ソフトウェア工学 71－1
（1990．2．8）

ソフトウェア動作仕様の段階的設計技法とその支援系

小山田正史　岩戸伝一

情報処理振興事業協会・技術センター

通信システムや交換システムなどに組込まれるリアルタイムソフトウェアの設計では、さまざまな技法や支援システムが開発されている。本論文では、リアルタイムソフトウェアに特徴的なシステムの動作仕様を段階的に構築できる設計技法について述べる。本技法は、システムの要求仕様(サービス仕様)から動作仕様(状態遷移図)を、5つの視点からを捉える。これらの視点を設計に段階的に適用するために、サービス機能、動作シナリオ、インタラクション、状態遷移からなる4つの設計ステップが提案される。さらに、編集、確認、変換の面から設計ステップを支援する支援システムについて述べる。

A Stepwise Design Methodology and Support System for Real-Time Software

Masashi Koyamada　Denichi Iwado

Software Technology Center
Information-Technology Promotion Agency, Japan (I. P. A.)
Shuwa-Shibakouen 3-Chome Bldg., 3-1-38 Shibakouen, Minato-Ku, Tokyo 105, Japan

The paper describes a methodology and support system for real-time software design. The methodology viewpoints consist of service hierarchy and object structure as static aspect and behavior scenario, messege sequence and state transition as dynamic aspect. Each of the viewpoints has a visual notation. The methodology provides design activities to construct specifications、which includes the viewpoints, in a stepwise manner. Also the support system based on the methodology is presented. The stepwise, visual approach helps less-exerienced designers to successfully design real-time software.

1 はじめに

近年、通信システムや制御システムの多様化、大規模化にともない、システムに組み込まれるリアルタイムソフトウェア開発の効率化が重要な課題になっている。リアルタイムソフトウェアを対象としたさまざまな設計技法や支援システムが提唱・開発されている。通信・交換システムの分野で広く使われている技法の一つとしてSDL (Specification and Description Language)[1]があるが、SDLは設計の視点(機能階層と状態遷移)とその表記法だけを提供している。また、リアルタイム構造化分析[2]では、システムの動作を制御フローと状態遷移によりとらえている。従来の技法の枠組みでシステムの設計を行うには高度なスキルが要求され、スキルをもった設計者の不足が開発のネックの一つになっている。

本論文の目的は、従来経験的に行われていたリアルタイムソフトウェア設計に対して、スキルのない設計者でも設計を円滑に進めるための一つの指針(技法)を与えることにある。対象とする工程は、リアルタイムソフトウェアに特徴的な、システムに与えられたサービスを明確に定義することからサービスを満たすシステムの動作仕様を定義することまでの開発過程である。

本論文で提案される設計技法は、システムを2つの静的な視点と3つの動的な視点からとらえる。これらの視点を設計作業に適用するために、設計過程を4つの設計ステップの流れとしてとらえる。また、設計のそれぞれの視点の仕様をより直観的に理解できるようにするために、各視点に視覚的な表記法が与えられている。

本論文では、まずリアルタイムソフトウェア設計における問題点について議論する(2節)。3節、4節でリアルタイムソフトウェアの動作設計に着目した設計技法を提案する。まず、3節で、システムの動作をとらえる5つの視点について述べる。4節で、5つの視点をベースとし、ソフトウェアの動作を段階的に設計する4つの設計ステップについて述べる。5節では、本技法を支援するシステムの構成と特徴について述べる。6節では、本技法の有効性について議論する。

2 背景

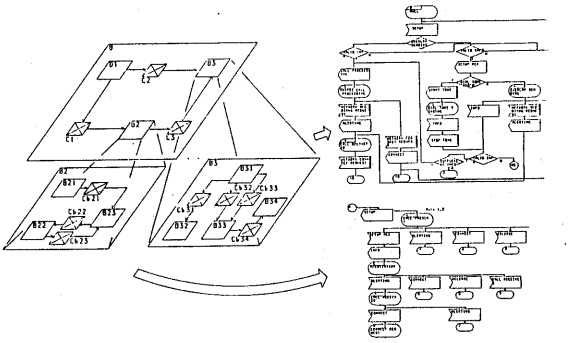
リアルタイムソフトウェアの設計では、一般に、システムを静的な側面と動的な側面とから捉えている。

(1) 静的な側面→機能階層構造による表現

(2) 動的な側面→状態遷移による表現

システムの静的な側面を表すために、機能構造を階層的に捉えることが、一般的なソフトウェア設計技法に広く採り入れられている。動的な側面は、リアルタイムソフトウェアの仕様化に特徴的なものであり、状態遷移図や状態遷移表が使われることが多い。

代表的な仕様化技法の一つであるSDL[1]では、ブロックインタラクションダイアグラムにより静的な構造を、プロセスダイアグラムにより動的なふるまいを表す(図1)。ブロックインタラクションダイアグラムでは、システムの論理的な構成要素(機能ブロック)を階層的に表現する。機能ブロックは互いに通信しあう情報(信号)の流れを表すチャンネルによって結び付けられている。各機能ブロック(に含まれる動作単位であるプロセス)の信号の通信手順を、有限状態マシンをベースとしたプロセスダイアグラム(状態遷移図)により表す。



(a) ブロックインタラクションダイアグラム　(b) プロセスダイアグラム(状態遷移図)

図1: 設計仕様化の視点(SDL)

従来の技法によっても、システムの動作仕様を正確に形式的に記述することができる。設計を円滑に進める上で、次のような問題点が挙げられる[7]。

(1) サービス仕様と動作仕様との対応

リアルタイムソフトウェア分野のシステムでは、システムが提供するさまざまなサービスは、複数の並行動作するプロセスが互いに通信し合いながら実現される。SDLの機能ブロック(あるいはプロセス)構成は、サービスではなく実現の構成を表している。このため、一つ一つのプロセスの動作を理解したとしても、システムのもつサービス全体を把握することはできない。サービスの追加や変更が発生したとき、どのプロセスに影響があるのかを明示的に表現できない。

(2) 静的仕様から動的仕様へのギャップ

SDLでは、システムの静的な仕様としてプロセス構成やプロセス間で通信し合う信号の集合が定義される。しかし、この仕様には、動的な仕様である状態遷移図を導出するための手がかりがほとんどない。動的な仕様(状態遷移図)をどのように設計するのかは、設計者の経験に任されている。このため、この設計を行うには、高度なスキルが要求される。ただし、補助的な仕様として、システムの動作例を表すメッセージシーケンスチャートが使われることはある。

(3) システムの状態の構造化・視覚化

複数の並行プロセスにより実現されるシステムの設計では、プロセス間の通信(プロトコル)だけでなく、シ

ステム全体(あるいはプロセス)の状態を理解することも重要である。一般に使われている状態遷移図や状態遷移表では、システムが動作する時点時点での状態は、状態の番号や名前によって知ることができる。しかし、番号や名前がシステムのどのような(論理的あるいは物理的な)状態に対応しているのかを理解していなければならない。SDL の状態遷移図では、システムの状態を直観的に理解するのを助けるために、状態に補助的な絵(Pictorial Element)を記述することができる。

このほかにも、リアルタイムソフトウェアの設計では、システムの種類に応じて、フォルトトレラント、超多重化などを考慮する必要がある[3]。本論文では、多くのリアルタイムソフトウェア設計に共通して適用することができ、かつ重要な役割を果す論理的な動作仕様の設計を中心に検討を進める。

3 動作設計の視点

3.1 視点の枠組み

本技法では、サービス空間、状態空間、時間の3つの軸により、設計の仕様を捉える(図2)。

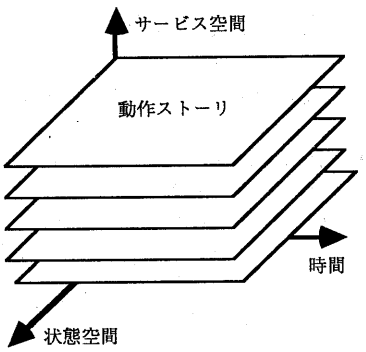


図2: 動作設計の視点

サービス空間: 対象システムがユーザ(あるいはシステム外部)に対して提供するサービスの集合である。

状態空間: 対象システムの実現の構造を表すためのものである。対象システムが動的に変化するときにとり得る可能な状態の集合である。

時間: 対象システムの動的なふるまい(状態の時系列)を表すために使われる論理的な時刻の集合である。

サービス空間と状態空間は、対象システムに要求される仕様と対象システムを実現する仕様とを静的な面からそれぞれ捉えたものである。動的な側面は、時間に対してサービス空間と状態空間を関係づけることにより表す。本技法では、システムの動作は、単にシステムの状態がどのように時間的に変化するのかによって表されるだけではない。システムに要求されるサービスの各々に着目して、システムの状態が時間的にどのように変化するのかをもとらえる。一つのサービスに対する状態空間と時間との関係を動作ストーリと呼ぶことにする。

本節では、以降、本技法における静的と動的な側面を構成する設計の視点について述べる。

3.2 静的な側面

本技法では、システムの静的な側面を、システムに要求されるサービスを階層的に表すサービス階層と、サービスを実現するための機能を階層的に表す対象階層とによってとらえる。

(1) サービス階層

システムが提供するサービスの包含関係をサービス階層として表す(図3)。サービス階層では、最上位のサービスがシステムのサービス全体に対応する。最下位のサービスは、システムが提供する最小単位のサービスに対応する。

サービスの包含関係は、択一的なものと順次的なものとに分けられる。択一的な包含関係では、包含されているサービスの中で、どれか一つをシステムは提供することができる。一方、順次的な包含関係では、システムは、包含されているサービスのすべてを定義された順序に従って提供する。

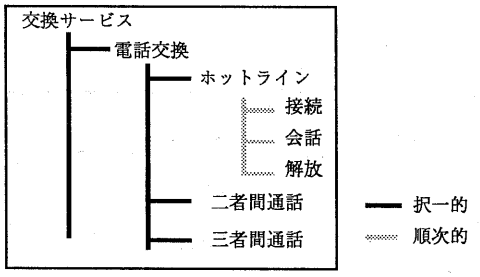


図3: サービス階層の例

(2) 対象階層

システム内外の状態を制御することを主とするシステムでは、システムのサービスを説明しようとするとき、制御の対象が必要となる。このような対象の包含関係を階層的に捉えたのが対象階層である。

対象階層における最上位の対象は、システムを構成したりシステムの外部環境に現われるすべての対象を含んでいる。最下位の対象は、サービスを説明するときに必要な制御対象を表している。図4に、対象階層の定義例を示す。

さらに、対象の論理的な構造を、構造アイコンにより表現する。次の簡単な電話機の例を考えてみよう。

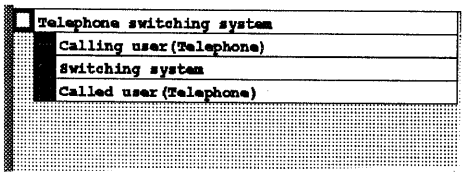


図4: 対象階層の例

「(1)電話機は、受話器、ダイアル、ベル、電話機本体からなる。(2)受話器には、受話器が下りている(オンフック)状態と上がっている(オフフック)状態とがある。....」

(1)において、電話機がもつ4つの要素(受話器、ダイアル、ベル、電話機本体)は電話機の状態を表すために常に必要である。このような要素の関係を構造体と呼ぶ。一方(2)において、受話器がもつ2つの要素(オンフックとオフフック)は、それぞれ受話器の状態に対応し常にどちらか一方が成立つ。このような要素の関係を共用体と呼ぶ。また、電話機本体のようにそれ以上の要素をもたないものをプリミティブと呼ぶ。構造アイコンには、アイコン名と視覚的な要素(シンボルや文字列)を割当てることができる。図5に、電話機の構造アイコンの定義例を示す。

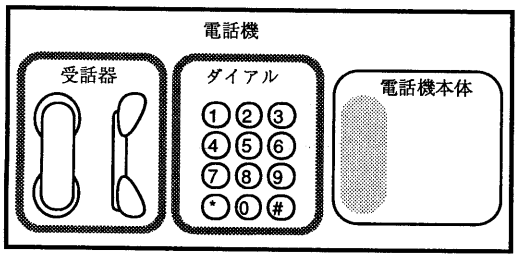


図5: 構造アイコンの例

このように、構造アイコンには、対象の状態構造を論理的に表す側面と視覚的に表す側面とがある。

3.3 動的な側面

システムの動的な側面を表すために、本技法では、(1) 動作シナリオ、(2)メッセージシーケンス、(3)状態遷移の3つの視点が用意されている。動作シナリオとメッセージシーケンスは、一つのサービスについて、システムの動作(動作ストーリ)を表すためのもので、状態遷移はサービス全体に対するシステムの動作を表すためのものである。

まず、次の簡単な動作説明(自然語による動作ストーリの記述)の例を考えてみよう。

「(1)受話器を上げて、(2)ダイアルすると、(3)相手の電話機の呼出し音が鳴る。次に、(4)相手が受話器を上げたら、(5)会話して、(6)相手が受話器を下ろしたら、(7)自分も受話器を下ろす。」

この例では、二者間通話のサービスを、おもに二つの電話機(自分と相手)に対する操作を時系列的に述べていることにより説明している。この説明を理解するとき、暗黙のうちに二つの電話機の状態を頭の中で考えている。例えば、(2)のダイアルのときには、自分側の電話機はダイアルされただけでなく受話器が上がった状態になっている。操作系列の各操作の前後の二つの電話機の状態を時系列的に捉える(アニメーションのように表示する)ことにより、上述の動作説明と同様の動作を理解することができる。動作ストーリでは、操作系列に着目した視点と状態系列に着目した視点との2つの視点から捉えることができる。

(1) 状態の変化に着目した動作

動作ストーリについて、状態の時間的な変化に着目した視点を動作シナリオと呼ぶ。各時点時点で、システム全体の状態を記述する。システム動作のアニメーションが動作シナリオの例である。アニメーションでは、動作の一連の流れが理解できるだけでなく、一コマ一コマを見るとその時点でのシステムの状態を理解することができる。

(2) 相互作用に着目した動作

動作ストーリについて、操作の時間的な系列に着目した視点をメッセージシーケンスと呼ぶ。(本技法では、操作はメッセージ通信により行われることを前提とする。このため、操作シーケンスではなく、メッセージシーケンスと呼ぶことにした。)この視点では、各時点時点で、システム状態の変化分(差分)だけを記述する。シーケンスチャートがこの視点の例である。システム内外で受け渡されるメッセージの方向と時系列的な流れを理解することができる。

両視点では、次のような違いがある。

- 理解しやすさ: 状態変化 > 相互作用
- 記述しやすさ: 状態変化 < 相互作用

(3) 対象に着目した動作

3つ目の視点である状態遷移は、対象が関係するすべてのサービスを実現するためのメッセージの入出力手順を状態遷移モデルを用いて記述する。システムを構成する対象に着目した動作を理解することができる。

4 動作設計のステップ

本技法では、要求仕様から状態遷移図を作成するまでを、4つの設計ステップとして捉えている(図6)。

まず、サービス機能ステップにおいて、システムの動

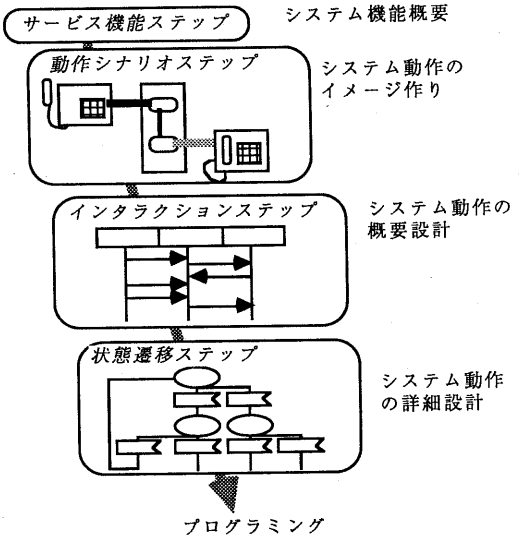


図6: 動作設計のステップ

作を設計するための枠組みとなるサービス階層と対象構造を明らかにする。次に、動作シナリオステップにおいて、直観的に理解しやすい動作シナリオの定義を行う。インタラクションステップでは、動作シナリオに対応して、対象間の相互作用を表すメッセージシーケンスを設計する。最後に、状態遷移ステップにおいて、さまざまなサービスに対して定義されたメッセージシーケンスをまとめて、最終的な動作仕様である状態遷移図を定義する。

(1) サービス機能ステップ

システムの動作を表すための枠組みとして、サービス階層と対象構造とを定義する。サービス機能ステップは、次の2つの作業からなる。

・サービスの洗い出しと階層化

まず、システムに対する要求仕様を整理して、サービス階層としてまとめる。サービス階層は、システムのユーザあるいは外部環境から見て、システムが提供する機能を階層的にまとめたものである(図3)。

・対象の洗い出しと構造化

個々のサービスに対するシステムの動作を説明したり理解するときには、操作の対象あるいは動作するもの(対象)が必要となる。このような対象を洗い出し、対象間の関連も定義する(図4)。さらに、対象の構造を構造アイコンにより定義する(図5)。対象の構造は、対象を実現するための詳細な内部構造を表すというよりも、サービスを説明するために必要な対象の論理的な構造を表す。

(2) 動作シナリオステップ

サービス階層の各サービス(最下位レベル)について、システムがどのように動作するのかを、関連する対象とその構造アイコンを使って記述する。

まず、1番目のフレームに、構造アイコンを使って各対象の初期状態を描く。システム動作の変化に対応して、一つ一つフレームを追加し、そのときの各対象の状態を構造アイコンによって描いていく(図7)。このようにして作成されたフレーム列を連続的に見ることにより、システムの時系列的な状態の変化を理解することができる。本ステップでは、一つのサービスについて、システムが状態空間上の時間的な変化(動作ストーリ)を、システムの(状態)構造に着目した視点からとらえ、システムの動作をスケッチする。

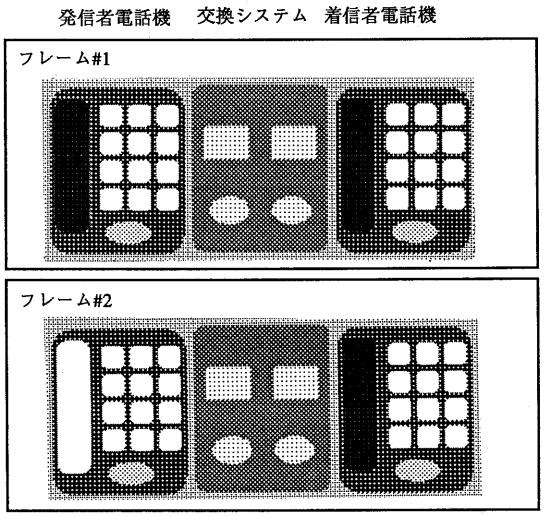


図7: 動作シナリオの例

このステップでは、対象間で受け渡されるメッセージデータを考える必要はない。

(3) インタラクションステップ

本ステップでは、各サービスについて、動作シナリオステップで作成された動作シナリオをもとに、対応するメッセージシーケンスを作成する。まず、対象間で受け渡されるメッセージ名を定義する。これには、対象に対応する構造アイコンの共用体が手がかりとなる。例えば、電話機の中の共用体の一つである受話器には、とり得る状態に対応する要素としてオンフックとオフフックがある。これらの要素を電話機に対するメッセージ名として使うことができる。次に、動作シナリオに現れる対象の状態変化に対応して、対象間のメッセージの流れを定義し、対応するメッセージ名を割当てる(図8)。

本ステップでは、一つのサービスについて、システムの状態空間上で時系列的に発生する相互作用を定義している。

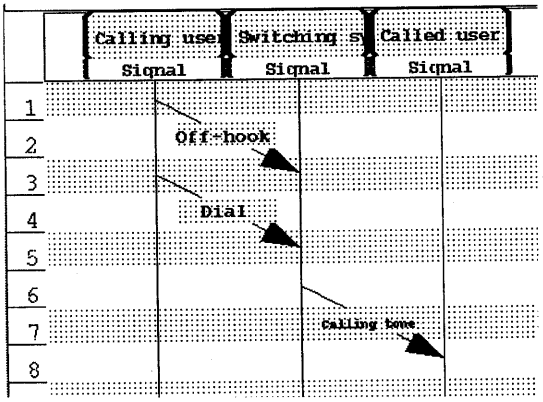


図8: メッセージシーケンスの例

(4) 状態遷移ステップ

本ステップでは、システムを構成するそれぞれの動作対象に対して、インタラクションステップで作成したメッセージ手順をすべて満たす状態遷移図(SDL)を作成する。それぞれの対象に対して、以下のことを行う。(1)対象の現れるメッセージシーケンスを抽出する。(2)抽出したメッセージシーケンスから、その対象の部分(ひとつの対象に着目したメッセージの入出力列)だけをさらに抽出する。(3)抽出した対象のメッセージの入出力列を、もとの対応する状態遷移図に加えていく(最初の状態遷移図は空)。

対象の可能な動作のすべてをメッセージシーケンス(動作例)として定義することは、システムが複雑になると記述量が膨大になり現実的ではない。このため、動作シナリオステップやインタラクションステップでは主要な動作例を数多く作成する。次に、上述の手順で作成した主要な動作を表す状態遷移図に、インタラクションステップでは与えられなかった例外的な処理などを加えて最終的な状態遷移図を定義する(図9)。

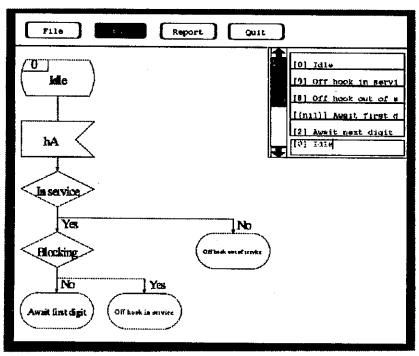


図9: 状態遷移図の例

本ステップでは、システムの各構成要素について、すべてのサービスを満たす状態空間上の可能な時間的な変化を定義する。

本技法では、サービスごとに、動作シナリオとメッセージシーケンスが作成される。このため、必要な対象が前もって与えられていれば、サービスごとに独立して動作シナリオステップとインタラクションステップを行うことができる。さらに、サービスに対するメッセージシーケンスを作成した時点で、そとつど、対象のもつ状態遷移図に関係するメッセージの入出力列を追加することにする。このように設計作業に捉えると、システム全体の設計がサービス機能、動作シナリオ、インタラクション、状態遷移と進むというよりも、システムのもつ一つ一つのサービスごとに動作シナリオ、インタラクション、状態遷移の3つの設計ステップが並行して進むと見なせる(図10)。この場合、状態遷移図は、すべてのサービスを満たす最終的な動作仕様だけでなく、設計途中の部分的な動作仕様であるともいえる。

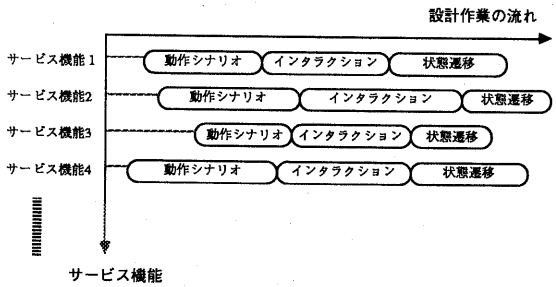


図10: 設計ステップの並列性

5 支援システム

本技法は、支援システムを前提とした技法といえる。例えば、動作シナリオの作成を手作業で行おうとすると、動作シナリオの一つ一つのフレームを設計者が手書きしなければならなくなり、現実的ではない。支援システムは、各設計ステップに対応した支援機能からなっている(図11)[9]。

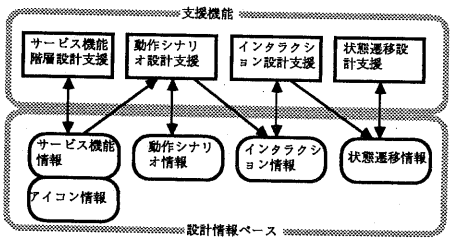


図11: 支援システムの構成

支援機能は、対応する設計ステップについて、(1)編集、(2)確認、(3)変換の3つの面から作業を支援する。編集の支援は、各設計ステップで作成される仕様を容易に作成・修正できるエディタを提供することにより行う。確認の支援として、設計した結果をレビューしたり動作を確認することができるアニメータやシミュレータを提供する。変換の支援は、設計した結果を他の設計ステップで有効に利用するためのものであり、設計仕様の変換機能を提供することにより行う。本支援システムが提供する機能項目を、表1に示す。また、設計ステップの並行作業を支援するために、本支援システムは複数のエディタを同時に表示し編集を行うことができるマルチウィンドウシステム(NeWSシステム)上に開発されている。

表1: 支援システムの機能

作業ステップ	フェーズ	支援内容
サービス機能	編集	サービス階層、対象階層、構造アイコンの編集
動作シナリオ	編集	動作シナリオの編集
	確認	動作シナリオのアニメーション
	変換	動作シナリオ→メッセージシーケンス変換
インタラクション	編集	メッセージシーケンスの編集
	確認	状態遷移シミュレーション
	変換	メッセージシーケンス→状態遷移変換
状態遷移	確認	状態遷移検索、状態遷移図レイアウト

6 考察

まず、2節で挙げた問題点に対して、本技法がどのように対処しているかを考察する。

・サービスと動作仕様との対応について

本技法では、一つのサービスに対する動作を、動作シナリオとメッセージシーケンスにより表している。メッセージシーケンスは、状態遷移の一つの遷移系列に対応付けられる。このため、システムのそれぞれのサービスが、動作仕様である動作シナリオ、メッセージシーケンス、状態遷移とどのように関連しているかを知ることができる。また、状態遷移→メッセージシーケンス(→動作シナリオ)→サービスとたどることができるので、状態遷移(あるいはその一部分)が関係しているサービスを得ることもできる。

・静的仕様と動的仕様とのギャップについて

本技法では、まずシステムのそれぞれのサービスについて動作設計を行い、次にすべてのサービスを満たす動作設計を行うというアプローチをとっている。また、サービスについての動作設計は、動作シナリオとインタラクションの2つの設計ステップからなっている。このように3つの設計ステップを設けているため、段階的に動的な仕様のギャップを埋めながら、設計を進めていくことができる。

・システム状態の構造化・視覚化について

本技法では、対象の構造を論理的・視覚的に表すために構造アイコンが導入された。構造アイコンを用いて記述される動作シナリオでは、システムが関係する内外の対象だけでなく、それぞれの対象の構造が時間的にどのように変化していくのかを知ることができる。構造アイコンの論理的な構造は、動作シナリオからメッセージシーケンスを生成するときにも使われている。

本技法では、システムの状態だけでなく、設計のそれぞれの視点に視覚的な表記法が用意されている。このため、さまざまな視点からシステムの構造や動作を理解することを助けている。

さらに、仕様の記述量、下流工程との関連、本技法に関連する研究について、以下に述べる。

・動作仕様の冗長性

一般に、設計の視点が多くなると、仕様の記述量も増加する。本技法でも、動作シナリオとメッセージシーケンスの導入により、動作仕様の記述量は従来の技法よりも増加する。さらに、動作シナリオとメッセージシーケンスは、状態遷移図の一部分に対応するため、内容的にも冗長している。

このため、支援システムの機能として、動作シナリオ→メッセージシーケンス変換とメッセージシーケンス→状態遷移図変換の2つのツールを用意している。これらのツールにより、設計者の実際の記述量を軽減することができる。

・下流工程との関係

本技法は、主としてリアルタイムソフト設計の上流工程を対象としている。実際の開発は、このあと、動作仕様(状態遷移)の解析や設計の下流工程(プログラムモジュールの設計)が続いている。

状態遷移の解析やプログラム変換などを支援するシステムとしては、SDL支援システム[4][5]、SDE [10]、Escort [11]などが挙げられる。

また、モジュール設計は従来の技法が有効である[7]。

・関連する研究

上述のSDL のほかにESTELLE[12] も、表記法は異なるが、ほぼ同様の概念をベースとしている。このため、SDL と同様の課題を抱えている。

要求定義から設計を対象とした技法としては、ジャクソン開発法がある。ジャクソン開発法は、システムが

関係するイベントの順序構造をとらえてはいるが、リアルタイムソフトウェアのように状態制御を中心としたシステムの動作を表すには不十分である。

リアルタイムシステムを対象とした技法として、リアルタイム構造化分析(例えば、[2])、STATEMATE [14]、ADARTS [15] などがある。これらは、制御データに着目した制御フローと状態遷移モデルの枠組みで動作仕様を捉えている。これらの技法では、本技法のもつ設計視点のサービス階層、動作シナリオ、メッセージシーケンスに対応する概念は含まれてはいない。ただし、STATEMATE の状態遷移モデルを表すstatechart は、対象の内部状態に対してではあるが、状態の構造をand/or結合により構造的に捉えている。

システムのサービスに着目した設計技法としてSAL [10]がある。SAL は、システムの各サービスをマルチログにより表現する。マルチログは、メッセージシーケンスに繰り返しや条件実行などの表現を拡張した概念とみなすことができる。従って、SAL は、インタラクションから状態遷移の設計ステップを支援している。

7 おわりに

以上、リアルタイムソフトウェアの動作仕様を段階的に設計するための技法とその支援システムについて述べた。従来から広く利用されている状態遷移モデルをベースとしているが、要求仕様から状態遷移モデルを構築するまでを段階的にかつ視覚的に支援していることが特徴である。

本技法により、リアルタイムソフトウェア設計の経験が浅い設計者に対しても、段階的に設計するための指針が与えられた。また、本技法で挙げた設計の視点は、すべて視覚的な表記法が用意されている。このため、さまざまな視点から直観的に理解しやすい形式でシステムの仕様をとらえることができるので、設計者自身が設計を行う時ばかりでなく、システムのや他の開発工程の技術者とのコミュニケーションの道具としても有効であると考える。

本研究は1987年末からSNAPSHOT(設計技法+支援システム)として始まり、現在、本論文で述べた設計技法の支援システム(プロトタイプ)を開発している段階である。今後、さまざまなリアルタイムソフトウェアに対して、本技法の適用評価を行っていく予定である。

謝辞

本技法の検討において、有益な御示唆と御討議を頂いた紫合治氏(NECアメリカ)、守屋慎次教授(東京電機大)、田中稔助教授(広島大)、菊野亨助教授(大阪大)、市川晴久氏(NTT)に感謝の意を表します。

参考文献

[1] CCITT SDL Recommendations Z. 100-Z. 104, 1984.

[2] Hatley, D. J. and Pirbhai, I. A. "Strategies for Real-Time System Specification," Dorset House, 1987.(和訳: 立田「リアルタイムシステムの構造化分析」日経BP, 1989).

[3] 川島「交換ソフトウェアの基礎」「交換・通信ソフトウェア仕様記述言語の標準化と技術展望」講習会資料,通信(学)交換研, 1987.

[4] Shigo, O. and Koyamada, M. "Designer's Work Environment for Communications Software," GLOBECOM '85, December 1985.

[5] 小山田「SDLに基づく交換ソフトの設計支援」「交換・通信ソフトウェア仕様記述言語の標準化と技術展望」講習会資料,通信(学)交換研, June 1987.

[6] 小山田、岩戸「リアルタイムソフトウェア向け視覚的設計支援の提案－動作設計技法の概要－」第39回情処全大, October 1989, pp. 1582-1583.

[7] 小山田、岩戸「リアルタイムソフトウェア向け視覚的設計技法とその環境- 動作設計技法-」IPA 技術センター第8回発表会論文集, October 1989, pp.21-29.

[8] 岩戸、小山田「リアルタイムソフトウェア向け視覚的設計支援の提案－支援システムの概要－」第39回情処全大, October 1989, pp. 1580-1581.

[9] 岩戸、小山田「リアルタイムソフトウェア向け視覚的設計技法とその環境- 支援システム-」IPA 技術センター第8回発表会論文集, October 1989, pp.30-38.

[10] Ichikawa, H., Itoh, M. and Shimasaki, M. "Protocol-Oriented Service Specifications and Their Transformation into CCITT Specification and Description Language," Trans. IECE Japan, Vol.69, No.4, April 1986.

[11] Wakahara,Y., Kakuda,Y., Ito,A. and Utsunomiya, E. "Escort: An Environment for Specifiying Communication Requirements," IEEE Software, March 1989.

[12] Budkowski, S. and Dembinski, "An Inroduct ion to Estelle: A Specification Language for Distributed Systems," Computer Networks and ISDN Systems, Vol.14, No.1, 1987.

[13] Jackson, M. A. "System Deveopment," Prentice-Hall, 1983.

[14] Harel,D., Lachover,H., et.al. "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," 10th ICSE, 1988.

[15] Gomaa, H. "Structuring Criteria for Real Time System Design," 11th ICSE, May 1989, pp. 290 - 301.