Vol.2013-SE-182 No.24
2013/10/25

組込みソフトウェア向けの故障木解析手法

高橋正和[†1]

本論文では，組込みソフトウェア向けの故障木解析（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（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．さらに組込みソフトウェア等のリアルタイム性を有するソフトウェアの FTA を行う上での注意点についても明確にしていなかった．そのため，技術者が新たに FT を作成することが困難であった．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．さらに FT の作成手順を前処理と作成処理に分割した．FT 前処理では対象となる組込みソフトウェアに対して，命令グループの識別，変数のスコープと使用／代入の識別，ソフトウェアモジュール起動方式の識別等を行い，ソフトウェアの構造を明らかにする．そして，FT 作成処理では，故障を生じる命令の入力変数を同定し，その入力変数の値が代入（計算）されている命令を追跡するルールを規定した．この作業を繰り返すことで，故障の原因を明らかにする．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．また，提案手法を適用することで，技術者間で作成する FT の差異が少なくなることも確認できた．これらのことより，組込みソフトウェアの FTA の品質が向上すると考えられる．

A Study of Fault Tree Analysis for Embedded Software

MASAKAZU TAKAHASHI[†1]

This manuscript proposes a Fault Tree Analysis (FTA) method for embedded 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. Furthermore, those FTA methods didn't pay attentions for software, such as embedded software, that had real time characteristics. Therefore, it was difficult to develop new FT from scratch for the engineers. Proposed FT method makes the inexperienced engineers develop appropriate FT by reviewing the FT templates and defining a procedure for developing whole FT. The procedure consists of the pretreatment and development step. In the pretreatment step, engineers conduct identification of instruction group, identification of scope and use/assign for variables, and identification of execution type of each software module. In the FT development step, we define following FT development rules; identifying the input variables in the statement that cause the failure, tracing the instructions that the values of variables are assigned (calculated), and repeating those operations clarifies the reasons of the failure. As a result of applying the proposed FTA method to the failure that the 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. Consequently, we consider that those improve the quality of software FTA for the embedded software.

1. はじめに

今日，コンピュータ化システムは自動車，プラント設備，航空宇宙機器等の制御等の様々な分野で使用されている．それらの中にはコンピュータ化システムに故障が発生した場合に，使用している人や周囲の環境に重大な被害をもたらすものがある．このようなコンピュータ化システムに対しては，設計段階や使用段階において安全性に対する十分な解析を行なければならない[1]．そして，安全性を実現するための対策を施さなければならない．コンピュータ化システムの安全性を実現する方法には様々あるが，本研究では，故障木解析手法(Fault Tree Analysis)を組込みソフトウェアに適用する方法について述べる．そして，実際の組込みソフトウェアに適用することで提案手法の評価を行う．

†1 山梨大学工学部コンピュータ理工学科
University of Yamanashi, Faculty of Engineering , department of Computer science and Eng.

2. 関連研究

ソフトウェアを対象とした様々な安全性解析手法が研究されている．安全性を解析する手法は，設計段階で用いられる故障の可能性がある箇所を網羅的に検討する方法と，特定の故障に対してその原因を明確にする方法に大別できる．故障の可能性がある箇所を網羅的に検討する方法に関しては以下の(1)で，特定の故障に対してその原因を明確にする方法に関しては以下の(2)で述べる．

(1) 故障の可能性がある箇所を網羅的に検討する方法

高橋らは，Failure Mode and Effect Analysis（FMEA）を用いて，医薬品製造に関わるコンピュータ化システム(DMCS)向けの運用に関わるリスクマネジメントを行う方法を提案した[2]．この論文における FMEA では，DMCS の運用方法・使用方法に故障モード（運用方法や使用方法の誤り）が発生した場合，DMCS および医薬品製造設備にどのような影響（システムの故障）を及ぼすか解析する．この論文

では既存 DMCS に関する FMEA の実施結果を収集し，それらを比較して，DMCS に共通に適用可能な故障モード，リスク低減策の一覧表を作成した．これらを使用することで適切なリスクマネジメントを実施出来るようにした．

Reese らは，Software Deviation Analysis(SDA)を用いて，プラントの制御ソフトウェアの安全性を解析する方法を提案した[3]．SDA は，対象となるプラントが定常状態にある時に制御ソフトウェアへの入力値が変化した際，プラントの状態がどのようになるか（悪影響を与えるか）を解析する手法である．この論文では制御ソフトウェアの仕様書から入力と出力の対応関係を表す Causality Diagram を作成し，それに対して Qualitative Mathematics を適用することで出力の変化を把握する．これにより，入力データの変化によりプラントが不安定状態になるかを分析できるようにした．

(2) 特定の故障に対して原因を明確にする方法

Friedman らは，非リアルタイムプログラムに対する Fault Tree Analysis (SFTA)手法を提案した．これを Software FTA（SFTA）と呼ぶ[4]．SFTA ではプログラムを実行した結果として生じる故障事象の原因を，Fault Tree（FT）を作成しながら追跡していく．この論文ではプログラムの基本構造である WHILE 文，IF 文，代入文に対して故障事象と原因の関係を表す FT のひな型を作成した．プログラムに実行経路を追跡しながら，FT のひな形を組み合わせて全体の FT を作成できるようにした．

Leveson らは，リアルタイムプログラムに対する SFTA 手法を提案した[5]．この論文の内容については 3 章で後述する．さらに，Leveson らは，作成する FT はプログラミング言語の特徴に依存しているとも述べており，その適用例として Ada 言語に対する FT の雛形を提案し，交差点の信号変更タイミングの制御に適用した[6]．

Helmer らは，コンピュータの安全を脅かす事象(管理者権限の奪取等)を 7 種類に分類した[7]．そして其々の事象に対する標準 FT を作成した．新規にシステムを開発する場合，それらの標準 FT を参考としながら，新規システムに応じて修正することで新規システムに適したセキュリティ対策を行えるようにした．

Dulac らはスペースシャトルの熱処理システムの安全性解析に Systems Theoretic Accident Modeling and Process (STAMP)を適用した[8]．STAMP とはシステムを構成する制御機器同士の相互作用により，システム全体に故障が生じる事がないことを確認する手法である．個々の制御機器の状態，安全性に関する制約条件を入力することで，個々の制御機器の故障により，システムが故障に至る事象を識別できるようにした．

Weber らは，アセンブラで記述された航空機制御プログラムに対して SFTA を用いて故障原因の分析を行った[9]．事前に航空機の制御を故障状態に陥れる可能性の高いプログラム部分を洗い出したうえで，想定される故障をリスト化した．そして，その故障の中でリスクが高いものに対して SFTA を実施した．これによりリスクの高い故障に対して効率的に故障原因を分析できるようにした．

3. Leveson の Software Fault Tree Analysis

Leveson らの提案したリアルタイムプログラムに対する SFTA 手法の概要を以下の(1)で，その手法を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適用した結果を以下の(2)で述べる．

(1) Software Fault Tree Analysis の概要

一般的な FTA では，はじめに，故障事象から，それを引き起こす原因事象を明らかにし，事象記号と論理記号を用いて故障事象と原因事象の対応関係に相当する FT(故障の原因となる事象の関係を，事象記号を用いて表した図)を作成する．そして，FT に現れた原因事象の１つ１つを新しい故障事象ととらえて，その原因事象を明らかにし，FT を作成する．この作業を，原因事象がそれ以上分解できなくなるまで繰り返す．最後に現れた原因事象が最終的な故障の原因となる[10]．図 1 に事象記号と論理記号を示す．

組込みソフトウェアの場合，出力される変数の値が異常となること，誤ったタイミングでモジュールが実行されること等が故障事象となる．はじめに，その故障事象を発生させる命令を同定し，その命令が故障事象を生じる際の入力変数の値を求める．これを原因事象ととらえる．そして，故障事象と原因事象の対応関係に相当する Software FT（SFT）を作成する．次に，その原因事象を生じる命令を同定する．これを新しい故障事象とみなして，その命令が故障事象を生じる際の入力変数の値を求める．この作業を原因事象がそれ以上追跡できなくなるまで繰り返す．最後に現れた原因事象が故障の原因となる．なお SFT 作成時に使用する記号は FT と同様である．

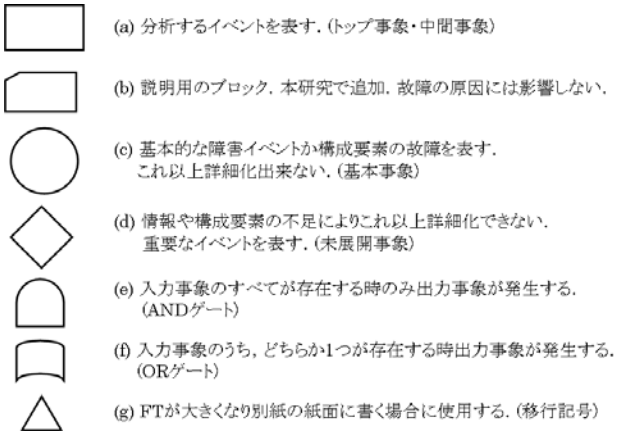

図 1　FT，SFT の作成に用いる記号群

一般的にプログラムは IF-THEN-ELSE 文，WHILE 文，関数呼び出し，代入文の基本的な命令を組み合わせることで作成されている．そこで Leveson らは，それらの基本命令に対応する SFT のひな型を作成した．図 2 に IF-THEN-ELSE 文，図 3 に WHILE 文，図 4 に関数呼び出し，そして図 5 に代入文の SFT のひな型を示す．プログラムに対する全体の SFTA は，これらの SFT のひな型を組み合わせて作成する．

図 6 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」という故障事象が生じる場合の SFT を作成する事例を紹介する．このプログラムの基本構造は IF-THEN-ELSE 文なので図 2 のひな型形を使用する．はじめに，IF-THEN-ELSE のひな型の上部に故障事象「x > 10」を記述する．次に IF-THEN-ELSE 文の条件節が True となる場合と False となる場合での原因事象を記述する．条件節が True となる場合の原因事象は「a ≧ b」かつ「x := 5」となることである．条件節が False となる場合の原因事象は「a not ≧ b （a < b)」かつ「x := g(x)」となることである．このプログラムの場合，これ以上の情報がないので，SFTA は作成完了となる．三番目に原因を分析する．条件節が True の場合は「x > 10」の原因として「x := 5」が実行されることが要求されるが，発生することはないので，これ以上の分析は行わない．条件節が False の場合は「x > 10」の原因として「x := g(x)」が実行されることが要求される．このプログラムの場合，g(x)の情報は公開されていないので，これ以上の分析を行うことはできない．従って，「x > 10」の原因は g(x)が 10 より大きい値を返すことになる．以上の結果から，上記プログラムを実行した時に「x > 10」となる故障事象が発生する原因は，「関数 g(x)の戻り値が 10 より大きい値となる」ことだと考えられる．

(2) Software Fault Tree Analysis の適用

Leveson らは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適用した．その概要を以下に述べる．

はじめに，（ハードウェアとソフトウェアを組み合わせた）スピン衛星の回転速度が速くなりすぎる故障事象に対して FT を作成した．そして，ハードウェアとソフトウェアがどのような状態となるとスピン衛星の回転速度が速くなりすぎるか，原因を明らかにした．この原因に対してハードウェアとソフトウェアで個別に FTA を行った．

ここで，ハードウェアを対象にした FTA の手法はすでに確立されているので，本論文では扱わない．

スピン衛星の回転速度が速くなるソフトウェア的な原因は「(変数) PERIOD の値が大きすぎる」,「(変数) LENGTH の値が小さすぎる」ということになった．以降では，これらの原因に対する SFT の作成について述べる．

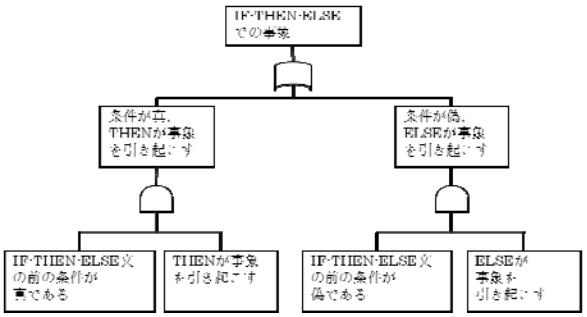

図 1　IF-THEN-ELSE のひな型（[5]より引用）

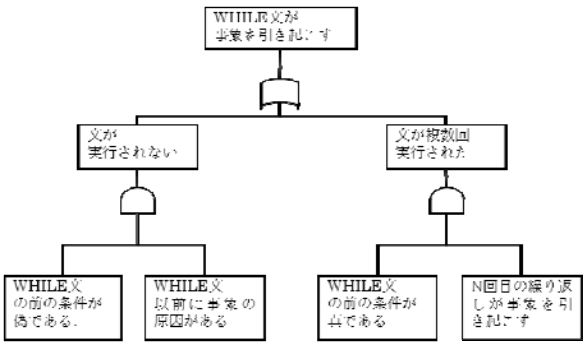

図 2　WHILE のひな型（[5]より引用）

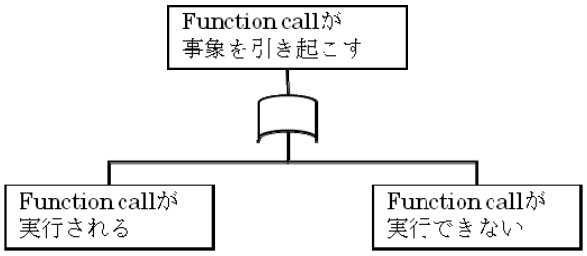

図 3　関数呼び出しのひな型（[5]より引用）

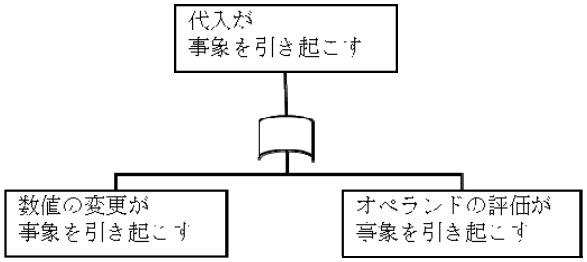

図 4　代入のひな型（[5]より引用）

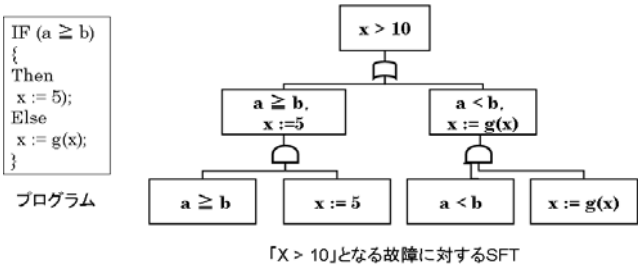

図 5　SFT の適用事例

図 7 にスピン衛星の制御プログラムの中で，上記の故障にかかわりのある部分のみを示す．標記は擬似 PASCAL である．図 8 に「(変数) PERIOD の値が大きすぎる」故障事象に対する SFT を示す．

以下の SFT の内容を説明する．

はじめに，この SFT の故障事象は「(変数) PERIOD の値が大きすぎる」とする．SFT の上から 1 列目の故障事象「PERIOD の値が大きすぎる」を記入する．

二番目に，変数 PERIOD の値を出力しているプログラム部分を見つける．該当部分は Function PERIOD の 8 行～10 行にある IF-THEN-ELSE 文中の 2 か所である．これ以外に変数 PERIOD の値を出力している部分はない．IF-THEN-ELSE 文の条件節「MS = SUN」が True の場合は「PERIOD := SUNP が大きい値となる」場合に変数 PERIOD が大きくなり，False の場合は「PERIOD := MAGP が大きい値となる」場合に変数 PERIOD が大きくなる．これらを原因事象と考える．これらのうち 1 つの原因事象が発生すれば故障事象が生じるので，SFT の上から 2 行目に「PERIOD := SUNP が大きい値となる」と「PERIOD:=MAGP が大きい値となる」を，OR ゲート（論理和記号）を用いて結合して記述する．

三番目に「PERIOD := SUNP が大きい値となる」原因事象は IF-THEN-ELSE 文の条件節が「MS = SUN」で，THEN 部分の「SUNP の値が大きい」場合に生じる．これらを，SFT の上から 3 行目にある IF-THEN-ELSE 文のひな型に記入する．同様に「PERIOD := MAGP が大きい値となる」原因事象は IF-THEN-ELSE 文の条件節が「MS NOT = SUN」で，THEN 部分の「SUNP の値が大きい」場合に生じる．これらを，SFT の上から 3 行目にある IF-THEN-ELSE 文のひな型に記入する．

四番目に，「MS = SUN」について SFT の上から 4 行目より下の部分について説明する．MS は Function PERIOD の 3 行～7 行にある複合 IF 文の中の 3 か所で代入されている．4 行目にある 1 つ目の MS への代入「MS := SUN」を考える．この命令が実行されるためには複合 IF 文の外側の IF 文の条件節において「SPINOK(SUNP)が True」となり，かつ「MS := SUN」が実行される必要がある．従って，これらを原因事象として SFT の上から 4 行目にある IF-THEN-ELSE 文のひな型に記入する．7 行目にある 3 つ目の MS への代入「MS := SUN」を考える．この命令が実行されるためには複合 IF 文の外側の IF 文の条件節において「SPINOK (SUNP)が False」となり，かつ，複合 IF 文の内側の IF 文の条件節において「SPINOK(MAGP)が False」となり，かつ，「MS := SUN」が実行される必要がある．従って，これらを原因事象として SFT の上から 4 行目にある IF-THEN-ELSE 文のひな型に記入する．なお，Function Period の 6 行目にある 2 つ目の MS への代入「MS := MAG」については，これが実行されると必ず「MS NOT = SUN」となるので，SFT の枝は作成しない．

五番目に「SPINOK（SUNP)が True」となる理由を考える．Function SPINOK（PER)（PER は SPINOK への引数）は PER の値により True あるいは False を返す．「SPINOK（SUNP)が True」となるのは Function SPINOK(PER)のコードより，「PER > 100」かつ「PER < 65000」なので，SFT の上から 5 行目に「SUNP > 100」と「SUNP < 65000」を，AND ゲートを用いて結合して記述する．ここで，Function SPINOK(PER)のコードより，これ以上の原因事象の追跡は不可能である．そのため，これらをこれ以上展開できない基本事象として記述する．全ての SFT の枝についてもこれ以上展開できない基本事象，あるいは，未展開事項となるまで同様の作業を繰り返す．

「PERIOD の値が大きすぎる」故障事象に対して作成した SFTA を図 8 に示す．この結果，「PERIOD の値が大きすぎる」故障事象の原因は「SUN PULSE をとり損なう」あるいは「CLOCK（TELEMETRY INTERRUPT）が多発する」，SUMP の値が「SUMP > 100」かつ「SUMP < 65000」となる，MAGP の値が「MAGP < 100」かつ「MAGP > 65000」となることだと分かる．

```
Function PERIOD:integer;/* redundant routines for calculating spin period */
        Begin
                If SPINOK(SUNP)   /* determine which of MAG or */
                        Then MS := SUN     /* sun info is working OK    */
                        Else If SPINOK (MAGP)
                                Then MS := MAG
                                Else MS := SUN     /* use sun if neither */
                If MS = SUN
                        Then PERIOD := SUNP        /* return sun-period or */
                        Else PERIOD := MAGP        /* mag period depending */
        End

Function SPINOK (PER:integer):Boolean;
        SPINOK := PER > 100 and PER < 65000;

Function LENGTH:integer;    /* returns tip-to-tip length of boom wires */
        LENGTH := max(T1,0) * max(T2,0);

Procedure RESTART4;                 /* telemetry interrupt (milsec) */
        Begin
                If DNCTR <> 1       /* count down until 0. Then */
                  Then DNCTR := DNCTR - 1  /* reload the count and bump */
                  Else begin                        /* the sun angle theta */
                        DNCTR := DNMAX;
                        THETA := THETA + 1 mod
                        End
        WDCSS := WDCSS + 1;         /* record msecs since sun */
        WDCTR := WDCTR + 1;         /* and update T/M word number */
        Case WDCTR mod 16 of        /* do one of 16 routines */
                0: ...
                *: ...
           14:TL1 := SAMPLE(L1)     /* sample boom one length pot */
           15:TL2 := SAMPLE(L2)     /* sample boom two length pot */
        End;
End

Procedure RESTART3          /* sun pulse interrupt */
        Begin
        SUNP := min(LASTP, WDCSS ) /* sun period is min of the last two periods */
        /* Reset cntr and divide period into 128 parts.
Each is the number of milliseconds for one pseudo-sun-degree */
        DNMAX := min((SUNP + 64) / 128 , 255 );         /* set counter maximum */
        DNCTR := DNMAX;                         /* and the counter too */
        THETA := 0;                                     /* reset sun angle */
        LASTP := WDCSS;                         /* record last period */
        WDCSS := 0;                                     /* reset words since sun */
        End

Procedure VBBH;   /* high rate sampling of spheres */
        Begin
                DisableInts;                            /* disable 8080 interrupts */
                For I := 0 to 127 do  /* and sample 128 points */
                        RAM[I] := DBRS;
        /* Because of the interrupt disable, update vital variables such as the no.
of words since sun, the sun angle and the phase-locked-loop down counter */
        WDLOST := 64;                           /* words lost during disable */
        WDCSS := WDCSS + WDLOST; /* fix words since sun pulse */
        While WDLOST >= DNCTR do /* update sun angle */
                Begin
                        WDLOST := WDLOST - DNCTR;
                        DNCTR := DNMAX;
                        THETA := THETA + 1 mod 128;
                End
        DNCTR := DNCTR - WDLOST  /* set new down count */
        EnableInts;
        End
```

図 6　スピン衛星のプログラム([5]より引用)

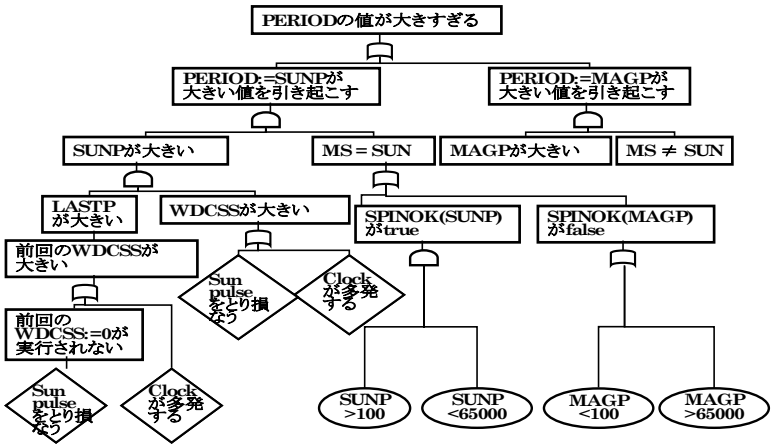

図 7 「PERIOD の値が大きすぎる」故障事象の SFT（[5]より引用）

4. 提案手法

4.1 提案手法の概要

3 章で述べた Leveson の SFTA 手法では，SFTA の考え方とプログラムの構造に応じた SFT のひな型については明らかになったが，具体的な SFT の作成手法については，述べられていなかった．そのため，経験の少ない技術者が新規に SFT を作成することは困難であった．

本研究では, SFTA 対象プログラムに対して SFTA が実施しやすいように前処理を行う．さらに，SFT 作成のルールを明確に規定する．これにより，技術者の経験に依存せずに，機械的に SFT を作成できるようにする．

図 8 に前処理の概要を示す．前処理では命令ブロックの識別, プログラム中の変数のスコープ区分(広域変数や局所変数) と使用・代入の区分，モジュールや関数等の実行形態の区分(周期，割込み)，割り込みの禁止と禁止解除の識別等の処理を行う．命令ブロックの識別では IF ブロック（多分岐構造やネスト構造を有する）や WHILE ブロック（ネスト構造を有する）等の複雑な命令を 1 つの命令ブロックとして識別し，一種のモジュールとして扱うことで，SFT のひな型の適用範囲を明らかにする．なお，多分岐やネスト構造を有する命令ブロックの SFT は図 2 から図 6 に示した基本構造の SFT のひな型を組み合わせて表現する．プログラム変数のスコープ区分・使用／代入の区分では，使用されている変数が広域変数か局所変数かを識別するとともに，変数が使用されているのか／変数に値が代入されているのかの区分を行う．プログラムの故障事象や原因事象は，命令ブロックで変数を使用することで生じる．そして命令ブロックで使用される変数は，別の場所でその変数に値が代入されている．このようにこれらの事象は変数の値の使用と代入により伝搬する．変数のスコープと使用・代入の識別を行うことで，原因事象の追跡を容易にすることができる．モジュールや関数等の実行区分では，プログラムを構成する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．前述の変数のスコープ区分と実行形態の区分を明らかにすることで，変数の値が更新されるタイミングを明らかにすることができる．例えば，変数 A が周期処理と割り込み処理の双方で更新される場合には，直近に更新された値を使用する必要がある．割り込み禁止と禁止解除では，割り込みの優先度を決定することで多重割り込みが生じた場合に変数の値が更新されるタイミングを明らかにすることができる．また，何らかの理由により割り込み禁止や割込み禁止解除ができなくなった場合の影響についても明らかにすることができる．

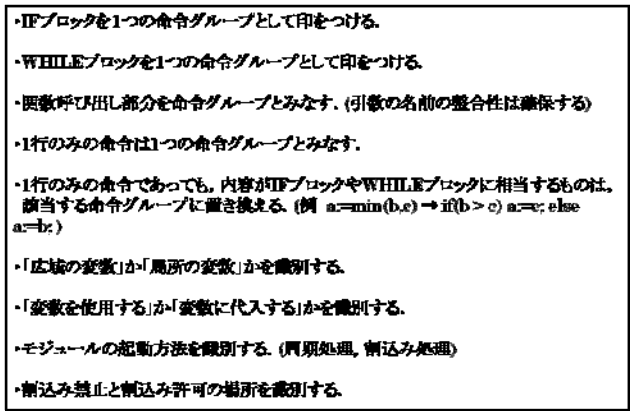

図 8 前処理の概要

図 9 に FT 作成ルールの概要を示す．FT 作成ルールは前処理が終わったプログラムの故障事象に対する SFT を作成する際の手順を表している．以下にルールを示す．はじめに STEP1 で故障を発生させる命令グループを識別する．そして，その命令グループが故障を出力する原因となる入力（使用されている変数）とその条件（例えば，x > 100，y の値が大きい　等）を識別する．次に，STEP2(a)で該当する命令グループの構造に応じた SFT のひな型を準備する．故障の原因にはプログラムが実行された結果生じるもの（例えば, 入力値が大きいため出力値が大きくなっており，

それを出力したために故障が生じる）と，実行されなかった結果生じるもの（例えば，WHILE 文の条件節が False となっており, WHILE 文が実行されないために前回計算した出力値が用いられたために故障が生じる）がある．そこで STEP2(b)で上記のケースを検討し，必要があれば故障の原因に追加する. そして STEP2(c)で該当する SFT のひな型に故障の原因事象群を記入する. 三番目に STEP3 では, STEP2 で作成した SFT の最下層に記述した原因事象の 1 つ 1 つについて，原因事象を生じさせる変数（使用する変数）を明らかにする．そして，その変数が計算されている（変数に値が代入されている）命令グループを識別する．その際，前処理で行った変数のスコープ区分と使用・代入の区分に関する情報を使用する．該当する変数が広域変数の場合は STEP3-1 を，該当する変数が局所変数の場合は STEP3-2 を実行する．該当する変数が複数の場所で代入されている場合はすべての場所を識別する．この作業を次の命令グループを追跡できなくなるまで行う（原因事象がこれ以上詳細化できなくなるまで行う）.

本研究では Leveson の提案した SFTA 手法に対して SFT の作成を容易にするための前処理と SFT を作成するためのルールを定義した．これらにより，SFT を機械的に作成できるようにした.

以下のSTEP1~STEP3のループを根源事象となるまで繰り返す.

STEP1: 故障を生じる変数が代入される命令グループと変数の値を同定する.

STEP2: (a) 命令グループに対応したFT雛型を準備する.
(b) 命令グループが必ず実行されるかを確認する.
実行できないケースがある場合は条件に追加する.
(c) FT雛型に故障を生じる条件を記述する. (前の条件も含む)
複数の条件をまとめることが出来ればまとめる.

STEP3-1: 事前条件が広域の変数の場合
(a)直前の「変数が代入される命令グループ」に移動する(STEP1へ戻る)
(b)その他の「変数が代入される命令グループ」に移動する. (STEP1へ戻る)

STEP3-2: 事前条件が局所の変数の場合
直前の「変数が代入される命令グループ」に移動する. (STEP1へ戻る)

補足: 必要に応じてFT内に説明用ブロックを挿入する事ができる. 説明用ブロックは, FTの論理構成には関係しない

図 9　FT 作成ルールの概要

4.2. 提案手法による SFTA の適用事例

本研究で提案した手法を評価するために Leveson らと同様にスピン衛星の「PERIOD の値が大きすぎる」故障事象に対する SFT を作成した結果を述べる.

はじめに，図 8 に従い前処理を行った（結果は省略）.

二番目に図 9 の STEP 1 を適用し「PERIOD の値が大きすぎる」故障事象を生じる命令グループと変数を同定した．該当する命令グループは関数 PERIOD の IF-THEN-ELSE 文(図 10 の①)内の MS であった. 変数 MS の値は「MS = SUN」の時の「PERIOD := SUNP」，および，「MS ≠ SUN」の時「PERIOD := MAGP」の 2 か所で更新されている.

```
Function PERIOD:integer; /* redundant routines for calculating spin
  period */
  Begin
        If SPINOK (SUNP)
                Then MS := SUN
                Else if SPINOK (MAGP)
                        Then MS := MAG
                        Else MS := SUN

        If MS = SUN
  ①             Then PERIOD := SUNP
                Else PERIOD := MAGP
  End
```

図 10　Function PERIOD のモジュール

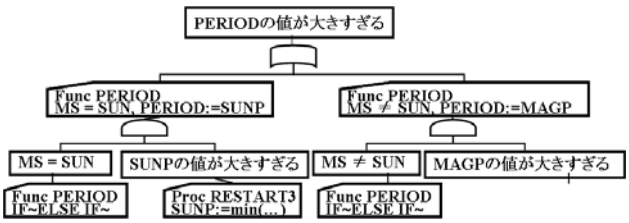

図 11　SFTA の適用事例（途中経過 1）

二番目に STEP2(a)で命令グループに対応した FT ひな型を準備する．命令グループはブロック IF 文であるので，IF-THEN-ELSE 文の SFA のひな型を適用する．STEP2(b)で命令グループが必ず実行されるかを確認する．この場合，THEN 節と ELSE 節の内容は，代入文であるので必ず実行される．STEP2(c)で FT ひな型に故障を生じる条件を記述する．ここでは，SFT のひな型にある「条件節が TRUE」となる場合と「条件節が FALSE」となる場合を記述する．SFT のひな型に従うと，前者は「MS = SUN」と「PERIOD := SUNP」となる. この時,「PERIOD := SUNP」なので「PERIOD の値が大きすぎる」故障事象は，「SUNP が大きい」原因事象に置き換えられる．後者は「MS NOT = SUN」と「PERIOD := MAGP」となる．この結果，「PERIOD の値が大きすぎる」故障事象は「MAGP が大きい」原因事象に置き換えられる．図 11 にここまでに作成した SFT を示す．なお，SFT の故障事象の下に分岐した枝の説明「Function PERIOD において MS = SUN で PERIOD := SUNP」と「Function PERIOD において MS NOT = SUN で PERIOD := MAGP」を追記した．その際，本研究で追加した図 1(b)の説明用のブロックを使用した.

三番目に上述した SFT の「SUNP が大きい」原因事象の SFT を作成する．以降，この SFTA の枝の作成に絞って説明する．前処理の結果から，SUNP は広域変数であることが分かっており，PROCEDURE RESTART3 の中で代入されていることが分かっている．従って，STEP3-1(b)により PROCEDURE RESTART3 に移動する．SUNP が大きいについて FT を作成する.

四番目に SFT 作成ルールの 2 回目の実行を行う．STEP1 で，SUNP の値を代入場所は PROCEDURE RESTART3 の「SUNP := min(LASTP, WDCSS)」であることを識別する.

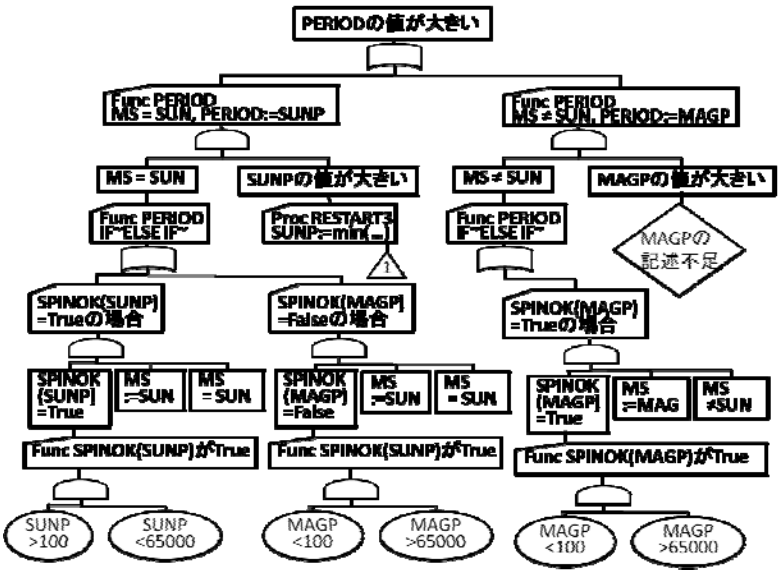

図 12 提案手法により作成した SFTA(1/3)

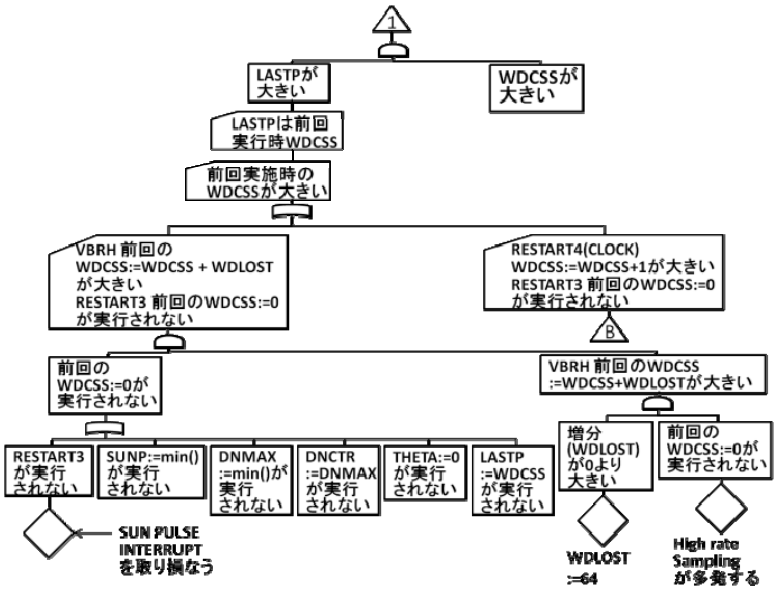

図 13 提案手法により作成した SFTA(2/3)

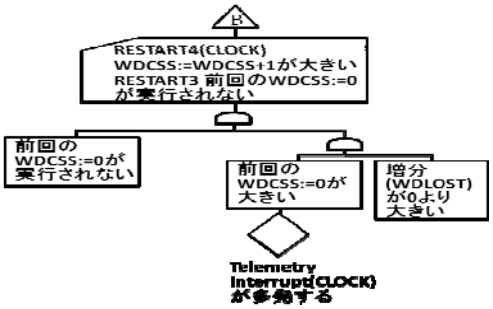

図 14 提案手法により作成した SFTA(3/3)

STEP2(a)で命令グループに対応した FT ひな型を準備する．ここで,「SUMP := min(A, B)」は「IF (A < B) THEN (SUMP := A) ELSE (SUMP := (SUMP :=B)」と考えることができるため，IF-THEN-ELSE 文のひな型を準備する．STEP2(b)で命令グループが必ず実行されるかを確認する．この IF-THEN-ELSE 文とその中の代入文は必ず実行されるので，

条件追加はない．STEP2(c)でひな型に故障を生じる条件「LASTP が大きい」および「WDCSS が大きい」を SFTA のひな型に記入する．

五番目に「LASTP が大きい」について分析する．以降，この枝の作成に絞って説明する．LASTP は前処理で局所変数であり同モジュール内で処理されている事がわかっている．従って，STEP3-1(a)を行う．同じモジュール内にある「LASTP が代入される命令グループ」に移動する．

以降，SFTA 作成ルールの 3 回目の実行を行う．この作業を原因事象がそれ以上展開できなくなるまで繰り返し実施する．その結果，図 12，図 13，図 14 に示す FT が出来上がった．最終的な故障の原因は「SUN PULSE INTERRUPT を取り損なう」，「HIGH RATE SAMPLING が多発する」，「CLOCK（TELEMETRY INTERRUPT）が多発する」となった．

4.3 提案手法の評価

提案手法を評価するため，提案手法を用いて作成した SFT と，Leveson らが作成した SFT を比較した．その結果，双方の SFT の形はほぼ同じであることが分かった．しかし，いくつかの差異が現れた．その理由について以下に述べる．

Leveson の SFT 結果で示された「CLOCK（TELEMETRY INTERRUPUT）が多発する」の形状に差異があった．Leveson の SFTA 手法では，SFT 作成ルールが定義されていなかった為，SFT 作成者によって作成過程がまとめられていたと考えられる（SFT 作成者が思考過程を省略している）．提案手法では，SFT 作成ルールで手順を明確にした為，SFT の省略がなく SFT の枝が全て記述されていた．SFT をまとめることで故障原因を検討した際の思考過程に関する情報が抜け落ちるため，SFT の理解が難しくなる．この点において提案手法のほうが，故障原因を分析するための道具としては優れていると考えられる．

故障事象「LENGTH の値が小さすぎる」に対しても SFT を作成した．その結果，「WDCTR が途中でリセットされる」，「DBRS が起動しない」の部分で SFT の差違があった．しかし，これらについては Leveson らの論文中に詳細な情報がないため，詳細な分析ができなかったことが原因である．そのため，差異が生じたことはやむを得ない．

以上により，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 SFT を機械的に作成できることが確認できた．

5. まとめと今後の課題

本論文では，組込みソフトウェアに適用可能な SFTA 手法を提案した．この手法では SFT を作成する際に前処理を行うことで，対象ソフトウェアの動作上の特徴を明らかにして，後続する SFT の作成を容易にする．さらに，SFT の作成では作成ルールを明確化した．これにより，FT を機械的に作成することが可能となり，技術者の経験に依存せずに適切な SFTA が作成できるようになった．

今後の課題としては，提案 SFTA 手法を他の組込みソフトウェアの故障原因解析に適用し，提案手法の適用可能範囲を明らかにする．さらに，そこで得られた知見から SFT 作成時の前処理とルールの改善を行っていく．さらに適切な SFT の作成を支援するツールを開発していく．

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) Michael A. Friedman: Automated Software Fault Tree Analysis for Pascal Program, Proc. Annual Reliability and Maintainability Symposium, pp.458-461(1993).
5) Nancy G. Leveson and Peter R, Harvey: Analyzing Software Safety, IEEE Transaction on Software Engineering, Vol. 9, No.5, pp.569-579(1983).
6) 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).
7) Guy Helmer, Johnny Wong, Mark Slagell, Vasant Honavar, Les Miller, and Robyn Lutz; A Software Fault Tree Approach to Requirement Analysis of an Intrusion Detection System, Journal of Systems and Software, Vol.67, No.1, pp.109-122(2003).
8) Nicolas Dulac, Nancy G. Leveson,: An Approach to Design for safety in Complex Systems, Proc. of International Symposium on Systems Engineering , pp393-407(2004).
9) 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).
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).