「スペックアウト」で
ソースコードを理解しよう

派生開発推進協議会
関西部会

2020-09-12

派生開発推進協議会（AFFORDD）の紹介

派生開発を成功させる方法を手に入れて、普及していくために、
研究会・地方部会でのグループ活動を行なっています。

（ホームページ）http://affordd.jp

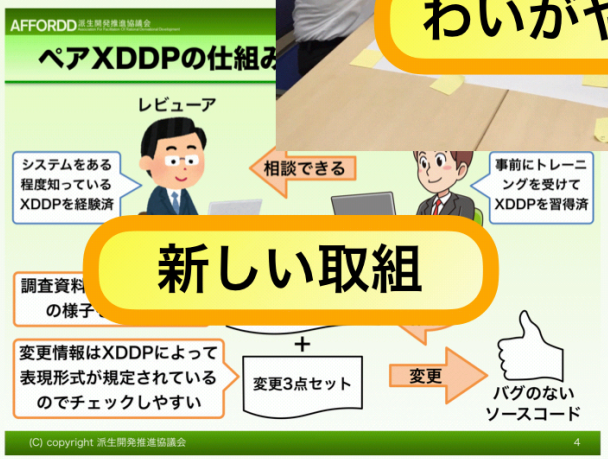
＋ 関西部会
ながの部会
（中部部会）

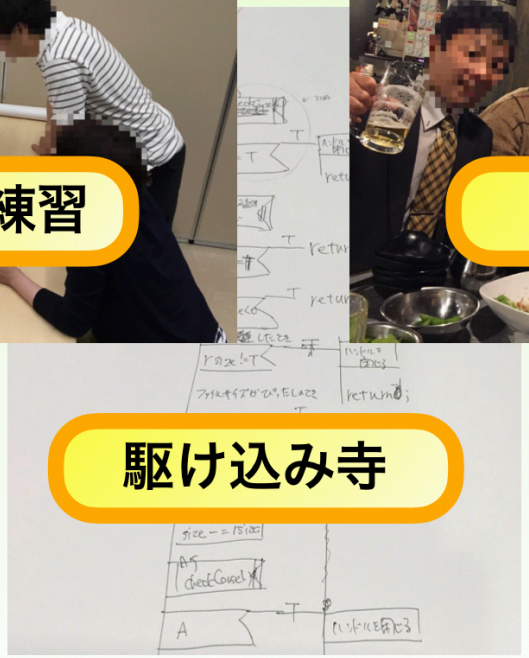
AFFORDD関西部会の紹介

派生開発推進協議会メンバーの関西地方の集まりです

基本的に月に1回、土曜日午後に自主勉強会をしています

見学・参加希望の方はメンバーまでご連絡ください

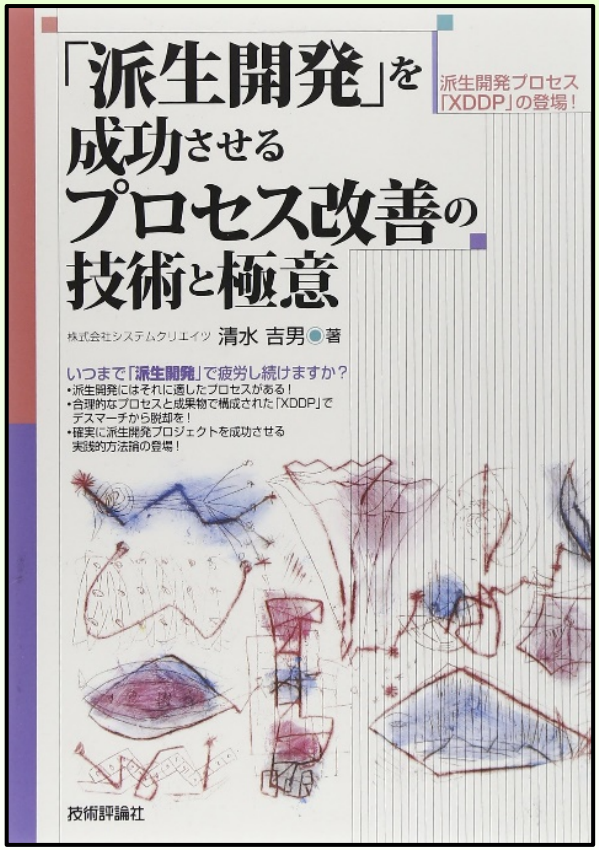




目次

スペックアウトとは

設計書の不足を補うために、ソースコードから必要な設計書を起こす行為を「スペックアウト」と呼ぶ

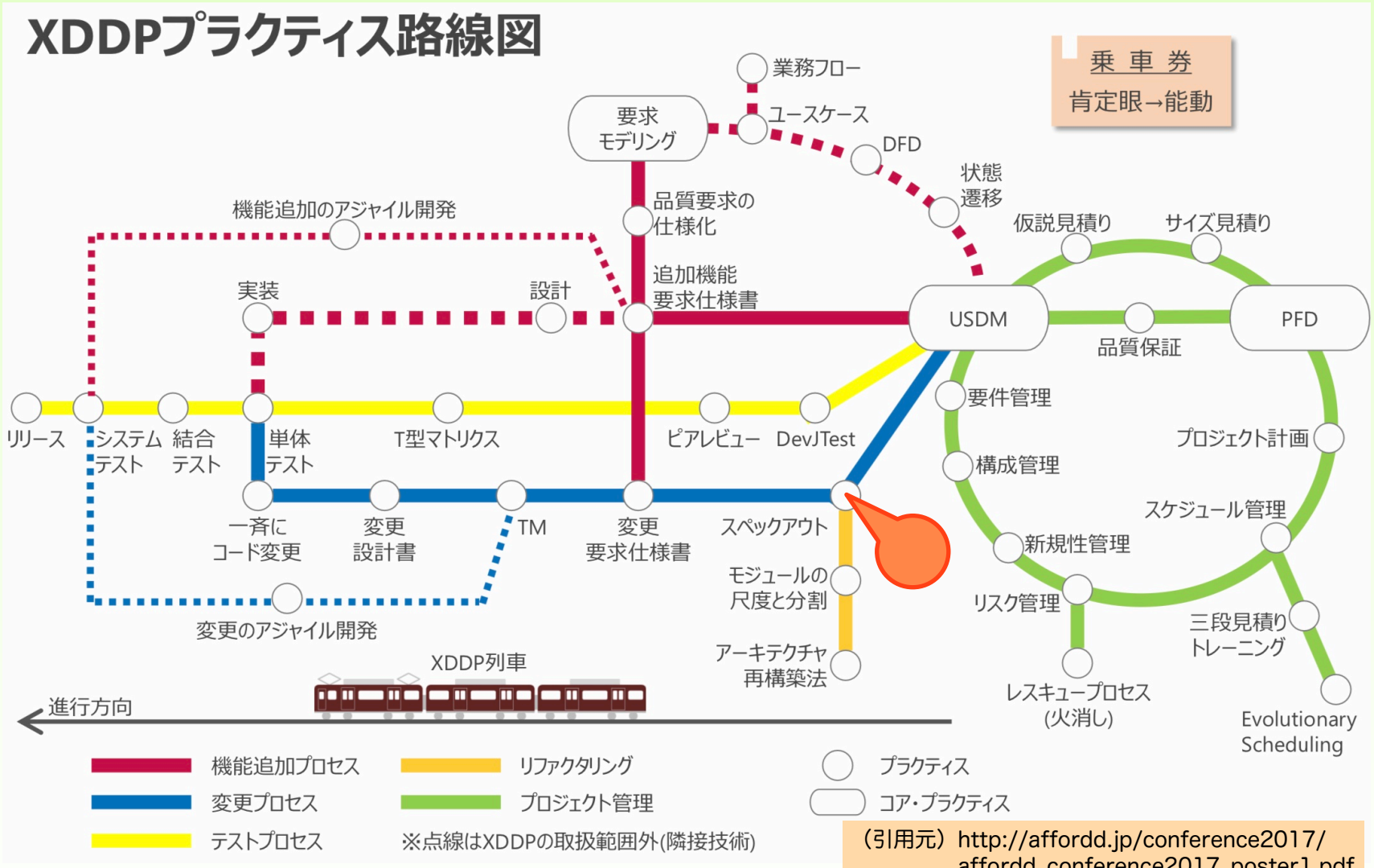


「スペックアウト」は、この書籍に
紹介されているXDDPにおける用語です

派生開発における使い方や注意点は
この書籍や関連資料を参照ください

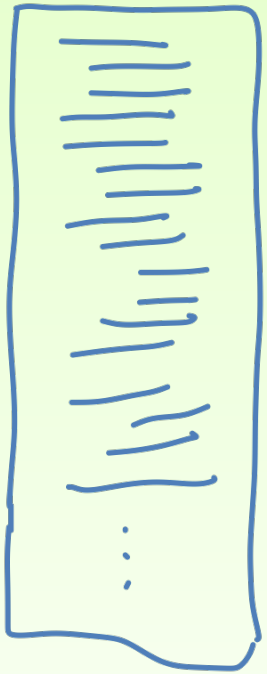
ここでは、スペックアウトの考え方と
手順・成果物のサンプルを紹介します

XDDPにおける位置付け

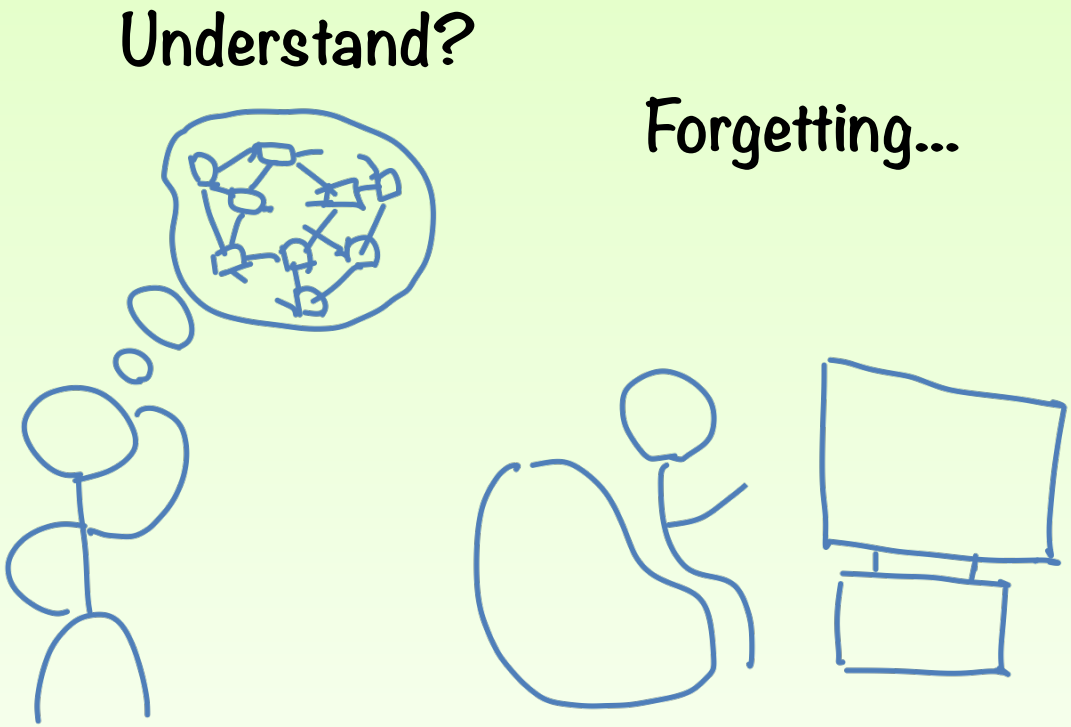


（引用元）http://affordd.jp/conference2017/affordd_conference2017_poster1.pdf

スペックアウトが無いとき

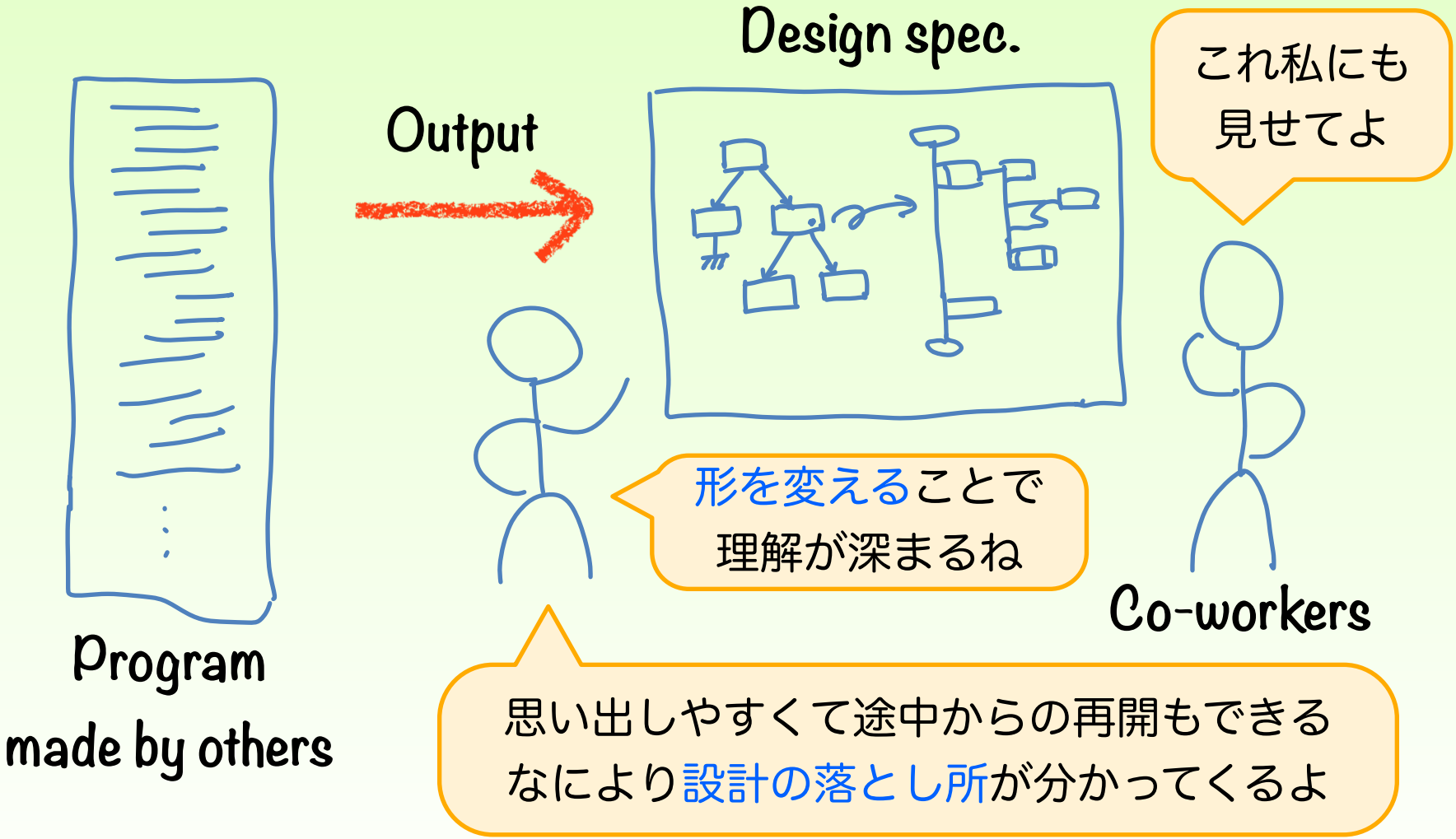


Program
made by others



どこまで理解できたのか判断できないし
すぐに忘れてしまうよね…

スペックアウトが有るとき



一覧とデータ参照の表現

一覧表

資料名	内容	今回の用途

ソースファイル	含まれる機能

関数名	説明

データ名	説明

データ参照表

データ名	f1	f2	f3	f4	f5
d1	●	●	●		●
d2	●		●	●	
d3	●			●	

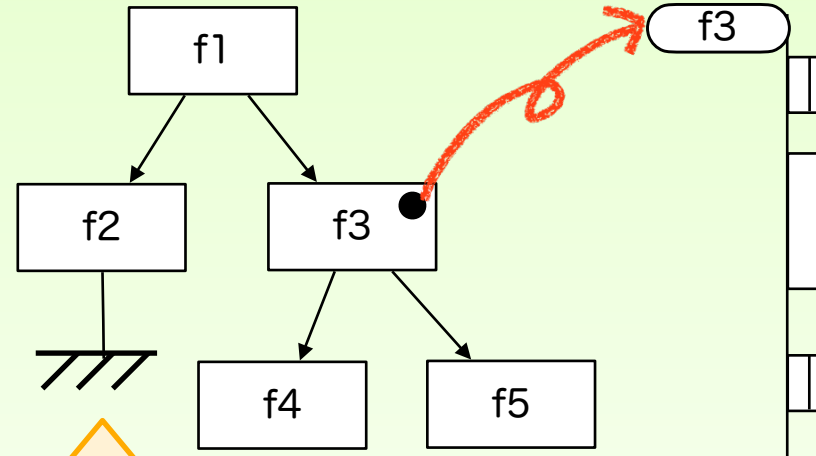
まず一覧にしてみる

参照関係を把握する

処理構造の表現

構造図

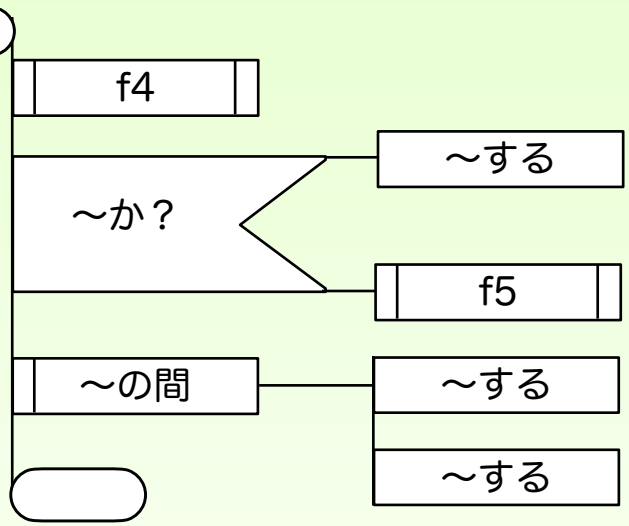
呼び出し構造を把握する



調査済を示す

PAD

処理手順を把握する



これらの他に、ER図、クラス図、状態遷移表、
AFD、シーケンス図など、必要に応じて活用

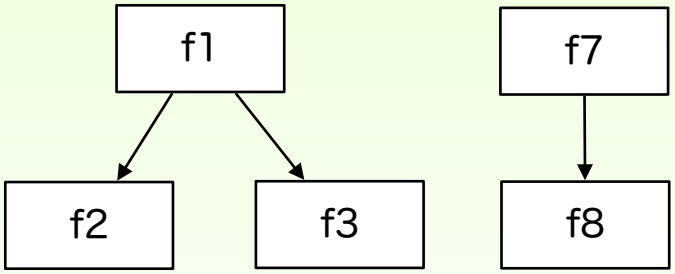
目的に応じてやり方を変える

スペックアウトの目的を明確にしてから取りかかる

知りたい機能は？　　知りたいことは？

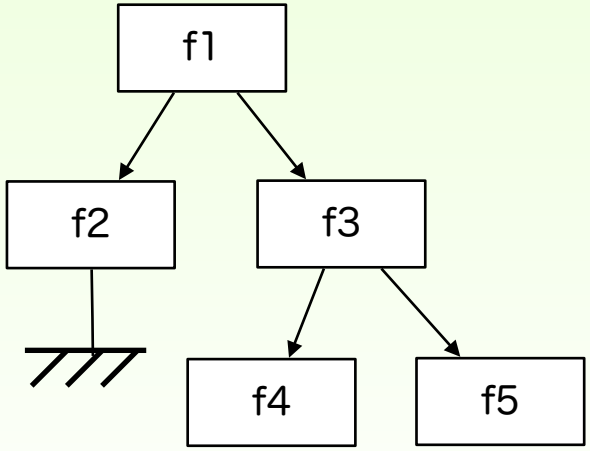
必要な調査資料は？　**投入できる時間**は？

事前調査が目的なら
広く浅く



「スペックアウト探検隊」
にならないこと

変更箇所を探すことが目的なら
ポイントを絞って深く



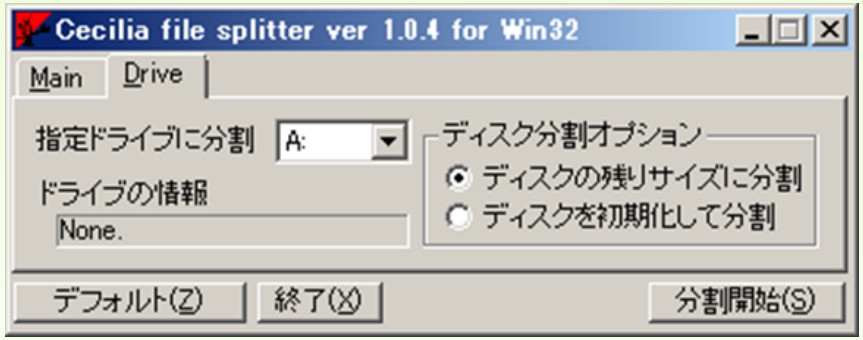
ケース・スタディ

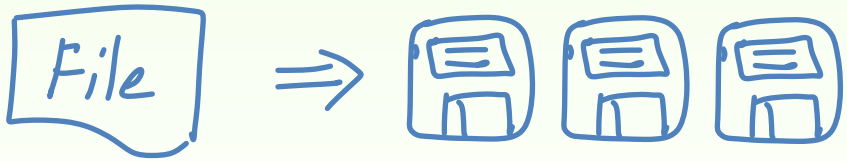
ソースコードしかない状況で仕様を把握してテストをしたい

→スペックアウトを活用できるかも？

ファイル分割ソフトを対象に手順・成果物のサンプルを紹介します

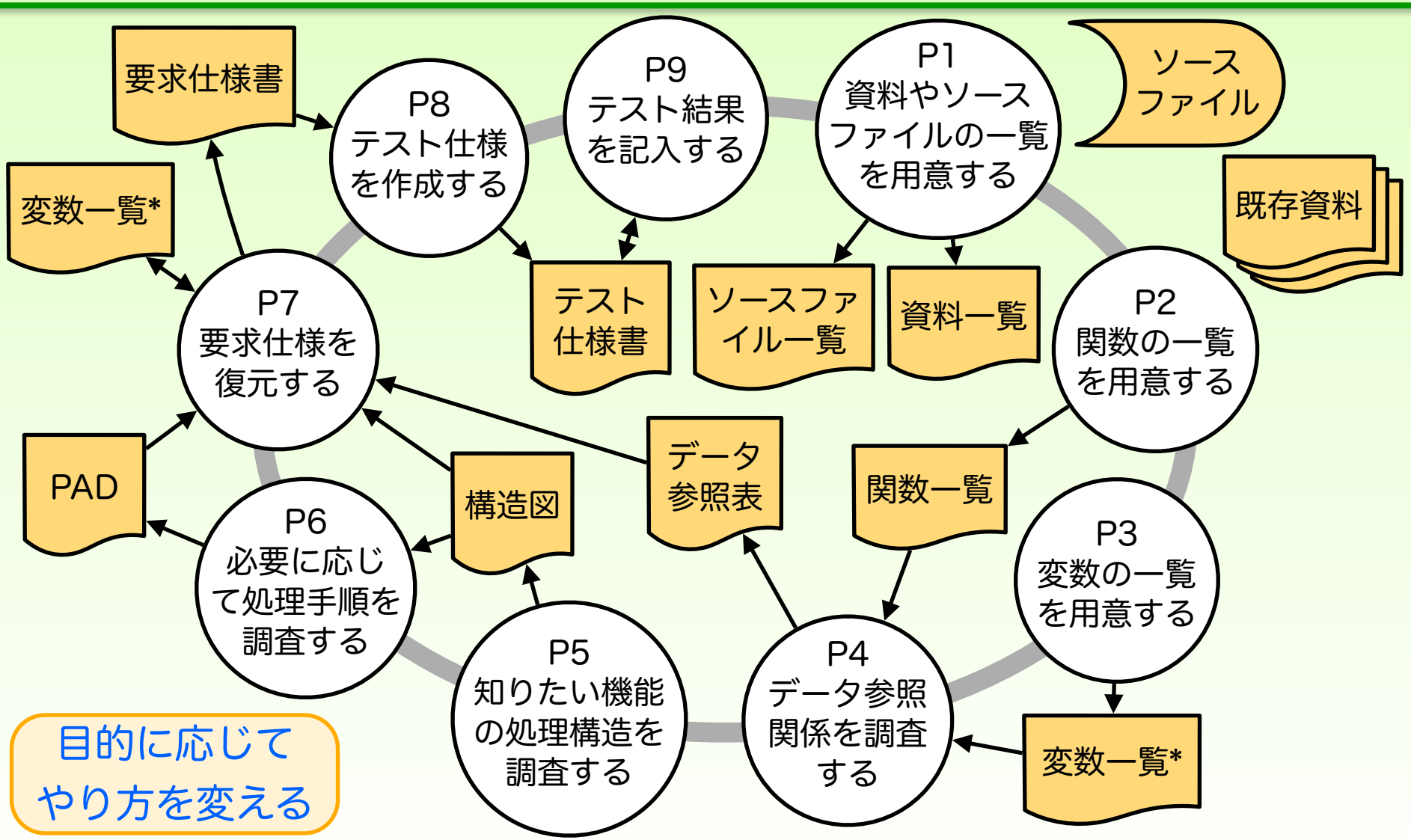
後で資料を参照しながら作業できるように詳細に書いていますが
まずはざっとイメージを掴んで頂ければと思います



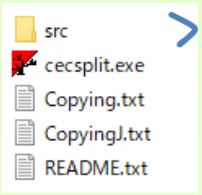


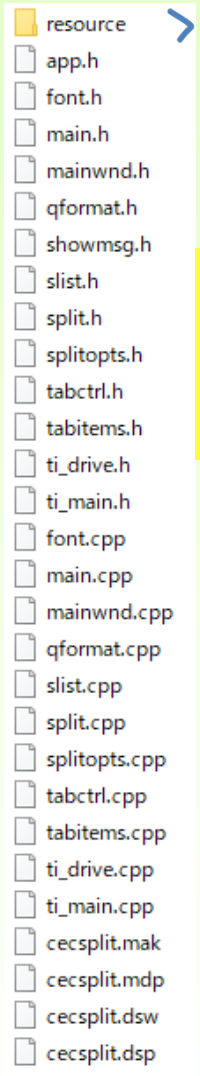
（コードの所在）https://sites.google.com/site/yikemac/cecsplit_1.0.4r1_win32.zip

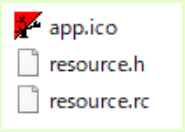
スペックアウト手順の例



P1 資料やソースファイルの一覧を用意する







dirコマンド等で
一覧を出力して
Excelで整形する

資料名	内容	今回の用途
Copying.txt	GPL2のテキスト	ライセンスの確認
CopyingJ.txt	GPL2の日本語訳テキスト	
README.txt	機能紹介など	機能概要の把握

ソースファイル名	ヘッダファイル名	含まれる機能
main.cpp	main.h	アプリケーションのエントリポイント
mainwnd.cpp	mainwnd.h	メインウインドウの表示
split.cpp	split.h	ファイル分割機能
splitopts.cpp	splitopts.h	ファイル分割オプションの管理
qformat.cpp	qformat.h	クイックフォーマット機能
tabctrl.cpp	tabctrl.h	タブコントロールの管理
tabitems.cpp	tabitems.h	タブアイテムの管理
ti_drive.cpp	ti_drive.h	タブアイテム「ドライブ」の機能
ti_main.cpp	ti_main.h	タブアイテム「メイン」の機能
slist.cpp	slist.h	リストデータの管理
font.cpp	font.h	アプリケーションで使用するフォントの設定
resource.rc	resource.h	GUIの構成やイベントIDの定義
	app.h	アプリケーションの情報
	showmsg.h	メッセージの設定

一覧があるだけでも
どこに何があるか
分かりやすくなるね

P2 関数の一覧を用意する

基本的に目的語と動詞を書く

正規表現でgrep検索（サクラエディタの例）

```
□検索条件  "^¥w+¥s*[(]"
検索対象    *.*
フォルダ    C:¥work¥cecsplit_1.0.4r1_win32¥src
    (サブフォルダを検索しない)
    (英大文字小文字を区別しない)
    (正規表現:bregonig.dll Ver.3.06 with Onigmo 5.15.0)
    (文字コードセットの自動判別)
    (一致した行を出力)


■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥font.cpp"  [SJIS]
・(    13,1    ): initAppfont (void)
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥main.cpp"  [SJIS]
・(    39,1    ): WinMain (HINSTANCE inst, HINSTANCE preinst, LPSTR cmdli
・(    88,1    ): parseCommand (LPSTR cmdline)
・(    98,1    ): initResources (HINSTANCE inst)
・(   119,1    ): initApp (HINSTANCE inst, int cmdshow)
・(   130,1    ): waitIdleMessage (void)
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥mainwnd.cpp"  [SJIS]
・(    30,1    ): initMainwnd (HINSTANCE inst)
・(    51,1    ): showMainwnd (HINSTANCE inst, int cmdshow)
・(   157,1    ): resizeMainwnd (void)
・(   234,1    ): mainwndOnCreate (HWND wnd, LPCREATESTRUCT createdata)
・(   245,1    ): mainwndOnClose (HWND wnd)
・(   256,1    ): mainwndOnDestroy (HWND wnd)
・(   266,1    ): mainwndOnCommand (HWND wnd, int id, HWND wndctl, UINT r
```

Excelで整形

クリックすると開ける

```
/* initialization for application font
  APPFONT is seted by this function
  return value: success is non-zero, failure is 0.
 */
int
initAppfont (void)
{
  appfont = CreateFont (
    12,
    0,
    0,
    0,
    FW_REGULAR,
    FALSE,
    FALSE,
```

ファイル名	関数名	説明
font	initAppfont	アプリケーションフォントを初期化する
main	WinMain	エントリポイント
	parseCommand	コマンド引数を解析する
	initResources	リソースを初期化する
	initApp	アプリケーションを初期化する
	waitIdleMessage	メッセージキューが空になるまで処理する
mainwnd	initMainwnd	メインウインドウを初期化する
	showMainwnd	メインウインドウを表示する
	resizeMainwnd	メインウインドウのサイズを変更する
	mainwndOnCreate	メインウインドウを生成するときの処理
	mainwndOnClose	メインウインドウを閉じるときの処理
	mainwndOnDestroy	メインウインドウを破棄するときの処理
	mainwndOnCommand	メインウインドウへのコマンドを処理する
	mainwndOnKeyUp	メインウインドウへのキー入力を処理する
	mainwndProc	メインウインドウへのメッセージを処理する
	getMainwnd	メインウインドウのハンドルを取得する
	onsplitMainwnd	分割開始前後にコントロールの状態を切り替える
qformat	qformat	クイックフォーマットをする
slist	clearSlist	サイズリストをクリアする
	makeSlist	サイズリストを生成する
	addSlist	サイズリストに項目を追加する
	getSlistSize	指定項目のサイズを取得する
	getSlistCount	サイズリストのカウントを取得する
split	checkCancel	分割をキャンセルしていないか確認する
	setCancel	分割をキャンセルする
	showSplited	分割中のメッセージを表示する
	showSplitedd	ドライブに分割中のメッセージを表示する
	waitSplitDrive	メッセージを表示してディスクの交換を待つ
	splitfile	ファイルに分割する

P3 変数の一覧を用意する

基本的に名詞を書く

正規表現でgrep検索

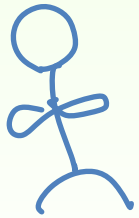
```
□検索条件  "^¥w+¥s+¥w+[^()*[;=]"
検索対象
フォルダ    C:¥work¥cecsplit_1.0.4r1_win32¥src
    (サブフォルダを検索しない)
    (英大文字小文字を区別しない)
    (正規表現:bregonig.dll Ver.3.06 with Onigmo 5.15.0)
    (文字コードセットの自動判別)
    (一致した行を出力)


■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥font.cpp"  [SJIS]
・(     6,1    ): HFONT appfont;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥font.h"  [SJIS]
・(     5,1    ): extern HFONT appfont;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥main.cpp"  [SJIS]
・(    35,1    ): static HWND mainwnd;
・(    36,1    ): static HACCEL accel;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥mainwnd.cpp"  [SJIS]
・(    18,1    ): static LPCSTR mainwnd_class = "main-window";
・(    23,1    ): static HWND mainwnd = NULL;
・(    24,1    ): static int nowsplitting = 0;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥split.cpp"  [SJIS]
・(    20,1    ): static int cancel_split = 0;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥splitopts.cpp"  [SJIS]
・(    14,1    ): static TIMAINOPTS timainopts;
・(    15,1    ): static TIDRIVEOPTS tidriveopts;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥tabctrl.cpp"  [SJIS]
・(    10,1    ): static HWND gl_tabwnd = NULL;
■"C:¥work¥cecsplit_1.0.4r1_win32¥src¥tabitems.cpp"  [SJIS]
・(    11,1    ): TABITEM tabitems[TAB_ITEMS] =
```

Excelで整形

関数スコープ外の変数のみ一覧する

ファイル名	変数名	説明
font	appfont	アプリケーションフォント
main	mainwnd	メインウインドウのハンドル
	accel	ショートカットキーのハンドル
mainwnd	mainwnd_class	メインウインドウのクラス名
	mainwnd	メインウインドウのハンドル
	nowsplitting	分割中フラグ
split	cancel_split	分割キャンセルフラグ
splitopts	timainopts	メインタブ設定の構造体
	tidriveopts	ドライブタブ設定の構造体
tabctrl	gl_tabwnd	タブコントロールのハンドル
tabitems	tabitems	タブ項目構造体の配列
	parentwnd	タブコントロールのハンドル
	selitem	選択しているタブ項目の番号
ti_drive	tidrivewnd	ドライブタブのハンドル
	lsDriveOldproc	ドライブ選択リストの元のメッセージプロシージャ
ti_main	timainwnd	メインタブのハンドル
	droptargetOldProc	自動分割アイコンの元のメッセージプロシージャ
	edittargetOldProc	分割ファイル入力欄の元のメッセージプロシージャ
	editreunionOldProc	結合ファイル入力欄の元のメッセージプロシージャ

正規表現を上手く
使えると便利だね

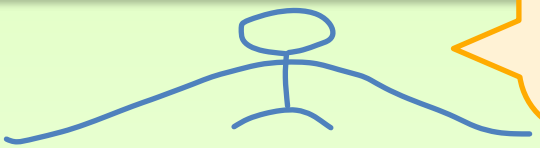


オブジェクト指向なら
クラス名の列があってもいいね

作業時間の目安：100行／時間程度

P4 データ参照関係を調査する

この表があると影響範囲が見て取れるね



Excelで行列を入れ替えて貼り付け

関数

変数		font	main					mainwnd										split											splitop								qformat	slist					tabctrl	
		initAppfont	WinMain	parseCommand	initResources	initApp	waitIdleMessage	initMainwnd	showMainwnd	resizeMainwnd	mainwndOnCreate	mainwndOnClose	mainwndOnDestro	mainwndOnComman	mainwndOnKeyUp	mainwndProc	getMainwnd	onsplitMainwnd	checkCancel	setCancel	showSplited	showSplitedd	waitSplitDrive	splitfile	splitdrive	splitnext	getSplitBuffer	splitBufferFree	makeNextfname	writerfile	writerfiled	writerfilererror	onsplit	initOptions	defOptions	setOptions	getOptions	getSplitopt	getTimainopts	getTidriveopts	saveOptions	rlineOptions	loadOptions	qformat	clearSlist	makeSlist	addSlist	getSlistSize	getSlistCount	initTabctrl	getTabwnd
font	appfont	C							R																																				
main	mainwnd		R			C	R																																						
	accel		R		CR		R																																						
mainwnd	mainwnd_class							R	R																																				
	mainwnd					R			C	R							R	R																												
	nowsplitting										R			R	R			U																												
split	cancel_split																		RU	U																										
splitopts	timainopts																																												
	tidriveopts																																												
tabctrl	gl_tabwnd																																												
tabitems	tabitems																																												
	parentwnd																																												
	selitem																																												
ti_drive	tidrivewnd																																												
	IsDriveOldproc																																												
	timainwnd																																												
	droptargetOldProc																																												

変数名でgrep検索

調査した範囲

調査する範囲

簡単にするなら
交点に●でもいい

Create 生成
Read 読み取り
Update 更新
Delete 削除

（参考）調査する機会が多いなら、ツールを用意しておくと時間短縮できる
https://affordd.jp/conference2017/affordd_conference2017_p5.pdf

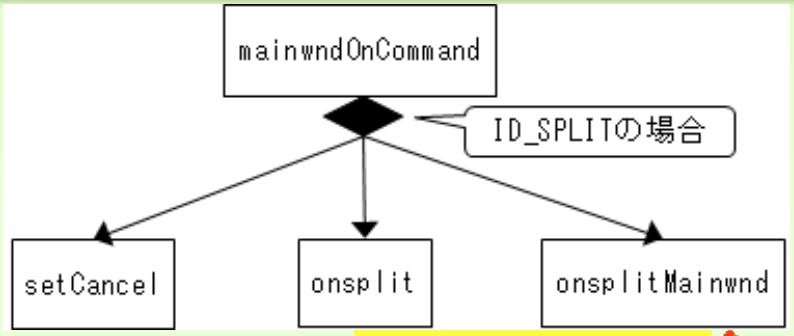
作業時間の目安：80交点／時間程度

P5 知りたい機能の処理構造を調査する

分割開始ボタンを
クリックしたときの
処理を調査したい場合

ファイル名	関数名	説明
font	initAppfont	アプリケーションフォントを初期化する
main	WinMain	エントリポイント
	parseCommand	コマンド引数を解析する
	initResources	リソースを初期化する
	initApp	アプリケーションを初期化する
	waitIdleMessage	次のメッセージが得られるまで待機する
mainwnd	initMainwnd	メインウインドウを初期化する
	showMainwnd	メインウインドウを表示する
	resizeMainwnd	メインウインドウのサイズを変更する
	mainwndOnCreate	メインウインドウを生成するときの処理
		理
		処理
	mainwndOnCommand	メインウインドウへのコマンドを処理する
	mainwndOnKeyUp	メインウインドウへのキー入力を処理する
	mainwndProc	メインウインドウへのメッセージを処理する

①関数一覧から当たりをつけて



③構造図に表現

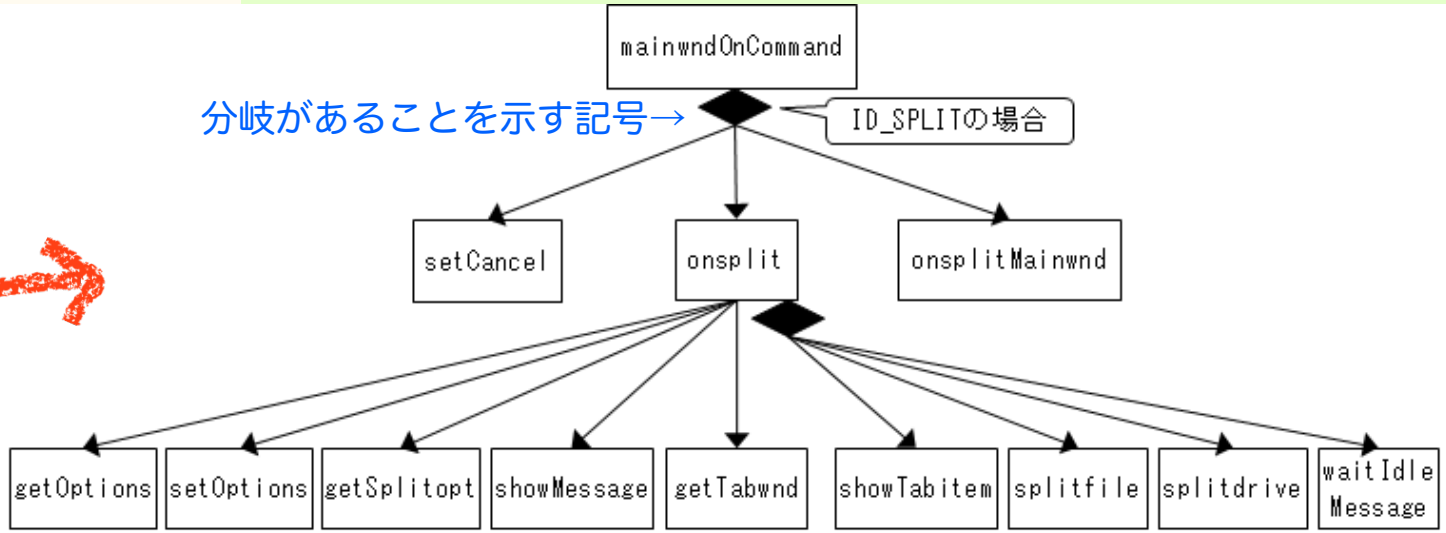
```
static void
mainwndOnCommand (HWND wnd, int id, HWND wndctl, UINT notify)
{
  if (id != ID_SPLIT && nowsplitting)
    return;

  switch (id)
    {
    case ID_CHFOCUS:
      SendMessage (tabitems[TabCtrl_GetCurSel (getTabwnd ())].dlg,
          WM_COMMAND, MAKEWPARAM (ID_INCFOCUS, 0L), NULL);
      break;

    case ID_SPLIT:
        if (nowsplitting)
          setCancel ();
        else
          {
            onsplitMainwnd (0);
            onsplit ();
            onsplitMainwnd (1);
          }
      break;

    case ID_EXIT:
      DestroyWindow (wnd);
      break;
    }

  return;
}
```

②コードを調査して

P5 知りたい機能の処理構造を調査する

```
void
onsplit (void)
{
  SPLITOPT sopts;
  REUNIONOPT ropts;

  getOptions ();
  setOptions ();

  if (!getSplitopt (&sopts, &ropts))
    {
      showMessage (SMSG_SPLIT_FAIL);
      return;
    }

  showMessage (SMSG_SPLIT_START);

  switch (TabCtrl_GetCurSel (getTabwnd ()))
    {
    case TI_MAIN:
    /* case TI_OPTION: */
      showTabitem (TI_MAIN);
      waitIdleMessage ();

      if (!splitfile (sopts, ropts))
        showMessage (SMSG_SPLIT_FAIL);
      else
        showMessage (SMSG_SPLIT_SUCCESS);

      break;

    case TI_DRIVE:
      showTabitem (TI_MAIN);
      waitIdleMessage ();

      if (!splitdrive (sopts, ropts))
        showMessage (SMSG_SPLIT_FAIL);
      else
        showMessage (SMSG_SPLIT_SUCCESS);

      break;

    default:
      SHOWINTERROR ("SPLIT onsplit () switch illegal default.");
      break;
    }

  return;
}[EOF]
```

中身を調査する必要がなければ
マクロやAPIは書かなくていい

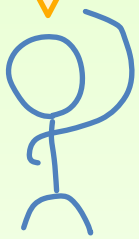


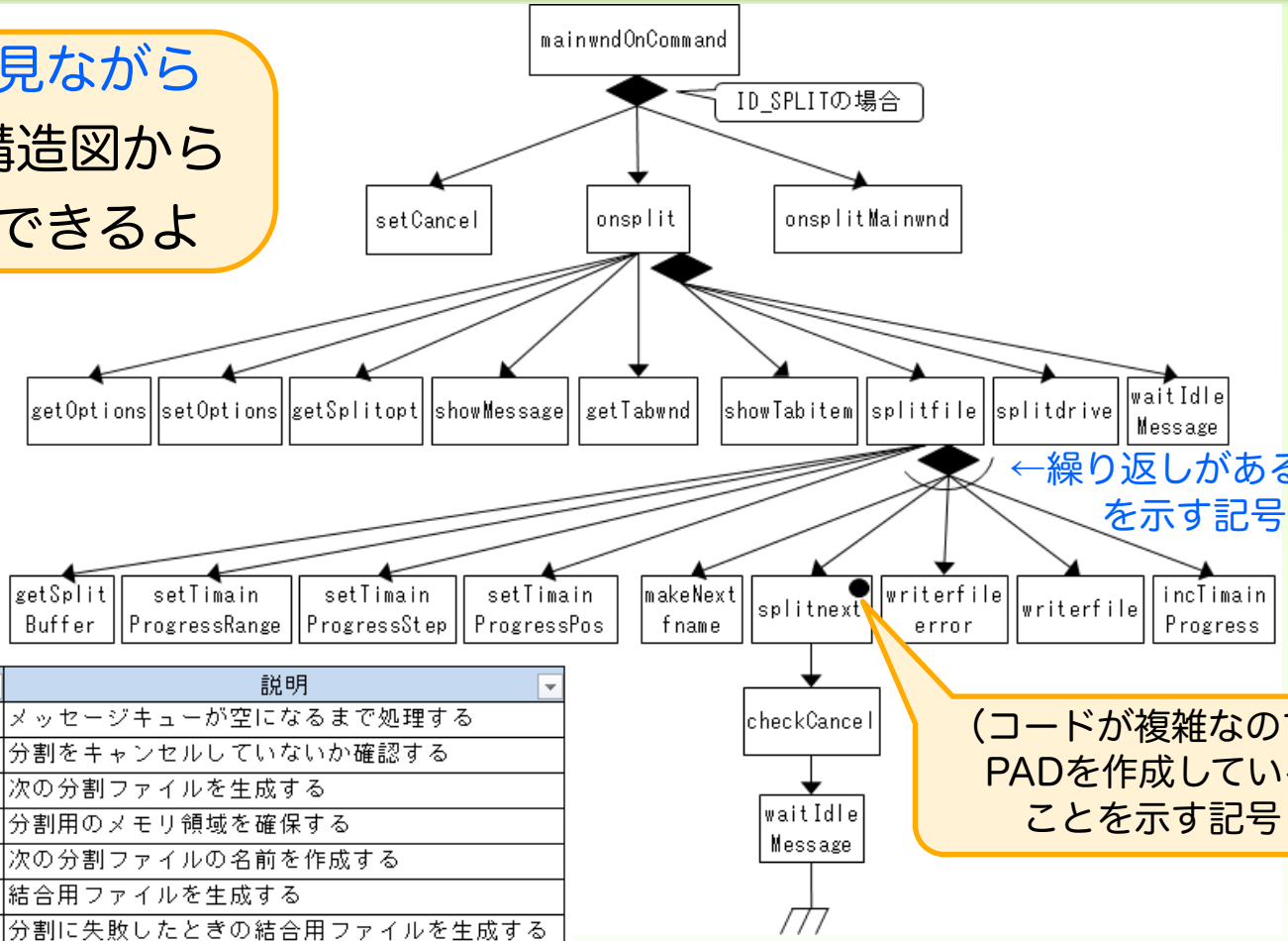
関数一覧の該当箇所を
並べて見ると分かりやすい

ファイル	関数名	説明
main	waitIdleMessage	メッセージキューが空になるまで処理する
mainwnd	mainwndOnCommand	メインウインドウへのコマンドを処理する
	onsplitMainwnd	分割開始前後にコントロールの状態を切り替える
split	splitfile	ファイルに分割する
	splitdrive	ドライブに分割する
	onsplit	分割を開始する
splitopts	setOptions	オプションを各タブに反映する
	getOptions	各タブからオプションに反映する
	getSplitopt	オプションをチェックして取得する
tabctrl	getTabwnd	タブコントロールのハンドルを取得する
tabitems	showTabitem	表示するタブを切り替える
ti_main	showMessage	メッセージを表示する

P5 知りたい機能の処理構造を調査する

実際にコードを見ながら
辿ってきたから構造図から
状況をイメージできるよ





関数一覧の該当箇所

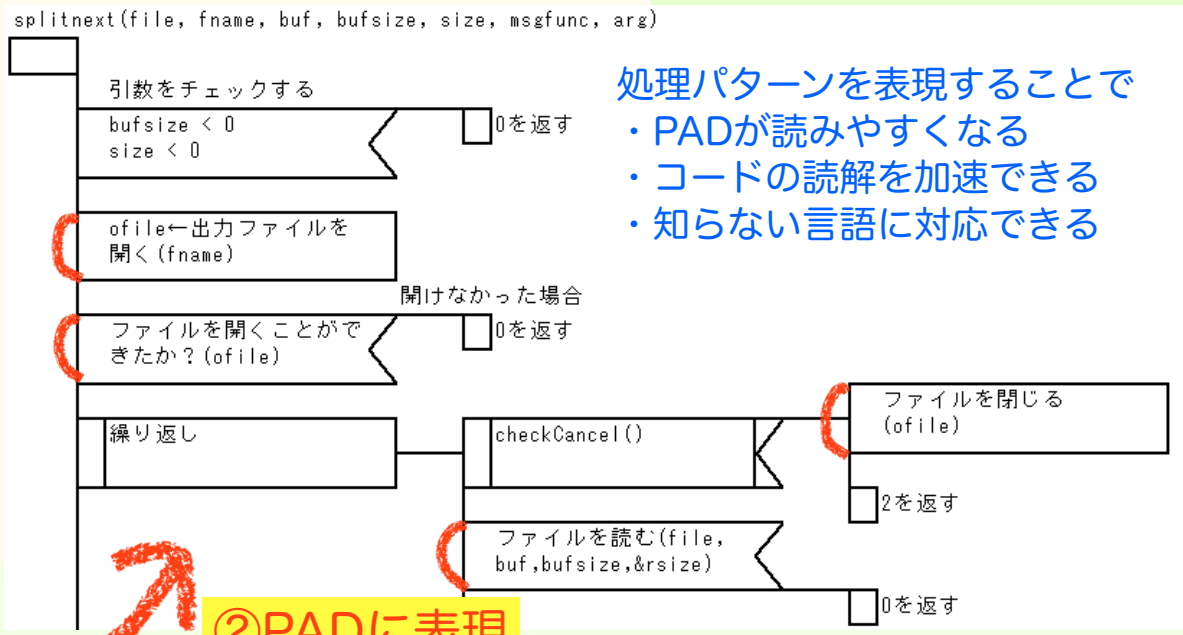
ファイル	関数名	説明
main	waitIdleMessage	メッセージキューが空になるまで処理する
split	checkCancel	分割をキャンセルしていないか確認する
	splitnext	次の分割ファイルを生成する
	getSplitBuffer	分割用のメモリ領域を確保する
	makeNextfname	次の分割ファイルの名前を作成する
	writerfile	結合用ファイルを生成する
	writerfileerror	分割に失敗したときの結合用ファイルを生成する
ti_main	setTimainProgressRange	プログレスバーの範囲を設定する
	setTimainProgressStep	プログレスバーのステップを設定する
	setTimainProgressPos	プログレスバーの値を設定する
	incTimainProgress	プログレスバーの値を増やす

作業時間の目安：30関数／時間程度

P6 必要に応じて処理手順を調査する

```
static int
splitnext (HANDLE file, LPCTSTR fname, LPVOID buf, long bufsize, long size, void (*msgfunc)(DWORD, DWORD, void *), void *arg)
{
  HANDLE ofile;
  long rsize, wsize, asize = size;

  /* the options is examined */
  if (bufsize < 0)
    return 0;
  if (size < 0)
    return 0;

  ofile = CreateFile (fname, GENERIC_WRITE, 0, NULL,
    CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
  if (ofile == INVALID_HANDLE_VALUE)
    return 0;

  for (;;)
    {
      if (checkCancel ())
        {
          CloseHandle (ofile);
          return 2;
        }

      if (! ReadFile (file, buf, bufsize, (DWORD *)&rsize, NULL))
        return 0;
```

①処理パターンを抽出して

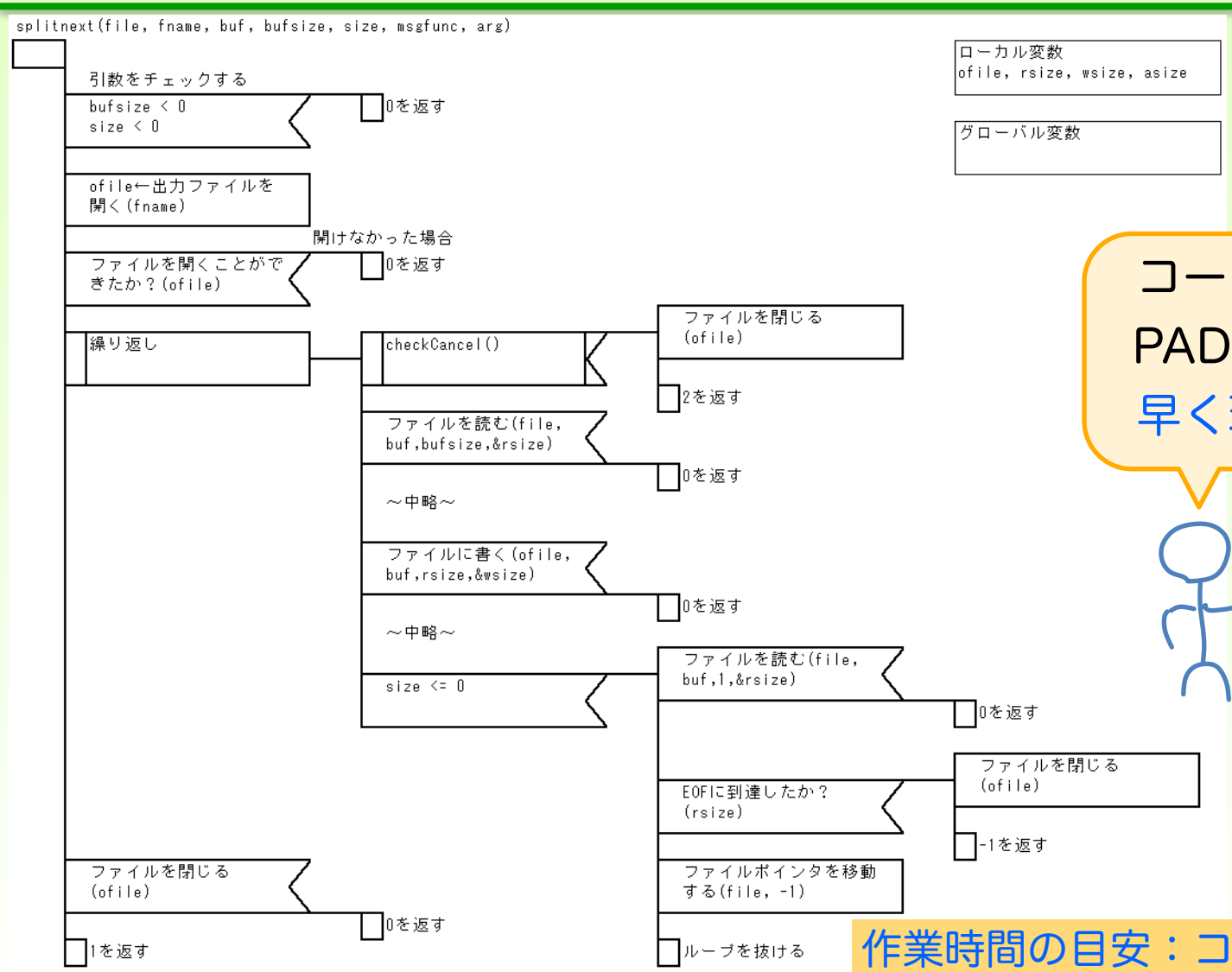


②PADに表現

処理パターンを表現することで
- PADが読みやすくなる
- コードの読解を加速できる
- 知らない言語に対応できる

処理分類	処理パターン	実装パターン
ファイル操作	ハンドル←出力ファイルを開く(ファイル名)	ハンドル = CreateFile (ファイル名, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
	ファイルを開くことができたか？(ハンドル)	if (ハンドル == INVALID_HANDLE_VALUE) ～開けなかった場合の処理～
	EOFに到達したか？(読み込んだサイズ)	if (! 読み込んだサイズ) ～EOFに到達した場合の処理～
	ファイルを読む(ハンドル,バッファ,読み込むサイズ,&読み込んだサイズ)	ReadFile (ハンドル, バッファ, 読み込むサイズ, &読み込んだサイズ, NULL)
	ファイルに書く(ハンドル,バッファ,書き込むサイズ,&書き込んだサイズ)	WriteFile (ハンドル, バッファ, 書き込むサイズ, &書き込んだサイズ, NULL)
	ファイルポインタを移動する(ハンドル, 移動量)	SetFilePointer (ハンドル, 移動量, NULL, FILE_CURRENT)
	ファイルを閉じる(ハンドル)	CloseHandle (ハンドル)

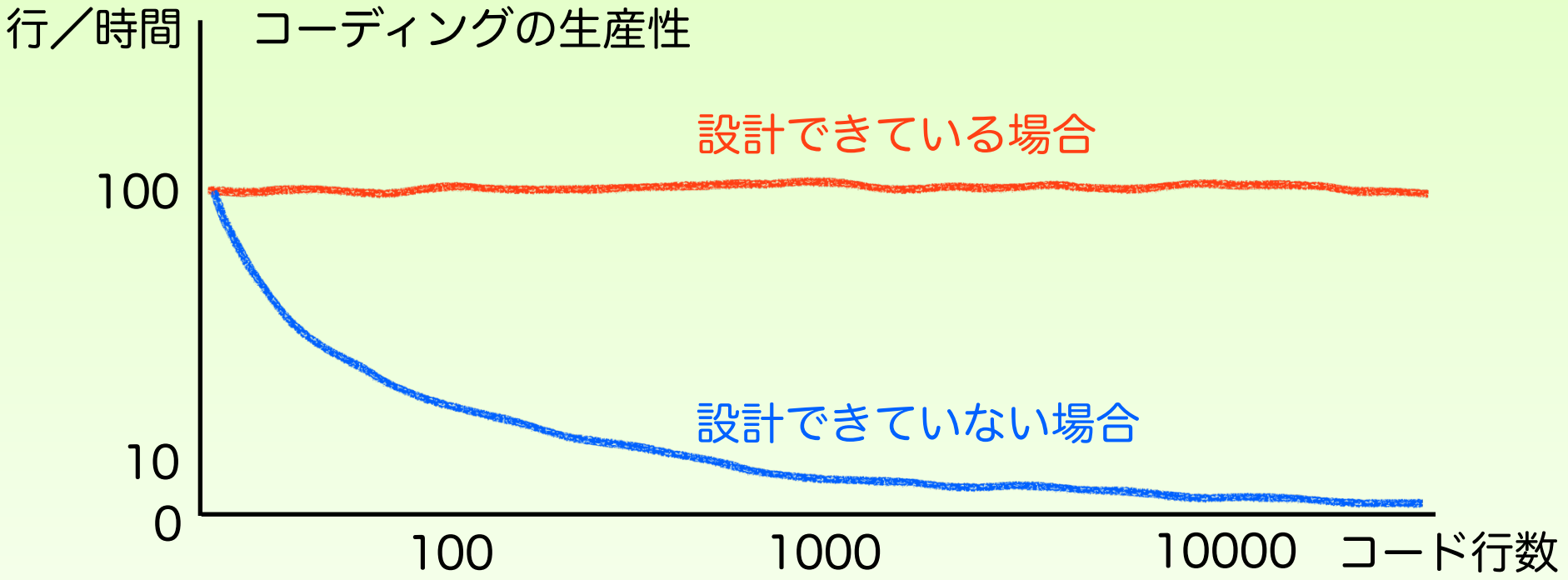
P6 必要に応じて処理手順を調査する



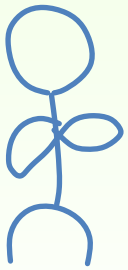
コードが複雑なら
PADを書いた方が
早く理解できるよ

作業時間の目安：コード300行／時間程度

設計と生産性の関係



100行／時間でコードを
書けるように設計しよう！



設計ができると仕事が
早く終わりそうだね
適切な要求仕様がある
ことも前提になりそう

P7 要求仕様を復元する

```
static void
mainwndOnCommand (HWND wnd, int id, HWND wndctl, UINT notify)
{
  if (id != ID_SPLIT && nowsplitting)
    return;

  switch (id)
    {
    case ID_CHFOCUS:
      SendMessage (tabitems[TabCtrl_GetCurSel (getTabwnd ())].dlg,
        WM_COMMAND, MAKEWPARAM (ID_INCFOCUS, 0L), NULL);
      break;
    case ID_SPLIT:
      if (nowsplitting)
        setCancel ();
      else
        {
          onsplitMainwnd (0);
          onsplit ();
          onsplitMainwnd (1);
        }
      break;

    case ID
      Destr
      break
    }

  return;
}
```

```
void
setCancel (void)
{
  cancel_split = 1;
}
```

関数一覧と変数一覧の該当箇所

ファイル	関数名	説明
mainwnd	mainwndOnCommand	メインウインドウへのコマンドを処理する
	onsplitMainwnd	分割開始前後にコントロールの状態を切り替える
split	checkCancel	分割をキャンセルしていないか確認する
	setCancel	分割をキャンセルする
	onsplit	分割を開始する

ファイル	変数名	説明	データ構造
mainwnd	nowsplitting	分割中フラグ	=有効(1)｜無効(0)
split	cancel_split	分割キャンセルフラグ	=有効(1)｜無効(0)

①TMの列にファイル名を並べておく

③コードや構造図・PADを参照しながら要求を書く
（イベントから始まる一連の動作を表現する）

カテゴリ	要求仕様			備考
分割機能 SPT	要求	SPT1	分割開始ボタンをクリックしたとき、既に分割中であればキャンセルする。分割中でなければ分割を開始する。	
		理由	分割に時間がかかる場合にキャンセルしたいから。	
		説明		
			＜分割開始ボタンの受付＞	
		□	SPT1-10	分割中フラグが有効であれば、分割キャンセルフラグを有効にして処理を終える。

②カテゴリと記号を決める

④理由を書く

⑤仕様を書く

機能分類TM

cecsplit	font	main	mainwnd	qformat	slist	split	splitopt	tabctrl	tabitems	ti_drive	ti_main
			mainwndOnCommand			setCancel					

⑥TMの交点に関数名を書く

（参考）USDMの書き方についての説明資料
https://affordd.jp/conference2017/affordd_conference2017_tutorial.pdf

P7 要求仕様を復元する

仕様に関連する変数の内容を調査しておく

```
void
onsplitMainwnd (int sw)
{
  if (! sw)
    {
    nowsplitting = 1;
      /* ONSPLIT */
      Button_SetText (GetD
      Button_Enable (GetDl
      Button_Enable (GetDl
    }
  else
    {
    nowsplitting = 0;
      /* OFFSPLIT */
      Button_SetText (GetD
      Button_Enable (GetDl
      Button_Enable (GetDl
    }
}

static int
checkCancel (void)
{
  waitIdleMessage ();

  if (cancel_split)
    {
    cancel_split = 0;
      return 1;
    }
}

void
setCancel (void)
{
  cancel_split = 1;
}
```

関数一覧と変数一覧の該当箇所

	ファイル	関数名	説明
●	mainwnd	mainwndOnCommand	メインウインドウへのコマンドを処理する
●		onsplitMainwnd	分割開始前後にコントロールの状態を切り替える
●	split	checkCancel	分割をキャンセルしていないか確認する
●		setCancel	分割をキャンセルする
●		onsplit	分割を開始する

	ファイル	変数名	説明	データ構造
●	mainwnd	nowsplitting	分割中フラグ	=有効(1)｜無効(0)
●	split	cancel_split	分割キャンセルフラグ	=有効(1)｜無効(0)

②該当する関数を調査して変数一覧にデータ構造を追記する

基本的にデータ・ディクショナリ形式で表現して仕様と対応付ける

変数	
font	appfont
main	mainwnd
	accel
mainwnd	mainwnd_class
	mainwnd
	nowsplitting
split	cancel_split

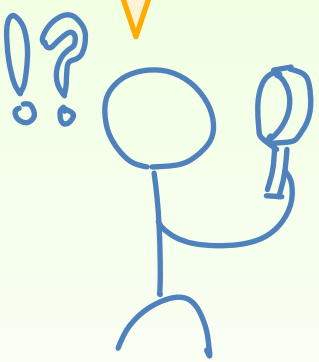
関数	font	main				mainwnd										split						
	initAppfont	WinMain	parseCommand	initResources	initApp	waitIdleMessage	initMainwnd	showMainwnd	resizeMainwnd	mainwndOnCreate	mainwndOnClose	mainwndOnDestro	mainwndOnComman	mainwndOnKeyUp	mainwndProc	getMainwnd	onsplitMainwnd	checkCancel	setCancel	showSplited	showSplitedd	waitSplitDrive	splitfile	splitdrive	splitnext
mainwnd				R				C	R							R	R				R				
										R			R	R			U								
split																	RU	U							

①データ参照表から変数の生成・更新箇所を探す

P7 要求仕様を復元する

関係者と読み合わせて
チェックしていく

この時点で色々な
問題に気付くかも



カテゴリ	要求仕様				備考
分割機能 SPT	要求	SPT1	分割開始ボタンをクリックしたとき、既に分割中であればキャンセルする。分割中でなければ分割を開始する。		
		理由	分割に時間がかかる場合にキャンセルしたいから。		
		説明			
			＜分割開始ボタンの受付＞		
		■	SPT1-10	分割中フラグが有効であれば、分割キャンセルフラグを有効にして処理を終える。	
		■	SPT1-11	分割中フラグが無効であれば、＜分割の開始＞をする。このとき、分割中フラグとコントロールの状態を下記のように切り替える。 ＜分割中＞ ・分割中フラグ＝有効 ・分割開始ボタンのテキスト＝"分割中止(&S)" ・終了ボタン＝無効 ・デフォルトボタン＝無効 ＜完了後＞ ・分割中フラグ＝無効 ・分割開始ボタンのテキスト＝"分割開始(&S)" ・終了ボタン＝有効 ・デフォルトボタン＝有効	
	要求	SPT2	分割を開始したとき、オプションを取得して、分割開始のメッセージを表示して、選択中のタブに応じてファイルに分割またはドライブに分割して、結果を表示する。		
		理由	指定サイズでの分割とディスク空き容量での分割を選択したいから。		
		説明			
			＜分割の開始＞		
		■	SPT2-10	各タブからオプションを取得する。異常があればエラーメッセージと"分割失敗!"を表示して、処理を終える。 ＜エラーメッセージ＞ ・一時メモリを確保できない場合：メモリ領域を取得できません。 ・分割サイズに0以下もしくはLONG_MAXを超える値を指定した場合：分割サイズが不正です。	
		■	STP2-11	分割開始のメッセージを表示する。 ・分割開始のメッセージ＝"分割開始..."	
		■	SPT2-12	選択中のタブを取得する。	
		■	STP2-13	メインタブを表示する。 【理由】メッセージ表示欄があるため	
		■	SPT2-14	選択中のタブがメインタブであれば＜ファイルに分割＞をする。 ドライブタブであれば＜ドライブに分割	
		■	STP2-15	分割を完了したら、"分割完了"または"分	

機能分類TM											
cecsplit											
	font	main	mainwnd	qformat	slist	split	splitopt	tabctrl	tabitems	ti_drive	ti_main
			mainwndOnCommand			setCancel					
			mainwndOnCommand onsplitMainwnd			onsplit					
						onsplit					
							getOptions setOptions getSplitopt				getTimainOpts setTimainOpts showMessage
											showMessage
								getTabwnd			
											showTabitem
						splitfile					

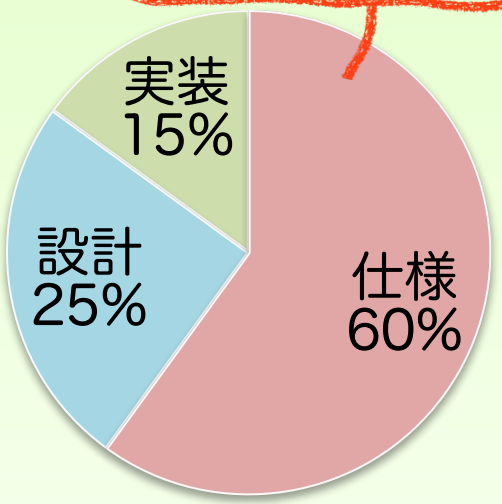
作業時間の目安：コード100行／時間程度

要求仕様と不具合の関係

改善が進んでいない開発現場の不具合状況

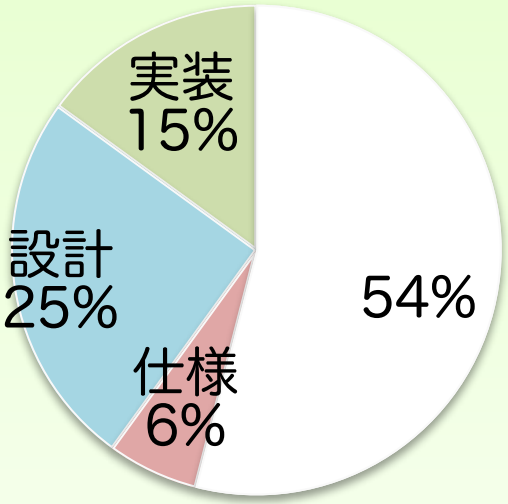
仕様に関する不具合：10～20件／KLOC

不具合発生工程：

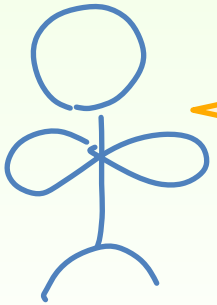


USDMにより達成可能な水準

1件／KLOC以下



不具合の半分は
要求仕様の適切な表現と
レビューで見つけよう！



特に要求の漏れや誤りは
事前にしっかりレビューして
もらわないとテストで
検出するのは難しそうだね

P8 テスト仕様を作成する

①TMの列に仕様番号を並べておく

②テストを効率的に進めるための手順・条件を書く
（準備や片付けも含めて作業しやすい順番に並べる）

					テスト結果	テスト対象TM							
						SPT1-10	SPT1-11	SPT2-10	STP2-11	SPT2-12	STP2-13	SPT2-14	STP2-15
カテゴリ			テスト仕様	備考	判定	メモ	1	0.2	0.6	1	1	1	0.5	1
分割機能のテスト TSPT	要求	TSPT1	分割ファイルを指定して、結合ファイルの自動生成をチェックして、分割開始ボタンをクリックする。分割完了したら、結合用のバッチファイルを実行して、結合したファイルと元のファイルの内容を比較して、一致していることを確認する											
		理由	基本的な分割動作を確認するため。											
		説明												
			＜事前準備＞											
		□	TSPT1-10	サイズが100MB以上のファイルを用意しておく。										
			＜確認方法＞											
		□	TSPT1-20	ファイル内容の比較には fc /b コマンドを用いる。										
		□	TSPT1-21	ファイルサイズの比較には下記のコマンドを用いる。 echo off & (for /f "delims=" %i in (' dir /a /s /b * ') do echo %i^,%~zi) & echo on										
			＜正常動作の確認＞											
		□	TSPT1-30	下記の分割サイズを指定して分割・結合したとき、分割後のファイルサイズ、結合後の比較結果が一致している。 また、開始時に"分割開始...."、完了後に"分割完了"が表示される。 ・"3.5FD 2HD 1.44MB"：1457664バイト				0.2	0.3	1	1	1	0.5	0.5
			＜キャンセル動作の確認＞											
		□	TSPT1-40	分割中に「分割中止」ボタンをクリックすると分割をキャンセルできる。 また、"分割はキャンセルされました。"および"分割失敗!"が表示される。			1							0.5
			＜異常動作の確認＞											
			TSPT1-50	分割サイズに下記を指定したとき"分割サイズが不正です。"と"分割失敗!"が表示される。 ①"A" ②"-1" ③"0" ④"2147483648"（LONG_MAX+1、単位は"byte"とする）					0.3					●

④テスト漏れをチェックしてテストやレビューの追加要否を判断する

自動テストにする場合はテストプログラムの要求仕様になる
（機能分類TMを追加すると良い）

③仕様網羅率を記入する

⑤読み合せてチェック

作業時間の目安：10仕様／時間程度

P9 テスト結果を記入する

作業しやすく速くなるから
テスト仕様を書きたくなるよ

					テスト結果		テスト対象TM							
					YYYY-mm-dd 担当者名		SPT1-10	SPT1-11	SPT2-10	SPT2-11	SPT2-12	SPT2-13	SPT2-14	SPT2-15
カテゴリ	テスト仕様			備考	判定	メモ	1	0.2	0.6	1	1	1	0.5	1
分割機能のテスト TSPT	要求	TSPT1	分割ファイルを指定して、結合ファイルの自動生成をチェックして、分割開始ボタンをクリックする。分割完了したら、結合用のバッチファイルを実行して、結合したファイルと元のファイルの内容を比較して、一致していることを確認する											
		理由	基本的な分割動作を確認するため。											
		説明												
			＜事前準備＞											
		■	TSPT1-10	サイズが100MB以上のファイルを用意しておく。		OK								
			＜確認方法＞											
		■	TSPT1-20	ファイル内容の比較には fc /b コマンドを用いる。		OK								
		■	TSPT1-21	ファイルサイズの比較には下記のコマンドを用いる。 echo off & (for /f "delims=" %i in (' dir /a /s /b * ') do echo %i^,%~zi) & echo on		OK								
			＜正常動作の確認＞											
		■	TSPT1-30	下記の分割サイズを指定して分割・結合したとき、分割後のファイルサイズ、結合後の比較結果が一致している。 また、開始時に"分割開始..."、完了後に"分割完了"が表示される。 ・"3.5FD 2HD 1.44MB"：1457664バイト		NG	□ファイルサイズが1バイト多い		0.2	0.3	1	1	1	0.5	0.5
			＜キャンセル動作の確認＞											
		■	TSPT1-40	分割中に「分割中止」ボタンをクリックすると分割をキャンセルできる。 また、"分割はキャンセルされました。"および"分割失敗!"が表示される。				1							0.5
			＜異常動作の確認＞											
		■	TSPT1-50	分割サイズに下記を指定したとき"分割サイズが不正です。"と"分割失敗!"が表示される。 ①"A" ②"-1" ③"0" ④"2147483648"（LONG_MAX+1、単位は"byte"とする）						0.3					●

作業日と担当者を記入

テストを実行しながら
判定とメモを記入する

作業時間のイメージ

例えば、3000行程度のソースファイルを想定すると

番号	プロセス	生産性	サイズ	見積り[時間]
P1	資料やソースファイルの一覧を用意する	100行／時間	20種類	0.2
P2	関数の一覧を用意する	100行／時間	60関数	0.5
P3	変数の一覧を用意する	100行／時間	30変数	0.3
P4	データ参照関係を調査する	80交点／時間	80交点	1
P5	知りたい機能の処理構造を調査する	30関数／時間	60関数	2
P6	必要に応じて処理手順を調査する	コード300行／時間	600行	2
P7	要求仕様を復元する	コード100行／時間	3000行	30
P8	テスト仕様を作成する	10仕様／時間	30仕様	3
P9	テスト結果を記入する	5仕様／時間	30仕様	6
	仕様変更を含む不具合を修正する	0.3件／時間	90件	300

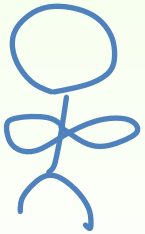
＋事前の習得期間

1日程度

2週間程度

3ヶ月程度

事前に作業時間を見積り
限られた時間の中で
最良の手順を選択しよう



不具合が早く見つかるなら
これくらい時間をかけても
いいかな？

砂漠を広げない

既に「ソースコードしかない」・・・砂漠の状態

担当者は、このことを批判するが・・・

多くの場合、新しい担当者も「砂漠」を広げてしまう

原因＝新規設計の技術も、スペックアウトの技術もない

新しく砂漠を広げないことが大事

スペックアウトによりソースコードの状況と、それを設計図表にしたときの関係が見えるので、新規設計時に設計の落とし所が分かるようになる

今回調査した所、修正する所を「オアシス」にして、混乱を鎮めていく

【参考文献】

[1] スペックアウト作業の進め方　～派生開発では、適切なスペックアウト技術が成功の決め手になる～　第6版 , 清水 吉男

[2] 構造化仕様書作成マニュアル　～構造化分析ガイドブック～　第5版, 清水 吉男

[3] 構造化設計のためのテキストブック 第1版, 清水 吉男