Vol.2015-SE-189 No.17
2015/7/23

ソフトウェアの故障木解析手法の研究

高橋正和[†1]　　小坂　力[†1]　難波礼治[†2]

本論文では，ソフトウェアの故障木解析（FTA：Fault Tree Analysis）手法を提案する．今までに，さまざまなソフトウェアに対する FTA 手法が提案されてきた．しかし，それらはソフトウェアの構造に対応した故障木（FT: Fault Tree）のひな形を準備し，それらを組み合わせることで FT を作成するという提案にとどまっていた．そのため，具体的な FT 作成手順が不明確であった．そこで本論文では，ソフトウェア構造に対する FT のひな形の構造を見直すとともに，FT の作成手順を定義することで，技術者の能力に依存せずに FT を作成できるようにした．本論文で提案する FTA 手法をスピン衛星の回転速が速くなりすぎる故障の原因分析に適用した結果，経験の浅い技術者でも適切な FT を作成することができた．

A Study for Software Fault Tree Analysis

MASAKAZU TAKAHASHI[†1]　RIKI KOSAKA[†1]
REIJI NANBA[†2]

This manuscript proposes a Fault Tree Analysis (FTA) method for software. Various FTA methods have already been proposed. The concept of those FTA methods is followings: preparing Fault Tree (FT) templates, and developing whole FT by combining those FT templates. Those are insufficient for developing FT, because a concrete procedure of developing FTA was unclear. Proposed FT method makes the inexperienced engineers develop appropriate FT by preparing adequate FT templates and defining a concrete procedure for developing whole FT. As a result of applying the proposed FTA method to the fault that a spinning satellite rotates too fast, inexperienced engineer could develop appropriate FT as well. We confirmed that the differences between FT that were developed by different engineers were reduced.

1. はじめに

今日，自動車，プラント，航空宇宙機器等の機械製品は，プログラムにより制御されている．この様なプログラムを制御プログラムと呼ぶ．制御プログラムに故障が発生すると，機械製品本体だけでなく，使用者，周囲の環境にも多大な被害を与える場合がある．そのため，機械製品の制御プログラムに対しては，要求に応じて設計と使用の段階における安全性を検討することが重要である[1].

安全性を検討する方法は，設計段階で網羅的に故障の可能性を検討して原因を明らかにする方法と，使用段階の特定の故障の原因を明らかにする方法に大別できる．前者について，高橋らは，Failure Mode and Effects Analysis (FMEA)を用いて，医薬品製造設備の制御に用いるプログラムに関わるリスクマネジメント方法を提案した[2]．そして Reese らは，Software Deviation Analysis (SDA)を用いて，プラントの制御プログラムの安全性を解析する方法を提案した[3]．これらの方法は，特にプラント，航空宇宙機器の制御プログラムへの適用が行われている．後者については，コードレベルで安全性を解析する方法とモジュールレベルで安全性を解析する方法に大別できる．コードレベルの解析方法として，Weber らは，アセンブラ言語で記述された航空機の制御プログラムに対して Fault Tree Analysis (FTA)を用いて故障原因の分析を行った[4]．Friedman らは，Pascal 言語で記述されたプログラムの故障に対する Fault Tree (FT)を自動作成する手法を提案した[5]．さらに，Leveson らは，プログラムの基本命令に対する FT の雛形を準備して，FT の雛型を組み合わせることでプログラム全体に対する FT を作成する方法を提案した[6]．モジュールレベルの解析方法として，Pai らは航空機の飛行統括監視システムの Unified Modeling Language で記述された設計文書から FT を作成し，システムの信頼性を求める方法を提案した[6]．Li らは，ソフトウェア部品毎とそれに対応した FT を準備して，ソフトウェア部品を組み合わせて作成したプログラムの FT を効率的に作成する方法を提案した[7]．Zhao らは，制御プログラムの Data Flow Diagram，Control Flow Diagram，構造図を入力として，モジュールレベルでの故障に関する FT を作成する方法を提案した[8]．しかし，これらの方法は，故障毎に原因分析を行うため時間がかかること，具体的な FT 作成手順が定義されていないこと，プログラムを構成するモジュールの起動の方法とタイミング，割り込みの禁止と禁止解除のタイミング，変数のスコープ等を考慮しなければならないこと等の理由により制御プログラムへの適用が進まなかった．

本研究では，C 言語やアセンブラで記述された制御プログラムに対して，必要な FT の雛形の準備，制御プログラムの FT を作成しやすくするための前処理，そして FTA 作成手順の規定を行うことで，故障に対する FT を作成する方法を提案する．

†1 山梨大学
University of Yamanashi
†2 第一工業大学
Daiichi Institute of Technology

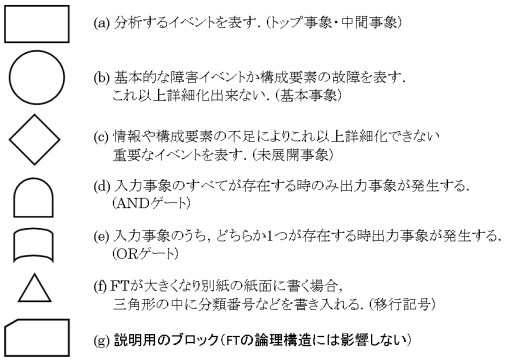

図 1　FT で使用する記号

Figure 1 Symbols for Describing Fault Free

2. Fault Tree Analysis の概要

2.1 ハードウェア向けの Fault Tree Analysis の概要

FTA の目的は，分析対象となるハードウェアやシステムに発生する望ましくない事象（故障）を引き起こす原因を摘出することである．FTA では故障を引き起こす原因をシステム（上位）レベルから構成要素（下位）レベルに向けて段階的に展開することで根本的な故障の原因を突き止める．原因を段階的に展開する様子は事象記号と論理記号を用いて図形的に記述する．図 1 の(a)から(f)に FT 記述に使用する記号を示す．以下に，ハードウェアの FTA の手順の概要を述べる．はじめに，ハードウェアの構成要素や使用環境を参考にしながら分析対象とする故障を決定する．この故障をトップ事象と呼ぶ．二番目に部品構成や使用環境を参照しながら，トップ事象を引き起こす直接の原因を明らかにする．これを中間事象と呼ぶ. 本論文では, 便宜上, これを 1 次中間事象と呼ぶ．その際，事象記号と論理記号を用いてトップ事象と 1 次中間事象に関する FT を記述する．三番目に 1 次中間事象を生じさせる原因を明らかにする．これを 2 次中間事象と呼ぶ．この作業を，原因が，それ以上，展開できなくなるまで繰り返す．最後に現れた原因（n 次中間事象）が故障の根本的な原因となる．これを基本事象と呼ぶ[10].

2.2 制御プログラム向けの Fault Tree Analysis の概要

制御プログラムでは，逐次的な命令実行の流れが，構成要素とその構造に相当する．プログラムは代入, if-then-else, while，function call の基本的な命令の組み合わせて作成されているので，事前にそれらの命令に対応した FT 雛形を準備する．そして，故障を生じた命令から，故障に至った命令実行の流れを遡りながら, FT 雛型を組み合わせていくことで，故障の原因を追及する．以下に FTA の作業手順の概要を示す.

はじめに，分析対象となる故障を決定する．制御プログラムの場合，想定される故障には以下のようなものがある：出力される値の異常，function やプログラムの誤ったタイミングでの実行，function やプログラムの未実行．次にその故障を引き起こす命令を同定する．三番目に故障の 1 次中間事象を検討する．1 次中間事象としては，該当するアルゴリズムの不備，該当する命令への入力値の異常等が考えられる．前者の場合には，該当するアルゴリズムを修正する．後者の場合には，さらに分析を続ける．1 次中間事象を引き起こす入力値を計算している命令を同定する．そして，故障と 1 次中間事象の関係を，FT 雛形と図 1 の論理記号を用いて記述する．四番目に 1 次中間事象を引き起こす 2 次中間事象を検討する．2 次中間事象の原因とその対処方法も一次中間事象と同様である．そして 1 次中間事象と 2 次中間事象の関係を, FT 雛形と図 1 の論理記号を用いて記述する．以降は，同様の手順により，n-1 次中間事象と n 次中間事象の関係を, FT 雛形と図 1 の論理記号を用いて記述する．この作業を n 次中間事象が，それ以上展開できなくなるまで繰り返す．最後に現れた n 次中間事象が基本事象となる.

図 2 に「if a ≧ b then x := 5　else x := g(x)」というプログラムを実行した後，「x > 10」という故障が生じる場合の FT を作成する．このプログラムの基本構造は if-then-else 文なので後述する if-then-else_if-else 命令の FT 雛形を使用する．はじめに，if-then-else_if-else のひな型の 1 行目に故障「x > 10」を記述する．次に 2 行目に if-then-else_if-else の条件節が true となる場合と false となる場合の一次中間事象を記述する．条件節が true となる場合の一次中間事象は「a ≧ b」かつ「x := 5」である．条件節が false となる場合の一次中間事象は「a not ≧ b」かつ「x := g(x)」である．三番目に原因を分析する．条件節が true の場合は「x > 10」の故障が生じるないので，追加の分析は行わない．条件節が false の場合は「x > 10」の故障が生じる原因として「x := g(x)」が 10 より大きいことが考えられる．従って，故障「x > 10」の基本事象は g(x)が 10 より大きい値を返すことになる．　さらに，必要があれば，g(x)の内容を分析する.

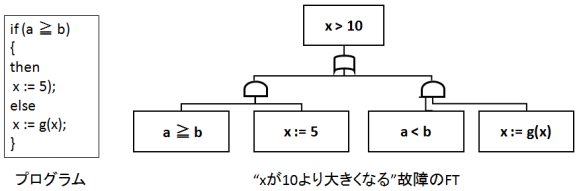

図 2　プログラムの FTA の結果の例

Figure 2 Example of FTA Result for Simple Program

3. 提案する Fault Tree Analysis 手法

提案手法では，　FT 雛型を整備し，FTA を容易にするための前処理を行い, 定義した FT 作成手順に従って FT を作成できるようにする．各節では，それらの説明をする.

3.1 FT 雛型の準備

プログラミング言語には, while 文と for 文, if 文と switch 文のように類似した機能を有する命令が存在する. Leveson らはプログラミング言語の特徴に応じて命令毎に FT 雛型

を準備する方法を提案している．この方法では命令に対して機械的に FT 雛型を適用することが可能だが，プログラミング言語毎に FT 雛型を準備する必要がある．そこで本研究では,類似した機能を有する命令を 1 種類に置き換え，置き換えた命令に対する FT 雛型を準備することで，様々なプログラミング言語に対応できるようにする．提案手法では，C 言語やアセンブラ言語で記述された既存の組込みプログラムを分析し，以下の 7 種類の FT 雛型を準備した．

(1) 代入文の雛型

図 3 に代入文の FT 雛形を示す．この雛型はプログラム中の代入文で故障あるいは中間故障が生じる場合に適用する．代入文が故障を引き起こすケースには,「代入した値が故障を引き起こす」場合，あるいは「代入文自体の誤りが故障を引き起こす」場合がある．

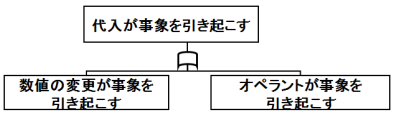

図 3 代入命令の FT 雛形

Figure 3 FT Template for Substitution

(2) if-then-else_if-else 文の雛型

図 4 に if-then-else_if-else 文の FT 雛形を示す．この雛型はプログラム中の if-then-else_if-else 文中で故障あるいは中間故障が生じる場合に適用する．if-then-else_if-else 文が故障を引き起こすケースには，「if (条件)が成立し，かつ，then 節内の命令群が故障を引き起こす」場合，「else_ if (条件)が成立し，かつ， else_if 節内の命令群が故障を引き起こす」場合，あるいは「 else 節内の命令群が故障を引き起こす」場合がある．なお, else_if 節が複数個ある場合には，この部分の FT の枝を増やす．

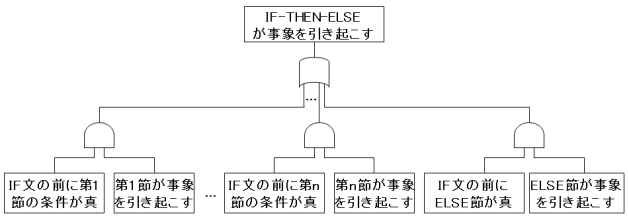

図 4 if-then-else_if-else 命令の FT 雛形

Figure 4 FT Template for if-then-else_if-else instruction

(3) while 文の雛型

図 5 に while 文の FT 雛形を示す. この雛型はプログラム中の while 文中で故障または中間故障が生じる場合に適用する．while 文が故障を引き起こすケースには,「while 命令が実行されない，かつ，while 命令以前の命令が故障を引き起こす」場合，あるいは「while 命令が複数回実行される，かつ，while 命令節の中で故障を引き起こす」場合がある．

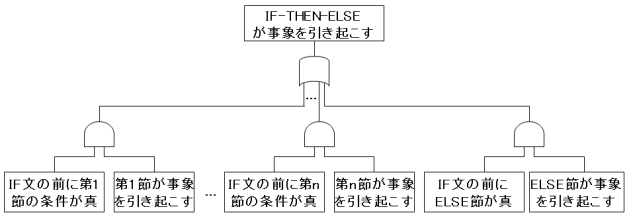

図 5 while 命令の FT 雛形

Figure 5 FT Template for While Instruction

(4) モジュール呼出文の雛型

図 6 にモジュール呼出文の FT 雛形を示す．この雛型はプログラム中のモジュール呼出文中で故障あるいは中間故障が生じる場合に適用する．モジュール呼出文が故障を引き起こすケースには,「モジュール呼出文が実行される，かつ，モジュール内で故障が生じる」場合，あるいは,「モジュール呼出文が実行されない，かつ，実行されないために故障が生じる」場合がある．前者の場合には，引き続きモジュール内の FTA を実施する．なお，ここで，モジュールとはサブルーチン，関数等とする．

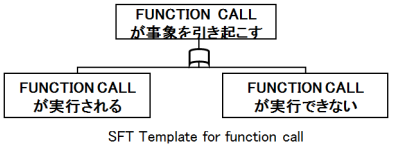

図 6 function call 命令の FT 雛形

Figure 6 FT Templates for Function Call

(5) 割込みルーチンの FT 雛型

図 7 に割込みルーチンの FT 雛型を示す．この雛型は割込みルーチン中で故障あるいは中間故障が生じる場合に適用する．割込みルーチンが故障を引き起こすケースには,「割込みが発生する，かつ，割込みルーチン内で故障が生じる」場合,「割込みが発生しない，かつ，割込みルーチンが実行されないために故障が生じる」場合，あるいは「割込み禁止が設定されている」場合がある．

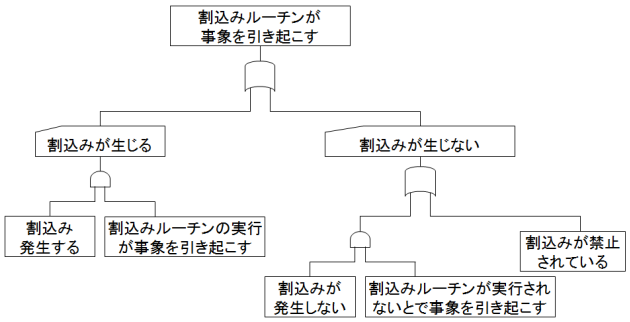

図 7 割込みルーチンの FT 雛型

Figure 7 FT Template for Interrupt Routine

(6) 命令が実行できない場合の FT 雛型

図 8 に命令が実行できない場合の FT 雛形を示す．この

雛型はモジュール内の特定の命令が実行できないために故障あるいは中間故障が発生する場合に適用する．module 内の n 番目の命令が実行できない理由として，1 番目から n-1 番目の命令について実行できない（途中でプログラムが停止する）可能性があるか検討する．さらに，module 自体が実行できない可能性があるか検討する．実行できない可能性のある命令については，さらに FTA を実施する．

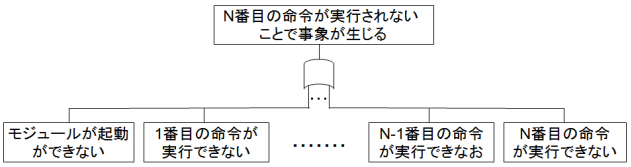

図 8　命令が実行できない場合の FT 雛形

Figure 8 FT Template for Non-Execution of Instructions

(7)　広域変数の FT 雛型

図 9 に広域変数の FT 雛型を示す．この雛型は，(1)から(6)の FT 雛型中で広域変数が使用されたことで故障または中間故障が生じる場合に適用する．広域変数が値を設定される全ての場所に対して FT の枝を作成する．全ての枝に対して以降の FT を作成する．

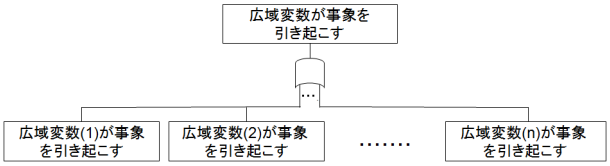

図 9　広域変数の FT 雛型

Figure 9 FT Template for Global Variables

3.2　FT を作成しやすくするための前処理の実施

前節で提案した FT 雛形を組み合わせて FT を作成するためには変数の追跡が必須である．加えて，複雑な階層構造をもつ命令に対して FT 雛型を適用することが必須である．そこで FT 実施前に FTA 対象プログラムに対して以下の処理を行う（以降，これらをまとめて前処理と呼ぶ）．

(1)　プログラムの実行に関わる情報リストの作成

FTA では，分析者がプログラムを見ながら，各命令で使用される変数の値を追跡する．そこで，FTA 実施前に以下の処理を行い，命令間で受け渡しをされる変数の値を追跡しやすいように広域変数と局所変数の識別，変数の使用や代入の区分，モジュールの実行形態(周期起動，割込み起動)の識別，割込み禁止と禁止解除のタイミングの識別　等を行う．広域変数と局所変数の識別では，全ての変数が広域変数か局所変数であるかを識別する．変数の使用や代入の区分では，個々の変数に対して，その変数の値を用いて計算をしている（使用する変数）のか，その変数が値を設定されている（代入される変数）のかを識別する．モジュールの実行形態の識別では，モジュールが周期処理，あるいは割り込み処理で実行されているのかを識別する．割込み禁止と禁止解除の識別では，割込み禁止の期間を明らかにして広域変数の更新の有無を明らかにする．

(2)　複雑なプログラム構造の分析

プログラムには while ループの中に if 文が入っているような階層構造を有するものがある．この場合，3.1 で作成した FT 雛型をそのまま適用することはできない．階層構造を有する命令で故障が生じた場合，故障は階層構造の内側から外側に向かって伝搬するので，階層構造の内側の命令から外側の命令に向けて FT 雛型を順次適用していく．FT 実施前にこのような命令の階層構造を明らかにする．なお，一番内側の階層を構成する命令を一次階層命令と呼ぶ．同様に，二番目に内側の階層を構成する命令を識別し，それを二次階層命令と呼ぶ．以降，一番外側（n 番目）の階層を構成する命令まで識別し，n 次階層命令と呼ぶ．

3.3　FT 作成ルールの制定

3.2 で述べた前処理を行ったプログラムの故障に対して FT を作成する．図 10 に示す FT 作成ルールを説明する．はじめに STEP1 で，FTA の対象とする故障（トップ事象）を決定し，故障を生じる命令を識別する．次に STEP2 a で，故障を生じる命令に応じた FT 雛型を準備する．故障を生じる命令が階層構造となっている場合には，故障を生じる命令が入っている階層の FT 雛型を準備する．さらに，故障には命令が実行された結果生じる故障（例えば，入力値が大きいため出力値が大きくなり，それを出力したために故障が生じる）と，命令が実行できない結果生じる故障（例えば，WHILE 文の条件節が False とり，命令群が実行されないために前回計算した値を出力したために故障が生じる）があるので，三番目に STEP2 b で，これらの場合を検討する．命令が実行できない結果生じる故障の場合は，必要に応じて「命令が実行できない場合の FT 雛型」を準備して FT の枝を追加する．四番目に STEP2 c で，STEP2 b と STEP2c で準備した FT に故障の条件や原因（1 次中間事象）を記入する．五番目に STEP3 a で，STEP2 c で記入した故障の条件や原因の全てに対して，使用する変数が広域変数か局所変数かを識別する．使用する変数が広域変数の場合は，STEP3 b-1 を実行し，広域変数の FT 雛型を準備し，全ての使用する変数に値を設定している命令を記述する．使用する変数が局所変数の場合は，STEP3 b-2 を実行して，直前の使用する変数に値を設定している命令を識別する．STEP3 b-1 または STEP3 b-2 が終了したら，STEP3 c を実行し，STEP2 に戻る．以降，STEP2 a から STEP3 c までの作業を故障の原因（n 次中間故障）がそれ以上追跡できなくなるまで繰り返し行う．追跡ができなくなった故障の原因を最終的な故障の原因（基本事象）とする．

なお，提案手法で FT を作成する場合には図 1(a)から(f)の記号に加えて(g)に示す故障の説明記述用の記号を追加する．この理由は，従来の FT では説明記述を原因の記号

を用いて記述していることも多く, FT の本質的な構造が分かりにくかったためである. 説明記号を導入することで FT の論理的な構造と説明を明確にして FT を読みやすくする. なお，説明記述用の記号は FT の論理的な構造に影響を与えないので，解析の際は無視する.

STEP1: 故障個所の同定
分析対象の故障の決定と故障を生じる命令の識別.

STEP2: 故障を生じる命令のFT作成
a. 命令に対応したFT雛型の準備.
b. 命令が実行されて生じる故障か，命令が実行できずに生じる故障かの検討.
命令が実行できずに生じる故障の場合は「命令が実行できない場合」のFT雛型を用いてFTの枝を追加.
c. 作成したFTに故障の条件と原因を記述.

STEP3: 故障原因の分析
a. 故障の条件や原因で使用する変数が広域変数か局所変数かの識別.
b-1. 使用する変数が広域変数の場合
広域変数のFT雛型の準備，使用する変数に値を設定している命令の識別，識別した命令の広域変数のFT雛型への記述.
b-2. 使用する変数が局所変数の場合
使用する変数に値を設定している命令の識別.
c. STEP2へ戻る.

図 10　FT 作成ルールの概要
Figure 10 FT Development Rules

3.4 FTA 支援ツールの試作

提案手法に対応した FT 作成支援ツールを試作した．開発言語には Java を，OS には Windows7 を使用した．FT 支援ツールは以下の機能を有する．①～⑤は前処理に対応する機能であり，⑥～⑧は FT 作成ルールに対応する機能である．図 11 に FT 支援ツールの FT 作成時の画面を示す.

① 命令群のリストを作成する.
② 広域変数と局所変数のリストを作成する.
③ 指定された変数の使用場所と代入場所のリストを作成する.
④ module の起動方法のリストを作成する.
⑤ 割り込み禁止と割り込み禁止解除が行われる箇所のリストを作成する.
⑥ 変数の使用箇所と代入箇所を自動的に追跡する.
⑦ 命令群に応じた FT 雛形を提供する.
⑧ FT 作成ルールに則って FT（ただし，表形式）を作成する.

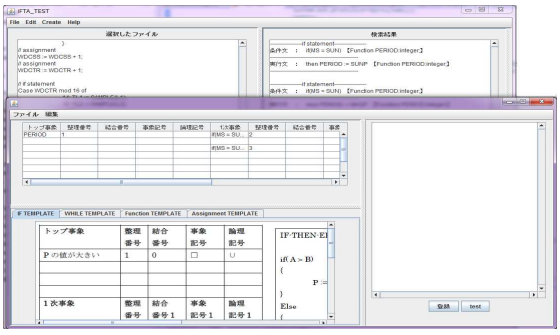

図 11　FT 作成支援ツール(FT 作成時)
Figure 11 FT Development Tool

4. 提案手法の適用と評価

提案手法の評価のため，スピン衛星の回転速度が速くなりすぎる故障に対して FT を作成した．この事例は Leveson らが FTA を実施したものである[6]. スピン衛星のシステムレベルの FTA の結果，制御プログラムの「変数 PERIOD の値が大きすぎる」と「変数 LENGTH の値が小さすぎる」の場合に回転速度が速くなりすぎることが分かった．本章では「変数 LENGTH の値が小さすぎる」故障に対して FT を作成する．図 12 に制御プログラムの構造を示す．図中の四角は主要な module，矢印は module の呼び出し関係，下向きの三角形は module の反復呼び出し, ひし形は排他の呼び出し，六角形は主要な広域変数，そして点線の矢印は広域変数の参照関係を表す．module の VBH と MONITOR SPIN はタイマにより周期的に起動される．RESTRAT3 は太陽パルス信号の割り込み，RESTART4 は通信割り込みで起動される．図 13 に制御プログラム（文献[6]から引用）を示す.

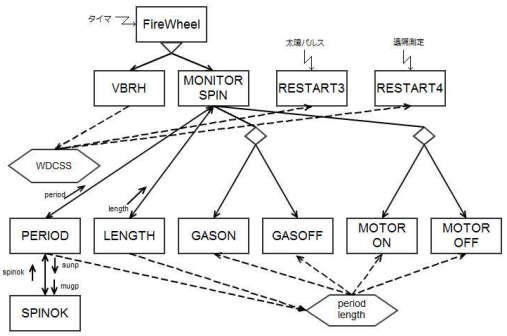

図 12　スピン衛星の制御プログラムの構造
Figure 12 Configuration of Program for Spin Satellite

```
FUNCTION PERIOD:INTEGER;
BEGIN
  IF SPINOK (SUNP)
    THEN MS :=SUN
    ELSE IF SINOK (MAGP)
      THEN MS := MAG
      ELSE MS :=SUN
  IF MS = SUN
    THEN PERIOD := SUNP
    ELSE PERIOD := MAGP
END

FUNCTION SPINOK (PER: INTEGER):
BOOLEAN;
  SPINOK := PER > 100 and PER < 6500;

FUNCTION LENGTH: INTEGER;
  LENGTH := MAX(TL1, 0) +MAX(TL2,0);

PROCEDURE RESTART4:
BEGIN
  IF DNCTR <> 1
    THEN DNCTR := DNCTR – 1
    ELSE BEGIN
           DNCTR := DNMAX;
           THETA := THETA +1 MOD
         END
  WDCSS := WDCSS + 1;
  WDCTR := WDCTR + 1;
  CASE WDCTR MOD 16 OF
    0: …
    .
    .
    14: TL1 := SAMPLE(L1)
    15: TL2 := SAMPLE(L2)
  END;
END
```

```
PROCEDURE RESTART3;
BEGIN
  SUNP := MIN(LASTP, WDCSS)
  DNMAX *= MIN := MIN((SUNP+64)/128. 255);
  DNCTR := DNMAX;
  THETA := 0;
  LASTP := WDCSS;
  WDCSS := 0;
END

PROCEDURE VBRH;
BEGIN
  DISABLEINTS;
  FOR I := 0 to 127 do
    RAM[I] := DBRS;
  WDLOST := 64;
  WDCSS := WDCSS + WDLOST;
  WHILE WDLOST >= DNCTR DO
    BEGIN
      WDLOST := WDLOST – DNCTR;
      DNCTR := DNMAX;
      THETA := THETA + 1 MOD 128;
    END
  DNCTR := DNCTR – WDLOST
  ENABLEINTS;

PROCEDURE MONITORSPIN;
BEGIN
  P := MIN(PERIOD/64, 255);
  L := MIN(LENGTH/16, 15);
  IF P < GASTOP[L]  THEN GASSOFF;
  IF P > GASBOT[L]  THEN GASON ;
  IF P > BOOMTOP[L]  THEN MOTOROFF
  ELSE IF P < BOOMTOP[L] THEN MOTORON
END
```

図 13　スピン衛星の制御プログラム（抜粋）
Figure 13 Program of Spin Satellite - Extracted -

4.1 提案手法の適用

図 14～15 に「変数 LENGTH の値が小さすぎる」故障に対する FT の作成について記述する．

事前に制御プログラムの前処理を行い，広域変数と局所変数の区分，命令の階層構造を明らかにしたもした．さらに，モジュール VBRH で割込み禁止と割込み禁止解除が行われていることを明らかにした．

1 回目の FT 作成手順を実施する．STEP1 を適用し，分析する故障（トップ事象）は「変数 LENGTH の値が小さすぎる」とし，故障を生じる命令が FUNCTION LENGTH であることを確認する．LENGTH は max(TL1,0)と max(TL2,0)の和であるので，故障の原因は「max(TL1,0)が小さすぎる」かつ「max(TL2,0)が小さすぎる」となるので，STEP2a を適用して，「max(TL1,0)が小さすぎる」と「max(TL2,0)が小さすぎる」を論理積で結合した FT を準備する．以降，「max(TL1,0)が小さすぎる」について記述する．STEP2b を適用して，LENGTH が実行されない可能性を検討する．この場合は，実行されない可能性はないので FT の枝の追加はない．STEP2c を適用して，FT の原因に「TL1 の値が小さすぎる」を記述する．STEP3a を適用して，TL1 が広域変数であることを確認する．STEP3b-1 を適用して，TL1 に値を設定しているのは RESTART4 の「TL1 := SAMPLE(L1)」のみであるので，広域変数の FT 雛型にこれを記述する．この後，STEP2 に戻る．

2 回目の FT 作成手順を実行する．RESTART4 の「TL1 := SAMPLE(L1)」は CASE 文の中にあるので，STEP2a を適用して，IF-THEN-ELSE_IF-ELSE の FT 雛型を準備する．STEP2 b を適用して，「TL1 := SAMPLE(L1)」が実行されない場合を検討する．この場合は，CASE 文の条件が「1:・・・（LEVESON の論文では記述が省略)」から「13:・・・」が実行される場合を追加する．STEP2c を適用して，「1:・・・」から「13:・・・」が実行される場合の原因を記述する．以降，「1:・・・」の場合を記述する．FT に「1:・・・」が実行される条件として「WDCTR mod 16 = 1」，原因として「・・・に誤りがある」を記述する．「・・・に誤りがある」については，これ以上の情報がないので，これを基本事象とする．STEP3a を適用して，WDCTR が局所変数であることを確認する．STEP3b-2 を適用して，WDCTR に値を設定しているのが RESTART4 の「WDTCR := WDCTR +1」であることを識別する．この後，STEP2 に戻る．

3 回目の FT 作成手順を実行する．STEP2a を適用して，RESTART4 の「WDTCR := WDCTR +1」は代入文であるので，代入文の FT 雛型を準備する．STEP2 b を適用して「WDTCR := WDCTR +1」が実行される場合と実行されない場合を検討する．実行される場合は RESTART4 が 1 回実行される，実行されない場合は RESTART4 が 1 回も実行されない場合と割込み禁止の場合があるので，割込みの FT 雛型を準備する．STEP2c を適用して，TESTART4 が 1 回

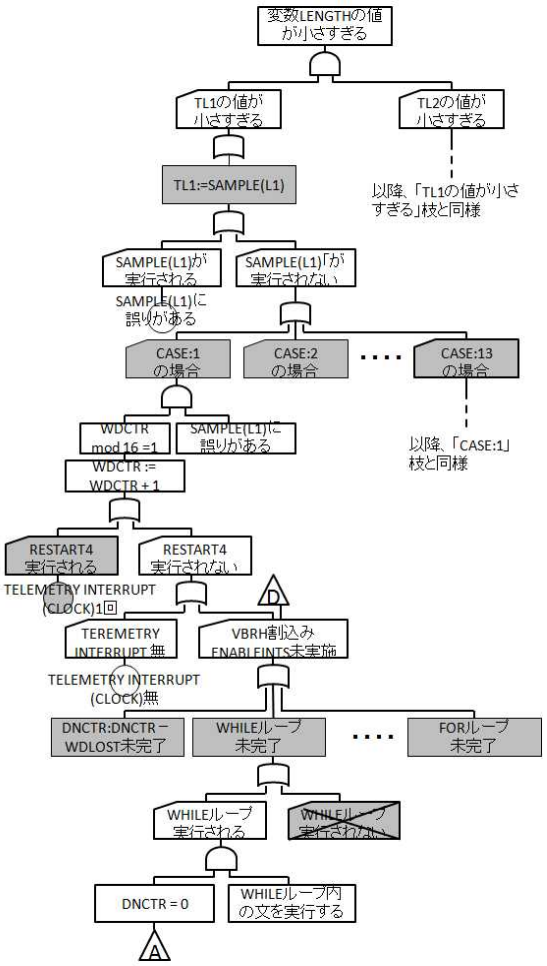

図 14 「変数 LENGTH の値が小さすぎる」故障の FT(1/2)
Figure 14 FT of "Variable LENGTH Too Small" (1/2)

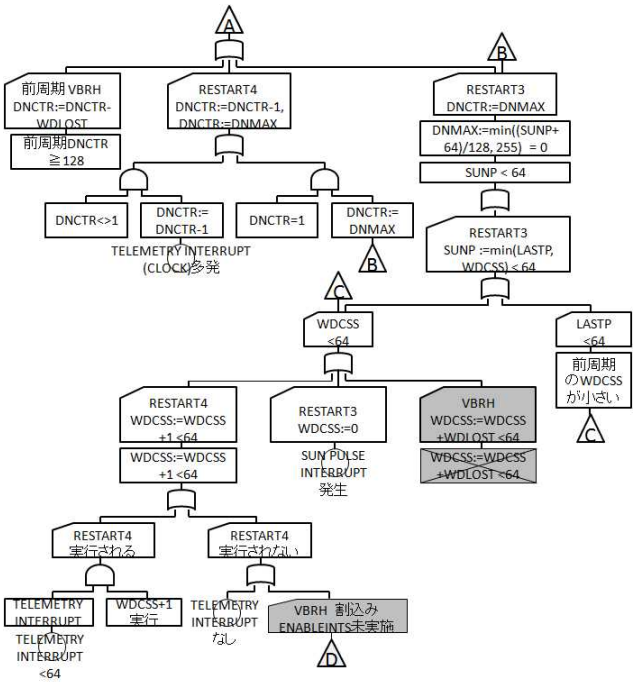

図 15 「変数 LENGTH の値が小さすぎる」故障の FT
Figure 15 FT of "Variable LENGTH Too Small" (2/2)

実行される場合は TELEMETRY INTERRUPT が 1 回発生したことが基本事象となる．RESATRT4 が 1 回も実行されない場合は TELEMETRY INTERRUPT が発生しないことが基本事象となる. 割込み禁止の場合は VBRH の ENABLEINTS が実行されないことが中間事象になる．ENABLEANTS には変数は含まれていないので，STEP3a から STEP3b-2 までの作業はない．この後，STEP2 に戻る．

4 回目の FT 作成手順を実行する. VBRH の ENABLEINTS は出力を行うだけの命令であるので，STEP2a の作業はない．STEP2 b を適用して，ENABLEINTS が実行されないない可能性を検討する．ENABLEINTS が実行されないのは，VBRH が ENABLEINTS より前の命令から先に進まなくなる場合であるので，命令が実行できない場合の FT 雛型を準備する．STEP2c を適用して各命令の先に進まなくなる可能性を検討する．その結果，命令が先に進まない可能性があるのは，WHILE ループのみで，WHILE ループの条件が「常に WDLOST >= DNCTR」となる場合である．WHILE ループの FT 雛型を準備し，条件が「常に WDLOST >= DNCTR」を記述する．ところで，WDLOST の初期値は 64 で, WHILE ループの中で「WDLOST := WDLOST - DNCTR」が無限回繰り返されることになるが, この条件のもとで「常に WDLOST >= DNCTR」となるのは「DNCTR = 0」の場合のみであるため，条件「常に WDLOST >= DNCTR」を「DNCTR = 0」に書きかえる．STEP3a を適用して，DNCTR が広域変数であることを確認する．STEP3b-1 を適用して，DNCTR に値を設定しているのは VBRH が前回起動された際の「DNTCR := DNTCR - WDLOST」，RESTART4 の「DNTCR := DNTCR-1」と「DNTCR := DNMAX」，そして RESTART3 の「DNCTR := DNMAX」であるので，広域変数の FT 雛型にこれらを記述する. この後, STEP2 に戻る．以降，RESTART3 の「DNCTR := DNMAX」について記述する．

5 回目の FT 作成手順を実行する．STEP2a を適用して，RESTART3 の DNMAX に値を代入している命令は RESTART3 の「DNMAX := min((SUNP+64)/128, 255)」であるので，代入の FT 雛型を準備する．この命令は必ず実行されるので STEP2 b の検討は必要ない．STEP2c で，FT 雛型に「DNMAX := 0」となる条件である「SUNP < 64」を記述する．STEP3a で SUNP が広域変数であることを確認する．STEP3b-1 を適用して，SUNP に値を設定しているのは RESTAR3 の「SUNP := min(LASTP, WDCSS)」のみであるので，広域変数の FT 雛型にこれを記述する．この後，STEP2 に戻る．

6 回目の FT 作成手順を実行する．「SUNP := min(LASTP, WDCSS)」が 64 より小さくなるのは，「LASTP < 64」または「WDCSS < 64」であるので, STEP2a を適用して,「LASTP < 64」と「WDCSS < 64」を論理和でつないだ FT を準備する．以降，「WDCSS < 64」について記述する．STEP2b を適用して，「WDCSS < 64」が実行されない可能性を検討する．この場合は，実行されない可能性はないので FT の枝の追加はない．STEP2c を適用して条件に「WDCSS < 64」を記述する. STEP3a を適用して WDCSS が広域変数であることを確認する．STEP3b-1 を適用して，WDCSS に値を設定しているのは RESTART4 の「WDCSS := WDCSS + 1」，RESTART3 の「WDCSS :=0」，および VBRH の「WDCSS := WDCSS + WDLOST」であるので，広域変数の FT 雛型にこれらを記述する．この後，STEP2 に戻る．以降，RESTART4 の「WDCSS := WDCSS + 1」について記述する．

7 回目の FT 作成手順を実行する．STEP2a を適用して，「WDCSS := WDCSS + 1」は代入文であるので，代入文の FT 雛型を準備する．STEP2 b を適用して「WDTCR := WDCTR +1」が実行される場合と実行されない場合を検討する．「WDTCR := WDCTR +1」が実行される場合は TEREMETRY INTERRUPT により RESTART4 が実行される場合であり,「WDCTR := WDCTR +1」が実行されない場合は TEREMETRY INTERRUPT がないか，VBRH により ENABLEINTS が未実行の場合である．割込みルーチンの FT 雛型を追加する．STEP2 c で TEREMETRY INTERRUPT により RESTART4 が実行される場合の条件は TEREMETRY INTERRUPT が 64 回未満発生することとなり, これが基本事象となる．TEREMETRY INTERRUPT がない場合は TEREMETRY INTERRUPT が発生しないこととなり, これが基本事象となる. そして VBRH により ENABLEINTS が未実行の場合は，既に FT を作成してあるので，その部分に移動すればよい（図 14 の三角形の D の箇所）.

以上により，「変数 LENGTH の値が小さすぎる」故障の FT が完成した.

4.2 提案手法の評価

評価のために, 提案手法を用いて作成した FT と Leveson らが作成した FT とを比較した．その結果，図 14 と図 15 の灰色で示した中間原因や説明に差異があるものの, FT の基本的な構造は同じであった．以下に FT の差異について説明する.

図 14 の CASE14 の「SAMPLE（L1)が実行されない」場合は，提案手法では CASE 1 から CASE13 が発生して CASE14 が発生しない場合を検討しているが，LEVESON らの手法では単に CASE14 が発生しない場合のみを検討している．結果は同様となるが，提案手法は全ての可能性を列挙しているのに対し，LEVESON らの手法では技術者が経験に基づいて検討を省略している.

図 14 の VBRH の「ENABLEINTS 未実行」の場合は，提案手法では VBRH の命令 1 個づつに対して, 処理が進まなくなる可能性を検討した．一方で，LEVESON らの手法では特に説明がないまま, WHILE 文で処理が進まなくなることが記述されている．これは，提案手法は FT 作成手順に従って全ての可能性を検討したが，LEVESON らの手法で

は技術者が経験に基づいて検討を省略したためである．

図 15 の VBRH の WDCSS := WDCSS + WDLOST の場合は，提案手法では，FT の雛型に基づいて故障の条件を挙げた後で，発生する可能性がないためバツ印をつけている．一方，LEVESON らの手法では，最初から発生しない故障として FT は作成されていない．これは，技術者が経験に基づいて検討を省略したためである．

図 15 の VBRH での割込み ENABLEINTS の未実行の場合は，提案手法では雛型に従って故障を検討している．一方，LEVESON らの手法では，多重故障を想定していないため検討が省略されている．

以上の比較結果より，提案手法を実装した支援ツールで作成した FT の方が，記述量は多くなるものの，より多くの故障の原因について検討している．加えて，故障原因を分析する過程を漏れなく記述してあるので，FT の理解が容易となる．さらに，時間が経過した後で，FT を見直した場合，理解が容易となる．加えて，提案手法を適用した場合，異なる分析者が FT を作成しても，同様の FT を作成することができた．

故障「変数 PERIOD の値が大きすぎる」に対しても FT の比較を行った．その結果も「変数 LENGTH の値が小さすぎる」と同様の結果となった．以上の結果から，スピン衛星の「回転速度が速くなりすぎる」故障の原因分析に提案手法を適用した結果，適切な FT を機械的に作成できることが確認できた．

作成したツールを評価するために複数の分析者に試作ツールを用いて FT を作成させた．その結果，以下の評価が得られた．

① 前処理で命令群が明らかになっているので，制御プログラムに FT 雛形をあてはめやすい．
② 変数の値の追跡が容易である．
③ 分析終了箇所といない未終了箇所の識別が容易である．

その一方で，画面が小さく文字が読みにくい，1 つの Window の中に情報を詰め込みすぎている，縦方向の分析（中間原因を 1 つづつ分析する方法）をサポートするべき，最終的な FT 形状の判断は技術者が行わなければならない等の改善点が挙げられた．

以上のことから，提案手法についての有用性が認められた一方で，支援ツールについては使用上の課題が多く残った．今後，支援ツールの改善を図っていく．

5. まとめと今後の課題

本論文では，制御プログラムに適用可能な FTA 手法を提案した．この手法で制御プログラムの特徴に合わせた FT の雛形を作成した．そして FT を作成する際に前処理を行うことで，対象となる制御プログラムの動作上の特徴を明らかにして，後続する FT の作成を支援した．さらに，FT の作成手順を明確化した．これにより，FT を機械的に作成することが可能となった．加えて，FTA の再現性も確保できるようになった．

今後の課題としては，提案 FTA 手法を他の制御プログラムに適用し，提案手法の適用可能範囲を明らかにする．そこで得られた知見から FT 作成時の前処理とルールの改善を行っていく．さらに，FT の作成を効率化するために支援ツールの改善を行っていく．

謝辞

本研究は平成 25 年度科学技術研究費助成事業「複数の故障解析技法を連携させた医薬品製造システムのリスクマネジメント手法」の支援により実施した．

参考文献

1) Nancy G. Leveson: Safeware System Safety and Computers，翔泳社 (2009)
2) 高橋正和，難波礼治，福江義則:医薬品製造にかかわるコンピュータ化システム向けの FMEA を用いた運用リスクマネジメント手法の提案，計測自動制御学会論文集，Vol.481，No.5，pp.285-294(2012).
3) Jon D. Reese，Nancy G. Leveson: Software Deviation Analysis: A Safeware Technique, Proc. of the 19th international conference on Software engineering, pp.250-260(1997).
4) Wolfgang Weber, Heidemarie Tondok, and Michael Bachmayer: Enhancing Software Safety by Fault Trees: Experiences from an Application to Flight Critical SW，Proc. of SAFECOMP2003，LNCS 2788，pp.289-302(2003).
5) Michael A. Friedman: Automated Software Fault Tree Analysis for Pascal Program, Proc. Annual Reliability and Maintainability Symposium, pp.458-461(1993).
6) Nancy G. Leveson and Peter R, Harvey: Analyzing Software Safety, IEEE Transaction on Software Engineering, Vol. 9, No.5, pp.569-579(1983).
7) Ganesh Pai and Joanne Dugan, Automatic Synthesis of Dynamic Fault Tree from UML System Model, Proc. of 13th International Symposium on Software Reliability Engineering, pp なし(2002).
8) Luyi Li, Minyan Lu, and Tingyang Gu, A Reuse-Oriented Auxiliary Construction Method for Software Fault Tree and Tool Implementation, Proc. of 2014 International Conference on reliability, Maintainability and Safety, pp451-456(2014).
9) Jinfu Zhao, Hong Zhang, and Cong Pan, Improved Reuse Integration of SFMEA and SFTA, Proc. of 2014 International Conference on reliability, Maintainability and Safety, pp552-557(2014).
10) 小野寺勝重: 国際標準化時代の実践 FTA 手法 -信頼性，保全性，安全性解析と品質保証- ，日科技連出版，(2000).
11) Nancy G. Leveson, Stephen C. Cha, and Timothy J. Shimeall; Safety Verification of ADA Programs Using Software Fault Trees, IEEE Software, Vol.8, Issue 4, pp.48-59(1993).