2008－SE－161 (8)
2008／9／26

組込みシステムの要求・設計アニメーションツール

紫合 治‡, 横山裕介‡, 佐藤潤也‡

組込みシステムの開発では, 開発の初期段階でシステムの要求を正確に把握することが重要になる. 特に, システムに含まれる装置やその装置が制御するガス・溶液等の性質を把握し, それらが全体としてどのように振舞うべきかを理解しなければならない. これらの要求事項は, 問題の要求者とシステムの開発者の間での十分な意思疎通がなされなければならない. 我々は, 問題の要求者の要求を開発者が正確に引き出すためのツールを目的として, システムの要求段階でのアニメーションが有効であるとの前提をもとに, 組込みシステムの要求・設計アニメーションツールを開発した. ここでは, もとの問題に含まれる装置やガス等の状態の変化の様子を動画表示することによって, システムの振舞いをアニメーション表示する方式を紹介する.

Requirement and Design Animator for Embedded System

Osamu Shigo‡, Yusuke Yokoyama‡ and Junya Sato‡

For reliable embedded system development, the requirement should be exactly understood during the early stage of development phases. Especially, the property of the problem domains including equipments and physical objects, like gas and liquid controlled by the equipments, should be grasped before the requirement, and total behaviors of these domains should be expressed as the system requirement. The gap of understanding between problem engineer and system developer becomes a terrible source of unreliable system. We have developed the requirement and design animation tool to narrow this gap. This tool animates the problem domains behavior, including the equipments and gas/liquid behavior, as a comic strip animation.

1. はじめに

近年, ユビキタス社会の進展に伴い, 多くの組込みシステムが開発され, その需要は益々増大しつつある. 組込みシステムのためのソフトウェア開発では, 開発の初期段階でソフトウェアによって制御される外部環境実体を把握することが最も重要になる[1]. そこでは制御される機器の性質や, その機器によって制御される外部実体(ガスや液体等)の性質, さらにその状況をシステムに通知するセンサー装置の性質等も分析する必要がある. これらの個々の装置や実体の把握に加えて, それが統合した場合のシステムとしての性質や振舞いに対する要求分析が重要になる. ここでは, ソフトウェアの構成や機能ではなく, 外部環境が全体としてどのような振舞いをするか, つまり問題そのものの把握が必要であり, それは解としてのソフトウェアの構造や機能とは区別して分析しなければならない[2].

‡東京電機大学 情報環境学部
School of information Environment, Tokyo Denki University

外部環境の装置や実体がどのように振舞うべきかを分析するのは, ソフトウェア技術者の役目というより, 問題提供者がソフトウェア技術者に伝える, または, 問題提供者から引き出すことである. このため, 問題向きの表現が要請される. 近年ソフトウェアの表現としてUML[3]が広く使われているが, UMLはソフトウェアの構造や振舞いについての記法であり, コンピュータやソフトウェアの技術者向けであって, 問題提供者向けとはいえない. このため, 要求分析の段階での問題提供者とソフトウェア技術者の思いのずれが要求定義の誤りを起こすことも多い. ここでは, 問題提供者の視点に立っての組込みシステムの振舞い(外部環境の装置や物理実体の振舞い)をアニメーションによって表現することで, 初期の要求分析での誤りを減らすことを目的としたツール[4]について紹介する.

以下に, 2 節で組込みシステムの振舞いの表現について, 3 節でアニメーションによる振舞い表現について考察し, 4 節でアニメーションのための基本モデルを定める. 5 節では, 4 節のモデルをもとに開発したアニメ

ーション作成システムを説明する．6節ではその利用例と考察を述べ，最後に 7 節で，まとめと今後の課題について述べる．

2. 組込みシステムの振舞い表現

ソフトウェアの振舞いを図形によって分かりやすく表現することを目的として，UMLが広く使われている．UMLは，システムの構造や振舞いを図形表現するための多くの図式表現からなる．振る舞いについては，アクティビティ図，ステート図（UML 2.0 では状態マシン），シーケンス図などが良く使われる．これらは，いずれもシステムの動作をイベントの発生や処理の起動ととらえ，それらがどのような順序で発生していくかを規定したものである．ステート図は，システムの状態を表わすことができるが，そこに記述されるのは，ある状態であるイベントが発生したとき，どのようなアクションを実行すべきかが記述される．振舞いが複雑な場合は，これらの図式をみてその振舞いに誤りはないかとか，抜けがないか等を判定することは難しい．これは，一般にUMLはソフトウェア技術者を対象としているのに，システムの振舞いの誤りやぬけを判断する問題分野の人は，コンピュータやソフトウェアの技術者ではない場合が多いためでもある．

組込みシステムの問題分野の技術者が，システムの振舞いを説明することを目的とする場合，UMLやその関連ツールにはない，もっと問題分野の人にとって分かり易い表現でなければならない．もちろん，その説明は，システムの開発者に対してなされるため，コンピュータやソフトウェアの技術者にとっても十分理解しやすいことが求められる．実際，多くの場合，これらの振舞いの説明は開発者（ソフトウェア技術者等）が要求者に「こんな動きでいいですか？」と詳細な要求事項を聞き出すツールとして使われることが多い．

問題分野の人向けの説明では，その問題に出てくる部品や装置の模型図をもとに，それぞれの部品や装置，さらにガスや溶液がどのように変化していくかを述べると分かり易い．これは，システムに含まれる装置やガス・溶液等が登場人物となる動画で，それらが時間と共に変化していく様子を示す方式が考えられる．但し，一般のアニメ動画とは違い，登場人物である部品や装置は，通常決まりきった変化しか起さない．例えば，給水弁はOFFかON，モータは停止か回転(回転速度が何段階かに変化することもある)等，事前に決められた状態の変化だけを考えるものとする．個別の部品は，状態変化も比較的単純であるが，多くの部品を含む装置やシステム全体の動きは複雑になる．ここでは，システムに含まれる部品や装置の状態が，全体としてどのように変化していくかをアニメーションする方式について述べる．なお，このアニメーションに登場するのは，問題を表現するための部品や装置であって，それを制御するためのコンピュータやソフトウェアではない．

3. アニメーションによる振舞いの表現

ここでは，組込みシステムのモデルとして Jackson の問題フレーム[2]の振舞いフレームを考える．振舞いフレームには，「必要とされる振舞い」と「命令された振舞い」があり，後者は前者に「操作者」を加えたモデルである．図1に「命令された振舞い」のフレーム図を示す．

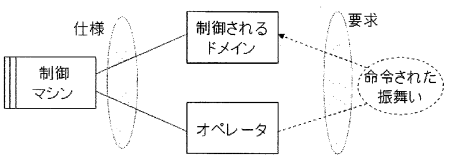

図1　命令された振舞いフレーム

図1で，「制御されるドメイン」が，組込みシステムのアニメーションの主な登場人物になる．このドメインは，マシンから見ると，制御命令となるイベントがインタフェースになっているが，要求から見ると，ドメインの状態が参照される[5]．アニメーションでは，要求者の視点を重視するので，マシンから見たインタフェースである命令やイベントは無視し，要求から見たドメインの状態のみをとらえることとする．

振舞いフレームの要求としては，システムに含まれる部品や装置ドメインの状態が，時間の経過や操作者のコマンドによってどのように変化していくかが示される．これは，部品や装置の状態をまとめたシステム状態の変化の様子を規定した状態マシンで示されることが多い．アニメーションは，この状態マシンの記述をもとに，操作者のコマンドや時間経過による状態遷移によって生ずる部品や装置の状態の変化を，絵で示していくことによって実現される．

ここで，各部品や装置には，その状態毎に独自の絵が定義されており，その部品や装置の状態変化により絵が変化することで，動きが表わされる．部品によっては，状態の絵自身が動画になることもある．例えば，ファンの回転状態は，羽がまわる様子を示す動画(いくつかの羽の絵が 0.2 秒程度の間隔で繰返すとまわっているようにみえる)をもつ．さらに，アニメーションの動きに

合わせた音を規定すると，より分かりやすくなる．音の規定は，ドアの開閉音のように状態の変化に伴って1回だけ発生するものと，ファンの回転音のように状態にいる間連続して発生するものがある．

アニメーションの原稿となる規定は，システム状態の中に部品や装置の状態を規定した状態マシンで記述するが，これは，状態指向の状態マシン[6][7]の概念に対応する．状態指向の状態マシンは通常の状態マシンのように状態遷移に伴うアクションを記述する代わりに，各状態で環境がどうなっているかを記述するもので，振舞いフレームの要求表現等により適した状態マシンといえる．図2に，部品状態を絵で表わした，状態指向の状態マシンの例を示す．

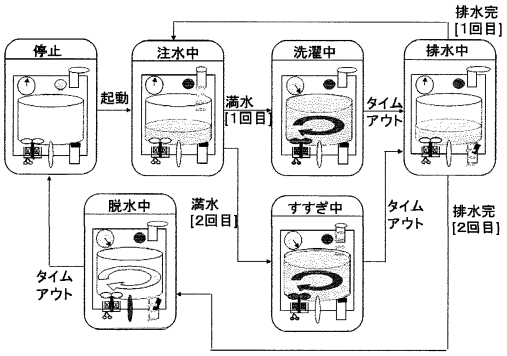

図2　状態指向の状態マシン(洗濯機)

4. アニメーションシステムのモデル

システムはいろいろなドメインを含む．ここでは最終的な設計の対象となる制御マシン(組込みソフトを含むコンピュータ)そのものは含めない．システムは，ドメインの集まりからなる．ドメインは，さらに詳細な要素からなる装置と，それ以上分割できない部品に分けられる．システム全体も，1つの装置とする．

4.1. 部品

部品(部品クラス)は，1つ以上の状態をもつ．各状態は，画像と音を持つことが出来る．画像は，1枚以上の絵と，繰返しの時間間隔の指定からなる．絵が1枚だけの場合は静止画状態を表わし，2枚以上の場合は動画状態を表わす．例えば，ファンは，停止と回転の2状態をもち，停止状態は1枚の絵からなる静止画を，回転状態は数枚の絵を繰り返す動画と連続した回転音を持つ．各状態に表れる絵は，サイズや位置が相対的に合っていなければならない．例えばファンの羽は，停止時と回転時では大きさは同じで，中心位置が一致していなければならない．

システムの背景や，装置の枠組み等，アニメーションで変化しない画像は，状態を1つだけ持つ部品とする．このような部品は，システムの振舞いには全く関係しないが，アニメーションの絵の構成にのみ必要になる．つまり，システムの要求分析では無視されるが，アニメーションでのみ必要になる．

部品の状態に，その状態のときに発生しうるイベントのリストを規定することができる．イベントは，ここでは操作者が介在するような動作で，例えば電源スイッチのON／OFF，電話機のダイアル，水道栓の開閉などがイベントになる．イベントの出現は部品の状態に規定されることに注意されたい．例えば，電源スイッチ部品はOFFとONの状態をもつが，OFF状態での可能なイベントはON，ON状態での可能なイベントはOFFになる．人間が介在しないイベントはここでは記述しない．例えば，給湯器のガス弁はシステムが開閉するので，ここでのイベントにはならない．

1つの部品情報は，まとめると以下のようになる．

部品　:=　部品名　{状態}*
状態　:=　状態名　画像　[音]　[イベントリスト]
画像　:=　静止画｜動画
静止画　:=　絵｜空
動画　:=　{絵}*　間隔時間
音　　:=　音源　間隔時間
イベントリスト　:=　{イベント名}*

4.2. 装置

装置は，いくつかの要素の組合せで表わされる．ここで要素とは部品かまたは別の装置である．装置は要素のレイアウトを規定する．ここでレイアウトは，要素毎にサイズ(矩形で示す)と位置(矩形の左上の位置)，さらに重なった場合の前後関係を規定する．要素は，要素名と部品名または装置名をもつ．装置は，同じ部品(部品クラス)または装置(装置クラス)を一つ以上含むことができる．装置もいくつかの状態をもつ．装置の状態は，含まれる要素の状態を規定することによって決まる．

ここではシステム全体も1つの装置と考える．例えば，洗濯機は，給水栓，排水栓，洗濯モータ，脱水モータ，洗濯槽，水，水位センサー，起動ボタン等からなる．このうち洗濯槽は状態を1つしか持たない，分析には必要のない部品である．さらに，電話交換システムのモデ

ルの場合は，発呼電話，着呼電話，回線，課金メータなどを含む装置と考える．ここで，発呼電話も，さらに分解すると，送受話機，ダイアル，トーン発生機などからなる装置となる．以上，まとめると，装置は以下のようになる．

装置 ::= 装置名 {要素}* レイアウト
{装置状態}*
要素 ::= 要素名 要素クラス名
要素クラス名 ::= 部品名 | 装置名
レイアウト ::= 装置サイズ {要素レイアウト}*
要素レイアウト ::= 要素名 要素サイズ 位置
装置状態 ::= 状態名 {要素名 状態名}*

4.3. 状態マシン

システムの振舞いは，システムの状態の変化の様子を規定する状態マシンによって表わされる．状態は前述したシステムの状態のインスタンスとする．つまり，状態マシンには，システム(装置)の状態が同じものが 1 つ以上現れることができる．例えば，洗濯機では，洗濯前の給水状態とすすぎ前の給水状態は，装置の状態(つまり，アニメーションに現れる部品達の状態)は同じでも，状態マシンとしては別の状態をとる．

状態の遷移は，システムの状態に含まれる部品状態で発生しうるイベント(部品状態のイベントリストに含まれるイベント)による遷移か，時間による遷移のいずれかとする．イベントによる遷移は，人間等が介在するイベントで遷移するもので，アニメーション実行時にはユーザが明示的にイベントを指示することによって遷移する．時間による遷移は，最終的なシステムのタイマーによるタイムアウトを表わす場合と，物理実体の時間による状態の変化を表わす場合に使う．洗濯機の例では，洗濯状態から排水状態に移るのは前者のタイマーによる時間遷移で，排水状態から次のすすぎのための給水状態に移るのは，水槽の水の排水による変化(満水から，未満水になり，さらに空になる変化)にかかる時間となり，後者の物理実体の状態の変化を表わす．

状態マシン ::= {状態}* 初期状態 遷移
状態 ::= 状態名 状態クラス名
遷移 ::= 開始状態 遷移条件 終了状態
遷移条件 ::= 時間遷移 | イベント遷移
時間遷移 ::= 数値 (秒)
イベント遷移 ::= 部品名．イベント名

5. アニメーション支援システム

5.1. システム構成

図3にシステムの構成を示す．部品作成エディタは，部品の状態の絵，または動画(2枚以上の絵の繰り返し表示)，音，さらに発生可能なイベント(操作命令等)を登録していく．装置作成エディタは，作成済の部品や装置を組合せて，新たな装置を作成する．装置の状態は，含まれる部品や装置の状態の組合せによって規定する．システム全体に対応する装置に対して，状態遷移作成エディタで状態マシンを作成する．ここでは、装置作成で登録したシステムの状態と，部品作成で登録した可能なイベントや時間遷移を使いながら状態マシンを作成する．出来上がった部品，装置，状態マシンから，アニメータによってアニメーションを表示する．

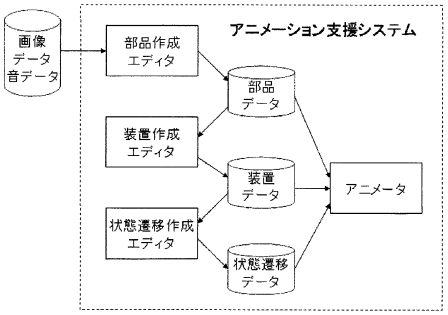

図3 アニメーションシステムの構成

5.2. 部品登録

最初に部品の状態の絵を既存の描画ツール(ペイントやパワーポイント等)によって作成し，画像ファイル(bmp, jpg, gif 等)として作っておく．

部品作成ツールは図4のような画面になる．まず部

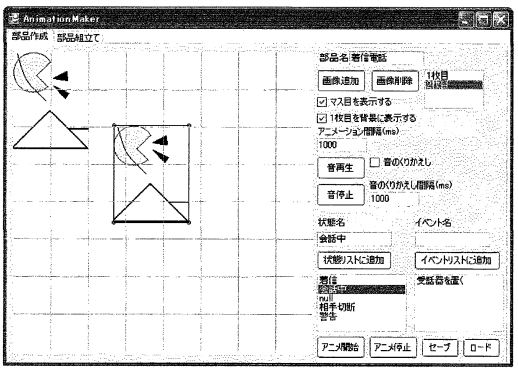

図4 部品作成画面

品名を記述する. 新規部品なら状態リストやイベントリストは空になり, 登録済部品なら定義された状態リストが表示される. 新規の場合, 状態名を記述し画像追加ボタンを押し, 登録済の画像ファイルを選択することで, 状態表示画面に画像が表示される. このとき, 透過色(マゼンダ)は白に変っている. 状態表示画面では, 選択された部品の拡大縮小や移動を必要に応じて行う. 動画状態にする場合は, 続けて2枚目以降の画像を登録すればよい. このとき, 1枚目の画像を背景に表示して, 2枚目以降を重ねながら画像の大きさや位置を調整し, さらにアニメーションの時間間隔を指定する. 音再生ボタンを押すと, 状態に音を登録できる. 音源は, 前もって準備しておく. 連続音に対しては, 音の繰り返しを指定する. 部品の状態に対して, イベント名を記述し, イベントリストに追加することができる. このイベント名は, 状態マシン登録の際, 状態遷移の条件として使われる.

5.3. 装置登録

装置の作成は部品組立て画面によって行う. 図5に装置作成画面を示す. 部品読込ボタンを押して, 既に作成した部品・装置の一覧を表示させる. 装置の登録では, まず登録すべき装置名を記述し, 部品・装置の一覧から, 含むべき要素を選択して追加ボタンを押すことにより, 要素が追加され, 装置画面に表示される. 追加された要素の, 拡大・縮小, 移動を行い適切な場所に配置することで装置を作成する. さらに, 追加した要素が, 他の要素と重なる場合, 前後関係を調整して表示することができる.

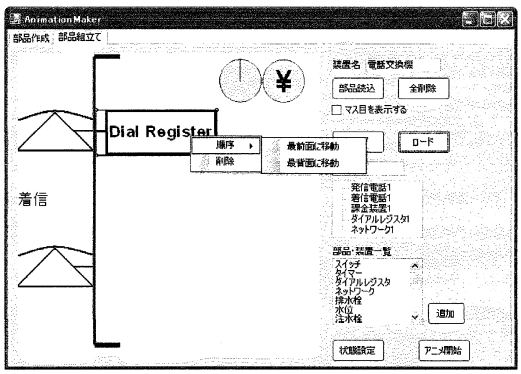

図5 装置作成画面

要素の追加では, 一覧の要素名(クラス名)と追加要素名(インスタンス名)を区別するために, 追加要素名にはもとの要素名に数値(1から順)が付加される. これは, 上部のテキストボックスで自由に名前を変更することも出来る.

全ての含まれる要素の追加と, それらの配置が終わったら, 状態設定ボタンを押して, 図6のような装置の状態設定画面によって状態設定を行う. 状態設定画面では, 追加した要素の部品や装置が部品リストに表示される. それから1つ選択すると, その部品・装置の状態が部品状態に表示される.

装置状態の登録では, まず, 装置の状態名を記述し, 全ての部品リストの要素に対して, その装置状態のときの要素の状態を選択し, 登録ボタンを押す. よく似た装置状態を登録する場合は, 装置の状態リストから登録済の状態を選択したあと, 異なる部品の状態のみ再指定し, 状態名を新たに記述して登録するとよい. なお, 部品状態を変更すると, 図5の装置作成画面にも部品の状態の絵が反映されるので, アニメーションの画面の様子を見ながら装置状態を作成できる.

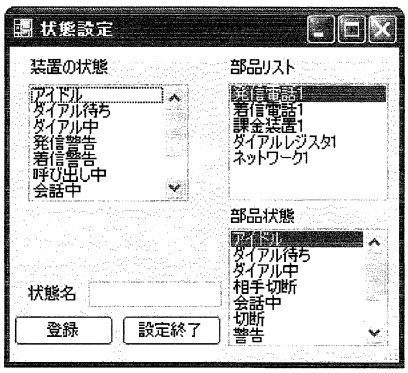

図6 装置の状態設定画面

5.4. 状態マシン登録

システム全体に対応する装置の情報をもとに, 状態マシンを作成する. 図7に状態マシンの作成画面例を示す. この画面は, 左に装置の状態の絵を, 右に状態遷移図を示す. 登録した装置名を選択し, 状態遷移編集モードにすることで状態マシンを作成する. 状態遷移画面の下に, その装置の状態と, その状態で含まれる部品のイベント一覧, さらにタイマー規定が表示されている.

状態を選択して, 状態遷移画面に状態(円)を登録できる. このとき, 状態遷移画面の状態名を変更したい場合は上部のテキストボックスで記述することができる.

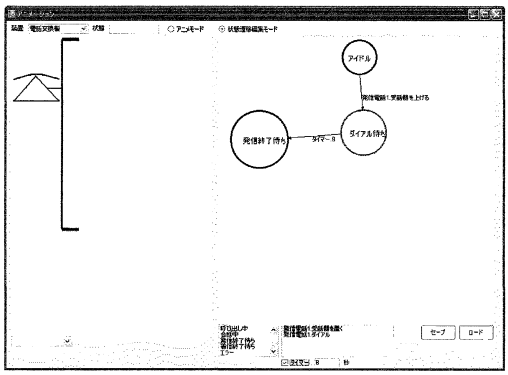

図7 状態遷移図作成画面

状態の遷移は，登録された開始状態をマウスで選択し，そのときのイベントリストからイベントを選択するか，またはタイマー選択と時間(秒)を記述した後，終了状態をマウスで選択することによって，遷移の矢印が表示される．遷移には，部品名.イベント名か，タイマー.秒数が表示される．

なお，遷移矢印は，中間点をつけて曲線にすることで，線の交差を分かりやすくすることが出来る．図8にこのようにしてできた，電話交換機の状態遷移図の例を示す．

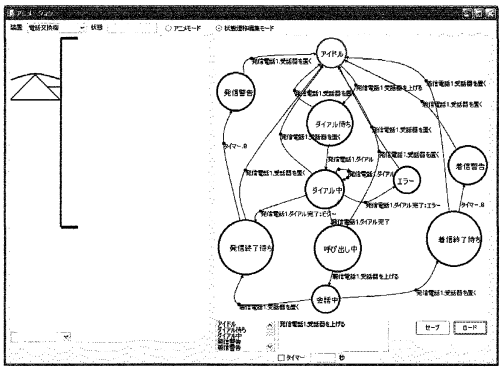

図8 電話交換機の状態遷移図完成例

5.5. アニメーションの実行

状態遷移図を作成した後，画面をアニメモードにするとアニメーションができる．アニメモードでは，最初に装置の初期状態の表示がなされる．そこで，装置状態画面の部品の絵を右クリックすると，その部品の現在の状態で可能なイベントが表示される．そのイベントを選択するか，またはタイマー指定での時間経過によって，状態遷移が起き，それに連動して，装置の部品の絵や動画の状況が変化し，指定された音が出る．

状態遷移で，現在の状態からタイマー遷移とイベント遷移の両方が定義されている場合，アニメーション画面の右下に残りの秒数が表示されるので，その時間内にイベントを選択すればイベント遷移になり，何もしないで残りの時間が0になるとタイマー遷移を起す．

図9に電話交換機の例では，呼出中状態から，着信側電話機の受話器を上げるイベントを選ぼうとしている様子を示す．なお，この状態では，呼出の電話機を右クリックすると，受話器を置くというイベントが選択できる．

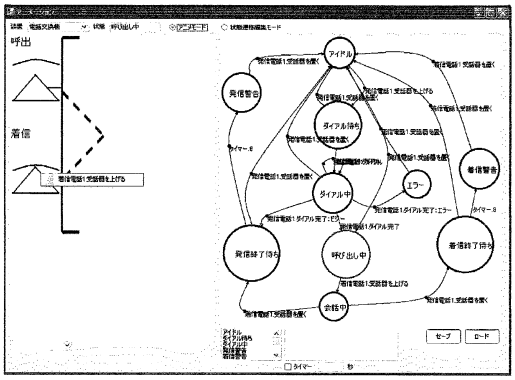

図9 電話交換機のアニメーション例

6. 利用例

ここで述べたシステムは，まだツールを試作したという段階にあり，実用での評価までは至っていない．簡単な利用例としては，図2に掲げたような洗濯機と，5で述べた電話交換機がある．一般にこのようなツールの問題点は，絵を描く手間がかかる事にある．実際には，パワーポイントを使ってシステムの全体像を書き，その部品毎に，状態絵を追加し，部品の状態毎の絵をコピーして，画像ファイルを作成した．音については，ネット上の素材や，適当な録音を使ったが，慣れないと結構な手間がかかった．洗濯機，電話交換機共に，大体1人で1日程度かかっている．画像ファイルができたあとは，それを組合せて，部品や装置を登録していくが，ここでは画像の配置のみで，サイズ変更はあまりやらないようにしたので，比較的簡単にできた．状態遷移も，画面が小さいのでレイアウトが大変だったことを除くと(始めからレイアウトを想定して作った)，比較的簡単に

できた．大体，素材の絵の作成と同じ程度の時間がかかった．

実際には最初から完全なアニメーションができるわけではなく，抜けや誤りを修正しながら作成することになる．ただ，抜けや誤りがあってもアニメーションとしてはそれなりに動作するので，誤りも分かりやすく，修正もしやすかった．また，要求の変更等についても，例えば，電話交換機で，アイドル状態から受話器を上げると直接呼出中状態に遷移するように変更して，すぐにアニメモードに戻って実行すると，ダイレクトコールの様子がよく分かる．これらは，特に開発者が要求者との打合せで，「ここはこうなるべき」とか，「これはこう変えて欲しい」等の話合いを直接アニメーションに反映しながら進めていく場合に効果がありそうだと思えた．

より本格的な利用例として，図 10 に示すガス給湯器システム[8]がある．これはガス給湯器の保守修理の教育用に開発されたもので，その振舞いを模型図でアニメ表示する．さらに，部品が故障した場合の動作も見ることができ，故障修理の教育にも使える．図で，左側の温度設定や運転スイッチ等はプログラムでハードコード化され，その他の給湯器の模型画部分が，ここで述べたアニメーション方式で作成された．但し，状態マシンの代りに，通常のフローチャートが使われた（ガス給湯器の仕様がフローチャートで書かれていたため）．

このシステムは当初は専用システムとして全てがハードコード化された方式で Visual C#によって開発されたが，その後，ガス会社の技術者が画像や振舞いの修正ができるように本論文で説明した方式を取り入れた．この程度の複雑さだと，画像の作成だけで 1 人で 1～2 週間かかった．それでも，ガス会社の技術者が直接システムを修正保守することができるようになり，当初の目的を果せた．この場合は，既存のシステムのアニメーションを作成するということで本論文の目的である要求の獲得とは異なるが，要求の獲得を目的とする場合は，ここまで詳細な絵は必要ない．

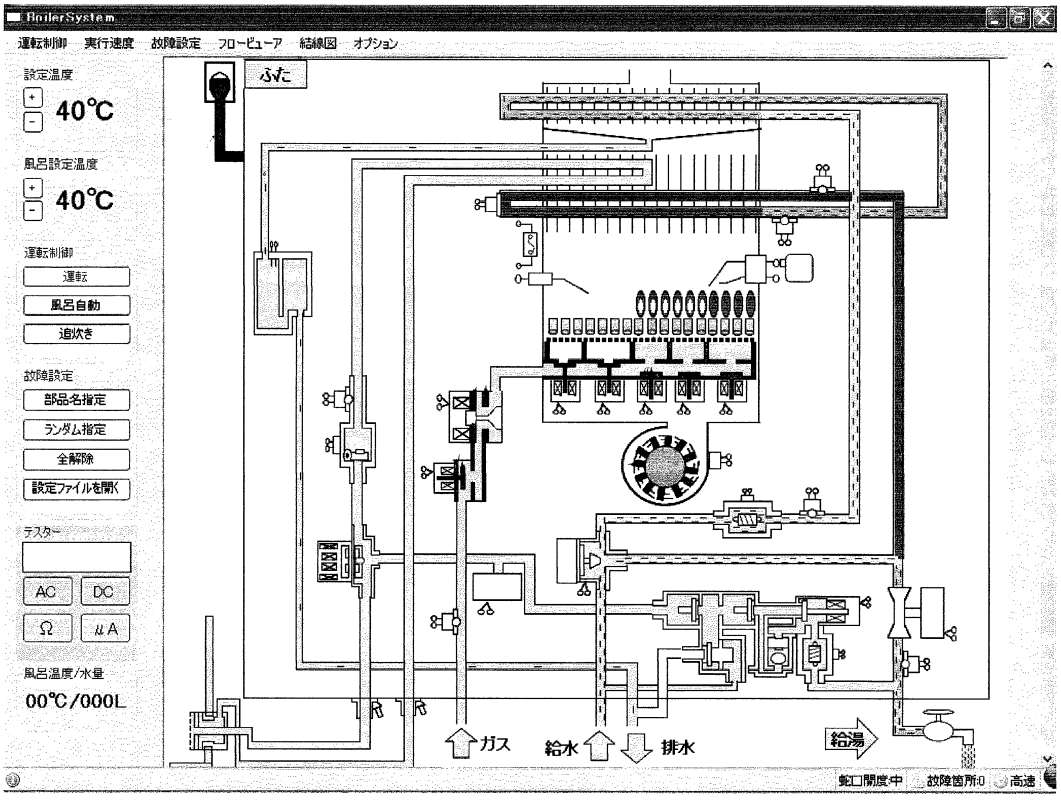

図 10 ガス給湯器システムの画面例

7. おわりに

組込みシステムの要求分析のために，要求者と開発者がシステムの振る舞いについて共通の理解を深めていくことを目的としたアニメーション作成支援システムについて述べた．このシステムは，組込みシステムに含まれる装置やガス・溶液など物理実体の状態の変化の様子を，システム全体の振舞いととらえ，模型図のアニメーション動画として表示するものである．電源スイッチや操作スイッチ等，オペレータによる操作を必要とするものについては，アニメーション中にその部品をクリックして，必要な操作を指示することができる．このように，ここでのアニメーションは，もとの問題で定義されたドメインの様子のみを見せ，解となるマシンやソフトウェアの構造等は無視している．これによって，問題提供者向きのツールになっている．

簡単な組込みシステムでの利用では，アニメーションの実際の有効性までは確認できなかったが，問題分野にあまり詳しくない学生にも，その振舞いを正しく理解させるのに大きな効果があった．より詳細な，実用的な利用では，アニメーションのもとになる画像の作成の手間がかかるが，その分要求者の意図を正確に表現できる．特に，部品が故障した場合については，アニメーション上で部品の故障時の動作を確認しながら，最終的なシステムに対する要求を確立するのに役立つものと思える．

システムの実用化に向けては，いくつかの課題がある．状態遷移では，ここではイベントかタイマー指定しか使えないが，さらに条件指定の追加や，条件で使われる簡単な変数の導入等が必要になろう．また，水槽の水の増加を示すには，水量を表わす数値変数とその数値による画像変更の機能も必要になろう．部品を故意に故障させ，例えばガス弁が閉じたままになった場合，システムがどのような振舞いをするかをアニメーションで自動表示する機能も追加したい．この場合，ガス等の物理実体の状態変化は，関連部品(ガス弁等)の状態によって決まるような規定が必要になる(6で述べたガス給湯器は物理実体の状態は関連部品の状態で決まるようになっている)．但し，このような機能を増やすにつれてシステム利用の複雑さが増えるので，さらに実際的なシステムの要求分析の利用による評価を通して，必要な機能追加を検討していきたい．

このシステムは連続的な変化を伴うシステムには利用できない．そこでは，物理実体の変化の様子を微分方程式等で分析すること等が必要になろう．ここで扱うのは，比較的簡単な離散系のシステムの分析であり，分析自体が複雑で難しいというのではなく，むしろ要求者の意図が開発者に十分伝わりきれない場合の支援を目的とした．このようなシステムは，顧客の要求を引き出すためのツールとして，有効であると考える．

今後は，実用的なシステムへの利用を通して，ここでのアニメーション方式の有効性を評価し，必要な改善を進めていきたい．

参考文献

[1] 紫合治:「組込みソフト設計のための物理実体(間接オブジェクト)の分析，情報処理学会 SIGSE ウィンターワークショップ 2006.

[2] M. Jackson, Problem Frames: Analyzing & Structuring Software Development Problems, Addison-Wesley Publishing Company, 2001.(榊原彰監訳，プロブレムフレーム:ソフトウェア開発問題の分析と構造化，翔泳社，2006 年)．

[3] James Rumbaugh 他著，石塚圭樹訳:UML リファレンスマニュアル，ピアソンエデュケーション，2002.

[4] 横山裕介，佐藤潤也，紫合治:「設計アニメーション支援ツール」，情報処理学会第 70 回全国大会，2008.

[5] 紫合治:「ジャクソン法(JSP)による状態遷移設計」，ソフトウェアエンジニアリングシンポジウム 2008.

[6] 紫合治:「状態指向のステートチャート」，ソフトウェア工学の基礎ワークショップ(FOSE05)，2005.

[7] 紫合治:「状態指向の状態遷移表」，情報処理学会ソフトウェア工学研究報告 No.157, 2007.

[8] 由井秀昌，野口貴大，紫合治，松本二郎，高山利美，金子通明:「アニメータ自動生成ツールによるバーチャル・ガス給湯器の構築」，情報処理学会第 69 回全国大会，2007.