東海大学大学 2010 年度 卒業論文

卒業研究論文

指導 清水 尚彦 教授

東海大学大学 情報理工学部
ソフトウェア開発工学科

7ADF1105 小沢　滋紀

平成23年1月31日

目 次

図 目 次

1　前期の内容

C 言語仕様書和訳 tex 化、C 言語チュートリアル和訳 tex 化 ET ロボコン 2010

2　後期の内容

MP3 エンコーダの解析

3　通年

組込みシステム用 UMLmarte の解析

4　C 言語仕様書、チュートリアル和訳 tex 化

C 言語仕様書和訳は、第 8 章を担当した。第 8 章の内容は構造体である。和訳を行い、専門用語¿等分からない所が多く苦労した。結果、和訳した文章は読みやすい文とはかけ離れた文章となってしまった。C 言語チュートリアルは第 1 章〜第 5 章を担当した。第 1 章〜第 5 章は C 言語の歴史と背景と概要といった内容であった。内容が容易であったため、仕様書よりは和訳を行いやすかった。それと同時に歴史や背景なども学ぶ事が出来た。この二つを tex 化して pdf ファイルを作成した。tex は pdf ファイルを作成する為の言語であり、始めて使用したので分からない所が多かったが細かい指定などが出来る為、きっちりとしたファイル作成には適していると感じた。

5　ET ロボコン 2010

ET ロボコン 2010 では、UML とインコース用ショートカットコースを担当した。UML では、コンセプトシートとフリーシートの二つを担当した。コンセプトシートとフリーシートは各チーム提出が決められていて、UML と走行タイムの二つ¿の結果から総合順位を出すということで非常に大事なものとなった。コンセプトシートは 1 枚提出で、内容は各チームの紹介と簡単な走行戦¿略の紹介となっている。私たちチームも簡単なチーム紹介と、数学モデルを用いた走行戦略を記述した。フリーシートは 5 枚提出で、ここで¿全体の走行戦略と UML の記述を行った。走行戦略は、モーターエンコーディング主体でというコンセプトで記述し数学モデル等から導き出される根拠に基づきコース走破を目指した。次に UML の記述は「marte」を用いて行った。ソフトとハードが明確に分けられ、見やすく分かりやすいものに¿仕上げようとしたがそういかなかった。原因として、「marte」のソフトとハードの記述部分にあたる「SRM」と「HRM」の内容の理解¿がすごく浅くとてもモデルに起こせるまでの知識レベルに達していなかった事が原因である。この状況から作成されたモデルは非常に中途半端なものとなってしまい、ソフトとハードが混同し見にくく分かりづらいものとなってしまった (図 e-1)。次に、インコース用ショートカットコースを担当した。この走行戦略は、インコースの難所を全て通らずに最短距離を走るというものである。速く走るために、常にモーター出力を限界に近い値を出し続ける為の工夫など試行錯誤しながら作成を行った。最終的に完成したのが大会当日になってしまったが走れるものは作成できた。しかし、大会当日の試走のときに、既存のモーターエンコーディング走行ではしっかりと走れない事が判明した。

そのため、私の作成したソースファイルはモーターエンコーディングが走行の半分を占めていたので、失格の可能性を考慮し走らせることはなかった。

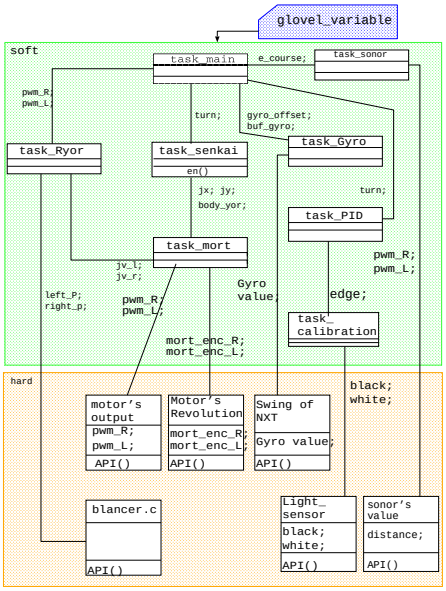

Fig. 1: class diagram

6　MP3 エンコーダの解析

MP3 エンコーダの解析は、ISO に沿って作られた dist10 というフリーツールを利用して行った。dist10 は C ソースファイルも見ることが出来、過去の諸先輩方も利用していた為採用した。最初は、ISO の仕様書を元に PCM データから MP3 データになるまでのデータフローと処理内容を追っていった。

6.1　処理の流れ

最初に元データとなる PCM データを入力し、そのデータが Hybrid-Filter-Bank(HFB) で処理される。HFB は、Sub-Filter-Band(SFB) と MDCT から成り立つ。SFB では、入力されたデータを 512sample に分けたのち時間成分に分割し 32 のサブバンドを生成する。次に MDCT で SFB で生成された 32 サブバンドを周波数成分に 1 サブバンドを 18 に分割する (3218=576)。すると 576sample のデー¿タが生成され、この 576sample のことを 1 グラニュールと呼ぶ。HFB での処理の次に、Iteration-Loop で処理が行われる。ここでは、主に量子化がメイン処理となる。1 グラニュールを量子化し、更にのちのハフマン符号表の最大値を越えていないか、判定や使用ビット数の上限を越えていないかといった判定を行う。Iteration-Loop は、Outer-Loop と Inner-Loop があり、Iteration-Loop を含めて 3 段の構成となってる。Iteration-Loop は 2 つのループを制御する為にあるようなもので、データのスペクトル値が 0 と判定し未使用ビット数の計算を行ったのちに全体の¿ループが終わる。スペクトル値が 0 以外を判定したときに、Outer-Loop が呼び出される仕組み¿となっている。Outer-Loop は、主にひずみに関する処理を行う。Outer-Loop は処理の一

番最初に Inner-Loop を呼び出す。つまり、Inner-Loop からのデータを元に歪み計算を行うことになる。Inner-Loop は、上記で示したように Outer-Loop の最初の処理で呼び出される。ここでのメイン処理は量子化である。最初に量子化を行い、その値に対して何度もループする仕組みとなっている。ループの条件は、量子化を行いその値が量子化表の範囲外になるまでまず量子化を行う。そして、範囲外になったらビット総和が使用可能ビット数より少なくなるまでループするといった条件である。つまり簡単に言えば、データ圧縮を出来る所まで行うということである。ここで得られた値を Outer-Loop に渡し、歪み計算を行い、スペクトル値が 0 を示すといった¿流れである。ここのループが終わると、次はハフマン符号化である。全作業で得られた値をハフマン表というものと照らし合わせ符号化することである。ここでは、3 つの組みわけがされ¿ており、0 が連続して続くところ、連続するデータ部分、データの出始めの部分といった 3 つである。0 が連続して続く所は高域成分であり、常に 0 が割り当てられる。連続したデータ部分は、メインビットになっている。ここは big-value と言われ、量子化された値からハフマン符号表と照らし合わせビットを割り当てる。次に、データの出始めの部分は 0、1 が連続して出力されるためその専用の 4 つ組みのハフマン符号表を用いてビットを割り当てる。その後、フレーム形成を行いビットストリームを生成する (図 m-1)。

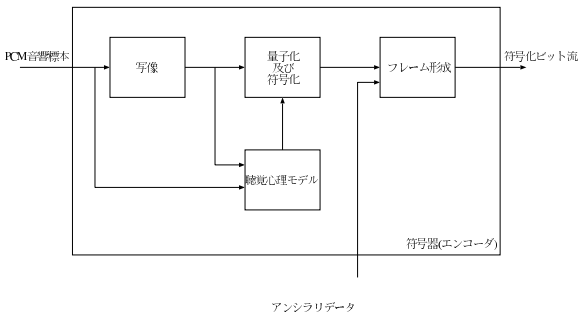

Fig. 2: 処理の流れ

6.2 Iteration loop

全サブセクションで記述した通り、Iteration-Loop がある。ここでの処理はエンコードする際の大半の時間をここで消費する。特に Inner-Loop の量子化で大幅な時間を割く。今回はそこに重点を当てて解析を行った。Iteration-Loop は、先ほど記述した通り 3 段構造となっている (図 m-2,m-3,m-4)。Iteration loop は、このループ全体を制御する所である。最初に使用ビットレートの設定の反復変数の再設定を行う。簡単に言えば、使用する変数の初期化である。その処理が終わると次にスペクトル値の判断を行う。スペクトル値が 0 であればループを抜けることになる。0 で無ければ圧縮出来るデータがあるということなので、outer loop を呼び出す。最後の処理に未使用ビット数の計算を行なってループから抜ける。

6.3 Outer loop

Outer loop では、一番最初に Inner loop を呼び出す。ここでは、Inner loop で行なわれた処理を元に処理を行う。処理内容は、主に歪みに関する処理を行う。Inner loop で詳しく説明するが量

子化されたデータはひずみが発生する。そのひずみはノイズとなるためノイズを除去するという作業になる。

6.4 Inner loop

Inner loop では、主にデータの量子化を行う。量子化は各 sample に対して行う。量子化した値は量子化表と呼ばれる表の最大値を越えていないかを判断する。越えていない場合は更に量子化を行う。量子化表の値を越えた場合は、それ以上圧縮出来ないという意味なので次のステップに移行する。次に量子化を行なった値を元にハフマン符号化で用いるビットの計算を行う。そこで値を 3 種類に分ける。1 つ目は 0 が連なるもの。2 つ目は、”1”,”0”,”-1”のいずれかになるもの。3 つ目はその他の 3 つになる。3 つ目のその他は bigvalue と呼ばれるものになりここは小さな値から大きな値まで全ての値が入る。ここのビット計算が終わると、最後にビット総和が使用可能ビット数よりも少ないか判断し、少なければループは終了し、大きければ更に量子化を行う仕組みになっている。

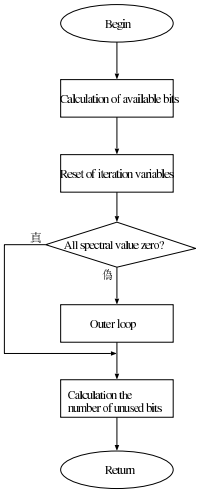

Fig. 3: Iteration-Loop

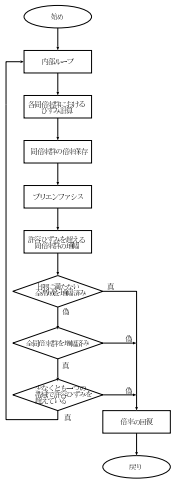

Fig. 4: Outer-loop

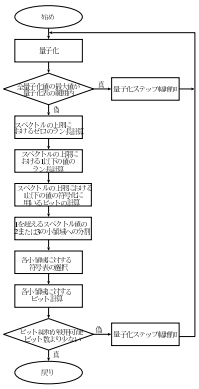

Fig. 5: Inner-loop

7 組込み UMLMARTE の解析

MARTE は、リアルタイム組込みシステム向けのドメイン特化モデリング言語で、UML の拡張プロファイルとして定義されている。OMG(Object-Management-Group) が公開した物で 2007 年に仕様書のベータ¿版がリリースされた (A UML Profile for MARTE, Beta1)。

7.1 組込み UMLMARTE の解析

次に MARTE の簡単な全体構造について説明する。MARTE はいくつかのサブプロファイルによって構成されていて、各サブプロファイルは”foundations”、”annexes”、”design model”、”analysis model”という 4 つのパッケージによってグループ分けされている (図 1)。

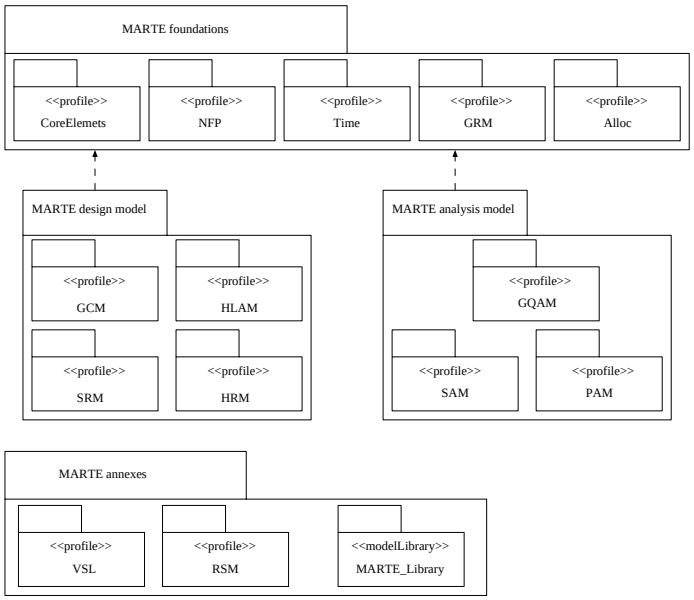

Fig. 6: Architecure of the MARTE Profile

7.2 foundation と annexes

次に、先ほど述べた”foundations”、”annexes”について触れてみる。foundations パッケージには、design model や analysis model で共通して用いられる各種概念要素が整理された形で定義されている。また、annexes パッケージには、共通に使用されるデータ型や列挙型の定義、その他のサブプロファイルを補助するためのオプション的なサブプロファイルなどが定義されている。また、これらのパッケージで特徴的なのは、非機能特性 (NFP:Non-Functional-Proerties) の値や時間に関わる制限値などをモデル上で厳密に表現して取り扱えるようになった。その NFP の表現は VSL(Value-Specification-Language) というテキスト形式の構文書式を利用する。リアルタイム組込みシステム開発において多くのソフトウェア開発の要件定義で特性値 (レスポンスタイム、データ転送レートなど) の正確な表現が重要でなり、これの表現が曖昧だとシミュレーションに影響を及ぼす可能性がある。上記の特性値の部分が NFP に相当すると言うことである。つまり、ここでの特性値をきっちりと決めることが非常に重要である (図 u-2)。次に”analysis”ついて説明する。システム解析には、様々な分野がある (信頼性、消費電力、メモリ消費など)。通常、各解析は独自の概念を扱っているがここでは共通して解析に用いられる。MARTE の GQAM(Generic-Quantitative-Analysis-Modeling) サブプロファイルには、共通した概念要素の定義と解析に究明な各種特性値などの情報を、注釈 (annotation) としてモデルに付記出来るようにする枠組みを与えている。今現在の仕様書 (beta1) では、GQAM を特殊化する形で PAM(Performance Analysis Modeling) と SAM(Schedulability Analysis Modeling) という 2 つのサブプロファイルが定義されている (図 u-3)。MARTE でのモデルベース解析では、解析したい状況を実行プラットフォームの構成と負荷シナリオとによってモデル化し、シナリオごとに具体的にリソースの割り当てを行ってモデル上で評価を行う。更に、GQAM は様々なモデルベース解析に対する共通した枠組みとなっているため、拡張性があり解析に適した仕様となっている。

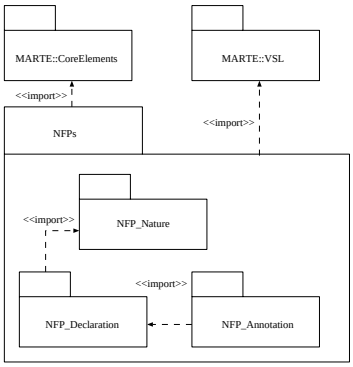

Fig. 7: structure and dependencies of the NFPs modeling package

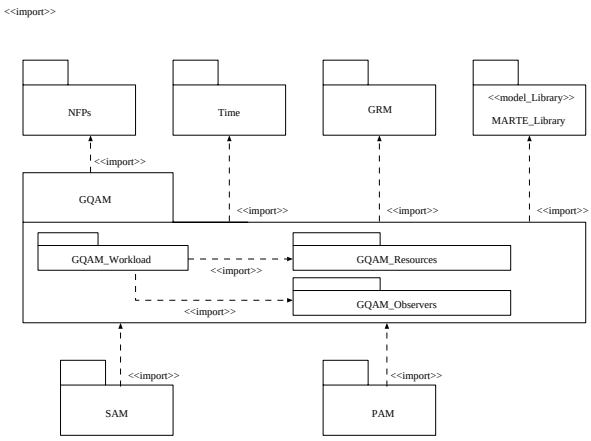

Fig. 8: Dependencies of GenericQuantitative-AnalysisModeling (GQAM) package

7.3 タイミングモデル

タイミングモデルについて説明する。組込みシステム開発には時間が要因となり起こるアクションが様々ある (割り込み処理、タイマーなど)。その中でタイミングに関する記述をきっちりとしなけれ¿ばならない。MARTE では、タイミングや時間に関するプロファイルが存在する。それが”Time Modeling(Time)”である。タイミングチャートなどは、UML1.4 がベースであり MARTE で使用するには UML2 に準じた変更が必要である。UML2 では、”時間”、”持続時間 ”、”測定”、”いくつかの時間制約の形”を表現する新しいメタクラスが追加された。更に必要であれば、適切なプロファイルが提供するより高度な時間モデルを扱うことも可能である (図 u-4)。

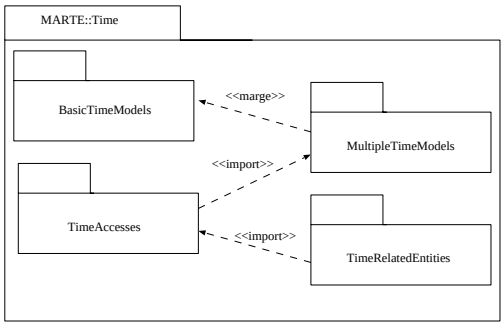

Fig. 9: structure of the Time domain model

8 GRM(GenericResouceModeling)

ここでは、GRM について説明する。ここは RTE(RealTimeEmbeded) アプリケーションの実行の為のプラットフォームの基本コンセプトの提供となる。様々な実行プラットフォームをモデリングするための機能を含む。プラットフォームの詳細を段階的に抽象化するボトムアッププロセスを組込み、設計上の関心に対応する実行プラットフォームに HRM(HardwareResouceModeling) と SRM(SoftwareResouceModeling) がある。次に HRM と SRM について説明する。

8.1 HRM(HardwareResouceModeling)

ここでは、HRM について説明する。HRM はハードウェアに関する記述法である。HRM は詳細な HW アーキテクチャ設計言語であり、詳細レベルは記述の正確性に依存する。HRM には様々なプロファイルが用意されている為設計者次第で事細かく記述することが出来る。先ほども言ったが多くのプロファイルが存在するためここでは、HRM 全体のプロファイルの構成図だけを示す。次に HRM の記述について説明する。(図 u-5,6)(図 u-7,8)

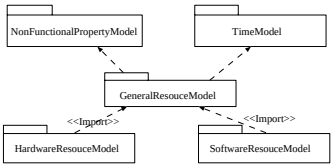

Fig. 10: HRM dependnceis part 1

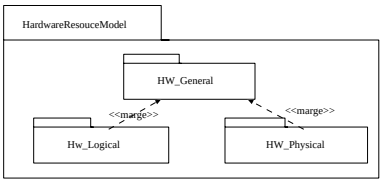

Fig. 11: HRM dependnceis part 2

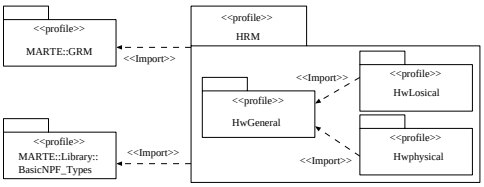

Fig. 12: HRM profile structure part 1

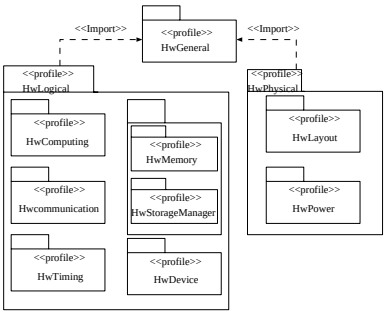

Fig. 13: HRM profile structure part 2

8.1.1 HRM の記述

ここでは、HRM の記述について説明する。HRM の記述は様々なプロファイルを用いて行なう。ここでは、簡単なもので行なってみようと思う。題材とする物は sn/x16bit を使用する。下記に全体構成を示す。(図 14) ここで注意してもらいたいことは、図 14 はモデルではなく図であるということである。これは、構成を示した図であり構成モデルではない。これでは、ハードウェアとソフトウェアが混ざってしまう。では、ハードウェアの記述について説明していきたいと思う。では、論理ビューから説明していく。

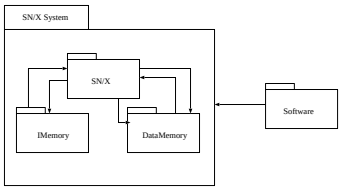

Fig. 14: SN/X structure

8.1.2 論理モデル

論理ビューではハードウェアリソースの機能分類に基づいた機能を記述する。下記の図を見てほしい (図 15) 。それぞれの機能毎の記述である。これを SN/X の構成に当てはめると次のようなモデルになる。これを SN/X の構成に当てはめると次のようなモデルになる (図 16,17)。次に物理モデルについて説明する。

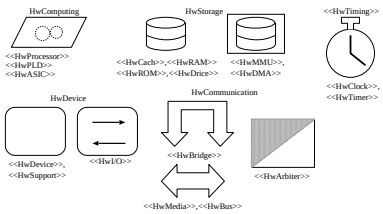

Fig. 15: Logical modeling

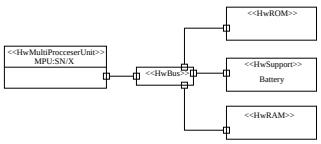

Fig. 16: SN/X Logical model part1

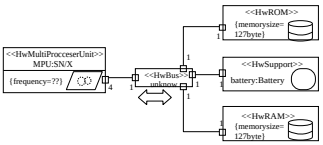

Fig. 17: SN/X Logical model part2

8.1.3 物理モデル

物理モデルでは、物理的なプロパティに関する記述を提供する。ハードウェアレイアウトは形 (チップ、カードなど) や寸法 (グリッドなど)、環境条件 (温度、湿度など) を記述し、ハードウェアパワーでは、電力消費や排熱量などを記述する。そのような記述を行う事で、そのシステムがどのようなものでありどのような環境で動作するものか明確に理解できる。下に、SN/X の物理モデルを示す。次は SRM について説明する。

8.2 SRM(SoftwareResouceModeling)

SRM はソフトウェアモデルの記述法である。SRM は設計段階でいうコーディングの段階で使用する。そもそも SRM は RT/E ソフトウェアの実行をサポートする API をモデリングするため

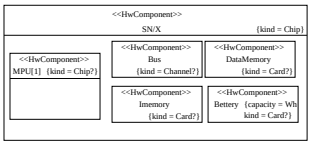

Fig. 18: SN/X Physical model

のプロファイルである (例えば RTOS や専用言語ライブラリ)。RT/E などはそれのドメイン専用の新しい API などは標準で用意されていない。つまり、様々な環境で開発された物の API は統一されていないということである。そこで、統一のとれていない API を統一的に表現する手段として SRM が存在する。今現在で UML を使った RTOS の API をモデリングすることは可能である。しかし、RT/E 独特のアーキテクトを今現在の UML でモデリングすることは出来ない。例えば、タスク処理であったり、割り込みやセマフォなどがそうである。しかし、SRM ではマルチタスクをモデル化することが可能であり、そのようなプロファイルは MARTE の標準サブプロファイルとして提供されている。では、実際に SRM を用いてモデリングを行なってみる。今回は ET ロボコン 2010 で配布された”sample.oil”ファイルを使用する。このファイルは、OSEK/JSP の製品であり、先で言った大会で使用した NXT に載せるファイルである。このファイルを用いる事でタスク管理等も行なえる為今回のモデリングに丁度良い。

8.2.1 SRM の記述

まずは oil ファイルの構成を示してみる (図-19)。ここから詳細にモデリングしていく。oil ファイルはマルチタスク設計の構成ファイルである。個々の CPU に対して OIL 記述が対応し、記述は OSEK システムオブジェクトはすべて OIL オブジェクトを使って記述する。今回はタスク設計をモデリングするために oil ファイルを使用したがその他の記述に対しても問題なく行なえる。また、このソースと C/C++のソースのモデルと組み合わせれば SRM が完成する。

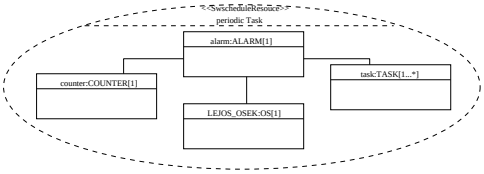

Fig. 19: oil file structure model

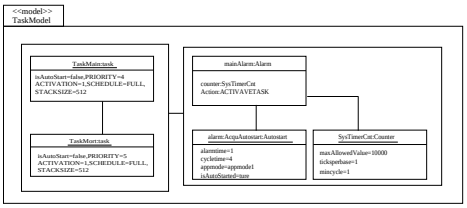

Fig. 20: task model

参考文献

[1] ET ロボコン 2010 公式サイト（オンライン），入手先
<http://www.etrobo.jp/2010/>
(参照 2010-06-09)

[2] UMLprofile for MARTE,Beta1（オンライン），入手先
<http://www.omg.org/docs/ptc/07-08-04.pdf/>
(参照 2010-06-09)

[3] JIS ディジタル記録媒体のための動画信号及び付随する音響信号の 1.5Mbit/s 符号化ー第三部，入手先