特集 ソフトウェアレビュー／ソフトウェアインスペクションと欠陥予防の現在

6 テストエンジニアが参加するアジャイルインスペクション

永田 敦 ソニー（株）B2B ソリューションズ 品質戦略部

アジャイルインスペクション[1]は，Tom Gilb が提案したソフトウェアドキュメントインスペクション手法である．サンプリングされた一部のドキュメントを一定時間内に定められた手順に従ってインスペクタが欠陥を指摘し，品質メトリクスを得る．一般的なインスペクションではインスペクタが網羅的に欠陥を指摘し，その指摘をもとに作成者に修正を促すが，アジャイルインスペクションでは，インスペクション，メトリクスによる判定，修正作業を品質基準を満たすまで繰り返す．本稿では，アジャイルインスペクションのインスペクタにテストエンジニアを加えた場合の効果と実施例を述べる．これまでパースペクティブベースドリーディング等の手法において，テストエンジニアの観点による上流インスペクションの効果は確認されているが，アジャイルインスペクションにおいてもその効果を示唆する結果が得られた．具体的には，テストエンジニアの参加により過去のトラブル事例，顧客の実行環境，顧客の想定外の利用方法を踏まえたインスペクションによる潜在的欠陥の早期発見である．

▶アジャイルインスペクションと従来のインスペクションの違い

従来のインスペクションの目的は，ドキュメントの欠陥を発見することであるのに対し，アジャイルインスペクションの目的は，高品質のドキュメントをはじめから生産することである．この背景は次の 3 つである．

- 対象ドキュメントの欠陥を網羅的に抽出することの限界の認識
- ドキュメントを書く側の質を上げていくことの必要性
- 上流で欠陥を防ぎ品質向上コストを下げることの必要性

従来のインスペクションは，そのプロセスを定義し，ルールを決めて，より多くの欠陥の抽出をしてドキュメントの質を上げようとしてきた．ドキュメントを動作するプログラムと置き換えると，インスペクションは欠陥を見つけるという目的の上でソフトウェアテストと同じ機能を果たしている．また，その性質も同様であり，テストでのバグ抽出能力は，テストのポリシー，戦略，分析設計実装などのプロセスや，テスターの知識，技量，経験により変わるようにインスペクションでの欠陥抽出能力も，そのプロセスや，インスペクタの知識，技量，経験により変わる．

テストによって，ある程度のバグを抽出し市場への流出を防ぐことができる．抽出されたバグは開発担当者に報告され，デバッグされ，確認テスト，回帰テストという手戻りプロセスを通る．抽出できなかったバグは，市場に出ていく．もし，ソフトウェアコードの品質が高ければ，この手戻りに対するコストを抑えることができ，また，流出のリスクが減る．同様に，インスペクションの場合も，もしドキュメントの品質が元から高ければ，インスペクションを行うコストやそれに伴う手戻りのコストが減り，インスペクションで抽出できない欠陥の数自身が減り，製品リスクが元から抑えられることになる．大もとのドキュメント自体の品質を上げていくことが，品質を向上する上で最も効率が良くなる．アジャイルインスペクションはまさにこのドキュメントの品質を元から高める目的から考え出されたものである．Tom Gilb は，これを “The Prevent — don’t Clean Principle” と呼んでいる[2]．つまり，アジャイルインスペクションでは，ドキュメントの欠陥を網羅的に探しそれを直していくことに注力するのではなく，一部分の品質を測り，他の部分の欠陥を予防することに重点をおく考えである．

品質の高いドキュメントとは，そのドキュメントに期

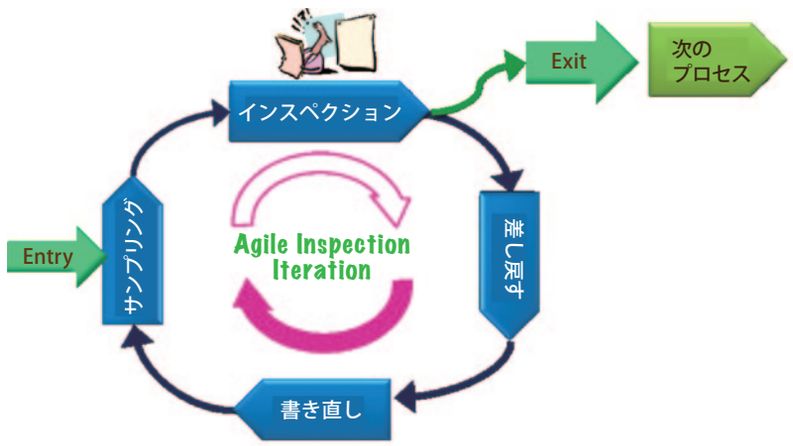

図-1　アジャイルインスペクションイテレーション

待されている目的に対し必要十分な記述がされているものをいう．そのようなものをはじめから書くためには，その書き手自身の書く技量が適当な品質になっていなければならない．アジャイルインスペクションが注力するところは，その書き手から生産された成果物の品質を測定することにある．

▶アジャイルインスペクションのプロセス

アジャイルインスペクションは以下のプロセスからなる[1]．

1. インスペクタ(チェッカー)を決める
2. ルール(インスペクションの観点)を決める
3. 合否判定のための欠陥密度の閾値を決める
4. 実施時間を決める
5. ドキュメントをサンプリングする
6. インスペクタにルールなどの説明をする
7. サンプルをインスペクションする
8. メトリクスを分析する
9. 欠陥密度の閾値により合否を判定する

これらのうち，5，6，7，8，9をアジャイルインスペクションイテレーションと呼ぶ(図-1)．

これらの手続きの簡単な説明を以下に述べる．

1. インスペクタ(チェッカー)を決める

1人のアジャイルインスペクションリーダーがインスペクションをガイドしていく．インスペクタは通常2名以上の少人数で行う．

2. ルール(インスペクションの観点)を決める

リーダーはルールを決める．ルールに逸脱しているものを欠陥とする．また，主要欠陥(メジャー欠陥)を定義し，その数を計測する．初めは3つ程度のルールで行う．文献1)では以下が推奨されている．

Clarity　　：十分明確か
Unambiguous　：曖昧でないか
Completeness　：完全か

そして，要求仕様に対しては，

No optional design：設計要素が入っていないか

が付け加えられることがある．

これらルールは，テーラリング☆1の対象となる．

3. 合否判定のための欠陥密度の閾値を決める

リーダーは閾値を決める．

インスペクションにおけるメトリクスはドキュメントの品質を測るというより，インスペクションを行った実績を表しているものが一般的である．しかし，アジャイルインスペクションでは，その欠陥の密度で品質を推定し，終了条件を決めていることになるので，合否を決める欠陥密度の閾値をどう決めるかが，アジャイルインスペクションの大きなカギになる．Tom Gilbは，これについて次のように述べている．最初のうちは，その閾値を10個／論理ページと決める．仕様書の“書き方”の改善をしながら，その閾値を徐々に上げていき，目標としては，1個／論理ページになるまで実施するとよいと述べている[1]．もちろん，この目標はあくまでも指針で，実際は，現状の仕様書の品質や，書き方の改善の施策など，現場の状況によりテーラリングして責任者が決めていくものである．

4. 実施時間を決める

リーダーはインスペクションの実施時間を決める．

1論理ページ，300語に対して10分から15分を目安にする．一般に時間を短くすると欠陥の検出数は小さくなる．また，ルールを増やすとその分時間が必要に

☆1 CMMIでは，組織・企業が業務の基本として定めた標準プロセスや開発標準などを手直しして，個別のプロジェクトや顧客の要求に合わせて実用的な標準(手順・成果物・指標など)を作成・実行することである．この場合は，成果によってパラメータをその組織に合わせて変えていくことを意味している．

なる．

5. ドキュメントをサンプリングする

リーダーは対象ドキュメントからインスペクションの対象部分を決定する．

英文で 300 語程度を 1 論理ページとする．図などが入る場合もあるので，実際のドキュメント上では 1 ページあるいは 2 ページが対象となる．

6. インスペクタにルールなどの説明をする

リーダーは，インスペクタに以下のことを説明する．

アジャイルインスペクションで使うルールの説明，実施時間，欠陥をどのように記録するか，メジャー欠陥の説明（要求仕様書であれば，設計項目が書かれている場合別途数える）など．

7. サンプルをインスペクションする

インスペクタは，決められた時間内に読んでチェックしていく．全部を読み終える必要はない．測定しているのは，全体の欠陥数ではなく，論理ページあたりの欠陥数であるからである．全部読むようプレッシャーをかけると，読む速度が速くなるなどして，逆に抽出率が悪くなる場合がある．

8. メトリクスを分析する

終了時刻になったらインスペクションを止める．

各インスペクタに，それぞれがチェックしたところの文章の語数を数える．語数は文字数ではなく，ことばとして意味のある文字列を 1 つとして数える．抽出した欠陥と調べた語数をリーダーに報告する．

9. 欠陥密度の閾値により合否を判定する．

リーダーは，メトリクス分析によって得られた欠陥密度から次のいずれかを決定する．

合格　：分析結果の欠陥密度が閾値より低い場合，次のプロセスに進む．

不合格：欠陥密度が閾値より高い場合，ドキュメントの質の改善のために作成者に差し戻す．リーダーは，差し戻す際に，インスペクタの協力も得て，欠陥のリストアップ，原因などをまとめて，気づきが分かるように報告する．

■ 全体品質の推定

アジャイルインスペクションは，ドキュメントの品質を測ると述べてきた．一方これは，書き手の品質を測っていると考えることもできる．その書き手の 1 ページをインスペクションし欠陥を抽出する過程で，書き方のスタイル，論理的思考，文書化能力，癖などが分かり，他のページにおいても同じ欠陥を作り出すリスクを推定することができる．したがって，書き手が変われば書き手ごとにアジャイルインスペクションイテレーションを行うほうが推定の精度を高められることが期待される．当然，差し戻される際には指摘部分だけを直すよう指示するのではなく，その欠陥を入れ込んだ原因，文書化の方法，癖，考え方を指摘して，他のページや今後書かれるページも改善していくように指摘しなければならない．なお，指摘するものは，他のレビュー方法と同様に，書き手自身に対する批判であってはならない．

■ Review Early

レビューは早いうちに“できているところ”からやる．

従来のインスペクションは，できあがったドキュメントを事前配布してインスペクタに読んでもらい，問題点を指摘していく．アジャイルインスペクションは，ドキュメントから 1 ～ 2 ページを抽出しインスペクションの対象とする．つまり，ドキュメントは完成していなくても，最初の 1 章の 1 節ができただけでもインスペクションを行うことができる．たとえば，要求仕様書が 2,000 ページにもなり，それを複数の書き手で書いていくとき，それぞれの担当の最初の部分ができたところで，逐次アジャイルインスペクションイテレーションを回していくことができる．ポイントは，このドキュメントのサンプリングと，早いうちにできているところからインスペクションを行うことにより，イテレーションの回転を速くできることである．これがアジャイルと名づけられた所以である．Gilb はこれを Review Early Principle と呼んでいる [2]．

▶テストエンジニアから見た欠陥指摘

Myers によるとソフトウェアテストは，欠陥を見つけるために行う破壊的なプロセスと見るべきと述べている．つまり，テストエンジニアは，欠陥を見つけることに業務としてのモチベーションを持っている（図 -2）．

また，テストエンジニア，特にシステムテストの担当者は，社内評価で発見されたトラブルや，市場でのトラブルの事例，原因とその再発防止策などに精通しており，評価する観点を多種多様に持っていることが多い．このモチベーションと観点をもって，アジャイルインスペクションに加わることにより，より多くの欠陥を抽出することが期待できる．

たとえば，図 -3 のように，要求仕様の書き手は，要求をどのように実現していくか，つまり創造するという観点が強い．一方で，テストエンジニアはそれを攻撃する観点，別の言い方をすると悲観的な観点を持ってインスペクションを行うので，より漏れのないインスペクションが期待できる．

ウォータフォールモデルの開発では，欠陥に対する修正コストは上流ほど小さく，下流に行くほど大きくな

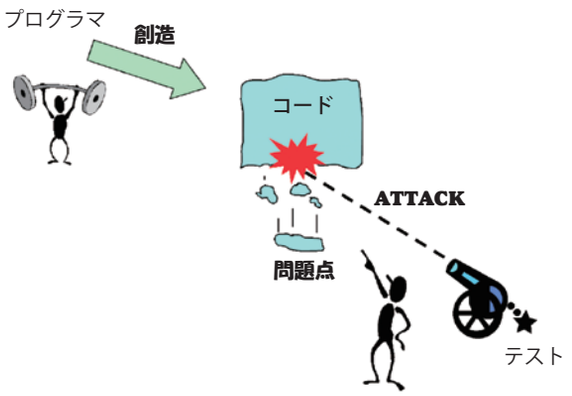

図 -2　設計者とテストエンジニアの観点の違い

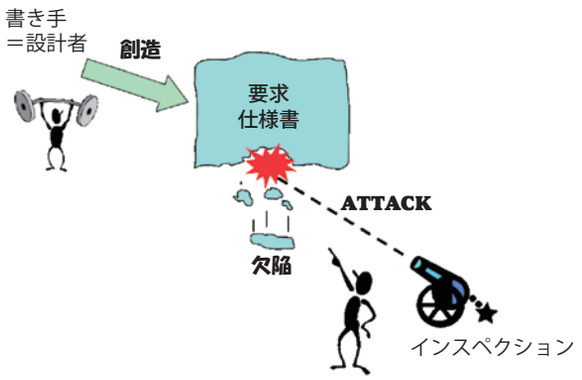

図 -3　インスペクションにおけるテストエンジニアの観点

る．開発のトータルコストを抑えるためには，その上流に位置するドキュメントである要求仕様書に対して行うことが最も効率的である．

▶アジャイルインスペクションの実施事例

■事例 1

事例として，有志によるワークショップでの結果を紹介する．対象ドキュメントはある製品の製品要求仕様書を用いた．インスペクタは組込み機器の開発者およびテストエンジニア（SQA 担当者）8 名で行った．使用したルールは，曖昧ではないこと，矛盾がないこと，論旨表現が明確なこと，設計要素が入っていないこととした．以上のルールで，後工程で致命的な問題を発生すると推測される表現をメジャー欠陥として計測した．結果を図 -4 に示す．図 -4 の横軸はインスペクタを，縦軸は 1 論理ページ（300 語）あたりのメジャー欠陥の指摘数である．表 -1 は図 -4 から次のような手順で算出したものである．

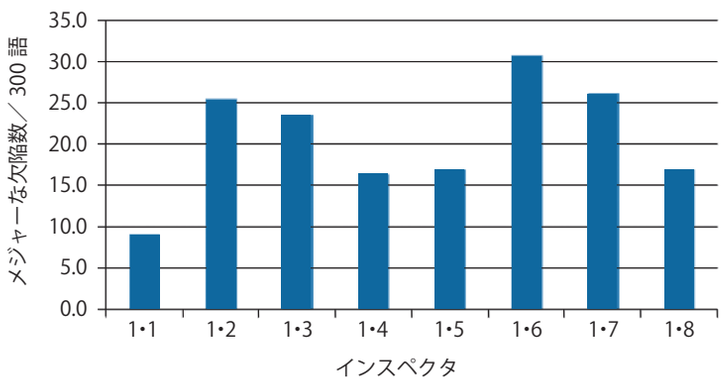

図 -4　事例 1 の結果

インスペクタの数	8 人
1 論理ページ（300 語）当たりのメジャーな欠陥数	61.60 個
欠陥抽出効率	33%
実際に存在する論理ページ当たりの欠陥数の推定数	187 個
ドキュメントの論理ページ数	15 ページ
ドキュメントのトータルの欠陥数の推定値	2,800 個
欠陥がプログラムに実装されると推測される割合	1/3
欠陥がプログラムに実装されると推測される件数	933 個
メジャーな欠陥 1 つ当たりの修正コスト（時間）	9 時間
欠陥によるトータルの修正コストの推定時間	8,400 時間

表 -1　事例 1 の結果

表 -1 中の白いセルは入力値，青は定数，赤は推定値を表している．文献 3）に示されているように，これらの定数は，Gilb がこれまで集めたデータと経験に基づくものであるが，実際にはその現場によってテーラリングしていく必要がある．

1 論理ページ（300 語）当たりのメジャーな欠陥数は，この場合，8 人のうち一番大きい数の 2 倍をユニークな欠陥数の数として推定している．この推定方法は，テーラリングの対象となる．

欠陥抽出効率は，抽出できる割合である．Gilb は，経験のないチェッカー（インスペクタ）は，1/3 しか欠陥を抽出できないことを示しており [1)]，本事例でも，初めて行うので 1/3 としてある．同文献において，十分に経験があるチェッカーは要求仕様書や設計書を対象にした場合，30％から 80％程度と，Gilb は述べている．この割合も，テーラリングの対象となる．実際に存在するメジャーな欠陥数は次の式を用いて推定する．

論理ページ当たりの欠陥数
＝抽出された欠陥数　×　欠陥抽出率

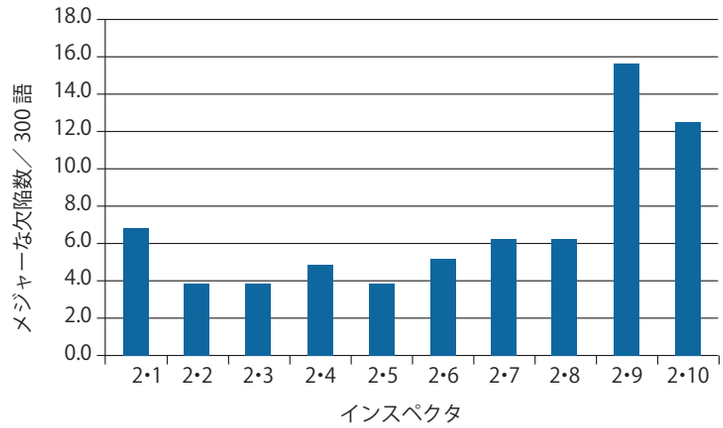

図 -5　事例 2 の結果

インスペクタの数	10 人
1 論理ページ(300 語) 当たりのメジャーな欠陥数	6.96 個
欠陥抽出効率	33%
実際に存在する論理ページ当たりの欠陥数の推定数	21.1 個
ドキュメントの論理ページ数	10 ページ
ドキュメントのトータルの欠陥数の推定値	211 個
欠陥がプログラムに実装されると推測される割合	1/3
欠陥がプログラムに実装されると推測される件数	70 個
メジャーな欠陥 1 つ当たりの修正コスト(時間)	9 時間
欠陥によるトータルの修正コストの推定時間	633 時間

表 -2　事例 2 の結果

ドキュメントの総論理ページを 17 ページとすれば，すべての欠陥推定数は 187 個となる．

ドキュメントのトータルの欠陥数の推定値
＝ 論理ページ当たりの欠陥数　×
ドキュメントの論理ページ数(1 ページ／ 300 語)

ただし，すべてのドキュメントの欠陥が具体的な不具合として実装されるとは限らない．Tom Gilb はこのうち 25% から 35% がプログラムの欠陥として実装されると推定している [1]．ここでは，欠陥がプログラムに実装されると推測される割合を 1/3 として，933 個となる．

欠陥がプログラムに実装されると推定される件数
＝ ドキュメントのトータルの欠陥数の推定値　×
欠陥がプログラムに実装されると推測される割合

この欠陥の 1 つ当たりの手戻り修正時間を 9 時間と見積もると，欠陥による総修正時間は，8,400 時間と推定される（ただし，1 つの修正で複数の欠陥が修正されることは考慮していない）．これらの推定値を，あらかじめ手順 3 で決めた閾値をもって，このドキュメントをもとに次のプロセスに移行するかを判断していく．このワークショップの参加者全員が，このドキュメントの品質では，次のプロセスには進めないと感じた．

■ 事例 2

事例 1 と同様なワークショップで，対象ドキュメントは文献 4）で使われている要求仕様書を用いた．インスペクタは事例 1 とは違う組込み機器の開発者およびテストエンジニア（SQA 担当者）10 名で行った．使用したルールは，事例 1 と同じである．

結果は，図 -5 のとおりである．図 -5 の縦軸，横軸は図 -4 と同様である．また，表 -2 も表 -1 と同様に算出した．

■ 考察

事例 1 は，インスペクタが普段従事しているドメインと同じ組込み系の製品要求仕様書を対象ドキュメントとした．開発者もテストエンジニアと同様に多くの欠陥を抽出している結果が得られた．ドメインが同じ場合は，開発者の観点と，テストエンジニアの観点が互いに補完しているので，抽出量はより多くなることが期待される．

図 -5 において，インスペクタ 2-1，2-8 と 2-9，2-10 では欠陥密度が倍以上の大きさになっている．2-1，2-8 は開発者，2-9，2-10 はテストエンジニア(SQA 担当者)である．この事例で使ったドキュメントは，IT システムのもので，インスペクタの普段の業務のドメイン（組込み系）とはドメインが異なっている．指摘記録からどのような欠陥指摘をしたかを調べてみると，開発担当はドメインの違いにより抽出がうまくできていない様子が伺えた．たとえば，あるエンジニアの分析では，“システムの起動”，“区間データの入力”，“登録を拒否する”などの記述で，“起動”，“入力”，“拒否”などの動詞に下線を引いている．これは，システムの“機能”を理解しようとしていると思われる．これは，開発者が，ドメインの違いのために，“どのように実現するか”という実装的な観点で対象を理解しようとしてしまい，欠陥が見つけられないことを示唆していると考えられる．一方，テストエンジニアは“悲観的な観点”をもって欠陥を見つけている痕跡がある．あるテストエンジニアは，キーワードに下線を引き，それに対して質問を複数書き込んでいた．たとえば，“テキストファイルに記述された飛行計画のリストを読み込み，その並び順に従って飛行計画を逐次的に入力する．”という文章に対して，“テキストファイル”に対して，“フォーマットは？”や“メディアは？”という書き込みがあり，“読み込み”に対して，“読み込み方法は？”という質問，“逐次的に入力する”に対しては，“誰が入力するのか？”という質問が書き込まれていた．

これらの質問は，テストエンジニアの観点からきているものと考えることができる．これから，たとえばファイルのフォーマットのシンタックスの異常，ファイルデータの禁則などのエラー処理などが想起される．

▶結論と今後の課題

本稿では，アジャイルインスペクションのプロセスを述べ，筆者らが実施したワークショップ形式での適用事例を紹介した．適用対象を組込み系機器の製品要求仕様書，業務系(エンタープライズ系)ソフトウェアの仕様書とし，組込み系のソフトウェア開発エンジニア，組込み系のテストエンジニア（SQA 担当者）をインスペクタとしてアジャイルインスペクションを実施した．組込み系ソフトウェアを対象とした事例では，開発エンジニア，テストエンジニアの間で特に大きな差は見られなかったが，業務系ソフトウェアを対象とした事例では，テストエンジニアのほうが多くの欠陥を指摘できた．アジャイルインスペクションの実施のための拘束時間はそれほど大きくなく，アジャイルインスペクションにテストエンジニアが参加することは比較的容易であり，その効果を示唆する結果も確認できた．今後の課題としては以下が挙げられる．

1. アジャイルインスペクションを導入するモチベーション
2. アジャイルインスペクションのパラメータのテーラリングの方法
3. それぞれのソフトウェア開発における位置づけを明確にする
4. 導入効果を測定するためにどのようなメトリクスが必要か
5. 書き手の技量を上げるための指示方法と教育
6. インスペクタ自身の技量の向上

参考文献

1) Gilb, T. : Agile Specification Quality Control : Shifting Emphasis from Cleanup to Sampling Defects, The International Council on Systems Engineering (INCOSE) (2005).
2) Gilb, T. : Engineer Your Review Process : Some Guidelines for Engineering Your Engineering Review Processes for Maximum Efficiency, http://www.gilb.com/tiki-download_file.php?fileId=143 (2007).
3) Gilb, T. : Inspection Facts : To Help You Make Good Decisions to Do Software Inspections and Reviews Properly, 日本科学技術連盟, ソフトウェア品質シンポジウム 2008.
4) 岡本博幸，内山敬太，鈴木彩子，高橋一仁，西山　茂，野中　誠：要求仕様書の特性に着目した個人レビュー手法の実験的評価，ソフトウェア品質管理研究会 第 20 年度 (2004 年度) 分科会成果報告，日本科学技術連盟 (2006).

(平成 21 年 3 月 2 日受付)

永田 敦　Atsushi.Nagata@jp.sony.com

1987 年ソニーに入社．業務用機器の組込みソフトウェア開発を手がける．現在，同社，B2B ソリューション事業本部，品質管理部門，品質戦略部において，ソフトウェアテスティングのプロセス改善に従事．