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

2003－ＳＬＤＭ－110　(5)
2003／5／15

UMLとSystemCを用いた設計手法の移動通信システムへの適用事例

周東 雅之† 戸倉 綾† 山下 智規‡

†株式会社モバイルテクノ 〒239-0847 神奈川県横須賀市光の丘3-4YRPセンタ1番館
‡富士通株式会社 〒211-8588 神奈川県川崎市中原区上小田中4-1-1
E-mail: †{Masayuki.Sutoh, Aya.Togura}@moteco.co.jp, ‡tomo@jp.fujitsu.com

あらまし システム設計にオブジェクト指向，UML(Unified Modeling Language)，SystemCを取り入れた設計手法を検討し，移動通信システムに適用した．検討した手法では，システムの早期検証，モデルの再利用，異なる抽象度の混在を可能とし，開発期間の短縮や設計品質の向上が期待できる．本稿では，移動通信システムの無線伝搬路上りリンク部分の分析設計手順とUML記述効果，SystemCでの実装とシミュレーション結果を報告する．

キーワード UML，SystemC，オブジェクト指向，移動通信システム

Application Example to Mobile Communication System of Design Methodology using UML and SystemC

Masayuki SUTOH† Aya TOGURA† and Tomonori YAMASHITA‡

† Mobile Techno Corp. YRP1Bankan4F, 3-4, Hikarino-oka, Yokosuka, Kanagawa, 239-0847 Japan
‡ FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki, Kanagawa, 211-8588 Japan
E-mail: †{Masayuki.Sutoh, Aya.Togura}@moteco.co.jp, ‡tomo@jp.fujitsu.com

Abstract We examined the system design methodology using object oriented analysis, UML(Unified Modeling Language) and SystemC, and applied to the mobile communication system simulation. By this methodology, we can expect early verification, reuse constructed models and build with a mix-level abstraction. This article introduces mobile communication system especially uplink channel with base station(BS) and mobile station(MS) analysis using UML and effects that written in UML. Also, we show the simulation results by the implemented with SystemC.

Keyword UML, SystemC, Object Oriented, Mobile Communication System

1. はじめに

近年，システム開発において，開発対象の大規模化，高機能化，複雑化にもかかわらず，いっそうの開発期間の短縮，コストの削減が求められており，この問題を解決する様々な開発手法が提案されている．例えばシステムLSI開発分野では，ソフトウェア開発分野で定着しているUML(Unified Modeling Language)と，ハードウェア動作を記述可能なシステム記述言語を用いて，方式検討から実装まで一貫して開発する設計手法が提案されている[1][2][3][4].

UMLを採用する設計手法は，ノーテーションが厳密に規定されているため，これを用いない場合に比べ，開発担当者間で同じ概念の共有や仕様の誤認識を指摘し易いと言われている．また，UMLとシステム記述言語を併用することにより，要求仕様分析からアルゴリズム検討，その実装までをシームレスに開発可能であり，システムの早期機能検証やハードウェア/ソフトウェア分割検討もシステマティックに行なえ，開発期間の短縮や検証レベルの向上を期待できる．

一方，ソフトウェア開発の分野では，早い時期からオブジェクト指向技術が導入され，再利用性，保守性，拡張性の高い開発が行なえることが実証されている．

以上のことから，我々はソフトウェアやSoCだけでなくハードウェア及びソフトウェアから構成されるより大規模なシステムの開発においても，オブジェクト指向とUML，システム記述言語を用いることにより，1)仕様理解の容易化，2)システムとしての再利用化(変更容易化)，3)変更の追跡可能化，4)記述抽象度の異なるモデルの混在，を実現可能と考えた．

本稿では，検討した設計手法と移動通信システムへ適用した事例を報告する．

2. UML と SystemC を用いた設計手法概要

ソフトウェア開発手法で著名であるラショナルユニファイドプロセス(RUP)[5]を参考に，検討した開発手法の作業手順を以下に，フローを図 1に示す．

1) 開発対象とするシステム全体を，何が行われるべきかに着目し，大きな粒度でユースケースを作成し，分析を行なう．分析結果を，UML を用いて記述する．もし既にこのレベルの分析モデルが存在していれば，この作業を行なう必要はない．
2) 開発対象部分から，更に細かな粒度で複数のユースケースを抽出する．抽出されたユースケース毎に優先度をつけ，優先順位に従いイベントフローを記述する．このイベントフローから静的分析を静的構造図(UML ではクラス図)，動的分析をメッセージフロー(UML ではシーケンス図またはコラボレーション図)で記述する．記述した各 UML 図は，イベントフローを元にその正当性を検証する．
3) 検証済みのユースケース毎の分析結果を統合し，開発対象システム全体のクラス図を作成する．この段階で揃った統合クラス図，ユースケース毎のイベントフロー，クラス図，シーケンス図の一群を分析モデルと呼ぶ．
4) 分析モデルから実装言語や制約に基づき，各クラスの抽象度とインタフェース(シグネイチャ)を検討する．ここで検討したモデルを設計モデルと呼ぶ．設計モデルでは分析モデルのクラスは使用目的に応じて統合，分解される場合が多い．
5) 設計モデルを SystemC により実装する．また，実装結果は再利用を促進するため，設計モデルと関連付けて管理する．必要な抽象度で既に実装が完了しているクラスがあれば，再利用を検討する．

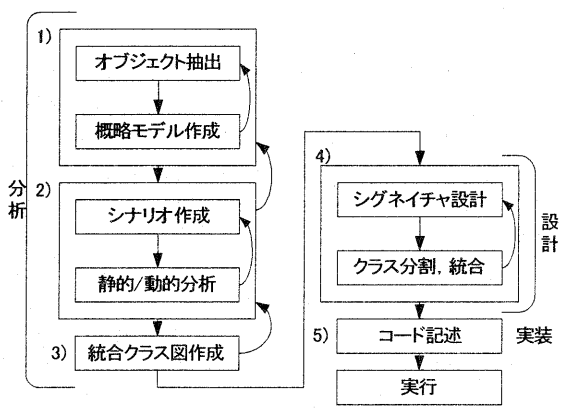

図 1 設計フロー

3. 分析作業

分析作業では，システム全体の分析と開発対象部分の分析を行なう．今回検討した手法を，移動通信システム上りリンク(移動局から基地局への通信)中の無線伝搬路部分に適用した．

3.1. システム全体の分析

対象となる移動通信システムは，図 2に示すように，複数の移動局が複数の基地局と無線伝搬路を介して通信を行なうシステムである．システム全体の分析において，いくつかの前提条件をつけ分析するモデルを明確にする．しかし，前提条件が多いほど汎用性の低いモデルとなるうえ，必要以上に細かな前提は逆にシステム本来の姿が見えなくなるため，必要最小限の前提条件をつけた．

[システム前提条件]
1) 上りリンクとする
2) 移動局，基地局をそれぞれ 1 台とする
3) 他の移動局からの干渉は，ガウス雑音と仮定する

上記 1)，2)の前提条件は，この前提条件が障害となるシステムにおいても，今回の分析結果は対象としているシステムの下層に相当するため，十分再利用可能なレベルであると考える．また，3)の前提条件は，移動局数が多い条件下では，他局間干渉は中央極限定理によりガウス雑音に近似できると言う統計的性質により証明されており問題が無いと考えた．以上から上記 3 つの前提条件は再利用に支障をきたさないと判断し，システム全体の分析を行った．以下にその結果を示す．

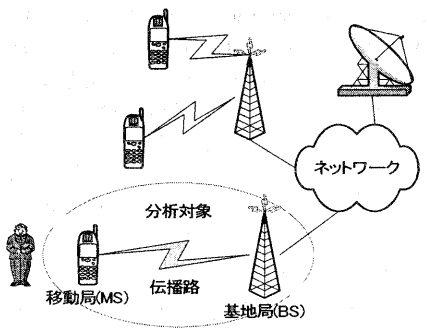

図 2 移動通信システムの概略図

[システムイベントフロー]
1) 移動局は送信情報(音声・文字 etc)を受け付ける．
2) 移動局は送信情報をシステムの方式に従った処理(符号化，変調 etc)を処理した送信データを作成する．

3) 移動局は送信データから電波を作成する.
4) 無線伝搬路(自然界)は電波に歪みを与える.
5) 基地局は電波を受信する.
6) 基地局は受信した電波をシステムの方式に従った処理(復調, 復号化 etc)を処理した受信データを作成する.

この結果, 図 3に示すシステムの概略クラス図が作成された.

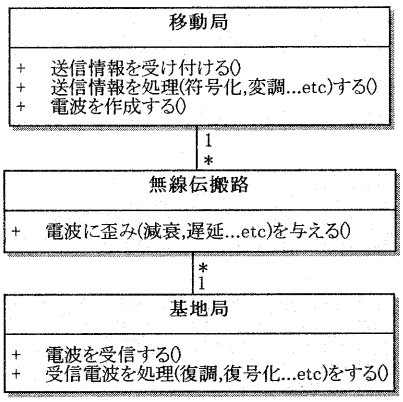

図 3 概略クラス図

3.2. 開発対象部分の分析

ここでは, 図 3の概略クラス図で示された開発対象部分である無線伝搬路に関して機能要求の明確化を行なうため, 無線伝搬路に関するユースケースを抽出する.

3.2.1. 機能要求の明確化

抽出した無線伝搬路に関するユースケースを, 以下に示すように優先度を付けて整理した.

1) 減衰する
2) 遅延する
3) 位相回転する
4) ガウス雑音を発生させる
5) マルチパスを発生させる

3.2.2. 静的側面の分析

機能要求の明確化において優先順位の上位に位置付けられた「減衰する」というユースケースに関して静的側面の分析を行なう. 以下にイベントフローを示す.

[前提条件]
- 移動局が 1 つ存在する
- 基地局が 1 つ存在する
- 移動局と基地局を結ぶパスが 1 つ存在する
- パスには減衰量が与えられている
- 移動局は移動しないものとする

[イベントフロー]
1) 移動局は送信データから電波を作成する
2) 移動局はパスに電波を送出する
3) 電波はパスによって減衰する
4) 基地局は電波を受信する

上記イベントフローより, ユースケース「減衰する」に関連するクラスとして「移動局」「基地局」「パス」「電波」を抽出した. 抽出したクラスから図 4に示す静的構造図(UML ではクラス図)を作成した.「減衰する」イベントフローより抽出されたクラスの中に, システム全体の分析中で既に抽出されたクラス「移動局」「基地局」が存在していることに注意する.

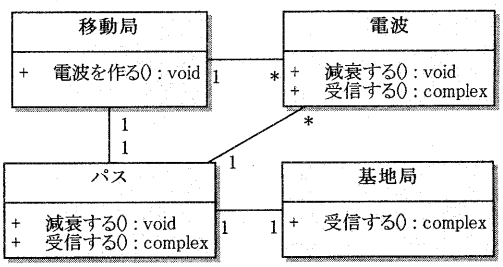

図 4 ユースケース「減衰する」の静的構造図 (クラス図)

3.2.3. 動的側面の分析

動的側面の分析では, システム全体の処理の流れと関連を開発対象部分に注目して整理する. ここでの作業は, クラスの分類と静的分析では抽出されないクラスを補足し, それらの動的な関連を明確にする. 具体的には, システム全体の分析で抽出されたクラスとユースケース「減衰する」で抽出されたクラスを Entity と性格付けられるクラスとし, ユースケース「減衰する」をコントロールする Control クラスを追加, コントローラとアクターとのインタフェースである Boundary クラスを設け, 各クラス間に流れる情報交換をメッセージフローとして記述する. 図 5にシーケンス図, 図 6にコラボレーション図を示す. 最終的にはコラボレーション図からユースケース「減衰する」の VOPC(View Of Participating Class)を作成する.

3.2.4. 個別ユースケースの検証

ユースケースを用いて, 作成したクラス図, シーケンス図, コラボレーション図の矛盾をチェックする. また, 該当ユースケースだけでなく, その他のユースケースを用いて静的構造に矛盾がないか, シーケンスに示されるオブジェクトの生存期間に矛盾がないか確認する.

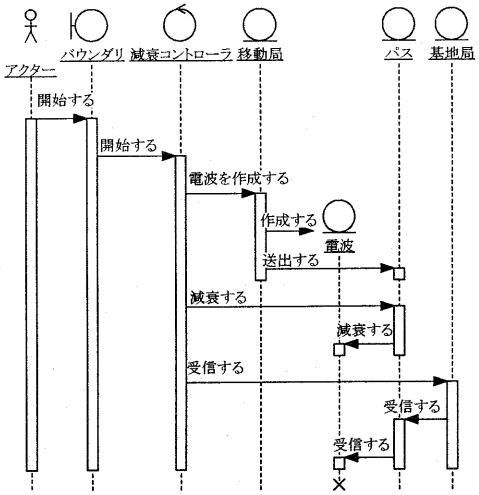

図 5 ユースケース「減衰する」の
シーケンス図

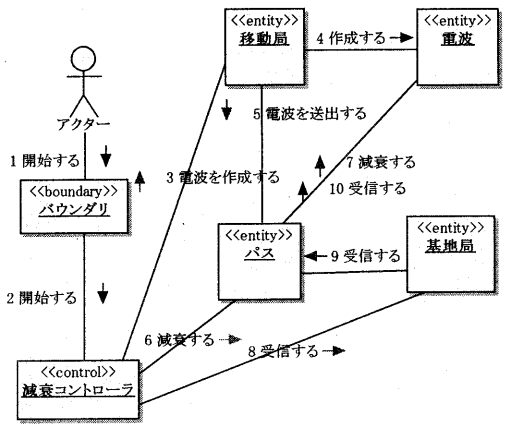

図 6 ユースケース「減衰する」の
コラボレーション図

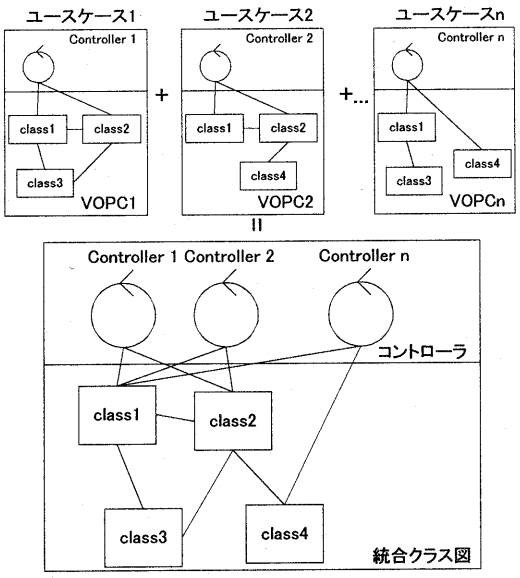

図 7 VOPC の統合イメージ

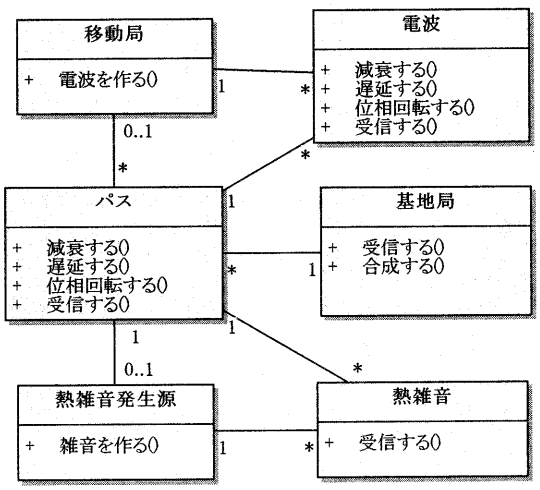

図 8 統合した無線伝搬路のクラス図

3.3. 分析クラス図の作成

前節では特定のユースケースに着目して VOPC を作成した．この時点では，ユースケース毎の VOPC が存在するはずである．ここでは機能要求の明確化でリストアップした全てのユースケースに対する VOPC を統合する．これにより，図 7に示す機能要求を満足する分析レベルの統合クラス図が得られる．得られたクラス図に対し，再度全てのユースケースに対する矛盾の有無をチェックすることで品質を確保する．図 8に統合された無線伝搬路のクラス図を示す．

4. 設計作業

設計作業では，SystemC により実装することを念頭に使用目的に応じた各クラスの抽象度とインタフェース(シグネイチャ)を設計する．もし C++や Java などその他の言語で実装する場合には，その言語にあったモデルをこの段階で検討することになる．今回，モデルの使用目的を方式検討としたため，全てのモジュールを時間概念のない Untimed Model として設計した．

分析レベルのクラス図から設計レベルのクラス図を生成する場合，3 つの考慮点がある．1 つ目は，クラ

スの統合/分配である．実装を考慮すると分析レベルのクラスを変更するほうが良い場合がある．今回は，電波クラス，熱雑音クラスをパスの属性とするかクラスのまま残すかを検討した．2 つ目は，機能と通信を分離するためにチャネルを配置するかを検討する．3 つ目はチャネルのデータ型をポインタにするか実体にするかを検討した．将来ハードウェアに実装する場合はポインタ型から変更する必要が生じるが，方式検討などでは速度面を考慮するとポインタ型にアドバンテージがあると予想できる．これらを整理すると表 1の 5 つの設計モデルを設計できる．設計モデルのクラス図を図 9～図 13に示す．なお，分析モデルと設計モデルの違いを明確にするため，減衰部分のみを示す．

表 1 設計クラスの実装方法

	電波/熱雑音	チャネルのデータ型(in/out)	
		MS-Path 間	Path-BS 間
(a)	クラス	実装なし	実装なし
(b)	パスの属性	実装なし	実装なし
(c)	クラス	ポインタ/ポインタ	ポインタ/実体
(d)	クラス	ポインタ/ポインタ	実体/実体
(e)	パスの属性	実体/実体	実体/実体

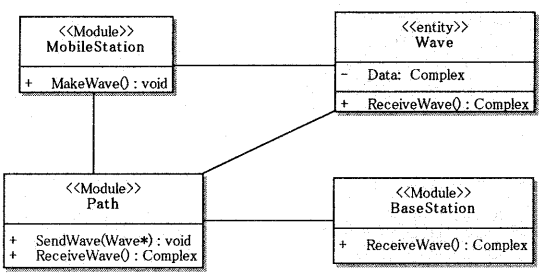

図 9 設計クラス図(a)

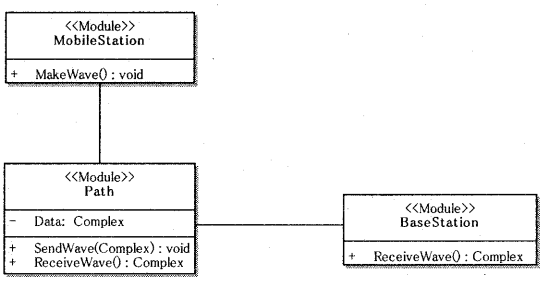

図 10 設計クラス図(b)

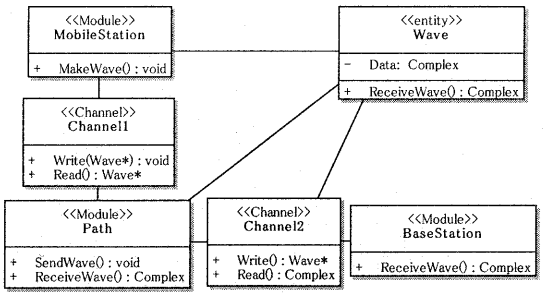

図 11 設計クラス図(c)

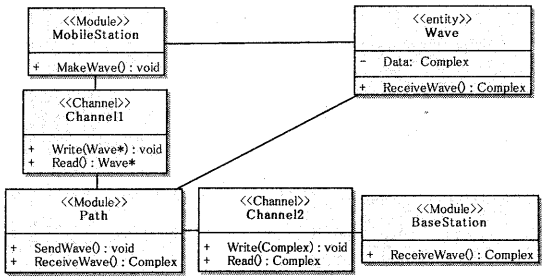

図 12 設計クラス図(d)

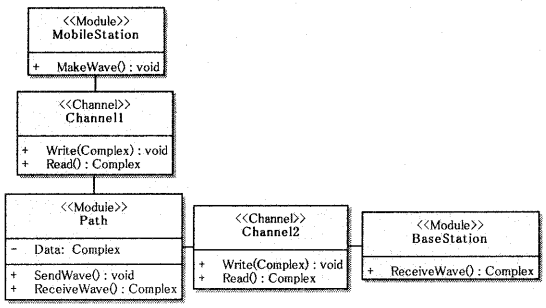

図 13 設計クラス図(e)

5. 実行と結果考察

今回は設計した 5 種のモデル(a)～(e)について，その特徴を比較検討するため，すべて SystemC により実装しシミュレーションを実行した．シミュレーションは，100000 サンプルの電波を移動局から基地局に送信させるものである．シミュレーション平均処理時間を図 14に示す．用いた測定環境は，Pentium4 1.8GHz, Memory 768MB, Windows2000, VC++6.0, SystemC2.0.1 である．結果より次の 2 つの特徴があることが分かる．

最初に，電波をパスの属性とした方が，電波クラスとして存在するよりも処理時間が少なかった．これは，

C++処理系の特徴であるが，モジュールとして実装した電波オブジェクトの生成/消滅に時間がかかることを示している．現実世界においても電波はパスの部分と考えることは不自然ではない．よって処理高速化のためモデル(a)よりモデル(b)の構造にするほうが良い．

次に，チャネルを導入したモデル(c)～(e)では全てに共通して処理時間が増大する結果となった．モデル(c)とモデル(d)の違いはチャネルでポインタか実体を扱うかが異なっている．

全体を通して方式検討のみが目的ならば，チャネル概念を実装する必要はなく，通常の C++で作成するのと同じ構造であるモデル(b)が最も処理時間が短く目的を達成するモデルと言える．しかし，実装まで一貫した設計を考慮すると，設計進捗状況によりモデル内に異なる抽象度のモジュールが存在する可能性もある．そのためにチャネルを実装しモジュール間の抽象度変換やプロトコル処理をチャネルに任せる方が柔軟に対応でき，方式検討から実装まで一貫した開発が可能になる．従って，(b)に対して(e)は約 2 倍の処理時間を必要とするが，チャネルを実装するメリットは大きく，目的によっては (e)の構成の設計モデルを開発しても良いと考えられる．

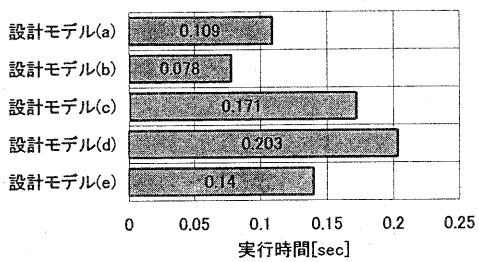

図 14 シミュレーション結果

6. まとめ

我々は，オブジェクト指向技術，UML と SystemC を取り入れた設計手法の分析，設計，実装フェーズの作業手順を定義し，それを移動通信システムモデルに適用した．

分析および設計時に UML を用いることにより，仕様を多角的な視点から見られるようになり，システム全体の仕様理解を容易にすることができた．また変更の追跡性は，ツールに依存する部分も残されているが，システムを変更するとき分析レベルから行なうトップダウン的な方法，設計レベルの変更によるボトムアップ的な影響範囲の識別も可能であり，さらに UML を用いたビジュアルなモデルでは変更箇所を追い易いことが分かった．

システム記述言語としてオブジェクト指向言語でありハードウェア記述も可能である SystemC を採用したことにより，抽象度の異なるモデルの混在を可能とした．その結果，分析段階のモデルや設計段階のモデルなど抽象度の異なるモデルを混在できるため，システム全体として分析レベルから設計レベルまでのモジュール(クラス)が再利用可能となった．

ただし異なる抽象度のモデルを混在させるためには，機能と通信を分離可能なチャネルを利用する必要があるが，チャネルの実装方法により処理時間にオーバーヘッドが存在することが分かった．従って，目的によりチャネルの使用方法を検討する必要があることを示した．

今後は更に設計を進め，実装段階まで一貫した開発における本設計手法のより具体的な効果を確認する予定である．

文　献

[1] Qiang Zhu, Akio Matsuda, Minoru Shoji, Shinya Kuwamura, Tsuneo Nakata, *"An Object-Oriented Design Process for System-on-Chip using UML"* ISSS2002

[2] SoC Design Methodology Bases on UML and C http://pr.fujitsu.com/en/news/2002/04/16-2.html

[3] UML for SoC Forum http://www.zipc.com/usocf/main.htm

[4] Akira Kawaguchi, *"Capturing and Analyzing Requirement"*, ASP-DAC 2003

[5] Rational Unified Process, Rational Software Corp., http://www.rational.co.jp/products/unified_process/index.html

[6] *"SystemC によるシステム設計"* 丸善 柿本勝・河原林政道・長谷川隆監訳

[7] Open SystemC Initiative http://www.systemc.org/

[8] *"UML モデリングのエッセンス第 2 版"* 翔泳社 羽生田栄一監訳

[9] *"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 渡辺博之・渡辺政彦・堀松和人・渡守武和記