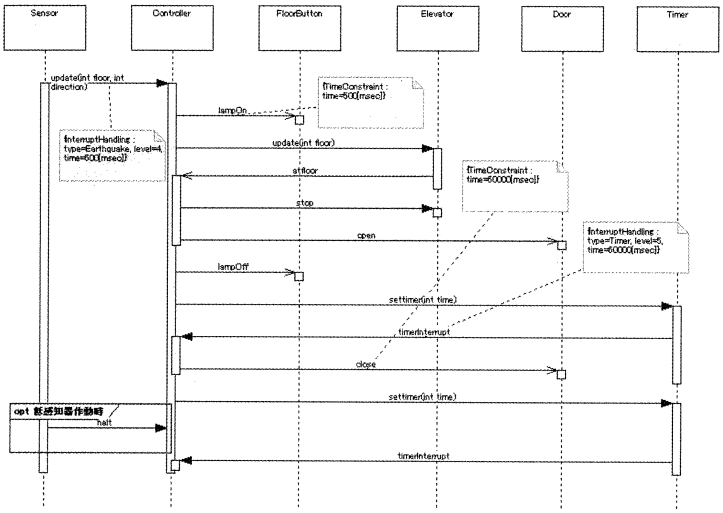

図 4　エレベータシステムのシーケンス図 (地震時管制運転)

```
1  シーケンス図のオブジェクト Elevator のメッセージ update がクラス図で定義されていません。
2  クラス図のクラス Door のメソッド timeout がシーケンス図で使用されていません。
3  クラス図のクラス Button のメソッド write がシーケンス図で使用されていません。
4  クラス図のクラス Button のメソッド lampOff の可視性 public は適切ではありません。
   適切な可視性は private です。
5  シーケンス図のオブジェクト Elevator で初期化処理 init が呼び出されていません。
```

図 5　実行結果 (一部抜粋)

表 4　検出したレビュー項目の内訳

レビュー項目	検出数
ダイアグラムの整合性	6
ダイアグラムの粒度・無駄の検出	32
割込み・多重割込みの発生	9
時間制約・タイムアウト処理	22
エラー処理	1
ドライバインタフェース	7
合計	77

ことが読み取れる．このレビュー項目を基に修正することで設計を改善することができる．ただし，レビュー項目の中には複数のレビュー項目が関連しているものもあるため，モデルの 1 箇所を修正するだけで数箇所が改善される可能性もある．

また，通常の UML 設計と本研究で定義した記法を使用した設計を比較すると，本記法の場合は割込み処理や時間制約等の情報が付加されていることにより，各ダイアグラムの中でも特にシーケンス図が非常に詳細に・明確に記述できていることがわかる．しかし，逆に本記法を使用しないと本ツールはレビューの対象となる問題点をほとんど検出できない．本ツールは提案した記法を用いて記述してもらうことが前提となっている．

表 5　割込み処理の入力データ

タイプ	レベル	処理時間 [単位]
Timer	5	60000[msec]
Timer	5	5000[msec]
Earthquake	4	400[msec]
Button	3	300[msec]

5.4　拡張部分に関する実行結果と考察

本節では，本研究で拡張を行った多重割込みに関する部分の実行結果に注目して考察を行う．適用例における割込み処理の内容を表 5 に示し，図 6 に多重割込みに関する部分の実行結果を示す．

図 6 の実行結果について考察する．表 5 で示したデータについてレベルの異なる 2 種類の割込み処理を抽出し，多重割込みが発生する可能性をレビュー

```
1  割込みレベル= 5、タイプ= Timer の処理と割込みレベル= 4、タイプ= Earthquake の処理で多重割込みが
   発生する可能性があります。発生した場合、通常処理は最大 60400[msec] 遅延する可能性があります。また、
   割込みレベル= 4、タイプ= Earthquake の処理が最大 60000[msec] 遅延する可能性があります。
2  割込みレベル= 5、タイプ= Timer の処理と割込みレベル= 3、タイプ= Button の処理で多重割込みが
   発生する可能性があります。発生した場合、通常処理は最大 60300[msec] 遅延する可能性があります。また、
   割込みレベル= 3、タイプ= Button の処理が最大 60000[msec] 遅延する可能性があります。
3  割込みレベル= 5、タイプ= Timer の処理と割込みレベル= 4、タイプ= Earthquake の処理で多重割込みが
   発生する可能性があります。発生した場合、通常処理は最大 5400[msec] 遅延する可能性があります。また、
   割込みレベル= 4、タイプ= Earthquake の処理が最大 5000[msec] 遅延する可能性があります。
4  割込みレベル= 5、タイプ= Timer の処理と割込みレベル= 3、タイプ= Button の処理で多重割込みが
   発生する可能性があります。発生した場合、通常処理は最大 5300[msec] 遅延する可能性があります。また、
   割込みレベル= 3、タイプ= Button の処理が最大 5000[msec] 遅延する可能性があります。
5  割込みレベル= 4、タイプ= Earthquake の処理と割込みレベル= 3、タイプ= Button の処理で多重割込みが
   発生する可能性があります。発生した場合、通常処理は最大 700[msec] 遅延する可能性があります。また、
   割込みレベル= 3、タイプ= Button の処理が最大 400[msec] 遅延する可能性があります。
```

図 6　多重割込みに関する実行結果

項目として挙げている．割込みによる遅延を，通常処理およびレベルの低い方の割込みについて，最大でどのくらいの遅延が発生するかを計算している．これらの項目を基に議論・修正を行うことでより品質の高い組込みシステムを開発できると考えられる．しかし，これらの多重割込みは必ずしも発生するわけではなく，逆に絶対に発生しえない多重割込みを挙げている可能性もある．この点に関してはさらに検討が必要である．

6　おわりに

本研究では，組込みシステムの中でも特に割込み処理に関して詳細なレビューを行えるような，UML の拡張記法を提案した．この記法を用いることで明確な設計が可能になり，さらに本ツールを用いてレビューを行う際に，より意義のあるレビューを行うことが可能になる．ツールを用いてレビューを行うことでレビューアの知識や経験に依存せず，一定レベルのレビューを行うことが可能となる．そして，検出された問題点を基にレビューを行うことで，実装前に問題点をできるだけ減らすことができる．それによって，手戻りによるコストの増大を防ぐことができ，さらに，最終成果物であるシステムの品質をより高いものすることができる．

今後の課題として，ステートマシン図との対応付けが挙げられる．本研究におけるツールの拡張ではシーケンス図中のすべての割込みを用いて多重割込み発生の可能性を調べているが，実際には起こりえない多重割込みを出力している可能性がある．なぜなら，組込みシステムには状態が複数ある場合がほとんどであり，異なる状態における割込み処理は同時には起こり得ないからである．ステートマシン図との対応を取り，割込み処理を状態ごとに分類した表等を生成し，それと多重割込みの発生可能性を併せて見ることで，実際に起こりえるかどうかの判断を簡単に行えるようになり，より効果的なレビューが行えると考えられる．

次に，比較実験の必要性が挙げられる．今回示した実験だけでは本記法およびツールの有効性を十分に示せたとは言い切れない．本記法の有効性を評価するためには，本記法を使用した場合と使用しなかった場合の比較実験を行うことによって，検出できたレビュー項目やレビューにかかった時間にどの程度違いが出るのか検証する必要がある．

参考文献

[1] 鈴木健司，上田賀一：UML による組込みソフトウェア設計のレビュー支援ツールの開発，組込みソフトウェアシンポジウム（2007）．

[2] 鈴木健司：組込みソフトウェア設計のためのレビュー支援ツールの開発，茨城大学大学院理工学研究科情報工学専攻修士学位論文（2008）

[3] T.M.McGinnity，L.P.Maguire：A CASE-tool oriented approach for embedded sysytems design，Microprocessors and Microsystems（2001）．

[4] 横山孝典：組込み制御ソフトウェアのアスペクト指向に基づく開発法，情報処理学会論文誌，Vol.47，No.4，pp.1185-1194（2006）．

[5] 金井勇人，岸知二：UML 設計に対するモデル検査のための検証パターンの提案と評価，情報処理学会ソフトウェアエンジニアリングシンポジウム，pp.185-192（2006）

[6] Chin-Feng FAN，Chun-Yin CHENG：Constraint-Based Software Specifications and Verification Using UML，IEICE TRANSACTIONS(D)，Vol.E89-D，No.6，pp.1914-1922（2006）