

2002年度
卒業論文

ホームネット・シミュレーションに向けて
－ ECHONET EMSの場合－

神戸大学工学部情報知能工学科
玉川世宗

指導教官 林晋

2003年2月21日

ホームネット・シミュレーションに向けて － ECHONET EMS の場合－

玉川世宗

要旨

今日の組み込みシステムの分野では、ソフトウェアの大規模化が顕著になってきている。これに対応すべく、林研究室の SMART プロジェクトは始まった。ここでは、UML を用いたオブジェクト指向的設計を取り入れた開発支援系の作成により、問題の解決に当たっている。昨年開発された SMART0.1 は単独の組み込みシステムを対象とし、設計したモデルを動作させ、検証する能力を有していた。現在、これを分散システム用に拡張しており、主なターゲットはホームネットワークである。

SMART プロジェクトにおける本研究の目標は、以下の 3 段階になっている。

1. SMART がホームネットワークのモデリングとシミュレーションに対応するために必要な新機能の提案
2. ホームネットワークのソフトウェアを作成し、新機能をプログラムで実現するための手法を探る
3. 並列動作が可能になった SMART への新機能の搭載

現在までの研究成果は、SMART の新機能のいくつかの提案と、新機能を実際のプログラムで実現させるための搭載元となる ECHONET エネルギー管理システム (EMS) のシミュレータの実装である。この EMS シミュレータは、EMS 制御の各方式をシミュレートするという基本要求を実現した段階であり、研究の現状としては、上の目標の 2 段階目の基礎を固めたあたりに位置している。本論文では、EMS シミュレータについては概要を、提案した新機能については、主要な成果である「モデル階層化機能の導入」の詳細な説明を行っていく。

ECHONET には通信レイヤが定義されており、ある階層の状態変化を対象としたモデリングを行う際、その中の遷移のアクションが、下位の層の状態変化も引き起こす場合がある。このような遷移を、階層構造を意識せず、1 つの Statechart 図の中に記述していくと、下位層の状態変化も全て組み込まれることになり、そのモデリングは巨大な量になってしまう。モデル階層化機能では、システムの階層構造に対応してモデルの階層を定義できるように Statechart 図のアクションに階層を持たせることで、この問題の解決を図った。

目次

第1章	序論	4
第2章	背景	6
2.1	UML	6
2.1.1	Statechart 図	6
2.2	ECHONET (Energy Conservation and Homecare Network)	7
2.2.1	ECHONET の通信レイヤ	7
2.3	EMS (Energy Management Service / Energy Management System)	8
第3章	EMS シミュレータ	10
3.1	ホームマップ	10
3.1.1	各家電機器の能力	10
3.1.2	シミュレーションの対象となる家庭と家電機器	11
3.1.3	電流値の扱いについて	12
3.2	EMS コントロールパネル	12
第4章	ホームネットワークモデリングのための SMART0.1 の機能拡充について	15
4.1	時間概念の導入	15
4.2	クラス概念の導入	16
4.3	モデル階層化機能の導入	16
4.3.1	問題提起	16
4.3.2	モデル階層化機能の提案	19
第5章	結論および今後の展開	29
5.1	結論	29
5.2	今後の展開	29
5.2.1	ECHONET 通信レイヤに対応した EMS シミュレータの実装	29
5.2.2	SMART のモデル階層化機能の搭載	30
5.2.3	並列動作に対応した SMART 上での EMS シミュレータの構築	30
	謝辞	31

参考文献	32
付録	33

第1章 序論

林研究室で開発された SMART(Statechart Modeling and Action Requesting Toolkits) は、UML の Statechart 図を用いた組み込みシステム開発支援のための設計ツールであり、設計したモデルを動作させ、検証する能力を持つ。当初、SMART は単独の組み込みシステムを対象としたツールを想定していたが、現在、これを分散システム用に拡張している。(以降、従来の SMART を SMART0.1 と呼び、現在のバージョンを SMART0.2 と呼ぶことにする。) SMART の新バージョンの主なターゲットは、最近注目されてきているホームネットワークである。本研究は、その SMART がホームネットワークのモデリングとシミュレーションに対応するために必要な機能の提案を実際のホームネットワークに即して行うことを目的とした。その主な成果は、モデル階層化機能の提案であるが、同時に、ホームネットワークの代表的規格である ECHONET に基づく、エネルギー管理システム (EMS) シミュレータを Java 言語を用いて作成し、提案した機能を Version0.2 以後の SMART のプロトタイプとして実現することを目指している。

本研究は同研究室の花山健二、森健司との共同研究の一環であり、花山は SMART の ActionSemantics エンジンの並列化、森は SMART の分散化を行っている。本研究によって提案された新機能は、この 2 つの研究の成果を待って、SMART に搭載されることになる。SMART0.1 の ActionSemantics エンジンは、逐次実行型であったが、花山の研究が完成すると SMART が種々の並列動作を行えるようになる。また、森の研究は、複数機器のモデリングと、各アプライアンスの物理インターフェイスの迅速なシミュレーションを目指している。花山、森の研究成果により、SMART0.1 が分散環境に対応できるようになり、ホームネットワークのモデリングとシミュレーションを行うための基盤が作られる。本研究はその基盤を踏まえて、まず、ホームネットワークにより特化した SMART ツールの機能を提案し、次に、Java 言語を用いて作成する ECHONET EMS シミュレータに提案した機能を反映させることで、それらを SMART に搭載する際のプログラミング手法を探っていく。

現在までの研究成果は以下の通りである。

- ECHONET EMS シミュレータの実装 (第 3 章で説明)
- 拡充すべき機能の提案。主には、モデル階層化機能の提案。(第 4 章で説明)

本論文の構成は、第 2 章が、研究の背景となる技術の概説、第 3 章、第 4 章が、上に挙げた研究成果の説明となっている。これからの研究の目標としては、まず、

ECHONET 通信レイヤを EMS シミュレータへ反映させる。その後、3 人の研究成果を統合させて、より一層、ホームネットワークのモデリングとシミュレーションに対応した SMART ツールを完成させる。そして、SMART のようなツールを簡易に作成するためのライブラリ群を提供する SMART パッケージを構築することが、SMART プロジェクトの最終目標である。

第2章 背景

本章では、本研究の背景となる UML、ECHONET 規格、EMS の概要を説明する。

2.1 UML

UML とは、Unified Modeling Language の略で、オブジェクト指向による設計をグラフィカルに表現する言語である。この論文で扱う Statechart 図の概要を以下に説明する。

2.1.1 Statechart 図

Statechart 図は、特定のオブジェクトの、その生存期間を通じての状態の変化やイベントへの応答を表す。

状態 (State)

状態はオブジェクトまたは相互作用の生存期間において、何らかの条件を満たす、何らかの動作を行う、もしくは何らかのイベントを持つ、などの状況を表現するものである。通常は角の丸い四角で表現される。状態は、内部に子状態を持つことが可能である。

遷移 (Transition)

遷移は状態間の関係を表現するものであり、状態間を結ぶ矢印で表される。遷移は、イベント名、ガード条件、アクションを持つ。

イベント名 この遷移を引き起こすイベントの名前。省略された場合は、その遷移は自動発火を起こす。

ガード条件 遷移が行われるためには評価結果が真でなければならない真偽値式。

アクション 遷移が引き起こされた結果として実行される動作。

2.2 ECHONET(Energy Conservation and Homecare Network)

ECHONETとは、日本の大手電機メーカーが中心となって設立したエコネットコンソーシアムが提唱している、家庭内の電灯線や無線を利用したネットワークの規格である。家庭内のさまざまなメーカーの設備機器や、コントローラが相互接続されて、これらが有機的な連携運転をすることで、省エネルギー、高齢者、在宅介護などに対応したホームシステムを実現することを目的としている。

ECHONETは通信回線と繋がったゲートウェイを介して外部から接続できるようになっており、ECHONET内にあるコントローラから直接機器を制御するほかに、遠隔操作による制御も可能である。このゲートウェイを利用することによって、利用者自身による遠隔操作のほか、ガス漏れ通知などのリモート保守も行なえる。

2.2.1 ECHONETの通信レイヤ

ECHONET機器の通信レイヤは、Fig2.1のような階層に分けられる。以下に、Fig2.1中の用語の説明を列挙する。

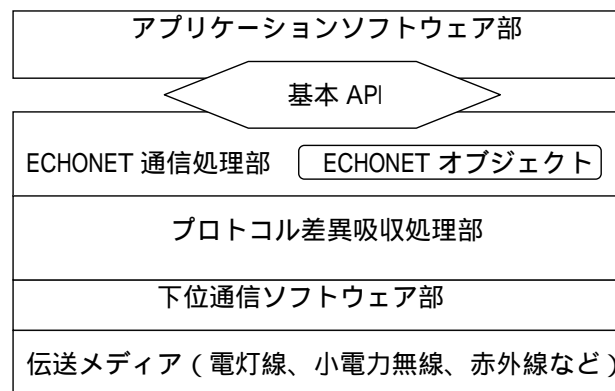


Fig 2.1: ECHONET communication layer

アプリケーションソフトウェア部

コントローラなどにおいて、システムに接続される機器を遠隔制御するソフトウェアや、エアコンや冷蔵庫などの個別の機器においてその機器本体の機能を実現するソフトウェア。

基本 API

アプリケーションソフトウェアと ECHONET 通信処理部との間のインターフェイス。ECHONET の基本的な機能を利用するためのインターフェイスである。主に、ECHONET 通信の運用（開始、停止など）、ECHONET 通信における送信や受信の機能に対して処理要求を出す。

ECHONET 通信処理部

アプリケーションソフトウェアが機器を遠隔制御や、モニタリングする際の処理を簡単にするための通信プロトコル処理を司る。そのための情報の保持、また自機器や他機器の状態などの様々な情報の管理を行う処理ブロック。

ECHONET オブジェクト

通信処理部が保持する保持する情報のうちネットワークに対して公開する情報やアクセス手順をモデル化したもの。各々の機器が持つ情報や制御対象がプロパティとして、またこれに対する操作方法（設定、参照）がサービスとして規定されている。

プロトコル差異吸収処理部

電灯線や小電力無線など複数のプロトコルの差異を吸収し、単一のネットワークとして見せることを目的としている。アドレス変換、通信種別変換、電文の分割・組み立てを行う処理ブロック。

下位通信ソフトウェア部

伝送メディア毎に特有の通信プロトコル処理を行うソフトウェア。

伝送メディア

通信を行うための物理媒体。ECHONET では、現在、電灯線、小電力無線、赤外線、ツイストペア線を対象としている。

2.3 EMS(Energy Management Service / Energy Management System)

EMS とは、エアコンなどの家電機器の最適運転や、照明のオン・オフ、さらにはエネルギーの使用状況を表示するなど、家庭におけるエネルギー消費のマネジ

メント（省エネ行動）を支援するシステムである。

本卒業研究は、SMARTがホームネットワークのモデリングとシミュレーションに対応するために必要な新機能を提案した後、ホームネットワークのソフトウェアを作成し、新機能をプログラムで実現するための手法を探ることを目標とする中期的研究の最初の段階である。そこで、具体的なソフトウェアとしてECHONET仕様中に紹介されているEMSのシミュレータを作成した。以下に、ECHONET仕様にあるEMSについて説明を行う。

ECHONET EMSは、制御対象となる機器の総消費電流値が設定値を超えないように、各機器の消費電流値をコントロールするが、その制御方式には次の3つがある。（各制御方式の図は付録を参照。）

(1) フィードバック方式ピークカット EMS

総消費電流量が設定値を超えた場合、コントローラが定められた規則に従って動作中の機器の能力を下げしていく。Fig8.1を参照。

(2) フィードフォワード方式ピークカット EMS

機器は自分の能力を変更しようとするとき、その旨をコントローラに申告する。コントローラは一定時間ごとに更新される総消費電流量と設定値を比較し、機器に消費可能な電流量の割り当てを行う。Fig8.2を参照。

(3) ハイブリッド方式ピークカット EMS

機器は自分の能力を変更しようとするとき、その旨をコントローラに申告する。コントローラは一定時間ごとに総消費電流量と設定値を比較し、機器に消費可能な電流量の割り当てを行う。その結果、もしも予測に反して総消費電流量が設定値を超えた場合、コントローラがある規則に従って動作中の機器の能力を下げしていく。（(1)方式と(2)方式のハイブリッド）Fig8.3を参照。

第3章 EMSシミュレータ

本章では、ECHONET仕様に即しながら、実装を行ったEMSシミュレータの概要を説明する。

EMSシミュレータは大きく分けて以下の2つの部分からなり、それぞれ3.1、3.2で説明を行う。

1. ホームマップ: 各家電の電源ON・OFFや、EMSの制御対象となる能力を設定する。
2. EMSコントロールパネル: EMSコントローラの電源のON・OFFや、制御の開始と停止などコントローラの操作を行う。

3.1 ホームマップ

シミュレート対象となる家庭の絵に、いくつかの家電が配置されており、そのアイコンをクリックすることで、各家電の電源ON・OFFや、EMSの制御対象となる能力を設定する。EMS制御が開始された後は、アイコンの絵が制御内容に対応し、変化する。また、アイコンの横には、現在の制御対象である能力値が表示される。一例としてFig3.1に、エアコンのアイコンの4種類の状態を表す。

	電源ON	電源OFF	能力低減	能力復帰
アイコン				

Fig 3.1: Icon of the Air Conditioner

3.1.1 各家電機器の能力

このホームパネルにおける各家電機器の能力変更は、EMS制御開始後、EMSコントローラの監視下に置かれる。Table3.1は、各家電のEMSの制御対象となる能力である。

照明	照度、電源オン・オフ
ホットカーペット	設定温度、電源オン・オフ
電気ストーブ	ヒーターパワー、電源オン・オフ
エアコン	設定温度、電源オン・オフ
食器洗浄器	水温、一時停止
衣類乾燥機	ヒーターパワー、一時停止

Table 3.1: Ability of the appliances



Fig 3.2: Homemap

3.1.2 シミュレーションの対象となる家庭と家電機器

Fig3.3は、シミュレーション対象の家庭の間取り図である。この絵は、フリーソフト「せっけい倶楽部」を用いて作成した。本シミュレータでは、1つの視点しか提供していないが、SMARTでEMSシミュレータを作成する場合は、SMART0.2で採用されたUser Interface ModelerのView Pointの概念に基づき、ユーザの操作に都合のよい視点から見た複数の絵を用意することになる。例えば、家電機器を上や横から見た図、また、多数の家電機器が存在する部屋を見る時のように、複数の家電機器をまとめて見る視点というのも考えられる。Table3.2は、各部屋と、配置されている家電機器の対応表である。

部屋	家電機器
リビング	リビング照明
	ホットカーペット
	電気ストーブ
	エアコン
キッチン	食器洗浄器
	キッチン照明
洗面所	衣類乾燥機
	洗面所照明

Table 3.2: Appliances in home



Fig 3.3: Layout of the house

3.1.3 電流値の扱いについて

このシミュレータでは、電流値を mA の単位で示す。また、ECHONET 仕様書に基づき、計測対象となる電流が交流の場合は、その実効値を扱うものとする。

3.2 EMS コントロールパネル

EMS コントロールパネルは、EMS コントローラの電源の ON・OFF、制御方式の選択、EMS コントローラが制御を行うのに必要なパラメータ設定、制御の開始と停止、制御状況のグラフィカルな表示を行うパネルである。

EMS コントロールパネルの起動と制御方式の選択

EMS シミュレータ上部の EMS ボタンを押下すると、EMS コントロールパネルが立ち上がる。このパネルの上部の電源入/切ボタンを押下すると、EMS コントローラが起動し、制御方式の選択リストが立ち上がるので、ここで、1つの制御方式を選択する。選択すると、EMS コントロールパネルに確認のメッセージが表示される。

パラメータ設定

制御方式を選択すると、パラメータ設定ボタンが押せるようになるので、このボタンを押下すると、パラメータテーブルが立ち上がる。ここに、EMS 制御を実行するのに必要なパラメータを入力する。但し、必要なパラメータは各制御方式により異なる。(Table3.3 を参照。)

制御名	EMS登録	優先順位	能力名	能力低減値	制御開始電流値(A)	制御終了電流値(A)	電流表示時間遅延(ms)
リビング照明	<input checked="" type="checkbox"/>	1	照度 (0-100%)	0.0			
ホットカーペット	<input checked="" type="checkbox"/>	3	設定温度	2.0			
電気ストーブ	<input checked="" type="checkbox"/>	2	ヒータパワー(W)	400			
エアコン	<input checked="" type="checkbox"/>	4	設定温度	2.0			
制御対象機器	<input type="checkbox"/>	-	-	-	10.0	10.0	1000

Fig 3.4: Parameter table

制御方式	パラメータ
フィードバック方式ピークカット EMS	制御開始電流値
	制御終了電流値
ハイブリッド方式ピークカット EMS	制御対象機器の登録
	制御対象機器の優先順位
	制御対象機器の能力低減値
フィードフォワード方式ピークカット EMS	制御開始電流値
	制御対象機器の登録

Table 3.3: Parameter required for EMS execution

EMS 制御開始

制御に必要なパラメータを入力し終えた後、パラメータテーブルの適用ボタンを押下すると、設定内容が EMS コントローラに反映され、確認のメッセージが

EMS コントロールパネルに表示される。また、制御開始ボタンが押せるようになる。このボタンを押下すると、制御が開始され、以後は、制御停止ボタンが押されるまで、一定時間ごとに総消費電流値と、各 EMS 制御対象各機器の電流値が表示される。また、一定時間ごと（現在は 7 秒ごとの設定）に“総消費電流値の入力”（Fig8.1 参照）が行われると、制御状況が EMS コントロールパネルにメッセージとして表示される。さらに、EMS 制御を開始すると、電流値表示イコライザーが立ち上がり、総消費電流値と、各制御対象機器の一定時間ごとの電流値をグラフィカルに表示する。尚、電流値表示の時間間隔は、パラメータテーブルにてユーザが任意に設定できる。

第4章 ホームネットワークモデリングのためのSMART0.1の機能拡充について

本章では、ホームネットワークのモデリングをSMARTが実現するためにSMART0.1が機能拡充すべきいくつかの点を、EMSシステムシミュレータのケース・スタディを踏まえて提案する。

4.1 時間概念の導入

ECHONETの規格には時間が秒単位で定義されている。ECHONETでは、アプライアンス同士がケーブルでつながるのではなく、ネット上で電文を飛ばすことで通信しあうため、電文の送受信の際、常にタイムアウトの問題が存在する。また、本研究で扱ったEMSの電流値制御も一定時間間隔で行われており、時間ドリヴンである。このように、SMARTパッケージでホームネットワークのモデリングを行う際に、時間の概念は必須であるが、SMART0.1はイベントドリヴンにしか対応していないので、時間概念の導入が必要となる。

さらに、現実のホームネットワークにおいては、それぞれのアプライアンスがクロックを持ち、そこには必ず、各クロックのずれが存在する。白物家電を主な対象機器としているECHONETにおいては、そのずれは大きな問題とはならないが、ホームネットワークを対象とするアプリケーションソフトウェアで、時間に厳しい制御を要求するものが今後出てくることも考えられる。このようなアプリケーションソフトウェアが正しく動作するかどうかを、SMARTを用いて検証する場合、意図的に各アプライアンスにずれたクロックを持たせることが必要となる。よって、導入すべき時間概念は、一つのクロックを全ての機器が参照するグローバル・クロックの概念ではなく、クロックを必要とするオブジェクトが、それぞれ固有のクロックを持つ、ローカル・クロックの概念に基づくものでなくてはならない。

今回作成したEMSシミュレータでは、EMSコントローラと、電流値表示を引き受けるクラスの2つにJava Swingのタイマオブジェクトを持たせ、時間の経過を監視している。このように、SMART上で各オブジェクトにローカル・クロックを持たせる場合、現バージョンのシミュレータのようにJavaで実装する場合は、Java言語のタイマもしくはスレッドを容易に生成できるAPIが実現される必要がある。

これは、Java ベースのシミュレータの場合であるが、SMART は UML 準拠を目指しているので、UML 規格に基づく時間概念を使用する必要がある。しかし、UML1.4 の規格では時間の概念は定義されていない。Bruce Powel Douglass は、UML において、タイマーやクロックといった時間発生源についてモデリングする手段を提供すべきという提案を OMG (Object Management Group オブジェクト指向関連技術の標準化団体。) に提出した (“リアルタイム UML 第 2 版” 参照)。しかし、現在標準化が進んでいる UML2.0 に、その提案が反映される様子はない。よって、Statechart 図を用いたモデリングを行う SMART では、ローカル・クロックをいかに導入するかについては、独自の方法論を確定する必要があるだろう。

4.2 クラス概念の導入

ECHONET では、通信を介しての制御や状態の確認を容易とすることを目的とし、各機器の持つ「機器としての動作機能」を「機器オブジェクト」として詳細の規定を行っている。「エアコンオブジェクト」や「冷蔵庫オブジェクト」のような各オブジェクトの仕様は、クラスとして別途個々にプロパティを規定しており、また、すべての機器オブジェクトに共通的に規定されるプロパティ構成を、「機器オブジェクトスーパークラス」として規定している。これらを SMART でモデリングする際には、オブジェクト指向言語にみられるクラス概念やクラス間の継承関係の概念が不可欠であるが、SMART0.1 には、クラス概念がなく、設計の単位がオブジェクトにしか対応していない。よって、これを導入する必要がある。

4.3 モデル階層化機能の導入

ここでは、本研究の主な成果であるモデル階層化機能について説明を行う。

4.3.1 問題提起

ECHONET では、2 章でも触れたように通信レイヤが定義されており、アプリケーションソフトウェア部、ECHOENT 通信処理部、プロトコル差異吸収処理部、下位通信ソフトウェア部、伝送メディアの 5 階層に分けられる。このような階層構造を持ったシステムのモデリングに SMART が対応するためには、モデルの階層を定義できることが必要である。今回作成した ECHONET EMS シミュレータでは、ECHONET 通信レイヤに完全に対応した実装は実現できていない。そのため、ECHONET 規格にある様々な伝送メディアごとの通信の様子などは、シミュレートできない。また、共同研究者の森が今回作成した SMART0.2 にも、モデル階層化機能がないため、通信レイヤの最下位層までを意識したモデリングを行う

と、巨大な Statechart 図になってしまう。

この問題の具体的な例を挙げてみる。Fig4.1は、フィードフォワード方式ピークカット EMS コントローラのアプリケーションソフトウェア部における状態変化を示した Statechart 図である。Fig4.1の“時間判定結果 [一定時間経過]/It 更新”という遷移は、It 更新というアクションを引き起こす。It (総消費電流値) を更新するためには、さらに下位の ECHONET 通信処理部に存在する ECHONET オブジェクトの電流値プロパティの値を、他機器である電流センサと通信して得た最新の値に書き換えなくてはならない。これは、アプリケーションソフトウェア部のみにとどまらず、最下位層の伝送メディアまでを含んだ ECHONET の通信レイヤ全体に影響が及ぶ実行内容のアクションといえる。このアクションの実行時における ECHONET 通信処理部の状態変化の Statechart 図を Fig4.2 に示す。また、ここでは Statechart 図におけるアクションの中で、次の 2 つの場合に該当するものを下位層に実行の影響が及ぶアクションとして定義する。

- 自機器の下位層に存在する情報を参照、制御するための要求を出し、その応答として、下位層からの通知を受け取る場合。
- 他機器に存在する情報を参照、制御するための要求が書かれた電文を送信し、その応答としての通知がかかれた電文を受信する場合。

このように、ECHONET の通信レイヤのような階層構造を持ったシステムの Statechart 図を用いたモデリングでは、より下位の階層へと実行の影響が及ぶアクションが存在する。このようなアクションを含む遷移を、階層構造を意識せず、厳密に Statechart 図で記述していくと、下位層の状態変化も全て一つの Statechart 図に組み込まれることになり、そのモデリングは膨大な量になってしまう。そして、そのような Statechart 図は複雑であり、理解が困難である。この問題を解決するためには、システムの階層に対応するモデルの階層を定義できることが必要である。

UML の Statechart 図はミーリー型状態図、ムーア型状態図における状態の入れ子を可能にしたものであり、状態の階層構造については明確に表現できるが、現在の UML 仕様では状態間の遷移によって引き起こされるアクションの階層については、それをいかに定義し表現するか規定されていない。しかし、現実には通信レイヤのような階層構造を持つシステムでは、階層全体に影響が及ぶ実行内容のアクションを、階層ごとにモデリングしたいことがある。このような場合に、Statechart 図の遷移におけるアクションの階層をいかに定義し表現するか、その具体的な手法を以下に提案する。

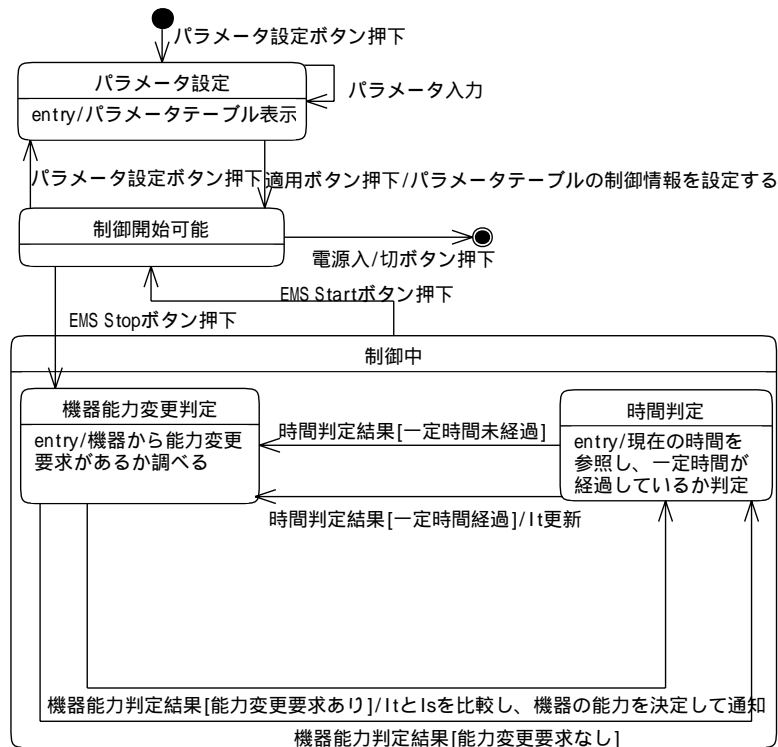


Fig 4.1: Statechart diagram of the layer application software($L_{a(a)}$) in case of feedforward EMS

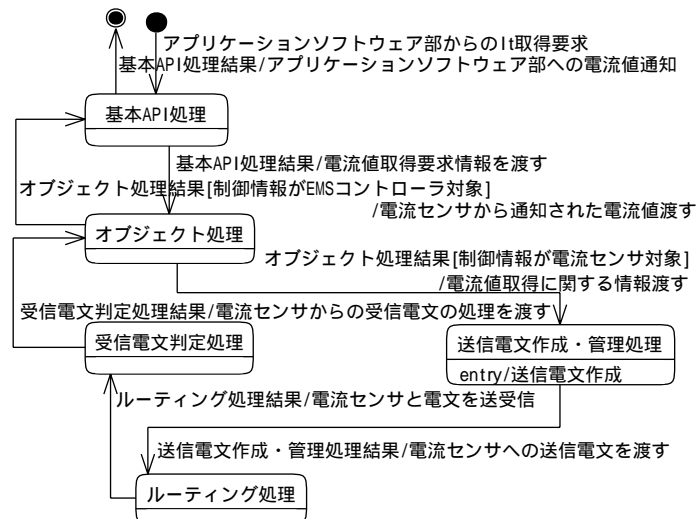


Fig 4.2: Statechart diagram of the layer ECHONET communication transaction($L_{c(c)}$) in case of feedforward EMS

4.3.2 モデル階層化機能の提案

現実の Statechart 図によるモデリングでは、状態名、遷移のイベント名、ガード条件、アクションはすべて、意味を持った文字列で表現されるが、表記の簡便性を考え、独自の表記法を以下に定義する。

階層構造に対応したモデルの表記法の定義

1. 階層を L で表す。現在のモデリング対象である階層の名前を L の添え字で、また、そのモデル中に含まれる最下位層の名前を添え字の $()$ の中に記す。
例) モデリング対象階層が x 、そのモデル中に含まれる最下位層も x 。

$$L_{x(x)}$$

2. 階層を構成する要素は、状態の集合と遷移の集合であり、それぞれ S と T で表す。それぞれ属する階層名を添え字で記す。
例) モデリング対象階層 x の構成要素は、状態の集合 S_x と、遷移の集合 T_x である。但し、モデル中に含まれる最下位層は y 。

$$L_{x(y)} = (S_x, T_x)$$

3. 階層 x に属する各状態は、 S_x に添え字で番号を振り、識別する。但し、初期状態と、最終状態は、番号の前にそれぞれ i と f をつけて記す。初期状態と終了状態が、階層に 1 つずつしかない場合は、番号は省略してもよい。初期状態と、最終状態以外の状態は、子状態を持ち得る。その場合は、子状態を $\{$ の中に列挙し、ない場合は、空にする。子状態として、1 つ下位の階層における状態変化を持つ場合は、その階層名を記す。また、状態が entry action, exit action, do-activity の 3 種類のアクションを持つ場合は、 $\{$ の中にそれぞれ、 $entry/$, $exit/$, $do/$ を記し、 $/$ の後に、アクションを記述する。
例 a) 状態の集合 S_x の構成要素は、 S_{xi}, S_{xf}, S_{x1} である。但し、 S_{x1} は entry action として、 A_{x1} を持つ。

$$S_x = (S_{xi}, S_{xf}, S_{x1}\{entry/A_{x1}\})$$

- 例 b) 状態の集合 S_x の構成要素は、 $S_{xi}, S_{xf}, S_{x1}, S_{x2}$ である。 S_{x1} は子状態として、 S_{x3} を持ち、 S_{x2} は子状態として、階層 y における状態変化を持つ。

$$S_x = (S_{xi}, S_{xf}, S_{x1}\{S_{x3}\}, S_{x2}\{L_y\})$$

4. 階層 x に属する各遷移は、 T_x に添え字で番号を振り、識別する。これが遷移のイベント名にあたる。但し、初期状態から出る遷移、最終状態へ向かう遷移は、番号の前にそれぞれ i と f をつけて記す。初期状態と終了状態が、階層

に1つずつしかない場合は、番号は省略してもよい。ガード条件と、アクションがある場合は、[]の中にガード条件を、/の後にアクションを書く。ガード条件は G 、アクションは、 A で表す。それぞれイベント名と同じ添え字をつける。また、|の後に、遷移の出発元となる状態を \rightarrow の左側に、遷移の到着先となる状態を \rightarrow の右側に記す。

例) 遷移の集合 T_x の構成要素は、 T_{xi}, T_{xf}, T_{x1} である。

$$T_x = (T_{xi}[G_{xi}]/A_{xi}|S_{xi} \rightarrow S_{x1}, T_{xf}[G_{xf}]/A_{xf}|S_{x2} \rightarrow S_{xf}, T_{x1}[G_{x1}]/A_{x1}|S_{x1} \rightarrow S_{x2})$$

階層のモデリング手法について

以上の表記法を用いて、階層のモデリングを行う手法について、以下に説明する。まず、用語を定義する。

開始モデリング階層 あるシステムのモデリング対象となる階層のうち、最初にモデリングを行う階層。モデリングにおける最上位層となる。

終了モデリング階層 あるシステムのモデリング対象となる階層のうち、最も下位に位置する階層。モデリングを行う者が、どこまで階層を掘り下げてモデリングするかによって決定される。

手順1 開始モデリング階層を決定し、Statechart 図によりモデリングを行う。

手順2 開始モデリング階層の Statechart 図の全ての遷移について、その結果引き起こされるアクションの実行内容が影響を及ぼす階層をそれぞれ調べる。

手順3 手順2の結果、さらに下位の層へ影響を及ぼす実行内容のアクションが存在しない場合、モデリングは終了であり、このときの終了モデリング階層は開始モデリング階層と同じである。さらに下位の層へ影響を及ぼす実行内容のアクションが存在する場合、モデリングを行う者は、モデリング対象とする階層をさらに下位の層まで含めるかどうか決定する。含めない場合は、モデリングは終了であり、このときの終了モデリング階層は開始モデリング階層と同じである。含める場合は、手順4に進む。

手順4 現在、モデリングを行っている階層の1つ下位の層における状態変化を包含する状態(以下、下位層の包含状態と呼ぶ)を作る。ただし、下位層包含状態の中身はここでは具体的に展開せず、1つ下位の層に実行が移ることを示すべく、 $L_{\text{下位層の名前}}$ と記しておく。中身となるべき1つ下位の層における状態変化は、これとは別にモデリングを行う。(手順5で行う。)手順2の結果、さらに下位の層に実行の影響が及ぶと判断したアクションを引き起こす遷移は、下位層包含状態へと向かう遷移と、そこから出る遷移に展開する。

図を用いて具体的に説明する。Fig4.3において、階層 x の遷移 T_{x1} が引き

起こすアクション A_{x1} は、1つ下位の層 y にも実行の影響が及ぶものとする。この場合、 T_{x1} は、Fig4.4のように、下位層 y の包含状態 S_{x3} と、そこへ向かう遷移 T_{x1}' 、そこから出る遷移 T_{x2} に展開される。この展開は、それまで T_{x1} で行っていた処理を、 T_{x1}' 、 S_{x3} 、 T_{x2} の順に行われる3つの処理に分割したことを意味する。

手順5 手順4でモデリングを行っていた階層の1つ下位の層における状態変化をモデリングする。この層における初期状態から出る遷移は、1つ上位の層から来る何らかの要求である。また、この層における終了状態へと向かう遷移は、1つ上位の層に送られる何らかの通知である。Fig4.5は、階層 y における状態変化をモデリングした Statechart 図である。

手順6 下位層にも実行の影響が及ぶアクションは、1つ下位の層における状態変化を引き起こす。多くの場合で、1つ下位の層における状態変化の経路は複数存在し、非決定的となる。しかし、このようなアクションをモデリングし、SMART 上で動作を検証する場合を考えると、状態変化の経路は決定的でなくてはならない。この場合は、1つ下位の層における状態変化の経路を1つに決定する情報が必要となる。この情報はシステムの状態によって様々なものが考えられるが、一例として下位層に実行の影響が及ぶアクションと、その実行結果、1つ下位の層の遷移のガード条件が真になる順序が対応づけられたリストを考える。Fig4.5の例では、 S_{y1} から S_{y4} に向かう遷移の組み合わせは2通り存在し、非決定的である。しかし、Table4.1のようなリストを作成することにより、遷移の組み合わせは、 T_{y2} - T_{y4} の1つに決定される。

アクション	展開後の遷移	真になるガード条件	真になる順序
A_{x1}	$T_{x1}'[G_{x1}']/A_{x1}'$, $T_{x2}[G_{x2}]/A_{x2}$	G_{y2}	1
		G_{y4}	2

Table 4.1: List of the L_x and L_y

さらに、Fig4.5において、階層 y の遷移 T_{y4} が引き起こすアクション A_{y4} は、1つ下位の層 z にも実行の影響が及ぶものとする。この場合、 T_{y4} は、Fig4.6のように、下位層 z の包含状態 S_{y5} と、そこへ向かう遷移 T_{y4}' 、そこから出る遷移 T_{y5} に展開される。Fig4.7に、階層 z における状態変化の Statechart 図を示す。Table4.2は、階層 z に実行の影響が及ぶアクション A_{y4} と、その実行結果、1つ下位の層の遷移のガード条件が真になる順序の対応リストである。

アクション	展開後の遷移	真になるガード条件	真になる順序
A_{y4}	$T'_{y4}[G_{y4}]/A_{y4}', T_{y5}[G_{y5}]/A_{y5}$	G_{zi1}	1
		G_{zf1}	2

Table 4.2: List of the L_y and L_z

手順 7 開始モデリング階層に再び焦点を移す。1つ下位の層までモデリング階層を掘り下げ、下位層包含状態が増えたことにより、手順 1 でモデリングした Statechart 図においては、抽象化されていた処理が具体化される場合がある。実際のシステムの処理では、下位層のプロパティ値を参照したり、下位層の機能を用いなくてはならないが、そこまでモデリングを掘り下げない場合は、モデリング対象となる階層の範囲で表現できるように、下位層のプロパティ値や機能を抽象化し、代わりとなる処理をモデリングすることになる。しかし、モデリング階層が掘り下げられると、それまで抽象的に表現していた処理が、具体的に表現できるようになる。このような場合は、開始モデリング階層のモデリングにおいて抽象化していた表現を具体化し、Statechart 図に修正を加える。これについては、後の“処理の具体化”の部分で、具体的な例を交えて説明する。

手順 8 開始モデリング階層から 1つ下位の層に焦点を移して、手順 2 から手順 7 の作業を繰り返す。この作業はモデリングを行う者が、終了モデリング階層を決定するまで続く。

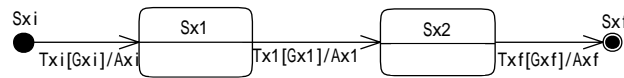


Fig 4.3: Statechart diagram of the layer $x(L_x(x))$

$$\begin{aligned}
 L_x(x) &= (S_x, T_x) \\
 S_x &= (S_{xi}, S_{xf}, S_{x1}\{\}, S_{x2}\{\}) \\
 T_x &= (T_{xi}[G_{xi}]/A_{xi}|S_{xi} \rightarrow S_{x1}, T_{xf}[G_{xf}]/A_{xf}|S_{x2} \rightarrow S_{xf}, T_{x1}[G_{x1}]/A_{x1}|S_{x1} \rightarrow S_{x2})
 \end{aligned}$$

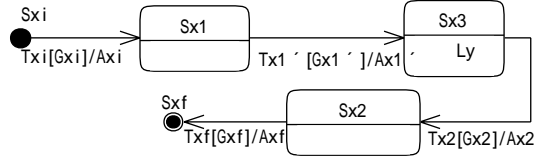


Fig 4.4: Statechart diagram of the layer $x(L_x(y))$

$$\begin{aligned}
 L_x(y) &= (S_x, T_x) \\
 S_x &= (S_{x_i}, S_{x_f}, S_{x_1}\{\}, S_{x_2}\{\}, S_{x_3}\{L_y\}) \\
 T_x &= (T_{x_i}[G_{x_i}]/A_{x_i} | S_{x_i} \rightarrow S_{x_1}, T_{x_f}[G_{x_f}]/A_{x_f} | S_{x_2} \rightarrow S_{x_f}, T_{x_1}[G_{x_1}]/A_{x_1} | S_{x_1} \rightarrow S_{x_3}, \\
 &\quad T_{x_2}[G_{x_2}]/A_{x_2} | S_{x_3} \rightarrow S_{x_2})
 \end{aligned}$$

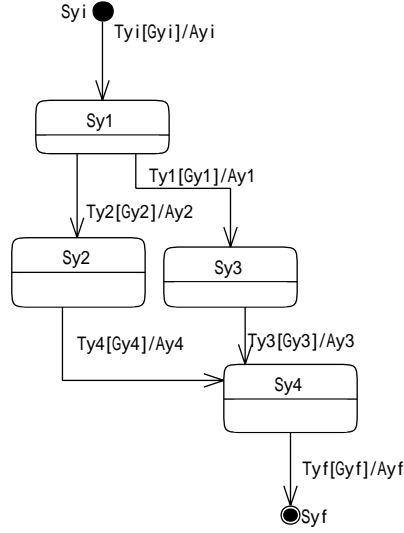


Fig 4.5: Statechart diagram of the layer $y(L_y(y))$

$$\begin{aligned}
 L_y(y) &= (S_y, T_y) \\
 S_y &= (S_{y_i}, S_{y_f}, S_{y_1}\{\}, S_{y_2}\{\}, S_{y_3}\{\}, S_{y_4}\{\}) \\
 T_y &= (T_{y_i}[G_{y_i}]/A_{y_i} | S_{y_i} \rightarrow S_{y_1}, T_{y_f}[G_{y_f}]/A_{y_f} | S_{y_4} \rightarrow S_{y_f}, T_{y_1}[G_{y_1}]/A_{y_1} | S_{y_1} \rightarrow S_{y_3}, \\
 &\quad T_{y_2}[G_{y_2}]/A_{y_2} | S_{y_1} \rightarrow S_{y_2}, T_{y_3}[G_{y_3}]/A_{y_3} | S_{y_3} \rightarrow S_{y_4}, T_{y_4}[G_{y_4}]/A_{y_4} | S_{y_2} \rightarrow S_{y_4})
 \end{aligned}$$

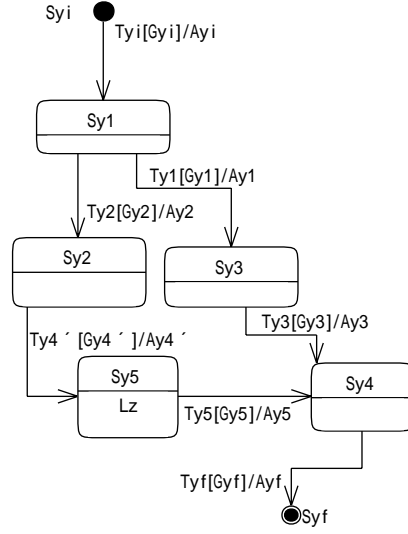


Fig 4.6: Statechart diagram of the layer $y(L_y(z))$

$$\begin{aligned}
 L_{y(z)} &= (S_y, T_y) \\
 S_y &= (S_{y_i}, S_{y_f}, S_{y_1}\{\}, S_{y_2}\{\}, S_{y_3}\{\}, S_{y_4}\{\}, S_{y_5}\{L_z\}) \\
 T_y &= (T_{y_i}[G_{y_i}]/A_{y_i} | S_{y_i} \rightarrow S_{y_1}, T_{y_f}[G_{y_f}]/A_{y_f} | S_{y_4} \rightarrow S_{y_f}, T_{y_1}[G_{y_1}]/A_{y_1} | S_{y_1} \rightarrow S_{y_3}, \\
 &T_{y_2}[G_{y_2}]/A_{y_2} | S_{y_1} \rightarrow S_{y_2}, T_{y_3}[G_{y_3}]/A_{y_3} | S_{y_3} \rightarrow S_{y_4}, T_{y_4}[G_{y_4}]/A_{y_4} | S_{y_2} \rightarrow S_{y_5}, \\
 &T_{y_5}[G_{y_5}]/A_{y_5} | S_{y_5} \rightarrow S_{y_4})
 \end{aligned}$$

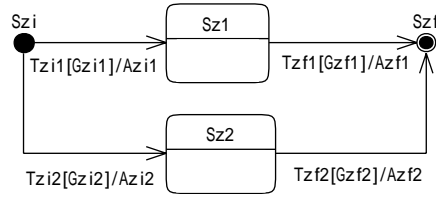


Fig 4.7: Statechart diagram of the layer $z(L_z(z))$

$$\begin{aligned}
 L_{z(z)} &= (S_z, T_z) \\
 S_z &= (S_{z_i}, S_{z_f}, S_{z_1}\{\}, S_{z_2}\{\}) \\
 T_z &= (T_{z_i1}[G_{z_i1}]/A_{z_i1} | S_{z_i} \rightarrow S_{z_1}, T_{z_i2}[G_{z_i2}]/A_{z_i2} | S_{z_i} \rightarrow S_{z_2}, T_{z_f1}[G_{z_f1}]/A_{z_f1} | S_{z_1} \rightarrow S_{z_f}, \\
 &T_{z_f2}[G_{z_f2}]/A_{z_f2} | S_{z_2} \rightarrow S_{z_f})
 \end{aligned}$$

Fig4.1 の例では、開始モデリング階層はアプリケーションソフトウェア部（以後、 L_a と表記する。）、また終了モデリング階層もアプリケーションソフトウェア部である。Fig4.1 の“判定結果 [一定時間経過]/It 更新”という遷移を展開し、ECHONET 通信処理部（以後、 L_c と表記する。）を終了モデリング階層として掘り下げた Statechart 図が、Fig4.8、Fig4.9 である。さらに、プロトコル差異吸収処理部まで掘り下げた Statechart 図が、Fig4.10、Fig4.11 である。

処理の具体化

手順 7 で説明したように、モデリング階層を掘り下げると、それまで抽象的だった処理が、具体的に表現できるようになる場合がある。以下に、例を使ってこれを説明する。

Fig4.9 を見ると、“初期設定処理”という状態がある。この状態に入ると、“ECHONET オブジェクトに自機器の Node ID を設定”というアクションが引き起こされる。このモデリングでは、ECHONET 通信処理部の範囲内で処理が行われている。しかし、実際の ECHONET では、Node ID は下位通信ソフトウェア部が保持しており、これを参照しに行く必要がある。まず、“初期設定処理”は、Node ID 情報の要求をプロトコル差異吸収処理部に出す (Fig4.10)。次に、プロトコル差異吸収処理部の“共有下位通信インターフェイス処理”がこの要求を受け取り、下位通信ソフトウェア部に Node ID 情報要求を出す (Fig4.12)。下位通信ソフトウェア部は、保持する Node ID を調べ、プロトコル差異吸収処理部に Node ID 通知を返す。プロトコル差異吸収処理部の“共通下位通信インターフェイス処理”がこれを受け取り、ECHONET 通信処理部へ通知する (Fig4.12)。そして、ECHONET 通信処理部の“初期設定処理”に Node ID 情報が通知されて処理は終わる (Fig4.10)。

このように、モデリング階層が下位通信ソフトウェア部まで掘り下げられたとき、Fig4.9 の“初期設定処理”に入ると引き起こされる“ECHONET オブジェクトに自機器の Node ID を設定”というアクションは、Fig4.10 まで具体化される。この処理の具体化は、プログラミング理論における「詳細化 (refinement)」の一種として考えることができる。上位層で抽象化していた処理を、下位層にモデリングを掘り下げるに従い、具体的な記述に換えていく。このことにより、モデリングを行う者は、初めから処理の詳細、つまりシステムの最下位層の状態変化を意識したモデリングを記述することなく、モデリングする階層に対応して、処理の内容を具体化していけばよい。このように、下位層に影響が及ぶアクションを具体化した後の、下位層における一連のアクションは、具体化する前のアクションを実装する。

また、プログラミング理論の「詳細化」の作業においては、具体化した仕様は、抽象化されていたときの仕様を満たさなくてはならないという条件が存在する。モデル階層において、この条件は次のように考えることができる。

下位層に影響を及ぼすアクションを具体化した後のモデルにおいて、上位層から処理が移り、引き起こされる下位層における一連のアクションの効果が、具体化する前のアクションの効果と同じであること。

上の条件では、「一連のアクションがあるアクションと同じ効果である」という表現になっているが、これを正確に記述するための方法は、今後考えていかななくてはならない。現在のところ、UMLのユースケースの precondition、postcondition (“Writing Effective Use Cases” 参照) の導入や、Z 記法による仕様記述 (“プログラム検証論” 参照) などが考えられる。

以上のモデル階層化機能を実現することで、階層構造を持ったシステムのモデリングにおいて、ある階層の処理の中身を、システムの階層に対応して、随時詳細化していくことが可能となる。そのため、システム全体を鳥瞰する視点でのモデリングから、個々の部分に関してのより詳細なモデリングまで、ユーザの目的に応じて、モデリング対象となる階層の範囲を選択できるようになる。

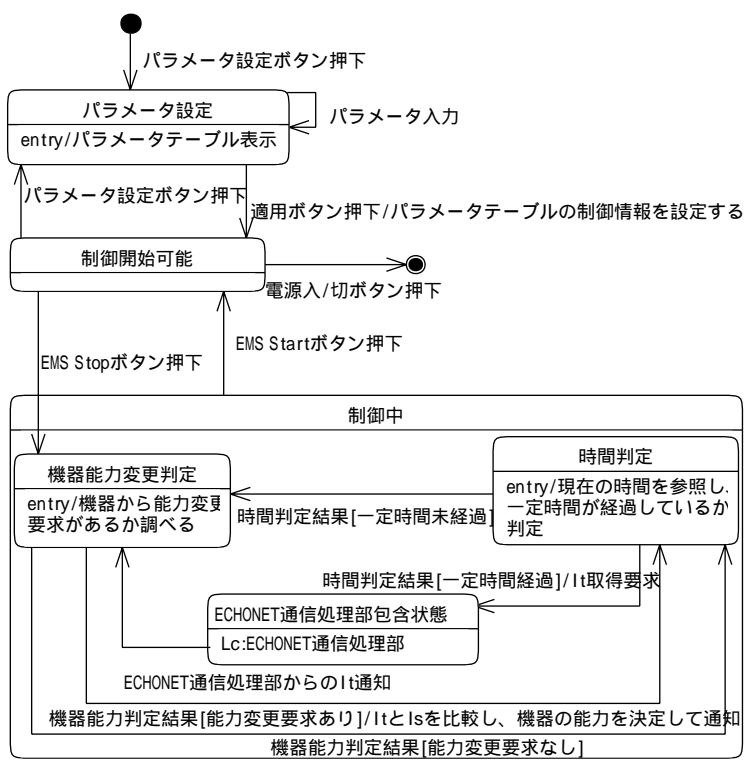


Fig 4.8: Statechart Diagram of the $L_{a(c)}$ in case of feedforward EMS

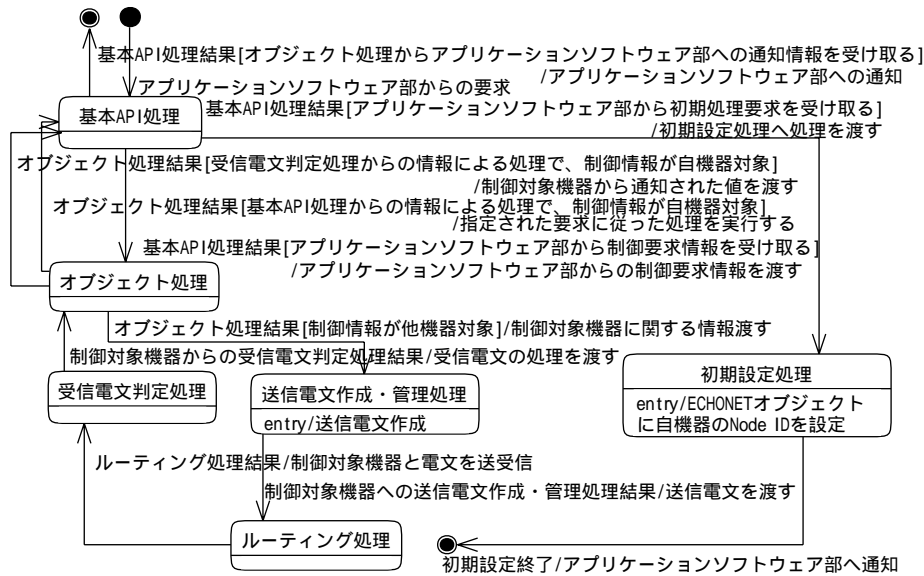


Fig 4.9: Statechart Diagram of the $L_{c(e)}$ in case of feedforwad EMS

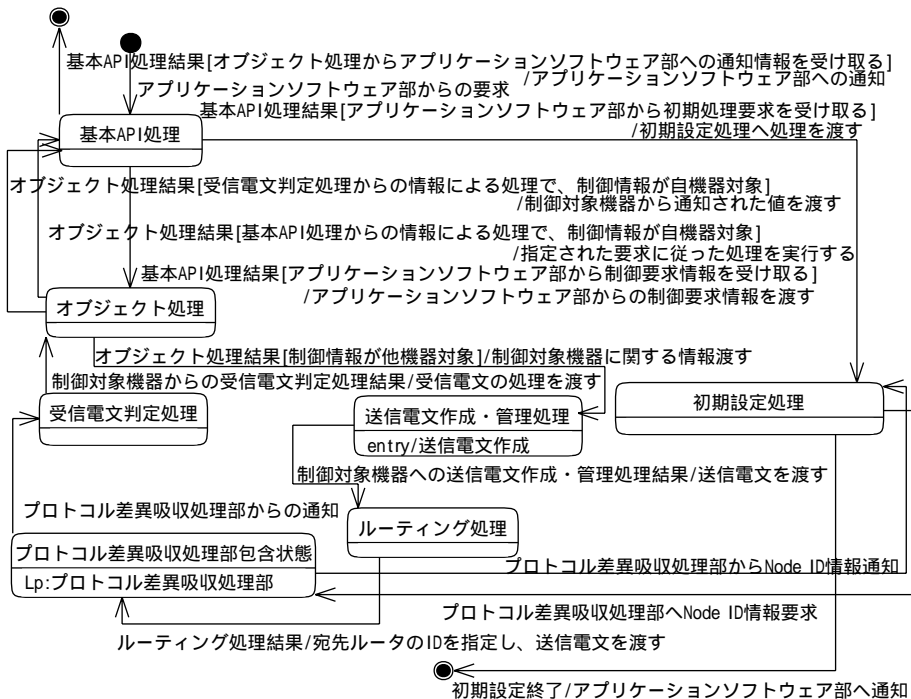


Fig 4.10: Statechart Diagram of the $L_{c(p)}$ in case of feedforwad EMS

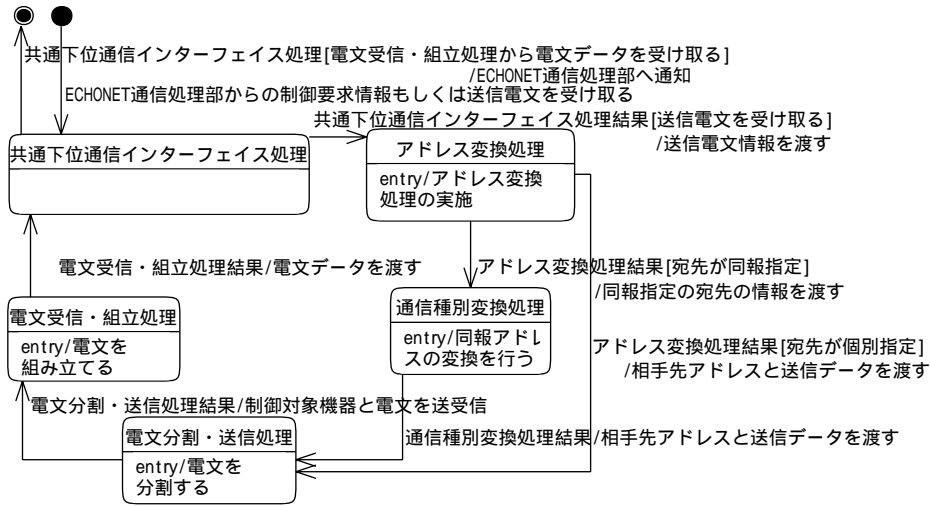


Fig 4.11: Statechart Diagram of the $L_p(p)$ in case of feedforwad EMS

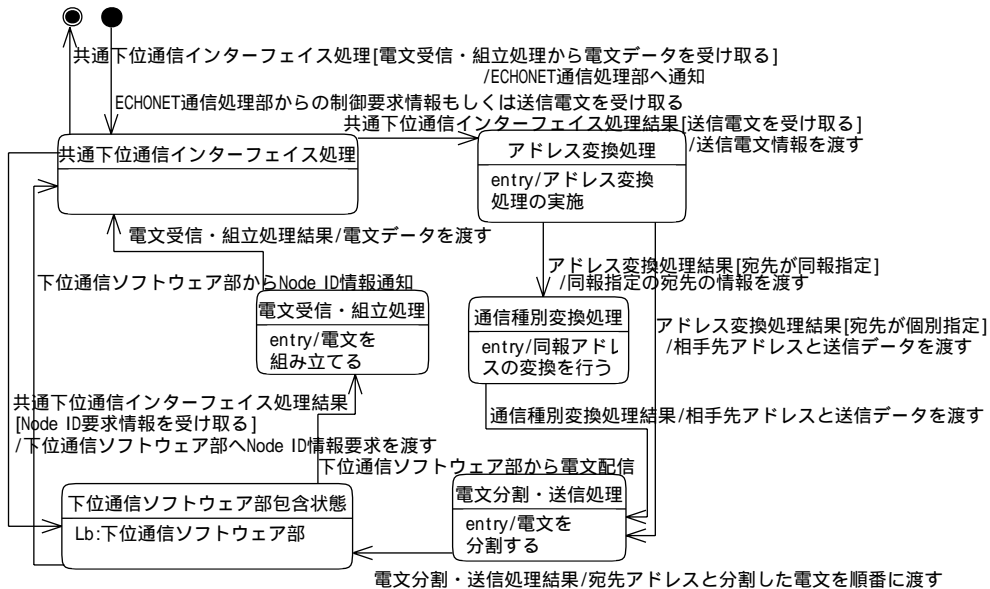


Fig 4.12: Statechart Diagram of the $L_p(b)$ in case of feedforwad EMS

第5章 結論および今後の展開

5.1 結論

本研究により、ホームネットワークのモデリング、シミュレーションツールとして SMART を発展させるために必要な重要機能のいくつかを抽出することができた。特にモデル階層化機能において、現在の UML 仕様では規定されていない、状態間の遷移によって引き起こされるアクションの階層の定義と表現について、ひとつの方法を提示した。これにより、Statechart 図におけるアクションの表現能力を高めることができる。このモデル階層化機能を SMART が搭載すれば、ホームネットワークのように階層構造を持つシステムのモデリングが容易になると考えられる。

また、ホームネットワークのケース・スタディとして ECHONET EMS に焦点を当て、そのシミュレータを実装した。今後このシミュレータは、今回提案した新機能を SMART に搭載する際のプログラミング手法を探る材料として活用していく予定である。

5.2 今後の展開

序論で述べたように、本卒業研究は、花山による SMART の ActionSemantics エンジンの並列化、森による SMART の分散化を技術的基盤として踏まえ、ホームネットワークのモデリングとシミュレーションにより特化した SMART ツールの機能の提案から、その搭載までを目標とする中期的研究の一部である。以下にこの研究の今後の展開を述べる。

5.2.1 ECHONET 通信レイヤに対応した EMS シミュレータの実装

今回作成した EMS シミュレータは、ECHONET 規格にある EMS アプリケーションソフトウェアの各制御方式をシミュレートすることに重点を置いており、ECHONET 通信レイヤの各階層に対応した実装にはなっていない。そのため、最下位層にあたる伝送メディアごとのプロトコルの違いなどをシミュレートすることはできない。今後、ECHONET 通信レイヤの各階層の役割を実装に反映させ、アプリケーションソフトウェア部より下位の層の動作もシミュレートできるものにしていく。その過程で、本論文で提案したモデル階層化機能を SMART に搭載

する手法を探っていく予定である。また、SMARTでのホームネットワークのモデリング、シミュレーションに必要な機能についてもさらに考察していきたい。

5.2.2 SMARTのモデル階層化機能の搭載

ECHONET通信レイヤに対応したEMSシミュレータの実装を行っていく中で、SMARTにモデル階層化機能を搭載するためのプログラミング手法を探り、そこでの経験を元に、SMARTへの搭載を実際に行っていく。その際、モデルの階層が、SMART0.2のStatechart ModelerとUser Interface Modeler上で、それぞれどのように視覚化されればユーザにとって都合がよいかを、森が提案したView Pointの概念に即して考えていく。

5.2.3 並列動作に対応したSMART上でのEMSシミュレータの構築

花山の研究であるAction Semanticsエンジンが完成すれば、並列動作に対応したSMARTが出来上がる。この上に、本研究の成果であるホームネットワークのモデリング、シミュレーションに必要な機能を搭載して、ホームネットワークのモデリングとシミュレーションに特化したSMARTツールを完成させる。そして、その代表的な使用例としてEMSシミュレータを構築する。

謝辞

はじめに、筆者がこの研究を進めるにあたり、理論的基礎から実装に至るまでの全過程において熱心な御指導、御教授をいただきました神戸大学工学部情報知能工学科 林晋教授に心から感謝致します。また、EMSシミュレータの設計において、筆者の質問に熱心に答え、アイデアを提供していただいた神戸大学工学部情報知能工学科 賀谷信幸教授に深く感謝致します。さらに、SMARTプロジェクト全般に渡り、多くの指針を提供してくださった松下電器産業 春名修介氏、株式会社EIZOの谷内上智春氏に感謝致します。最後に、常日頃より様々な面でお世話になったCS33のメンバーにも感謝致します。

参考文献

[UMLS 01] Object Management Group, “OMG Unified Modeling Language Specification” <http://www.omg.org> (2001)

[WERU 98] Alistair Cockburn, “Writing Effective Use Cases”, Addison-Wesley Pub Co (Sd) (1998)

[UMLU 99] Grady Booch, James Rumbaugh, Ivar Jacobson, “The Unified Modeling Language User Guide”, Addison-Wesley Pub Co (Sd) (1999)

[Alhi 98] Sinan Si Alhir, “UML IN A NUTSHELL: A Dektop Quick Reference”, O'Reilly & Associates (1998)

[Brue 01] Bruce Powel Douglass, “リアルタイム UML 第2版”, 翔泳社 (2001)

[Haya 95] 林晋, “プログラム検証論”, 共立出版 (1995)

[Echo 02] ECHONET CONSORTIUM, “The ECHONET Specification Version2.11”, <http://www.echonet.gr.jp> (2002)

[Yach 02] 谷内上智春, “UML を用いた組み込みシステムの開発支援系”, 卒業論文, 神戸大学 (2002)

付録

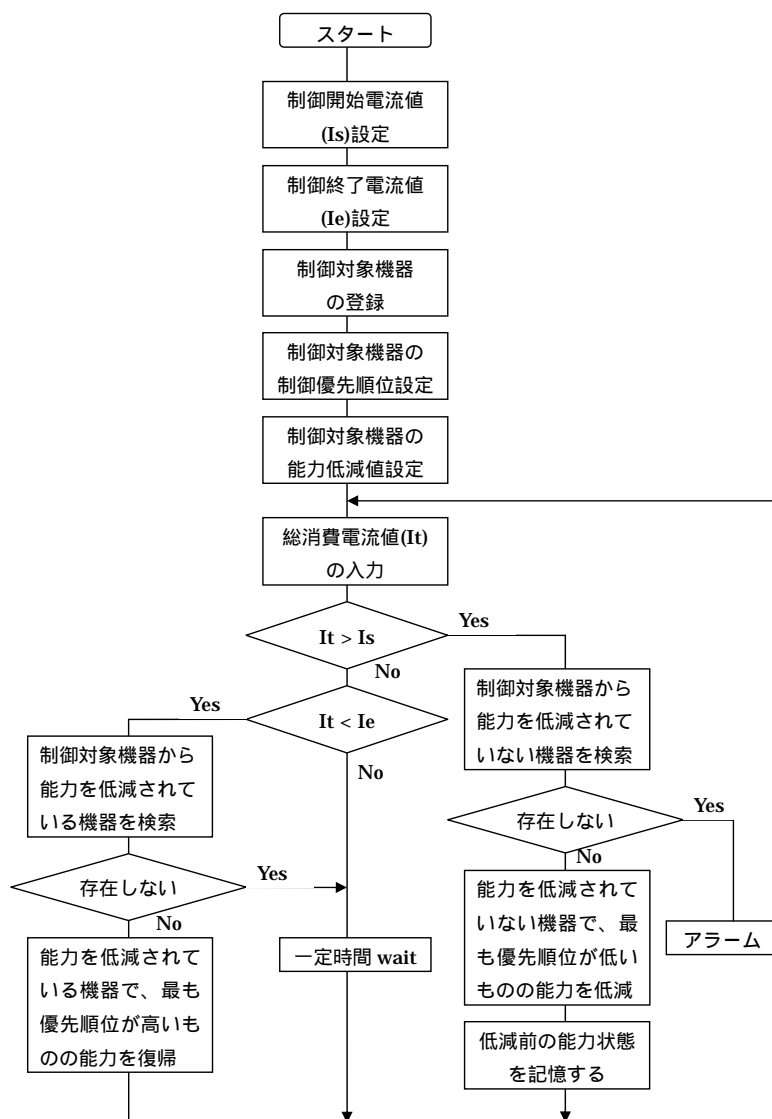


Fig 5.1: Feedback EMS

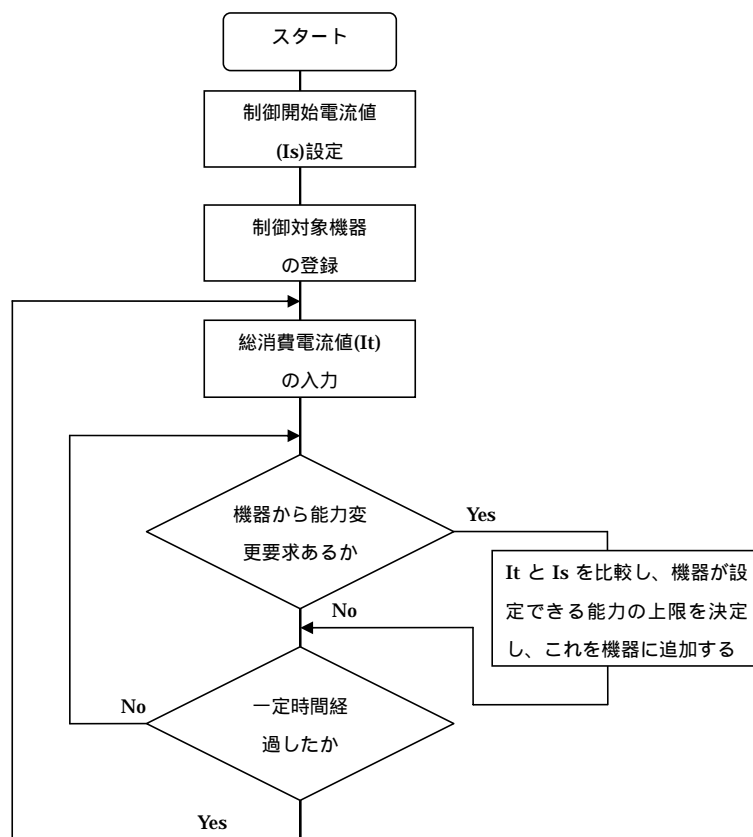


Fig 5.2: Feedforward EMS

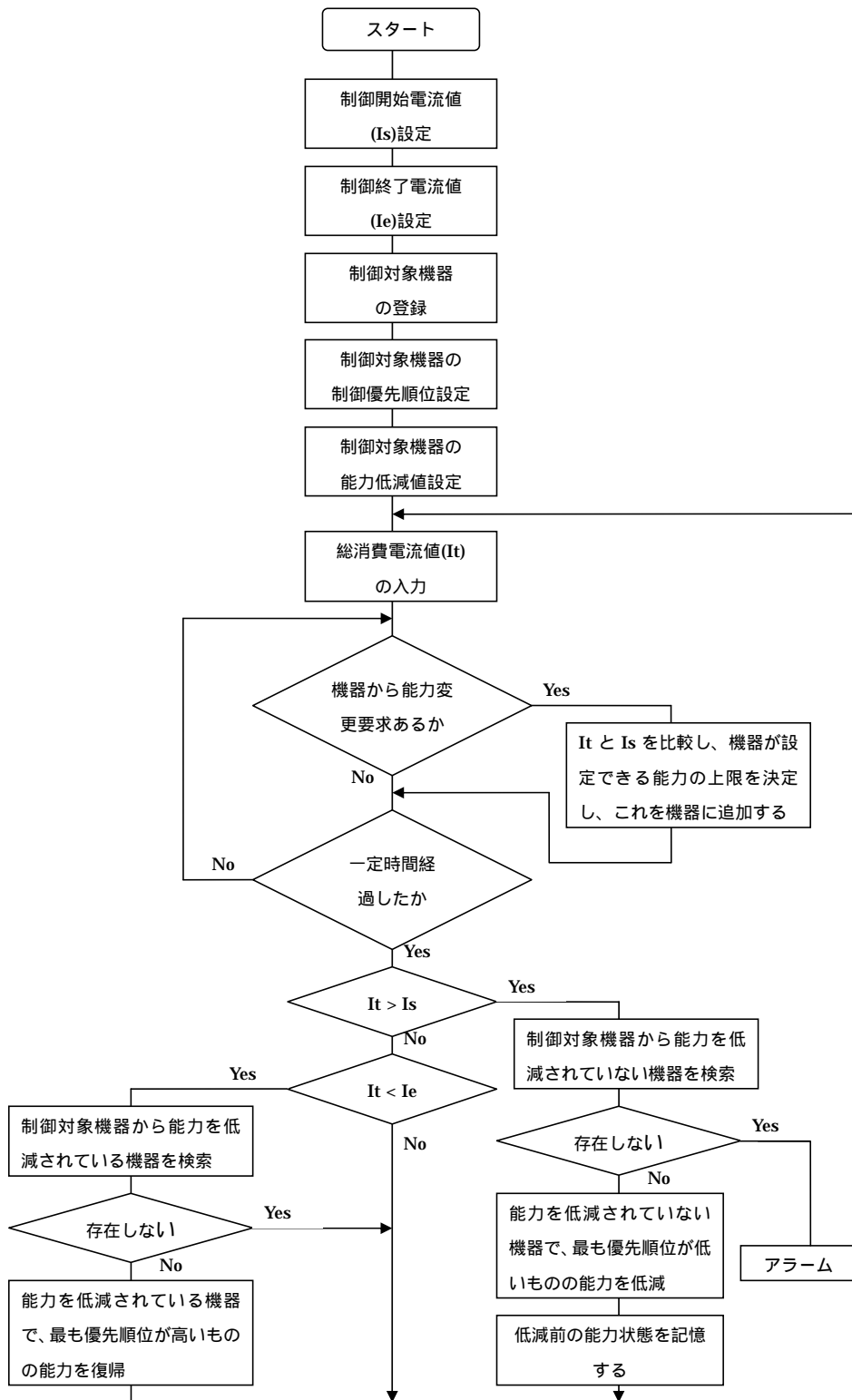


Fig 5.3: Hybrid EMS