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

2007－ＳＬＤＭ－129／2007－ＥＭＢ－4 (12)
2007／3／16

組込みシステムの外部環境分析のためのUMLプロファイル

瀬戸 敏喜† 金川 太俊† 鵜林 尚靖† 鷲見 毅‡ 平山 雅之‡

†九州工業大学大学院 情報工学研究科 〒820-8502 飯塚市川津 680-4
‡ (株) 東芝 ソフトウェア技術センター 〒212-8582 川崎市幸区小向東芝町 1
E-mail: †{seto, kanagawa}@minnie.ai.kyutech.ac.jp, ubayashi@ai.kyutech.ac.jp
‡{takeshi.sumi, masayuki.hirayama}@toshiba.co.jp

あらまし 本稿は，外部環境を考慮した信頼性の高い組込みシステムを開発するための UML プロファイルを提案する．組込みシステムはアクチュエータを用いて外部環境に影響を与えるだけでなく，センサを通してそれらからの影響も受ける．この UML プロファイルでは，システムや環境と，それらの関係をあらわすステレオタイプを定義している．さらに，これらのモデル要素に対して，OCL 記述により制約を与えている．また，我々はこの UML プロファイルに対応した UML モデルエディタを開発している．このエディタを用いることで，OCL 記述による制約を破るモデル要素が検出できる．この UML プロファイルによって，システムとその外部環境を厳密に記述することが可能となる．

キーワード UML プロファイル，組込みシステム，外部環境

A UML Profile for analyzing the external environments of embedded systems

Toshiki Seto† Hirotoshi Kanagawa† Naoyasu Ubayashi†
Takeshi Sumi‡ and Masayuki Hirayama‡

† Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology
680-4, Kawazu, Iizuka-shi, 820-8502, Japan
‡ Software Engineering Center, TOSHIBA Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki-shi, 212-8582, Japan
E-mail: †{seto, kanagawa}@minnie.ai.kyutech.ac.jp, ubayashi@ai.kyutech.ac.jp
‡{takeshi.sumi, masayuki.hirayama}@toshiba.co.jp

Abstract This paper proposes a UML profile for constructing reliable embedded systems that take into account the external environments. Embedded systems not only affect their external environments through actuators but also are affected by their environments through sensors. The UML profile provides a set of stereotypes for representing systems, environments and associations among them. These model elements can be constrained by OCL descriptions. We also provide a UML model editor for supporting this UML profile. The model elements that violate the OCL descriptions can be detected by the editor. Using the UML profile, we can describe systems and their external environments rigorously.

Keyword UML Profile, embedded system, external environment

1. はじめに

組込みシステムは，様々な環境で利用され，その影響に応じて動作する．そのため，システムの動作検証では，外部からの影響を考慮する必要がある．

我々は，組込みシステムの外部からの影響を分析する手法として，外部環境分析手法を提案している[1][2][3]．この手法では，開発対象となる組込みシステムの仕様から外部環境のモデルを作成する．このようにして作成した外部環境モデルを，組込みシステムの動作検証に用いることで，システム外部の要素に起因する不具合を検出することが可能となる．

外部環境のモデルを用いて，組込みシステムの不具合をより効果的に検出するためには，外部環境の記述に厳密さが必要となる．しかし，手法の中で外部環境を記述するために用いられている表記法は通常の UML [4] であり，外部環境の厳密な記述は容易ではない．例えば，ある組込みシステムが持つ光センサが，外部環境の要素である環境光と関連を持っている場合，図 1のような記述では，その関連が環境光の観測を意味しているのか，ノイズとしての入力を意味しているのかなどが明確ではない．そのため，分析者がノートなどを用いて補足しなければな

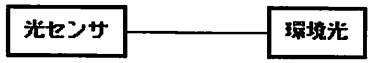

図 1：通常の UML 表記の例

らない．しかし，これではモデル自体には十分な情報を含められず，組込みシステムの動作検証で用いる際に，人による判断が必要になる．

そこで本稿では，組込みシステムの外部環境の厳密な記述を容易にする UML プロファイルと，外部環境モデルの要素に対する OCL（Object Constraint Language）[5] による制約の記述について述べる．これは図 2 で示す通り，外部環境を考慮した組込みシステムの動作検証を行う為の外部環境モデルに，厳密な定義を与えるためのものである．

以降，2 節では本プロファイルの定義と，これを用いた外部環境分析の例について述べる．3 節では，UML の拡張メカニズムを説明し，本プロファイルに基づく拡張 UML メタモデルを示した上で，OCL によって記述した外部環境に対する制約について述べる．その後，4 節では外部環境の記述のための支援ツールについて述べ，5 節で本稿のまとめと今後の課題を述べる．

2. 外部環境記述用 UML プロファイル

2.1. UML プロファイルの概要

本プロファイルは，UML に対する外部環境の記述のための拡張を定義している．このプロファイルに基づいて外部環境を記述することで，その厳密な記述が容易に行えるようになる．

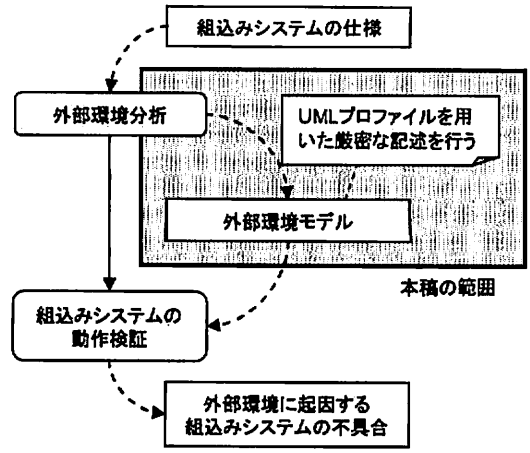

図 2：本稿の研究における位置づけ

外部環境記述用 UML プロファイルでは，組込みシステムの外部環境を含めた記述のために，システムを構成する要素とそれらの関連に関する拡張を定義している．具体的には，クラスに対する拡張として，<<Context>>, <<Hardware>>, <<Software>>の 3 種類のステレオタイプを，関連に対する拡張として，<<Observe>>, <<Control>>, <<Noise>>, <<Transfer>>, <<Affect>>の 5 種類のステレオタイプを定義している．表 1に本プロファイルの定義を示す．

2.2. 適用例

この節では，本 UML プロファイルを用いた外部環境分析の例を示す．分析の対象とする組込みシス

表 1：外部環境記述用 UML プロファイルの定義

名前	適用する要素	定義
<<Context>>	Class	組込みシステムの動作に影響する外部環境の要素を意味する．システムに対する外部からの入力やそれに影響を与える要素がこれに分類される．
<<Hardware>>	Class	組込みシステムを構成するハードウェアを意味する．
<<Software>>	Class	組込みシステムを構成するソフトウェアを意味する．
<<Observe>>	Association	ハードウェアが外部環境の要素を観測する関係を意味する．
<<Control>>	Association	ハードウェアが外部環境の要素を制御する関係を意味する．
<<Noise>>	Association	ハードウェアが外部環境の要素を観測する際に，観測の対象以外の要素が混ざった状態で入力されることがある．このときのハードウェアと観測の対象以外の要素との関係を意味する．
<<Transfer>>	Association	ハードウェアが外部環境の要素を観測する際に，直接的に観測することができず，別の要素に変換しなければならない場合がある．このときの要素同士の変換の関係を意味する．
<<Affect>>	Association	外部環境の要素が変換される際に，ハードウェアが観測したい要素とは別の要素が影響を与える場合がある．このときの変換後の要素と影響を与える要素の関係を意味する．

- ライントレーサは地面に引かれたラインに沿って走行する自律走行ロボットである．
- 地面の色は白，ラインの色は黒とする．
- ラインの検出には1個の光センサを用いる．
- 走行体の進行方向は，前輪の向きをステアモータで変化させることで制御する．
- 走行体の速度は，ドライブモータのパワーを変化させることで制御する．
- 光センサの受光部への入射光が「暗い」と「ライン上」，「明るい」と「ライン外」と判断する．
- ステアモータやドライブモータは，回転方向と回転の強さを制御できる．

図 3：ライントレーサの仕様（簡易版）

テムは，ライントレーサとする．ライントレーサの仕様は図 3のとおりである．

外部環境分析の手順は，外部環境の構造の定義(Step1)，外部環境の変化の定義(Step2)の2ステップからなる．Step1 では，対象となる組込みシステムがどのような要素から影響を受け，どのような構造を持っているかを明確にする．Step2 では，Step1 で得られた外部環境の要素がどのように変化するかを明らかにし，それをもとに外部環境の状態を定義する．このようにして得られたものが，外部環境のモデルとなる．この中で，本プロファイルにより拡張されるのは，Step1 の結果である外部環境の構造の記述である．図 4に，本プロファイルに基づくライントレーサの外部環境の構造の記述を示す．

(1) 外部環境の構造の定義（Step1）

Step1 で外部環境の構造を定義する際には，まずシステムが持つハードウェアに着目し，それらがシステムに対して持つ役割を考える．ライントレーサが持つハードウェアの場合，光センサが「ラインとの位置関係の観測」，ステアモータが「進行方向の制御」，ドライブモータが「速度の制御」である．これらをもとにして得られるのが，「ラインとの位置関係」，「進行方向」，「速度」の3つの要素である．これらを起点としてその他の外部環境の要素を抽出していく．今回は簡単のために「ラインとの位置関係」のみ分析を進める．

分析の起点となる要素を抽出したら，それらがハードウェアで観測，制御できる形になるまでの変換の過程を明確にし，そこから新たな外部環境の要素を追加する．ライントレーサの場合，「ラインとの位置関係」は「真下の地面の色」に変換され，それがさらに「地面からの反射光」へと変換されることで，光センサが検出する「受光部への入射光」になる．これにより，「光センサ」は「地面からの反射光」を

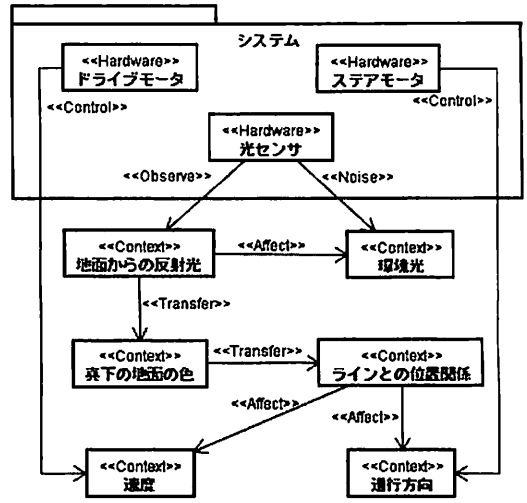

図 4：ライントレーサの外部環境モデル

- 上限を決定する要因
- 下限を決定する要因
- 対応関係を持つ要因
- システムによる観測を阻害する要因
- システムによる制御を阻害する要因

図 5：外部環境を洗い出すためのガイドワード

観測し，これが「真下の地面の色」を経て「ラインとの位置関係」から変換されると分かる．

次に，抽出した外部環境の要素に影響を与える要素を洗い出し，それを外部環境に加える．このとき，洗い出しのヒントとして図 5に示す5つのガイドワード [6] を用いる．ガイドワードとは，物事を導き出すためにあらかじめ設定されるキーワードである．例えば，「対応関係を持つ要因」というガイドワードを参考にすることで，「地面からの反射光」が地面への入射光に対応して変化することを導き出せる．このとき，地面への入射光となるのは，ライントレーサの周囲にある「環境光」なので，それを外部環境に加える．さらに，「システムによる観測を阻害する要因」というガイドワードから，光センサの受光部への入射光として，「地面からの反射光」以外に「環境光」も入力されることが導き出される．これは，「光センサ」へのノイズとなる．また，「ラインとの位置関係」は，「進行方向」と「速度」の影響によって変化するので，その関係も書き加える．

(2) 外部環境の変化の定義（Step2）

Step2 では，Step1 で抽出した外部環境の各要素が

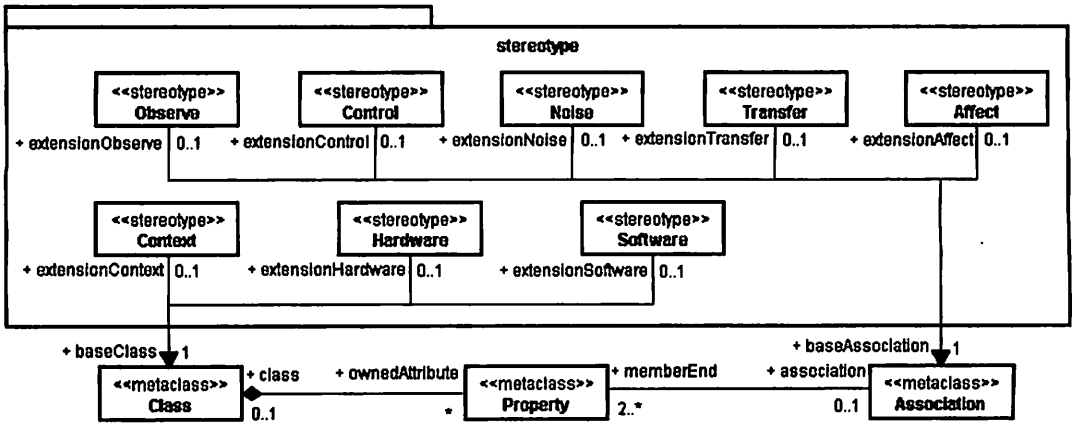

図 6：外部環境記述用 UML プロファイルの拡張 UML メタモデル（一部）

取りうる状態を定義し，それらの遷移関係とその遷移条件を明確にする．「地面からの反射光」の場合，その取りうる状態は「明るい」，「暗い」の二つであり，これらの間には，「明るい→暗い」，「暗い→明るい」，「明るい→明るい」，「暗い→暗い」の４つの遷移関係がある．これらの遷移条件は，「真下の地面の色」，「環境光」によって決まる．「暗い→明るい」の遷移条件の例として，「真下の地面の色」が「白い→黒い」と遷移した場合がある．

このような外部環境モデルの各要素が取りうる状態を組み合わせることによって，システム動作環境を網羅的に想定でき，システムの不具合の発見が容易になる．例えば，「真下の地面の色」が「黒い」状態のときに，「環境光」が「弱い→強い」の遷移をすると，「地面からの反射光」が「暗い→明るい」と遷移し，システムが誤動作する不具合が発見できる．このとき，外部環境記述用 UML プロファイルを用いていれば，<<Affect>>関連によって<<Transfer>>関連の対応関係が破られていることが分かり，不具合につながる可能性があることが分かりやすくなる．

3. OCL による外部環境記述の制約

3.1. 外部環境記述の制約の概要

本プロファイルでは，記述できるモデルに制約を与えている．本プロファイルは，外部環境の厳密な記述の実現を目的としているが，表記法がどれだけ厳密でも，記述したモデルが誤りを含んでいては意味がない．そこで，モデルに制約を与え，その制約が満たされているかによって，外部環境が正しく記述できているかを判断できるようにした．

本プロファイルにおいて，外部環境のモデルに対する制約は OCL [5] で記述されている．OCL とは，オブジェクト指向モデルに対して制約を与える言語であり，UML の仕様の一部として定義されている．

3.2. UML プロファイルの UML 拡張メカニズム

OCL による外部環境の記述に対する制限の記述を説明する前に，UML プロファイルによる UML 拡張のメカニズムについて説明する．これは，OCL がオブジェクトモデルに対して情報を追加する言語であり，今回情報を追加する対象が本プロファイルによる拡張 UML メタモデルであるためである．

UML プロファイルは，UML メタモデル内の Class や Association といったメタクラスに対して拡張を施すためのメカニズムである．実際には，拡張したいメタクラスに対して，ステレオタイプを意味するクラスから Extension と呼ばれる特殊な関連を持たせることで，拡張を実現している．

本プロファイルの拡張 UML メタモデルの一部を図 6に示す．この図の中で<<stereotype>>と付けられたクラスが，本プロファイルで定義されたステレオタイプとなる．また，黒塗りの矢印は，Extension のことである．そして，<<stereotype>>クラスと Extension で関連付けされた<<metaclass>>クラスが拡張の対象をあらわしている．

3.3. OCL による外部環境の制約の記述

本節では，外部環境記述の制限を記述した OCL 式の読み方について図 6を用いて説明し，これが外部環境の記述に制限を与えていることを示す．本プロファイルで定義した OCL 式を表 2に示している．

表 2から<<Hardware>>に対する「<<Observe>>か

<<Control>>を持つ Association を合わせて一つだけ持つ」の制約を説明する．この制約は，ハードウェアが，システム外部の観測または制御のいずれか一つの役割を必ず持っていることを意味している．以降で，OCL 式がこれを表現していることを示す．

OCL 式の起点となるオブジェクトは何に対する制約かで決まるので，この例では Hardware を起点として OCL 式を見ていけば良い．まず，

```
inv: baseClass.ownedAttribute.association
```

までを説明する．先頭の「inv:」はその式がオブジェクトの生存期間中，常に真でなければならないという，不変条件と呼ばれる制約であることを示している．それ以降に記述されている部分では，メタモデル内の Hardware クラスから関連をたどることで得られる Association クラスのオブジェクトの集合が返される．次の

```
->select(extensionObserve->notEmpty()
    or extensionControl->notEmpty())
```

では，直前で得られた集合の要素からカッコ内の条件式が真となる要素のみを含んだ部分集合が返される．この条件式は，ステレオタイプとして<<Observe>>もしくは<<Control>>を持つ Association において真となるため，ここではその集合が返される．そして，最後に集合の要素の個数を返す size() を用いて，

```
->size() = 1
```

と記述することで，<<Hardware>> に対する「<<Observe>>か<<Control>>を持つ Association を合わせて一つだけ持つ」という制約を表現する OCL 式となる．

4. 支援ツールによる外部環境の記述

本 UML プロファイルに基づいた外部環境の記述には，それに特化したツールを利用したい．汎用の

表 2：外部環境記述用プロファイルで定義した OCL 式（一部）

対象	区分	OCL
<<Context>>	Class	意図：Association のない<<Context>>はシステムと無関係であり，記述する必要はない． 内容：必ず Association を持つ． 記述： inv: baseClass.ownedAttribute.association->notEmpty()
<<Hardware>>	Class	意図：<<Hardware>>は，<<Context>>を<<Observe>>もしくは<<Control>>するという役割を一つだけ持つ． 内容：<<Observe>>か<<Control>>を持つ Association を合わせて一つだけ持つ． 記述： inv: baseClass.ownedAttribute.association ->select(extensionObserve->notEmpty() or extensionControl->notEmpty()) ->size() = 1
		意図：<<Hardware>>は<<Software>>と協調動作することでその役割を果たす． 内容：<<Software>>を持つ Class との Association を一つ以上持つ． 記述： inv: baseClass.ownedAttribute.association ->select(memberEnd.class.extensionSoftware->notEmpty())->size() >= 1
		意図：<<Noise>>しか受け取らない<<Hardware>>はシステムにとって意味がない． 内容：<<Noise>>を持つ Association を持つならば，<<Observe>>を持つ Association も持つ． 記述： inv: baseClass.ownedAttribute.association.extensionNoise->notEmpty() implies baseClass.ownedAttribute.association.extensionObserve->notEmpty()
<<Software>>	Association	無し．
<<Observe>> <<Control>> <<Noise>>	Association	意図：<<Observe>>，<<Control>>，<<Noise>>は，<<Hardware>>と<<Context>>の間の関係を表すものである． 内容：<<Hardware>>を持つ Class と<<Context>>を持つ Class の間にしか存在しない． 記述： inv: baseAssociation.memberEnd.class ->forAll(extensionHardware->notEmpty() or extensionContext->notEmpty()) and baseAssociation.memberEnd.class.extensionHardware->notEmtpy() and baseAssociation.memberEnd.class.extensionContext->notEmpty()
<<Transfer>> <<Affect>>	Association	意図：<<Transfer>>，<<Affect>>は，<<Context>>同士の間の関係を表すものである． 内容：<<Context>>を持つ Class 同士の間にしか存在しない． 記述： inv: baseAssociation.memberEnd.class->forAll(extensionContext->notEmpty())

UMLツールを用いることもできるが，外部環境の記述に特化したものを用いたほうがモデル作成の効率が良いためだ．そこで我々は，Eclipse [7] プラグインとして開発中の「拡張可能な UML モデルエディタ」を用いてこれを実現する．

「拡張可能な UML モデルエディタ」は，ユーザによる UML メタモデルの拡張と，それに対応したモデルエディタの自動生成という機能を持っている．UML メタモデルの拡張機能では，一部のメタクラスのみに拡張を許し，これを拡張ポイントとして UML メタモデルを拡張することができる．Class や Association も拡張ポイントとなっているので，図 6 に示したメタモデルをそのまま記述することで，外部環境の記述に特化したモデルエディタが生成できる．拡張エディタ生成機能を用いて生成した外部環境記述用モデルエディタの実行画面を図 7に示す．

さらに，UML メタモデルの拡張機能の一つとして，OCL による制約の追加も可能である．これにより，モデルエディタの検査機構が拡張される．そのため，3 節で述べた OCL による外部環境記述の制約を UML メタモデルに与えることで，外部環境の記述を検査することができる．例えば，図 7において光センサと環境光の関連が<<Observe>>だった場合，光センサが<<Observe>>を 2 つ持っていることになるため，表 2の<<Hardware>>に関する制約の一つが破られていることが検出される．

5. まとめと今後の課題

本稿では，組込みシステムの外部環境分析における外部環境の記述のための UML プロファイルについて述べた．このプロファイルを用いることで，外部環境をより厳密に記述できるようになる．

また，このプロファイルでは，OCL による UML メタモデルの制約の記述も行っている．これにより，このプロファイルに基づいて記述された外部環境モデルの構造上の誤りを検出することが可能となる．

我々は今後，外部環境をさらに厳密に記述できるよう，本プロファイルを発展させていく．例えば，いくつかのステレオタイプを細分化し，OCL による制約をより詳細に与えていく予定である．例えば，<<Hardware>>は，<<Sensor>>や<<Actuator>>などに区分することを考えている．

また，外部環境を含めた組込みシステムの検証を Alloy [8] などの形式手法を用いて行うための方法も検討していく予定である．このとき，外部環境のモデルを形式言語に変換することが必要であるが，本プロファイルに基づいて記述された外部環境モデルを用いると，このための負担を軽減することができると考えている．なぜならば，外部環境の構造に対する不変条件として，本稿で定義した OCL による記述が利用できると考えられるためである．

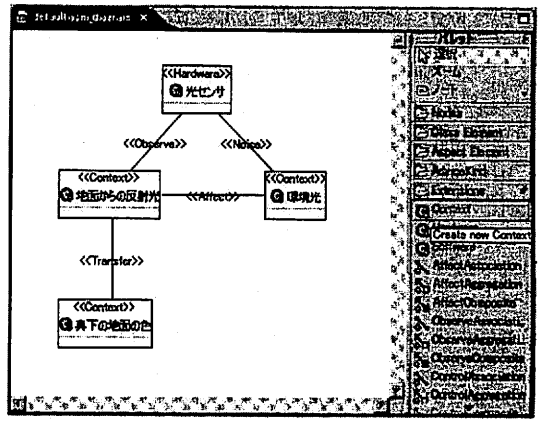

図 7：拡張可能な UML モデルエディタ実行画面

さらに，外部環境のモデルから様々な形式手法に対応する複数の形式言語へ，機械的に変換できないか検討していく予定である．形式手法には様々な種類があり，それぞれに得意とする分野が異なっている．そのため，検証を行いたい項目ごとに，それに適した形式検証手法を用いることができるとより効果的と考えられる．これが実現すれば，外部環境を考慮した組込みシステムの動作検証において，様々な形式手法を容易に活用できるようになる．

文　献

[1] 鷲見毅，平山雅之，鵜林尚靖，“組込みシステムにおける外部環境の分析，”情報処理学会 ソフトウェア工学研究会，SE-146, pp.33-40, 2004.

[2] 鷲見毅，平山雅之，鵜林尚靖，“組込みシステムにおける動作条件分析手法の提案，”電子情報通信学会 ソフトウエアサイエンス研究会 (2005.8.4-5)，pp.19-24，2005.

[3] 金川太俊，瀬戸敏喜，鵜林尚靖，鷲見毅，平山雅之，“組込みシステムにおける外部環境分析の提案，”第 8 回 組込みシステム技術に関するサマーワークショップ SWEST8 予稿集，pp.75-82，2006.

[4] OMG UML Resource Page, http://www.uml.org/.

[5] ヨシュ・ヴァルメル，アーネク・クレッペ，竹村 司（訳），“UML/MDA のためのオブジェクト制約言語 OCL 第 2 版，”エスアイビー・アクセス，2004.

[6] Nancy G.Leveson, “Safeware : System Safety and Computers,” Addison-Wesley Pub (Sd), 1995.

[7] Eclipse.org, http://www.eclipse.org/.

[8] Daniel Jackson, “Software Abstractions: Logic, Language, and Analysis,” The MIT Press, 2006.