ソフトウェア工学 140－2
（２００３．３．６）

組込みソフトウェア開発におけるユースケース分析・設計方式の提案

細川 卓誠[i]　　鶴見 知生[ii]
岡本 鉄兵[iii]　　小泉 寿男[i]

オブジェクト指向によるシステム開発では，システムの機能要件を定義するために，UML (Unified Model Language) のユースケース・ダイアグラムを用いる．しかしながら，ユースケース・ダイアグラムのユースケース記述は，詳細な記述内容やユースケース抽出のガイドラインは定められていないため，分析者によって記述される機能要件の粒度や品質が異なる．その結果，以後の設計工程で用いる他のダイアグラムにシステムの機能要件が正確に反映されないという問題が発生することがある．

本稿では，ユースケース・ダイアグラムの拡張によって，開発対象システムの分析と設計をシームレスに行う分析・設計方式を提案し，組込みシステムのソフトウェア開発への適用と評価を行う．本方式では，ユースケース・ダイアグラム上に新たにコントローラとアクションと呼ばれる 2 つのモデルを追加した拡張ユースケース (XUC : eXtended Use-Case) を用いる．アクションは，システムの機能要件を実現するための単機能部品であり，ライブラリ化することで再利用の対象とする．コントローラは，システムの機能要件毎に存在し，機能要件の内容から適用された 1 つ以上のアクションを制御する．XUC を用いることによって，システムの機能要件と実装上の構成との対応関係を明確にし，上流工程である分析の段階で実装までの見通しの立ったシームレスなシステム開発の実現を目指す．

A Proposal of Use-Case Analysis / Design Method in Embedded Software Developments

HOSOKAWA Takanobu[i], TSURUMI Tomoo[ii],
OKAMOTO Teppei[iii] and KOIZUMI Hisao[i]

In object-oriented system development, UML (Unified Modeling Language) Use Case Diagrams are used to describe the functionality of a system. However, the particle size and quality of functional requirements are different by the engineers who analyzed and described it, because the Use Case Diagrams are not defined in detail about the contents of Use Cases description and the guideline of Use Cases extraction. Therefore, it becomes the cause of the problem in which the functional requirements for a system are not correctly reflected in other diagrams used at future design processes.

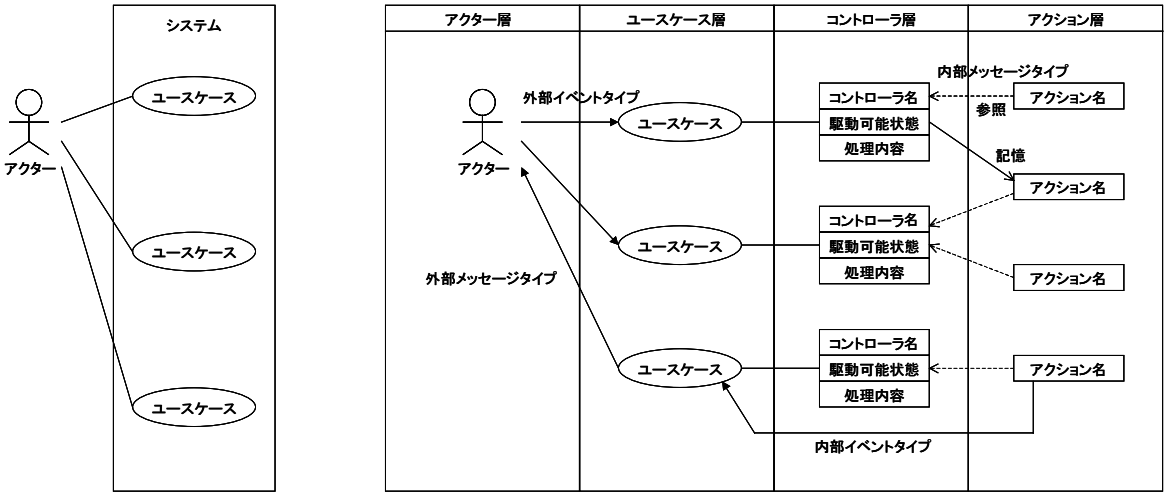
In this paper, we propose a system analysis / design method which makes analysis processes and design processes seamless by extending Use Case Diagrams, and we apply the method to embedded software development and evaluate it. The method uses XUC (eXtended Use Case) in which the two models calling Controllers and Actions are newly added to Use Case Diagrams. Actions are the single functional parts to implement the system, are reused by making library list. Controllers and Use Cases are the relations of 1 to 1. And Controllers controls one or more Actions applied from functional requirements. By using XUC, the relation between analysis processes and design processes is clarified. And XUC aims at realization of a systems development method in which the implementation of the system can foresee in analysis phases.

1. はじめに

本稿における提案の適用領域は，組込みシステム (Embedded System) とする．組込みシステムは，我々の身の周りにある電化製品のほとんどのものに応用されている．近年，応用分野の多様化とそれを支える技術基盤の進歩により，組込みシステム（特にソフトウェア）は，大規模化・複雑化する傾向にある．その一方で，製品サイクルの短期間化も進んでいることから，組込みシステムの開発生産性の改善が課題となっている．このような危機的状況を解決するために，成果物の再利用性と仕様変更に対する柔軟性に優れているとして，ビジネスアプリケーション領域において既に浸透しているオブジェクト指向開発技術を組込みシステム開発に適用することが注目されている．

オブジェクト指向のシステム開発では，仕様書の記述に UML (Unified Modeling Language) を用いる．システム開発エンジニアは，システムを UML で規定された複数のダイアグラムを用いて開発対象を多面的な視点からに捉え

[i] 東京電機大学大学院理工学研究科
Graduate School of Science and Engineering, Tokyo Denki University
[ii] 横河電機株式会社
Yokogawa Electric Corporation
[iii] ペンタックス株式会社
PENTAX Corporation



（a） ユースケース･ダイアグラム　　（b） XUC : eXtended Use Case

図 1　ユースケース･ダイアグラムと XUC

て，開発を行う．このモデリング作業を通じて問題領域や問題解決の方針，戦略などを決定する．UML では，システムの機能要件を定義するためにユースケース・ダイアグラムを用いる．しかしながら，ユースケース・ダイアグラムのユースケース記述では，詳細な記述内容やユースケース抽出のガイドラインは定められていないため，分析者によって記述される機能要件の粒度や品質が異なる．その結果，以後の設計工程で用いる他のダイアグラムにシステムの機能要件が正確に反映されないという問題が発生することがある．

著者らは，この問題を解決するための 1 つの手段として，ユースケース・ダイアグラムの拡張によって，システムの機能要件を定義する過程を通じて，分析と設計をシームレスに行うことのできる拡張ユースケース (XUC : eXtended Use-Case) を考案し，プロトタイプを開発した．XUC では，コントローラとアクションと呼ばれる 2 つの必要最小限の設計情報をユースケース・ダイアグラムに付加することによって，システムの機能要件と実装上の構成との対応関係を明確にし，上流工程である分析の段階で実装までの見通しの立ったシームレスなシステム開発の実現と，ソフトウェア部品の再利用の見通すことも可能なシステム開発の実現を目指す．

本稿では，開発した XUC のプロトタイプを提案し，適用例を挙げて評価を行い，最後にまとめる．

2. ユースケース記述における問題点

オブジェクト指向開発における出発点は，ユースケース・ダイアグラムを用いてシステムの機能要件を定義することである．ユースケース・ダイアグラムでは，アクターと呼ばれるサービスを受ける対象と，ユースケースと呼ばれるサービスを提供するシステムの満足すべき機能やサービスとの間の相互関係を記述する. また, ユースケースは，テストケース作成の情報源としても用いられる．

このように，ユースケースを記述することは，システム開発において最も重要なシステムが満足すべき機能要件を定義する作業であるため，ユースケースに関しては，ユースケースの解釈や記述項目の提案，形式的な構造の定義など様々な研究が進められている．しかしながら，一般にユースケースは，自然言語で記述するため，その表現の豊富さゆえ，一義的に記述することが難しく，曖昧なものになりがちである．また，ユースケース・ダイアグラム上でのユースケース記述は，システムの内部をブラックボックスとして捉えるため，実現手段までは規定しない．このような種々の理由から，分析者によって記述される機能要件の粒度や品質が異なる場合がある．その結果，以後の設計工程で用いる他のダイアグラムにシステムの機能要件が正確に反映されないという問題が起こり得る．

3. XUC：拡張ユースケースの提案

2 章で述べたユースケース記述における問題点を解決するために，著者らはユースケース・ダイアグラムをイベントドリヴンなモデルとして扱うことを目指して拡張することによって，分析と設計をシームレスに行うことのできるダイアグラムである拡張ユースケース (XUC : eXtended Use-Case) を考案し, プロトタイプを開発した. 以下, XUC の概要について述べる．

3.1　XUC の構成

XUC の構成について**図 1** に示す．XUC では，UML で規定されているユースケース・ダイアグラムと同様にアクターを記述するアクター層とユースケースを記述するユースケース層の他に，新たにコントローラ層とアクション層を加えた 4 つの層を用いて開発対象のシステムの機能要件を定義する．

以下に XUC で使用される構成要素について定義する．

構成要素は，それぞれを識別できるような個別の名称を持ち，イベントタイプおよびメッセージタイプは，その方向を有向線で明確に記述する．

アクター：

システムと相互作用するオブジェクト（ユーザ，保守管理者，外部システムなど）であり，内部イベントタイプを発行するサブアクター（システムに内蔵されたタイマなど）については，XUC ではアクターとせずにアクションが内部イベントタイプを発行するものとして扱う．

ユースケース：

アクターがシステムに対して要求するサービス．または，システムが行うべき機能．アクターからの外部イベントタイプ，またはアクションからの内部イベントタイプによって駆動する．1 つのユースケースは，1 つのイベントタイプによってのみ駆動される．

外部イベントタイプ／内部イベントタイプ：

ユースケースを駆動させるためにシステムに与える指令の種類．スイッチやボタンなどアクターによってシステム外部から入力される外部イベントタイプと，タイマやセンサなどから入力されるシステム内部で発行される内部イベントタイプがある．ある 1 つのイベントタイプは，一般に属性値として複数個のイベントを持っている．

外部メッセージタイプ／内部メッセージタイプ：

システムがユースケースの駆動結果をアクターに対して出力する外部メッセージタイプと，システム内部のコントローラとアクションの間でやり取りされる内部メッセージタイプがある．内部メッセージタイプは，記憶と参照の 2 種類に区別して記述する．イベントタイプと同様に，ある 1 つのメッセージタイプは，一般に属性値として複数個のメッセージを持っている．

コントローラ：

ユースケースと 1 対 1 の関係で存在するため，1 つのイベントタイプによってのみ駆動されるものとして扱う．対応したユースケースを実現させるために必要な 1 つ以上のアクションの手順を制御する．1 つのコントローラは，1 つの機能要件を実現するしくみを抽象化した XUC モデル上のオブジェクトであり，アクションの手順制御の詳細定義は，コントローラ毎に状態遷移図や状態遷移表，シーケンス図のいずれかの 1 つまたは複数の動的なモデルを記述して補完する．また，コントローラ間の関係は，常に互いに疎である． コントローラは，自身が駆動可能である状態内にいる時にアクターからの外部イベント，またはアクションからの内部イベントを受取ると駆動する．それ以外の状態では，イベントを受取っても全て無視をする．

アクション：

システムの機能要件を実現するための構成要素となる単機能の汎用ソフトウェア部品であり，ライブラリ化することによって再利用の対象とする．アクションは，単体または複数が組合わさることで機能要件を満足する．ただし，アクション同士における関係は，互いに疎であり，

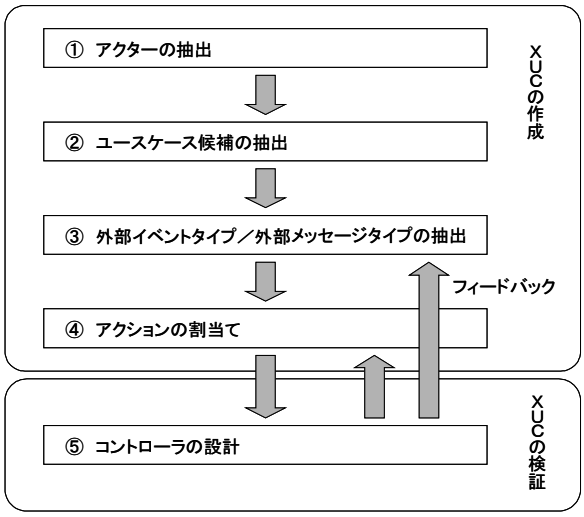


図 2　XUC の作成フロー

コントローラを介することではじめて相互作用することができる．アクションの粒度は，開発するシステムドメインによって様々に変わり，ある 1 つのコントローラとアクションのセットが別のドメインでは，1 つのアクションとして扱われる場合もある．

3.2　XUC の作成フロー

XUC の大きなの特徴は，XUC を記述するための明確な手順とガイドラインを提供していることである．したがって，XUC を用いる分析者は，この手順とガイドラインに則って分析・設計を進めればよい．XUC の作成フローは，以下の 5 工程に分かれる．

① アクターを抽出する
② ユースケースの候補を抽出する
③ 外部イベントタイプ／外部メッセージタイプを抽出する
④ アクションを割当てる
⑤ コントローラを設計する

①～④工程を実行すると XUC を記述することができる．そして，⑤工程で記述した XUC を検証し，必要に応じてアクションを追加などする場合は④工程に，イベントタイプを追加・削除などする場合は③工程にフィードバックすることで XUC を洗練させてゆく（**図 2**）．以下に XUC 作成フローの 5 工程について詳細を述べる．

（1）アクターの抽出

この工程では，システムの外部にあってシステムと相互作用するアクターを抽出する．

（2）ユースケース候補の抽出

この工程では，アクターがシステムに対して要求するユースケースの候補を抽出する．以後の手順を踏んでゆくことでユースケースは自ずと洗練されてゆくので，この段階では，ユースケースの候補として考えられるものを列挙してゆくだけでよい．

(3) 外部イベントタイプ／外部メッセージタイプの抽出

この工程では，ユースケースを駆動する合図の入力となる外部イベントタイプと，ユースケース駆動結果の出力となる外部メッセージタイプを抽出する．外部イベントタイプおよび外部メッセージタイプの抽出には，

- ユースケースの内容から，そのユースケースを駆動するイベントタイプ／メッセージタイプを抽出するアプローチ
- アクターが操作するシステムのスイッチやボタンなどのシステム外部とのインタフェースからイベントタイプ／メッセージタイプを抽出するアプローチ

以上2つのアプローチがある．こうして抽出したイベントタイプ／メッセージタイプを用いてアクターとユースケースを有向線で結び，それぞれの関係を明確にする．

このとき，ある1つのユースケースに複数のイベントタイプが関係付けられた場合は，ユースケースを分割し，1つのユースケースは，1つのイベントタイプを持つようになるまで分割を繰り返す．このことは，「1つのユースケースは1つのイベントタイプによってのみ駆動される」という定義に基づいている．また，ユースケースの候補がアクターとの間に関係を何も持っていない場合は，このユースケースは，内部イベントタイプによって駆動されるユースケースであるとみなし，内部イベントタイプを発行するアクションを記述して，ユースケースとの間に有向線を用いて関係を明確にする．

(4) アクションの割当て

この工程では，ユースケースと1対1の関係で存在するコントローラに対して，そのコントローラが満足すべき機能要件を実現するために必要なアクションをライブラリから選択し割当てる．ライブラリからアクションを選択して再利用することを前提としているが，割当てるべきアクションがライブラリに存在しないときは，新たにアクションを作成し，ライブラリに追加登録することで，再利用可能なソフトウェア部品の充実を図る．

適用したアクションとコントローラ間の情報のやりとりに　ついては，内部メッセージタイプとして有向線を用いて関係を明確にする．とくに内部メッセージタイプは，記憶と参照の2種類に区別して記述する．

また，1つのコントローラに複数のアクションが集中している場合は，ユースケース抽出工程にフィードバックして，イベントタイプを分割し，ユースケースの再抽出を行うことが望ましい．

(5) コントローラの設計

この工程では，必要に応じてコントローラが制御するアクションの動的な振舞いについて，状態遷移図や状態遷移表，シーケンス図のいずれか1つまたは複数のモデルを用いて記述することで，XUC モデル上では明確にできないコントローラ内部の制御構造の詳細を補完する．

とくに組込みシステムのようなドメインでは，状態遷移表を用いたコントローラ詳細記述が起こりうる全てのケースについて検証できるので，適している．その結果，アクションを新たに追加する場合や，イベントタイプの追加や削除などを行う必要がある場合は，該当する前工程にフィードバックしてXUCを洗練させてゆく．

4. XUC の適用例 ーラジコンカー・システムの開発ー

- ユーザの持つ送信機からの信号を受信機で受取り,SW制御によってステアリング情報とアクセル情報が各サーボに通知され方向転換,速度調整を行う.
- ユーザから自動走行の信号を受信すると,内蔵されているプログラムを読むことで数種類のデモ走行を行う機能がシステムには備わっている.
- デモ走行はユーザからの停止信号を受信することで停止する.
- ユーザからの停止信号を受信しなくても,開始してから5分後には自動的に停止する.
- ラジコン本体のバッテリー残量を本体上のランプの点灯によりユーザに通知する.

図 3　ラジコンカー･システムの機能概要

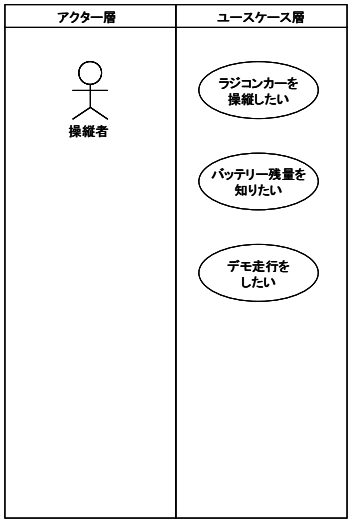


図 4　XUC の作成過程①

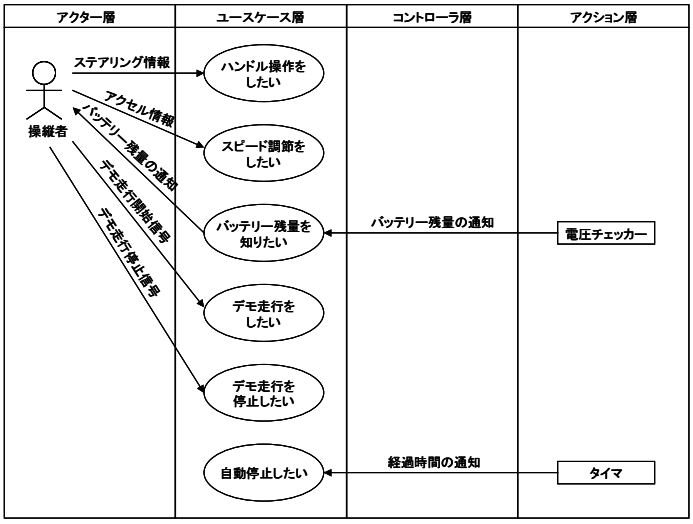


図 5　XUC の作成過程②

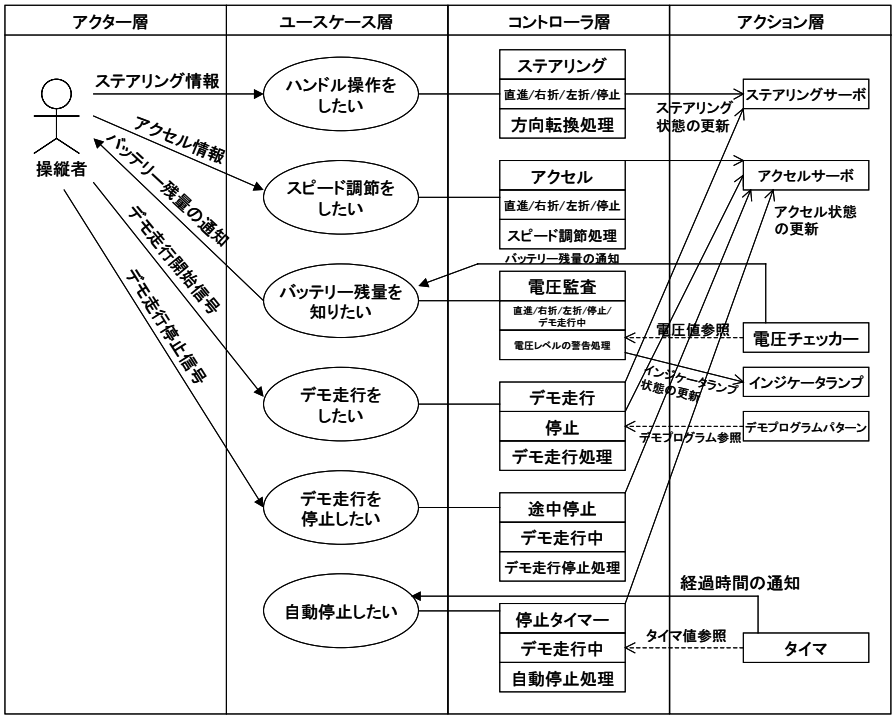


図 6　ラジコンカー･システムの XUC 記述例

3 章で提案した XUC をラジコンカー・システムの開発に適用した例について述べる．まず，**図 3** に開発対象となるラジコンカー･システムの機能概要を示す. 以下に XUC の作成フローに従って XUC を記述する.

（1）アクターの抽出

ラジコンカー・システムにおけるアクターは,「操縦者」である.

（2）ユースケース候補の抽出

ユースケース候補としては,「ラジコンカーを操縦したい」「バッテリー残量を知りたい」「デモ走行をしたい」を抽出する．この段階までの XUC 記述を**図 4** に示す.

（3）外部イベントタイプ／外部メッセージタイプの抽出

ユースケース「ラジコンカーを操縦したい」は，機能概要から 2 つの外部イベントタイプ「ステアリング情報」と「アクセル情報」を持つことがわかる．したがって，1 つのユースケースは 1 つのイベントタイプにのみ駆動されるという定義より，ユースケースを「ハンドル操作をしたい」と「スピード調節をしたい」の 2 つに分割する.

ユースケース「バッテリー残量を知りたい」は，機能概要からシステム内部でバッテリー残量をチェックしてその結果をランプの点灯で通知することがわかる. したがって，ユースケース「バッテリー残量を知りたい」は，外部イベントタイプを持たず，アクション「電圧チェッカー」が内部イベントタイプ「バッテリー残量の通知」を発行し，外部メッセージとして「バッテリー残量の通知」を発行することがわかる.

ユースケース「デモ走行をしたい」は，機能概要から 2 つの外部イベントタイプ「デモ走行開始信号」と「デモ走行停止信号」を持ち，さらにアクション「タイマ」が内部イベントタイプ「経過時間の通知」を発行し，5 分経過すると自動停止することがわかるので，ユースケースを「デモ走行したい」と「デモ走行を停止したい」と「自動停止したい」の 3 つに分割する.

この段階までの XUC 記述を**図 5** に示す.

（4）アクションの割当て

(3) の工程で分割したユースケース毎にコントローラを配置し，各コントローラが満足すべき機能要件を実現するために必要なアクションをライブラリから選択し割当てる. また，適用したアクションとコントローラ間の内部メッセージタイプを記述する.

以後，コントローラ設計の工程で，必要に応じてフィードバックをかけながら XUC を洗練してゆく．完成したラジコンカー・システムの XUC 記述例を**図 6** に示す.

5. XUC の評価

XUC を作成することによって，曖昧さを排除し，論理的に矛盾のない機能要件を定義することができた．また，上流工程の段階で，以後の設計工程へシームレスに移行することができることを確認した.

XUC では，ユースケースをイベントドリヴンであると規定したことによって，システムの入出力関係が明確になり，各機能要件を実現するコントローラ上で実装上の制御手順を定義するので, 組込みシステムの開発に適している. 今後の課題は、以下の通りである。

① ユースケースの階層化への対応

② ライブラリからのアクション適用の具体的方法
③ 異なる複数のイベントタイプから駆動されるユースケースの取扱い
④ コントローラのアクション化による再利用
⑤ XUC におけるフレームワークの検討

6. むすび

本稿では, ユースケース・ダイアグラムを拡張した XUC (eXtended Use Case) を提案し, XUC を組込みソフトウェア開発への適用例を述べ, 評価を行った. XUC についての今後の課題は, 5 章で述べた通りである. 今後は, 適用事例を拡げ, 本方式の有効性について更なる検証を行ってゆきたい. また, 組込みソフトウェアの生産性向上のメトリクスとして注目されている機能量を XUC から導出する手法についても検討する予定である.

参 考 文 献

1） Jacobson, I., Christersson, M., Jonsson, P., and Overgaard, G.: Object-Oriented Software Engineering: A Use-Case Driven Approach, Addison-Wesley, Reading, MA,1992.
2） 中谷多哉子, 玉井哲雄: ユースケースフレームワークの検討』, ソフトウェアシンポジウム 2000.
3） 中谷多哉子:ユースケースを用いる要求分析,SLagoon, 2001.
4） 大西淳, 郷健太郎: 要求工学, 共立出版, 2002.
5） 渡辺博之, 渡辺政彦, 堀松和人, 渡守武和記: 組み込み UML eUML によるオブジェクト指向組み込みシステム開発, 翔泳社, 2002.
6） Bruce Douglass : Real-Time UML, Second Edition Developing Efficient Object for Embedded Systems, 株式会社オージス総研訳, 渡辺博之 監訳, 翔泳社, 2001.
7） S.シュレィアー/ S.J.メラー: 続オブジェクト指向 システム分析 オブジェクトライフサイクル, 本位田真一, 伊藤潔 監訳, 近代科学社, 1995.
8） S.シュレイアー/ S.J.メラー: オブジェクト指向 システム分析 上流 CASE のためのモデル化手法,本位田真一, 山口亨 訳, 近代科学社, 1995.
9） 岡本鉄兵, 小泉寿男: 組込みソフトウェア開発における XML を用いた再利用支援方式, 第 136 回情報処理学会ソフトウェア工学研究会, 2002.
10）オブジェクトの広場, http://www.ogis-ri.co.jp/otc/hiroba/