エレベーター制御用ソフトウェアの自動生成*

5J－3

田村 一賢，猪野 仁，尾関 彰宏，皆川 真紀，加地 浩一†
株式会社 東芝 研究開発センター システム・ソフトウェア生産技術研究所‡

1 はじめに

近年、ソフトウェアの大規模化、複雑化、および短ライフサイクル化に伴う開発・保守の問題に対して、ソフトウェアの仕様を形式的に記述し、プログラム作成を自動化するアプローチが盛んになっている。

特に制御用ソフトウェアにおいては、状態遷移モデルをベースとした仕様化技術が用いられるようになってきており、比較的小規模な仕様を持つ電子レンジや冷蔵庫等のマイコンソフトウェアの分野に実適用可能となっている [1][2]。

本稿では、多くの運転モードを持ち、運転モード間の関係が複雑で、比較的規模の大きいという特徴を持った分野にこのようなアプローチを適用するための方法を、エレベーターのシーケンス制御ソフトウェアを例にとって報告する。

2 状態遷移モデル適用における問題

上記のような特徴を持った対象に一般の状態遷移モデルを適用しようとした場合、運転モードが多くなるに従い、運転モード間の切り替えの仕様が複雑になるため、次のような問題が生じてくる。

- 運転モード内と運転モード間の遷移の区別が困難になり、個々の運転モードの流れが見えにくい
- どの運転モードを優先的に実行するかという運転間の優先関係がわかりにくい
- 運転モードのあらゆる状態から、他の運転モードのあらゆる状態への遷移の可能性を記述する必要があり、記述が繁雑になる

3 仕様記述モデル

これらの問題を解決するため、状態遷移モデルを用いた仕様記述モデルを次のように拡張することが必要になる。

- 個々の運転モードの流れの見通しを良くするため、運転モード毎に切り分けて記述を行なう
- 運転モードの記述と、優先関係に従って選択と切り替えの処理を行なう運転管理の記述を切り分ける

この方針に従い、仕様記述モデルは図1のような構成とし、仕様記述言語とその実行メカニズムを考える。

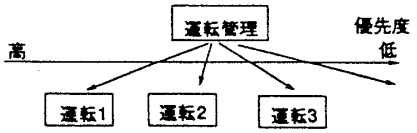

図 1: 仕様記述モデルの構成

ただし、制御用ソフトウェアではリアルタイム処理上の問題から、メモリ使用量と単位時間内での処理量に制約が課されており、これらを考慮する必要がある。

3.1 仕様記述言語

上記モデルの実現において、運転管理に対しては、運転の優先度と開始条件を示すために表形式の記述を用い、運転に対しては、個々の運転モードの動作の流れと、動作の連続性を表現できるようにするため、状態遷移モデルを拡張した記述を用いる。

優先度表

運転間の優先関係と切り替えのための条件を示す。運転の優先度順に運転名と運転の開始条件を記述する表形式の記述。

高 ↕ 低

運転仕様名	開始条件
地震時管制運転	地震検知器ON
火災時管制運転	火災検知器ON
平常運転	–

図 2: 優先度表の例

運転記述

運転を記述するための仕様記述で、状態遷移モデルに実装上の制約を考慮した記述。

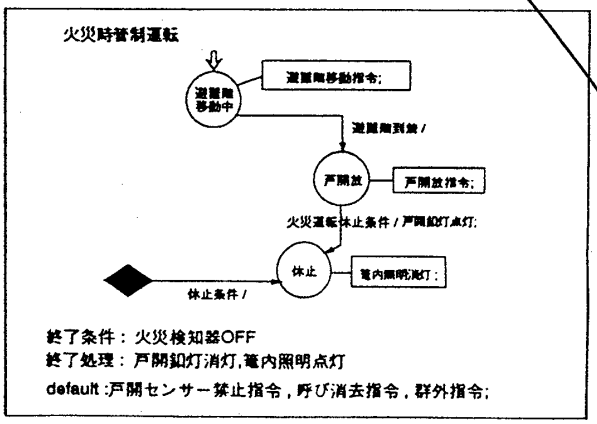

図 3: 運転記述の例

*Automatic Programming for Elevator Control Software

†Kazuyoshi TAMURA, Masashi INO, Akihiro OZEKI,Maki MINAGAWA, Koichi KAJI
‡R & D Center, Systems & Software Engineering lab., Toshiba Corp.

運転記述の状態遷移図については、以下のように拡張する。

- メモリ容量を圧縮するために、ある運転の全ての状態を通じて共通に実行する必要のある処理を表す default 処理記述を設ける
- 運転の任意の状態で切り替えが生じても、当該運転を正常に終了できるようにするための終了条件記述と終了処理記述を設ける
- 運転記述では、運転を切り分けて表現するために、運転が切り替わる時の遷移と処理を直接表現することができなくなる。そこで、運転が切り替わる時に、どの状態に遷移するかを表現する振り分け記述を設ける

3.2 仕様記述モデルのメカニズム

それぞれの仕様記述言語は、CPU 負荷率の軽減という実装上の制約を満たすため、条件の評価と処理の実行が最も少くなる次に示す動作メカニズムで解釈する。

優先度表

step1: 優先度の高い順に開始条件を評価する。

step2: 開始条件が成立した場合には該当する運転を起動し優先度表の評価を終了する。

step3: 開始条件が不成立の場合には下位の運転の開始条件を評価する。

step4: step2 へ。

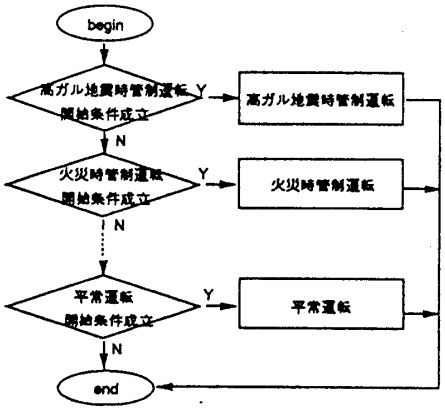

図 4: 優先度表の動作メカニズム

運転記述

step1: 終了条件の評価を行ない、満たしている場合には終了処理を実行して終了する。

step2: default 処理の実行する。

step3: 今まで運転が動作していなかった場合には、振り分け記述からの遷移のイベント評価を行ない、成立している遷移を実行する。不成立の場合には初期状態に遷移する。

step4: 状態における遷移の評価を行なう。現在の状態から出ている遷移でのイベントを評価し、成立している遷移のアクションを実行、遷移先へ遷移する。

ただし、状態遷移図の評価をタイムスライス方式で行ない、その 1 サイクルに 1 つの遷移までのみ実行することにしている。

4 適用実験による評価

この仕様記述言語を用いて実際のエレベーター制御ソフトウェアの仕様を記述し、プログラムを自動生成することによって、本方式の評価を行なった。

その結果、運転モード間の関係は優先度表、振り分け記述、終了処理記述の 3 つで表現することができ、個々の運転モードが独立に記述可能であることが示せた。

またエレベーターにおける

- CPU 負荷率を従来方式と同程度に抑える
- メモリ使用量を従来方式の 1.5 倍以内に抑える

という実装上の制約に対して、表 1に示すようにこの制約を満足する値が得られ、我々の提案する方法を実適用できることが確認できた。

表 1: 評価結果

評価項目	従来方式との比較
オブジェクトサイズ	1.30 倍
CPU 負荷率	1.03 倍

5 おわりに

今後更にソフトウェアの規模が大きくなることが予想されるため、最適化によるオブジェクトサイズの圧縮や、CPU 負荷率の軽減等を検討し、実製番への適用を行なっていく。

また、形式的に仕様を記述したことにより、仕様レベルでの検証やレビューが可能になったので、仕様検証系などによる早い段階での品質作り込みによる生産性の向上を目指す。

参考文献

[1] 岸他, “状態遷移モデルに基づくプログラム部品合成システムの開発”, 情報処理学会研究報告 91-SE-80, 1991.

[2] 清水他, “金融機器組み込みソフト向け CASE”, 情報処理学会ソフトウェア工学研究報告 93-SE-91, 1993.

エレベーター制御用ソフトウェアの自動生成*

6J－3

田村 一賢, 猪野 仁, 尾関 彰宏, 皆川 真紀, 加地 浩一†
株式会社 東芝 研究開発センター システム・ソフトウェア生産技術研究所‡

1 はじめに

近年、ソフトウェアの大規模化、複雑化、および短ライフサイクル化に伴う開発・保守の問題に対して、ソフトウェアの仕様を形式的に記述し、プログラム作成を自動化するアプローチが盛んになっている。

特に制御用ソフトウェアにおいては、状態遷移モデルをベースとした仕様化技術が用いられるようになってきており、比較的小規模な仕様を持つ電子レンジや冷蔵庫等のマイコンソフトウェアの分野に実適用可能となっている[1][2]。

本稿では、多くの運転モードを持ち、運転モード間の関係が複雑で、比較的規模の大きいという特徴を持った分野にこのようなアプローチを適用するための方法を、エレベーターのシーケンス制御ソフトウェアを例にとって報告する。

2 状態遷移モデル適用における問題

上記のような特徴を持った対象に一般の状態遷移モデルを適用しようとした場合、運転モードが多くなるに従い、運転モード間の切り替えの仕様が複雑になるため、次のような問題が生じてくる。

- 運転モード内と運転モード間の遷移の区別が困難になり、個々の運転モードの流れが見えにくい
- どの運転モードを優先的に実行するかという運転間の優先関係がわかりにくい
- 運転モードのあらゆる状態から、他の運転モードのあらゆる状態への遷移の可能性を記述する必要があり、記述が繁雑になる

3 仕様記述モデル

これらの問題を解決するため、状態遷移モデルを用いた仕様記述モデルを次のように拡張することが必要になる。

- 個々の運転モードの流れの見通しを良くするため、運転モード毎に切り分けて記述を行なう
- 運転モードの記述と、優先関係に従って選択と切り替えの処理を行なう運転管理の記述を切り分ける

この方針に従い、仕様記述モデルは図1のような構成とし、仕様記述言語とその実行メカニズムを考える。

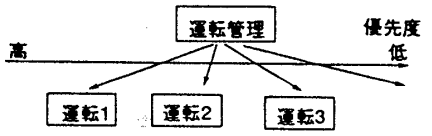

図 1: 仕様記述モデルの構成

ただし、制御用ソフトウェアではリアルタイム処理上の問題から、メモリ使用量と単位時間内での処理量に制約が課されており、これらを考慮する必要がある。

3.1 仕様記述言語

上記モデルの実現において、運転管理に対しては、運転の優先度と開始条件を示すために表形式の記述を用い、運転に対しては、個々の運転モードの動作の流れと、動作の連続性を表現できるようにするため、状態遷移モデルを拡張した記述を用いる。

優先度表

運転間の優先関係と切り替えのための条件を示す。運転の優先度順に運転名と運転の開始条件を記述する表形式の記述。

高 ↑ 低 ↓

運転仕様名	開始条件
地震時管制運転	地震検知器ON
火災時管制運転	火災検知器ON
平常運転	-

図 2: 優先度表の例

運転記述

運転を記述するための仕様記述で、状態遷移モデルに実装上の制約を考慮した記述。

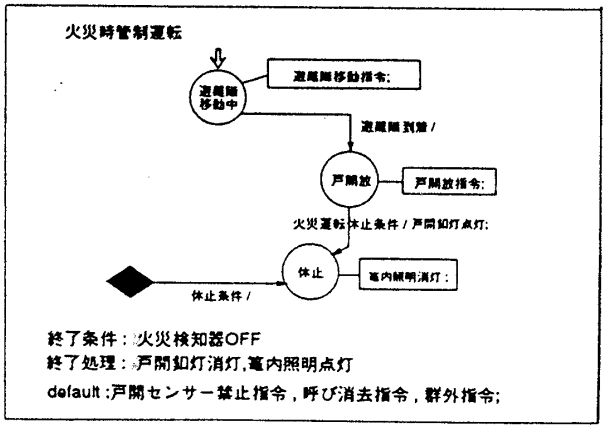

図 3: 運転記述の例

*Automatic Programming for Elevator Control Software

†Kazuyoshi TAMURA, Masashi INO, Akihiro OZEKI,Maki MINAGAWA, Koichi KAJI

‡R & D Center, Systems & Software Engineering lab., Toshiba Corp.

運転記述の状態遷移図については、以下のように拡張する。

- メモリ容量を圧縮するために、ある運転の全ての状態を通じて共通に実行する必要のある処理を表す default 処理記述を設ける
- 運転の任意の状態で切り替えが生じても、当該運転を正常に終了できるようにするための終了条件記述と終了処理記述を設ける
- 運転記述では、運転を切り分けて表現するために、運転が切り替わる時の遷移と処理を直接表現することができなくなる。そこで、運転が切り替わる時に、どの状態に遷移するかを表現する振り分け記述を設ける

3.2 仕様記述モデルのメカニズム

それぞれの仕様記述言語は、CPU 負荷率の軽減という実装上の制約を満たすため、条件の評価と処理の実行が最も少くなる次に示す動作メカニズムで解釈する。

優先度表

step1: 優先度の高い順に開始条件を評価する。

step2: 開始条件が成立した場合には該当する運転を起動し優先度表の評価を終了する。

step3: 開始条件が不成立の場合には下位の運転の開始条件を評価する。

step4: step2 へ。

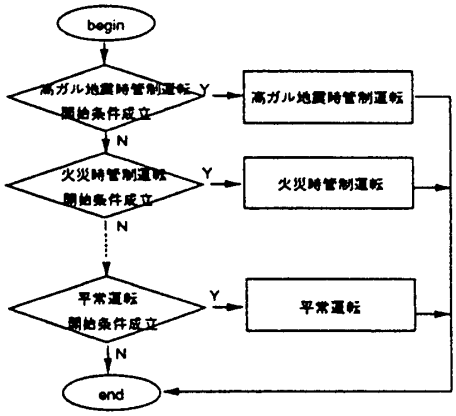

図 4: 優先度表の動作メカニズム

運転記述

step1: 終了条件の評価を行ない、満たしている場合には終了処理を実行して終了する。

step2: default 処理の実行する。

step3: 今まで運転が動作していなかった場合には、振り分け記述からの遷移のイベント評価を行ない、成立している遷移を実行する。不成立の場合には初期状態に遷移する。

step4: 状態における遷移の評価を行なう。現在の状態から出ている遷移でのイベントを評価し、成立している遷移のアクションを実行、遷移先へ遷移する。

ただし、状態遷移図の評価をタイムスライス方式で行ない、その 1 サイクルに 1 つの遷移までのみ実行することにしている。

4 適用実験による評価

この仕様記述言語を用いて実際のエレベーター制御ソフトウェアの仕様を記述し、プログラムを自動生成することによって、本方式の評価を行なった。

その結果、運転モード間の関係は優先度表、振り分け記述、終了処理記述の 3 つで表現することができ、個々の運転モードが独立に記述可能であることが示せた。

またエレベーターにおける

- CPU 負荷率を従来方式と同程度に抑える
- メモリ使用量を従来方式の 1.5 倍以内に抑える

という実装上の制約に対して、表 1に示すようにこの制約を満足する値が得られ、我々の提案する方法を実適用できることが確認できた。

表 1: 評価結果

評価項目	従来方式との比較
オブジェクトサイズ	1.30 倍
CPU 負荷率	1.03 倍

5 おわりに

今後更にソフトウェアの規模が大きくなることが予想されるため、最適化によるオブジェクトサイズの圧縮や、CPU 負荷率の軽減等を検討し、実製番への適用を行なっていく。

また、形式的に仕様を記述したことにより、仕様レベルでの検証やレビューが可能になったので、仕様検証系などによる早い段階での品質作り込みによる生産性の向上を目指す。

参考文献

[1] 岸他, “状態遷移モデルに基づくプログラム部品合成システムの開発”, 情報処理学会研究報告 91-SE-80, 1991.

[2] 清水他, “金融機器組み込みソフト向け CASE”, 情報処理学会ソフトウェア工学研究報告 93-SE-91, 1993.