

ソフトウェア品質シンポジウム

MDAツールによるソフトウェア品質保証

株式会社 IHIエスキューブ

品質保証部

○平井 宣

E-mail : hirai9589@ihi-g.com

本日の内容について

- 研究のきっかけ：QCD原体験とSEの生産性
- 着想：開発ドキュメントの適正量とは
- MDAによるドキュメント削減と品質確保
- 評価方法と結果
- 評価に対する考察
- まとめ

想定参加者

- 開発期間が短くSEの確保で苦勞しているPM
- 運用保守で忙しいわりに先行きが見通せないSE
- コンサルが入って要件コントロールに苦勞しているPM
- 開発ドキュメントをコンパクトにしたいQA

QCD意識が高まった原体験



'Post Office in a Box' concept trialled to ensure community presence

By Robin Manning | 16 March 2020 | 2 min read

A new 'Post Office in a Box' concept is being trialled by the Post Office in five locations across the UK, with the aim of ensuring its services are made available where there is an "urgent need".



たった一人のPMが約400人の設計者を束ね、Doors(※)でQCDをコントロールしていた。

ドキュメント体系はISO/IEC 12207準拠と思われる。書式は国内と大きく異なりMS Word中心。

QCDの統合的管理を考えるきっかけとなった。


※IBM Rational Doors: Word/Excelなどのセンテンスをリンクさせ台帳管理できるツール。組込み系で実績が多い。他にSiemens SoftwareのPolarionなどが知られている。

品質王国日本の開発ドキュメント事情

日本の設計書はデータ項目が多く、ドキュメントのフォーマットにこだわることで、生産性や品質を確保する傾向がある。

以下は、神奈川県におけるHER-SYS（感染症管理システム）の入力要領。

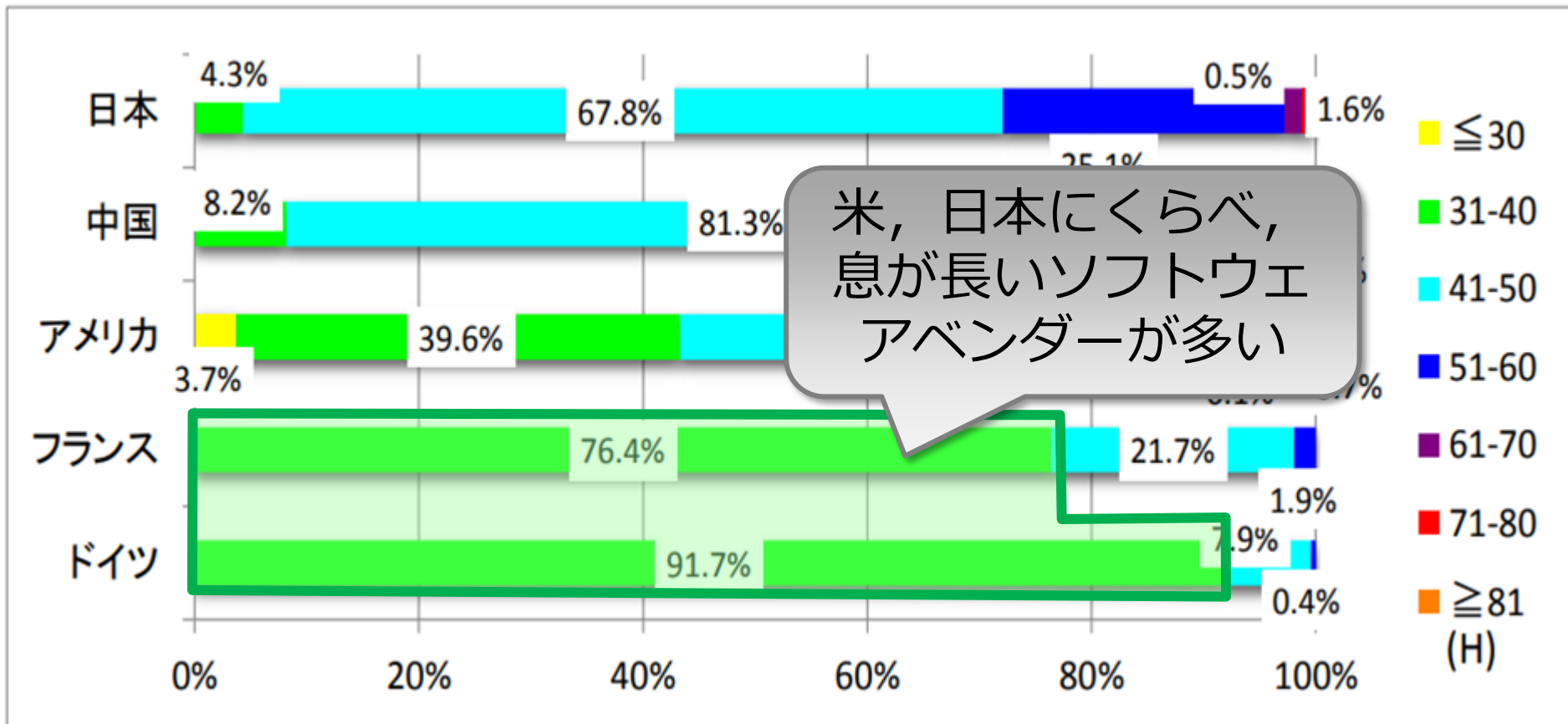
原則、全項目の記載をお願いします



留意点 1	電話番号はなるべく携帯電話の番号を記入してください
留意点 2	症状は「13.その他」に記入してください
留意点 3	「年齢」を必ず記入してください
留意点 4	重症化リスクあり、かつ、新型コロナ治療薬の投与が必要な方、重症化リスクあり、かつ、コロナり患により新たに酸素投与が必要な方 「13.その他」に「0」を記入
留意点 5	妊婦「12.妊娠」にチェックを記入
留意点 6	入院を要する「入院の必要性」に「有」と記入

神奈川県公式サイト

欧米と日本の労働時間(週平均(h))



出典 2016年 IPA/同志社大学

日本のソフトウェア技術者の生産性及び処遇の向上効果研究 より

開発ドキュメントによるQCDコントロールの課題

機械，工作機の設計は，材料，機構，構造で品質指標がほぼ定まる。
システム構築の場合，ドキュメントでQCDは確保できるのか？

カテゴリ	フェーズ	主な開発ドキュメント
PJ管理	立上げ, 計画, 実行, 終結	実施決裁書, プロジェクト計画書, 開発品質計画書, リスクジスター, 各フェーズ完了判定
開発	要件定義	現行稼働システム一覧, システム構成図, 機能関連図, DFD, 画帳一覧, 画面イメージ, 概念データモデル, セキュリティ要件, メニュー体系
	基本設計	機能一覧, 画帳一覧詳細, 画帳レイアウト, バッチ一覧, システムインタフェース一覧, システム方式, アプリ処理方式, 論理データベース定義, CRUD定義, 権限定義, 非機能定義, データ移行計画書
	詳細設計～単体テスト	内部処理フロー, プログラム管理台帳, リリース記録, 単体テスト仕様兼報告書
	結合テスト	結合テスト仕様書兼報告書, 品質分析報告書, 障害票 変理台帳, 障害管理, キャパプラ, リソース計測, インターネットQoS
	セキュリティアップデート	実施計画書, システム停止連絡票, セキュリティパッチ管理台帳
	変更, 改修	各種手順書(NW, サーバ, PC配付, バックアップ・リストア, 構成変更, プログラム変更, 特権ID申請, 利用者申請)

開発ドキュメントの適正量とは

IPA/SEC ソフトウェア開発分析データ集2022 抜粋

[ページ/KSLOC]					[件/KSLOC]								
工程	N	最小	P25	中央		N	最小	P25	中央	P75	最大	平均	標準偏差
基本設計	112	0.57	5.05	10.62									
詳細設計	112	0.17	6.34	13.21	結合テスト(テストケース)	665	0.0	22.2	51.7	120.1	100,000.0	484.9	4,821.2
					総合テスト(テストケース)	620	0.0	6.4	16.0	40.7	219,000.0	653.9	9,799.1

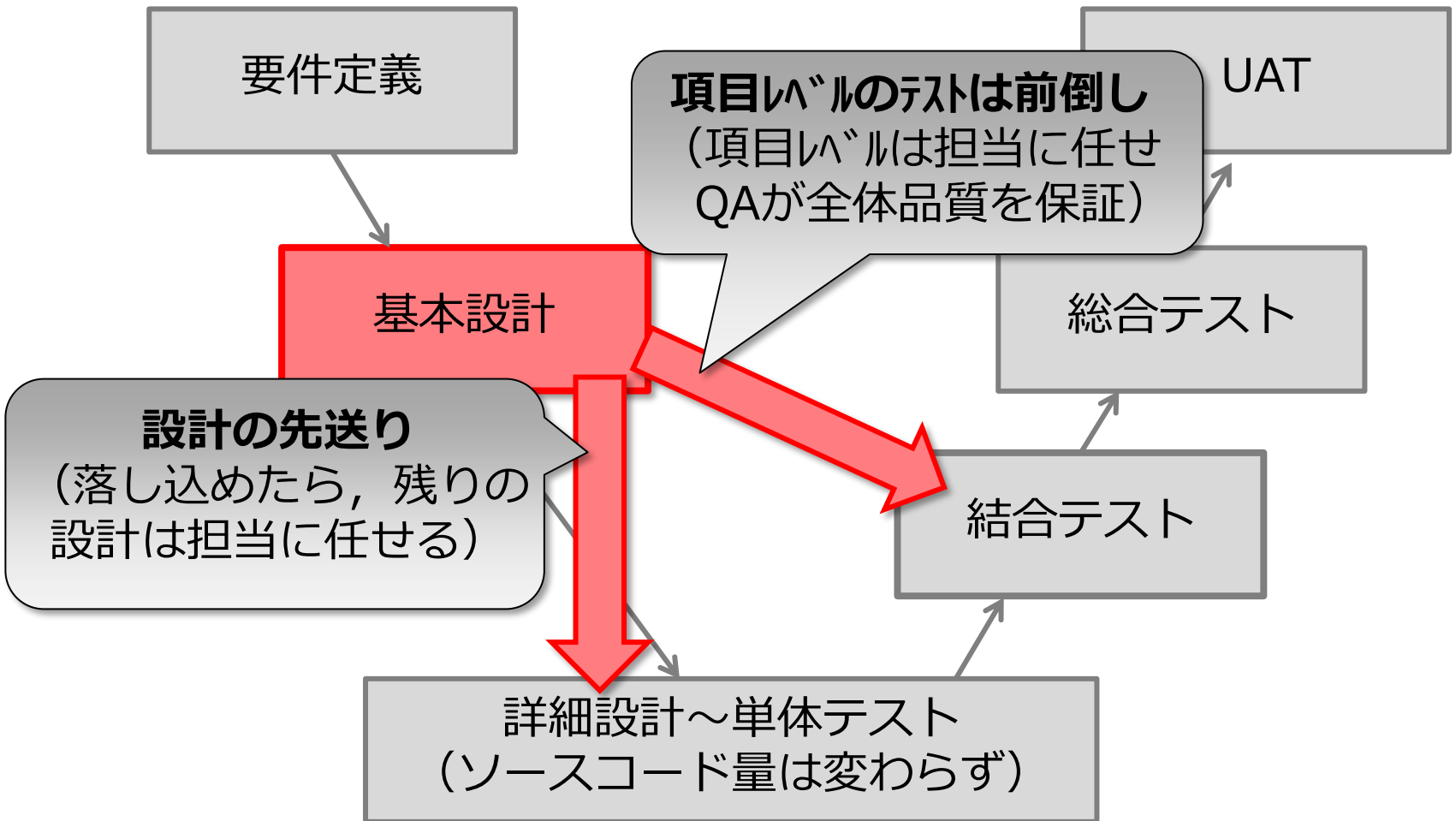
例えば、開発コスト1億円、工期12ヶ月は、ソースコード500KSLOC相当。
基本設計書は5,310頁、結合テスト仕様書は862頁（1頁30件換算）。
10cm箱ファイルで6箱相当。



- 1画面いくら、1バッチいくらで見ると多くはないか
- ソースコードを印刷した量（約7,000頁）と比べてどうか。
- 迫る納期の中で、書かないと進まないのか

V字モデルに基づくドキュメント削減アプローチ

「開発メンバーへの落とし込み」に最低限必要な基本設計書を見極め、他のアウトプットを絞り込む



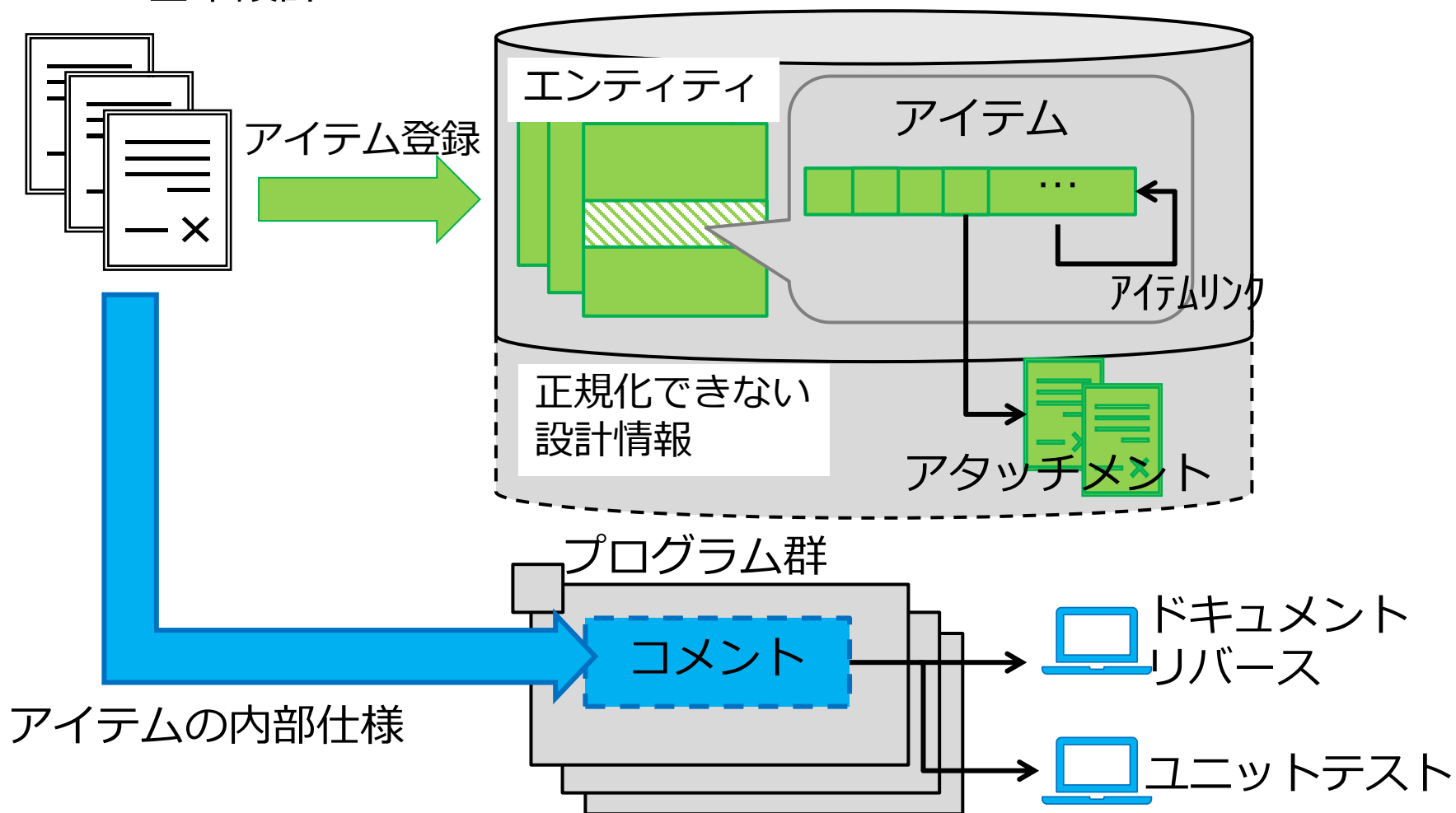
※V字モデルとはIPA/SEC 共通フレームにおいて各プロセスのインプットや相関性を図式化したモデルのことである。

メタモデルによる設計書のデータベース化

設計情報をメタモデルに基づいてデータベース化するものと、
担当者に任せるものに仕分け、任せるものは、情報量を減らさずに、
プログラムに追い出したり、モックアップで表現する。

AsIs : 基本設計

ToBe : メタモデル



MDAツールによるアイテム管理

localhost:59338/mda.html

MDAツール(試作)

システム名: プロマネ リバースエンジ テストFW

エントティツリー

- 要件定義
- 基本設計
 - インポート
 - エクスポート
 - リロード
 - 全てリリース
- 運用・保守
- 組合せ情報
 - 機能とDB
 - 進捗と要員管理
 - 結合テスト
 - 総合テスト

アイテムリスト

	概要	更新日付	担当
0	機能 A	This is a sample task description.	2021-12-1...
0	機能 A	This	2021-12-1...
0	機能 A	This	021/01/01 平井
0	承認依頼一覧	申請	2021-12-0...
0	承認一覧	審査	2021-12-0...
0	申請	審査	2021-12-0...
0	審査・承認	申請書を参照モードで開き、審議す	2021-12-0...
0	Aras In Basket	審査・承認依頼が登録される。	2021-12-0...
0	採番マスク	申請書に示すコード(画面番号や見	2021-12-0...
0	申請書マスク	申請書種別毎に申請書のテンプレ	2021-12-0...

アタッチリスト

2009_010.jpg 2009_010.jpg

© 2021 IHI Scube Co.,Ltd.

localhost:59338/mda.html#

画面エンティティのアイテム例

MDAツール(試作)

進捗・コスト管理 品質管理 ドキュメント出力

システム名:

サブシステム名:

メタモデル

画面エンティティ

画面アイテム

編集中	機能名	画面名	規模	難度	イベント	更新日
	起票	進捗一覧			一覧表示	
	起票	申請済一覧			一覧表示	
✓	起票	申請			申請	
	起票					
	申請書登録					
	申請書登録					
	申請書登録					
	申請書登録					
	権限マスタ					

申請書登録.jpg

ワークフロー

No.	項目名	項目種別	必須
01	カテゴリ	コンボボックス	○
02	申請書名	コンボボックス	○
03	タイトル	文字列	○
04	コード	ダイアログ	○
05	申請概要	テンプレート	○
06	添付ファイル	リスト	
07	備考	テキストエリア	
08	コード	ボタン	
09	審査	ボタン	
10	承認	ボタン	
11	削除	ボタン	
12	流用	ボタン	
13	OK	ボタン	
14	CANCEL	ボタン	

画面イメージ

機能とDBエンティティのアイテム例

機能名	画面名	イベント	テーブル	CRUD	更新日	担当
起票	進捗一覧	一覧表示	申請書	R	2023/07...	平井
起票	申請済一覧	一覧表示	申請書	R	2023-08...	

機能とDB
エンティティ

機能名	画面名	イベント	テーブル	CRUD	更新日	担当
起票	進捗一覧	一覧表示	申請書	R	2023/07...	平井
起票	申請済一覧	一覧表示	申請書	R	2023-08...	
起票	申請	申請	申請書マスタ	R	2023/07...	平井
起票	申請	申請	申請書	C	2023/07...	平井
起票	流用申請	流用申請	申請書		2023/07...	平井
申請書登録	登録	開発・生産	申請書マスタ	C	2023/07...	平井
申請書登録	登録	財務	申請書マスタ	C	2023/07...	平井
申請書登録	登録	総務	申請書マスタ	C	2023/07...	平井

結合テストエンティティのアイテム例

機能名	画面名	イベント	ケース	内容	テーブル	更新区分
起票	申請	申請	A01-01	申請	申請書マスタ	R

ディフェクト	完・未完	更新日付	担当
		2021/01/01	平井

結合テスト
エンティティ

編集	機能名	画面名	イベント	ケース	内容	テーブル	更新区分	ディフェクト	完・未完	更新日付	担当
	起票	申請	申請	A01-01	申請	申請書マスタ	R			2021/01/01	平井
	起票	申請	申請	A01-02	申請	申請書	C			2021/01/01	平井
	起票	流用申請	流用申請	A01-03	流用申請	申請書	CR			2021/01/01	平井
	起票	進捗一覧	一覧表示	A01-04	一覧表示	申請書	R			2021/01/01	平井
	起票	進捗一覧	一覧表示	A01-05	権限フィル...	申請書	R			2021/01/01	平井
	起票	申請済一覧	一覧表示	A01-06		申請書	R			2021/01/01	平井

MDAにおける3つのアイテム

開発ドキュメント	エンティティ名	アイテムの登録例	主なプロパティ	主なアタッチメント
基本設計書	機能	開発機能の一覧	機能名, 概要	機能概要, DFD
	画面	開発画面の一覧	機能名, 画面名, 規模, 難易度, イベント, 指摘件数	遷移図, 画面イメージ, 項目定義, メッセージ一覧
	帳票	開発帳票の一覧	機能名, 帳票名, 規模, 難易度, 指摘件数	帳票イメージ, 項目定義, 指摘事項
	バッチ	開発バッチの一覧	機能名, バッチ, 規模, 難易度, 指摘件数	入出力I/F(項目レベル), ケリー一覧, 指摘事項
	外部IF	他システム間のインターフェース	機能名, IF名, ターゲットシステム, 入出力, 連携手段	入出力I/F(項目レベル), ケリー一覧, 指摘事項
結合テスト仕様書	権限	ロールとグループの関係	ロール, グループ, 備考, 指摘件数	権限内容の捕捉, 職制情報, 現行仕様書
	機能とDB	機能とDBの関係	機能名, 画面名, イベント, テーブル, CRUD	-
	機能と担当者	機能と担当者の関係	機能名, 画面名, 帳票名, バッチ, IF, 付帯作業, 予定, 実績, 工数, 担当者	-
	機能とPG	機能とプログラムの関係	機能名, 画面名, イベント, 処理タイプ, PG名, 担当者	プログラム処理方式設計書
結合テスト仕様書	結合テスト	画面別イベントの一覧	機能名, 画面名, イベント, ケース, 内容, テーブル, 更新区分, デフォルト, 完了日, 担当者	テスト証跡, 横展メモ

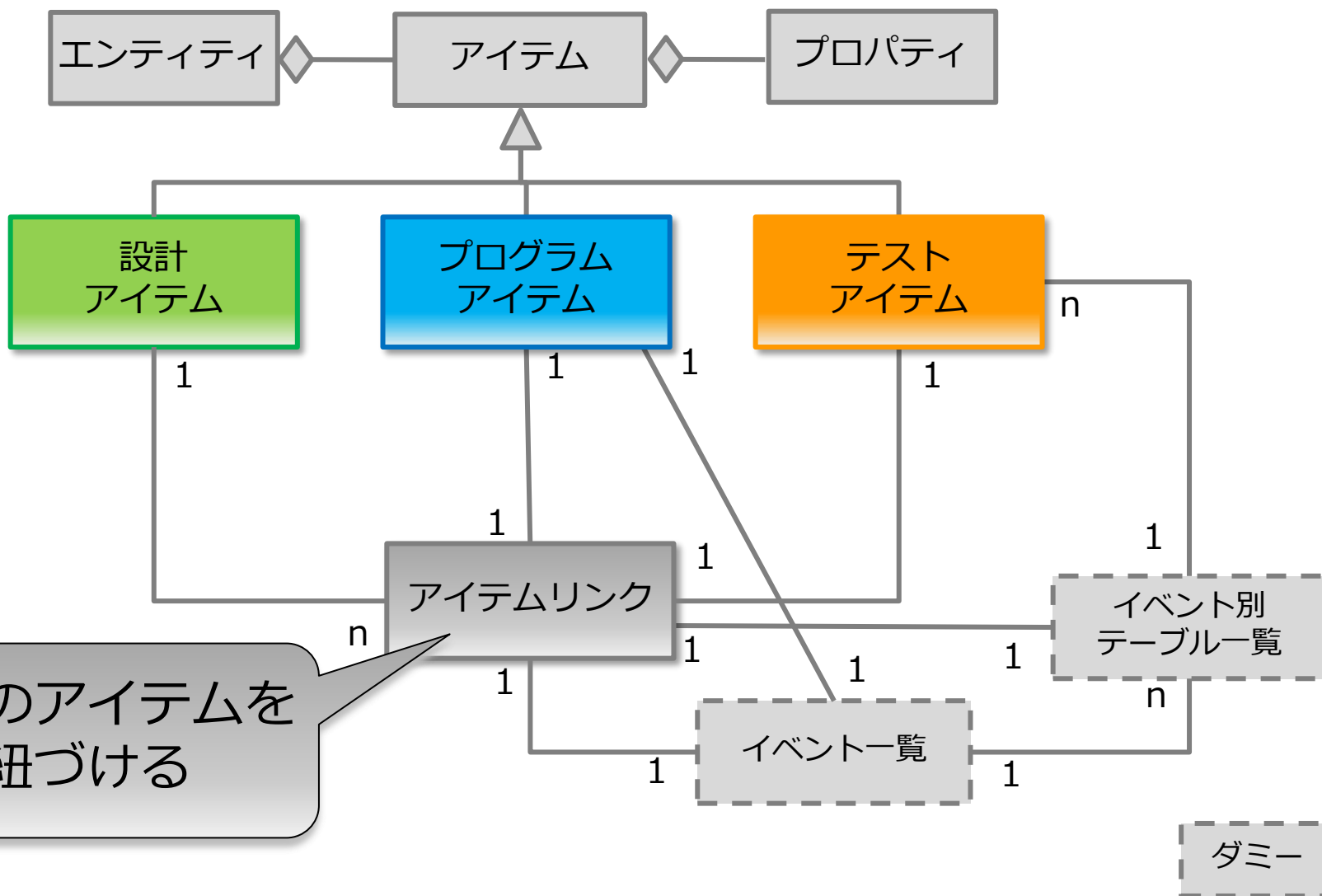
設計アイテム

プログラムアイテム

テストアイテム

アイテムの構造

設計アイテムからプログラムアイテムとテストアイテムを導く。プログラムアイテムで単体テスト，テストアイテムで結合テストを実施。



3つのアイテムを
紐づける

テストアイテムリンクの目的

機能とDBの
設計アイテム

アイテム
リンク

プログラム
アイテム

結合テストの
テストアイテム

機能名	画面名	イベント	テーブル	更新区分	更新日	担当
申請書登録	改正		申請書マスタ	C	2023/7/19	平井
申請書登録	登録	営業	申請書マスタ	C	2023/7/19	平井
申請書登録	登録	開発・生産	申請書マスタ	C	2023/7/19	平井

設計アイテムをもとに
テストアイテムを登録
する

機能名	画面名	イベント	ケース	内容	テーブル	更新区分	デフォクト	完・未完	更新日付	担当
申請書登録	改正		C04-33		申請書マスタ	C	0	完	2023/7/19	平井
申請書登録	登録	営業	C04-22	取引基本契約	申請書マスタ	C	2	完	2023/7/19	平井
申請書登録	登録	営業	C04-23	個別契約	申請書マスタ	C	1	完	2023/7/19	平井
申請書登録	登録	営業	C04-24	業績	申請書マスタ	C	0	完	2023/7/19	平井
申請書登録	登録	営業	C04-25	NDA	申請書マスタ	C	0	完	2023/7/19	平井
申請書登録	登録	開発・生産	C04-26	見積り	申請書マスタ	C	2	完	2023/7/19	平井
申請書登録	登録	開発・生産	C04-27	調達	申請書マスタ	C	3	完	2023/7/19	平井
申請書登録	登録	開発・生産	C04-28	研修、教育	申請書マスタ	C	1	完	2023/7/19	平井

プログラムアイテムリンクのメリット

(1) コメントによるドキュメントリバーース

```
* If many entries are to be made into a Hashtable,
* creating it with a sufficiently large capacity may allow the entries
* to be inserted more efficiently than letting it perform automatic
* rehashing as needed to grow the table.
*
* This example creates a hashtable of numbers. It uses the names of
* the numbers as keys:
*
* <pre> {@code
*   Hashtable<String, Integer> numbers
*     = new Hashtable<String, Integer>();
*   numbers.put("one", 1);
*   numbers.put("two", 2);
*   numbers.put("three", 3);}</pre>
*
* <p>To retrieve a number, use the following code:
*
* <pre> {@code
*   Integer n = numbers.get("two");
*   if (n != null) {
*     System.out.println("two = " + n);
*   }</pre>
*
* <p>The iterators returned by the iterator method of the
```

OpenJDK: Hashtableの
コメント記述

OpenJDK 19 / Hashtable — DevDocs

Search...

Hashtable

- Hashtable.clear()
- Hashtable.clone()
- Hashtable.compute()
- Hashtable.computeIfAbsent()
- Hashtable.computeIfPresent()
- Hashtable.contains()
- Hashtable.containsKey()
- Hashtable.containsValue()
- Hashtable.elements()
- Hashtable.entrySet()
- Hashtable.equals()
- Hashtable.get()
- Hashtable.hashCode()
- Hashtable<K, V> Hashtable()
- Hashtable<K, V> Hashtable(int initialCapacity)
- Hashtable<K, V> Hashtable(int initialCapacity, float loadFactor)
- Hashtable<K, V> Hashtable(int initialCapacity, float loadFactor, boolean allowSameKey)
- Hashtable.isEmpty()

Show more... (1176)

To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the `hashCode` method and the `equals` method.

An instance of `Hashtable` has two parameters that affect its performance: *initial capacity* and *load factor*. The *capacity* is the number of *buckets* in the hash table, and the *initial capacity* is simply the capacity at the time the hash table is created. Note that the hash table is *open*: in the case of a "hash collision", a single bucket stores multiple entries, which must be searched sequentially. The *load factor* is a measure of how full the hash table is allowed to get before its capacity is automatically increased. The initial capacity and load factor parameters are merely hints to the implementation. The exact details as to when and whether the rehash method is invoked are implementation-dependent.

Generally, the default load factor (.75) offers a good tradeoff between time and space costs. Higher values decrease the space overhead but increase the time cost to look up an entry (which is reflected in most `Hashtable` operations, including `get` and `put`).

The initial capacity controls a tradeoff between wasted space and the need for `rehash` operations, which are time-consuming. No `rehash` operations are performed if the initial capacity is large enough to hold all entries the `Hashtable` will contain.

If many entries are to be made into a `Hashtable`, creating it with a sufficiently large capacity may allow the entries to be inserted more efficiently than letting it perform automatic rehashing as needed to grow the table.

This example creates a hashtable of numbers. It uses the names of the numbers as keys:

```
Hashtable<String, Integer> numbers
= new Hashtable<String, Integer>();
numbers.put("one", 1);
numbers.put("two", 2);
numbers.put("three", 3);
```

You're browsing the OpenJDK 19 documentation. To browse all docs, go to devdocs.io (or press esc).

生成されたJavaDoc

※Java以外はDoxygenやJSDocを活用

(2) アサーションによるユニットテスト

The image shows a Visual Studio interface with three main panels:

- ソースコードパネル (Source Code Panel):** Displays the code for the `CreateBTree()` method. The assertions are highlighted with a red box:

```
[TestMethod]
10 個の参照
public void CreateBTree()
{
    var btree = new BTree<int, int>(Degree);

    Node<int, int> root = btree.Root;
    Assert.IsNotNull(root);
    Assert.IsNotNull(root.Entries);
    Assert.IsNotNull(root.Children);
    Assert.AreEqual(0, root.Entries.Count);
    Assert.AreEqual(0, root.Children.Count);
}
```
- テスト実行 (Test Execution):** A large green arrow points from the code to the test results panel.
- テスト結果パネル (Test Results Panel):** Shows the execution of `BTree.UnitTest` with 7 successful tests. The test names are highlighted with a red box:

テスト	期間	特徴
✓ BTree.UnitTest (7)	31 ミリ秒	
✓ BTree.UnitTest (7)	31 ミリ秒	
✓ BTreeTest (7)	31 ミリ秒	
✓ CreateBTree	15 ミリ秒	
✓ DeleteNodes	14 ミリ秒	
✓ DeleteNonExistingNode	< 1 ミリ秒	
✓ InsertMultipleNodesToSplit	1 ミリ秒	
✓ InsertOneNode	< 1 ミリ秒	
✓ SearchNodes	1 ミリ秒	
✓ SearchNonExistingNode	< 1 ミリ秒	

■ 評価モデル

- ・ 過去 P J から MDA を構築し、品質データを比較
- ・ 過去 P J は匿名化が必要であるため評価モデルを策定
- ・ 比較する品質データは基本設計書と結合テスト仕様書

■ 評価観点

評価観点	評価方法
開発ドキュメントが MDA で減るか	評価モデルの設計書頁数とテスト件数を比較する
基本設計の先送りは品質に問題ないか	設計アイテムとテストアイテムを比較して抜け漏れを確認する
基本設計の変更に対し生産性に問題はないか	設計アイテムの追加、変更、削除に対してテストアイテムとの関連付けを確認する

評価モデル

- 対象システム名：ワークフローシステム
- 開発コスト：¥5,000,000（基本設計～総合テスト）
- 開発期間：5ヶ月
- 成果物：

プログラム：画面(7本), 帳票(0本), バッチ(1本), インターフェース(1本)

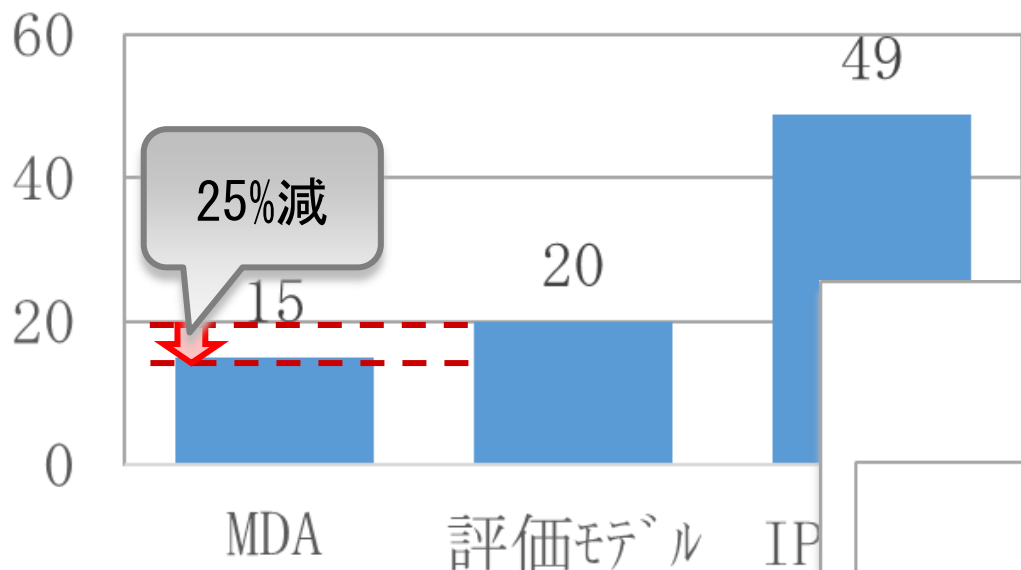
開発ドキュメント：基本設計書, 結合テスト仕様書兼報告書

【品質・工数の実績データ】

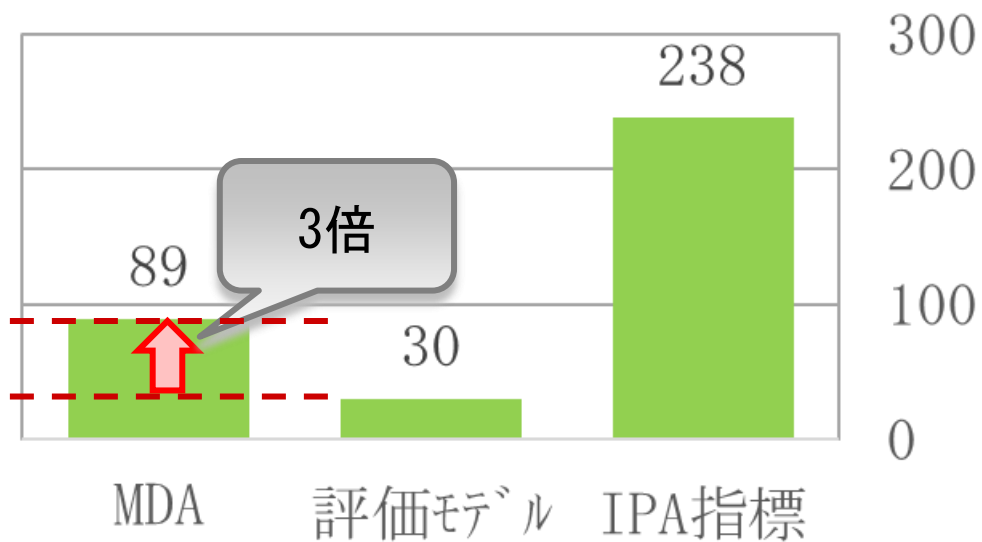
工程	基本設計	詳細設計～単体テスト	結合テスト	総合テスト
工数(h)	160.0	600.0	80.0	80.0
設計書(頁)	20	5	10	10
テスト件数	-	40	30	5
バグ件数	-	20	7	2
指摘件数	20	20	5	2
KSLOC	-	4.6	-	-

評価 1 : 設計書とテスト件数の比較

基本設計書の页数比較



結合テストの件数比較



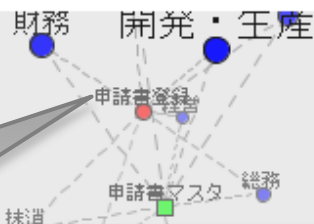
※IPA指標はIPA/SEC ソフトウェア開発分析データ集2022より引用

評価 2 : テスト漏れの有無

Morcego

設計アイテム

設計アイテムの
プロパティを
点線で結線

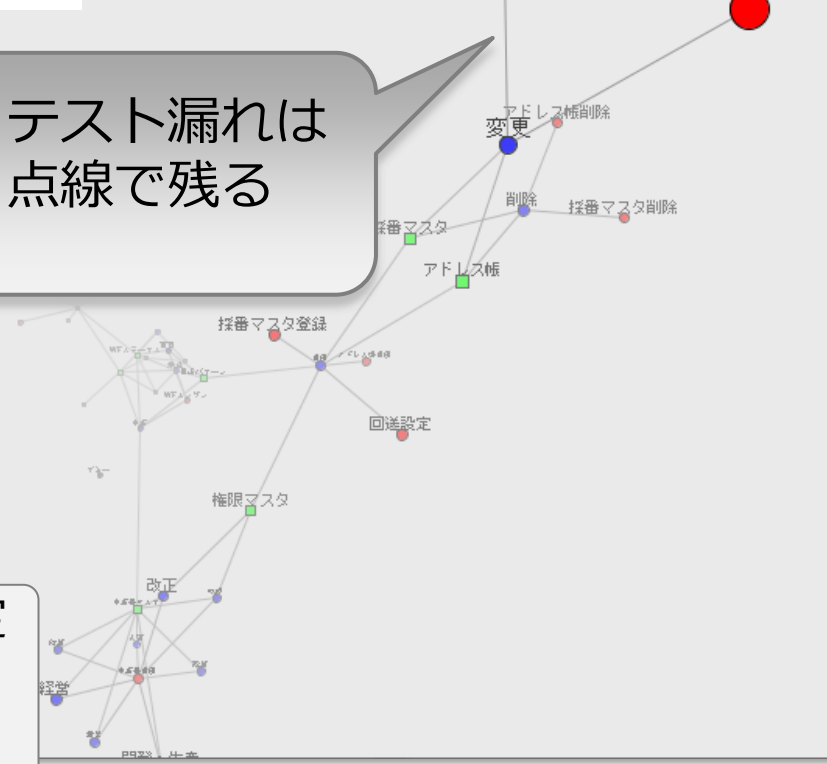


Morcego

テストアイテム

テスト漏れは
点線で残る

アドレス帳変更
採番マスター



- 【凡例】
- イベント
 - 画面・バッチ
 - テーブル
 - テスト予定
 - テスト済

評価3：設計変更への追従性

変更後の設計アイテム

アイテムキー	設計属性		
変更アイテム			
追加アイテム			
削除アイテム			



更新は自動反映



変更前のテストアイテム

アイテムキー	設計属性		
変更アイテム			
削除アイテム			

マッチングで
追加・削除を識別

設計変更の有無	設計アイテム	テストアイテム	アイテムリンク	影響検索
追加	○	×	×	可
変更	○	○	○	可※
削除	×	○	○	可

※アイテムの更新日付で比較を要する

評価結果に対する考察

■ 得られた知見

- MDA手法により，設計書量が減り，テスト件数も減る見通しを立てることができた。
- アイテムリンクにより，テストの網羅性や設計変更への追従性が確保できることが分かった。

■ 前提条件，課題

- 訴求ポイントを絞るため，評価モデルや検証範囲が限定的．例えば実PJで実用面での深堀など。
- 担当に落とし込み，担当分を積上げるPJに有効で，ウツアなど設計と実装の担当が分かれるPJは不向き。

成果（まとめ）

■ 成果

- ・ 設計書をコンパクトにすることで、テスト件数が減り、生産性も高まる手法「MDA」を発見できた。
- ・ MDAツールは、設計情報やテスト仕様をデータベース化し、抜け漏れを防止できるので、QCD見える化に有効である。

■ 展望

QCD見える化は、DXでも注目される大規模Agileへの対応や大規模PJのQCD対策(リスク早期発見, 未然防止)に不可欠な技術である。
品保の活動の中でさらに育てていきたい。

ご清聴ありがとうございました。

ソフトウェア品質シンポジウム
