無線列車制御システム評価における形式化技術の適用

福岡　博*

An Application of a Formal Technology to an Appraisal of a Train Control System with a Wireless Communication System

Hiroshi FUKUOKA

For a train control system with a wireless communication system, new architectures for wireless communications and train controls are introduced, and then conventional arguments often fail to focus on the superiority regarding the configurations. For such a reason, it is essential to model various properties properly and examine them sufficiently in advance. In this paper, an evaluation problem of a train delay for degraded situation is selected among such evaluation problems for a train control system with a wireless communication system. Then, Unified Modeling Language (UML), which provides many kinds of graphical expressions to examine the validity of the system requirements, is applied. In addition, a formal technique is introduced for proper modeling and sufficient examination of the evaluation.

キーワード：無線ベース列車制御システム，信頼性評価，UML，形式的手法

１．はじめに

無線ベース列車制御システムは，各列車が自己の位置を把握し，これらの情報を基に地上装置が列車と無線伝送しながら，列車間の間隔制御などを行うものであって，従来からある軌道回路を基にした固定的な閉そくシステムとは違う柔軟な列車制御が行えるため，コスト的にも有利で保守もしやすいといわれている。また，列車位置がほぼ正確にわかることから，それを用いた様々な機能を付加することができるなど，次世代の列車制御システムとして期待されている。しかし，一方で，無線通信や制御に関する新しいアーキテクチャが導入されるため，単純に従来のシステムの延長線上でその構成の優劣を議論できないケースも少なくない。このため，各種の性質について，適切にモデル化し，事前に十分な検証を行っておくことが求められている。ここでは，そのような無線ベースの列車制御システムの評価のうち，不具合発生時の列車の遅延に関する発生頻度の評価問題について述べ，その際に，システムの要件の定義について，グラフィカルな表現で要件の妥当性を検討しやすいという特性を持っている仕様記述言語UML（Unified Modeling Language：統一モデリング言語）を適用し，さらに，遅延の評価に関してクリティカルなテーマとなった共通原因故障問題について，形式化技術を用いて検証した結果について述べる。

＊　信号通信技術研究部

２．不具合発生時の列車遅延評価問題

ここでは，無線を用いた列車制御システムにおける不具合等による列車の遅延に関する発生頻度の評価について述べる。

2.1　遅延の発生頻度に関する評価

評価指標としては，列車遅延に結びつくような故障等の発生があった時の列車の遅延時間に関する発生頻度を評価したものとする。そのため，

「特定の遅延事故の発生形態に関して，遅延事故が発生する頻度とその際の遅延時間を求め，これを全ての発生形態に関して考察したもの」

を考え，これが特定の基準を満足するか否かを検証する。

2.2　故障進展モデル

前節で定義した遅延事故の発生形態のモデル化について，遅延時間を各事故形態に対して考えた場合，一般的には，

(1) 故障等が発生する。

(2) それがサブシステムの故障，列車制御システムの不

具合，列車運行における遅延の発生として波及してゆく。
(3) これに対して，それらが波及しないような検知 / 対応（運転士 / 指令等の操作を含む）がそれぞれの段階でなされる。
(4) それらが結果的には有効に働かず，遅延を発生させてしまった場合には，その遅延が早期に回復するような復旧手法（運転士/指令等の操作を含む）が機能する。

というシナリオによって事態は進行すると考えられる。そのため，遅延時間の評価は，これらの要素を含めて考えることによって行うことができる。

このような，故障が起きて様々な条件から事故が拡大あるいは収束してゆく様子を，ここでは故障進展と呼ぶことにし，これをモデル化し，遅延時間等について考察することを故障進展解析と呼ぶ。つまり，故障の起きる要因と波及との関係をモデル化し，故障形態の分類別に，故障の波及に関わる要因の組み合わせと，それにより生ずる遅延時間等を考察するものである。

2.3 モデル化の方法

はじめに述べたように，無線を用いた列車制御システムは，新しいアーキテクチャを様々な箇所で用いているため，通常のシステムの延長線上では考えられない部分が多い。そのために，評価の過程で検討が不十分になり，モデル化が不完全なものとなって，評価が十分に行えないという結果になる可能性がある。そこで，そのような状況に対応するために，モデルを定義しようとしている技術者だけではなく，モデル化したい情報を持っている技術者に対しても理解しやすい，そして，曖昧でないモデル化手法が必要である。

これに対して，故障解析の分野では，特性要因図，FMEA や HAZOP，さらに ETA/FTA などの様々な特長を持っている手法がある。しかし，定義しようとしている故障形態が本当に現実の状況を網羅したものなのかどうかという観点からは，これらの既存の手法は，故障を列挙的に数え上げて構造化してゆく方法であるため，必ずしもシステム全体が見通しよくモデル化できるものであるとは言えない。

そこで，ここでは，問題領域を単純化して，自然に無理なくとらえることができるようにするという意味で，システムの構造図などをベースに，グラフィカルなイメージでモデルを構造化することにした。

具体的には，今回のモデル化では，要素の故障，サブシステムの故障，システムの不具合，列車の動き，運転士の操作，指令の処置などが，複雑な形で関係を持つことになるが，それらの間の呼応関係を，以下に示すように，さまざまな要素の間の情報交換ととらえることで，モデル化を行う。

(1) 故障要因の解析

故障要因を解析する装置のアーキテクチャ図を用い，図上に，機能上の不具合を要素として表現し，一つの機能に不具合が生じた時に他の機能に影響を生ずることをその要素間の関係として表現する。これによって，故障に関する関係図（これを故障要因解析図と呼ぶ）を作成する。

(2) 故障伝搬の解析

(1) で設定した不具合の要素間の関係を，システム全体の機能に影響が及ぶ故障要因の要素としてとらえ，この要素から影響を受けて不具合を生ずるサブシステム，システムの機能について，これをそれぞれ要素として表現し，故障が故障要因，サブシステム，システムへと伝搬してゆく様子を，要素間の関係としてとらえる。これにより，故障の波及の様子を要素間の関係図として表現する。

(3) 復旧シーケンスの評価

装置の故障，運転士，指令によるオペレーションを含めた回復動作などの復旧時の一連の状況を表現するため，故障の発生現象，各機能の動作を要素として表し，それらの要素間の呼応状況を関係として表現する。ここでは，適宜シーケンス図等を用いて表現する。この結果あるいは表現された関係の一部は，復旧シーケンスに関する進展の状況を表すものとして，故障進展の関係図に追加する。

(4) 故障進展シナリオの評価

(1)，(2)，(3) で作成した故障進展をもとに，これらを整理して，シナリオとしてまとめる。出来上がったシナリオに対して，発生率と遅延時間を見積もり，最終的にシナリオ全体に関するモデルを生成する。

次章では，この評価モデルの UML による表現と，さらに，これをベースとして，特にクリティカルなテーマとなった共通原因故障問題への，形式的な検証手法の適用について述べる。

3．UML による記述と形式的な検証

前節で述べたように，問題のモデル化においては，問題領域を単純化して，自然に無理なくとらえるために，システムの構造図などをベースに，グラフィカルなイメージでモデルを構造化する手法を用いた。

ここでは，このモデルをベースに，厳密なモデル化，十分な検証を行うことができるものとして，このモデルと同様の構造を持ち，また，近年，広く普及してきた仕様記述言語である UML（Unified Modeling Language：統一モデリング言語）を用いて記述し，さらに形式的に評価工程を運用する手法について述べる。

3.1 UMLによる記述

UMLは，仕様をわかりやすく，なおかつ正確に記述する目的で開発された仕様記述言語である。システムの構造を抽象化した形で，構造的かつ形式的に記述することができるようにしているため，近年多くのソフトウェア開発で用いられるようになってきた。また，そのため，形式的な検証作業の入口としてUMLを用いる研究が盛んに行われている[1),2)]。

さて，基本的にUMLは，様々なシーンで使われることを想定して構成されている多様な表現法をもつ言語であるので，ここでは，UMLのこれらの利点を生かした形で，対象とする問題に適したモデル化を行うことになる。これは，具体的には，アプリケーションの性質を損なうことなく，できれば自然に，なおかつ検証のために有用な形となるように表現するということである。

この意味では，前節で定義したように，ここで扱っている「故障の発生・進展・復旧モデル」は，すでに「要素間の呼応関係」の形でモデル化を行っているので，基本的には，それらの関係は，UMLにおける「クラス」間の「関連」によって記述することができる。ただし，復旧シーケンスのように，特に複雑な呼応関係が生ずるケースでは，状況を明らかにするために，シーケンス図や，さらにはステート図などを併用する。

3.2 形式的な検証

UMLは，わかりやすいモデルの表現を提供するが，それだけでは，複雑な性質を満足していることを判定することはできない。そのために，形式的な検証技術を用いて，これを保障することを考える。

ここでは，そのための検証手法として，モデル検査（model checking）手法を採用する。この手法は，有限状態遷移系に対する形式的手法の一種で，これをソフトウェアを含めた設計仕様の検証に適用する研究が，近年，盛んに行われている[3)]。

ここで，モデル検査手法は，基本的に有限遷移系の到達可能な状態を数え上げることによって仕様を検証する方法であるが，システムを表すパラメータの数に対して，到達可能な状態数が指数的に増大する（状態数爆発（state explosion））という問題がある。そこで，この手法の適用を試みていた初期の段階では，多数の状態をすべて検査するために非常に大きなメモリ空間・時間を要し[4)]，実際の検証問題において求められる大きさのモデルをそのままの形で検証することは困難であった。ところが，この状況は，二分決定グラフ（BDD - Binary Decision Diagram）と呼ばれるデータ構造を取り入れて，必要な空間・時間量を節約することで，大きく改善された[5)]。この手法は特にハードウェア設計の分野では，実用的な検証手法として大きく脚光を浴びることとなったが，一方で，これをハードウェア設計以外の分野に適用しようとしたときには，もともとの状態数が多いために，状態数の爆発はいまだに簡単に対応することはできない問題となっている[6)]。

そこで，そのような状況では，モデル化により状態数そのものを減らすことが必要になる。しかし，一般に，モデル化は高度な技術が必要な作業であるため，現実のモデルの動作を忠実に反映しながら，なおかつ検証に耐えるような抽象化を行うことは，非常に困難な作業となる。

ここでは，そのような観点から，有効な抽象化手法を考え，この手法をベースに，UMLからモデル検査用のモデルを抽出して，これを用いて検証を行う。

3.3 モデルの抽象化

前節で述べたように，モデル検査用の言語の記述内容，モデル検査用のツールで探索できる状態空間の大きさは限られているため，大規模なシステムの検証を行うためには，対象とするモデルから，検証内容が的確に反映されていて，効率的な検証ができる検証用のモデルを抽出する必要がある。

ここでは，このためのベースとなるモデル化手法として，状態間の関係を定型化して，明示的に表示するネットワーク型信頼性モデルである定型化状態影響ダイアグラム（TSID：Typed State Influence Diagram，図1）[7)]を用いる。

なお，TSIDモデルでは，各システムの状態に関する関係式（オペレータ）としては，一般的な関数関係を設定することも可能であるが，ここでは，検証手法で利用できる状態間の関係式，および検証ツールで利用できる状態空間の大きさを考慮して，適切な関係式を導入することになる。

さて，TSIDモデルは，さまざまな要素の状態間の呼応関係をモデルとして表現したものである。そのため，3.1節で述べたUMLモデルに対しては，各「クラス」における「状態（の変化）」を考え，これに対応する「関連」をTSIDモデルにおける「状態」とこれに対応するTSID

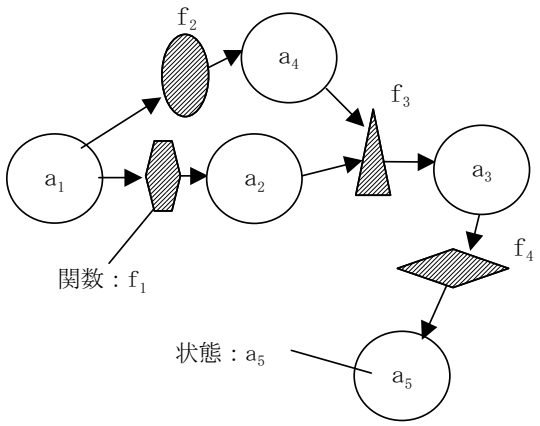


図1　定型化状態影響ダイアグラム

の「呼応関係」に置き換えることにより，UMLモデルをTSIDモデルへ変換する。

なお，一旦TSIDモデルに変換された後は，TSIDの要素のネットワークにおいて，サブネットワークとして独立しているものは，基本的には，ひとつの要素の状態変化を表すものに変換可能である。ただし，次節で述べるように，実用上は，他の要素に対して依存関係がないかを，慎重に確認しながら行うことになる。また，モデル上は関係が設定されていても，実際上は関係がないものとして扱ってもよい（近似的に呼応関係をはずしてもよい）場合なども，この時点で検証する。このようにして，モデルを抽象化することができる。

3.4　検証する性質の選定

モデル検査では，考えられる全ての遷移状態について，指定した性質（特定の設計仕様）が成り立つか否かを自動的に調べるものであるが，どのような性質を調べるべきかは，アプリケーション･システムとしての適用業務による。

また，文献１）で述べられているように，検証に用いるツールについては，大きなモデルは扱えない，またツールごとに得意／不得意な分野があるなど，さまざまな制約があり，それにあわせて，適切な検証用のモデル化を行う必要がある。

一方，ここでは，前節で述べた抽象化により，モデルのコンパクト化がはかられること，また，モデル化の対象としている故障進展における故障の伝播が，基本的に階層構造となっているシステムの構成に従って順に評価される部分が大きかったため，大部分は，単純に，故障進展のシナリオに沿って順番に計算してゆくことができた。そのため，その過程では，特に形式的手法を適用すべき検証項目として設定する必要性が生ずる要素がみられなかった。

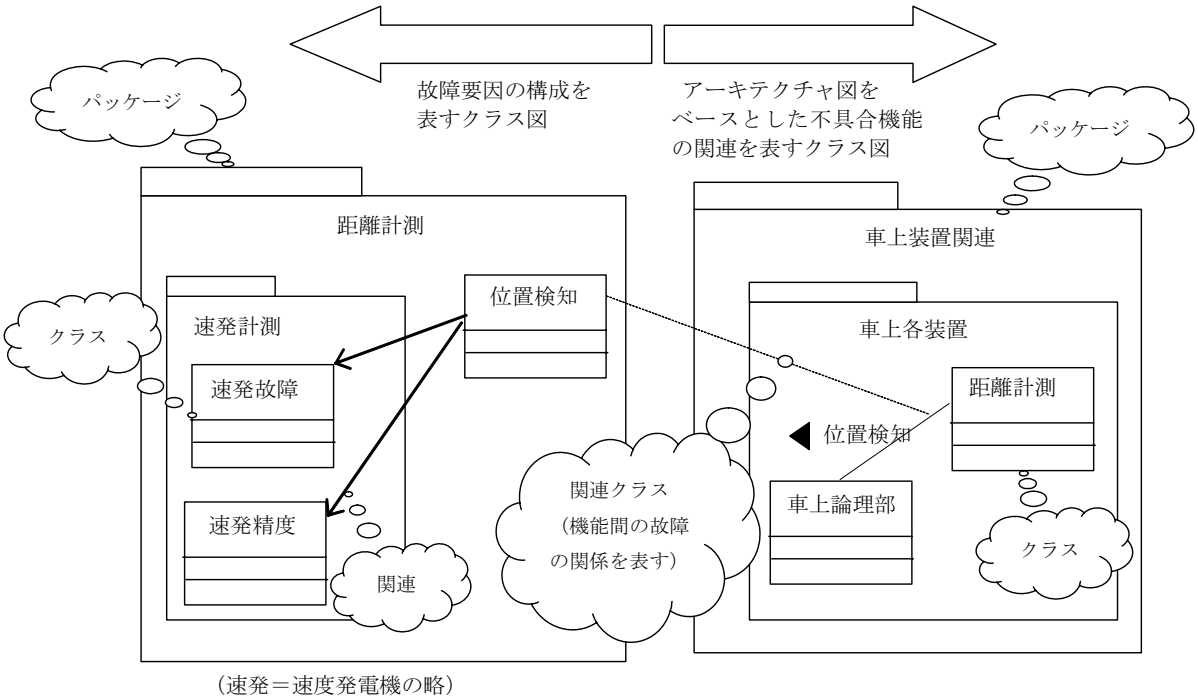
ただし，前節で述べたように，そのようなシナリオの設定は，少なからず，要素間／サブシステム間／システム間の独立性を仮定して得られている部分に依っていることが大きく，逆に，そのような部分での依存関係の「切捨て」が，全体のシステムの振る舞いに与える影響について，具体的にチェックしておく必要性が生じた。

そこで，ここでは，そのような依存関係，すなわち共通原因故障のシステム全体への影響の有無を感度解析的に検証すること（実際は，ある程度大きな共通原因故障が要素として設定されても，影響がないことを確認すること）に注目して，モデル化を行った。

また，検証システムの検証機能にあわせて，ここでは，ドミナントなシナリオ（遅延時間が大きく，なおかつその発生頻度が高いもの）に注目してモデル化することにした。すなわち，たとえば，遅延時間が3時間以上のシナリオで10年に1回の発生頻度となるような共通原因起因のものがないとするというような性質を検証するものとする。

3.5　検証ツールの利用

ここでは，モデル検査のための検証ツールとして，AT&Tベル研究所により開発されたSPINを用いる。これは，Promelaという仕様記述言語により検証したい動作仕様を記述し，検証する性質を，ラベル，表明，性質オートマトンまたはLTL（線形時相論理）により記述することによって，その仕様のシミュレーションや検証を



（速発＝速度発電機の略）

図２　UMLによる記述例（クラス図の一部）

行うものであって，近年，並行プロセスのタイミング検証などによく使われている[8) 9) 10)]。

TSIDモデルからPromelaへの変換は，TSIDモデルにおける状態とその呼応関係をPromelaにおける「process」内の状態とその呼応関係に置き換えることにより行う。

なお，Promela／Spinの組み合わせによる検証は，前述の通り，並行プロセスおけるデッドロック検出などを得意とするものであるが，Promela／Spinがツールとして整備されていること，また，アルゴリズムをPromelaで書くと，探索すべき状態空間を自動的に構成してくれるので，感度解析的にアルゴリズムを次々と変更していっても自動的に対応してくれるといったメリットがあることで，有効に機能している。

4．ケーススタディ

4.1　対象とするシステムの概要

ここで対象とする無線を用いた列車制御システムは，位置検知用の速度発電機と位置補正用のトランスポンダを用いた列車位置検知装置から得られる位置情報と，これをもとに拠点装置が作成した経路と停止限界点により，車上で速度照査パターンを発生させ，列車間の間隔制御を行うもので，車上装置と拠点装置間は無線通信で情報交換を行うものである。

また，列車の運用形態は，環状，複線の通勤線区（約35キロ）で，線区全体で50列車（上り，下りを含む）が同時に運行し，拠点装置は線区に対して5台（受け持ち地域は，対象線区をほぼ均等に5分割したもの）と想定している。

4.2　シナリオに関する評価

はじめに，故障進展に関するクラス図を作成した。図2にその一部を示す。これは，故障要因に関するクラス図の一部で，右半分にアーキテクチャ図上に不具合となる機能が示され，左側に故障要因が示されている。各要素は，UMLのクラス，あるいは関連クラスとして表され，これらの呼応関係は，UMLの関連として表現されている。ここでは，距離計測から車上論理部への故障の影響である「位置検知」は「速発故障」と「速発精度」から生じていることが示されており，これらは故障要因として次のステップ（故障の波及を示すクラス図）で用いられることになる。

また，復旧シーケンスについては，はじめはシーケンス図により解析した。ただし，復旧時のシーケンスの構成については，担当者間の打ち合わせなど，複雑な復旧時の情報のやり取りについては，シーケンスごとに大きな差異は見受けられず，結局，（近似的に）実際の復旧手法の違いに起因する数種類の要素に集約された。

さて，これを，TSIDに変換し，さらに3.5節で述べたように，検証が必要となる共通原因故障に関する影響の部分のみをもとのTSIDに対する「差分」の形で，検証モデルとして取り出した。図3にそのような検証モデルの例を示す。ここでは，車上装置に関する4つの機能に関する共通原因故障が，

(1) 車上のデータベースのリセットの必要性の有無に関する違いとして，車上システム全体のリセット時間（列車の緊急停止に続く制御システムの再起動時間）に影響する場合

(2) 位置検知機能の喪失の有無に関する違いとして，

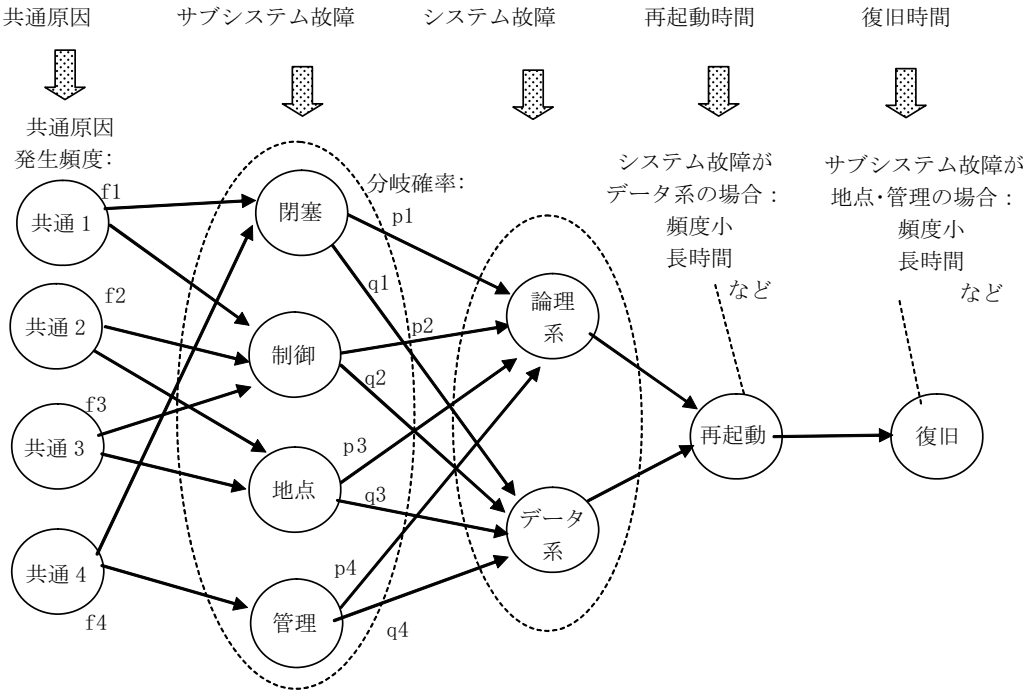


図３　検証モデルの例：車上制御装置の場合（概要）

救援列車の必要性あるいは救援列車による復旧時間に影響する場合

として，モデル化され，さまざまなパラメータが感度解析的に設定されることになる。

このモデルは，Promela で記述されるが，検証条件は，3.4 節で示すように，発生頻度と遅延時間の両方で設定されるため，Promela の呼応関係では，頻度のオーダーと遅延時間が離散値で近似された数値が設定されるようにして，状態数の集約を図っている。

さらに，Promela で記述されたモデルを，パラメータ，モデル構成を変化させて，SPIN により検証して，これらの共通原因故障がシステム全体の評価に大きな影響を与えないことを検証する。

たとえば，図 3 の例では，表明（assert 式）：

assert(! (istep_s == 1 && areax_s <= 6 && areay_s >=7))

［これは，故障時（istep_s == 1）に「発生率が 2^{-6} 以上（2のべき乗の指数の負号をとったもの：areax_s <= 6）で，なおかつ遅延時間が 7x20 分（2 時間 20 分）以上（areay_ s >=7 ）」のシナリオがないことを表している（数字は，いずれも元のモデルからの「差分」として設定した値）］

によって，検証条件を設定しているが，この例では，

(Spin Version 4.3.0 -- 22 June 2007)

+ Partial Order Reduction

Full statespace search for:

……（略）……

State-vector 112 byte, depth reached 56, errors: 0

という検証にパスしたというメッセージが得られる。

ちなみに，ここで，発生率側の条件を 2^{-7} 以上と厳しくしてみると，

pan: assertion violated ! ((((istep_s==1) && (areax_s <= 7)) && (areay_s >= 7))) (at depth 54)

というエラーメッセージが出力され，エラー時のシミュレーション機能を使うと，発生率が 2^{-7} で，遅延時間が 8x20分のシナリオがあることがわかり，その内容を解析することができる。このように，検証に失敗した場合は，そのケースを吟味して再度パラメータ等を設定し直すこともでき，このような手続きを感度解析的に繰り返すことにより，検証作業を進めてゆくことができる。

なお，SPIN の実行は，いずれの場合もパソコン（Pentium 4, Linux）上で数十秒以内で終了し，実用的な範囲内で検証作業ができたものと考えられる。

５．おわりに

無線ベース列車制御システムは，無線通信や制御に関する新しいアーキテクチャが導入されるため，単純に従来のシステムの延長線上でその構成の優劣を議論できないことも多い。このため，各種の性質について，適切にモデル化し，事前に十分な検証を行っておくことが求められている。しかし，それには妥当性が検討しやすく，厳密な検証ができるモデル化手法が必要となる。

ここでは，そのような観点から，直観的に理解しやすいモデル化を行い，同時にクリティカルな検証項目について，厳密な検証を行うという目的で，UMLによるモデル記述とモデル検査（model checking）手法による形式的な検証の組み合わせによる評価手法を示した。

また，この新たに開発した手法を無線ベース列車制御システムにおける不具合発生時の列車遅延評価問題に適用した例を示し，この手法が，実用的に有効に運用できるものであることを確認した。

今後は，他のツールの導入，他の指標の評価問題，あるいは他のシステム検証問題などに適用を拡大してゆく予定である。

文　献

１）岸知二，“モデル検査技術による UML 設計検証”，情報処理学会誌，2006 年 5 月，pp. 498-505

２）Y. Ledru, “Using Jaza to Animate RoZ Specifications of UML Class Diagrams”, 30th Annual IEEE/NASA Software Engineering Workshop SEW-30 (SEW'06) , pp. 253-262

３）青木利晃，“形式的手法による高信頼性組み込みソフトウェア開発”，情報処理学会誌，2006 年 5 月，pp. 491-497

４）福岡博 ,et al.，“ペトリネットによる連動仕様の検証”，鉄道総研報告 , Vol.9, No.11, pp.19-24, 1995

５）越村三幸, et al.,“鉄道信号システムの連動図表と連動装置のモデル検査”，第二回 システム検証の科学技術シンポジウム，2005 年 10 月 20 日

６）亀山幸義, et. al.,“ケーススタディ : モデル検査と定理証明を用いた鉄道信号制御システムの検証”，第1回「システム検証の科学技術」シンポジウム , 2004 年 2 月 4 日～6 日

７）Hiroshi Fukuoka, “Reliability Evaluation Method for the Railway System : A Model for Complicated Dependency”, Quarterly Report of RTRI, Vol.43, No.4, pp.192-196, 2002

８）Alessandro Cimatti, et al.,“Model Checking Safety-Critical Software with SPIN : An Application to a Railway Interlocking System,”Lecture Notes in Computer Science, Vol.1516, pp.284-296, 1998

９）Gnesi, S., Lenzini, et al.,“An Automatic SPIN Validation of a Safety Critical Railway Control System”, In Proc. of Int. Conf. on Dependable Systems and Networks, (2000), pp. 119-124

10）Kirsten Winter, et al., “Modelling Large Interlocking Systems and Model Checking Small Ones”, In M. Oudshoorn(ed.), 26th Australasian Computer Science Conference (ACSC 2003), Australian Computer Science Communications, Vol. 16, 2003