特集論文

# 高信頼性を実現するシステム仕様検査技術

上野浩一郎*
磯田　誠*
市原利浩*

*System Specification Verification Tool for Highly-dependable System*

*Koichiro Ueno, Makoto Isoda, Toshihiro Ichihara*

**要　旨**

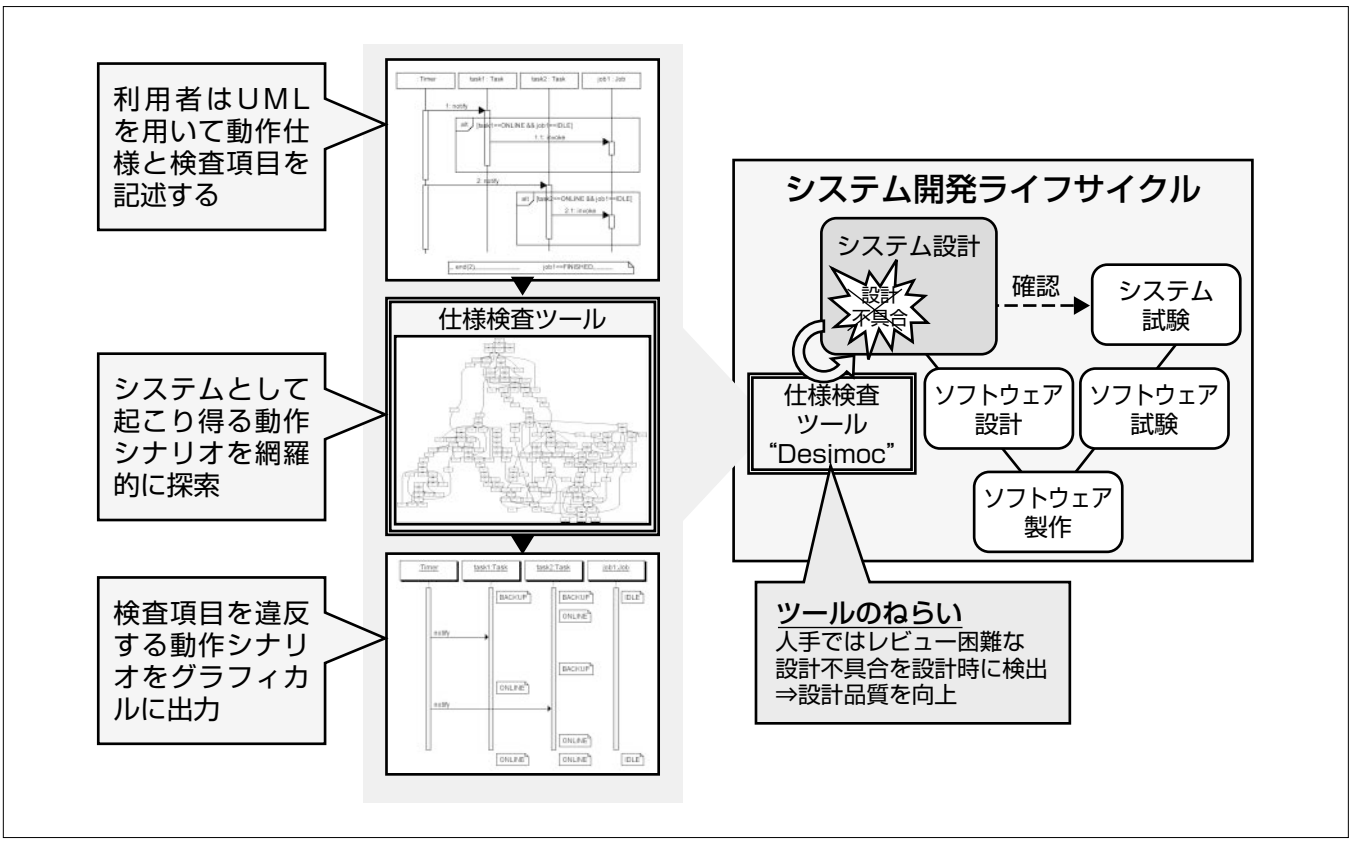
近年，社会活動が情報システムへの依存度を増しているため，情報システムに対して一層の信頼性が求められている。しかし従来のシステム開発では，システム仕様を人手でレビューしており，設計不具合が設計工程から流出する可能性があった。さらにシステム試験では，流出した設計不具合を含む仕様を正しいとして試験するため，不具合を検出できない危険がある。したがってシステムの高信頼性を実現するには，システム仕様中の設計不具合を網羅的に摘出する必要がある。

そこで三菱電機では，設計品質の向上を目的に，システム仕様を網羅的に自動検査するツール“Desimoc”を開発した。システム設計者は，Desimocに対してシステムの動作仕様と，システムが満たすべき検査項目を入力する。Desimocは入力された動作仕様を基に，システムとして起こり得る動作シナリオを網羅的に探索し，検査項目に反する動作シナリオを検出する。Desimocで扱う検査内容を次に示す。

①到達可能性（初期状態から特定状態に到達できる）
②安全性（デッドロックや無限ループがない）
③活性（いつか必ず特定状況が発生する）
④公平性（各々の並行処理が必ず進む）

Desimocが網羅的に動作シナリオを探索するため，人手のレビューでは検出困難な設計不具合も検出可能である。

今回，Desimocを開発済みシステム仕様に対して試行適用し，マニュアル運用で回復すべき稀（まれ）な異常動作を検出できた。この試行によってDesimocで設計不具合をなくし，システムの信頼性が向上することを確認した。




**システム仕様を検査するツール“Desimoc”**

Desimocは，人手ではレビュー困難な設計不具合をシステム設計時に検出するためのツールである。Desimocの利用者はUML（Unified Modeling Language）を用いて，検査対象システムの動作仕様と検査項目を記述する。Desimocはこれらを入力して，システムとして起こり得る動作シナリオを網羅的に探索する。そして，検査項目を違反する動作シナリオをグラフィカルに出力する。設計者はDesimocが出力した動作仕様から設計不具合を修正する。

*情報技術総合研究所

## 1. まえがき

近年，社会活動は情報システムに依存し，障害が発生した場合の経済的かつ人的損失の影響は非常に大きい。しかしシステムが大規模化・複雑化した結果，開発時に不具合を洩(も)れなく発見することが従来に比べ困難になっている。

これに対して経済産業省は，システムに関係する組織が遵守すべき“情報システムの信頼性向上に関するガイドライン”[(1)]を公表した。このガイドラインでは，高信頼性を実現する手段として“形式手法”の適用を推奨している。

当社は，情報システムを高信頼化するために，形式手法の一つであるモデル検査技術を用いて，システム設計段階で仕様の正しさを検査するツール“Desimoc”を開発した。本稿ではDesimocの概要と適用例について述べる。

## 2. 仕様検査技術

### 2.1 形式手法とモデル検査

形式手法とは，計算機科学における論理学や離散数学に基づいて，システムの仕様を記述し検証する技術の総称である。本稿で述べる仕様検査ツールDesimocは，形式手法の一つであるモデル検査技術を採用している。モデル検査とは，オートマトン／時相論理／グラフ理論を用いて，システム動作を網羅的に検査する技術である。モデル検査を適用できる対象例を次に示す。

①システムを構成するサーバ間のメッセージシーケンス
②Webサービスや分散オブジェクト間の呼び出しシーケンス
③通信装置と対向装置間の通信プロトコル仕様

モデル検査技術の実システム開発への適用は欧米が先行しているが，最近では国内でもこの技術に取り組み始めている。例えば宇宙航空研究開発機構では，宇宙機の制御ソフトウェアの検査に適用している[(2)]。また産業技術総合研究所では，モデル検査の理論，利用プロセス，ツールなどの体系をMCBOK2008(Model Checking Body Of Knowledge)として公開した[(3)]。

### 2.2 仕様検査ツールのねらい

本稿で述べる仕様検査ツールは，モデル検査技術を用いて，システム設計時に仕様の正しさを検査するツールである。従来のシステム開発では，システム仕様を人手でレビューしていた(**図1**)。これでは設計不具合が下流工程に流出する可能性があった。設計不具合が信頼性にとって問題なのは，流出した設計不具合を含む仕様を正しいとしてシステム試験するため，試験でも不具合を検出できない危険が増す点である。仕様検査ツールのねらいは，設計品質を向上させることである。仕様検査ツールを用いたシステム開発では，ツールが仕様を網羅的に検査するので，設計不具合が下流工程へ流出することを抑止できる。

## 3. 仕様検査ツール“Desimoc”

### 3.1 ツール機能

Desimocは，**図2**に示す①②③の順に動作する。

(1) 仕様モデルの入力(**図2**①)

開発者が作成した“仕様モデル”を入力する。仕様モデルとは，システムがどのように振る舞うかを記述した“動作仕様”と，システムが常に満たすべき制約である“検査項目”である。検査可能な項目は次の4種類である。

①到達可能性(初期状態から特定状態に到達できる)
②安全性(デッドロックや無限ループがない)
③活性(いつか必ず特定状況が発生する)
④公平性(各々の並行処理が必ず進む)

(2) 動作シナリオの探索(**図2**②)

(1)で入力した動作仕様に対して，実行タイミングの組合せによって様々な動作シナリオが起こり得る。これらの動作シナリオを，Desimocは重複なく網羅的に探索する。

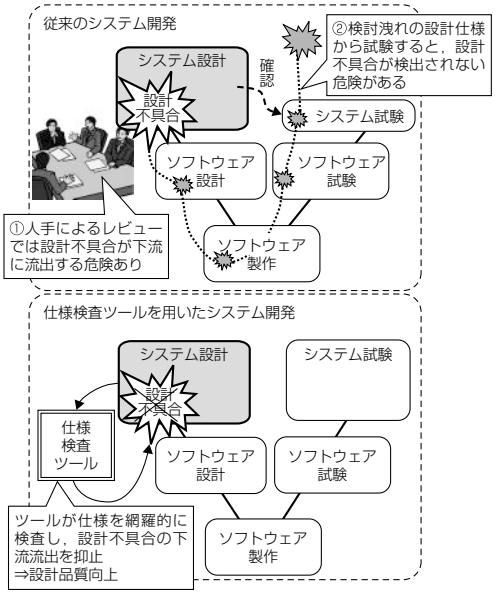



**図1．仕様検査ツールのねらい**

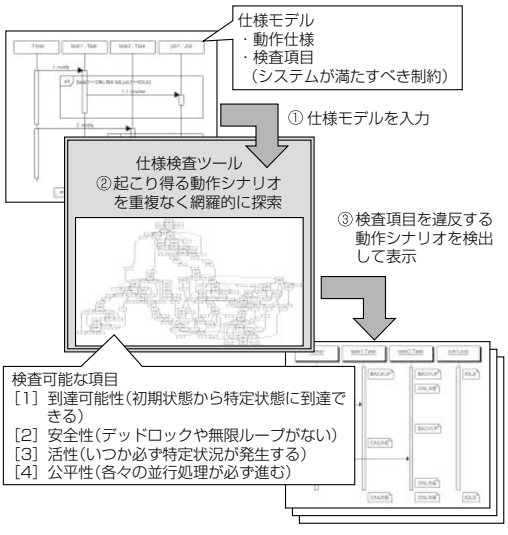



**図2．仕様検査ツール“Desimoc”の動作概要**

“動作シナリオを重複なく網羅的に探索”する点が，システム動作を確率的に模擬するシミュレーションとの違いである。

⑶　検査項目違反の動作シナリオの出力(**図2**③)

⑵で探索した動作シナリオのうち，⑴で入力した検査項目を違反する動作シナリオを検出して，グラフィカルに表示する。

### 3.2　利用方法と効果

Desimocの利用方法と効果は，**図3**に示すように三つある。

⑴　設計不具合の修正

システムでは様々な動作シナリオが起こり得るため，すべてを人手でレビューするのは困難である。Desimocが出力する検査項目を違反する動作シナリオを確認することによって，稀なタイミングでしか発生しない動作の検討洩れを抑止し，設計品質を向上させる。

⑵　異常時の回復仕様の追加

検査項目を違反する動作シナリオの中には，例えばネットワークの回線障害など，それが発生しないことを前提にできない内容がある。この場合，検査項目を違反する動作シナリオに対して，システム設計で“異常時の回復仕様”を追加し，この仕様に対してシステム試験で“異常系の試験ケース”を作成する。Desimocは網羅的に検査項目を違反する動作シナリオを出力するので，異常系の試験ケースの網羅性が向上し，試験品質を向上させる。

⑶　拡張開発時の仕様確認

情報システムの拡張開発時，仕様モデルの動作仕様は拡張のため変更されるが，検査項目はシステムとして本質的なため不変であることが多い。この場合，仕様変更した動作仕様と，前回開発の検査項目をDesimocに入力して，検査項目を違反する動作シナリオがないことを確認すべきである。このように，仕様変更に伴う悪影響の有無を設計時に確認することによって，安全なシステム移行を実現する。

### 3.3　Desimocの特長

従来の仕様検査ツールの利用者は，**図4**に示すように，固定的に埋め込まれた汎用(はんよう)記法に則(のっと)って，仕様モデルを記述し，出力された動作シナリオを解釈する必要があった。この場合，記法が汎用的過ぎるため，開発者にとってツールを使いこなすハードルが高かった。

Desimocの特長は，入力する仕様モデルと出力する動作シナリオの記法を，開発者が担当する分野の特徴を反映した記法に交換してプラグイン可能な点である。このようにプラグイン可能な記法をドメイン特化記法と呼ぶ。特定分野向けに記述しやすく読みやすいドメイン特化記法を，Desimocにプラグインして利用することによって，開発者とツール間のギャップを解決する。

## 4.　適　用　例

### 4.1　適用対象

仕様検査ツールDesimocの効果を確認するために開発済みシステムに試行適用した。対象はすでに稼働中の社会インフラシステムであり，システム仕様書からいくつか仕様を抜粋して，Desimocを用いて仕様検査した。次に，説明のために単純化した仕様を用いてDesimocの入出力例を紹介し，適用結果について述べる。なおDesimocの入出力例は，この適用対象向けに新たに発案した“分散システム向けのドメイン特化記法”に基づいている。

### 4.2　入力する仕様モデル

**図5**は，検査対象とした“タイマによるジョブ起動”に対する仕様モデルである。動作仕様として，Timerはtask1とtask2にnotifyを送信し，その際の内部状態に応じてtask1又はtask2がjob1にinvokeを送信する。検査項目は“job1はFINISHED”になることである。次に記法について述べる。構成要素間のメッセージ送受信は，UML[注1)⑷]シーケンス図(**図5**①)で記述する。各構成要素の状態遷移はUML状態マシン図(**図5**②③)で記述する。検査項目は，UMLシーケンス図中にノート(**図5**④)で記述する。検査項目における“end(2)”とはメッセージ番号2のnotifyが実

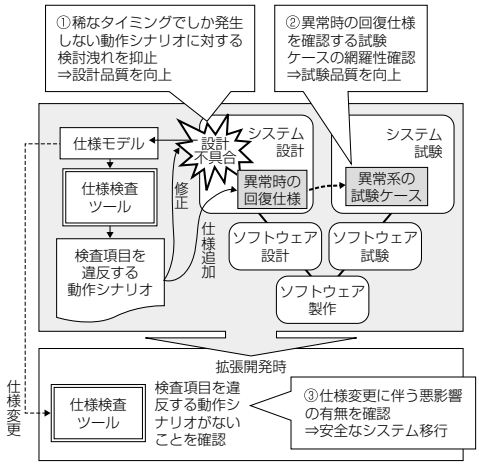



**図3．仕様検査ツール“Desimoc”の適用効果**

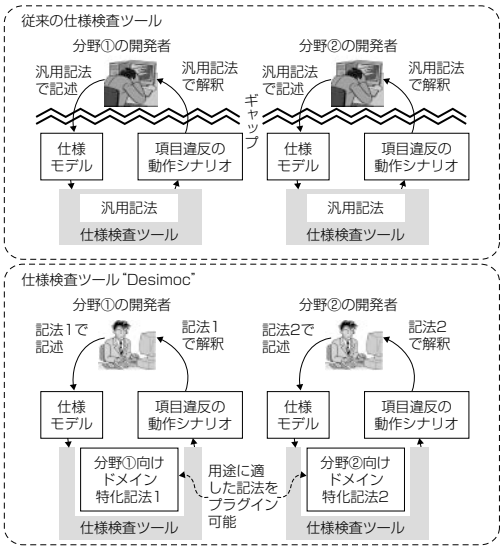



**図4．仕様検査ツール“Desimoc”の特長**

行終了した時点を示す。したがってこの検査項目は“メッセージ番号2のnotifyが実行終了した後はいつか必ずjob1がFINISHEDになる”ことを意味している。このようにメッセージ番号を使って検査項目を記述可能な点がこのドメイン特化記法の特長である。**図5**に示したend(実行終了)以外にも“メッセージ送信時点”“メッセージ受信時点”を指定可能である。

**図5**の仕様モデルでは，Taskの内部状態がBACKUPとONLINEを任意のタイミングで遷移(**図5**②)し，この内部状態に依存してjob1に対するinvoke送信が決定する(**図5**①)。このため状態に応じて様々な動作シナリオが起こり得る。なお**図5**①のシーケンスはすべて同期メッセージなのでメッセージ送受信の順番は固定である。一方，非同期メッセージの場合は送受信のタイミングが非決定的なので更に多くの動作シナリオが起こり得る。Desimocはこのような非同期メッセージの検査にも対応している。

(注1) UMLは，Object Management Group Inc. の登録商標である。

### 4.3 出力する動作シナリオ

**図6**は，**図5**の仕様モデルで起こり得る動作シナリオのうち，検査項目を違反する動作シナリオの一例である。この動作シナリオは，Timerがnotifyを送信した際，task1とtask2がいずれもBACKUPであるため，job1にinvokeを送信せず，job1の内部状態がIDLEのままとなっている。job1がFINISHEDではないので検査項目を違反していることが分かる。なお**図6**のドメイン特化記法は，UMLシーケンス図に状態遷移を組み合わせて表示するものに拡張している。具体的には，メッセージ送受信(**図6**①)と，各構成要素の内部状態(**図6**②③④)を組み合わせて表示する構成を採用した。

### 4.4 適 用 結 果

開発済みシステムへの試行適用によって，システムの高信頼化に次の2点で役立つことを確認した。

(1) 適用対象システムはすでに長期稼働中で，仕様が成熟していたため，新たな設計不具合は検出しなかった。ただしDesimocを用いることによって，マニュアル運用でシステム異常を回復すべき稀(まれ)な動作シナリオを検出できることを確認した。これらは，運用マニュアルの改善につなげる。

(2) モデル化記法として，汎用(はんよう)的で自動検査できないUMLではなく，この適用対象に合わせたドメイン特化記法を発案して用いた。このドメイン特化記法を用いて様々な仕様変更を記述して実験することによって，拡張開発時の仕様変更による影響の有無を容易に確認できることを確認した。

## 5. む す び

システムの信頼性を向上させる仕様検査ツールDesimocについて，その概要と適用例について述べた。今回は，開発済みシステムに対する試行を述べたが，現在は新規開発システムに対して適用評価を進めている。

高信頼なシステム実現のため，次に必要なのはプログラム品質の向上である。今後は，プログラムの実行シナリオを網羅的に探索し，稀にしか発生しないケースを試験可能とするツールを開発する。これらのツールによって，設計から試験のライフサイクル全体で信頼性向上を図っていく。

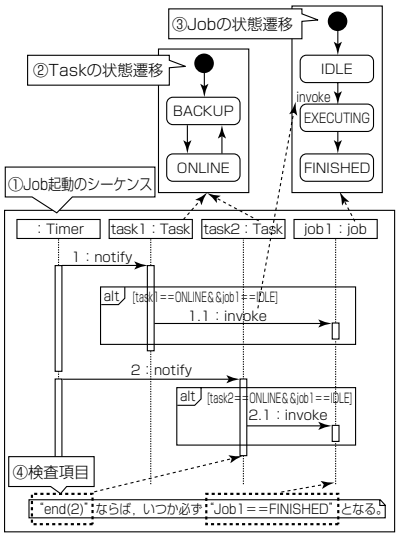



**図5．仕様検査ツール“Desimoc”の入力例**

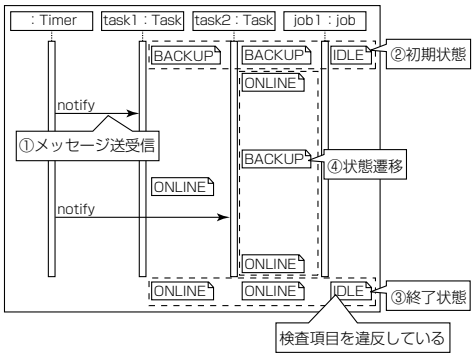



**図6．仕様検査ツール“Desimoc”の出力例**

### 参 考 文 献

(1) 経済産業省：情報システムの信頼性向上に関するガイドライン第2版（2009）
http://www.meti.go.jp/press/20090324004/20090324004-4.pdf

(2) 宇宙航空研究開発機構：「ソフトウェア独立検証と有効性確認」技術の研究
http://stage.tksc.jaxa.jp/jxithp/

(3) 西原秀明，ほか：MCBOK2008：ソフトウェア開発のためのモデル検査知識体系，産業技術総合研究所（2009）
http://unit.aist.go.jp/cvs/tr-data/PS2009-009.pdf

(4) OMG：Unified Modeling Language
http://www.uml.org