TO5：影響箇所の気付き

メンバー：　小澤、白川、南、清水、森島

1．これまでの活動報告

【WG発足の目的】

派生開発の難しいところは、変更に対して仕様上で関連する箇所と、予想外のところで影響する箇所があることである。特に変更に関連して影響する箇所の見落としが大きな問題になる。もっと効果的に影響範囲、影響箇所を気づく方法は？

【WG活動報告】 2011年～2012年

①課題検討　→　対策の立案

課題1)人・現物の質に依存した影響分析

仕様の視点、関数視点、データ視点、機能視点・・

↳ 効率化と精度向上からツール利用で影響箇所を絞り込む

↳ 市販の分析ツール等を利用したTMの作成を提案

課題2)対象の可読性が悪い（ファイルが多い場合に一枚の紙面では見切れない）

モレに気づいて貰えるTMの見せ方を検討

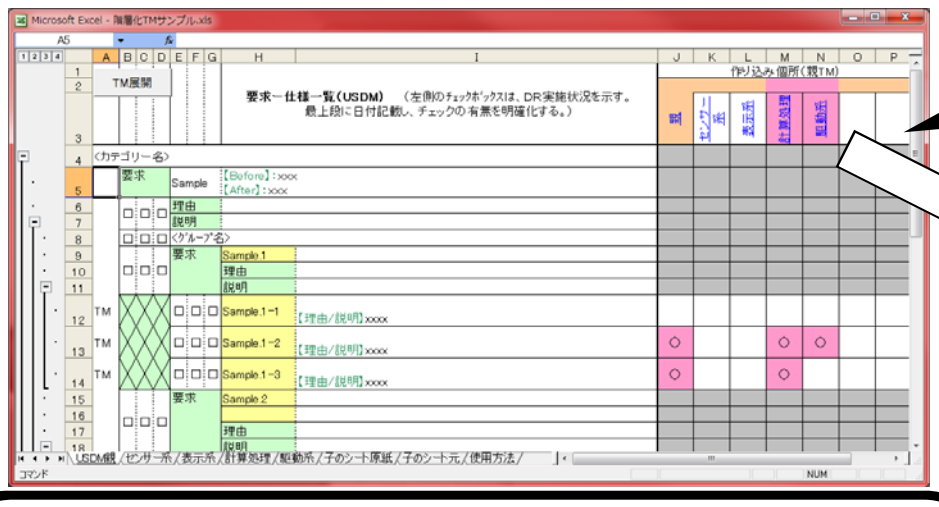
↳ 階層化TMの提案

②対策の試行

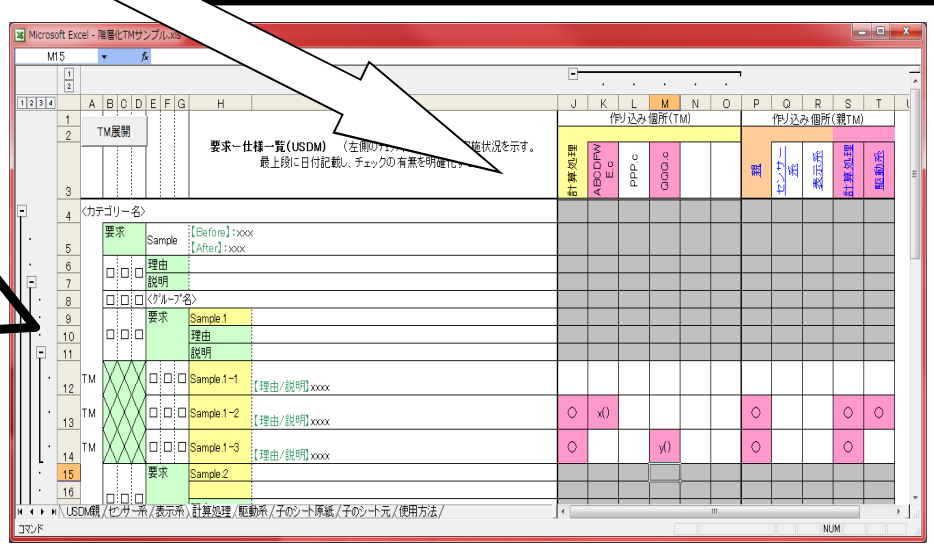
・階層化TMの試行

研究会メンバー各社で実際のプロジェクトに適用し、効果の確認と、新たな課題の抽出を実施

1）階層化TMの書式



【親シート】親シートにはどの様な子がいるかを明確化し関係する箇所のセルは色付けされます。子は各シートに分れ表現されます。



【子シート】親シートの“計算処理”のシートの例です。“計算処理”の階層にある各ファイルが展開され親シートと同様に関係するセルが色付けされます。

2）試行結果

【使い勝手から（課題）】

1）Excelのシート数（30枚以上：仮想メモリが関係？）が増えた場合、作業性が大きく低下する。

2）USDM作成段階で、TMの上位階層を作成・検討しているが、その作業性が悪い。

3）開発の規模によっては、階層化しないほうがTMの見通しが良い。

【影響箇所への気づき（課題）】

1）見通しは良くなったが、影響箇所に気づくには人に依存する部分が残る

２．活動の振り返り→新たな課題の創出→提案

活動の振り返り

USDM+TM

変更設計書

3点セットだけでは、
影響範囲が漏れている！！

ほかの生産物は？

SpecOut

でも

人により区々！！

作り方の決まりもない！！

今のＳｐｅｃＯｕｔの問題点
・調査は個人任せ　　・生成物もバラバラ　　・調査の範囲が不明確
だから
人により影響範囲が見えたり、見えなかったりしている。

新たな課題の創出

提案：
属人生を無くす影響範囲の調査方法を定義。

（誰もが同じように影響範囲の調査ができる。）

提案

USDM：仕様変更パターンのマトリックス

			変更パターン			
			A	B	C	D
要求1						
	要求1．1					
		仕様1.1.1		○		
		仕様1.1.2	○			
	要求1．2					
		仕様1.2.1			○	

1つの仕様に対して変更パターンが決まってくる！！

変更パターン毎の調査方法（SpecOut作成ガイド）

変更パターン	調査方法					
	DFD	STD	シーケンス図	コラボレーション図	・	・
A	○					
B			○			
C		○				
D				○		

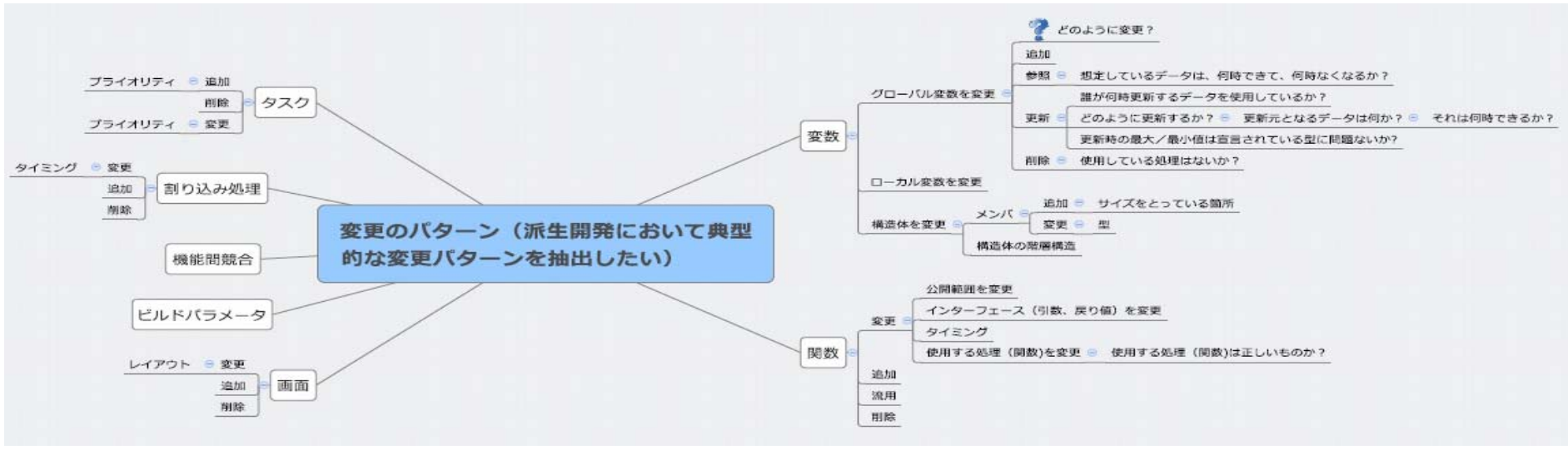
3．本年度の取り組み内容

【変更パターンによる気づき】

前年度までの活動、階層化TMを適用していく中で、
変更によって受ける影響は変更タイプ毎に共通性があるのでは！？　という気づきがあった。

調査：

・考えられる変更タイプを抽出



・変更タイプ毎に影響範囲の気づきを得る為の調査対象・範囲を確認
・影響に気づきやすくする調査資料の表現方法を検討
・変更タイプ毎の調査対象と調査資料のマトリックス表を作成

変更タイプ		(出来る限り具体的)どんな変更か	何を調査すれば良いか？	その表現方法は？	注意点など	アクティビティー図	通信処理手段	ER図	フローチャート	ブロック図	SCM(モジュール構造図)	使用箇所確認・検証(Grep等)	メモリマップ	ソースのDiff	シーケンス図	タイミングチャート	DRBFM	事例1	事例2
変数・定数																			
	追加	1)グローバル変数の追加(関数内に閉じた自動変数は含まない)	1)変数確保領域の空き状況。 2)ローケートのされ方。 （空き領域の使用方法。）	1)メモリマップ。 2)ヘッダーファイルのDiff結果。	2)は、組み込み系の場合、データの追加により 他のデータアドレスに影響が出るなどあるので、要注意。								○	○					
	変更																		
	型	1)データの受け持つ領域の変化によるサイズの変更をした。	1)変数確保領域の空き状況。 2)ローケートのされ方。 （空き領域の使用方法。）	1)メモリマップ。 2)ヘッダーファイルのDiff結果。	2)は、組み込み系の場合、データの追加により 他のデータアドレスに影響が出るなどあるので、要注意。								○	○					
	サイズ	1)配列における添字の数の変更をした。	1)変数確保領域の空き状況。 2)ローケートのされ方。 （空き領域の使用方法。） 3)配列の名前による使用箇所の検索結果	1)メモリマップ。 2)ヘッダーファイルのDiff結果。 3)添字が変化したことによる処理の影響。 たとえば、添字数をループで使用している場合など。(Grep:影響のないことの説明)								○	○	○					
	中身	1)作られるデータの作り方の変更	※データの持っている意味合いによって調査内容は変化する。 1)制御を示す変数: 2)データを示す変数:			○	○				○	○			○	○			
			作成箇所を検索する	grepで検索する				○		○	○	○							
	公開範囲（スコープの変更）																		
	定数値	1)定数値を変更した。(defineなど)	1)定数値使用箇所の検索結果。	1)すべての使用箇所において影響がないことを確認し、表現する。(Grep:影響のないことの説明)								○							
	削除	1)変数を削除した。	1)変数使用箇所の検索結果。	1)削除した変数が各使用箇所で影響のないことを確認する。(Grep:影響のないことの説明)	1)変数の削除は単に変数を削除することは少なく、機能の削除に伴う変数の削除が主になると考えられる。 ゆえに、機能削除による処理削除範囲内でしか、変数を使用していないことを確認すればよい。							○							
関数																			
	追加	1)新規関数の追加	1)追加箇所の妥当性																
	変更																		
	インターフェース	1)引数を追加。	1)上位関数の一覧。 2)追加された引数は、そこからのコールで期待したデータが受け取	1)関数の呼び出し元ツリー(コーラツリー)							○								
		2)リターンバリューの内容の変更	1)コールツリーからの戻り値の使用方法の確認。	1)呼び出し元から各コールに問題ないことのエビデンスを残す。							○								
	処理(ロジック)	1)関数内部の処理を変更する(条件分岐やアルゴリズムの変更、演算の追加等々)	1)似たようなロジックを用いている処理は他にないかを確認する。	1)CCFinderなど利用できないか？(CCFinder)							○							● どこまで調査すれば気づけた？	
	処理(ロジック)	処理の行数が増える変更 処理の時間が大きく変わる変更 変化点がタスクを跨ぐ場合の変更	処理時間。CPUの占有時間	並行タスクの一覧 各タスクの制限、動作処理の内容											○	○			●
	処理(ロジック)	ロジックの追加	追加箇所の妥当性 追加箇所以降の処理の実行条件に変化はないかを調べる(調べるネストの深さは？)	変化点と変更点の一覧	変更箇所から変化点を見極め、その処理を影響ないか調べる				○										
	削除		1)削除関数のコーラツリー。 2)削除された関数の関数外部に対する出力データを他のどこで使	2)使用箇所全てにおいて、出力結果のその後の使用のされ方を調査し影響がないことを説明す															

・不具合事例で、上記分類の検証を実施
・変更タイプ毎に調査対象・範囲に共通性があることが分かった！

今後の取り組み：

・変更タイプによる気づきの手順化、提案
・実際の開発テーマで適用し、効果の立証

シートNo.3