アーキテクチャ記述言語による設計と一体化した制御システムのリスク分析*

吉村　悠[†a)]　日高　隆博[††]　中條　直也[†††]　高田　広章[†]
高浜　盛雄[†]

Risk Analysis of Control System Unified with Design Using Architecture Description Language*

Yu YOSHIMURA[†a)], Takahiro HIDAKA[††], Naoya CHUJO[†††], Hiroaki TAKADA[†], and Morio TAKAHAMA[†]

あらまし　本論文では制御システムの大規模・複雑化に対応するリスク分析を提案する．システム設計の上流工程をモデルとして記述する枠組みであるアーキテクチャ記述言語と，故障に関する情報を記述するエラーモデルに着目し，設計工程と体系的になったリスク分析システムを開発した．設計者は，システムをアーキクチャ記述言語の一種である AADL によって記述し，コンポーネントごとのエラーモデルを確率状態遷移として記述するものとする．我々は，このアーキテクチャ記述とエラーモデル記述を用いたリスク分析として，エラーモデル合成と簡約化を定義した．エラーモデル合成によってシステムの故障を計算することができ，簡約化によって大規模なシステムでもリスク分析ができるように状態数を削減し，計算量が小さくなることを示す．リスク分析システムによって，設計システムの潜在的な危険性をリスクとして表し，システムの安全性評価が可能となる．また，リスクが高い箇所を優先的に対策することによる効率的なシステムの設計変更を提案する．自動車制御システムの重要なサブシステムである電動パワーステアリング・システムを事例として，開発システムの有効性を考察する．

キーワード　アーキテクチャ記述言語，エラーモデル，AADL，リスク分析

1. まえがき

制御システムが大規模・複雑化しているため，手戻りによる影響が大きくなり，設計が困難になっている．このため，上流工程でシステムの設計をモデル化するための枠組みであるアーキテクチャ記述言語 (ADL) が注目されている．一方，自動車に代表される制御システムでは安全性が重要であるが，制御システムの複雑化によって設計段階での安全性の確保が困難になっている．そのため，危険性を設計の早期段階で見積もることが求められている．また，安全性に対する懸念から機能安全規格 [1] の策定が進んでおり，定量的なシステムの安全性を示す必要に迫られている．

このような背景から，ADL のサブ言語として故障の影響による振舞いを記述するエラーモデルが規定された．そして，エラーモデルと ADL を組み合わせて安全性を分析する手法が提案されている [2], [3]．文献 [2] では，FMEA や FTA による安全性評価手法を提案している．しかし，エラーモデルをそのまま計算しているので，計算量が膨大になるという問題がある．すなわち，対象システムの規模が大きくなると，安全性評価が行えなくなるおそれがある．文献 [3] では，エラーモデルから Generalized Stochastic Petri Net への変換によるシミュレーション手法を提案している．しかし，計算量が膨大である上に，システムの安全性評価

† 名古屋大学大学院情報科学研究科，名古屋市
Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya-shi, 464–8603 Japan
†† 名古屋大学大学院情報科学研究科附属組込みシステム研究センター，名古屋市
Center for Embedded Computing Systems, Nagoya University, Nagoya-shi, 464–8601 Japan
††† 愛知工業大学情報科学部情報科学科，豊田市
Faculty of Information Science, Aichi Institute of Technology, Toyota-shi, 470–0392 Japan
a) E-mail: yoshimura@nagoya-u.jp
* 本論文はシステム開発論文である．

方法までは扱っていない．そこで我々は，文献 [4] で，FSM による簡約化によって計算量の問題を解決できると見当をつけたが，エラーモデルの解釈にあいまいさがあることが分かった．

本論文では，ADL とエラーモデルを用いてシステムの危険性をリスクとして表示することでシステムの安全性を分析し，設計に反映できることを示す．エラーモデルの定義にあいまい性のない定義を与え，リスク分析手法であるエラーモデル合成と簡約化を確率モデルによって定義する．また，簡約化によって計算量が小さくなることを示す．

本論文の **2.** で，ADL について説明する．**3.** で，エラーモデル合成と簡約化の定義に必要な確率モデルについて説明する．**4.** で，開発したリスク分析システムを示す．**5.** で，自動車制御システムの重要なサブシステムである電動パワーステアリング (EPS) を事例として，開発システムの効果を示す．最後に本論文をまとめる．

2. アーキテクチャ記述言語

アーキテクチャ記述言語 (ADL：Architecture Description Language) とは，システムやソフトウェアの仕様を記述するための言語である．システムの仕様を ADL によって網羅できることが望ましいが，従来の自然言語よりも記述範囲が小さくなってしまう．そのため，ADL では対象とするシステムにおける記述範囲の大きさが重要となる．

ADL は多数提案されており，汎用性の高い UML [5] が広く知られている．対象別の ADL として，UML2.0 のシステム工学向けの拡張付きサブセットである SysML [6] や自動車制御向けの EAST-ADL2 [7]，航空宇宙向けの AADL [8] などがある．これらの中で，安全性が重要なシステムを対象として策定されたのが AADL であり，エラーモデルとして Error Model Annex [9] が定義されている．そのため，本論文では AADL を用いてシステム設計を行う．

2.1 Architecture Analysis & Design Language

Architecture Analysis & Design Language (AADL) は自動車・飛行機等の技術者協会である SAE の委員会 AS-2 が言語仕様を策定した ADL である．AADL で定義している，システム/ソフトウェア/ハードウェアに分類されたコンポーネント（図 1）とその相互関係によって記述する．主な関係として，ハードウェアとソフトウェアをまたがったデータフローやソフトウェアをハードウェアに割り当てるバインドが挙げられる．本論文では，AADL で記述されたアーキテクチャを AADL モデルと呼ぶ．AADL モデルはコンポーネントがサブコンポーネントをもつことによって階層構造をなし，最上位はシステムコンポーネントの構成となる．

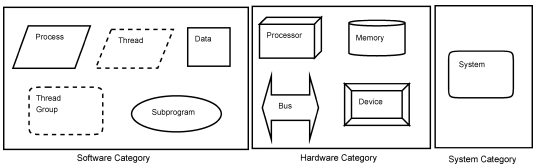

図 1　AADL コンポーネント
Fig. 1　AADL components.

2.2 Error Model Annex

Error Model Annex とは，仕様外の振舞いの情報を AADL に追加するため，SAE で標準化された言語である．コンポーネントのエラーモデルを確率状態遷移として表現している．AADL では，ハードウェアコンポーネントとソフトウェアコンポーネントのいずれもエラーモデルを確率状態遷移として表現するため，ハードウェアとソフトウェアをまたがった一元的なエラーモデルを記述できる．また，コンポーネントを介した階層構造をもたせることができる．ここで，フォールトとはコンポーネント内の構造やデータが異常な状態であり，故障とはフォールトによる異常によってコンポーネントの機能を失うことである．エラーモデルでは，フォールトを状態として，故障をエラーモデル間の接続であるエラー伝搬として表現する．

Error Model Annex では，状態を error state として宣言する．error state は initial error state として，初期状態が一つのみ宣言されなければならない．状態間の遷移を error event または，エラー伝搬 error propagation として宣言する．エラー伝搬は，コンポーネント外部へ影響を伝える out error propagation と外部からの影響を受けて遷移する in error propagation に分けられる．error event と out error propagation には，$[0, 1]$ の固定確率またはポアソン過程のパラメータ $\lambda \in \mathbb{R}^{\geq 0}$ を指定する．

エラーモデル間のエラー伝搬の接続は，ネームマッチング及び AADL モデルの情報を利用する．例えば，AADL モデルでデータフローが記述されると，エラー伝搬はその方向に沿って伝搬する．エラーモデル

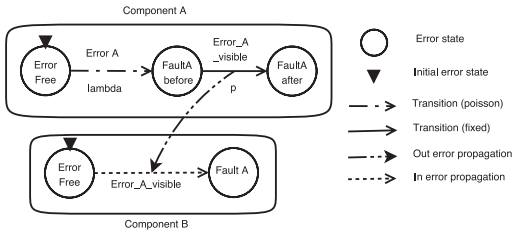

図 2　エラーモデルの例
Fig. 2　An example of error model.

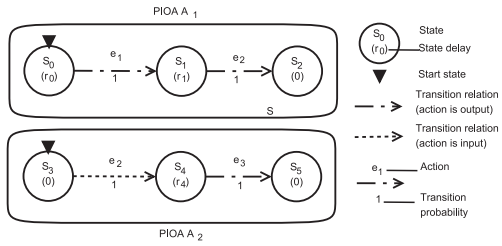

図 3　PIOA の例
Fig. 3　An example of PIOA.

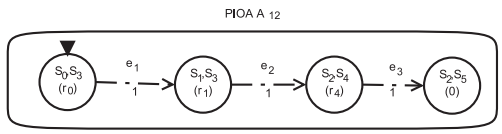

図 4　PIOA の合成の例
Fig. 4　An example of PIOA composition.

の例を図 2 に示す．このエラーモデルでは，初期状態 ErrorFree をもち，$1 - e^{-\lambda t}$ の確率で状態 FaultAbefore に遷移する．状態 FaultAbefore では，確率 p によってコンポーネント外部へ影響を伝えるエラー伝搬 Error_A_visible が起きることを表している．また，ポアソン分布に従う確率によってフォールトの起きやすさが表現され，固定確率によって検知や伝搬の起きやすさを表現する．しかし，ある状態から固定確率とポアソン過程による遷移が両方存在する場合における遷移など，Error Model Annex は解釈にあいまいさが存在するという問題を抱えている．

3. 確率モデル

本章では，エラーモデル合成と簡約化の定義に必要な確率モデルの Probabilistic I/O Automata (PIOA) と，連続時間マルコフ連鎖 (CTMC) について説明する．

3.1 Probabilistic I/O Automata

Probabilistic I/O Automata [10] は，IO オートマトン [11] のポアソン過程による確率拡張であり，式 (1) で定義される．E は互いに素である input，output，internal action の三つの集合に分けられ，それぞれ E^{out}，E^{in}，E^{int} と表す．E^{int} は使用しないので，本論文では $E^{int} = \phi$ として扱う．図 3 に PIOA の例を示す．

$$
\begin{aligned}
&A = \{S, s^I, E, \Delta, \mu, \sigma\} \\
&S\text{：状態集合} \\
&s^I\text{：初期状態} \\
&E\text{：アクション集合} \\
&\Delta\text{：遷移関係 } (\Delta \subseteq S \times E \times S) \\
&\mu\text{：遷移確率関数 } (\mu : (S \times E \times S) \to [0,1]) \\
&\sigma\text{：状態遅延関数 } (\sigma : S \to \mathbb{R}^{\geq 0})
\end{aligned} \tag{1}
$$

状態からの E^{out} による遷移の間隔は状態遅延関数の値をパラメータとする指数分布に従い，遷移確率関数によって遷移先が決定される．また，$\forall s \in S$ に対し，$\sigma(s) > 0$ であることは，$(s, e, r) \subset \Delta$ である $e \in E^{out}$ と $r \in S$ が存在することと同値である．

$E^{in} = \phi$ である PIOA は閉じているといい，一意の連続時間マルコフ連鎖 (CTMC) に変換できる．

PIOA の遷移による状態とアクション列から PIOA の合成が定義される．PIOA を $A_i = \{S_i, s_i^I, E_i, \Delta_i, \mu_i, \sigma_i\}.i \in I$ とすると，$\forall i, j (i \neq j) \in I. E_i^{out} \cap E_j^{out} = \phi$ のとき，合成後の PIOA$\prod_{i \in I} A_i$ が存在する．図 3 の PIOAA_1 と A_2 を合成した PIOA A_{12} を図 4 に示す．

3.2 連続時間マルコフ連鎖

連続時間マルコフ連鎖 (CTMC) [12] とは，マルコフ連鎖を連続時間に適応したものである．CTMC 中の状態からの遷移は，ポアソン過程に従う．状態集合を S，生成行列を $\mathbf{Q} : |S| \times |S| \to \mathbb{R}$ とすると，CTMC $M = (S, \mathbf{Q})$ として定義できる．$s_i, s_j \in S$ として，生成行列の要素を以下のように表す．

$$
\mathbf{Q}(s_i, s_j) = \begin{cases} \lambda \in \mathbb{R}^{\geq 0} & i \neq j \\ -\sum_{i \neq k} \mathbf{Q}(s_i, s_k) & i = j \end{cases} \tag{2}
$$

時刻 $t \geq 0$ における各状態の確率分布 $\mathbf{Pr}$ は，先頭要素が 1 でその他の要素が 0 である長さ $|S|$ のベクトル $\pi(0)$ を用いて，以下の式で計算できる．

$$
\mathbf{Pr} = \pi(0) * exp(\mathbf{Q}t) \tag{3}
$$

CTMC の状態集合 S を集合としてまとめた $P = Y_0, Y_1 ..., Y_n$ を partition と呼ぶ．partition の要素 $Y_i (0 \leq i \leq n)$ を状態とする新しい CTMC を作ることを lumping [13] と呼ぶ．lumping は CTMC の状態数を削減する強力な手法である．lumping には exact lumping と ordinary lumping の 2 種類あり，後者について説明する．

Ordinary lumping は partition P が以下の条件を満たすとき，lumping が可能である．

$$i \neq j . \forall Y_i, Y_j \in P . \forall s, \widehat{s} \in Y_i$$
$$\sum\nolimits_{s' \in Y_j} \mathbf{Q}(s, s') = \sum\nolimits_{s' \in Y_j} \mathbf{Q}(\widehat{s}, s') \quad (4)$$

新しい CTMC の生成行列 $\acute{\mathbf{Q}}$ は以下のようになる．

$$\acute{\mathbf{Q}}(Y_i, Y_j) = \begin{cases} \sum_{s \in Y_j} \mathbf{Q}(\acute{s} \in Y_i, s) & i \neq j \\ -\sum_{s \in P \setminus \{Y_j\}} \mathbf{Q}(\acute{s} \in Y_i, s) & i = j \end{cases} \quad (5)$$

4. リスク分析システム

システムにおけるリスクとは，危険な事象の起きやすさとその事象による影響の度合との組合せとされている [14]．ここで，エラーモデルがフォールトを状態として，フォールトの起きやすさを遷移確率として表現していることに着目する．システムの故障による影響の度合を影響度として，状態に影響度を対応づけることで定量的なリスクの計算ができる．本論文では，リスクをシステムが故障に至る確率とその故障による影響度の積と定義する．

本章では，エラーモデル合成と簡約化によるリスク分析システムを説明する．制御システムでは，故障した場合にユーザによる回復が期待できないものが多いものとして，恒久故障のみを分析するシステムとして開発した．まず，設計と一体化したリスク分析の流れを示す．次に，エラーモデルを PIOA と影響度関数によって定義し直し，**2.2** で述べた解釈のあいまいさを回避する．エラーモデル合成は状態の直積によって行うため，状態数が非常に多くなる．そこで，簡約化を定義して状態数を削減する．また，簡約化によってリスクを計算する．

4.1 設計と一体化したリスク分析

想定するシステムの設計とリスク分析の流れを図 5 に示す．まず AADL によってシステムを設計し，AADL

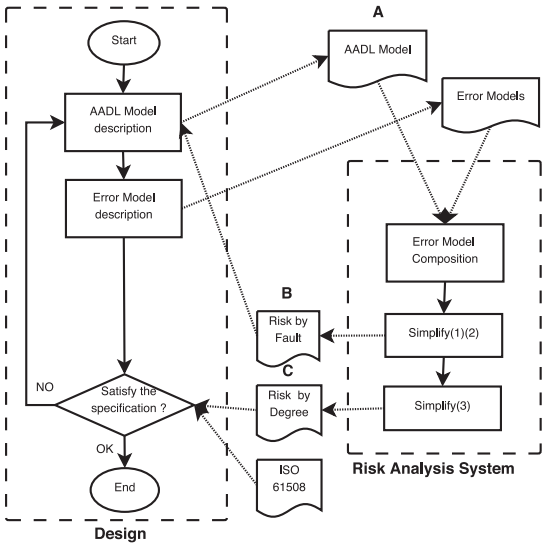

図 5 リスク分析システムの全体フロー
Fig. 5 Flow diagram of risk analysis system.

モデルを作成する．次に，システムを構成するコンポーネントのエラーモデルを記述する．この AADL モデルとエラーモデルを入力として，リスク分析システムがフォールト別リスクと影響度別リスクを計算する．影響度別リスクを制御システムに求められる安全規格 [14] などに照らして，安全性を判別する．自動車の場合は，安全規格 [1] に照らして安全性を判別する．要求される安全性を満たしてない場合は，システムの設計変更を行う．フォールト別リスクから，リスクが高い箇所を優先的に設計変更をすることで，効率的なリスクの低減を行うことが期待できる．また，より上位のシステムがリスク分析できるように，状態数を削減する．設計段階で安全性評価と再設計につなげられることが，リスク分析システムの特長である．

4.2 エラーモデルの定義

エラーモデルの解釈のあいまいさを回避するために，エラーモデルを PIOA と影響度関数の二つによって定義し直す．ここで問題になるのが，エラーモデルの遷移には固定確率とポアソン遷移の 2 種類存在するが，PIOA での遷移はポアソン過程しかない点である．PIOA での状態遅延関数の値が大きいほど，遷移する間隔が小さくなることに着目する．これを利用し，固定確率による遷移を状態遅延関数の値が非常に大きい状態からの遷移とみなして，状態遅延関数の値を $\sigma_m \in \mathbb{R}^{\geq 0}$ とする．これにより，error state からの遷

移は，状態遅延関数の値によって，ポアソン過程または固定確率のどちらか一意に決まる．エラーモデルの状態と遷移はそのまま用い，out error propagation と error event は E^{out} に，in error propagation は E^{in} に置き換える．

影響度関数を $D : S \to \mathbb{R}$ と定義し，エラーモデルを $\mathrm{EM} = \{S, s^I, E, \Delta, \mu, \sigma, D\}$ とする．つまり，EM は PIOA に影響度関数を追加したものである．D は状態に対応する影響度であり，値が大きいほどシステムに与える影響が大きいことを表す．関数の値が負の場合は，故障の影響を安全側に倒した場合（フェイルセーフ）を表す．システムに影響を与えない状態 s は $D(s) = 0$ である．

4.3 エラーモデル合成

エラーモデル合成を PIOA の合成と影響度関数の合成によって定義する．PIOA の合成条件として E^{out} は互いに素である必要があるため，エラーモデルは重複した error propagation 名をもつことができない．複数のエラーモデルから，あるエラーモデルの同じ状態に遷移させたい場合は，エラー伝搬を複数個作る必要がある．また，影響度関数の合成を以下のように定義する．

$$D(\langle s_i : i \in I \rangle) = \sum_{i \in I} D_i(s_i) \tag{6}$$

つまり，エラーモデルの状態の組合せによって影響度が決定する．

4.4 簡 約 化

設計するシステムの規模が拡大してもリスク分析ができるように状態数を削減し，リスクを計算する操作として簡約化を定義する．簡約化は以下の三つの手順によって行う．

（1） システム故障からの遷移の削除
（2） 固定確率による遷移の削除
（3） 同じ影響度である状態の統合

4.4.1 システム故障からの遷移の削除

システム故障をシステムの動作に影響がある状態とする．恒久的なフォールトしか発生しないと仮定したので，システム故障は恒久的であり正常に戻ることはない．システムが故障したらシステムは修理されると仮定し，システム故障までの遷移までを扱う．そのため，$s \in S.D(s) > 0 \wedge \sigma(s) < \sigma_m$ である状態からの遷移をすべて削除し，状態遅延関数の値を 0 にする．

4.4.2 固定確率による遷移の削除

フォールト発生を表すポアソン過程のパラメータは一般的に 1 より非常に小さい．一方，σ_m の値は非常に大きいと仮定した．よって，状態遅延関数の値は σ_m に比べて非常に小さく，故障確率に与える影響が無視できるほど小さい．そのため，状態遅延関数の値を $\sigma_m + \lambda \approx \sigma_m$ と近似できる．同様に遷移確率関数の値も $\frac{\lambda}{\sigma_m + \lambda} \approx 0$ と近似できる．更に，PIOA の定義より $s, r \in S.e \in E^{out}.\mu(s, e, r) = 0$ は $(s, e, r) \notin \Delta$ と同値であるため，$\sigma(s) \geq \sigma_m$ である状態から遷移が削除できる．

固定確率による遷移の削除は，ポアソン過程の分割によって行う．固定確率による複数状態への分岐は，ポアソン過程の分割とみなして thinning [15] 手法で分割する．$\forall s_i, s_j, s_k \in S \wedge \forall e_i, e_j \in E^{out}.(s_i, e_i, s_j), (s_j, e_j, s_k) \in \Delta$ に対して $\sigma(s_j) \geq \sigma_m$ のとき $\Delta \cup \{(s_i, e_k, s_k)\}/\{(s_i, e_i, s_j), (s_j, e_j, s_k)\}.E^{out} \cup \{e_k\}/\{e_i, e_j\}$ とする．ただし，$\mu(s_i, e_k, s_k) = \mu(s_i, e_i, s_j) \times \mu(s_j, e_j, s_k), e_k = e_i e_j$ である．

次に，PIOA を $i \neq j.s_i, s_j \in S.e \in E^{out}.\mathbf{Q}(s_i, s_j) = \sigma(s_i) \times \mu(s_i, e, s_j)$ にて CTMC に変換する．CTMC の各状態の確率分布を式 (3) によって計算し，状態に対応づけられた影響度を掛け合わせることでフォールト別リスクを計算する．

4.4.3 同じ影響度である状態の統合

システムの安全性を判別するため，また，より上位のシステムがリスク分析できるように，同じ影響度である状態を統合し，影響度別リスクを計算する．$s_i, s_j \in S.D(s_i) = D(s_j)$ の関係にある状態を同値類として，S の partition を作成する．この partion を **3.2** で述べた ordinary lumping によって状態を統合する．影響度のある状態からの遷移を削除したので，影響度がある状態は影響度ごとに状態がすべて統合される．統合時に式 (5) によって生成行列の要素が足され，生成行列から式 (3) により，各状態の確率分布を計算する．最後に，各状態に対応づけられた影響度と確率分布の積を取ることで，影響度別リスクを計算する．

5. 適 用 事 例

設計と一体化したリスク分析の効果を示すために，自動車制御システムの重要なサブシステムである電動パワーステアリング (EPS) を設計し，リスク分析を

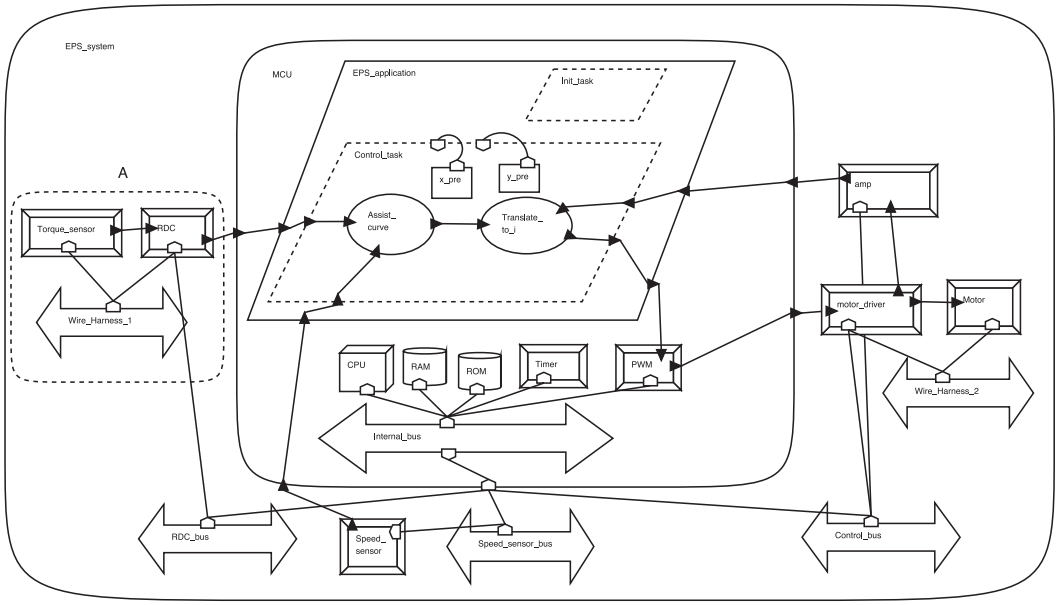

図 6　EPS の AADL モデル
Fig. 6　AADL model of EPS system.

行う．更に，リスク分析結果を用いてシステムの設計変更を行い，リスクの変化を検証する．

5.1　EPS システムの AADL モデル記述

文献 [16] を参考にして，EPS システムの AADL モデルを記述した．この AADL モデルが図 5 の A にあたり，図 6 に示す．EPS システムとは，ハンドル操作によるねじれ角をトルクセンサによって検出し，タイヤの操舵をモータのトルクによってアシストするシステムである．EPS システムでは，トルクセンサと車速センサの二つを入力として，コントローラが必要なアシスト量を決定し，モータをサーボ制御する．トルクのサーボ制御のために，電流センサを用いている．

トルクセンサ周りの AADL モデルを拡大したものを図 7 に示す．トルクセンサを図 6 のデバイスコンポーネント Torque_Sensor で表しており，トルクセンサのデータが RDC コンポーネントへ流れることを表している．また，このデータを Wire_Harness_1 を伝っていることを表している．このように AADL モデルによって，EPS システムを構成するコンポーネント及びデータの流れる方向，データが流れるコンポーネントを表現している．

5.2　エラーモデル記述

トルクセンサ周り（図 6 の A の部分）のエラーモデルを図 7 に示す．エラー伝搬を接続するために，基本

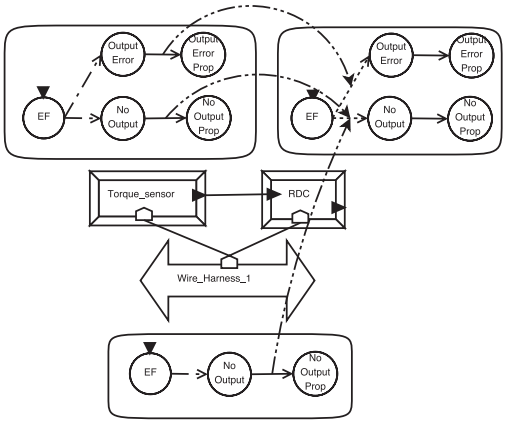

図 7　トルクセンサ周りの AADL モデルとエラーモデル
Fig. 7　AADL model and error model around the torque sensor.

的なフォールトを出力なしと出力異常の二つと定めた．図 7 の Torque_Sensor では，それぞれ NoOutput と OutputError とした．正常状態 EF から，これらのフォールトの状態へポアソン過程によってどちらかに遷移する．フォールト状態になると，すぐに NoOutputProp ないし OutputErrorProp に遷移する．この遷移時にデータが流れるコンポーネント RDC へ影響が伝搬して，RDC のエラーモデルの状態を遷移させ

表 1　影響度の定量化
Table 1　Degree quantification.

System harm	Subsystem harm	Degree
die	run away	4
cannot drive	stop	2
repellent	least operation	1
nothing	error free	0
	redundant faults	0

表 2　フォールト別リスク
Table 2　Risk by fault.

Component	Degree	Probability	Risk
CPU	4	0.08	0.32
CPU	4	0.08	0.32
CPU	4	0.08	0.32
CPU	2	0.08	0.16
PWM	4	0.03	0.13
…	以下略		…

表 3　影響度別リスク
Table 3　Risk by degree.

State	Degree	Probability	Risk
s_4	4	0.4734	1.8936
s_3	2	0.2169	0.4338
s_2	1	0.0013	0.0013
s_0, s_1	0	0.3084	0.0
	sum	1.0000	2.3287

る．このように EPS システム全体のフォールトの発生と影響の伝搬をエラーモデルとして記述した．バスである Wire_Harness_1 には，出力なしのみフォールトを記述した．一方，各コンポーネントに固有のフォールトを記述しているものもある．CPU を例にすると，加算器の異常などである．

エラーモデル内で確率的にフォールトが発生するのは，ハードウェアコンポーネントのエラーモデルに限った．ソフトウェアコンポーネント内部の確率的なフォールト発生はリスク分析システムに入力可能であるが，確立したモデル化手法がないと判断したためである．ソフトウェア・コンポーネントのエラーモデルは，入力値の異常やバインドされたコンポーネントの異常による動作やデータの内容をフォールトとして記述した．

フォールト発生率を，文献 [17] を参考にして，コンポーネントが表す部品の複雑さで決定されると仮定し，フォールト発生率を $1.0 \times 10^{(-4,-5,-6)}$ (1/hour) の 3 段階で設定した．つまり，CPU などの複雑な電子部品は高いフォールト発生率を与え，ワイヤーなどの機械的な部品しか含まないようなコンポーネントには低いフォールト発生率を与えた．また，システムの初期化時には経験的に故障が多いことが知られているため，電子部品に対して初期化時のフォールトを導入し，稼動中のフォールト発生率と区別した．初期化時のフォールト発生率はそのコンポーネントのフォールト発生率の 10 倍とした．

定量的な影響度を設定し，表 1 に示す．まず，安全規格 [1] を参考にして，最上位のシステムが人に与える危害を 4 段階で定量化した．次に，サブシステムである EPS システムの故障を上位システムの危害に当てはめた．EPS システムの故障を，暴走と停止，最小動作の三つとし，それぞれ 4，2，1 の影響度になるようにエラーモデルに与えた．

エラーモデルは，22 個のコンポーネントに付加し，合計 55 個の確率的なフォールト発生と 188 個の状態，226 個の遷移が存在する程度の規模となった．

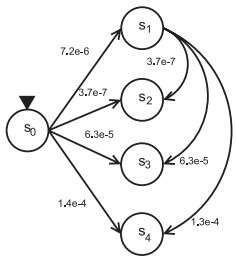

図 8　影響度別 CTMC
Fig. 8　CTMC by degree.

5.3　リスク分析

エラーモデル合成と簡約化を行うリスク分析システムを ruby で実装した．このシステムは図 5 の Risk Analysis System の破線に示すように，エラーモデルを入力として，フォールト別と影響度別のそれぞれのリスクを出力する．CTMC で各状態の確率を計算するためには，稼動時間 t を定める必要がある．自動車は 2 年に 1 度，全体的な検査と整備が行われるので，自動車に求められる稼動時間を 1 日 8 時間が 2 年間続くと仮定し，$t = 5840$ (hour) を与えた．

出力したフォールト別リスクを表 2 に，影響度別リスクを表 3 に示す．影響度に確率を掛けた数値がリスクである．これらの表は，図 5 のそれぞれ B と C に対応する．また，影響度別の CTMC を図 8 に示す．CTMC に状態名は存在しないが，表 3 と対応をとりやすいよう便宜的に付加した．図 8 の CTMC から式 (3) によって，表 3 の確率を計算している．

次に，簡約化による状態数削減の効果を検証す

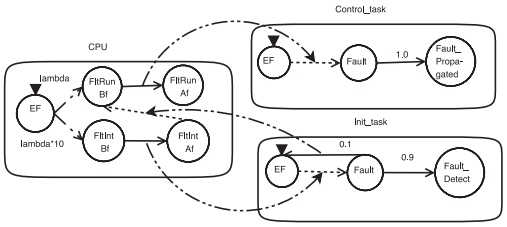

図 9　初期化タスクによる初期検査
Fig. 9　Initial checking on init_task.

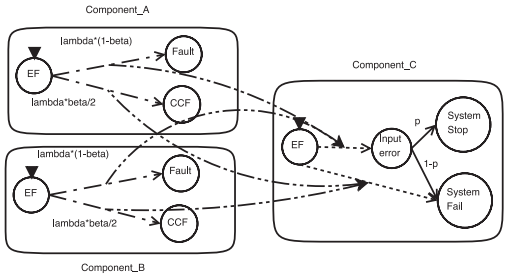

図 10　二重系のエラーモデル
Fig. 10　Error model of double modular redundancy.

る．合成対象 ($i \in I$) のエラーモデルの状態数を $|S_i|$ とし，故障原因の総数を n とする．このとき，エラーモデル合成後のエラーモデルの状態数 $|S|$ は，$\prod_{i\in I}|S_i| \geq |S| \geq \sum_{k=0}^{n} {}_n\mathrm{C}_k$ の関係になる．そのため，エラーモデル合成だけでは，状態数が $\sum_{k=0}^{54} {}_{54}\mathrm{C}_k \doteqdot 1.8 \times 10^{16}$ 以上になってしまう．一方，簡約化 (1)(2) を行うと，状態数は 104 に低下した．更に，簡約化 (3) を行うことで，フォールトが発生しても影響度がない状態の数を m としたとき，状態数は最大でも $(\sum_{k=0}^{m} {}_m\mathrm{C}_k) +$ (与えた影響度の数) に抑えられる．EPS システムにおいて $m = 2$ であり，状態数が 5 に低下した．EPS システムのリスク分析の処理時間は 10 秒程度であり，エラーモデルの個数を p，エラーモデルの平均状態数を q とすると，処理時間は $O(q^p)$ となる．そのため，処理時間の短縮に状態数の削減が非常に効果的である．

5.4 設計変更

システムの設計変更によるリスクの変化を検証するため，表 2 を参照してリスクが高いフォールトを優先的に対策した．設計変更の方法として，フォールト発生率が低いコンポーネントに交換する，またはフォールトを検知して影響度が小さい動作に変更することが考えられる．システム暴走の状態への遷移を，影響度が小さいシステム停止に遷移するように設計変更した．

フォールト検知には，以下を想定した．

- 特定動作時の検知
- 動作状態の監視
- 値の比較検査

システムには，初期化や終了処理などの動作状態が存在する．これら特定の動作状態でのフォールト検知を特定動作時の検知とする．特定動作時の検知としてよく行われるのが，システム初期化時の検査である．EPS システムの AADL モデルに初期化タスク (Init_task) による故障検知を導入した．図 9 に示す．

表 4　設計変更後の影響度別リスク
Table 4　Risk after modified by degree.

Degree	Probability	diff	Risk	diff
4	0.2376	−0.2358	0.9504	−0.9432
2	0.4727	+0.2558	0.9454	+0.5116
1	0.0012	−0.0001	0.0012	−0.0001
0	0.2885	−0.0199	0.0	±0
sum	1.0000	±0	1.8970	−0.4317

この図では，CPU のあるフォールトを Init_task によって 90%の確率で検知されることを表現している．検知に失敗した場合は，CPU のエラーモデルの状態が遷移し，次のコンポーネントへエラー伝搬によって影響が波及することを表現している．

動作状態の監視として，ウオッチ・ドッグ・タイマー (WDT) による MCU の停止や暴走の検知がある．EPS システムの AADL モデルに，WDT による監視を導入した．

値の比較検査として，センサの二重化を行った．多重系では，コンポーネントの故障に対する独立性が重要である．故障率 λ の二重系の独立性を機能安全規格 [14] を参考に，共通原因故障の発生率 $\beta(\lambda \leq \beta \leq 1)$ で表現した．$\beta = \lambda$ のとき，故障に対して独立であり，逆に $\beta = 1$ のとき，完全に従属である．二重化したアーキテクチャのエラーモデルを図 10 に示す．

故障検知率の設定は，機能安全規格 [14] を参考に High (99%)，Midium (90%)，Low (60%) のいずれかを与えた．

耐故障性を高めたシステムのリスク分析の結果を表 4 に示す．フォールト発生の検知後に故障の影響度を低下させる対策を行ったため，表 3 に比べてシステムとしてのリスクは表 3 の 2.3287 から，表 4 の 1.8970 へと低下した．一方，正常な状態 (degree = 0) にいる確率は表 3 の 0.3084 から，表 4 の 0.2885 とわずかに減少した．これは，コンポーネントを追加し

たことによりシステム故障を起こす確率が上昇していることを示している．

5.5 考　　察

自動車の重要なサブシステムである EPS を対象にリスク分析を行った．コンポーネントの粒度として，標準化されている AADL のコンポーネントを利用した．本論文では，エラーモデル合成と簡約化によって，粒度の細かいエラーモデルから粒度の粗いエラーモデルを作成することを提案している．この分析を他のサブシステムに適応し，その結果をシステム全体へ階層的に適応することで，自動車全体のような大規模かつ複雑な制御システムでも安全性が分析可能と考えられる．

本論文では，事例においてフォールト発生率を文献 [17] を参考に，3 段階で設定した．それにより，大量のフォールトが存在するシステムから，フォールト発生率と影響度の高いフォールトをリスクの大きさとして定量的に発見することができた点で適切であると考えられる．また，初期時のフォールト発生率をそのコンポーネントのフォールト発生率の 10 倍としたが，これも同様の理由により適切と考えられる．実際にはコンポーネントが表す部品を厳密に測定することでフォールト発生率を計算することが望まれる．しかし，この場合においても，リスクを定量的に計算できる提案手法の特長は変わらない．リスク分析システムでは，コンポーネントのフォールト発生率を与えて，システムのリスクを計算している．逆に，システムが許容できるリスクを定めることで，コンポーネントが許容できるフォールト発生率を求めることも考えられる．また，許容リスクとともにコスト制約を加えることで，システム構成の最適化を図ることが考えられる．

今後の課題として，一時的なフォールトへの対応と，ソフトウェアの確率的なフォールト発生のモデル化が挙げられる．また，大規模なシステムに適用できることをより示すために，リスク分析システムを改良することが必要である．改良には，エラー伝搬を AADL モデルにマッピングすることによる，より理解しやすい表示などが考えられる．

6. む　す　び

アーキテクチャ記述言語とエラーモデルによる設計と一体化したリスク分析を提案した．エラーモデルの合成と簡約化を定義し，コンポーネント故障を表現したエラーモデルから，システム全体の影響度別リスクとフォールト別リスクを計算するリスク分析システムを開発した．

適用事例として自動車の重要なサブシステムである電動パワーステアリング (EPS) システムの設計とリスク分析を行った．事例によりエラーモデルの合成と簡約化による計算量の削減効果を評価した．また，アーキテクチャレベルでの設計変更によるシステム全体のリスク低減効果を評価できることを示した．開発システムは，大規模な制御システムに対しても拡張可能であり，システム設計の上流工程でのリスク分析と対策に適用できる．

文　　献

[1] ISO/DIS 26262, “Road vehicles – Functional safety,” Technical Committee 22, 2009.

[2] D. Chen, R. Johansson, H. Lönn, Y. Papadopoulos, A. Sandberg, F. Törner, and M. Törngren, “Modelling support for design of safety-critical automotive embedded systems,” Computer Safety, Reliability, and Security, vol.5219, pp.72–85, Sept. 2008.

[3] A.E. Rugina, K. Kanoun, and M. Kaâniche, “A system dependability modeling framework using AADL and GSPNs,” Architecting Dependable Systems IV, vol.4615, pp.14–38, Aug. 2007.

[4] 吉村　悠, 日高隆博, 小林隆史, 手嶋茂晴, 中條直也, 高田広章, 高浜盛雄, “自動車制御システムのエラーモデル記述による安全性分析手法,” 信学技報, DC2008-64, 2008.

[5] OMG: Unified Modeling Language (UML) Specification, http://www.uml.org, 2009.

[6] OMG: Systems Modeling Language (SysML) Specification, http://www.sysml.org, 2008.

[7] V. Debruyne, F. Simonot-Lion, and Y. Trinquet, “EAST-ADL — An architecture description language,” IFIP International Federation for Information Processing, vol.176, pp.181–195, Oct. 2005.

[8] AS-5506A, “Architecture Analysis & Design Language (AADL),” SAE International, Jan. 2009.

[9] AS-5506/1, “SAE architecture analysis and design language (AADL) annex volume 1: Annex A: Graphical AADL notation, annex C: AADL meta-model and interchange formats, annex D: Language compliance and application program interface annex E: Error model annex,” SAE International, June 2006.

[10] S.-H. Wu, S.A. Smolka, and E.W. Stark, “Composition and behaviors of probabilistic I/O automata,” Theor. Comput. Sci., vol.176, no.1-2, pp.1–38, April 1997.

[11] N.A. Lynch and M. Tuttle, “Hierarchical correctness proofs for distributed algorithms,” Proc. 6th Annual ACM Symposium on Principles of Distributed Computing, 1987.

[12] A.N. Kolmogoroff, “Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung,” Math. Ann,

vol.104, pp.415–458, 1931.
[13] P. Buchholz, “Exact and ordinary lumpability in finite markov chains,” J. Applied Probability, vol.31, pp.59–75, 1994.
[14] IEC 61508, “Functional safety of electrical/electronic/programmable electronic safety-related systems,” International Electrotechnical Commission, July 1998.
[15] R.B. シナジ（著），今野紀雄，林　俊一（訳），マルコフ連鎖から格子確率モデルへ，シュプリンガー・ジャパン，東京，2001.
[16] 大川　進，本田　昭，よくわかる図解とシミュレーションで学ぶ自動車のモーションコントロール技術入門，山海堂，2006.
[17] Department of Defence, “Reliability prediction of electronic equipment,” MIL-HDBK-217F, 1991.

（平成 22 年 1 月 8 日受付，5 月 9 日再受付）

吉村　悠

2010 名古屋大学大学院情報科学研究科博士前期課程了．在学中に高信頼アーキテクチャの研究に従事．

日高　隆博

2002 名古屋大学大学院工学研究科博士後期課程単位取得退学．名古屋大学大学院情報科学研究科附属組込みシステム研究センター研究員．ソフトウェア工学・車載ソフトウェアプラットフォームの研究に従事．

中條　直也　（正員）

1982 名古屋大学大学院工学研究科博士前期課程了．豊田中央研究所を経て，現在，愛知工業大学情報科学部情報科学科教授．自動車用の電子機器，組込みシステム等の研究に従事．博士（工学）．情報処理学会，電気学会，IEEE 各会員．

高田　広章　（正員）

名古屋大学大学院情報科学研究科情報システム学専攻教授．1988 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同専攻助手，豊橋技術科学大学情報工学系助教授等を経て，2003 年より現職．2006 年より大学院情報科学研究科附属組込みシステム研究センター長を兼務．リアルタイム OS，リアルタイムスケジューリング理論，組込みシステム開発技術等の研究に従事．オープンソースの ITRON 仕様 OS 等を開発する TOPPERS プロジェクトを主宰．博士（理学）．IEEE，ACM，情報処理学会，日本ソフトウェア科学会各会員．

高浜　盛雄

1971 北海道大学大学院工学研究科電気工学専攻修士課程了．三菱重工業（株）名古屋航空機製作所を経て，2001 愛知工科大学教授．2004 名古屋大学大学院情報科学研究科教授，2010 年 3 月定年退職．ロバスト飛行制御システム，自律再構成型高信頼制御システムの研究に従事．工博．計測自動制御学会，システム制御情報学会各会員．