eUML～組込み向けオブジェクト指向開発プロセスとUMLによるモデリングガイドライン～のご紹介

株式会社オージス総研
オブジェクトテクノロジー・ソリューション事業部　チーフコンサルタント
渡辺　博之

1．はじめに

本稿では、現在、株式会社オージス総研とキャッツ株式会社を中心に策定を進めている「eUML」について、その目的と概要を、簡単にご紹介していきたいと思います。

eUMLとは、文字通り、embeddedすなわち組込み分野においてUMLを使ったオブジェクト指向開発を進めていく際に必要となる、開発プロセスとさまざまな作業ガイドラインについて定義したものです。なお、現時点では、あくまでも作業の指針としてのガイドラインに過ぎませんが、将来的には、先頭についているeは、embeddedだけでなく、モデル実行を意味するexecution、UMLの拡張を意味するenhanced、など他の多くの意味をあらわすような、つまり最終的にはモデルベースで組込みシステムを設計・実装するための定義にしていきたいと考えています。eUMLの策定作業は、まだまだ開始したばかりですので、今回紹介する内容も現段階での構想に近いものです。それでも、eUMLのおおよそのイメージは掴んでいただけると思いますので、以降順番にご紹介していきたいと思います。

2．eUMLの背景

この一年間で、組込みシステムの開発にオブジェクト指向を使おう、という動きが一挙に加速してきました。従来から適用が進んでいたFA機器やOA機器に加え、最近では次世代携帯電話やカーナビなどの分野においてオブジェクト指向開発への急激なシフトが始まっています。また、ソフトウェア開発の現場にUMLという共通のモデリング言語が定着してきたことで、組込み分野でも、ソースレベルではなく、まず図面、すなわちモデルをベースとした開発をしようじゃないか、といった議論も広がり始めています。われわれも、ここ数年で、随分とこのような作業のお手伝いをさせていただいてきましたが、残念ながら、現場を見ている限りでは、オブジェクト指向への移行の障壁は依然として高い、というのが実感です。そして、この要因として考えられるものをまとめると、大きく以下の3点に集約されるように思えます。

●オブジェクト指向開発の上流工程における自由度の高さ
ハードウェア操作や各種の制御処理が多い組込み分野では、何をクラスにすればよいのか、何をタスクにすればよいのか、といった明確な基準が存在せず、開発者の習熟度に頼らざるを得ない。
●開発プロセスの曖昧さ
RUP など、UML を使った新しい開発手順が定義されてはいるものの、汎用化されすぎており、組込みシステムの開発に直接使えるようなものになっていない。
●UML による表現の限界
現時点の UML では、タスクやハンドラ、タイマー、ROM/RAM などといった組込み特有の要素を表現する方法が定義されていない。また、UML で記述されたモデルとそれがどう実装されるかというルールが未定義のため、詳細な設計までモデルで記述することができない。

3．eUML の目的

われわれは、これらの問題を解決するためには、通常の UML や開発プロセスではなく、組込み分野に特化して抽象度を下げた、より具体的な開発プロセスと作業ガイドラインが必要ではないか、と考えました。そして、それを定義するものが、まさに今回紹介する eUML になります。eUML では、具体的に以下の 2 つの指針を定義しています。
●組込み分野に特化したオブジェクト指向開発プロセス
●組込み分野のモデリングに役立つさまざまなガイドラインの定義
これらの指針に沿って開発作業を進めていくことで、前述したような障壁に十分対応できるのではないかと考えています。

4．eUML の位置付け

最初に、eUML の内容を説明する前に、eUML 自体の位置付けについて簡単に触れておきたいと思います。以下の（図1）からも分かるように、eUML は特定の分野に依存しない組込み分野全体に適用できる開発ガイドラインを目指しています。そして、最終的には、各ドメインごとにより詳細なガイドラインが定義されることを想定しています。

5．eUML の内容と開発プロセス

前述したように、ビジネス系の分野では、開発プロセスとしてラショナル統一プロセス（以降、RUP と表記）が主流となりつつあります。しかし、組込みシステムを想定した場合、ハードウェア制御、複雑なビヘイビアや並行性、さまざまな開発環境にあわせたキーメカニズムなど、既存の RUP だけでは対応しきれない特徴が多く存在しています。eUML では、RUP をベースにしながら、それらの特徴にあわせた組込み分野に最適な開発プロセスを定義します。まず、eUML 開発プロセスの大きな特徴は、

●システムのビヘイビア分析
●タスク設計
●アーキテクチャ・メカニズム設計
を新たに主ワークフローとして定義している点です。(図2)
システムのビヘイビア分析は、従来のユースケース分析だけでは見つからない並行性やアルゴリズムなど、システム内部の複雑なビヘイビアを、状態図やアクティビティ図などを使って開発初期からきちんと分析しようという作業です。また、タスク設計は、組込みシステムに欠かせないタスクの設計とオブジェクトへのマッピング作業などを含み、アーキテクチャ・メカニズム設計は、オブジェクトやメモリの管理、並行性実現メカニズム、割り込み処理などのターゲット実装のための設計作業を含んだものです。また、それ以外にも、
●クラスの段階的な抽出
●オブジェクト単位でのビヘイビアを重視
●システムを階層的に管理するコンポーネントなど中粒度の概念の導入
なども組込みに特化した特徴的な部分です。

図1　eUMLの位置付け

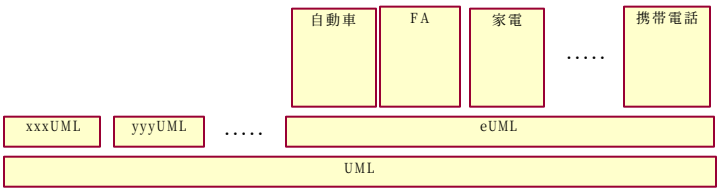
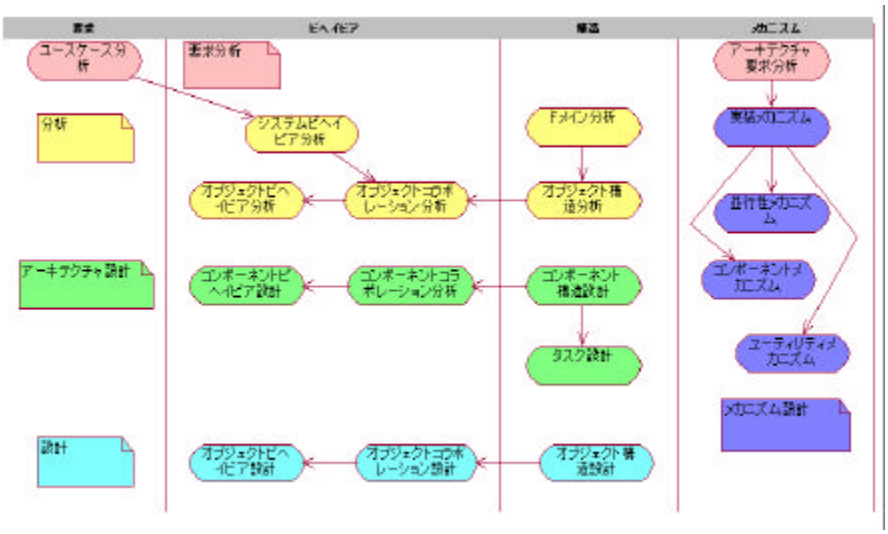


図2　eUMLの開発プロセス



６．モデリングのガイドライン

eUMLでは、組込み分野をモデリングした場合に、共通に出現すると思われるモデル要素のカテゴリをあらかじめ定義し、ステレオタイプで表現します。これらの要素を使いながらモデリングを進めることで、開発経験に左右されない属人性を排除した安定した品質のモデルを作成することができます。

eUMLでは、ユースケース、クラス、パッケージ、タスクなどのモデル要素について、これらの分類を行い、それらをeUMLの基本アーキテクチャと定義しています。

以下では、現在策定中の基本アーキテクチャのいくつかを簡単に紹介します。

＜ユースケース＞

まずアクターとしては、

●ユーザーを中心としたシステムのサービスを享受できる「主アクター」

●ユースケース実現のために相互作用を行うハードウェアやタイマーなどの「副アクター」

に分類します。システム全体のユースケースは主に主アクターを中心に導出することにより、組込み分野にありがちな詳細すぎるユースケース定義に陥ることを防ぎます。また、ユースケース自体も、大きく

●一般機能 《usual》

●保守機能 《maintenance》

●検査機能 《examination》

●デバッグ機能 《debug》

に分類し、さらにこれらから使用されるユースケースとして、

●共通機能 《utility》

●例外機能 《exception》

を定義しています。

これらの分類にしたがってユースケースを導出することで、もれ・抜けのないユースケースを作成することが可能となります。

＜パッケージ＞

当初から再利用を考慮したモデルを作成するために、eUML では開発当初から、開発対象をレイヤ化したドメインとして分離し、パッケージで表します。

現時点で定義中のドメインには、以下のような種類があります（図３参照）。

●インタフェース 《interface》

アクターとの相互作用に責任を持つ

●制御 《control》

システムの制御動作に責任を持つ

●サービス 《service》

システムが提供するサービス方法やその結果に責任をもつ

●対象 《target》

システムが取り扱う情報やモノの構造を表す

●機構 《mechanism》

装置の本質的な構造単位や制御単位を表したもの

●ハードウェア 《hardware》

実際の操作対象となるハードウェアを抽象化したもの

そして、これらのドメインの内部で定義されるクラスに対しても、局所化を図るためUMLで定義済みの以下のステレオタイプを使用します。

●バウンダリ 《boundary》
外部との相互作用に責任を持つ

●コントロール 《control》
制御動作に責任を持つ

●エンティティ 《entity》
システムが扱う実体の構造や機能に責任を持つ

また、各カテゴリに共通に存在するものとして、以下のような仕様クラスを新たに定義します。

●仕様 《spec》
クラスに共通の特性や振る舞いのルールなどをまとめたもの

たとえば、インタフェースドメイン内部は、さらに図4のようなクラスで構成されます。この場合、インタフェースがプロトコル制御などを行うコントロールクラス、インタフェース方法はプロトコルのパラメータ（タイムアウト時間や再送回数）などを定義する仕様クラス、インタフェースエンティティがLEDやブザーの間隔や回数などを表すエンティティクラスに該当します。

また、eUMLでは、各ドメインの特徴にあわせて、さらに上記カテゴリを詳細化します。たとえば、図5にあるように、制御ドメイン内部のコントロールクラスは、以下のような3つのクラスに分類されます。

●コーディネータ《coordinator》
必要なオブジェクトの呼び出しに責任を持つ

●状態依存コントローラ《state dependent control》
状態遷移に基づいた制御動作を行う

●調停コントローラ 《mediator》
複数のコントロールの調停を行う

7．eUMLに関する今後の予定

以上、eUMLの概要について簡単に紹介してきました。先に述べたように、eUMLの策定は開始したばかりで、まだまだ内容の詳細化と洗練が必要です。現在の予定としては、原案がまとまり次第、有識者による検討を行い、以降はパブリックドメインとして公開していく予定です。また、将来的には、eUMLで定義される開発プロセスやガイドラインを盛り込んだ開発ツールの提供も予定しています。乞うご期待ください。

図3　eUMLのパッケージ

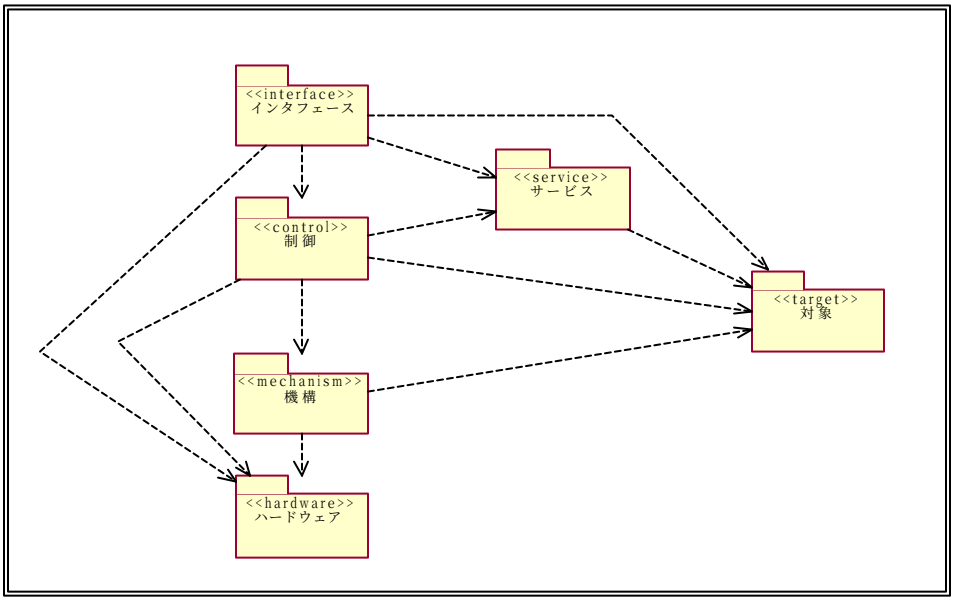


図4　インタフェースドメイン内のクラス

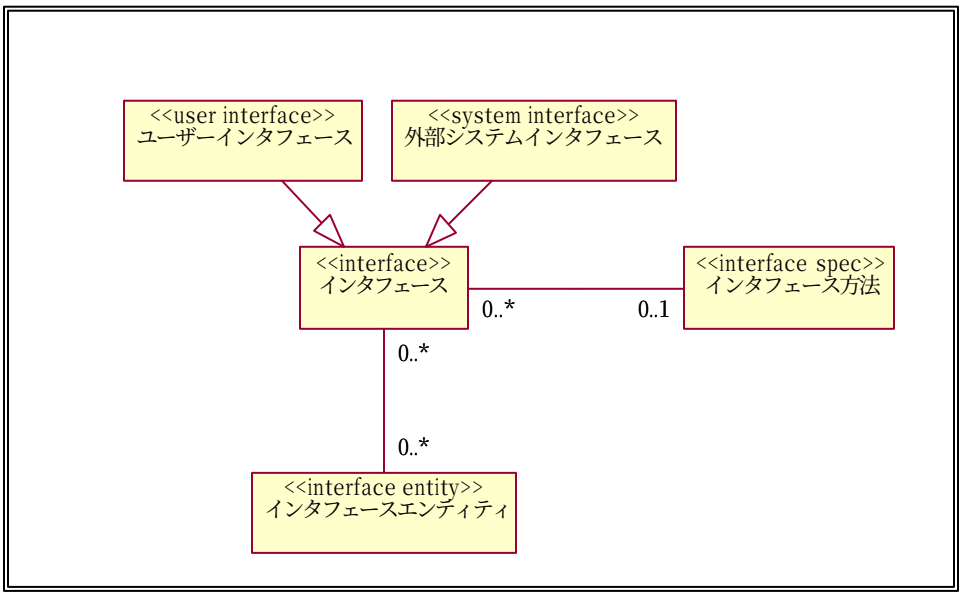
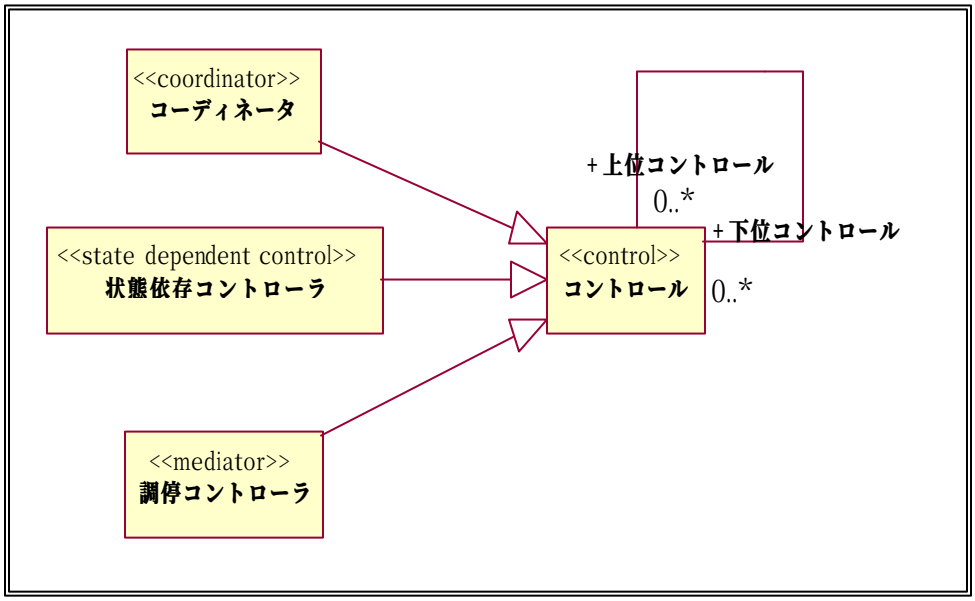


図5　制御ドメイン内のクラス



国内出展予定展示会のご案内

◎　第15回　MST2001 Embedded Technology Conference and Exhibition
開催日時：2001年11月20日(火)～2001年11月22日(木)　　場所：東京ビッグサイト

組込み系の世界最大のトレードShow　ESC出展予定！

◎ Embedded System Conference Chicago
開催日時：2001年7月9日(月)～7月12日(木)　場所：Navy Pier

◎ Embedded System Conference Boston
開催日時：2001年9月4日(火)～9月　7日(金)　場所：Hynes Convention Center