JFP
metaphor Lab.

# 要求論
# 要求分析、SLPの使い方、
# 国際標準規格の場合

2014年7月28日
株式会社ジェーエフピー

目次

**要求論：要求分析、SLPの使い方、国際標準規格の場合**
２つの例題を用いて、要求仕様の要求分析の方法と、併せてSLPの使い方を述べます。

これらの議論を踏まえ、機能安全ISO 26262の要求仕様に関して、問題を提起します。

**1 結論**
ソフトウェア要求仕様の曖昧さ等の不備により、開発の問題を生じさせることはしばしば見られることです。
それゆえ、要求仕様書を明確に正しくということが叫ばれます。

本論では、そのような要望に対して十分応えていると思われる、よくできた要求仕様を例にとって、
よくできた要求仕様であっても問題が生じることがあることを表し、その対策を提示します。

本項の結論を先に言えば、以下となります。

（1）「形式」と「意味」
要求仕様の不備の問題は「形式」と「意味」という概念によりよく説明ができます。
要求仕様書に関しては、仕様内容が分からないとして、仕様の受け手は仕様の出し手に非難されたり、
また、その逆であったりする場合がしばしばあります。
しかし、このようなあつれきを生まないために、仕様内容、すなわち「意味」が分からなくても、仕様を記述する「形式」を整えることで、「意味」に迫る方法があります。
本稿ではこのことを示します。

また、要求仕様書がよく出来ていても、
後続の工程に要求項目がトレース（追跡）されていなくては、折角の要求仕様書も十分にいかされているとはいえません。
ただし、次のことがいえます。

（2）要求トレーサビリティ
要求項目のトレーサビリティ（追跡性）を各工程の成果物に正確に反映させようとするなら、
まずは要求項目をよく分析する必要があります。
要求項目の条件に漏れがあったまま、その要求項目を追跡対象にしてしまうと、
後続の設計文書やプログラムコード（実装）領域に、要求項目の反映されていない個所が生じます。
そのような事例と、事例をやや一般化する議論を述べます。

以上、本稿では、要求仕様は、要求項目の「形式」と「意味」を整えることが、
仕様内容の確定と追跡性の２つの点から重要であることを述べます。

（3）「要求仕様の内容はどのレベルまで書くべきか」という課題
以上の議論、特にも「形式」と「意味」のことを踏まえた上で、
よく言われる「要求仕様の内容はどのレベルまで書くべきか」という課題について議論します。
そのために、要求仕様を厳密に書くべきであるという立場の、
国際標準規格「機能安全」の１つであるISO 26262を取り上げます。
この規格における要求仕様のレベルに関する考え方を紹介し、かつその要求仕様の事例を２つ取り上げ、
・事例の１つは設計や実装まで書けるレベルまで「意味」が書かれていないこと
・事例のもう１つは記述が誤解を生むような「形式」化がなされていないのではないかということ
の２点を問題提起します。

国際標準規格の要求仕様を記述するにあたり、SLPの「形式」と「意味」の考え方を参考にされることを期待するものです。

## 2 「形式」と「意味」

本項では一見かなりよくできた要求仕様書を例に、要求分析を行い、要求の表現がまだ不十分であることを示します。
不備の説明を「形式」と「意味」という用語を用いて行います。

よくできた要求仕様書の例として以下を取り上げます。
これを例題１と呼びます。

《例題１》

---

仕様書例　　（“SCADE”のテキストより）

【クルーズ制御仕様】

・ クルーズオン制御

1. クルーズ制御状態がオフで，ONボタン又はResumeボタンが押され，車速が制御可能範囲内で，
   アクセルペダルとブレーキペダルが踏まれていない時
2. クルーズ制御状態がスタンバイで，車速が制御可能範囲内で，アクセルペダルが踏まれていないとき

・ クルーズオフ制御

3. 初期状態は必ずオフである
4. ブレーキペダルが踏まれるか，OFFボタンが押された時

・ クルーズスタンバイ制御

5. クルーズ制御状態がオフで，ONボタン又はResumeボタンが押され，車速が制御可能範囲内でない又は，
   アクセルペダルが踏まれて，ブレーキペダルが踏まれていない時
6. クルーズ制御状態がオンで，車速が制御可能範囲内でない又は，アクセルペダルが踏まれている時

---

《注》
この事例はテキスト『SCADE Drive(TM)5.1定期講習会（入門編）』のp.25にあります。
本論では要求項目としては1～6までを取り上げていますが、そのテキストには更に項目が書かれています。
本論ではこの6項目とした理由は、対象範囲がまとまっているからです。

参考までに、そのすべてを記すと以下となります。

---

・ クルーズ目標車速の算出

7. ONボタン又は SETボタンが押された時に,車速がクルーズ制御可能範囲内にある場合は,
   現在の車速をクルーズ目標車速とする．もし，目標車速が設定可能範囲の上限以上の場合は,
   上限値に，下限値以下の場合は，下限値に設定する．
8. UPボタンが押された時は，現在の目標車速にIncだけ（増加分)足し,
   DOWNボタンが押された時は，現在の目標車速から Decだけ(減少分）引く．
9. クルーズ目標車速の算出は，クルーズ制御がオン又はスタンバイの時だけ実行する．

・ 目標スロットル開度の算出

10. 制御状態がオンの時は，クルーズ目標車速と現車速の差をフィードバック制御（PI制御）に
    よって目標スロットル開度を計算し,制御状態がオン以外の時は,アクセル開度を目標スロッ
    トル開度に設定する．
11. PI制御の積分器は，クルーズ制御状態がオンになった時にリセットする．
    （立ち上がりエッジでリセットをかける．）
12. 積分器には，リミッターを設定する．
13. PI制御の演算は，クルーズ制御状態がオンの時だけ実行する．

---

そしてこのテキストは、要求仕様の検討のためではなく、
SCADEを活用した設計を行うための例題となっています。

「この仕様を基に制御システム全体の設計を考えます.
再利用性，ライブラリ化を考慮し，制御モデルをサブシステムとして設計します.
特に，制御の前提条件となるような仕様は，別サブシステムとして設計することを推奨します.
これによって，構造的な設計ができ，保守性，トレース性が向上します.
また，同時に各サブシステムの入力，出力，関係についても考えます.」

SLPでは、要求仕様に加えて技術仕様（実現するための技術的な仕様）を付加しなければならないと
主張していますが、おなじことがSCADEの例題でも記述されています。
すなわち、テキストでは上の仕様に基づいて実現のための設計作業がなされています。

SLPの主張とSCADEのそれとの違いは、SLPでは技術仕様に行く前に、もう少し要求仕様を分析しよう、というものです。
何故なら、近年、ソフトウェア規模が想像以上に大きくなったので、
設計に行く前に不明な要求を明らかにしておかないと、
設計の際に、要求策定プロセスで行うべきことを設計者が担わなければならなくなるからです。
規模が小さな時代は、要求策定プロセスの負担はさほど重荷ではありませんでした。
しかしいまや、システムの規模の大きさを前に、要求仕様をより明確にしてかかるべき、というのが SLP の主張です。

### 2.1 《要求分析０》「字面」を単純に分析する

例題１をここでは単純に「字面」通りに分析しました。

通常は、要求分析は、要求の全体の意味を理解し、内容の分類や整理をし、その過不足を検討します。
しかしここでは意味ではなく、文の形式を対象とします。
これを「字面」と呼ぶことにします。

字面通りとは、SLPの構文に合わせて、文を主語（や目的語）とその述語に分析し、単位文を作って行き、
かつ単位文を「ならば」の関係で結ぶことです。
SLPでは主語や目的語をメンバー名と呼び、述語をメンバーの状態を指す状態名と呼んでいます。
「ならば」の前後の単位文は前件と後件と呼ばれるものです。
要求仕様書の世界では、事実の世界を記述しますから、
前件と後件は時間的前後関係を持った**因果的な関係**を記述しています。

以下の論理記述欄では、例題１の要求を順にSLPの文法で記述しています。
これを「字面」と呼んでいます。

### 2.1.1 要求項目１

**・ クルーズオン制御**

1. クルーズ制御状態がオフで，ONボタン又はResumeボタンが押され，車速が制御可能範囲内で，アクセルペダルとブレーキペダルが踏まれていない時

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
            else
            endif
        else
        endif
    else
    endif
else
endif
```

#### 2.1.1.1 複数のメンバー名に関する注意

SLP構文では、『主語述語論理モデル』を標榜し、
１主語＋１述語を文の基本である単位文としています。
しかし先の例（要求項目１の論理記述欄）では、

```
if <ONボタン又はResumeボタン>が{押され}
```

とあり、
メンバー名欄に２つの主語（メンバー）が存在しています。
本来これは、

```
if <ONボタン>が{押され}

else
  if <Resumeボタン>が{押され}
```

と書かれるべきものです。
現在のSLPには、このような複数の主語をまとめて書いた
簡略的な表記を、自動分解し、単位文とする機能はありません。
現在ではまとめられた全体が主語となります。
主語を複数に分ける記号の導入も必要で、
却って煩雑になると判断したからですが、今後の課題とします。
このことは述語に関しても当てはまります。

ただし、「又は」が「と」の場合、

```
if <ONボタンとResumeボタン>が{押され}
```

の中の「と」を「同時に押された場合」などと解釈する可能性も出てきます。
このような場合には、

if <ONボタン>が{押され}
  if <Resumeボタン>が{押され}

と、単純にifを２行に並べるという訳には行きません。
「と」などの関係詞が何処まで意味を含むか、自然言語に関しては、
形式化と同時に意味の領域を明確にする作業も必要です。
因みに「同時に押された場合」は次のような記述が可能です。
①　if <ONボタンとResumeボタン>が{同時に押され}
②　if <ONボタン>が{押され}
　　if <Resumeボタン>が{ONボタンと同時に押され}

このような様々な解釈があることから、
形式化の過程で、書き手が意味を正確にして行くことが求められます。

### 2.1.1.2 述語の様相などの注意

if <ONボタン又はResumeボタン>が{押され}

は、述語が「押され」と中途半端になっています。
「押された」にすべきと思いますが、
SLPは文字列が同じであれば、用語が同一であるとしますから、
書き手が「押され」に統一するのであれば、それでも構いません。

ただし、誰もが普通に用いる用語で単位文を作るのが
SLPの規範ですから（規則ではありません）、
「押された」とか「押される」などのように末尾の文字まで記述すべきです。
しかし、末尾まで記述したとしても、この例のように、
「た」とか「る」の時制や様相に関する問題が残りますから、
時制や様相の違う場合には、異なる述語として正確に書かないと行けません。

最後に、用語の類似性のチェックなどを行い、
用語を安定させることもできます。

SLPでは様相に関する構文は作っていません。
したがって検査の対象外です。
注意を喚起する意味で、
「となる」（現在の変化）
「である」（現在の状態）
「であった」（過去の状態）
などを書き手が登録することができるようにしています。

### 2.1.1.3 長い主語をどう書くか

主語は文書という集合において、文をつくるための必須の要素です。
システムはメンバーの振る舞いと考えれば、主語はメンバーにあたります。
主語が述語を持てば意味を作ることになり、メンバーがどういう状態であるかが表現されることになります。

しかしこのメンバーの特定が大変に長い場合があります。

「じゅげむ、じゅげむ」がその例です。
臆さず「じゅげむ」方式で書けば良いと思います。
SLPは近く、長く書いた用語を他の用語で言い換える定義機能をリリースする計画です。

メンバー名の形容詞など、例えば「赤い」とか「大きい」とかの形容詞、
また「楽しく」、「厳密に回転した」などの用語をAとすれば、
これらによって修飾されるメンバーMは、
「A１∧A２∧A３∧、、、、∧AnであるM」というような構造を持ちます。
これを文で表現すると
「”MはA1であり、かつ、MはA２であり、かつ、、、、、MはAnである”というようなMである」
というように表現できます。
すなわち、M（主語）はｎ個の属性（述語）を持ちます。

SLPは単位文から構成されていると主張しますが、
SLPの世界は、
主語の個々の属性（述語）をその場に応じて（条件に応じて）選択する世界を描いている、と言えます。

### 2.1.2 要求項目２

・ クルーズオン制御

2. クルーズ制御状態がスタンバイで，車速が制御可能範囲内で，アクセルペダルが踏まれていない時

```
if <クルーズ制御状態>が{スタンバイ}
    if <車速>が{制御可能範囲内}
        if <アクセルペダル>が{踏まれていない}
            Do<クルーズ制御>を{オン}せよ
        else
        endif
    else
    endif
else
endif
```

### 2.1.3 要求項目３

・ クルーズオフ制御

3. 初期状態は必ずオフである

```
if <当該のシステム>が{初期状態}
    Do<クルーズ制御>は{オフ}である
else
endif
```

### 2.1.4 要求項目４

・ クルーズオフ制御

4. ブレーキペダルが踏まれるか，OFFボタンが押された時

```
if <ブレーキペダル>が{踏まれる}
    Do<クルーズ制御>を{オフ}とせよ
else
    if <OFFボタン>が{押された}
        Do<クルーズ制御>を{オフ}とせよ
    else
    endif
endif
```

### 2.1.5 要求項目５

・ クルーズスタンバイ制御

5. クルーズ制御状態がオフで，ONボタン又はResumeボタンが押され，車速が制御可能範囲内でない又は、アクセルペダルが踏まれて，ブレーキペダルが踏まれていない時

#### 2.1.5.1 「または」の係る箇所

この文の解釈は論理記述欄に書いた通りですが、
この文の意味解釈をせず、字面を単純に分析するだけですと、実際には複数の表現が可能です。
「車速が制御可能範囲内でない又は」の「又は」のかかる箇所を下記のように３つに解釈することができます。
このような場合には、仕様の出し手に実際の意味を確認する必要があります。

解釈①(論理記述欄と同じ)

(クルーズ制御状態がオフで(文p)，ONボタン又はRtsumtボタンが押され(q)，
　(車速が制御可能範囲内でない(r))
又は
　(アクセルペダルが踏まれて(s)，ブレーキペダル踏まれていない(t)時))
記号化：p ∧ q ∧ ( r ∨ ( s ∧ t ) )

解釈②
(クルーズ制御状態がオフで，ONボタン又はRtsumtボタンが押され，車速が制御可能範囲内でない)
又は
(アクセルペダルが踏まれて，ブレーキペダル踏まれていない時)
記号化：( p ∧ q ∧ r ) ∨ ( s ∧ t )
解釈③
(クルーズ制御状態がオフで，ONボタン又はRtsumtボタンが押され，
　(車速が制御可能範囲内でない 又は アクセルペダルが踏まれて)
　ブレーキペダル踏まれていない時)
記号化：p ∧ q ∧ ( r ∨ s ) ∧ t

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押された}
        if <車速>が{¬制御可能範囲内}
            Do<クルーズ制御>を{スタンバイ}とせよ
        else
            if <アクセルペダル>が{踏まれている}
                if <ブレーキペダル>が{¬踏まれている}
                    Do<クルーズ制御>を{スタンバイ}とせよ
                else
                endif
            else
            endif
        endif
    else
    endif
else
endif
```

### 2.1.6 要求項目６

・ **クルーズスタンバイ制御**

6. クルーズ制御状態がオンで，車速が制御可能範囲内でない又は，アクセルペダルが踏まれている時

```
if <クルーズ制御状態>が{オン}
    if <車速>が{¬制御可能範囲内}
        Do<クルーズ制御>を{スタンバイ}とせよ
    else
        if <アクセルペダル>が{踏まれている}
            Do<クルーズ制御>を{スタンバイ}とせよ
        else
        endif
    endif
else
endif
```

### 2.2 《要求分析１》「形式」を整える

《要求分析０》で行われた要求分析を、
要求分析者は、この論理記述欄にコピーし、相互の関係を照合しました。
《要求分析０》での要求１～要求６をそのまま引き写し、
要求事項が例題１の何番目の項目に書かれているものかをコメント行に記述しました。

分析の過程を述べ、併せてSLPの使い方を説明します。

（１）「形式」化

SLPの方法を適用すると、例題１の或る要求項目の中には、
SLPのif-else-endif構文におけるelse（背反条件）のケースが記載されていないことを
明示できます。
しかし、分析者は、ある要求項目にelseのケースが記載されていなくても、
他の箇所の要求項目に書いているだろうと推測します。
そして、両者を突き合わせれば、elseのケースもハッキリするだろうと推測します。
**もしその通りであるとすれば、要求項目の全体は、過不足無く記述されていることになり、**
**そうでなければ、内容に漏れがあることになります。**
**この作業は、個々の文をSLP構文で「形式」化することにより、判明します。**

（２）「意味」を問う

「形式」化の過程で、仕様の内容（意味）が分からないところが出て来る場合があります。
論理記述欄の４行目は「アクセルペダルとブレーキペダル」の２つが一緒になって
１つのメンバー（主語）を構成していますが、
２つのうちのいずれかが動作（踏まれたり、踏まれていなかったり）した場合には、
クルーズ制御はどうすべきであろうかという疑問が生じます。
**これはクルーズ制御の仕様内容に関する疑問です。**
**「意味」の問題であり「形式」の問題ではありません。**
**このような場合には、出題者、**
**すなわち要求仕様の出し手（発注者）に確認するしか手だてはありません。**

（３）SLP構文のDo文が「である」の場合

初期状態の記述を行っている２８行目の表現は、対応する要求項目（項番３）の内容からして、
「○○を××とせよ」ではなく、「○○は××である」という表現にすべきでしょう。
しかし、SLP構文では、単独で状態を「である」と記述する文（Is文）はありません。

そこでDo文をIs文と読み替えて使っていただくことになります。
理由は、Do文で「（と）せよ」によって変化させられた状態は
「である」状態になるので、両者は同じこととSLPでは解釈しています。

（４）「字面」を超えて解釈する
また、状態名「初期状態」に、何が初期状態であるかを明示するために、
「当該のシステム」という主語（メンバー名）を割り当てています。
SLPでは状態名は、その対象名（メンバー名）を持つと前提しているからです。

ところで、「当該のシステム」という表現は仕様記述には存在しません。
そこで、この解釈は「「字面」を単純に分析する」ものではありません。
主語を埋めるために、仕様を解釈し、
「初期状態」という述語を持ちうる主語は「当該のシステム」としました。
この例は、「字面」分析においては、存在しない言葉を補完する場合には、
行間にある意味を推し量らなければならないことを物語ります。
「字面」だけでは仕様を正しく分析できないことを示します。

この例でいえば、一般にシステムが起動していないときには
どんな制御もオフの状態であると考える訳ですが、この解釈に誤りがないのか、
念のため**これは要求の出し手に確認すべきです。**
システムの起動時の状態は最近では様々なレベルがあるからです。

（５）用語の統一の仕方
用語、特に状態名は下記に記しますように、表現が微妙に異なります。
SLPでは用語の異同を文字列で判断しますから、複数の表現を同一の意味で使用する場合には、
用語を同じにしておく必要があります。
（なお、用語の同一性を段階的に強化して行く計画をSLPは持っています。）
（ａ）「踏まれていない」（４行など）は、「踏まれている」の否定形を考え、
「￢踏まれている」にしてもよいと思います。
（あるいは「踏まれている」を「￢踏まれていない」にしてもよいでしょう。）
すると、そのメンバー名は「踏まれている」という状態名（と、その否定）しか
持たないことがはっきりするからです。
SLPには用語の曖昧さの検査や、文の矛盾の検査機能がありますが、
コトバを統一することにより、読み手の誤解のリスクを低減することができます。
（ｂ）「押され」という状態名も「押されている」とすることにより、状態名が明確になります。
（ｃ）「制御可能範囲内」という状態名も「である」と付記するとより明示的になります。
存在しているという状態が判明になるからです。
もっとも否定形は「￢制御可能範囲内」と書いた方が簡潔であるともいえます。
「オン」や「オフ」（「ON」や「OFF」）も同様です。
もっとも「オン」や「オフ」などは制御系の常套句でもあるので、
「である」という存在の状態を表す表現は付加しなくてもいいでしょう。

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のifの4つの重なりは、例題１の項番１のケースである。
            else
            endif
        else
        endif
    else
    endif
else
    if <クルーズ制御状態>が{スタンバイ}
        if <車速>が{制御可能範囲内}
            if <アクセルペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上の13行目のelseとifの3つの重なりは、例題１の項番２のケースである。
            else
            endif
        else
        endif
    else
    endif
endif

if <当該のシステム>が{初期状態}
    Do<クルーズ制御>は{オフ}である
    (:このケースは、例題１の項番３のケースである。
else
endif
if <ブレーキペダル>が{踏まれる}
    Do<クルーズ制御>を{オフ}とせよ
    (:このケースは、例題１の項番４の前半「ブレーキペダルが踏まれる」のケースである。
else
    if <OFFボタン>が{押された}
        Do<クルーズ制御>を{オフ}とせよ
        (:このケースは、例題１の項番４の後半「OFFボタンが押された時」のケースである。
    else
    endif
endif

if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押された}
        if <車速>が{¬制御可能範囲内}
            Do<クルーズ制御>を{スタンバイ}とせよ
            (:このケースは、例題１の項番５の前半「、、、又は」の前までのケースである。
        else
```

```
            if <アクセルペダル>が{踏まれている}
                if <ブレーキペダル>が{¬踏まれている}
                    Do<クルーズ制御>を{スタンバイ}とせよ
                    (:このケースは、例題１の項番５の後半「又は」以降のケースである。
                else
                endif
            else
            endif
        endif
    else
    endif
else
    if <クルーズ制御状態>が{オン}
        if <車速>が{¬制御可能範囲内}
            Do<クルーズ制御>を{スタンバイ}とせよ
            (:このケースは、例題１の項番６の前半「、、、又は」の前までのケースである。
        else
            if <アクセルペダル>が{踏まれている}
                Do<クルーズ制御>を{スタンバイ}とせよ
                (:このケースは、例題１の項番６の後半「又は」以降のケースである。
            else
            endif
        endif
    else
    endif
endif
```

2.2.1 《要求分析１－１》字面で分析されたもの同士を照合する

字面で分析されたもの同士を照合する理由は、
else条件での記述が抜けた箇所が、他の要求項目で補われているかもしれないからです。

実際、アクセルやブレーキのペダルが「踏まれた」場合の記述が、
《要求分析１》（前掲）の論理記述欄の３２行目にあります。
（下記の論理記述欄ではなく《要求分析１》の３２行目をご覧ください。以下の要求分析１－２、－３における「行目」も《要求分析１》の論理記述欄に関するものです。）
４９、５０行目、また６６行目にもあります。
（行数の確認のために、論理記述欄の印刷を行い、以下をお読みになることをおすすめします。）

これら（３２，４９、５０、６６行目）を
《要求分析１》の７行目のelseの場合として扱っていいかもしれません。
つまり、この７行目のelseは「アクセルやブレーキのペダル」が「踏まれていない」の否定であるので、
「踏まれた」場合となるからです。
しかし、そうではないという考えも成り立ちます。
つまり、この「アクセルやブレーキのペダル」が「踏まれた」場合には、その上位の条件が存在します。
上位の条件は「車速が制御範囲内である」場合という条件があり、
さらにその上に「ONボタン又はResumeボタンが押された」という条件もあります。

《要求分析１》の７行目はこのような条件が重なった上で「踏まれた」場合なのですが、

他方《要求分析１》の３２行目は、いきなり無条件で「ブレーキペダルは踏まれる」場合とあります。

さて、この場合にはどう考えたらよいのでしょう。
無条件にブレーキペダルが踏まれたときは何であれ（他の条件に関わりなく）、
「クルーズ制御をオフとせよ」ということでしょうか。

これは条件が重なった場合の「意味」の問題です。
一方は条件が多重であり、他方は条件が１つです。
ですから、形式は異なっています。
しかし「意味」は同じかもしれません。
このことの**確認**は、やはり要求の出し手に行う必要があります。
内容（「意味」）の分からない人が、内容を問う場合には、
このような「形式」の違いを明確にしながら問うてはどうでしょうか。
ドメイン知識が不足ゆえに問う訳ですが、その際「形式」の違いを明示して問う方が、
エンジニアとしての立場を表明できるのではないでしょうか。
教える側に安心感を与えると思われます。

以下では、次の議論に進むために、
形式は同じであっても同じ「意味」が当該条件の箇所に入るとして分析を続けます。
そう考えることにより、下記《要求分析１－１》の論理記述欄の８、９行には
「ブレーキペダルが踏まれる」場合を挿入しました。
しかし、１５行目以降に、ある考えが生まれした。（コメント行参照）
そこで別途、《要求分析１－２》を記述することとしました。

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のifの4つの重なりは、例題１の項番１のケースである。
            else
                if <ブレーキペダル>が{踏まれる}
                    Do nothing
                    (:ここでは、例題１の項番４の前半「ブレーキペダルが踏まれるか」のケースを追加した。
                    (:要求分析１の３３行では「Do<クルーズ制御>を{オフ}とせよ」とあるが、
                    (:もともとこのケースでは、「<クルーズ制御>が{オフ}]であるため、再度「オフ」にする(:必要はない。
                    (:よって「Do nothing」（何もしない。次行に移る）となる。
                    (:
                    (:ところで、ここ《要求分析１-１》の条件else（７行目）以下では、
                    (:<ブレーキペダル>と<アクセルペダル>の両者ともが{踏まれる}場合も存在する。
                    (:このような目で《要求分析１》の４９行目（if <アクセルペダル>が{踏まれている}）を(:みると、
                    (:この上の行の４８行目のelse（＝<車速>が{制御可能範囲内}）は、
                    (:《要求分析１-１》の３行目のif（＝<車速>が{制御可能範囲内}）と同じ意味の条件を指(:しており、
                    (:かつ、その上位の条件も同一であるので、
                    (:これらの条件（《要求分析１》の４８行目のelseと《要求分析１－１》の３行目）の
                    (:下位の行の構文は同一としてよいといえる。
                    (:そこで４９行目以降を、この７行目のelseに追加して書き改める。
                    (:書き改めたものを次の《要求分析１－２》に反映させる。
                else
                endif
            endif
        else
        endif
    else
    endif
else
    if <クルーズ制御状態>が{スタンバイ}
        if <車速>が{制御可能範囲内}
            if <アクセルペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のelseとifの3つの重なりは、例題１の項番２のケースである。
            else
            endif
        else
        endif
    else
    endif
endif
```

### 2.2.2 《要求分析１－２》要求分析１－１を更に分析する

《要求分析１－１》での１５行目以降の考えを反映し、論理記述を改めました。
その結果、《要求分析１－２》の論理記述欄の８行目以降が書き改められました。

ところで《要求分析１－２》の２６行目のelseは<車速>が{￢制御可能範囲内}のケースであり、
これと同じケースが《要求分析１》の４５行目以降に書かれています。
そこで、これを反映した要求分析を別途、《要求分析１－３》として記述します。

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のifの4つの重なりは、例題１の項番１のケースである。
            else
                if <アクセルペダル>が{踏まれている}
                    if <ブレーキペダル>が{￢踏まれている}
                        Do<クルーズ制御>を{スタンバイ}とせよ
                        (:このケースは、例題１の項番５の後半「又は」以降のケースである。
                    else
                        Do nothing
                        (: <ブレーキペダル>が{踏まれる}ケースである。
                        (:ここでは、例題１の項番４の前半「ブレーキペダルが踏まれるか」のケースを
                        追加(:した。
                        (:「踏まれる」と「踏まれている」の時制の微妙な違いもある。
                        (:SLPではこのような違いは形式化していない。
                        (:重要な違いが発生する場合には表現で補うこととする、としている。
                    endif
                else
                    Do nothing
                    (: ここでは、上の場合分けからして、
                    (:<アクセルペダル>が{￢踏まれる}かつ<ブレーキペダル>が{踏まれる}ケースとな
る。
                endif
            endif
        else
        endif
    else
    endif
else
    if <クルーズ制御状態>が{スタンバイ}
        if <車速>が{制御可能範囲内}
            if <アクセルペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のelseとifの3つの重なりは、例題１の項番２のケースである。
            else
            endif
```

```
        else
        endif
    else
    endif
endif
```

### 2.2.3 《要求分析１－３》要求分析１－２を更に分析する

この《要求分析１－３》では、《要求分析１－２》の２６行目のelse以降に、
《要求分析１》の４６行目を挿入しています。
《要求分析１－２》の２６行目のelseは<車速>が{┬制御可能範囲内}の場合であり、
これは《要求分析１》の４５行目と同じ条件だからです。

次に《要求分析１－３》の３２行目に移りますが、
ここでのelseは、<ONボタン又はResumeボタン>が{押されていない}で、
かつ、その上の条件が<クルーズ制御状態>が{オフ}の場合です。
しかしこの場合にどのようにすべきかは、例題１の中には書かれていません。

これは「車速」のことや、「ONボタン又はResumeボタン」のことが分かっていれば、どのようにすべきかは分かるでしょう。
**恐らくは、<ONボタン又はResumeボタン>が{押されていない}ので、<クルーズ制御状態>は{オフ}をそのまま維持する、ということと思われますが、**
**それらの「意味」が分からず釈然としないところです。**
コメントに”Pending”と記し、後で要求の出し手に確認すべきところです。

さらに《要求分析１－３》の４１行目のelseは、<アクセルペダル>が{┬踏まれていない}、すなわち{踏まれた}時です。
このelseでは、その上の条件は、<車速>が{制御可能範囲内}の時であり、
さらにその上の<クルーズ制御状態>が{スタンバイ}であるという条件が重なっています。

他方、《要求分析１》の３２行目では、その上位に何の条件もなしに
if <ブレーキペダル>が{踏まれる}とあります。
そして、この場合には、
　Do<クルーズ制御>を{オフ}とせよ、とあります。
そこで、ここでは例題１の項番４の前半「ブレーキペダルが踏まれる」のケースを当てはめてみます。

しかし、このように条件のないケースでの制御を、上位に制約条件がある場合に対して、
単純に当てはめてよいのか、繰り返しになるますが、
これも**確認**の必要があります。そこでPendingと置きます。

次に、この《要求分析１－３》の論理記述欄の４６行目は
<車速>が{┬制御可能範囲内}のケースであり、
その上位が<クルーズ制御状態>が{スタンバイ}の時です。
この条件は、《要求分析１》の６１行目以降に、

```
    if <クルーズ制御状態>が{オン}
        if <車速>が{┬制御可能範囲内}
            Do<クルーズ制御>を{スタンバイ}とせよ
            (:このケースは、例題１の項番６の前半「、、、又は」の前までのケースである。
```

がとあり、ここでは、<車速>が{┬制御可能範囲内}という条件の時は、

〈クルーズ制御状態〉の{オン}を{スタンバイ}に変える、とあるからです。
すなわち、ここでは、〈車速〉が{┬制御可能範囲内}の場合には、
この場合の上位の条件が何であれ、〈クルーズ制御〉は{スタンバイ}とすべきものと解釈されるからです。
したがって《要求分析１－３》の４６行目が、Do　Nothing（なにもしない）となる理由は、
〈クルーズ制御〉を{スタンバイ}の時に、
〈車速〉が{┬制御可能範囲内}であるから、
あらためて〈クルーズ制御〉を{スタンバイ}とする必要は無いからです。
しかし、確認の必要があります。

次に《要求分析１－３》の５０行目のelseをどう考えるべきか。
〈クルーズ制御状態〉というメンバー名は、オフとスタンバイとオンの状態を持つようです。
したがって、この５０行目に至る前にはすでに、オフとスタンバイを使用済みであるので、
ここでは〈クルーズ制御状態〉が｛オン｝の場合となる、と考えられます。
しかし、そうでは無いかもしれません。
少なくとも仕様では状態を３つに限定するとは言っていません（書かれていません）。
そこで、ここでもif〈クルーズ制御状態〉が｛オン｝と立てます。
すると例題１の項番４のケースの反対の条件の場合が当てはまるように思われます。
項番４では、ブレーキペダルが踏まれず、OFFボタンも押されなかった場合にはどうするかの
記述がありませんが、
この《要求分析１－３》の５１行では〈クルーズ制御状態〉が｛オン｝という条件が位置するので
５５行目の〈ブレーキペダル〉が{踏まれず}、
〈OFFボタン〉も{押されなかった}場合（５９行目）には、
〈クルーズ制御状態〉は｛オン｝のままであろうと推測されます。
そこで、Do　Nothingと記します。
（要、確認です。）

さて、《要求分析１－３》の６４行目のelseはどう考えるべきでしょうか。
このelseのメンバー名は、〈クルーズ制御状態〉です。
先にこのメンバー名の状態名はオフとスタンバイとオンの３つしか持たないのではないか、
と述べました。
そして《要求分析１》においても、{初期状態}のメンバー名も〈当該のシステム〉としています。
一般に、{初期状態}はシステム全体に関する属性だからです。

しかし、{初期状態}のメンバー名は〈クルーズ制御状態〉としてもよいのかもしれません。
すなわち、〈クルーズ制御状態〉が{初期状態}であるとき、と考えることができます。
そう考えて論理を組んでみます。ただし、要、確認で、Penndingと記します。

同じように例題仕様１の項番３「・クルーズオフ制御 3. 初期状態は必ずオフである」の
初期状態に関する記述に関しても、「初期状態の場合には」と考え、if条件を立てます。
このif条件のelseの場合には、メンバー〈クルーズ制御状態〉の「オン」、「オフ」、「スタンバイ」、
そして「初期状態」の全状態名を記述したので、Do Nothing（「なにも機能はない」）とします。
Pendingとして、問う必要はありません。

**以上で例題１の要求分析は終了しました。**

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のifの4つの重なりは、例題１の項番１のケースである。
            else
                if <アクセルペダル>が{踏まれている}
                    if <ブレーキペダル>が{¬踏まれている}
                        Do<クルーズ制御>を{スタンバイ}とせよ
                        (:このケースは、例題１の項番５の後半「又は」以降のケースである。
                    else
                        Do nothing
                        (: <ブレーキペダル>が{踏まれる}ケースである。
                        (:ここでは、例題１の項番４の前半「ブレーキペダルが踏まれるか」のケースを追
                        加(:した。
                        (:「踏まれる」と「踏まれている」の時制の微妙な違いもある。
                        (:SLPではこのような違いは形式化していない。
                        (:重要な違いが発生する場合には表現で補うこととする、としている。
                    endif
                else
                    Do nothing
                    (: ここでは、上の場合分けからして、
                    (:<アクセルペダル>が{¬踏まれる}かつ<ブレーキペダル>が{踏まれる}ケースとなる。
                endif
            endif
        else
            (:<車速>が{¬制御可能範囲内}のケースである。
            (:このケースには分析１の４６行目に書かれているものを挿入した。
            Do<クルーズ制御>を{スタンバイ}とせよ
            (:このケースは、例題１の項番５の前半「、、、又は」の前までのケースである。
        endif
    else
        (:Pending
    endif
else
    if <クルーズ制御状態>が{スタンバイ}
        if <車速>が{制御可能範囲内}
            if <アクセルペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のelseとifの3つの重なりは、例題１の項番２のケースである。
            else
                Do<クルーズ制御>を{オフ}とせよ
                (:このケースは、例題１の項番４の前半「ブレーキペダルが踏まれるか」のケースである。
                (:Pending
            endif
        else
            Do nothing
```

```
            (:Pending
        endif
    else
        if <クルーズ制御状態>が{オン}
            if <ブレーキペダル>が{踏まれる}
                Do<クルーズ制御>を{オフ}とせよ
                (:このケースは、例題１の項番４の前半「ブレーキペダルが踏まれる」のケースである。
            else
                if <OFFボタン>が{押された}
                    Do<クルーズ制御>を{オフ}とせよ
                    (:このケースは、例題１の項番４の後半「OFFボタンが押された時」のケースである。
                else
                    (:Pending
                    Do nothing
                endif
            endif
        else
            if <クルーズ制御状態>が{初期状態}
                Do<クルーズ制御>は{オフ}である
                (:このケースは、例題１の項番３のケースである。
            (:Pending
            else
                Do nothing
            endif
        endif
    endif
endif
```

## 2.3 《要求分析２》「意味」を深める

要求分析においては、「意味」を「深める」必要があります。
「深める」とは要求仕様に書かれていないことであっても、
公知の知識（事実）を手がかりに、要求の知識を補うことをいいます。
物理や科学的知識、法律等に関しては、特定のドメイン知識に依らず、公知の知識と考えられます。
このような知識が要求に関係しそうな場合には、要求分析の中に、折り込む必要があります。

次の例題２で見るように「（水の）沸騰」という場合に、
沸騰の状態を温度で具体的な数字に置き換えることも要求分析ではごく普通の方法です。
これは要求の意味を外延的（具体的に）に示すことといえます。
公知の知識を用いたり、外延的な表現を用いたりすることを、
ここでは「意味」を深める、と比喩的にいうことにします。

例題２は公知の知識や外延化で要求の意味を補う要求仕様の例です。

《例題２》　出典：JASA安全性向上委員会2013年SSQ-WG より（制作：中村洋氏、(株)レンタコーチ）

---

企画要求仕様

PR01 ： 電気ポットの容量は2 リットルとし、10度C から沸騰するまでの時間は、15 分以内とする。
PR02 ： 電源コンセントをつなぐと、直ちに作動し、ヒータで加熱を始め、沸騰に達したら、90 度C に保温する。
PR03 ： 再沸騰ボタンが押されたら、再沸騰を始める。
PR04 ： 水が加えられ、温度が低下したら、再沸騰を始める。
PR05 ： 保温中であれば、お湯を注ぐことができる。

---

この例題２の要求に関して、まずは次のように分析作業を行います。
ここでは「字面」方式ではなく、「意味」を少し考えてみます。
といっても、５つの要求項目を眺めて「保温」という用語が２箇所にあるので、
これを同じ要求項目にまとめてみる、というだけのことです。
PR02とPR05に「保温」がありますので、これらを１つにし、要求項目の記号を「Req」とします。
すると、要求項目は以下となります。
その後は「字面」分析を行います。

---

企画要求仕様

Req 01 ： 電気ポットの容量は2 リットルとし、10度C から沸騰するまでの時間は、15 分以内とする。
Req 02 ： 電源コンセントをつなぐと、直ちに作動し、ヒータで加熱を始め、沸騰に達したら、90度C に保温する。
　　保温中であれば、お湯を注ぐことができる。
Req 03 ： 再沸騰ボタンが押されたら、再沸騰を始める。
Req 04 ： 水が加えられ、温度が低下したら、再沸騰を始める。

---

### 2.3.1 Req 01

Req 01 ： 電気ポットの容量は2 リットルとし、10度C から沸騰するまでの時間は、15 分以内とする。

```
(:要求番号：Req 01
Do<電気ポットの容量>を{2リットル}とせよ
Do<10度Cから沸騰するまでの時間>を{15分}とせよ
(:【意味の不足】10℃以外の場合は？　15分以内にする方法は？　これらが書かれていない。
(:ここでは【意味の不足】という指摘を、10℃等の説明を個々に補ってもらうことではなく、
(:要求を「２リットルの水を湧かすのに15分を要する加熱器の性能」と解釈することで補った。
(:この解釈は要求を外延的（個々具体的に）ではなく、内包的に（一般化して）解釈し、
(:具体的な事柄は内包的な解釈から引き出して対応せよ、という立場である。
(:熱量や電力の物理法則（普遍的知識＝内包的知識）が機器の制御に応用されるべきである。
(:また加熱器（ヒータ）の性能が不合理なものでないかも当初に検討されるべきである。
(:下記では、加熱器が通常の環境に設置されているという前提で、
(:加熱器の動作が正常であるか否かに視点をおいて、要求を分析した。

if <水量>が{2リットル}である
    if <水温>が{10度C}である
        if <水の加熱開始から沸騰までの時間>が{15分以内}であった
            Do<加熱器の性能>を{正常と判断}せよ
            (:ただし、沸騰までの時間が15分以内とあるが、「以内」がどこまでかの確認が必要である。
            (:12,3分ならOKだが、5分も「以内」であることには間違いがないから、
            (:そのような高機能（高価）な加熱器はこの要求仕様の対象外と思われるが。
        else
            (:沸騰までの経過時間が15分より多かった場合である。
            Do<加熱器の性能>を{異常と判断}せよ
            (:正常ではなく異常と判断された場合には、異常通知のシグナルを出すべきかもしれない。
            (:要、確認。
        endif
    else
        (:水温の物理学的知識からいえば、
        (:水温が10度C未満の場合には、沸騰までの時間は15分よりも多く掛かり、
        (:水温が10度C以上の場合には、沸騰までの時間は15分未満となる。
        (:このような知識を踏まえ、加熱器の性能を監視する機能が考えられる。要、確認。
        if <水温>が{10度C未満}
            if <水の加熱開始から沸騰までの時間>が{15分より多い}
                Do<加熱器の性能>を{正常と判断}せよ
                (:ただし、この時間をどこの設定するかは加熱器の性能による。
                (:沸騰までの時間が長く掛かりすぎたなら異常である。
            else
                Do<加熱器の性能>を{異常と判断}せよ
                (:性能が物理的法則に反している。上記参照。
            endif
        else
            (:記述割愛。10度Cより高い場合である。
        endif
    endif
else
    (:このポットの最大容量は２リットルである。
    (:２リットルより容量が少ない場合に
    (:上のような論理を組み入れて加熱器の性能を検査する必要があるかも確認する必要がある。
```

```
    (:機器が故障し火災などの事故をどう防ぐかなどの
    (:安全（機能安全）対策をどこまで行うか、が関係してくる。
endif
```

2.3.2 Req 02

Req 02 ： 電源コンセントをつなぐと、直ちに作動し、ヒータで加熱を始め、沸騰に達したら、90度C に保温する。

保温中であれば、お湯を注ぐことができる。

```
(:要求番号：Req 02
if <電源コンセント>が{接続}された
    Do<ヒータでの加熱>を{開始}せよ
    (:ヒータの稼働を水への加熱と見なすこととする。
    (:また要求では「直ちに」とあるので、コンセントをつないでから、
    (:加熱のボタンを別途押すということではないようである。
    if <水温>が{沸騰}した
        Do<ヒータでの加熱>を{停止}せよ
        (:ヒータの加熱動作を停止することで、水温が下がることを想定している。
        (:他に水温を下げるための機器はないと考えている。
        if <水温>が{90度Cに低下}
            (:水温を感知する温度センサーのようなものを備える必要がある。
            (:「10度C」の記載の箇所でも同じことが言える。
            Do<水温>を{90度Cで維持}せよ
            (:水温を90度Cを維持できる制御が必要であろう。
            if <給湯ボタン>が{押下}された
                (:ここでは給湯ボタンを新たに設定し、給湯を表現している。
                (:他に方法があるかもしれない。要、確認。
                Do<お湯>を{給湯}せよ
            else
                Do nothing
            endif
        else
            Do nothing
            (:ヒータの加熱を停止したままとする。
        endif
    else
        (:[Pending]
        (:条件（形式）を埋める意味が書かれていない。【意味の（形式的観点からの）欠落】といえる。
        (:<<確認項目>>  保温中は？沸騰する前は？
    endif
else
    (:  状態が変わらない場合なので何もしない
    Do nothing
endif
```

2.3.3 Req 03

Req 03 ： 再沸騰ボタンが押されたら、再沸騰を始める。

```
(:  要求番号：Req 03
(:
(:字面をそのまま書くと下の論理記述となるが、沸騰中であれば、加熱がなされない筈である。
(:仮に沸騰中に、再沸騰ボタンを押してみるなどと考えてみる。
(:すると、再沸騰ボタンが押されてもヒータでの加熱は開始しない筈である。
(:なぜなら、すでにヒータは加熱しているからである。
(:このとき「すでに加熱動作に入っている場合には、再加熱動作は無視される」などの知識（意味）が
(:が援用されている。
(:しかし、そうではないかもしれない。「加熱されれば、さらに加熱される（温度は強くなる）」
(:という機能も可能かもしれない。そういう知識も働く。
(:追加すべき知識（意味）がもっとあるともいえるし、また余分な知識を考えすぎるということもありう
る。
(:この点から、この要求分析は、【形式と意味を過不足なく補う作業】といえる。

if <再沸騰ボタン>が{押される}
    Do<ヒータでの加熱>を{開始}せよ
else
    Do nothing
    (:  状態が変わらない場合なので何もしない
endif
```

### 2.3.4 Req 04

Req 04 ： 水が加えられ、温度が低下したら、再沸騰を始める。

```
(:  要求番号：Req 04
if <水>が{ポットに加えられた}
    Do<水温>が{低下}する
    (:Do文を事実文に読み替えている。
    (:【意味と形式の不足】　「水温が低下」とは何度Cのことを指すのか。
    (:また保温状態は90度Cであるので、保温状態との条件の関係を明確にする必要がある。
    Do<水>を{再沸騰}させよ
    (:ここで「再沸騰」は、具体的にはヒータで加熱してそれにより再沸騰するという意味である。

    (:以上みたように、要求分析とは意味の内包化（一般化）と外延化（具体化）を
    (:繰り返してゆくことである。「refinement」と言われることでもあろう。
    (:この繰り返しの中で、電気ポットの性能や機能をどの程度のものにしたらよいかが、明らかになる。
    (:そして、性能や機能の特定に応じて、加熱、温度、沸騰等の諸元の物理学的関係も明らかになる。
    (:この例題は、この辺に留める。
else
    (:状態が変わらない場合なので何もしない。
    Do nothing
endif
```

### 2.4 《要求分析３》従来方式

ところで、先の２つの要求分析の方法をご覧になると、
実質的に今まで行って来たことと変わらないと思われるかもしれません。
変わっているのは、様式を用いる（形式に当てはめる）ぐらいのものである、というものです。

その通りなのですが、従来の方式は、形式的に文を分析するのではなく、内容を理解してから、分析を行うというものです。
この方式ですと、要求の内容を知らないと中々問いも発することができません。
仕様内容の知識を知るまで、小さくなっていなくてはなりません。

従来方式で知識を蓄えてから分析に入ったところで、所詮は知らないことの方が多いのですから、
要求仕様の内容がよく分からなくても、《要求分析１》で行ったように、
SLP構文で素直に文の形式を整え、問いを明確にして行った方が断然効率が上がると思います。
さらにまた、明確な問いが増えれば、小さくなっている必要はありません。
メンタルの改善はソフトエンジニアのモラルの向上にもつながります。

従来の方式もみてみます。例題１を用います。
分析者は次のように考え、そしてSLPを適用しました。
---
■クルーズ制御状態には、状態が、オン、オフ、スタンバイの３種がある。
　→所与の条件により、クルーズ制御の状態の３種のいずれかを出力するのが、この仕様の指示である。
　　なお、初期状態は必ずオフである。

以下のファクターに関する状態は以下である。
■ONボタン：押された、押されていない

■OFFボタン：押された、押されていない
■Resumeボタン：押された、押されていない
　→各ボタンは独立で存在し、どれか一つが押された場合、それ以外のボタンは押されていない状態とする（同時に押される事はない）
　→仕様からは読み取れなかったが、全てのボタンが押されていない場合もあるものと想定する。

■車速：制御可能範囲内、制御可能範囲外
■アクセルペダル：踏まれている、踏まれていない
■ブレーキペダル：踏まれている、踏まれていない

この仕様では、クルーズ制御状態を変化させる場合のことを列挙しており、クルーズ制御状態を遷移させない場合のことは書かれていない。
内容は暗黙的に仕様から読み取れるが、仕様作成者に正しいかどうか確認すべきである。
同様にボタンの仕様についても問い合わせるなどして確認すべきである。
（実際にはボタンは１つで、例えば、ボタンの位置で、ON、OFF、Resumeを切り替えているかもしれない。）
（この場合には、物理的にこの３つの状態（ON、OFF、Resume）が同時に成り立つことはない。）

この仕様はクルーズ制御状態を遷移させる事が目的なので、
現在のクルーズ制御状態と現在の要素の状態から次のクルーズ制御状態を求めるようにSLPで記述していく。
switch文を使用して、現在のクルーズ制御状態のオン、オフ、スタンバイの条件を用意する。
各case文では、if文で要素の判定を行う。判定結果としてクルーズ制御状態の変更を行う場合は、Do文で記述する。
クルーズ制御状態を変化させない場合は、「なにもしない」とDo nothing文で明示する。
Do nothingと明示できない場合（確信が無い場合）には、[Pending]とコメントする。
（検査では、構文を記述していないので、構文エラーとなる。）

switch文でクルーズ制御状態を遷移させる前に、クルーズ制御状態の初期状態を明示的に記述するため、
if文で〈初期化処理〉を{行う場合}という条件を設定し、
その条件の場合には、Do文でクルーズ制御状態をオフにするという記述を追加した。（２行目。）

なお、行数は100を超えたが、単位機能に分割しなかった。
共通化できる部分がある場合は、子単位機能（Fn文）に分割し、１単位機能の行数を削減し、わかりやすく整理できる。
---
この事例は、要求を字面だけで分析する場合（要求分析１）に比べ、Pending（未定、未確定）事項が少なくなっています。
これは、要求仕様の内容を整理（理解）し、条件同士を照合し、同じ場合には２つを一緒にする、
また異なる場合でもある条件の下に、他の条件が入らないかなど、
先の《要求分析１（１－１～１－３）》と同じようなことを行ったからです。
すなわち、条件という「形式」を比較し、条件を再構成し、
構成された条件（前件）に適った、仕様内容＝「意味」を後件の位置に置いた、といえます。

しかしこのようにしても、22行目等のように「Pending」事項（不明な「意味」）が何箇所か存在しています。
このことは、従来のように、仕様内容を理解してから、分析に掛からなければならないという手順を
一定の合理的な方法であれば、その方法から内容に入る根拠がある、ということを物語ります。

ドメイン知識が無いからといって、最初から小さくなることはありません。
知識を獲得する方法を持てば、巨大な知識（要求仕様）にも立ち向かうことができます。
また誰もが同じような方式で行いますから、成果物の均質性が保てます。

また確認すべき項目の数には多い少ないが出てくるかもしれませんが、
条件を絞って問いますので、的確な回答が期待できます。
そして時間も合理的に無駄無く的確に使うことができるといえます。

```
(:クルーズ制御状態の初期状態は必ずオフ
if <初期化処理>を{行う場合}
    Do<クルーズ制御状態>を{オフ}せよ
else
    (:初期化処理ではない場合はクルーズ制御状態を変更しない
    Do nothing
endif

(:状態遷移判定処理が定期的に繰り返し動作するものとする
switch <クルーズ制御状態>が
case{オン}
    if <OFFボタン>が{押された}
        Do<クルーズ制御状態>を{オフ}せよ
    else
        if <ブレーキペダル>が{踏まれている}
            Do<クルーズ制御状態>を{オフ}せよ
        else
            (:ブレーキペダルが踏まれていない
            (:クルーズ制御状態がオンの場合、ONボタン、Resumeボタンが押された場合の動作は定義されてい(:ないので、その判定は不要
            if <車速>が{制御可能範囲内}
                if <アクセルペダル>が{踏まれていない}
                    (:[Pending]
                    (:車速が制御可能範囲内＋ブレーキペダルが踏まれていない＋アクセルペダルが踏まれて(:いない場合の動作は定義されていない
                else
                    (:車速が制御可能範囲内＋ブレーキペダルが踏まれていない＋アクセルペダルが踏まれて(:いる場合はスタンバイ
                    Do<クルーズ制御状態>を{スタンバイ}せよ
                endif
            else
                (:車速が制御可能範囲内ではない
                Do<クルーズ制御状態>を{スタンバイ}せよ
            endif
        endif
    endif
```

```
case{オフ}
    if <OFFボタン>が{押された}
        (:クルーズ制御状態をオフにするが、既にオフの状態である
        Do nothing
    else
        if <ブレーキペダル>が{踏まれている}
            (:クルーズ制御状態をオフにするが、既にオフの状態である
            Do nothing
        else
            (:ブレーキペダルは踏まれていない
            if <ONボタン>が{押された}
                if <車速>が{制御可能範囲内}
                    if <アクセルペダル>が{踏まれていない}
                        if <ブレーキペダル>が{踏まれていない}
                            (:車速が制御範囲内＋アクセルペダルとブレーキペダルが踏まれていない
                            Do<クルーズ制御状態>を{オン}せよ
                        else
                            (:ブレーキペダルが踏まれている場合はオフにするが、既にオフの状態である
                            Do nothing
                        endif
                    else
                        (:[Pending]
                        (:車速が制御範囲内＋アクセルペダルが踏まれている場合の動作は定義されていない
                    endif
                else
                    (:車速が制御可能範囲内でない
                    Do<クルーズ制御状態>を{スタンバイ}せよ
                endif
            else
                if <Resumeボタン>が{押された}
                    (:ONボタンが押された場合と同じ
                    if <車速>が{制御可能範囲内}
                        if <アクセルペダル>が{踏まれていない}
                            if <ブレーキペダル>が{踏まれていない}
                                (:車速が制御範囲内＋アクセルペダルとブレーキペダルが踏まれていない
                                Do<クルーズ制御状態>を{オン}せよ
                            else
                                (:ブレーキペダルが踏まれている場合はオフにするが、既にオフの状態であ
                                (:る
                                Do nothing
                            endif
                        else
                            (:[Pending]
                            (:車速が制御範囲内＋アクセルペダルが踏まれている場合の動作は定義されてい
                            (:ない
                        endif
                    else
                        (:車速が制御可能範囲内でない
```

```
                        Do<クルーズ制御状態>を{スタンバイ}せよ
                    endif
                else
                    (:[Pending]
                    (:何も押されていない場合の動作は定義されていない(状態が変わらない)
                endif
            endif
        endif
    endif
case{スタンバイ}
    if <OFFボタン>が{押された}
        Do<クルーズ制御状態>を{オフ}せよ
    else
        (:クルーズ制御状態がスタンバイの場合、ONボタン、Resumeボタンが押された場合の動作は定義されて
        (:いないので、その判定は不要
        if <ブレーキペダル>が{踏まれている}
            Do<クルーズ制御状態>を{オフ}せよ
        else
            (:ブレーキペダルが踏まれていない
            if <車速>が{制御可能範囲内}
                if <アクセルペダル>が{踏まれていない}
                    Do<クルーズ制御状態>を{オン}せよ
                else
                    (:[Pending]
                    (:車速が制御可能範囲内＋アクセルペダルが踏まれている場合の動作は定義されていない
                endif
            else
                (:[Pending]
                (:車速が制御範囲外の動作は定義されていない
            endif
        endif
    endif
elsw
    (:クルーズ制御状態がオン、オフ、スタンバイではないのは有り得ない
    Do nothing
endsw
```

## 2.5 「形式」と「意味」に関する結論

いずれの分析も疑問がゼロという所まで行きませんでした。
《要求分析１》はelseの条件に関するPending事項を多く残し、
同じ例題を要求分析した《要求分析３》は事前に「意味」を整理してから、
SLPでの記述を行いましたが、それでもPending事項が残りました。

また別の例題では（例題２）、「意味を深める」必要をいいました。
要求には濃淡があります。
例題２は例題１に比べて、条件等が丁寧に書かれたものではありませんが、
取り上げたものが電気ポットですから、身近にあり、使い方などが想像しやすいものです。
したがって、要求仕様をさほど詳しく述べなくても理解ができやすいものでした。

しかし、要求を正確にして行くためには、物理学等科学の知識の援用や、
要求の表現をより具体的なもの（外延的なもの）に変える必要のあることを示しています。

以上を踏まえて要求分析をする上での注意点を整理すると以下になります。

１．よくできた仕様書も、もう一度「形式」を整えてみる。
　例題１の仕様書は、よくできた仕様書ですが、漏れがありました。
　漏れは条件の漏れであり、まずはこの手の「形式」を整えてみることが大事です。
　if-else-endifという形式で条件を洗い出すと、漏れが出てきました。

２．「意味」を明確にする。また深める。
　このような漏れ（if-else-endifの特にelse）に対して、要求の出し手は、
　「当然分かってしかるべき」と反論するような気がします。
　要求の出し手は次のようにいいそうです。
　「言外に言っている」とか、あるいは「そこまで細かく言わないといけないとは、
　あなたが不勉強である」。
　しかし事実の問題として、専門的、あるいは得意とする分野以外のことは、
　人は案外知らないものかもしれません。
　実際のそれらの知恵を動員してモノをつくる、という場合には、知識の曖昧さを知らされ、
　専門家すら改めて知識を確認する、書籍を読み直すなどはよくあることです。
　　例題２では、要求に書かれている以外の知識を導入すべきことを書いています。
　開発対象に関する固有の知識は、
　それが特殊なものであれば要求の出し手も最初はその知識を求めないと思いますが、
　それ以外の科学や法律などの公知の知識に関しては、
　要求分析者も知っておく必要があるでしょう。
　また要求者の漠然とした表現をより厳密な表現に変えて行く（外延化）場合にも、
　公知の知識を用いることが正確性を増します。

以上、「形式」と「意味」の２つを常に携えてアプローチすることが要求分析では重要であることを述べました。

## 3 要求トレーサビリティ

冒頭の「結論」の項で述べましたように、要求のトレーサビリティに関しては、次のことがいえます。

要求項目のトレーサビリティ（追跡性）を各工程の成果物に正確に反映させようとするなら、まずは要求項目をよく分析する必要があります。
要求項目の条件に漏れがあったまま、その要求項目を追跡対象にしてしまうと、

後続の設計文書やプログラムコード（実装）領域に、要求項目の反映されていない個所が生じます。

要求項目の条件に漏れがあるとは、要求項目が形式的に不完全であることを指します。
形式的に不完全とは、要求項目がif-else-endifの構造を満たしていないことを指します。
if-else-endifを構成する前件や後件（Do文）が存在しなければなりませ。
しかし前件や後件の「意味」内容まで正しくなければならないとは言っていません。

もちろん、形式のみではなく「意味」も正しいことが望まれます。
しかし、正確さを客観的に確定する作業は、思うほど容易な作業ではありません。
ここでは意味の正確さが確定されている状態とは、
仕様の内容に関する疑問が、要求仕様の出し手によって明確にされている状態を指すこととします。別の言い方をすれば、要求の出し手が、前件が正しいと認め、かつ後件も正しいと認めている状態です。

Do文は「せよ」という命令文ですので、その正誤は属人的です。要求がなかなか決まらない、あるいは定着しないという場面をご想像ください。要求の出し手に迷いが生じた場合、受け手には決めようがありません。このような場合には、せめて「形式」（条件の構造）を整え、仕様の出し手が「意味」を提示しやすく図るようにすること、これがSLPの要求分析に対する考え方です。
SLPでは要求のトレーサビリティを正しく後続の文書（機能仕様書等）に反映させるためには、
「形式」と「意味」の整った要求項目を要求の対象とせよ、と主張します。

### 3.1 どれを要求にするか

要求のトレーサビリティという場合、どれを要求にするかがまず問われなければなりません。
SLPの主張は、「形式」と「意味」を整えたものを要求（項目）とせよ、というものです。
しかし例題１で見たように、６つの要求項目はいずれも形式化を行うと意味の未確定なところが残りました。これでは正しい要求とは言えません。

用語を導入します。
要求を確定して行く過程では、未確定状態の要求を要求の「候補者(candidate)」と呼ぶことができます。
すると、要求が確定したものは要求の「確定者(finalist)」と呼ぶことができます。
そしてSLPでは「要求は不確定な事項を含む候補者(candidate)のままにとどめるのではなく、確定者(finalist)になるまで分析されるべきだ」と主張します。

要求を確定するための方法は、項番3.3で述べるように、様々の方法がありますが、SLPでは「原要求項目」という概念を用い、提示された要求項目の識別子を「原要求項目」欄に登録することから、要求の分析を始めることができるようになっています。例題１でいえば、１～６の要求をそのまま原要求項目とすることができます。先に見たようにこれらの原要求項目はif-else-endif等の要求分析を通じて、未確定の内容を少なくして行きます。

「原要求項目」欄に要求識別子（番号や簡単な項目タイトルなど）を登録すれば、この識別子は後続の要求分析にそのまま継承されます。要求を細かく分割して行く場合がありますが、その場合にも要求識別子は子の単位機能に継承（トレース）されて行きます。

他方、要求項目のif-else-endif等で「形式」を整え、「意味」内容を確定してから、確定後のものを要求項目とする方法もあります。これを行ってのち、要求項目を「原要求項目」として登録します。いずれの方法であっても要求の分割を行った場合には分割された子の単位機能に要求識別子の継承がなされます。

以下では、「要求」の形式を整え、「意味」を確定して行く方法について述べます。
例題１を例にとって述べます。

### 3.2 要求項目の重複の整理の仕方

例題１の要求項目は重複しているものがあります。
両者を整理しまとめることで、要求が簡潔になります。
要求の量と複雑さが増大している中では要求を簡潔にすることは重要な作業となります。

以下ではその方法を述べます。
ここでも「形式」と「意味」の問題が考え方の基本です。
また要求が実施（実行）される要求の順序の問題も考慮が必要となります。

例として、次のような要求項目《R1》と《R2》を考えます。

```
《R1》
Do<A>を{B}せよ
if <C>は{D}ならば
   Do<E>を{F}せよ
else
   Do<G>を{H}せよ
endif

《R2》
if <C>は{D}ならば
  if <L>は{M}ならば
     Do<N>を{O}せよ
  else
     Do<P>を{Q}せよ
  endif
else
  Do<R>を{S}せよ
endif
```

《R1》と《R2》において、要求仕様の内容に関して、処理的な順番を考慮する必要の無いばあいには、下記の《統合案》が可能です。

```
《統合案》
Do<A>を{B}せよ・・・①
if <C>は{D}ならば
  Do<E>を{F}せよ・・・②
  if <L>は{M}ならば・・・③
     Do<N>を{O}せよ
  else
     Do<P>を{Q}せよ
  endif
else
  Do<R>を{S}せよ
  Do<G>を{H}せよ・・・④
endif
```

この《統合案》は「形式」的には何ら問題がありません。
①は問題がないように思われます。《R2》では先頭に「if <C>は{D}」があり、
《R1》のように「Do<A>を{B}せよ」がありません。
しかしもし、「Do<A>を{B}せよ」が内容的に（「意味」的に）
「if <C>は{D}」に影響をあたえるものであるならば、
《R2》では何らの影響も受けずに「if <C>は{D}」が存在しますから、
相互の影響について、考慮が必要です。
ここで「影響」とは、メンバー名や状態名が、前の行の「意味」によって
影響を受ける場合です。

②は《R１》と同じ位置にありますから、《統合案》は問題ありません。

しかし③は《R2》の場合と同じようにこの位置でよいのかは不明です。
というのも、③は②に影響を受けないかは「意味」の問題があるからです。

同じようなことは、④についてもあてはまります。
④も同じ条件下（「<C>は{D}でない」）ですので、OKのように見えます。
しかしこれは、「Do<R>を{S}せよ」と「Do<G>を{H}せよ」が相互に独立であれば
問題はありませんが、相互に影響を及ぼす場合には、検討を要します。
「意味」の問題となります。

「形式」の観点から重複を整理し簡潔なものにすることは、そこにはアリゴリズムが存在し、従ってSLPの機能とする能性を持っています。また、elseを立てることによる網羅性ばかりではなく、そもそもifやDoを過不足なく立てられる「意味」の網羅性に関しても、研究の余地があります。意味の「オントロジー」として今後調査研究をしたいと考えております。

### 3.3 要求の特定法

最初に提示された個々の要求項目に関して、どれを要求項目にするかを決める方法には、様々の方法があります。
以下の表は、本稿で試みられたものの一覧表です。

| 方法 | 形式を整える | 意味を確定しようとする | 要求分析 | 説明 | 要求分析の結果、意味を確定できたか | 分析例 |
|---|---|---|---|---|---|---|
| （1） | × | × | していない | 例題1の要求項目をそのまま要求項目とする。 | 確定作業自体をせず | |
| （2） | × | ○ | している | 要求の形式ではなく、意味を確定する。 | 未確定は残った | 要求分析3 |
| （3） | ○ | × | 〃 | 要求の形式化のみを行い、Pendingの指定は行うが、確定作業はしない。 | 形式化のみで意味確定はせず | 要求分析0 |
| （4） | ○ | ○ | 〃 | 意味確定を部分的に行った。 | 未確定は残った | 要求分析4 |
| （5） | 〃 | 〃 | 〃 | 個々の要求項目を要求とするのではなく、より上位の大分類の概念を要求とした。 | 〃 | 要求分析5 |
| （6） | 〃 | 〃 | 〃 | 個々の要求項目を1つにまとめ、全体を要求とする。 | 〃 | 要求分析1－3 |

（1）の方法は、要求分析をしていないものです。この場合には、トレーサビリティを行う場合には、与えられた要求項目をそのまま要求項目とすると考えられます。
（2）はSLPを用いることもなく、不明な内容（意味）を明らかにしようというものです。
（3）は要求の形式化を試みただけで、要求の意味内容の確定をしようとしなかったものです。
（4）は形式化に加えて、意味も確定しようとするものです。
（5）は最初に提示された要求項目では、要求分析を進めて行くと、要求の重複があることが分かったので、重複を防ぐために、個々の要求項目の、さらに上位の項目（大分類）で、要求項目をしたらどうかという試みを示しています。

（６）は大分類でも未確定な項目が残るので、全体を１つの要求としてはどうかという試みです。それでも未確定は残るのですが、重複が避けられるのではないかという試みです。

どの方法がベストかは、甲乙がつけがたいところです。
なによりも重要なのは、要求項目を決定する場合には、その中に未確定事項を残さないことですが、例題では要求の出し手が不在で、確認にしようがないのですが、どの方法であっても未確定事項を残さないようにすることが第一に肝心です。

最初に提示された要求項目を要求項目とする場合にも、またこれらの要求項目を１つにまとめて要求項目とするように変える場合にも、未確定事項を残さないことです。なぜならば、未確定事項も後ろの工程にそのままトレースされるからです。
（３）は要求の確定はしていませんが、SLPを用いていますから、今後SLPで要求分析を行うと、（４）のようになります。またこのとき、要求項目を「原要求項目」として、原要求項目欄に記入していると考えられますから、要求識別子の継承は自然に行われることになります。このことは先に「どれを要求にするか」の項で述べた通りです。
（５）、（６）は要求の重複を避けるための方法を述べています。

以下の後続（子）の単位機能においては、上の（３）～（６）に関する事例を述べます。

### 3.3.1 （３）：個々の要求項目をそのまま要求としてみなしてトレースする場合：《要求分析０》

ここでは、要求項目をそのまま要求にしてトレースをしようとした場合を見てみます。
「そのまま」とは《要求分析０》でみたように、字面をSLPで整えただけで、
それをそのまま要求項目にしてしまうことをいいます。

要求項目１のみを論理記述欄に例示しています。
前掲と異なるのは、未確定な個所（ここではelseの個所）に「Pending」と記述し、
要求の出し手に問うべき個所を明示していることです。

《要求分析０》はPending事項を残したままですから、
このままトレースの対象にするのは、明らかに不都合です。
要求の出し手に確認すべきです。

```
(:要求項目１．クルーズ制御状態がオフで，ONボタン又はResumeボタンが押され，
(:車速が制御可能範囲内で，アクセルペダルとブレーキペダルが踏まれていない時

if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
            else
                (:[Pending]
            endif
        else
            (:[Pending]
        endif
    else
        (:[Pending]
    endif
else
    (:[Pending]
endif
```

### 3.3.2 （４）：個々の要求項目の重複を整理し１つにまとめ、要求の未確定を残したままトレースする場合：《要求分析４》

下記の論理記述は、１～６の要求項目（「原要求項目」）の重複を整理し1つにまとめた場合に、
全体の中で原要求項目がどこに位置するかを表現したものです。
例えば、論理記述欄の６行目に「原要求番号１」と書いているのが、例題１の要求項目の項番１に相当します。
しかし、この方法では項番の割り当てられない行が存在しています。９行目等がそうです。
「Do Nothing」と書いてはいますが、コメント行に疑問が書かれていますから、実際には「Pending」として要求の出し手に確認せざるを得ません。
この行は要求は「候補者」のままです。

```
if <クルーズ制御状態>が{オフである}
    if <ONボタンまたはResume>が{押されている}
        if <車速>が{制御可能範囲内である}
            if <アクセルペダル>が{踏まれてない}
                if <ブレーキペダル>が{踏まれてない}
                    (:原要求番号 1.
                    Do<クルーズ制御>を{オン}せよ
                else
                    (:クルーズ制御オフを維持するのであろう
                    (:スタンバイ状態でもない、車速セットするまでの中間状態なのか？（以下同様）
                    Do nothing
                endif
            else
                (:クルーズ制御オフを維持するのであろう
                Do nothing
            endif
        else
            if <アクセルペダル>が{踏まれてない}
                (:クルーズ制御オフを維持するのであろう
                Do nothing
            else
                if <ブレーキペダル>が{踏まれてない}
                    (:原要求番号 5.
                    Do<クルーズ制御>を{スタンバイ}せよ
                else
                    (:クルーズ制御オフを維持するのであろう
                    Do nothing
                endif
            endif
        endif
    else
        (:ボタンOFFが押されている
        (:原要求番号 3.　（ボタンOFF時の初期状態）
        Do<クルーズ制御>を{オフ}せよ
    endif
```

```
else
    if <クルーズ制御状態>が{スタンバイ}
        if <ボタン>が{ONまたはResumeが押されている}
            if <車速>が{制御可能範囲内である}
                if <アクセルペダル>が{踏まれてない}
                    (:原要求番号 2.
                    Do<クルーズ制御>を{オン}せよ
                else
                    (:スタンバイ状態を維持するのであろう
                    Do nothing
                endif
            else
                (:スタンバイ状態を維持するのであろう
                Do nothing
            endif
        else
            (:ボタンOFFが押された
            (:原要求番号 4.  (ボタンOFFが押された時)
            Do<クルーズ制御>を{オフ}せよ
        endif
    else
        (:クルーズ制御状態がオンである
        if <ONボタンまたはResume>が{押されている}
            if <車速>が{制御可能範囲内である}
                if <アクセルペダル>が{踏まれている}
                    (:原要求番号 6.
                    Do<クルーズ制御>を{スタンバイ}せよ
                else
                    if <ブレーキペダル>が{踏まれている}
                        (:原要求番号 4.
                        Do<クルーズ制御>を{オフ}せよ
                    else
                        (:クルーズ制御オンを維持するのであろう
                        Do nothing
                    endif
                endif
            else
                (:原要求番号 6.
                Do<クルーズ制御>を{スタンバイ}せよ
            endif
        else
            (:OFFボタンが押された
            (:原要求番号 4.  (ボタンOFFが押された時)
            Do<クルーズ制御>を{オフ}せよ
        endif
    endif
endif
```

### 3.3.3 （５）個々の要求項目の上位概念を要求項目とした場合：《要求分析５》

ここでは個々の要求項目ではなく、上位の概念（または大分類）で要求をくくることを試みました。
論理記述欄の「条件１．【クルーズオン制御】」は、例題１の要求項目の１と２の上位概念です。
しかしそれでもカバーし切れないケースが存在しています。
論理記述欄の９行目がそうです。「だろう」と推定しています。
そこで、上位概念(大分類）を用いながらも条件を規定することを試みました。
１４行目の「条件5.【クルーズスタンバイ制御】（アクセルペダルが踏まれて，ブレーキペダルが踏まれていない時）」がそれです。
「（アクセルペダルが踏まれて，ブレーキペダルが踏まれていない時）」として、上位概念(大分類）の中の個々の条件を明らかにして行けば、原要求項目との関係づけることができると考えられます。しかしここでも原要求項目に未確定事項が存在するとすれば（事実存在する訳ですが）、関係づけはお手上げです。１６行目のelseの中は未確定事項です。

このように個々の要求項目の上位概念を要求項目とする方法でも、未確定事項が存在する以上、要求は要求候補者の域を出ていません。

```
if <クルーズ制御状態>が{オフである}
    if <ボタン>が{ONまたはResumeが押されている}
        if <車速>が{制御可能範囲内である}
            if <アクセルペダル>が{踏まれてない}
                if <ブレーキペダル>が{踏まれてない}
                    (:条件1.【クルーズオン制御】
                    Do<クルーズ制御>を{オン}せよ
                else
                    (:Pending クルーズ制御オフを維持するのであろう
                    Do nothing
                endif
            else
                if <ブレーキペダル>が{踏まれてない}
                    (:条件5.【クルーズスタンバイ制御】（アクセルペダルが踏まれて，ブレーキペダルが踏(:まれていない時）
                    Do<クルーズ制御>を{スタンバイ}せよ
                else
                    (:Pending アクセル／ブレーキ同時踏みの場合はどうするのか？
                    (:Pending クルーズ制御オフを維持するのであろう
                    Do nothing
                endif
            endif
        else
            (:Pending 条件5.の車速が制御可能範囲外なら、アクセル／ブレーキはDon't careか？
            (:条件5.【クルーズスタンバイ制御】（車速が制御可能範囲内でないとき）
            Do<クルーズ制御>を{スタンバイ}せよ
        endif
    else
        (:ボタンOFFが押された
        (:条件3.【クルーズオフ制御】（初期状態）
        Do<クルーズ制御>を{オフ}せよ
    endif
```

```
else
    if <クルーズ制御状態>が{スタンバイ}
        if <ボタン>が{ONまたはResumeが押されている}
            if <車速>が{制御可能範囲内である}
                if <アクセルペダル>が{踏まれてない}
                    (:Pending 条件2.の車速が制御可能範囲内なら、ブレーキはDon't careか？
                    (:条件2.【クルーズオン制御】
                    Do<クルーズ制御>を{オン}せよ
                else
                    if <ブレーキペダル>が{踏まれてない}
                        (:Pending スタンバイ状態を維持するのであろう
                        Do nothing
                    else
                        (:Pending 条件4.の「ブレーキペダルが踏まれるか」は、車速／アクセルはDon't
care
                        (:か？
                        (:条件4.【クルーズオフ制御】（ブレーキペダルが踏まれた）
                        Do<クルーズ制御>を{オフ}せよ
                    endif
                endif
            else
                (:Pending スタンバイ状態∧ONまたはResumeが押されている∧車速が制御可能範囲外ならばア
                (:クセル／ブレーキはDon't careか？
                (:Pending スタンバイ状態を維持するのであろう
                Do nothing
            endif
        else
            (:ボタンOFFが押された
            (:Pending 条件4.の「OFF ボタンが押された時」は、車速／アクセル／ブレーキはDon't careか？
            (:条件4.【クルーズオフ制御】（ボタンOFFが押された時）
            Do<クルーズ制御>を{オフ}せよ
        endif
    else
        (:クルーズ制御状態がオンである
        if <ボタン>が{ONまたはResumeが押されている}
            if <車速>が{制御可能範囲内である}
                if <アクセルペダル>が{踏まれている}
                    (:条件6.【クルーズスタンバイ制御】
                    Do<クルーズ制御>を{スタンバイ}せよ
                else
                    if <ブレーキペダル>が{踏まれている}
                        (:条件4.【クルーズオフ制御】（ブレーキペダルが踏まれた）
                        Do<クルーズ制御>を{オフ}せよ
                    else
                        (:クルーズ制御オンを維持するのであろう
                        Do nothing
                    endif
                endif
            else
```

```
                (:Pending 条件6.の「車速が制御可能範囲内でない」なら、アクセル／ブレーキはDon't
care
                (:か？
                (:条件6.【クルーズスタンバイ制御】
                Do<クルーズ制御>を{スタンバイ}せよ
            endif
        else
            (:ボタンOFFが押された
            (:Pending 条件4.の「OFF ボタンが押された時」は、車速／アクセル／ブレーキはDon't care
か？
            (:条件4.【クルーズオフ制御】（ボタンOFFが押された時）
            Do<クルーズ制御>を{オフ}せよ
        endif
    endif
endif
```

3.3.4 （６）個々の要求項目を１つにまとめて、これ全体を要求としてトレースする場合：《要求分析１－３》
要求項目の「字面」をSLPで分析するだけでもダメで、かといって１つに個々の要求項目を整理しまとめてもPendingが残り、そこで上位の概念でまとめても同じくPendingが残りました。
結論は最初から明らかで、elseが原要求項目に記されていないからです。
このことはすでに見てきたとおりですが、その中でも、要求確定が比較的よくできた例として、論理記述欄は先に行った《要求分析１－３》を載せます。これは要求項目を１つにまとめた例です（Pendingはそのままですが）。

```
if <クルーズ制御状態>が{オフ}
    if <ONボタン又はResumeボタン>が{押され}
        if <車速>が{制御可能範囲内}
            if <アクセルペダルとブレーキペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のifの4つの重なりは、例題１の項番１のケースである。
            else
                if <アクセルペダル>が{踏まれている}
                    if <ブレーキペダル>が{￢踏まれている}
                        Do<クルーズ制御>を{スタンバイ}とせよ
                        (:このケースは、例題１の項番５の後半「又は」以降のケースである。
                    else
                        Do nothing
                        (: <ブレーキペダル>が{踏まれる}ケースである。
                        (:ここでは、例題１の項番４の前半「ブレーキペダルが踏まれるか」のケースを
                        追加(:した。
                        (:「踏まれる」と「踏まれている」の時制の微妙な違いもある。
                        (:SLPではこのような違いは形式化していない。
                        (:重要な違いが発生する場合には表現で補うこととする、としている。
                    endif
                else
                    Do nothing
                    (: ここでは、上の場合分けからして、
                    (:<アクセルペダル>が{￢踏まれる}かつ<ブレーキペダル>が{踏まれる}ケースとな
る。
                endif
```

```
        else
            (:<車速>が{¬制御可能範囲内}のケースである。
            (:このケースには分析１の４６行目に書かれているものを挿入した。
            Do<クルーズ制御>を{スタンバイ}とせよ
            (:このケースは、例題１の項番５の前半「、、、又は」の前までのケースである。
        endif
    else
        (:Pending
    endif
else
    if <クルーズ制御状態>が{スタンバイ}
        if <車速>が{制御可能範囲内}
            if <アクセルペダル>が{踏まれていない}
                Do<クルーズ制御>を{オン}せよ
                (:上のelseとifの3つの重なりは、例題１の項番２のケースである。
            else
                Do<クルーズ制御>を{オフ}とせよ
                (:このケースは、例題１の項番４の前半「ブレーキペダルが踏まれるか」のケースであ
る。
                (:Pending
            endif
        else
            Do nothing
            (:Pending
        endif
    else
        if <クルーズ制御状態>が{オン}
            if <ブレーキペダル>が{踏まれる}
                Do<クルーズ制御>を{オフ}とせよ
                (:このケースは、例題１の項番４の前半「ブレーキペダルが踏まれる」のケースである。
            else
                if <OFFボタン>が{押された}
                    Do<クルーズ制御>を{オフ}とせよ
                    (:このケースは、例題１の項番４の後半「OFFボタンが押された時」のケースである。
                else
                    (:Pending
                    Do nothing
                endif
            endif
        else
            if <クルーズ制御状態>が{初期状態}
                Do<クルーズ制御>は{オフ}である
                (:このケースは、例題１の項番３のケースである。
            (:Pending
            else
                Do nothing
            endif
        endif
    endif
endif
```

### 3.4 要求項目をトレースするSLPの操作法

以上の議論はどの方法がよいという訳ではありません。

未確定事項をなくすること、重複をなくし、簡潔にすること。その上で、要求項目を特定するのがよいと思いますが、これは要求の受け手の立場であって、他方要求の出し手は個々の要求項目（原要求項目）は反映（トレース）されているのかと問うでしょう。

このような観点からは、最初に出された要求項目を原要求項目として、「原要求項目」欄に登録した上で、要求分析により「形式」と「意味」を明確にして行くことが自然な方法と思われます。

SLPはこのような方法を備えています。要求項目をトレースするためには「原要求項目」の欄を用い、以下のように行います。

① 未確認事項を無くした最初の要求を原要求として「原要求項目」欄に登録する。
この原要求項目を単位機能に分割して行けば、原要求項目が最下の単位機能に自然に反映され、
原要求項目の所在がわかります。
例題１で言えば、「要求項目１」とか「２」とかを記したものを「原要求項目」欄に記入し、
機能分割を行えば、それらの「１」等が下位に継承されます。

② 例題２では、要求の原案の番号である「PR01」等を「Req 01」等にまとめたりしました。
　このような場合には、「PR01」等と「Req 01」等の両方を、「原要求項目」欄に記入すれば、
両者が下位に継承され、どちらの項番でも任意の単位機能に、
どの要求が反映されているのかを見ることができます。

## 4 要求仕様はどこまで書くべきか

要求には内容の濃淡があります。
要求の受け手は、常にこの問題に悩まされます。
本稿では、「形式」を整え「意味」を問えば、受け手は臆する必要は無いと言っています。
SLPはこのような備えを持っています。

口答、つぶやき、またメモでも要求たりえます。
これらを整理し、項番を振り、SLPの形式で漏れや用語をチェックし、意味を確認します。
このような要求分析により、形式の整った、一定のレベルの意味内容を持った要求仕様ができます。

しかし、こうしたとしても意味内容の濃淡の問題は解決しません。
一体どこまで（どのレベルの内容の濃度で）要求仕様書を書けばよいのでしょうか。
その解答は「要求仕様は目的に応じた内容を、目的がかなう濃度で書くべき」というものです。

「仕様が分からないから見積もりができません。」
とするなら、見積書を書ける程度に、仕様内容を濃くするしかありません。
「形式」と「意味」を整え、
条件の積み重なる個所や、意味内容をもっと掘り下げるべきところを明示し、
見積もりが狂う原因を事前に明確にしておくことが大事と思います。
また、「ザアーッとでいいから、見積もりを書いて。」と言われます。
弱りますが、「ザアーッ」とやってみます。
抽象度の高いところ、あるいはボンヤリしたところにも「形式」と「意味」があります。

要求仕様をどこまで書くかは、目的次第で、

「要求仕様は目的に応じた内容を、目的がかなう濃度で書くべき」です。
しかし「形式」と「意味」で要求を分析する、が肝要なところです。

その上で、機能安全の規格をみてみます。

### 4.1 ISO 26262の要求仕様書のレベル

ＪＡＳＰＡＲが最近、
『機能安全対応のための解説書【ソフトウェア編】（ISO 26262-6:2011）一般社団法人　JASPAR編』
を上梓しました。

要求仕様書をどこまで書けばよいかという問いに、同書は次のように書いています（P.9）。
少し長いですが、規格の趣の全体が感じられますので、そのまま引用します。
---
　技術安全要求から、 ASILを含むかたちで、ソフトウェアの実装・検証が可能なレベルに詳細化されたものがソフトウェア安全要求仕様である。具体的な要求の作成作業としては､安全要求の仕様およびマネージメント(Part8-6)の一部としてPart3で求められた機能安全要求に対応した技術安全要求(Part4-6にて作成）とシステム設計仕様をもとに､ソフトウェアの実装レベルにまでに詳細化していくことが求められる(「図６．１ソフトウェア安全要求の作成」を参照)。
　また、同時にハードウェアソフトウェアインタフェース仕様(Part5-6)についても詳細化していくことが求められる。
　作成したソフトウェア安全要求､ハードウェアソフトウェアインタフェース仕様が技術安全要求、システム設計、ハードウェア制約に従っていることを検証することが求められている。
---
要するに、実装と検証が可能なレベルの要求仕様が求められる、と書かれています。
また参考までに、ここで述べられている「図６．１」は、要求をさらに技術的な知識で補うものでなければ、詳細なレベルの要求仕様とはならない、ということを図示しています。
そのまま掲載します。

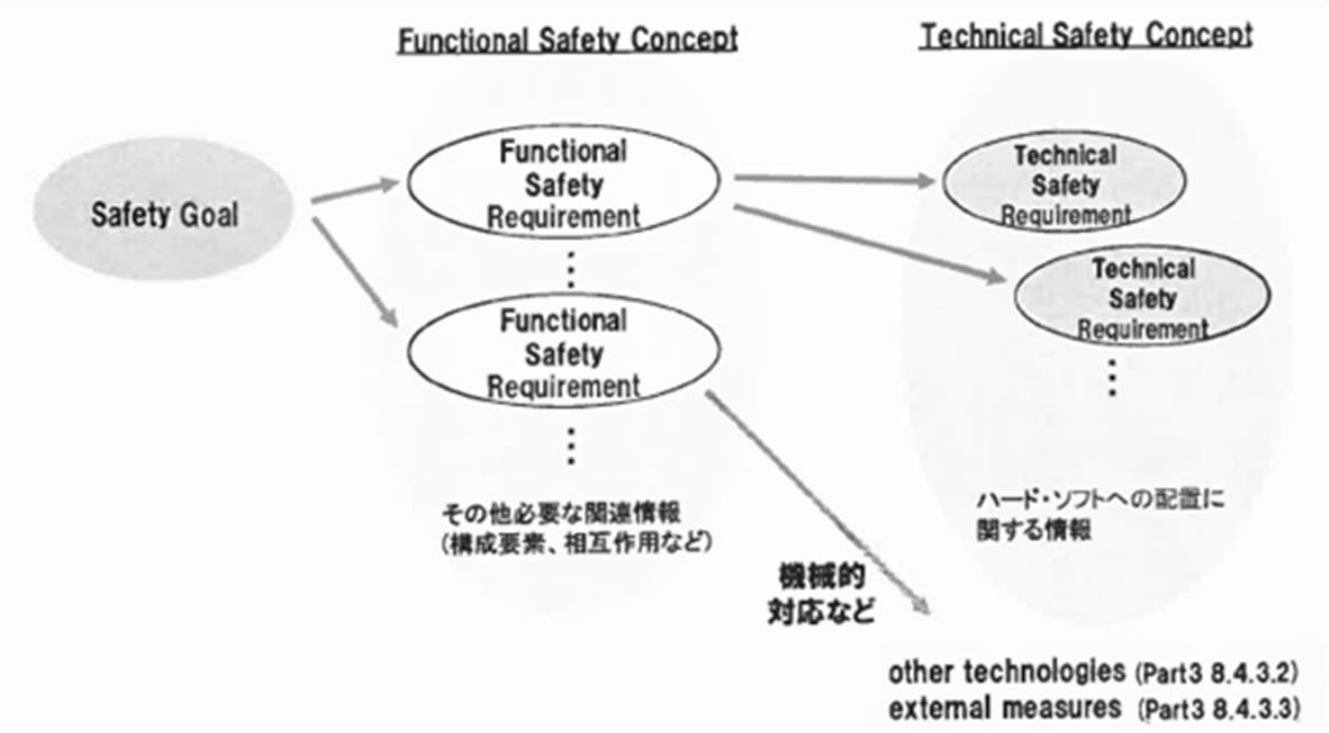



図 6-1　ソフトウェア安全要求の作成

## 4.2 ISO 26262の安全要求仕様事例１の検討

そこで同書が事例としてあげている要求仕様書が、「実装・検証が可能なレベルに詳細化されたもの」であるのか、検討してみようと思います。
次の例が同書のページ22～23に書かれています。

| | |
|---|---|
| ソフトウェア安全要求 No(ID) | SSR-001 |
| 参照（要求元） | TSR-005 |
| ASIL レベル | C |
| 要求概要 | 各入力信号、アクチュエータまたは制御系の故障判定時に下記の動作を行なうことで異常な出力をすることを回避する。<br>● フェールセーフリレーCut<br>● 各アクチュエータの駆動停止<br>● ABS ワーニングランプを点灯 |
| ソフトウェア仕様 | 故障時出力処理<br>故障時は通常の出力処理に優先して下記処理を実施する。<br>● 故障時出力判定条件<br>条件：OR ― 4 輪のうちいずれかがスピードセンサ断線状態<br>― 4 輪のうちいずれかがスピードセンサショート状態<br>― 車輪速センサモニタ故障状態<br>― 4 輪のうちいずれかがソレノイド異常状態<br>― ポンプモータ異常状態<br>成立時：フェールセーフリレーOFF 出力 |
| | 4 輪のソレノイド OFF 出力<br>ポンプモータ OFF 出力<br>ABS ウォーニングランプ ON 出力<br>● 故障時出力解除条件<br>なし<br>● 初期値<br>フェールセーフリレーON 出力<br>4 輪のソレノイド OFF 出力<br>ポンプモータ OFF 出力<br>ABS ウォーニングランプ OFF 出力 |
| ハードウェア I/F・相互作用 | フェールセーフリレー出力<br>4 輪のソレノイド出力<br>ポンプモータ出力<br>ABS ウォーニングランプ出力 |
| 非安全系 I/F・相互作用 | |
| 制限事項 | |

### 4.2.1 安全要求仕様事例１の要求分析

要求概要は次のようになっています。

---

各入力信号､アクチュエータまたは制御系の故障判定時に下記の動作を行なうことで異常な出力をすることを回避する。

- フェールセーフリレーCut
- 各アクチュエータ駆動停止
- ABSワーニングランプを点灯

---

この読み手は「意味」を知らない人だとします。
すると、この内容を理解するために「形式」を使おうとします。
「各入力信号､アクチュエータまたは制御系の故障判定時」は、
「各入力信号の故障判定」､「アクチュエータの故障判定」または「制御系の故障判定」の時
と用語が結合されました。・・・形式化
次に、これら３事象の間にある「､」､「または」､「の時」をどう形式化すれば良いのか迷います。
３事象は同時に起こるのか、それとも１つずつなのか。
事象の「形式」化ができません。・・・形式化
また、３事象は「故障」なのか「判定」まで入るのか､「意味」がはっきりしません。
おそらく､「判定」されなければ「故障」とは見なされませんので、
判定するための基準が必要で、その基準も自動判定するであろうと､「意味」を想像します。・・・意味解釈
いずれの故障でもなかったと判定された場合は、正常動作なのだろうと思われますが、
これも念のため確認します。
（正常動作とうまく関係するように設計しなければならないと想像されます。）・・・意味解釈

```
if <各入力信号>が{故障判定}された
    (:「判定」とあるが、自動判定されて、「故障判定時の異常出力を回避する」を行う。
    Fn [ 故障判定時の異常出力を回避する ]
else
    if <アクチュエータ>が{故障判定}された
        Fn [ 故障判定時の異常出力を回避する ]
    else
        if <制御系>が{故障判定}された
            Fn [ 故障判定時の異常出力を回避する ]
        else
            (:Pending：正常動作なので、何もする必要はないだろう。
            Do nothing
        endif
    endif
endif
```

#### 4.2.1.1 故障判定時の異常出力を回避する

ここでは以下の３種を行うこととされています。

- フェールセーフリレーCut
- 各アクチュエータ駆動停止
- ABSワーニングランプを点灯

おそらくこれらをすべて行え、ということでしょうが、その順番はどうなのか、
読み手は迷います。「意味」を知らないからです。
仕様の出し手から「常識」だろうといわれそうです。
そこで「形式」的なことの確認としては、同時に可能（順序性無縁）かを問う必要があります。

しかし、「ソフトウェア仕様」（添付の表）にこの「故障時出力処理」が書かれています。
これを読むことで、これらの疑問の幾つかが解消しそうです。

故障時に異常出力をしないために、「通常の出力」処理ではなく、
通常出力に割り込んで優先的に、以下の処理をしなければならないとあります。

しかし、よく「意味」を読み解くと、
「判定」のためには、「条件：ＯＲ」に書かれたことを常に監視しておく必要がありそうです。
常に監視しておいて、５つの条件のいずれかが発生したなら、
「成立時」のことを行え、ということのようです。
この成立時の用語をみてみると、先に「概要」に書いてある３種のことのようだということが分かります。
しかし、形式的には概要では３種書いてあり、「成立時」では４つ書いてありますから、
「意味」の確認が必要です。
このことにより、概要で書かれた３種をもう少し詳しく書くことができます。
また初期条件や「故障時出力解除条件」も書かれていますので、
書かれている内容がどのような動きをするのか、理解が及ぶようになります。
----------
故障時は通常の出力処理に優先して下記処理を実施する。
● 故障時出力判定条件
　条件：ＯＲ－－４輪のうちいずれかがスピードセンサ断線状態
　　－４輪のうちいずれかがスピードセンサショート状態
　　－車輪速センサモニタ故障状態
　　－４輪のうちいずれかがソレノイド異常状態
　　－ポンプモータ異常状態
　成立時：フェールセーフリレーＯＦＦ出力
　　４輪のソレノイドＯＦＦ出カ
　　ポンプモータＯＦＦ出力
　　ＡＢＳウォーニングランプＯＮ出力
● 故障時出力解除条件　　なし
● 初期値
　フェールセーフリレーＯＮ出力
　４輪のソレノイドＯＦＦ出力
　ポンプモータ　ＯＦＦ出力
　ＡＢＳウォーニングランプＯＦＦ出力
----------

下記の論理記述では、初期状態とそうでない（稼働中）時とを、論理的に記述しています。
これがそのまま、処理の流れまで記述する設計になるということではありません。
場合分けがこれで過不足ないかを確認するためのものです。

```
(:最初「概要」のレベルの意味解釈で書いたものです。
Do<フェールセーフリレー>を{Cut}せよ
Do<各アクチュエータ駆動>を{停止}せよ
Do<ABSワーニングランプ>を{点灯}せよ

(:↓　以下は「ソフトウェア仕様」をみて、上の３行を書き換えたものです。
(:しかし、例えば「Do<フェールセーフリレー>を{Cut}せよ」が下位のＦｎの[故障時異常出力を停止せよ]の中の
(:どの項目に相当するのか、分析者は実は「意味」を分かっていません。。要、確認事項です

if <当該システム>が{初期状態}
    Fn [ 故障時異常出力停止機能の初期化を行う ]
else
    (:システムが稼働中。また下記に関しては故障が同時の場合の処理に関してもどう処理するか確認の必要がある。
    if <４輪のうちいずれかのスピードセンサ>が{断線状態}である
        Fn [ 故障時異常出力を停止せよ ]
    else
        if <４輪のうちいずれかのスピードセンサ>が{ショート状態}である
            Fn [ 故障時異常出力を停止せよ ]
        else
            if <車輪速センサモニタ>が{故障状態}
                Fn [ 故障時異常出力を停止せよ ]
            else
                if <４輪のうちいずれかのソレノイド>が{異常状態}
                    Fn [ 故障時異常出力を停止せよ ]
                else
                    if <ポンプモータ>が{異常状態}
                        Fn [ 故障時異常出力を停止せよ ]
                    else
                        Do nothing
                    endif
                endif
            endif
        endif
    endif
endif
```

4.2.1.1.1 故障時異常出力停止機能の初期化を行う

● 初期値
フェールセーフリレーＯＮ出力
４輪のソレノイドＯＦＦ出力
ポンプモータ ＯＦＦ出力
ＡＢＳウォーニングランプＯＦＦ出力

```
Do<フェールセーフリレーＯＮ>を{出力}せよ
Do<４輪のソレノイドＯＦＦ>を{出力}せよ
Do<ポンプモータ ＯＦＦ>を{出力}せよ
Do<ＡＢＳウォーニングランプＯＦＦ>を{出力}せよ
```

4.2.1.1.2 故障時異常出力を停止せよ

```
Do<フェールセーフリレーＯＦＦ>を{出力}せよ
Do<４輪のソレノイドＯＦＦ>を{出力}せよ
Do<ポンプモータ ＯＦＦ>を{出力}せよ
Do<ＡＢＳウォーニングランプＯＮ>を{出力}せよ
```

4.2.2 分析者のコメント

安全要求仕様事例１のみでは、分析者は「意味」を把握できず、
「実装・検証が可能なレベルに詳細化されたもの」とは言えないようです。
これは対象の世界を知らない者には等しくあてはまるもので、
初学者には最初は「意味」を理解するための教科書的文書が必要です。
社内における人事異動に際してもこのような問題が発生します。

ところで、実際の仕様書の場合には、任意の要求項目は、事例１のように他の要求項目と関係づけられて理解されます。
実装され、検証されるに足る要求仕様書は、関連付けられた要求項目の全体を指し、設計書はこれをもとに作成されます。したがって設計者は全体を関連付けて読み解くことが求められます。しかし要求仕様に関する知識が、「全体をくまなく見なければわからない」ではいかにも非効率です。知識が必要な都度参照されるように整えられていることが、今日的な意味での知識に関する効率性と思われます。したがって、この例では「異常出力」に関する技術的な知識がすぐに参照できる知識の構造をなしていることが求められます。

しかし残念ながら、SLPはこのような知識の参照効率の問題（「意味」問題）には応えることはできませんが、知識を階層化することによって、狭められた箇所（else条件）に適切な知識があるように形式化する術を提供しています。これにより、例えばレビューが発散することなく効率的に行うことができます。

4.3 ISO 26262の安全要求仕様事例２の検討

次の例はソフトウェア仕様が数式やタイミングチャートで表現されたものです。
（『機能安全対応のための解説書【ソフトウェア編】（ISO 26262-6:2011）一般社団法人　JASPAR編』のページ23～24より）
コトバで表現された要求は概要として以下のみです。

要求概要
ノイズ等による急激な車輪速度の変化を回避するため車輪速度算出時にフィルタ演算処理を実施する。

しかもこの概要の中には、「ため（に）」という用語があり、これは要求の内容を書いているものではありません。
要求の説明であり、ここでは実際に行うことの意図や理由を述べています。
つまり、行うべきことは、「車輪速度算出時にフィルタ演算処理」を行うことであるが、
その意図・理由は「ノイズ等による急激な車輪速度の変化を回避する」ことである、と述べています。

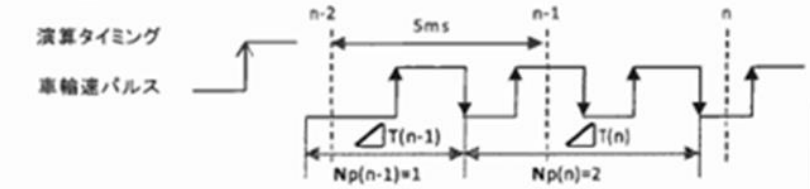
SLPはこのような場合、どのように使用すればよいのでしょうか。
ここでのSLPの問題とその対策は以下です。
① 問題：意図や理由をどう処理すべきか。
対策：SLPの「意図・理由」欄にその旨を書き、具体的な要求内容を論理記述欄に書く。

② 問題：数式はSLPで書き直すのか。
対策：そのまま数式に命名し、SLPの論理記述欄ではそれをメンバー的に扱う。
理由：数式の役割は複雑な現象を数と式というコトバで端的に表現することにあります。
もちろん数も式もコトバですから、自然言語で表現することは可能です。
実際数式の説明は自然言語で行っています。
しかし、さすがにそうだからと言って、仕様に書かれた数式を
ことあらためてSLPで記述することは非効率です。
事例のような演算は現在はシミュレーション・ツールで実現できるでしょう。
そこで、論理記述欄には数式に命名を行い、「Do<〇〇の数式>を{実行}せよ」
という表現でOKです。要求の内容が正しく伝わります。

実際事例の「安全関連要求」に関しては、悪路の場合と通常の場合とで、
使用すべきフィルタリングの数式が異なりますので、
２つの数式にそれぞれ「悪路の場合の数式」と「通常の場合の数式」と命名し
各々を特定します。

③ 問題：図はSLPでどう扱うのか。
対策：仕様の中身に関係のある図であれば、数式のように扱う。
内容説明であれば、論理記述欄に反映しない。
理由：3DCGやゲームで映像を作成する場合には、
図や絵コンテで示されたものが、説明というよりも要求内容そのものです。
このような場合、SLPでは数式の場合と同じように、対象に命名し識別します。

以上を踏まえて、論理記述欄を記述します。（論理記述欄、参照）
---

| | |
|---|---|
| ソフトウェア安全要求 No(ID) | SSR-002 |
| 参照（要求元） | TSR-007 |
| ASIL レベル | C |
| 要求概要 | ノイズ等による急激な車輪速度の変化を回避するため車輪速度算出時にフィルタ演算処理を実施する。 |
| ソフトウェア仕様 | 車輪速度演算<br>Vx＊＊＝Np/T　注）＊＊＝FR, FL, RR, RL<br>  <br><br>車両速度フィルタ演算　※安全関連要求<br>●通常時<br>条件　：悪路状態でない<br>成立時：VWheel＊＊(n)＝(VX＊＊(n)＋VX＊＊(n-1))/2<br>●悪路<br>条件　：悪路状態<br>成立時：VWheel＊＊(n)＝Vwheel＊＊(n-1)<br>【処理フロー／モデル】 |
| ハードウェア I/F・相互作用 | |
| 非安全系 I/F・相互作用 | |
| 制限事項 | |

```
(:「車輪速度算出」はソフトウェア仕様に記述されている「車輪速度演算」に従い行う。
Do<「車輪速度算出」>を{ソフトウェア仕様に記述されている「車輪速度演算」に従い実行}せよ

(:「フィルタ演算処理」はソフトウェア仕様に記述されている「車両速度フィルタ演算」に従い行う。
(:以下の論理が「車両速度フィルタ演算」の条件に相当する。
if <道路>が{悪路状態でない}
    Do<「通常の場合の数式」>を{実行}せよ
else
    Do<「悪路の場合の数式」>を{実行}せよ
endif
```

### 4.3.1 意図・理由も論理記述欄に書いてみる

「要求概要」には「～するため」という目的（意図や理由）を指す表現が使われています。
前の項（親）では、この目的を「意図・理由」欄に書くことを薦めました。
要求の内容には直接関係しないからです。
しかし、「～するため」で指される内容は、要求の内容に重要な影響を与えますから、
あえて「意図・理由」欄に書くことを薦めています。
（例えば、仕様の出し手が「この実装は要求の意図を実現していない。どう実装するかを例で挙げた（だけだから、意図を汲み取って実現してもらいたかった。それがプロの筈。」と言った場合、受け（手はどう反応すべきでしょうか。意図や理由の抽象性（内包性）を受け手は具体的に（外延的に）（詳細化して行く必要があります。

しかし、論理記述欄に意図や理由を書くこともできます。
その場合の記述内容は、抽象性が高いものなので、実装する場合はその内容を技術的に具体化しないといけませんが、読み手になすべきことが伝わります。

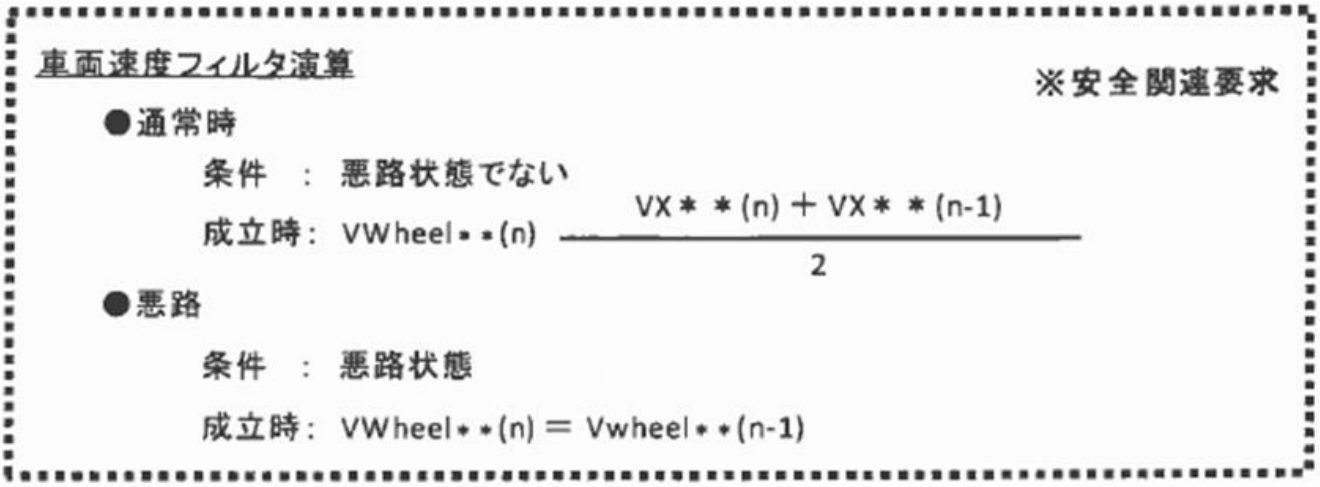
例では、要求概要には
「ノイズ等による急激な車輪速度の変化を回避するため車輪速度算出時にフィルタ演算処理を実施する。」
と書かれてあります。
これをif-else-endifで表現する場合、
if-else-endifの前件と後件の関係は因果的関係で記述しますから（2.1参照）、
「車輪速度算出時にフィルタ演算処理を実施する」ことが
「ノイズ等による急激な車輪速度の変化を回避する」ことをもたらすと形式化します。
また、「車輪速度算出時にフィルタ演算処理を実施する」は、
「もし車輪速度算出を行い、かつその時、フィルタ演算処理を実施するならばと」
と敢えて条件文で記述します。
これにより背反的条件（else）を記述せざるを得ず（形式化）、
このことは「安全関連要求」に関するリスクを明示的に表現することになります。
　　Do<ノイズ等による急激な車輪速度の変化>を{回避させる}
　　Do<ノイズ等による急激な車輪速度の変化>を{￢回避させる}
が、その表現です。

```
if <その自動車>が{車輪速度算出を行う}
    (:「車輪速度算出」はソフトウェア仕様に記述されている「車輪速度演算」に従い行う。
    Do<「車輪速度算出」>を{ソフトウェア仕様に記述されている「車輪速度演算」に従い実行}せよ

    (:「フィルタ演算処理」はソフトウェア仕様に記述されている「車両速度フィルタ演算」に従い行う。
    (:以下の論理が「車両速度フィルタ演算」に相当する。
    if <道路>が{悪路状態でない}
        Do<「通常の場合の数式」>を{実行}せよ
    else
        Do<「悪路の場合の数式」>を{実行}せよ
    endif
    Do<ノイズ等による急激な車輪速度の変化>を{回避させる}
else
    Do<ノイズ等による急激な車輪速度の変化>を{￢回避させる}
endif
```

### 4.4 「意味」を補完する「形式」を

ところで、上掲の例題には、SLPで形式化を行う上で、
よく分からない表現がある、という指摘がありました。
仕様の「意味」が分かっていれば、出てこない指摘だとは思いますが、次の箇所です。




ここには「条件」と「成立時」が書いてあります。
疑問は数式は悪路であるか否かの道路の状態を判断する数式であるのか、それとも悪路であるならばこの数式を実行しろという数式なのか、というものでした。
正解は、悪路であったならば数式のように１つ前の速度を速度とし、悪路でない通常場合であるならば、現在と１つ前の速度との平均を取りなさい、というものです。
しかしいわば稚拙な疑問を投げ掛けた原因は３つあります。
１つは、数式の「意味」がよく分からないこと、
２つ目は、「意味」が分からない人にもわかるように、条件が成立することと数式との関係とが明示的に記述されていなかったこと、
３つ目は、２つ目と関係しますが、悪路であるか否かの条件を成立せしめる事柄が書かれていなかったことです。もっともこの事例はサンプルですから、本物の仕様には書かれると思われますが、このサンプルの範囲のみで仕様を考えた場合には、記述はありませんでしたので、仕様の受け手は迷ったという訳です。

このことは、疑問を指摘した者の「意味」理解力に問題ありとするか、記述の「形式」に問題がありとするか、議論の分かれるところです。
そしてまた、このように「整理」された（あるいは「分類」された）、しかし「意味」の理解力がなければ読み解けない要求仕様書が散見され、「意味」の分からない初学者を悩ませているのも事実のようです。「形式」による補完が望まれるところです。

項１の「結論」の（３）に関わる議論です。