平成 28 年度　卒業論文

既存リスク分析手法と STAMP の比較

Comparison of existing risk analysis method and STAMP

指導　松野　裕　准教授

日本大学理工学部応用情報工学科

3001　阿部惇朗

目次

第 1 章　序論

1.1　研究背景

これまでのシステムは、閉じた環境内でのシステムが主であったが、近年、大規模・複雑化したシステム、オープンシステムが多く見られるようになった。システム障害もシステムの構成要素のみならず、構成要素間やシステムと人間との相互作用に起因するものが発生している。システム全体の安全性、セキュリティのリスク分析はより重要となっており、STAMP などの相互作用に着目した分析手法が注目されている。

しかし、構成要素に着目した従来の分析手法との比較などはまだ始まったばかりである。

1.2　研究目的

本研究では、事例を実際に既存のリスク分析手法と STAMP でリスク分析を行い、比較を行う。それを今後の STAMP 分析の事例の一つとして参考になったら幸いである。

1.3　構成

本論文は、本章も含めて 4 章から構成される。

第 1 章では、序論として本研究の背景・目的を述べる。

第 2 章では、準備として STAMP、FTA、ET ロボコンの説明について述べる。

第 3 章では、STAMP、FTA の施行結果を述べる。

第 4 章では、結論として本研究の成果、今後の課題について述べる。

第 2 章　準備

2.1　STAMP

2.1.1　STAMP の概要

20 世紀に開発されたシステムの多くは、構成要素が少なく、それらの役割も明確であり、それゆえアクシデントが起きた場合は、構成要素のどれかが根本原因であり、アクシデントに至ったのかを分析することは容易であった。

しかし、21 世紀になると、開発されるシステムの規模は非常に大きくなり、構成要素も爆発的に増大するとともに、要素間の相互作用も複雑になり、個々の要素の役割を理解しただけでは、もはやシステムを理解できなくなった。そのような相互作用が複雑なシステムにおけるアクシデントの原因は、一つの構成要素に限定できる要因だけではなく、複数の要素間の相互作用による要因も考える必要があった。

マサチューセッツ工科大学の Leveson 教授は、システムの安全性は構成要素の相互作用から創発されるものであり、個々の要素を分割して分析するべきではないと述べた。そして、現代のシステムのアクシデントの多くは、システム構成要素の故障によって起きるのではなく、システムの中で安全のための制御を行う要素（コントローラー : Controller）と制御される要素（被コントロールプロセス : Controlled Process）の相互作用が働かないことによって起きるというアクシデントモデルを提唱した。このモデルを STAMP（Systems-Theoretic Accident Model and Process）: システム理論に基づくアクシデントモデルと呼ぶ。

STAMP モデルでは、システムの様々な階層でコントローラーと被コントロールプロセスに該当する要素が存在しており、それらの相互作用が適切に働くことによりシステムの安全が実現されるとする。STAMP モデルにおいて、アクシデントは相互作用が適切に働かないことによって起こり、具体的にはコントローラーから被コントロールプロセスへの必要な制御指示（コントロールアクション : Control Action）が適切に与えられないために起こるとしている。そして、不適切なコントロールアクションが与えられる要因として、コントローラー自身が想定する被コントロールプロセスの状態（プロセスモデル : Process Model）が、実際の被コントロールプロセスの状態を正しく反映できてないことが主要な要因であるとしている。たとえコントローラーも被コントロールプロセスも

故障せずに、仕様通りに正しく動作していても、このような認識の不整合により不適切なコントロールアクションが与えられ、最終的にアクシデントにつながるというアクションモデルなのである。

2.1.2　STAMP の手順

ここでは、STAMP の手順について説明する。

・step0 準備 1：アクシデント・ハザード・安全制約の識別

最初のステップとして、アクシデント、ハザード、安全制約の 3 つを決める。これらを決めるのは他のシステムエンジニアリングのハザード分析にも共通するが、STAMP/STPA でのハザードの定義は、他の分析法のものと異なっている可能性に注意する必要がある。以下に、アクシデント、ハザード、安全制約の定義を示す。

・ アクシデント

望んでもいないし計画もしていない、損失につながるようなイベントのことである。損失は、人命喪失、怪我、物損、環境汚染、ミッション喪失、経済的損失といったものである。STAMP/STPA では安全工学の技法を出来るだけ広く適用する意図で、アクシデントも広めの定義としている。

・ ハザード

環境のある最悪な条件と重なることでアクシデントにつながるような、システムの状態もしくは条件のことである。この定義には、安全性と信頼性の混同を回避するために単なる故障とハザードを区別し、分析可能性も高めるといった意図があり、以下のような二つの点を念頭においた定義となっている。まず、ハザードがコントロール対象のシステムの境界内のものであることを定義の後半で述べている。コントロール対象のシステムの範囲、コントロール範囲外の環境との境界が明確になっている必要がある。また、ハザードが実際に損失につながるのは、ハザードと組み合わさる、最悪の環境条件の存在が必要であることが定義の前半で述べられている。

・ 安全制約

ハザードが認識されると、それらからシステムを安全に保つための要件もしくは制約を導ける。トップダウンに考える場合、まず高レベルの安全制約が導かれることになる。安全制約はこの段階ですべて確定するとは限らず、STPA の実施によっても導出、修正されることもある。

・step0 準備２：コントロールストラクチャの構築

システムにおいて、安全制約の実現に関係するコンポーネントおよび、コンポーネント間の相互作用を分析し、制御構造図を構築する。アクシデント、ハザード、安全制約を考えるのはアクシデントのモデルやハザード解析法によらず一般的な手順である。しかしながら、コントロールストラクチャを構築するのは STAMP/STPA のユニークな点となっている。システムのコンポーネントに機能を割り当てるといったことは一般のシステムエンジニアリングの活動でも行われるが、そのようなエンジニアリングのドキュメントとしてコントロールストラクチャを使うことは一般に行われていない。

STAMP のコントロールストラクチャは、システムの機能的な設計をうまくグラフィカルにモデルとして表現するもので、優れたドキュメントとして多くの人に役立つものである。このステップにおいて抽象度を下げないことが大切である。

・step1：非安全なコントロールアクションの抽出

コントロールストラクチャをベースに、制御対象のプロセスに対するコントローラーからのアクションのうち、以下の４つのタイプの安全でないコントロールアクションを抽出する。

1. 与えられないとハザード : 安全のために必要とされるコントロールアクションが与えられないことがハザードにつながる。
2. 与えられるとハザード : ハザードにつながる非安全なコントロールアクションが与えられる。
3. 早すぎ、遅すぎ、誤順序でハザード : 安全のためのものであり得るコントロールアクションが、早すぎて、遅すぎて、または順序通りに与えられないこ

とでハザードにつながる。

4. 早すぎる停止、長すぎる適用でハザード：安全のためのコントロールアクションの停止が早すぎる、もしくは適用が長すぎることがハザードにつながる。

上記の 4 つ以外に 5 番目のタイプとして、「必要なコントロールアクションが与えられたけども追従されない」という場合もあるが、この 5 番目の可能性については次のステップで分析する。

step1 の分析に使うフォーマットに決まりはないが、非安全なコントロールアクションのドキュメント化のためには、以下のような表を用いるのが便利とされている。

表 1　Step1 非安全なコントロールアクション識別に用いる表の例

コントロールアクション	与えられないとハザード	与えられるとハザード	早すぎ、遅すぎ、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
コントロールアクション	条件	条件	条件	条件
・・・・	・・・・	・・・・	・・・・	・・・・

UCA は、安全のための必要なルールと関連付けて考えることができるため、UCA の分析から安全制約を作り出すことも可能である。このため、安全制約を最初の準備の段階でなくここで作成したり、作成済みの安全制約を UCA の分析結果をもとに見直したりすることもできる。

・step2：非安全なコントロールの原因の特定

非安全なコントロールアクション （UCA）を抽出したのち、非安全なコントロール（につながるシナリオ）の原因の特定をする。必ずしもこれらのステップを完全に別のものとして逐次的に行う必要はなく、一部の UCA を抽出した段階で原因の分析を行う場合もあり得る。このステップで、前述の 5 番目のタイプのシナリオである、安全のために必要なコントロールアクションの不適切な実行を考慮する。

基本的に、安全制約を保つためのコントロールループやその構成要素を確認し、以下のような点を分析する。

1. プロセスモデルが正しくなくなってしまう原因も含め、どのようにして非安全なコントロール、ひいてはハザードにつながるかを、分析する。
2. 必要なコントロールアクションが与えられたにもかかわらず、適切に実行されないのかの原因を分析する。

以下は、コントロールループに齟齬の原因となりうるものの 11 つのガイドワードである。この項目を検討し、原因を抽出する。

(1) コントロール入力や外部情報の誤りや喪失
(2) 不適切なコントロールアルゴリズム
(3) 不整合、不完全、または、不正確なプロセスモデル。不適切な操作。
(4) コンポーネントの不具合。経年による変化。
(5) 不適切なフィードバック、あるいはフィードバックの喪失。フィードバックの遅れ。
(6) 不正確な情報の供給、または情報の欠如。測定の不正確性。フィードバックの遅れ。
(7) 操作の遅れ。
(8) 不適切または無効なコントロールアクション、コントロールアクションの喪失。
(9) コントロールアクションの衝突。プロセス入力の喪失または誤り。
(10)未確認、または範囲外の障害。
(11)システムにハザードを引き起こすプロセス出力。

分析によって得られたシナリオは、システムを保護するためのコントロールを識別するために用いられる。通常は、このような保護のためのコントロールの設計は STPA の解析そのものではない。STPA の解析結果を用いてドメインの専門家によってなされる。

次に非安全なコントロールにつながるシナリオを記述する。UCA 一つに対して複数のシナリオを記述していき、その対策を分析する。

2.2　FTA

2.2.1　FTA の概要

故障の木解析（FTA: Fault Tree Analysis）とは、製品の故障、およびそれにより発生した事故の原因を分析する手法である。機器の信頼性、安全性を高めるために利用されている。また、定量的な故障の発生頻度分析のために、原因の潜在危険を論理的にたどり、それぞれの発生確率を評価する手法でもある。このため製品の信頼性向上のため、さらに消費者が安全に製品を使用するために製品の安全性改善に利用されている。FTA は、米国 BELL 研究所の H.A WATSON が考案し、1965 年ボーイング社により完成された。

FTA では、製品の好ましくない事象を初めに仮定し、それについて考えられる故障・事故に至った道筋を、発生確率とともに、故障の木図（FT 図）で表し分析していく。FTA では、製品の上位の故障・事故から、下位の原因へとトップダウン的に展開していく。装置の故障の発生確率は、FT 図に示されたいろいろな故障の原因事象を、ブール代数を用いて重複をなくすことにより理論的にかつ正確に算出することができる。

2.2.2　FTA の手順

まず初めに、起こしてはならない事象、頂上事象を定め、FT 図の最上位に置く。次に中間事象、すなわち第一次要因事象を列挙し、さらに第二次要因事象、第三次要因事象と因果関係をトップダウン的に展開していく。最下位の展開できない基本事象まで列挙した後、それぞれの基本事象の発生確率を計算する。そこから逆に確率を集計していき、頂上事象の確率を求める。頂上事象の発生確率が過大ならば、確率が最大のルートに対して対策を講じ、頂上事象の確率が十分に小さくなるまで解析を繰り返す。

2.2.3　確率の計算

下位事象 A、B などが一つでも起きれば上位事象 X が起きる場合は、事象 A の確率と事象 B の確率の加算をもって上位事象 X の確率とする。この関係を表すには、上位事象 X と下位事象 A、B の間に OR ゲートの論理記号を介在する。下位事象 A、B が同時に起きるときに限って上位事象 X が起きる場合は、事象 A の確率と事象 B の確率の積をもって上位事象 X の確率とする。この関係を表すには、上位事象 X と下位事象 A、B の間に OR ゲートの論理記号を介在する。

2.3　ET ロボコン

2.3.1　ET ロボコンの概要

ET ロボコンとは、「組込みシステム」分野における技術教育をテーマに、決められた走行体で指定コースを自律走行する競技である。

同一のハードウェアに、UML 等で分析・設計したソフトウェアを搭載し競う、ソフトウェアの優劣を競うコンテストである。

コースの難所の１つにルックアップゲートがある。ルックアップゲートとは、255mm の走行体で 235mm のルックアップゲートをくぐるもので、超音波センサでルックアップゲートを検知し、尻尾を出して傾斜して走り、ゲートを通過する。

本研究では、このルックアップゲートを通過する難所を分析対象とする。

図１　走行体とルックアップゲート通過の様子

2.3.2　ソースコード

本研究で使用した「ルックアップゲート通過」の部分のソースコードを以下に示す。

```
//ルックアップゲート
if(course == 2 && ev3_ultrasonic_sensor_get_distance(sonar_sensor) <= 20 ){
        //倒立から尻尾走行へ移行
        ev3_motor_rotate(tail_motor,62,+100,true);
        ev3_motor_set_power (EV3_PORT_B, 100);
        ev3_motor_set_power (EV3_PORT_C, 100);
        tslp_tsk(100);
        ev3_motor_set_power (EV3_PORT_B, 0);
        ev3_motor_set_power (EV3_PORT_C, 0);
        tslp_tsk(1000);
        /*  倒立振子 API 初期化
        balance_init();
        while(1){
                //尻尾走行開始からの距離
                rad_l =    ev3_motor_get_counts(left_motor);
                rad_r =    ev3_motor_get_counts(right_motor);
                distance2 = 8.2 * M_PI * (rad_l + rad_r) / 720;
                //ゴール判定・倒立
                if( 106 > distance2 - distance1 ){
                        ev3_motor_stop(left_motor, false);
                        ev3_motor_stop(right_motor, false);
                        zg = distance2 - distance1;
                        fprintf(bt, "distance = %f ¥n",zg);
                        ev3_motor_rotate(tail_motor,1,+70,true);
                        tslp_tsk(100);
                        ev3_motor_rotate(tail_motor,1,+70,true);
                        tslp_tsk(100);
                        ev3_motor_rotate(tail_motor,1,+70,true);
                        tslp_tsk(100);
```

```
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);
    ev3_motor_rotate(tail_motor,1,+70,true);
    tslp_tsk(100);

    ter_tsk(BT_TASK);
    fclose(bt);
    ext_tsk();
}
```

```
//超音波距離検知
if ((ev3_ultrasonic_sensor_get_distance(sonar_sensor) <= 3)
&& (ev3_ultrasonic_sensor_get_distance(sonar_sensor) >= 0)
&& (ev3_motor_get_counts(tail_motor) >= 75)){
        ev3_motor_rotate(tail_motor,-1,+100,true);
        }
//ライントレース
//前進
fprintf(bt, "color = %d ¥n",ev3_color_sensor_get_reflect(color_sensor));
        if( 45 > distance2 - distance1 && flag_gate == 0 ){
                if (ev3_color_sensor_get_reflect(color_sensor) >= 14){
                        ev3_motor_set_power(left_motor, 10);
                        ev3_motor_set_power(right_motor, 30);
                }else{
                        ev3_motor_set_power(left_motor, 30);
                        ev3_motor_set_power(right_motor, 10);
                }
        }else if(45 <= distance2 - distance1 && flag_gate == 0){
                flag_gate = 1;
                rad_l =    ev3_motor_get_counts(left_motor);
                rad_r =    ev3_motor_get_counts(right_motor);
                distance3 = 8.2 * M_PI * (rad_l + rad_r) / 720;
                distance4 = distance3 - 10;
        }else if( distance3 - distance4 < 45    && flag_gate == 1){   //後退する
                if (ev3_color_sensor_get_reflect(color_sensor) >= 14){
                        ev3_motor_set_power(left_motor, -10);
                        ev3_motor_set_power(right_motor, -30);
                }else{
                        ev3_motor_set_power(left_motor, -30);
                        ev3_motor_set_power(right_motor, -10);
                }
                rad_l =    ev3_motor_get_counts(left_motor);
                rad_r =    ev3_motor_get_counts(right_motor);
```

```
            distance4 = 8.2 * M_PI * (rad_l + rad_r) / 720;
        }else if(distance3 - distance4 >= 45    && flag_gate == 1){
            flag_gate = 2;
        }else if(flag_gate == 2){
            if (ev3_color_sensor_get_reflect(color_sensor) >= 14){
                ev3_motor_set_power(left_motor, 10);
                ev3_motor_set_power(right_motor, 30);
            }else{
                ev3_motor_set_power(left_motor, 30);
                ev3_motor_set_power(right_motor, 10);
                }
            }
            tslp_tsk(4);
        }
}
```

第 3 章　研究結果

3.1　STAMP の試行結果

3.1.1　step0　準備 1

まず、STAMP/STPA の手順に沿って、アクシデント・ハザード・安全制約を識別する。「ルックアップゲートを通過する」という目的を分析対象とした時、アクシデント・ハザード・安全制約は以下の表の通りに設定した。以下に示す。

表 2　アクシデント・ハザード・安全制約の識別

アクシデント	ハザード	安全制約
ゲートに接触	車体が適切な角度まで傾いていない	車体が傾くまで走行してはいけない
コースアウト	ライントレースできていない	EV3 は常にラインをトレースしなければならない
傾斜時に転倒	スピードが速い	ゴール後に一定の速度になっている

アクシデント「ゲートに接触」に対しては、ハザード「車体が適切な角度まで傾いていない」と、安全制約「車体が傾くまで走行してはいけない」を識別した。同様に、アクシデント「コースアウト」に対しては、ハザード「ライントレースできていない」と、安全制約「EV3 は常にラインをトレースしなければならない」を識別した。アクシデント「傾斜時に転倒」に対しては、ハザード「スピードが速い」と、安全制約「ゴール後に一定の速度になっている」を識別した。

3.1.2　step0　準備 2

次にコントロールストラクチャを構築、作図する。「ルックアップゲート」を通過する動作を中心にコンポーネントを考え、コントロールループを意識して構築する。なお今回のコントロールストラクチャにはコントロールループは無いがそれでも STAMP/STPA で分析するとこは可能である。

ライン・ルックアップゲートのコンポーネントからセンサには色・時間などの情報が渡される。同様に、センサから EV3 には輝度・時間の情報が渡される。EV3 からモーターには走行指示・減速指示・傾斜指示が送られ、これが今回のコントロールアクションとなる。モーターから EV3 には回転回数の情報が渡される。

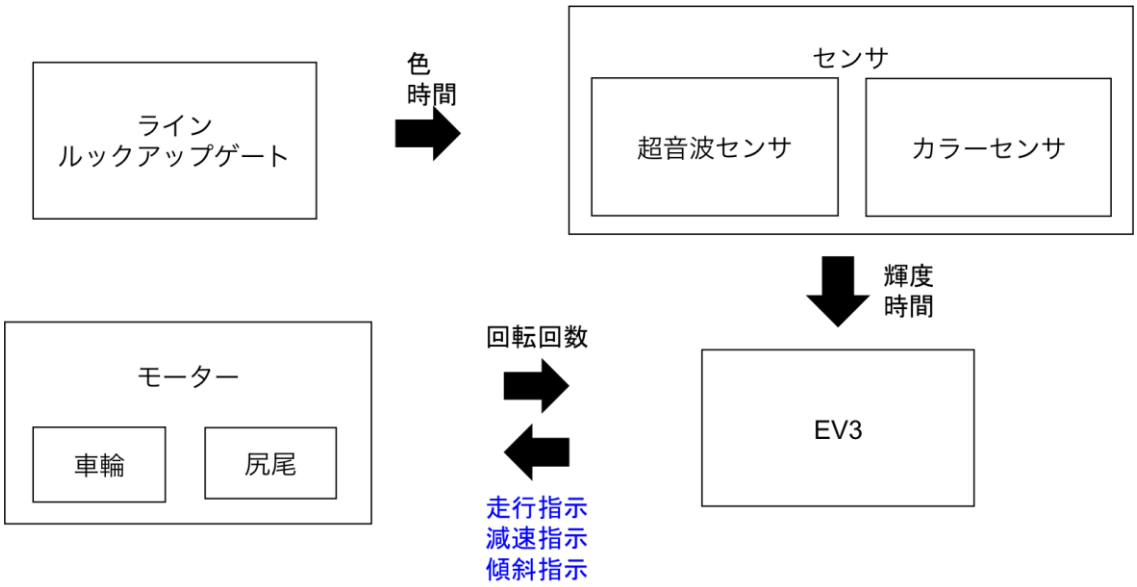

図 2　コントロールストラクチャ

3.1.3　step 1

次に安全でないコントールアクション(UCA)の識別を行う。

1.2 で構築したコントールストラクチャ内のコントロールアクションを行、4 つのガイドワード(与えられないとハザード・与えるとハザード・早すぎ、遅すぎ、誤順序でハザード・早すぎる停止、長すぎる適用でハザード)を列に表を作り、UCA の識別を行った。結果、18 個の UCA を識別することができた。以下の表に示す。

表 3　UCA の抽出

コントロールアクション	与えないとハザード	与えるとハザード	早すぎ，遅すぎ，誤順序でハザード	早すぎる停止，長すぎる適用でハザード
傾く	超音波センサから EV3 に測定結果が伝わらないため，傾かない UCA1	超音波センサから EV3 に誤った測定結果を伝えたため，傾かない UCA3	超音波センサから EV3 に測定結果が遅れて伝わるため，ゲートにぶつかる UCA5	EV3 からモーターへのコントロールアクションが早すぎる停止により適切な角度まで傾かない UCA7
	EV3 からモーターに命令が伝わらないため，傾かない UCA2	EV3 からモーターに誤った命令がつたわるため，傾かない UCA4	EV3 からモーターに命令が遅れて伝わるため，ゲートにぶつかる UCA6	
減速	モーターから EV3 に測定結果が伝わらないため，距離が測れず減速できない UCA8	モーターから EV3 に誤った測定結果が伝わるため，任意の場所以外で減速する UCA9	モーターから EV3 に遅れて測定結果が伝わるため，任意の場所以外で減速する UCA10	
走行	カラーセンサから EV3 に測定結果が伝わらないため，コースアウトする UCA11	カラーセンサから EV3 に誤った測定結果を伝えたため，コースアウトする UCA13	カラーセンサから EV3 に測定結果が遅れて伝わるため，コースアウトする UCA15	走行命令が早すぎる停止により，コースアウトする UCA17
	EV3 からモーターに命令が伝わらないため，コースアウトする UCA12	EV3 からモーターに誤った命令を伝えたため，コースアウトする UCA14	EV3 からモーターに命令が遅れて伝わるため，コースアウトする UCA16	走行命令が長すぎる適用により，コースアウトする UCA18

3.1.4　step2

1.3 で識別した UCA をもとに、HCF の特定を行う。

まず、1.2 で構築したコントールストラクチャに 13 個のガイドワードを当てはめる。その結果を以下に示す。

13 個のガイドワードを当てはめる必要は必ずしもないが、当てはめることで動作の流れがわかりやすくなり、また次のステップで作成する表を作成させやすくなる。

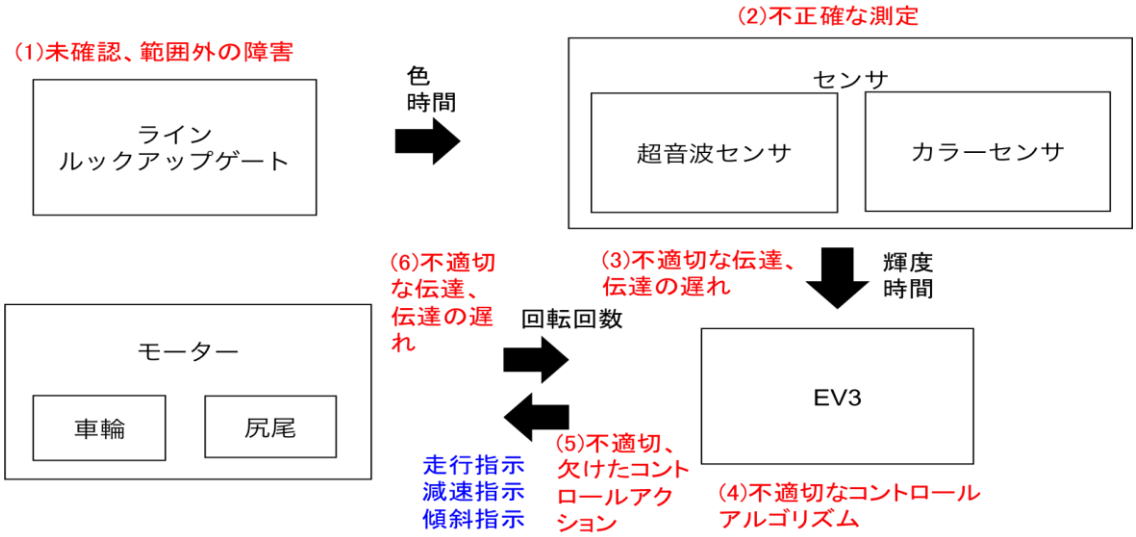

図 3　ガイドワードを当てはめたコントロールストラクチャ

また、先ほど当てはめたガイドワードと UCA をもとに HVF の特定を行う。今回の結果では、ET ロボコンというソフトウェアの競技の分析を行っているため、対処できないと思われるハードの部分は HCF として除外している。結果を以下に示す。

表 4　HCF の特定 1

	(1)　未　確認・範囲外の障害	(2)不正確な測定	(3)不適切な伝達・伝達の遅れ	(4)不適切なコントロールアルゴリズム	(5)不適切、欠けたコントロールアクション	(6)不適切な伝達、伝達の遅れ
超音波センサから EV3 に測定結果が伝わらないため、傾かない UCA1			超音波センサから EV3 への伝達が不適切			
EV3 からモーターに命令が伝わらないため、傾かない UCA2				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	
超音波センサから EV3 に誤った測定結果を伝えたため、傾かない UCA3	ルックアップゲートの状態により想定外の測定結果が伝わる	超音波センサの測定結果が不正確	超音波センサから EV3 への不適切な伝達			
EV3 からモーターに誤った命令を伝えたため、傾かない UCA4				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	

表 5　HCF の特定 2

	(1) 未確認・範囲外の障害	(2)不正確な測定	(3)不適切な伝達・伝達の遅れ	(4)不適切なコントロールアルゴリズム	(5)不適切、欠けたコントロールアクション	(6)不適切な伝達、伝達の遅れ
超音波センサから EV3 に測定結果が遅れて伝わるため、ゲートにぶつかる UCA5			超音波センサから EV3 への伝達の遅れ			
EV3 からモーターに命令が遅れて伝わるため、ゲートにぶつかる UCA6				プログラムのアルゴリズムが不適切	EV3 からのコントロールアクションの遅れ	
EV3 からモーターへのコントロールアクションが早すぎる停止により適切な角度まで傾かない UCA7				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	

表 6　HCF の特定 3

	(1)　未　確認・範囲外の障害	(2)不正確な測定	(3)不適切な伝達・伝達の遅れ	(4)不適切なコントロールアルゴリズム	(5)不適切、欠けたコントロールアクション	(6)不適切な伝達、伝達の遅れ
モーターから EV3 に測定結果が伝わらないため、距離が測れず減速できない UCA8						モーターからの不適切な伝達
モーターから EV3 に誤った測定結果が伝わるため、任意の場所以外で減速する UCA9	コースの状態					モーターからの不適切な伝達
モーターから EV3 に遅れて測定結果が伝わるため、任意の場所以外で減速する UCA10						モーターからの伝達の遅れ

表 7　HCF の特定 4

	(1)　未　確認・範囲外の障害	(2)不正確な測定	(3)不適切な伝達・伝達の遅れ	(4)不適切なコントロールアルゴリズム	(5)不適切、欠けたコントロールアクション	(6)不適切な伝達、伝達の遅れ
カラーセンサから EV3 に測定結果が伝わらないため、コースアウトする UCA11			カラーセンサから EV3 への伝達が不適切			
EV3 からモーターに命令が伝わらないため、コースアウトする UCA12				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	
カラーセンサから EV3 に誤った測定結果を伝えたため、コースアウトする UCA13	外乱光、コースの状態などにより想定外の測定結果が伝わる	カラーセンサの測定結果が不正確	カラーセンサから EV3 への不適切な伝達			
EV3 からモーターに誤った命令を伝えたため、コースアウトする UCA14				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	

表 8　HCF の特定 5

	(1)　未　確認・範囲外の障害	(2)不正確な測定	(3)不適切な伝達・伝達の遅れ	(4)不適切なコントロールアルゴリズム	(5)不適切、欠けたコントロールアクション	(6)不適切な伝達、伝達の遅れ
カラーセンサから EV3 に測定結果が遅れて伝わるため、コースアウトする UCA15			カラーセンサから EV3 への伝達の遅れ			
EV3 からモーターに命令が遅れて伝わるため、コースアウトする UCA16				プログラムのアルゴリズムが不適切	EV3 からのコントロールアクションの遅れ	
走行命令が早すぎる停止により、コースアウトする UCA17				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	
走行命令が長すぎる適用により、コースアウトする UCA18				プログラムのアルゴリズムが不適切	EV3 からの不適切なコントロールアクション	

3.1.5　シナリオと対策

1.4 で特定した HCF に対するシナリオと、その対策を UCA 毎に記述する。

UCA3:超音波センサからEV3に誤った測定結果を伝えたため、傾かない

シナリオ 1
(1)外部の要因により想定外の測定結果が伝わる。

対策：ゴールを通過するまで超音波センサの測定結果を反映しない。

シナリオ 2
(2) 超音波センサの故障などにより誤ったの測定結果が伝わる。

対策：想定外の値の測定結果が伝達された場合、距離を計測して傾くプログラムに切り替える。

図 4　シナリオと対策(一部抜粋)

3.2　FTA 分析の試行

STAMP/STPA 分析との比較を行うために、同じ事例で FTA 分析を行う。比較を行うということで、今回頂上事象として「ルックアップゲートを通過しない」と定め、分析を行った。

頂上事象のツリー分析を行った結果、３つの中間事象「走行不能」「コースアウト」「ゲートに接触」が得られた。それを以下に示す。

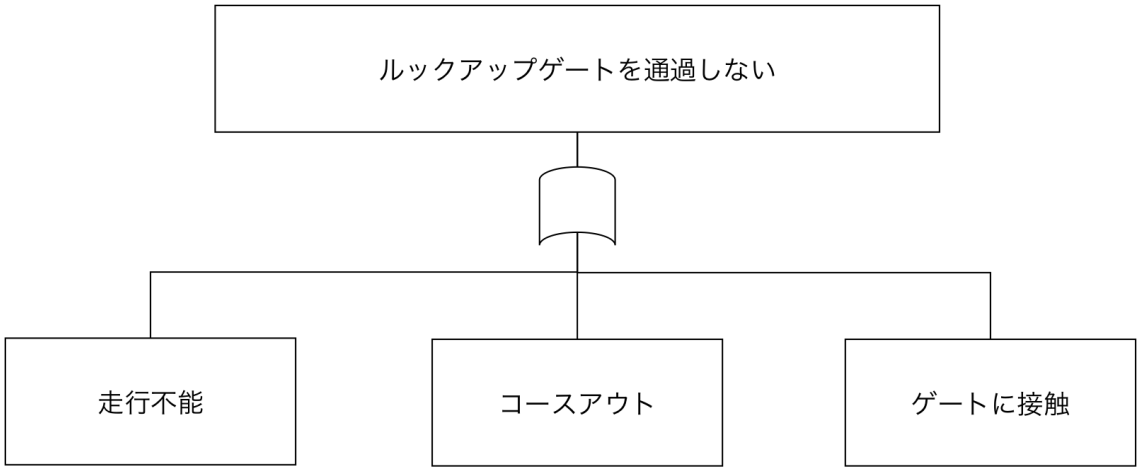

図 5　頂上事象の FTA 分析

また、中間事象ごとに分析を行った。まず以下に中間事象「走行不能」について分析をおこなったツリーを示す。前述の通りハード系の故障に関しては、これ以上展開しないものとする。

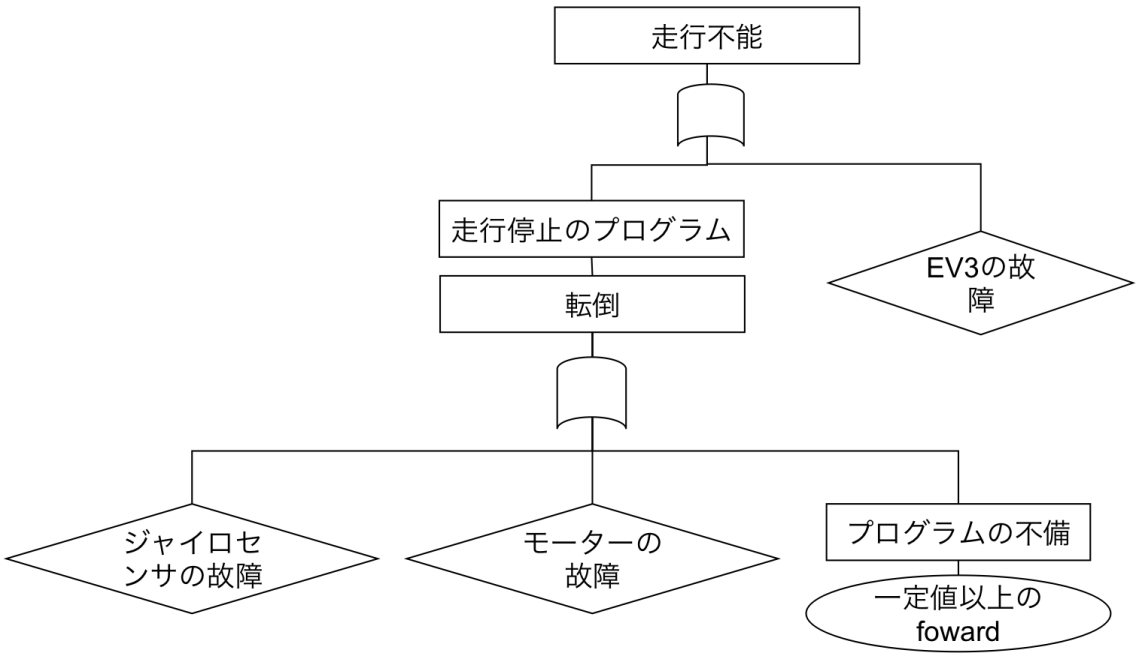

図 6　中間事象「走行不能」の FTA 分析

結果として、「一定値以上の foward」に原因があげられることが判明した。これは、速さに関するパラメーターの値で、速さに異常がある場合走行不能になると考えられる。

次に以下に中間事象「コースアウト」の分析ツリーを示す。

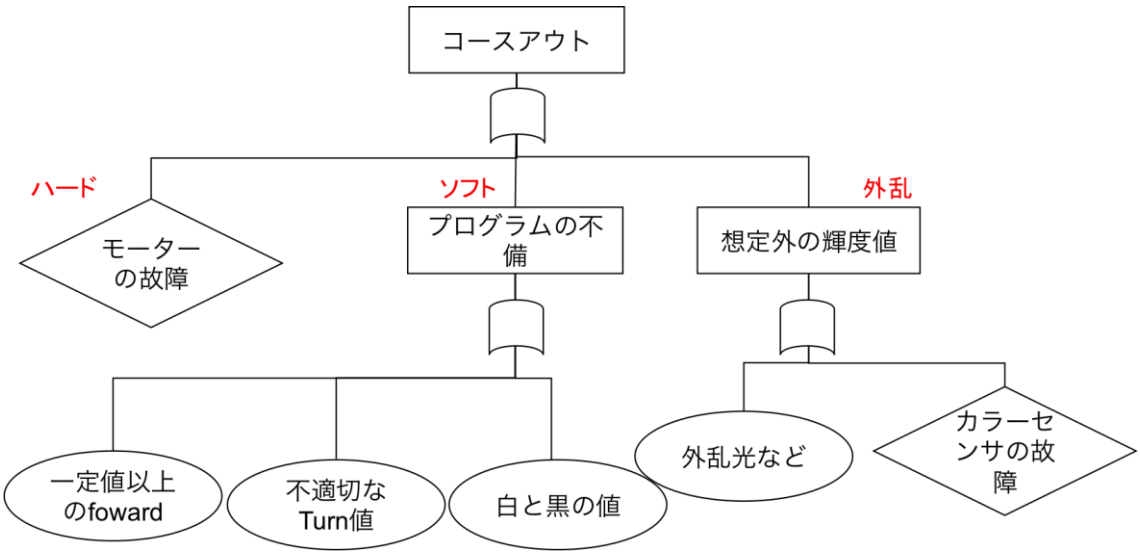

図 7　中間事象「コースアウト」の FTA 分析

結果として、プログラム上の「一定値以上の foward」「不適切な turn 値」「白と黒の値」と、外乱の要因「外乱光など」が原因としてあげられることが判明した。これは、forward が速さのパラメーター、turn が曲がる時の強さのパラメーターであり、「コースアウト」に関してはこの二つのパラメーターと外乱光が要因とあげられる。

次に以下に中間事象「ゲートに接触」の分析ツリーを示す。

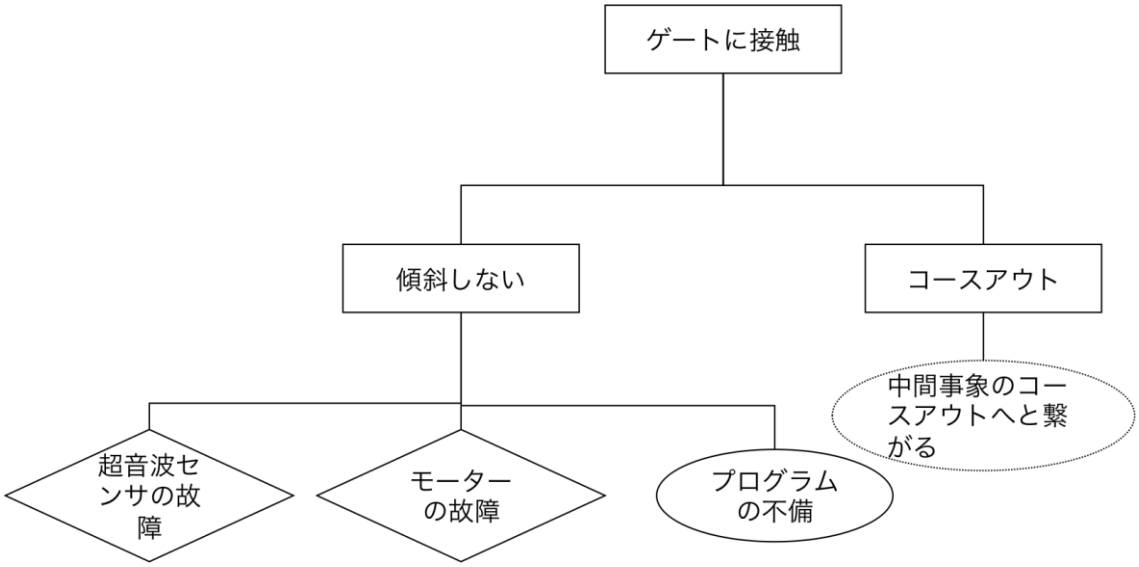

図 8　中間事象「ゲートに接触」の FTA 分析

結果として、プログラムに不備があることが原因としてあげられることが判明した。作成は設計の段階で行ったため、パラメーターの部分まで分析することができなかったが、プログラムに不備があると傾斜しないということが考えられる。

3.3　分析結果の比較

STAMP と FTA の分析結果は以下の表のようになった。

表 9　STAMP と FTA の分析結果の比較

	ハード	ソフト	他
FTA	・ジャイロセンサ ・カラーセンサ ・超音波センサ ・モーター ・EV3	・速度の設定 ・曲がる時の強さ ・輝度値の設定	・外乱光
STAMP	・ジャイロセンサ ・カラーセンサ ・超音波センサ ・モーター ・EV3 ・ケーブル	・プログラム全体 ・待機時間など	・外乱光

ハード面に関しては、STAMP 分析では繋がり（ケーブル）に着目することができた。また、ソフト面では、FTA は関数やパラメーターレベルまでの分析となったが、STAMP では「プログラム全体」レベルの分析となった。

次に、今回の比較結果をもとに STAMP と FTA 自体の比較を行った。その結果、以下の表の通りになった。

表 10　STAMP と FTA の比較

	モデル	着眼点	粒度	網羅(MECE)性
FTA	木構造 （ツリー）	コンポーネント単体	細かい	確信が持てない場合がある
STAMP	コントロールストラクチャ（ネットワーク）	コンポーネントとコンポーネント間の流れ	大まか	コントロールストラクチャが描ければ、その範囲における網羅性は定義できる

第 4 章　結論

ET ロボコンという実際に参加した事例を STAMP、FTA で施行することによって、より詳細な分析結果が得られた。また、結果をもとに比較を行ったことにより STAMP の特徴や有用性などが確認できた。

しかし、STAMP に関しては、何故こういう結果になったのかという二重の分析や、正しいコントロールストラクチャが描けているかという正確性など課題が残るものとなった。

まだ始まったばかりである STAMP の研究の一つの事例となれば幸いである。

文献

[1] 一般財団法人機械振興協会　故障の木解析　FTA の概要
http://www.jspmi.or.jp/system/l_cont.php?ctid=130403&rid=831(2017/02/15)
[2] FTA 解析　事例解説
http://www.geocities.jp/takaro_u/fta.html(2017/02/15)
[3]情報処理推進機構 「はじめての STAMP/STPA」(2016/04)

謝辞

本研究にあたり、研究の遂行及び、本論文のご指導いただきました松野裕准教授に心より感謝いたします。