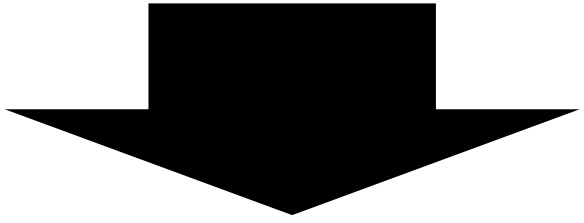
Markdown と Doxygen を駆使して設計工程の簡略化を図る

片 岡　　健

ソフトウェア保守現場でありがちなこと

- 機能追加の案件に対して・・・
  - 最初から実装レベルで対応方針の当たりがついている
  - 実装内容が分かっているのにわざわざ設計工程の文書を作成するのは煩わしい
  - とはいえ、リリース後の保守を考えると設計書はあった方がよい



Markdown ＋ Doxygen

Markdown とは

- マークアップ言語の一種
- プレーンテキスト形式の文書から HTML を生成するために開発された

  (出典：Wikipedia『Markdown』 より)

- Markdown と HTML の対応例

Markdown		HTML
# レベル１の見出し	⇒	レベル１の見出し
### レベル３の見出し	⇒	レベル３の見出し
* すいか * メロン	⇒	 - すいか - メロン

Doxygen とは

- C++、C、Java、Objective-C、Python、PHP 等、様々なプログラミング言語をサポートするドキュメンテーションジェネレーター
- ソースファイルから HTML、PDF 等に出力可能
- 設定によりソースコードの構造を視覚化も可能
- V1.8.0 より Markdown もサポート

(出典：Doxygen 公式 HP より)

具体例 1

【ソースコード】

```
/*****************************************************
 *↓
 * @brief 暗号化プロセス呼出ライブラリ データ送受信処理
 *↓
 *↓
 * @par 概要↓
 *      暗号化プロセスへのデータ送受信を行う↓
 *↓
 * @param[in]       iCryptFflg   暗号化プロセス指示フラ
 * @param           pCCP  暗号化プロセス呼出データ構造
 * @retval  0   正常↓
 * @retval  -1  エラー↓
 *↓
 *↓
 * @dot↓
 * digraph iCryptProcGateXmit {↓
 *     node [shape=Mrecord, fontname=Helvetica, fontsi
 *     a [ label="iCryptProcGateEnc" URL="¥ref iCryptP
 *     b [ label="iCryptProcGateDec" URL="¥ref iCryptP
 *     c [ label="iCryptProcGateXmit" URL="¥ref iCrypt
 *     d [ label="vMQPutData"];↓
 *     e [ label="vMQGetData"];↓
 *     f [ label="isSameTrx"];↓
 *     a -> c↓
 *     b -> c↓
 *     c -> d↓
 *     c -> e↓
 *     C -> f↓
 *↓
 * }↓
 * @enddot↓
 *↓
 *↓
 * ----↓
 *↓
 * ## 処理内容↓
 *↓
 *****************************************************
int iCryptProcGateXmit(eCryptFlg iCryptFflg , stCallCr
{↓
```

【HTML】

int iCryptProcGateXmit (eCryptFlg iCryptFflg,
stCallCryptProc * pCCP
)

暗号化プロセス呼出ライブラリ データ送受信処理

概要

暗号化プロセスへのデータ送受信を行う

Parameters

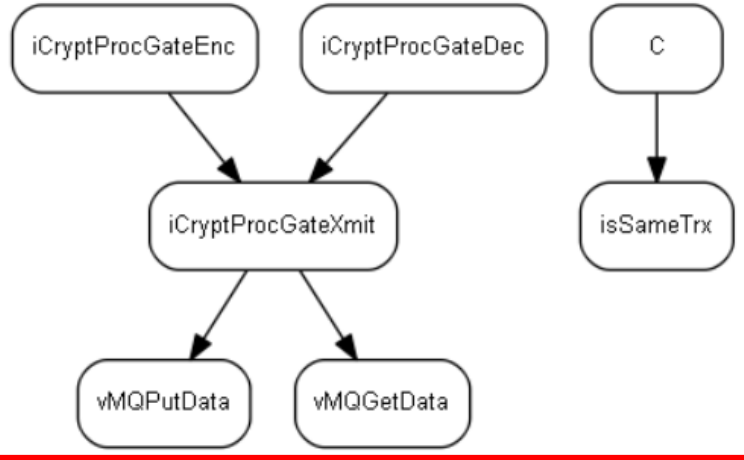
[in] **iCryptFflg** 暗号化プロセス指示フラグ

pCCP 暗号化プロセス呼出データ構造体ポインタ

Return values

0 正常

-1 エラー



具体例２

【ソースコード】

```
/**
 * ### 送信データをセット
 * パラメータをもとに、暗号化プロセスに送
 * - パラメータ内の暗号化取引MQ送信デ
 *   送信データ内にセットする。
 * - パラメータの暗号化プロセス指示フラグ
 *   暗号化プロセスインターフェイス構造体
 */
memcpy(&sendProc, &pCCP->cryptTrxSendDat
((stCryptProc *)sendProc.achCryptProcDat


/**
 * ### データ送信
 * セットしたデータを、パラメータ内のMQ情
 * により暗号化プロセスに送信する。
 * > @ref vMQPutData
 * - データ送信に失敗した場合は、エラーロ
 *   戻り値にエラーを返却して処理を終了す
 *   > vLogFormatMQ(char *, char *, stCr
 */
pfvMQPutData(*pCCP->pCryptMQ, (char *)&s
if ((*pCCP->pCryptMQ)->MQCompCode != MQC
  {
    pfvLogFormatMQ(achErrFuncName,
                   "PutData",
                   &(*pCCP->pCryptMQ)->r
    return ERROR_CALLCRYPTPROC_RET;
  }

/**
 * ### データ受信待ち処理
 * 暗号化プロセスからデータを受信する。
 * この受信処理は同一の取引特定コードのデ
 * > データ受信待ちループ条件：無限ルー
 */
while(1)
  {
```

【HTML】

処理内容

送信データをセット

パラメータをもとに、暗号化プロセスに送信するデータを作成する。

- パラメータ内の暗号化取引MQ送信データ(_stCallCryptProc.cryptTrxSendData)を、送信データ内にセットする。
- パラメータの暗号化プロセス指示フラグ(iCryptFflg)を、送信データ内の 暗号化プロセスインターフェイス構造体(_stCryptProc.eCFlg)にセットする。

データ送信

セットしたデータを、パラメータ内のMQ情報(_stCallCryptProc.pCryptMQ) により暗号化プロセスに送信する。

> vMQPutData

- データ送信に失敗した場合は、エラーログを出力し、戻り値にエラーを返却して処理を終了する。

> vLogFormatMQ(char *, char *, stCryptQ *)

データ受信待ち処理

暗号化プロセスからデータを受信する。この受信処理は同一の取引特定コードのデータを受信するまでループする。

> データ受信待ちループ条件: 無限ループ

データ受信

パラメータ内のMQ情報(_stCallCryptProc.pCryptMQ)により 暗号化プロセスからの応答データを受信する。

> vMQGetData

- データ受信がタイムアウトにより失敗した場合は、エラーログに タイムアウトメッセージを出力し、戻り値にエラーを返却して処理を終了する。

まとめ・利点

- ソースコードを Input にして設計書ができる

【コスト面の利点】

- 設計工程の工数圧縮が狙える
  (本来は設計書とソースコードは別々の成果物だが、この方法だと
  ソースコードのみで二役担える)

【品質面の利点】

- 『設計→実装』時の漏れの予防につながる
  (コメントを書いている以上、対応するコードは書きましょう)

【その他】

- ~~ソフトウェア『テスト』と何が関係あるんだ？~~

ご清聴ありがとうございました