JFP
metaphor Lab.

SLP 開発の発端と
新たなツールの説明

2014 年 3 月 7 日
株式会社ジェーエフピー

目次

＜注記＞
本資料にページ番号の表記はありません。
※slp 文書であれば、特別な設定も必要なく、アウトライン番号をクリックすることで該当の記載箇所に遷移することができます。

SLP開発の発端と新たなツールの説明
ここでは、SLPツール開発の発端と、ツールの全般的な説明をパンフに基づいて行います。

またCQ出版『インターフェース』に当社員がマンガエッセーを載せていた時代があります。

フジワラヒロタツの現場検証
http://www.cqpub.co.jp/interface/column/genba/default.htm

1 問題の発端と改善策

《表－１》は、不具合が起きた際の「なぜなぜ５回」表です。
ほぼ実例に近いものです。
（誤記もありますが、このようなものが現場で通用しているということもあり、そのまま載せます。）
多くの会社はこのような表を作り、不具合の改善を行っていると思われます。

しかし、このようなもので目の見張るほどの改善ができたという話を聞いたことがありません。
何が原因なのでしょうか。

1.1 例《表－１》

《表－１》

現況	Bug作り込み要因(why1)	why2	why3	why4	why5	対策案
コメント8(別項目)の再提起内容に対する対応: Jobキャンセル時、JobCompleteしてから課金ライブラリを呼んでいたため、課金されなかった。強制チケットまとめ時に課金を呼び出してからJobCompleteするようにシーケンスを修正 7／4昼頃送付したが、チケットまとめ場合に実際はまとめクリップをするのに課金しない(NG) 7／5の疑問点:まとめ時に、チケットが1枚だけの場合、まとめクリップせずに終わると、いったん出したエラーが解除される。→F部長から仕様割り切りとすると回答あり。	・チケットまとめモジュール動作の理解不足 ・Jobキャンセル時の動作確認不足。 ・チケットまとめ動作時の考慮が抜けていた ・課金アプリでのチェックが不足していた	・チケットまとめモジュール理解が必要と思っていなかった。 ・チケットまとめは機械的に、APIを呼ぶだけで良いと思っていた。 ・チケットまとめする・しないは判定できなかったので、チケット発行上意味があるか疑念があった。 ・団体ごとのごとの複数部まとめの考慮が抜けたのは、チケットまとめで1枚ものは意味が無いという気持ちがあったから。 ・課金アプリが自課の試験で動かなかった。最近になって動くようになったが、使用方法などを担当者が多忙で聴けなかった。	・通常の個別チケットと異なり、チケットまとめは動作を止められないので、チケットまとめエラーでの検討が不要だと思いこんでしまったから。	・設計時のレビューで、不要との思い込みから、レビュー対象にできなかった。		思い込まずに不要と思っても、レビューの議題にあげる

1.2 改善がはかばかしくない原因

改善がはかばかしくない理由は、
改善策がほんとうの原因に対して行われていないからでしょう。

以下に行う議論の結論は、
不具合の原因は文書（要求仕様等の）が存在しないから、というものです。

では、早速以下論じます。
《表－１》の不具合の本当の原因は何なのでしょうか。

《表－２》

現況	Bug作り込み要因(why1)
コメント8(別項目)の再提起内容に対する対応: Jobキャンセル時、JobCompleteしてから課金ライブラリを呼んでいたため、課金されなかった。強制チケットまとめ時に課金を呼び出してからJobCompleteするようにシーケンスを修正 7／4昼頃送付したが、チケットまとめ場合に実際はまとめクリップをするのに課金しない(NG) 7／5の疑問点:まとめ時に、チケットが1枚だけの場合、まとめクリップせずに終わると、いったん出したエラーが解除される。→F部長から仕様割り切りとすると回答あり。	・チケットまとめモジュール動作の理解不足 ・Jobキャンセル時の動作確認不足。 ・チケットまとめ動作時の考慮が抜けていた ・課金アプリでのチェックが不足していた

《表－２》のwhy1の欄には、現況欄にある課題に対して、
・チケットまとめモジュール動作の理解不足、とか
・チケットまとめ動作時の考慮が抜けていた
とかが、書かれています。
また、
・Jobキャンセル時の動作確認不足。
・課金アプリでのチェックが不足していた
とあります。
両者の違いは何でしょうか。

前者は仕様の内容に関係しそうですし、後者は検査（テストやチェック）の仕方に関係しそうです。
「・チケットまとめモジュール動作の理解不足」は「理解不足」とありますから、
理解を充足させる知識内容が存在していることを指しているからです。
また「・チケットまとめ動作時の考慮が抜けていた」も「考慮」すべき内容がある筈です。
他方「・Jobキャンセル時の動作確認不足。」は動作内容は知っているが、
その動作がその通りであったかを「確認」していなかった、ということですから、
知識の問題ではなく、検査行為の有無を指しています。
「・課金アプリでのチェックが不足していた」は明らかに検査不足を指しています。

しかし、why2を見ると、「意味」があるかどうか分からないということが書かれてあり、
検査の問題とした「・Jobキャンセル時の動作確認不足。」もこの所為で
動作の確認をないがしろにしてしまった、とも取れます。

《表－３》

Bug作り込み要因(why1)	why2
・チケットまとめモジュール動作の理解不足 ・Jobキャンセル時の動作確認不足。 ・チケットまとめ動作時の考慮が抜けていた ・課金アプリでのチェックが不足していた	・チケットまとめモジュール理解が必要と思っていなかった。 ・チケットまとめは機械的に、APIを呼ぶだけで良いと思っていた。 ・チケットまとめする・しないは判定できなかったので、チケット発行上意味があるか疑念があった。 ・団体ごとのごとの複数部まとめの考慮が抜けたのは、チケットまとめで1枚ものは意味が無いという気持ちがあったから。 ・課金アプリが自課の試験で動かなかった。最近になって動くようになったが、使用方法などを担当者が多忙で聴けなかった。

とすれば、知識内容に関する事柄が、不具合の作り込みの原因の多くを占めている、ということが言えます。

もっとも《表－４》のwhy3以降になりますと、「思い込み」という要因により、
知識内容の有無の問題ではなく、知識内容があったにも関わらず、何らかの理由で
本人たちが「思い込ん」でしまったという、いわばあちら（知識）側に問題があったのではなく
こちら（開発）の仕方（取り組み方、チェック体制）に問題があった、と取れます。

《表－４》

why3	why4	why5	対策案
・通常の個別チケットと異なり、チケットまとめは動作を止められないので、チケットまとめエラーでの検討が不要だと思いこんでしまったから。	・設計時のレビューで、不要との思い込みから、レビュー対象にできなかった。		思い込まずに不要と思っても、レビューの議題にあげる

そこで、「思い込み」はどんな思い込みだったのかと問うたところ、
「条件は足りている」と早とちりして、深く掘り下げなかったからということでした。

他方、実際に知識内容（要求仕様等）のものが存在したかということを尋ねたところ、
「しない」とは言い難い、ということでした。
正式にオーソライズされたものは無いが、メモや昔の仕様は存在するとのことでした。

文書の有無に関わらず、内容があるから、条件を、不足ではあったが、設定した、
ということだからです。

ただし、知識があやふやだった、条件の設定が、それ故、甘かった。
あるいは、知識が明文化されたものが無かったので、条件の設定もキチンと行ういう方針が
プロジェクトに徹底されていなかった。
あるいは、条件設定の方法論が明文化されていなかった、
等が言えるかもしれません。

こうなると不具合の原因は、一意に決めることは困難となります。
しかし、どの原因にも共通していることは、文書化の徹底具合にあるような気がします。
文書化がしっかりしていれば、知識のあやふやさも防げたような気がします。
条件の設定もその方法が共通化され、「見える化」でもされているならば、
ひとりの「思い込み」を他の人がカバーできたかもしれません。

冒頭の結論「不具合の原因は文書（要求仕様等の）が存在しないから」はこのように理解したいと思います。

1.3 《改善策》文書を作る

ここでいう文書とは要求仕様文書のことを指します。

《表－１》の文書は要求仕様ではなく、開発途中の課題管理の資料です。
当該のプロジェクトにおいては、あまりきっちりとした要求仕様書がないことが、開発の改善を妨げてるようです。
この課題管理の文書もメモ的な、開発者のみが了解を得れればよいというような、いわば内輪の文書です。
《表－３》をあらためて見ると、
why2とwhy3の項目同士の対応関係も、門外の者にはあまりよく分かりません。
様式がないようです。

このようなことですから、もしかして、要求仕様書もそもそもキチンと書いていないのかもしれません。
決まりや様式も中途半端なまま書いているのかもしれません。

ここでは要求仕様の実物を提示する訳には行きませんので、
要求仕様書は一般的に言われている
「要求項目の漏れや、表現の不統一による誤解」を誘発するような記述になっていないか、
という観点から議論を進めます。

まずかなりできの良いと思われる要求仕様書の検討から始めます。

このような議論を進めるに当たっては、
要求仕様が正しく書かれていないと後続の設計はうまく行くはずはない、という経験則に基づいています。

しかしと同時に、要求仕様やまた設計仕様書も無くてもうまく行っていた時代があったとか、
あるいは今ではかなりの高級言語や、アプリケーション作成ツールなどが出来、
この手の文書無しでもＯＫということがあることも認識した上での議論とお考え下さい。

そして更にその上で、こんにちのソフトウェア規模の増大は、文書なしというのは無理、
文書を正確にしないと、後続工程での開発モジュールの組み合わせで混乱が増大する、

という単純な理屈と経験に基づいて、文書化の必要性を主張しています。
そう、ご理解ください。

1.3.1 良いと思われる仕様書でも問題がある

良いと思われる仕様書でも問題があります。
ましてや、、、
というようなことは、よくある話です。

よくありすぎて、開発現場は慣れっこになっているかもしれません。

それではいけないということで、要求仕様を記述するツールを作りました。

2 新しいツール：SLPパンフの説明

本項以降はすべて、ツールSLPパンフレットの補足的な説明を行っています。

実際のパンフレットは別途サイトからダウンロード可能です。

2.1 ツール諸元

ツールSLPの名称等、諸元について書いています。

2.1.1 名称

要求仕様記述ツール

SLP

SPEC L-PERFECT™
スペックLパーフェクト

SLPの正式名称は"SPEC L-PERFECT"ですが、この頃では"SLP"という略称が多く使われるようになっています。

2.1.2 概要

本ソフトウェアは要求仕様を
簡単な文法で正確に

記述するツールです

SLPは記述のための簡単な文法を持っています。

2.1.3 方法

Logically

『主語述語論理モデル』

SLPは論理的であることを基本にしています。
論理学の簡単な規則に従っています。
「形式」的です。
また文を主語（または目的語）と述語で記述し、
それらの２つがペアで存在するものを「意味」と呼んでいます。
SLPは片方だけでは［意味］を形成しないとしています。

2.1.4 実現目標

要求仕様書が論理整合性と一意性を持ちます

要求から実装まで要求のトレーサビリティを実現しています

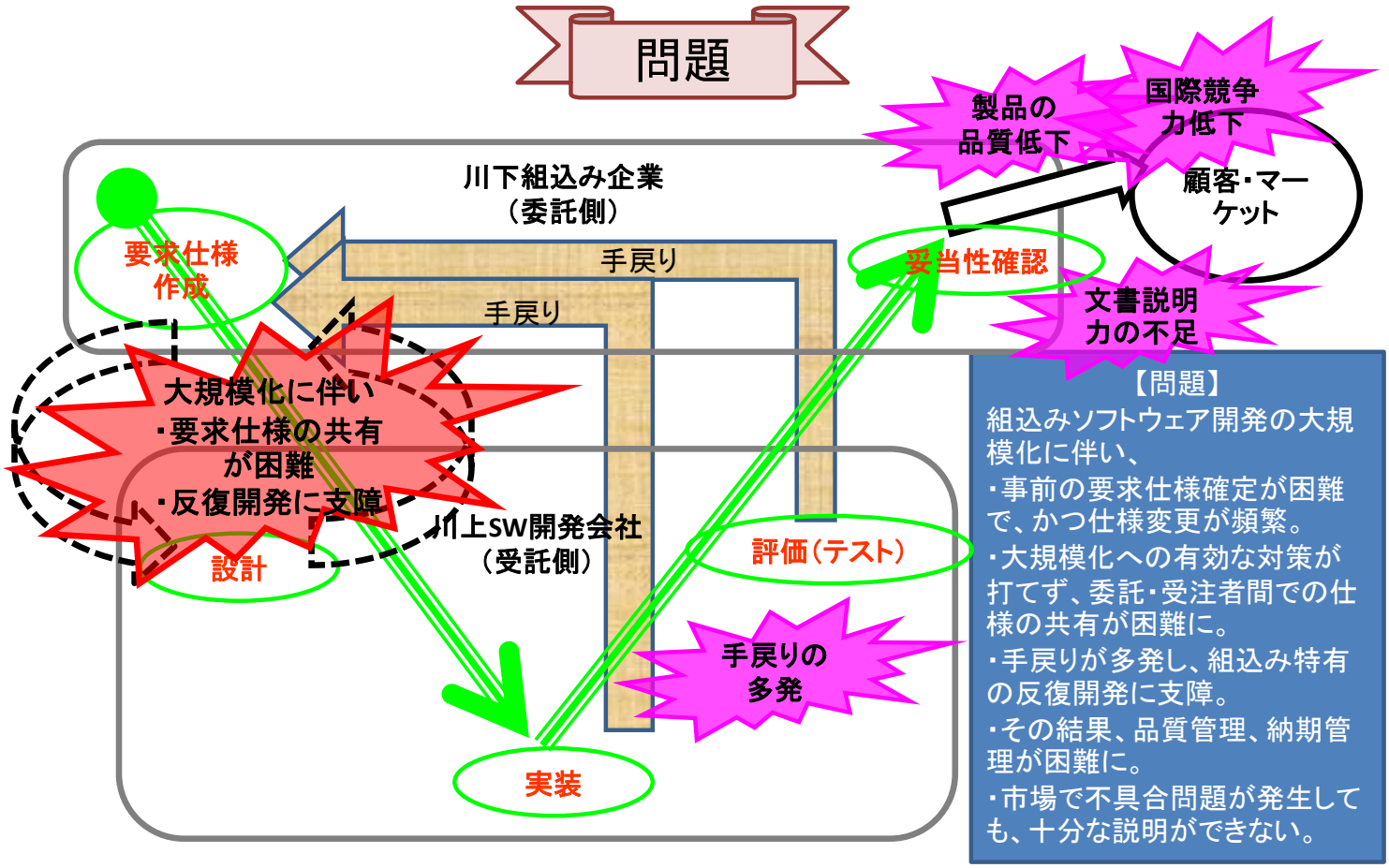
操作と運用に高いユーザビリティを発揮します

「意味」が矛盾しないようにします。
「矛盾」とは同時に「AはBであり、かつAはBでない」と主張する場合です。
文書の中にこのような記述があった場合、ノーと指摘します。
「一意性」とは、用語がいろいろにぶれないことをいい、用語がぶれないために用語の類似性などの機能を提供しています。
「トレーサビリティ」とは、要求項目がSLP文書に反映されることを指し、そのための方法を提供しています。
「高いユーザビリティ」とは、記述の構文が選択画面に適宜表示されるなどの操作性と
記法の統一による文書の均質化などの運用面での使いやすさを指しています。

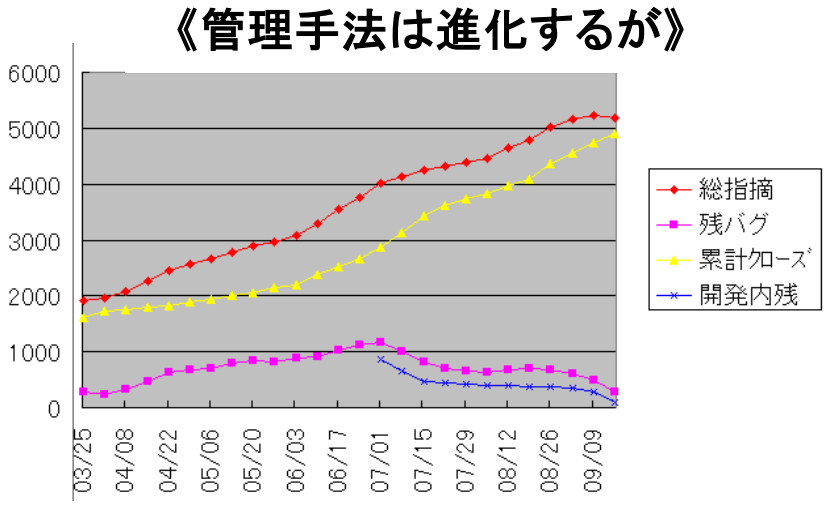
2.2 開発プロセスの問題、管理手法、改善の姿

以下、開発プロセス関して、問題、管理手法、改善のイメージについて述べます。

2.2.1 開発プロセスでの問題

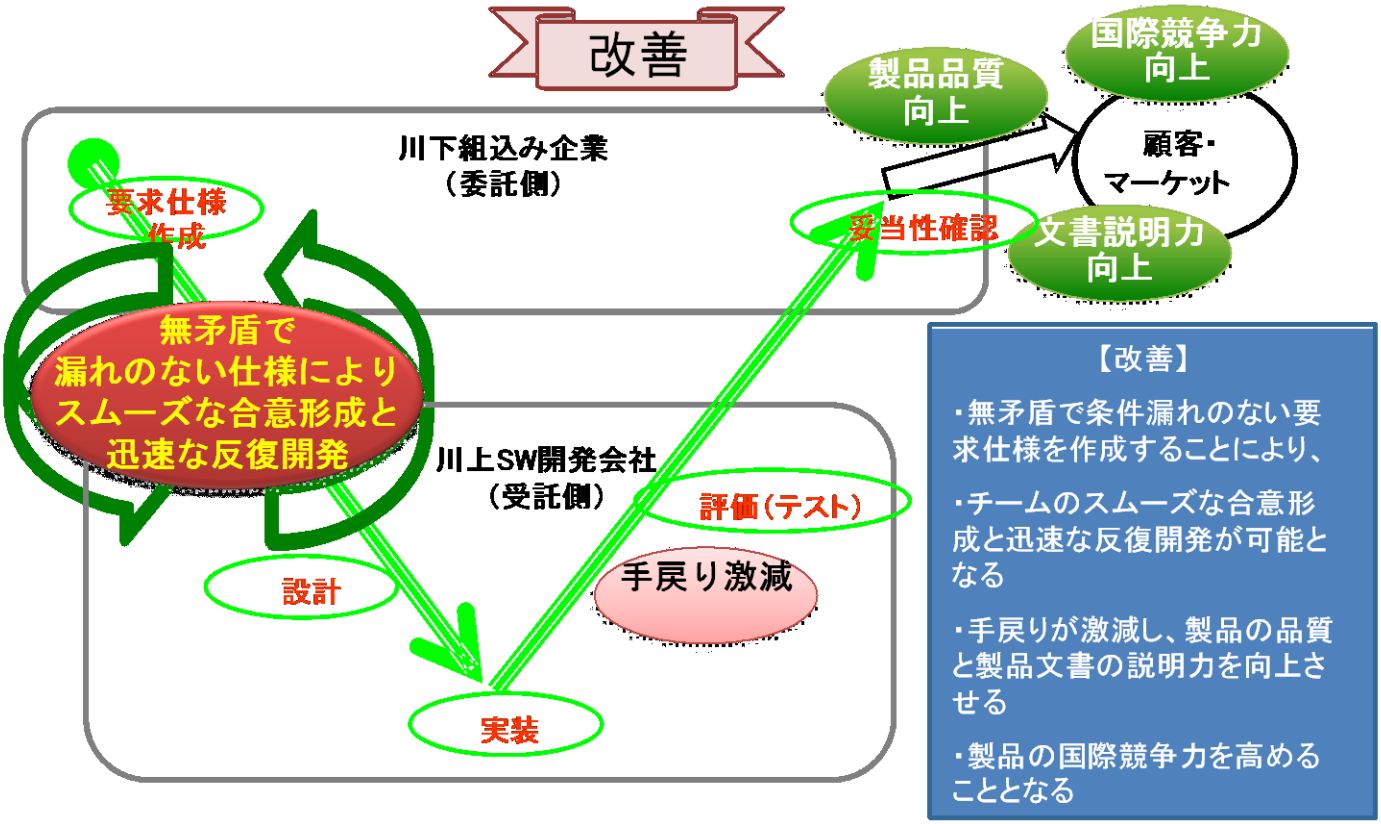


2.2.2 管理手法は進化するが、、、。



管理手法ばかりが進化しているような気がします。

2.2.3 開発プロセスを改善するとこうなる



2.3 ツールSLPはこのようなもの

SLPの基本的考え方や記述の構文について述べ、また実際の記述の例を示します。

2.3.1 《SLPの基本的な考え方》

《SLPの基本的な考え方》

ソフトウェア要求仕様書はプログラムが実現すべき内容からなる文の集まりです。
文は内容の説明文も含みますが、
最終的には実現すべきことが
「～ならば、～せよ、そうでないならば、～せよ」という条件付き命令文と、
「～せよ」だけの無条件命令文からなります。
「～」は文で、１つの主語や目的語（対象物）と１つの述語（対象物の状態）からなります（単位文）。
「ならば」と「そうでないならば」を条件接続詞と呼ぶこととします。
すると、要求仕様書は単位文と条件接続詞で記述されるといえます。
これがSLPの基本前提です。

2.3.2 《構文》

《SLP構文》

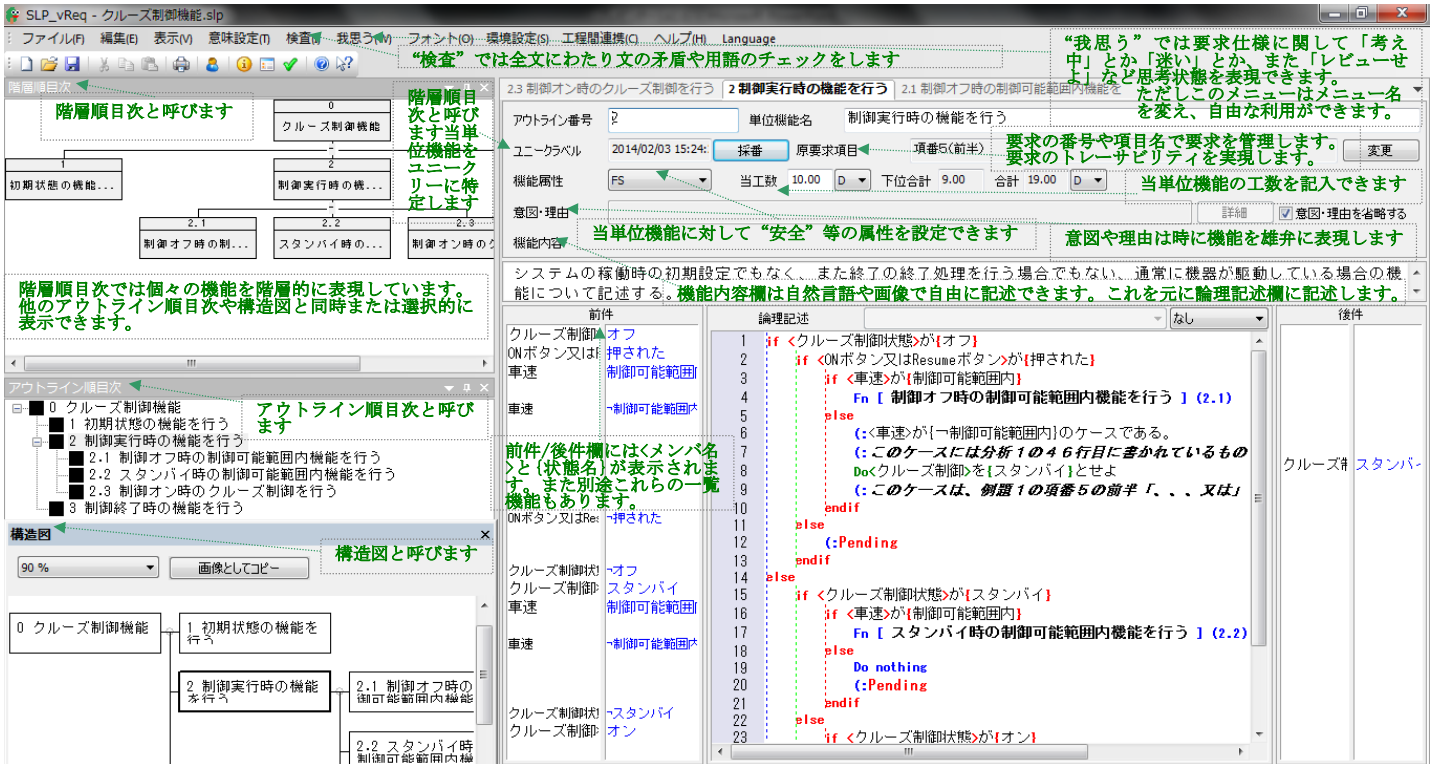
１．２つの基本構文と３つの派生構文と、２つの参考文からなります。これらをSLP構文と呼びます。
基本構文は、例示の**Do文**と**if文**です。

派生構文は基本構文を基に、単位機能を表す**Fn文**、if文の派生としての**switch文**、繰り返しの**loop文**です。
参考文はコメント文と改行文です。

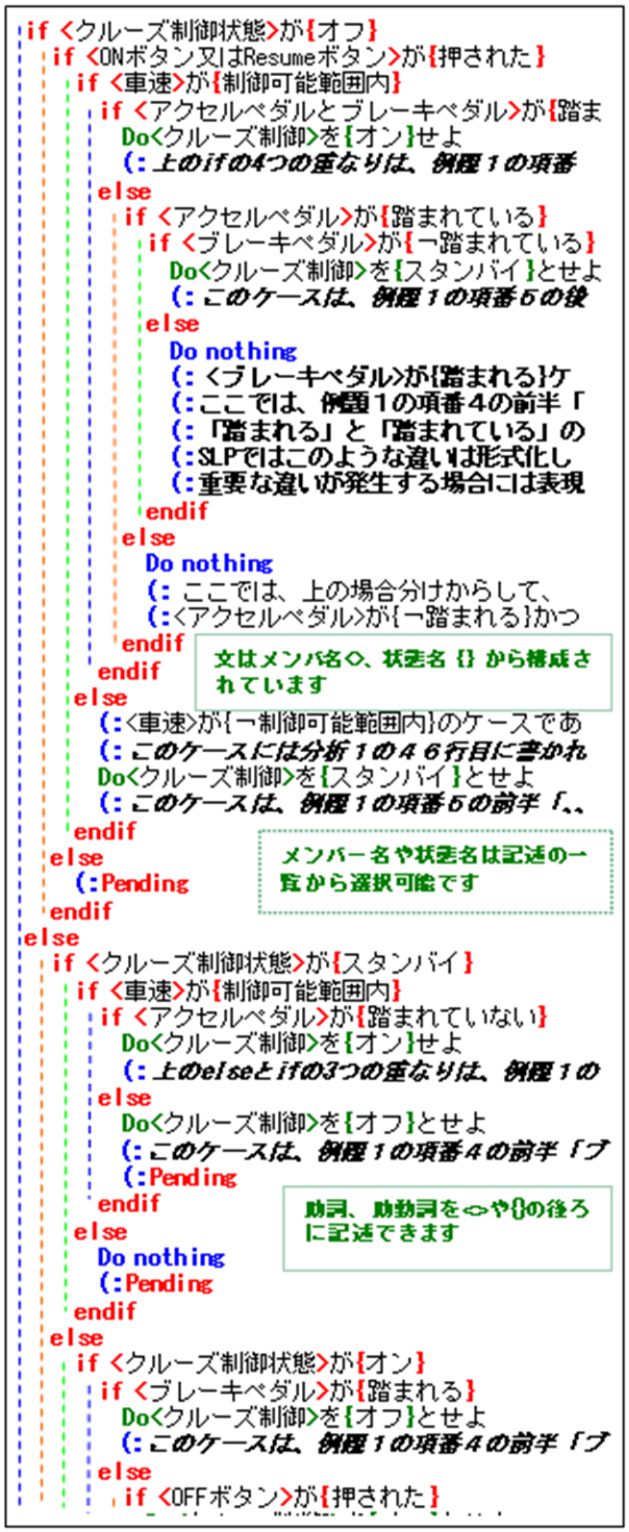
２．単位文はSLPを記述するための基本単位となります。
単位文は「<対象物>は{ある状態を持つ}」という陳述形式や、
「<対象物>を{ある状態にせよ}」という命令形式で表現されます。
対象物の名前を「**メンバー名**」と呼ぶこととします。
このメンバー名は単位文の主語や目的語に相当します。
対象の状態を「**状態名**」で表します。
状態名は陳述述語や命令述語に相当します。
たとえば「実行ボタンがONであるならば」は「if <実行ボタン>が{ON}である」と記述します。
“if”に対しては必ず“else”が付帯され、if-else-endifの構造を持ちます。入れ子の構造です。
メンバー名の状態を設定（命令）する時には、「Do<実行ボタン>を｛OFF｝せよ」と記述します。

３．要求仕様内容は**階層構造**的に表現できます。大きな文書も内容の親子・兄弟関係が明確にされます。
階層構造化される単位を**単位機能**と呼び、SLP構文で論理的に記述されます。
これにより文が矛盾無く記述されているか、また用語が文書全体の中で統一されているかを検査できます。
このことは、製品企画、要求分析者、開発者、テスト者等の思い違いを激減させます。
納期や開発予算を最適化します。

2.3.3 記述例《階層化したもの》



2.3.3.1 記述例《階層化しないもの》



この個所が
別途単位機能として
階層化される

この個所が
別途単位機能として
階層化される

2.3.3.2 仕様書例　（“SCADE”のテキストより）

記述例は下記の例題です。

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時

・ **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

2.3.4 SLPの効果

自然言語で書かれた要求仕様書をSLPで記述することによって、
要求仕様書は**論理整合性**（論理無矛盾、条件網羅性）と**一意性（用語統一、意味明確）**を獲得することができます。

記述はSLP構文で**単位文**を書くことで行います。また複数の単位文をまとめ、**単位機能**として**階層構造化**できます。
記述は構文のメニュー選択で行います。次に記述可能な構文をガイドします。
さらに以前に書いたメンバー名や状態名を選択するなど、ユーザビリティを高めています。
SLP構文はif-else-endif構文により条件網羅性と**形式無矛盾**を保証します。
用語の登録により、**意味矛盾**も検査されます。これらにより文書の**論理無矛盾（形式無矛盾、意味無矛盾）**を保証します。
SLPでは単位文の指す意味を**単位意味**と呼びます。**レビュー**も単位文ごとにできます。
その場合、SLPでのレビューは単位意味ごとに仕様を確認することになりますので、
レビューにありがちな細かな確認ミスを防止できます。

仕様変更は開発にはつきものです。仕様変更とは、SLP的にいえば
「単位文の用語を変えること、あるいは単位文同士の接続関係を変えること」といえます。
SLPでは各々の単位文は仕様書全体に渡って関連づけられていますから、
用語や接続関係を変更することによる影響範囲も分かります。
変更を確実に効率的に行うことができます。

仕様変更の**差分**管理も行うことができます。
if-else-endif構造を**決定表**に変換し、条件の一覧性を提供します。また**テストの仕様原案**を提供します。
SLP構文で記述した文書を**通常文**に逆変換し、
条件の網羅性や用語の統一を自然言語に近い表現で行うことができます。
要求の識別子を指定することで、要求の**トレーサビリティ**を実現できます。

2.4 クルーズ制御の例を実際にやってみる

実際にお試しください。

例(項番2.3.3.2)はよく書かれた仕様です。しかしSLPで要求分析をしてみると、仕様には書かれていないことも知らないと記述できない所があることがわかります。このことは、書き手は仕様の意味をすべて知らなければならないことを物語っています。

しかし、すべてを知らないからといって臆することはありません。SLPは「形式」と「意味」の違いを明確にした上で、「意味」にアプローチする方法を提供しています。このことは仕様の受け手は「形式」さえ確かであれば、少なくとも最初のうちだけ新たな領域の知識を持たなくても許されるという根拠を示しているように思われます。

ファイル『要求論：要求分析、SLPの使い方、国際標準規格の場合.slp』に
ツールを実際に使ってみた事例が載っています。
（当社サイトより、ダウンロードできます。）
ここでは「形式」や「意味」のこと、
要求のトレーサビリティのことについて書き、
また最後に機能安全ISO 26262の要求仕様の記述に関して、問題提起をしています。