Vol.2010-SE-170 No.5
2010/11/11

セキュリティ要件のレビューにおけるチェックリストの表記方法の比較

坂東祐司[†]　森崎修司[†]　松本健一[†]

ソフトウェアレビューの代表的な支援技法としてチェックリストが使用されている．チェックリストの質問にレビューアが答えていくことでソフトウェアに含まれる欠陥を指摘できる．質問項目の中にはある項目を確認すれば他の項目を確認する必要がなくなるものがある．本稿では，セキュリティ上の問題発見を目的とし，質問項目間に依存関係があるチェックリストを用いて，4,500LOC のソースコードと要件定義書を実務者 92 名にレビューしてもらった結果を報告する．被験者を，(a)チェックリストを与えないグループ，(b)順序つきチェックリストを与えたグループ，(c)フローチャート形式で表記したチェックリストを与えたグループ，にグループ分けし，正答数と正答率を比較したところ，(a)と(b)，及び，(a)と(c)の間で正答数に統計的有意差があった．また，(b)よりも(c)のグループのほうが正答率が大きくなる結果が得られた．

A Comparison of the Notations of Checklists in Security Software Inspection

Yuji Bando,[†] Shuji Morisaki, [†] and Ken-ichi Matsumoto[†]

Checklists are used as a typical reading support for software inspection. Inspectors can detect defects by answering the questions in a checklist. In some checklists, inspectors need not answer a question if the prior question is finished because some questions have dependency other questions. In this paper, we conducted an experiment with a checklist that has several questions having dependencies. The inspection targets are 4,500 LOC source code and requirement document. Subjects are divided into three groups (a) ad-hoc group, (b) checklist of ordered questions (Guided Checklist) group, and (c) checklist noted by flowchart-like notation (Security Goal Indicator Tree) group. The number of correctly detected defects has statistical significance between groups (a) and (b). Groups (a) and (c) have also statistical significance.

1. はじめに

欠陥の早期発見を目的に，多くのソフトウェア開発プロジェクトにおいてソフトウェアレビューが実施されている．ソフトウェアレビューは，目視により要件定義書やソースコードといった中間成果物から潜在的欠陥や矛盾を発見，摘出する活動のことであり，ソフトウェアの静的解析として位置づけられている．欠陥を早期に発見することから，成果物の品質向上や開発工数の削減，開発効率の向上に効果がある [1)]．

しかし，ソフトウェアレビューは，プログラムを実行し，正しく動作するかを確認しながら進めるソフトウェアテストと異なり，進め方に明示的な手順がない．そのため，レビュー支援技法として，リーディングテクニックと呼ぶ読み進め方が多数提案されている[2)3)4)]．また，それら技法の間での評価が数多く報告されている[5)6)]．

ソフトウェアレビューの代表的なリーディングテクニックとして，チェックリストが使用されている．チェックリストは，記載された質問を参照し，対象のドキュメントを確認して欠陥を発見する技法である．また，その中には質問項目間に依存関係を持ったものも存在する．チェックリストの質問項目に依存関係があり，ある項目を確認すれば他の項目を確認しなくてもよい．

本稿では，質問項目間に依存関係を持つセキュリティ要件を調べるチェックリストを使用し，その表記方法による効果を実験を通じて評価する．たとえば，“システムにおいてユーザからの要求を認証するコードが 1 か所だけである．”という質問項目と“見つかった認証コードに対し，統一した認証方式を使用している．”という質問項目がある場合，どちらか一方の質問項目を確認することができれば，もう一方を確認することなく，セキュリティ要件が満たされていることになる．実験では，架空の書籍貸出し販売用の Web サイトのサーバサイドロジックを実装した 4,500LOC の Java ソースコードとその要件定義書を対象とし，チェックリストを与えない方法（ad-hoc），順序つきチェックリストを与えた場合（GC: Guided Checklists）[7)]，フローチャート形式で表記したチェックリストを与えた場合（SGIT : Security Goal Indicator Tree）[7)]について，欠陥指摘数，正答数を比較した．

以降，2 章では今回実験で使用したリーディングテクニックを示す．3 章ではチェックリストの表記方法と指摘欠陥の関係を調査するために実施した実験について説明する．4 章にはその結果，5 章でまとめる．

[†]奈良先端科学技術大学院大学　情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

2. チェックリストの表記方法

本章では，文献 7）に示されている表記方法の概要と具体例を示す．

2.1 Guided Checklists

GC（Guided Checklists）は，順序つきのチェックリストで導入説明と複数の質問，そのヒントから構成されている．チェックリストの質問項目の間には依存関係があり，構造化されている．図 1 にチェックリストの質問項目とヒントの例を示す．

図のようにチェックリストの質問は 3 段で構成され，上段にはどのレビュー対象を参照するか，レビュー対象の中から何を発見するかということが記述されている．中段には，レビュー対象の中でどの部分に着目し，マークを付すべきか書かれている．これにより欠陥や問題点を発見するだけではなく，レビュー対象から該当する記述を発見し，その部分に欠陥がないか調べることにより，正しく表記，実装されているかどうかを確認することができる．下段には，正しく表記，実装されていない部分（欠陥）を発見した場合に，記録する必要のある項目を示している．なお，文書の中でマークする必要がない場合，質問項目は 2 段構成になることもある．その場合，1 段目には，どの文書を参照するか，文書の中から何を発見するかということが記述され，2 段目には，欠陥を発見した場合，何を記録するのかが記述されている．また，ヒントには，質問に対するより詳細な説明や被験者が何を探せばいいのかが書かれている．

2.2 Security Goal Indicator Tree

SGIT（Security Goal Indicator Tree）は，フローチャート形式で表記されたチェックリストで，GC と同じように質問項目（指標）とヒントから構成される．図 2 に SGIT の質問項目（指標）とヒントの例を示す．SGIT の指標 1 は GC の質問項目 1 に対応している．図のように，SGIT の最上部の長方形の中には，SGIT 全体でチェックできるセキュリティ要件が記述されている．また，どの文書を参照するか，文章の中から何を発見するかといった指標は六角形で示され，指標どうしは木構造で結合されている．結合には全ての指標の内容を確認しなければならない AND（図 2 中「AND」で結合されている指標）と，いずれかを確認すればよい OR（図 2 中「OR」で結合されている指標）がある．

図 2 では，指標 1 と指標 5 は OR で結合されているので，要件「安全なパスワード通知」が満たされているかどうかを判断するには，いずれかを確認するだけで済む．一方，AND で結合された指標は，全ての内容を確認する必要がある．

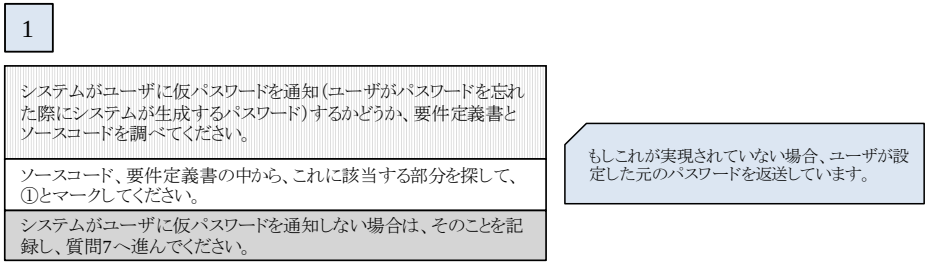

図 1　チェックリストの質問項目とヒントの例

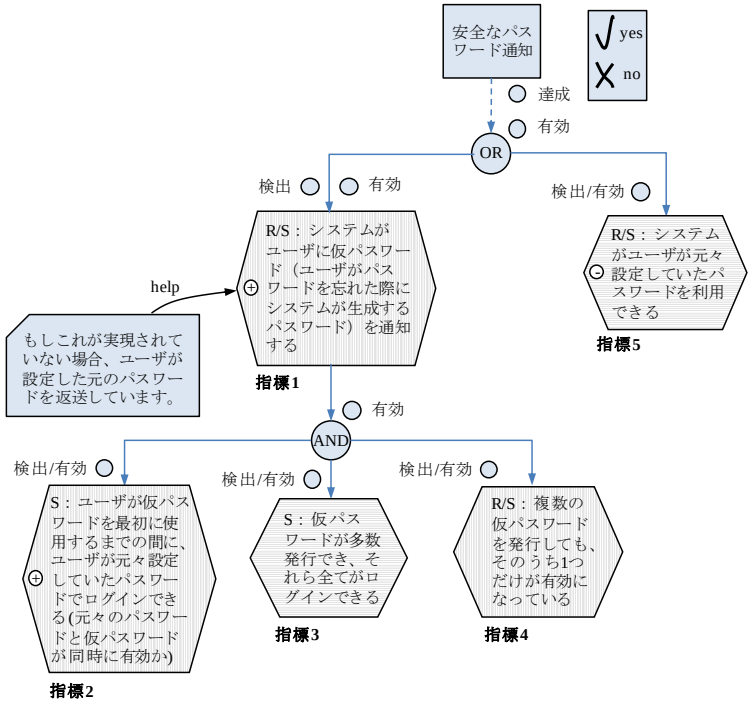

図 2　SGIT の質問項目（指標）とヒントの例

指標にはポジティブ（肯定的）なものとネガティブ（否定的）なものがある．SGITでは，ポジティブな指標は六角形の内側の左端に「+」が記入され，チェックする内容は縦縞模様の六角形の内側に記入される（図 2 中の「指標 1」）．また，ネガティブな指標には「-」が記入され，チェックする内容は横縞模様の六角形の内側に記入される（図 2 中の「指標 5」）．

ポジティブな指標は要件を満たすために必要な項目である．図 2 の『安全なパスワード通知』の例では，指標 1「R/S: システムがユーザに仮パスワード（ユーザがパスワードを忘れた際にシステムが生成するパスワード）を通知する」ことが満たされていれば，図 2 中の「検出」欄に「レ（Yes）」を記入する．指標を満たさなければ，図中の「検出」欄に「×（No）」を記入する．

ネガティブな指標を満たしている箇所が検出されたときは，そこが成果物から除外されれば要件を満たすことになる．図 2 の『安全なパスワード通知』の例では，指標 5「R/S: システムがユーザが元々設定していたパスワードを利用できる」に該当する箇所が成果物から除外されれば，安全にパスワードを通知できることを意味する．もし，該当する箇所が検出されれば，図中の「検出/有効」欄には「×（No）」を記入する．

サブツリーの無い指標（指標の六角形を始点とする矢印が存在しない場合: 図 2 では指標 5）の場合は，指標の内容が確認された時点で検出は終了する．そのため，サブツリーのない指標では，指標の「検出」と「有効」が一致している．

サブツリーのある指標（指標の六角形を始点とする矢印が存在する場合: 図 2 では指標 1）の場合は，その指標の内容が有効であると判断されたことに加え，サブツリー以下の指標の内容が有効と判断されれば，有効となる．そのため，サブツリーのある指標では，指標の「検出」と「有効」が別になっている（サブツリーの指標が有効であると判断され，かつ，「検出」されなければならないため）．

3. 実験

3.1 概要

質問項目の間に依存関係があるチェックリストの表記方法と指摘欠陥の関係を明らかにすることを目的とし，GC, SGIT，チェックリストを与えない（ad-hoc）被験者にグループ分けし，表 1 のような実験を実施した．

本実験の題材は，Web サーバとして実装された書籍販売，貸し出しをする架空のシステムである．レビュー対象物は JSP（Java Server Pages）を含む Java ソースコード計 4500 行程度と日本語で記述された要件定義書で，補助資料として概要設計書を配布した．被験者グループによっては，GC, SGIT のの説明書を配布した．これらの資料をもとに，安全なパスワード通知，安全なユーザ登録，シングルアクセスポイントの 3 つのセキュリティ要件が満たされているかを調べてもらった．

実験はソフトウェア開発の実務者に実験の主旨を説明の上，参加を呼びかけた．当日のグループ分けのために，参加登録時に被験者のソフトウェアレビューの経験年数をはじめとした事前アンケートを実施した．参加者は，組み込みソフトウェア開発や Web アプリケーション開発，パッケージ開発に携わる実務者 92 人である．当日は，事前アンケートをもとに被験者のソフトウェアレビューの経験年数にばらつきが少なくなるよう 5 つのグループに分けて実験を実施した．参加登録時の情報をもとに被験者のグループ分けを行った．当日，業務都合等により，参加登録した人全てが参加できたわけではないので，被験者グループ毎の人数が異なっている．

GC は，安全なユーザ登録とシングルアクセスポイントのセキュリティ要件が満たされているかを確認する被験者グループ，安全なパスワード通知とシングルアクセスポイントのセキュリティ要件が満たされているかを確認する被験者グループの 2 つのグループに分けた．それぞれのグループの内訳は前者が 20 人，後者が 19 人である．

表 1　実験概要

実験題材	Web サーバとして実装された書籍販売，貸し出しをする架空のシステム
レビュー対象物	Java ソースコード　4500 行程度 （JSP: Java Server Pages を含む） 要件定義書（日本語）
実施時間	75 分
補助資料	概要設計書 読み進め方の説明書
セキュリティ要件	安全なユーザ登録（SRP） 安全なパスワード通知（SPR） シングルアクセスポイント（SAP）
被験者グループと人数	GC 2 グループ（SRP/SAP: 20 人，SPR/SAP: 19 人） SGIT 2 グループ（SRP/SAP: 18 人，SPR/SAP: 17 人） ad-hoc（18 人）

SGIT も GC と同様に，安全なユーザ登録とシングルアクセスポイントのセキュリティ要件が満たされているかを確認する被験者グループ，安全なパスワード通知とシングルアクセスポイントのセキュリティ要件が満たされているかを確認する被験者グループの 2 つのグループに分けた．それぞれのグループの内訳は前者が 18 人，後者が 17 人である．

ad-hoc の被験者グループには，3 つのセキュリティ要件が満たされているかを確認してもらった．ad-hoc を使用した被験者は 18 人である．

実験終了後のアンケートで，被験者にレビューの経験やプログラミングの経験，セキュリティの知識がどの程度あるのかといった回答を収集した．

以降，今回の実験で使用した具体的な指標を説明する．

3.1.1 GC

GC では，安全なパスワード通知，安全なユーザ登録，シングルアクセスポイントの 3 つのセキュリティ要件が満たされているかを確認することができる．表 2 に「安全なパスワード通知」を満たすための質問項目（1 段目）とヒントの一部を示す．

表 2　GC の「安全なパスワード通知」を満たすための質問項目とヒント

質問項目 1	
質問	システムがユーザに仮パスワードを通知（ユーザがパスワードを忘れた際にシステムが生成するパスワード）するかどうか，要件定義書とソースコードを調べてください．
ヒント	もしこれが実現されていない場合，ユーザが設定した元のパスワードを返送しています．
質問項目 2	
質問	ユーザが仮パスワードを最初に使用するまでの間に，ユーザが元々設定していたパスワードでログインできるかどうか（元々のパスワードと仮パスワードが同時に有効か），ソースコードを調べてください．
ヒント	そうでなければ，悪意のある攻撃者がパスワード通知機能を使うことで，正当なユーザを締め出すことができてしまいます．正当なユーザは元々のパスワードが無効になるため，ログインの際に仮パスワードを使用しなければいけませんが，仮パスワードは攻撃者しか知りません．

3.1.2 SGIT

SGIT でも GC と同様の指標を用いた．指標は安全なパスワード通知，安全なユーザ登録，シングルアクセスポイントの 3 つのセキュリティ要件が満たされているかを確認するものである．図 3 に「安全なパスワード通知」を満たすための質問項目（指標）とヒントの一部を示す．表 2 の質問項目 1 と図 3 の指標 1，表 2 の質問項目 2 が図 3 の指標 5 と対応する．その他の全ての指標についても SGIT と GC で同じものを用いている．

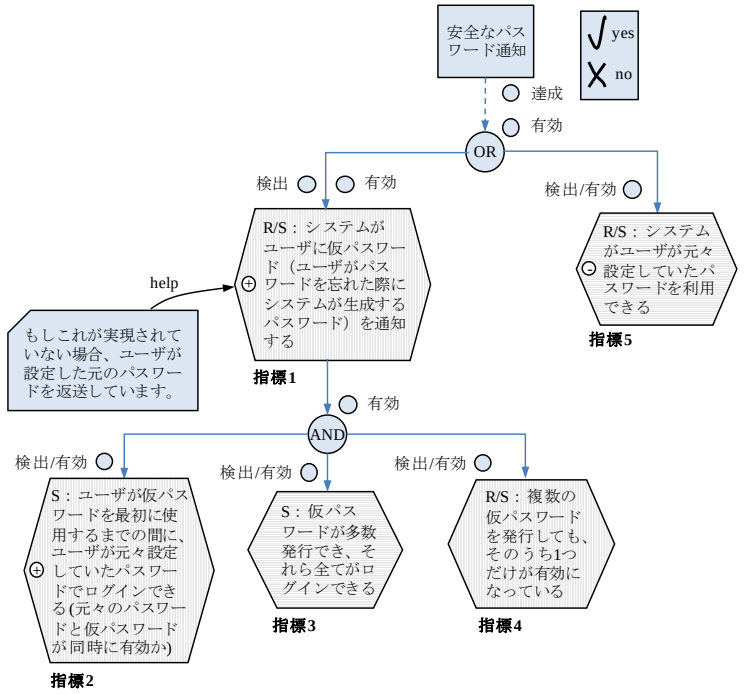

図 3　SGIT の「安全なパスワード通知」を満たすための質問項目とヒント

3.1.3 ad-hoc

ad-hoc は，GC や SGIT のようにリーディングテクニックを利用せず，被験者のこれまでの経験や普段のレビューで欠陥をみつけるときと同じようにインスペクションを実施する方法である．GC や SGIT と同様に，シングルアクセスポイント，安全なユーザ登録，安全なパスワード通知の 3 つのセキュリティ要件が満たされているかを確認してもらうために大まかな方針を与えた．

表 3 に，GC の表 2 や SGIT の図 3 に対応する「安全なパスワード通知」を調べるために被験者に与えた方針を示す．被験者は，成果物（要件定義書，ソースコード）に含まれるセキュリティ要件を満たさない箇所を見つけたら，欠陥リストに記録する．欠陥リストには，セキュリティ要件を満たさない箇所が含まれる文書名／欠陥の具体的な箇所／欠陥の簡単な説明／欠陥がどのセキュリティ要件を満たさないものであるかを記録する．

3.2 構造化された質問項目（指標）

本節では，セキュリティ要件，安全なユーザ登録，安全なパスワード通知の GC 表記と SGIT 表記を比較する．シングルアクセスポイントのセキュリティ要件にはチェックリストの構造がないため，本節では割愛している．

表 3　ad-hoc の「安全なパスワード通知」を満たすためのヒント

セキュリティ要件：安全なパスワード通知
ユーザ名とパスワードの認証を使用するシステムは，パスワードが忘れられた場合に再発行する仕組みを用意しておくべきです．一般的に，ユーザがパスワードを設定済みの状態で，ユーザにパスワードを送り返すことは好ましくないとされていますが，ユーザがパスワードを思い出す可能性が低いことと，ユーザは安全でない通信路ではパスワードを送信しないだろうということから，システムが自動生成した一時的な仮パスワードをユーザに送信します．この方法を使用するシステムには注意点があります．仮パスワードを発行した時点でユーザが元々設定したパスワードが無効になると，悪意ある攻撃者がパスワードの再発行を要求し，ユーザの元のパスワードを使用不可にさせることができ，正当なユーザが締め出される可能性があります．また，仮パスワードを同時に複数発行でき，それらが全てでログインが可能になる場合，悪意ある攻撃者が有効なパスワードを推測しやすくなります．また，仮パスワードが安全でない通信路で渡される場合には，仮パスワードではパスワード変更以外の機能を利用できないように制限することが望まれます．

3.2.1 安全なユーザ登録

安全なユーザ登録について GC と SGIT の対応を図 4，図 5 に示す．図 4 は GC の構造を示している．質問項目 2 の結果によって，質問項目 3 の実施を省略する構造となっている．図 5 は図 4 の GC に対応する SGIT である．図 5 では，指標 1，2，4 が AND で結合されており，指標 1, 2, 4 のいずれかが満たされていなければ，指標 3 を確認する必要はない．

3.2.2 安全なパスワード通知

安全なパスワード通知について GC と SGIT の対応を図 6，図 7 に示す．GC（図 6）では，質問項目 1 の結果によっては, 2~6 をとばして，質問項目 7 を確認する場合もある．SGIT（図 7）では，指標 1，7，が OR で結合されており，これらの指標に記述されていることのうちどちらかを確認することができれば「安全なパスワード通知」が満たされていることが確認できる．その他の指標に記述されていることを確認する必要はない．

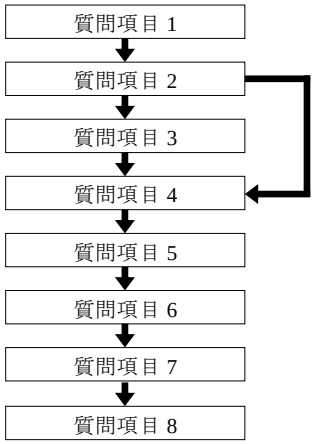

図 4　GC の構造化された部分

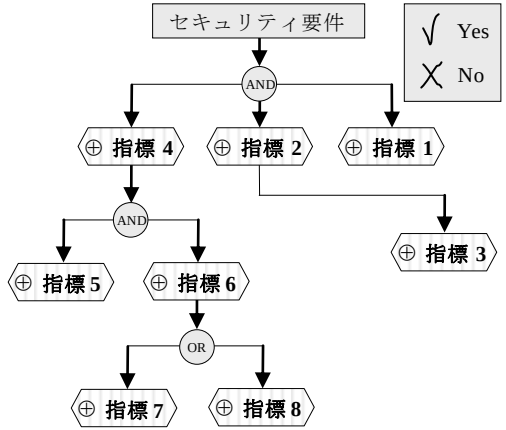

図 5　SGIT の構造化された部分

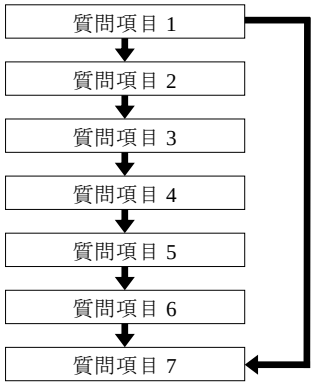

図 6　GC の構造化された部分

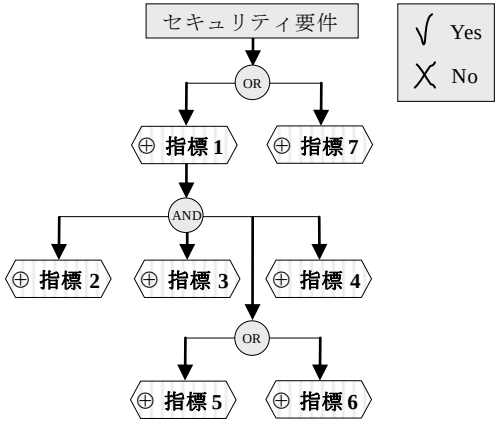

図 7　SGIT の構造化された部分

3.3 手順

実験の手順を以下に示す.

(1) 事前アンケートの実施

実験への参加登録時（実施日よりも 1 週間以上前）にアンケートを実施し，被験者の経験にばらつきがないようにどの方法でレビューをしてもらうかを決める.

(2) レビューの実施（75 分間）

それぞれで決められたレビュー方法に従い，75 分間でできるところまで，ソースコードと要件定義書に含まれるセキュリティ要件を満たさない箇所を 1 人で発見してもらう．発見した場合には，欠陥が見つかった文書，ページ数等の欠陥の見つかった箇所，欠陥の説明，欠陥がどのセキュリティ要件を満たさないものであるかを記録してもらう.

(3) アンケートの実施

実験終了後，被験者がレビューの経験やプログラミングの経験，セキュリティの知識をどの程度持っているのかを調査した.

4. 結果

4.1 レビュー方法による比較

ad-hoc，GC，SGIT の 3 つのレビュー方法の正答数と正答率を図 8, 9 に示す．図 8, 9 では，被験者グループ（SRP, SPR, SAP の組合せ）を区別せずに集計している.

4.1.1 正答数の比較

図 8 は, レビュー方法ごとに指摘された欠陥の正答数の分布を示したグラフである．グラフは，横軸にレビュー方法，縦軸に正答数をとった箱ひげ図である．箱の下端は第 1 四分位点（25 パーセンタイル），上端は第 3 四分位点（75 パーセンタイル）を示し，ひげの末端は 5 パーセンタイルと 95 パーセンタイル，箱中の実線は中央値を示している.

本実験では，レビューで使用する方法により，正答数に差が生じた．図 8 からわかるように，GC や SGIT を使用すると正答数が増加した．平均値は ad-hoc が 1.76 件，GC が 3.43 件，SGIT が 3.00 件となり，中央値は ad-hoc が 1.00 件，GC と SGIT が 3.00 件となった. また, SGIT を使用した被験者は少なくとも 1 つは欠陥を発見できている.

さらに，ad-hoc と GC の間で U 検定を実施したところ，p 値が 0.00133 となり，有意水準 1%の統計的有意差があるという結果が得られた. ad-hoc と SGIT の間でも p 値が 0.00602 となり，有意水準 1%の統計的有意差があるという結果が得られた.

4.1.2 正答率の比較

図 9 は, レビュー方法ごとに指摘された欠陥の正答率の分布を示したグラフである．グラフは，横軸にレビュー方法，縦軸に正答数をとった箱ひげ図である．箱の下端は第 1 四分位点（25 パーセンタイル），上端は第 3 四分位点（75 パーセンタイル）を示し，ひげの末端は 5 パーセンタイルと 95 パーセンタイルであり，それよりも外側の点は外れ値として表示している.

正答率の比較では, ad-hoc と GC, SGIT との間に統計的有意差はなかった. しかし，正答率の比較においても，レビュー方法により差が生じた.

図 9 からわかるように，どのレビュー方法を用いても正答率 100%の人が最も多くなった．しかし，25 パーセンタイルは ad-hoc よりも GC，GC よりも SGIT のほうが高くなった．また，正答率の平均値についても ad-hoc が 76.4%，GC が 86.9%，SGIT が 90.8%となり，SGIT の正答率が最も高いという結果になった．さらに，ばらつきについても SGIT が一番小さくなり，正答率が高い人が多いという結果が得られた.

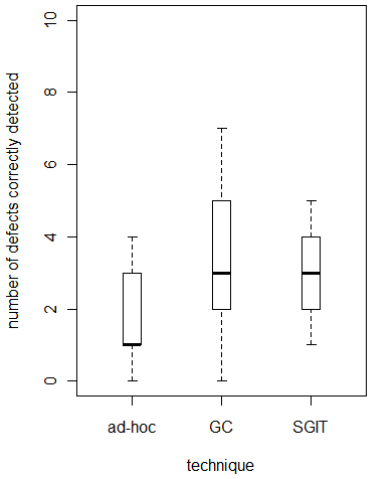

図 8　レビュー方法ごとの正答数の分布

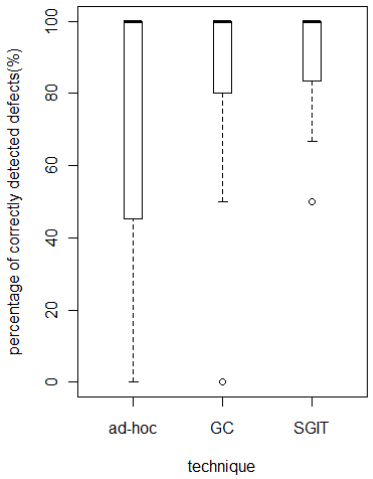

図 9　レビュー方法ごとの正答率の分布

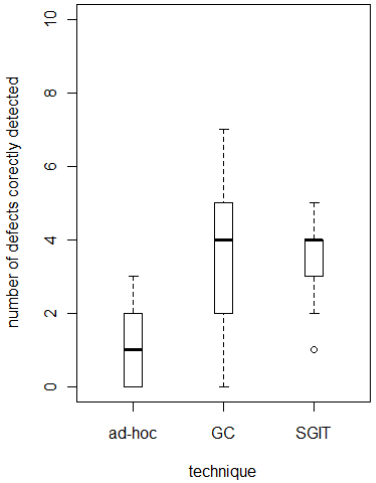

図 10　レビュー方法ごとの正答数の分布

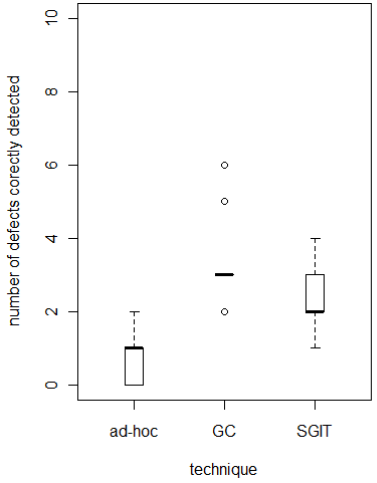

図 11　レビュー方法ごとの正答数の分布

4.2　構造化された質問項目（指標）の比較

4.2.1 安全なユーザ登録における正答数の比較

図 10 には，安全なユーザ登録におけるレビュー方法ごとの正答率の分布を示す．

図 10 からわかるように，GC や SGIT を使用すると，ad-hoc よりも正答率が増加していることがわかる．また，ad-hoc と GC の間では p 値が 0.000000509 となり，有意水準 1%の統計的有意差があることがわかった．同様に，ad-hoc と SGIT の間では p 値が 0.000261 となり，有意水準 1%の統計的有意差があることがわかった．

4.2.2 安全なパスワード通知における正答数の比較

図 11 には, 安全なパスワード通知におけるレビュー方法ごとの正答率の分布を示す．

図 11 からわかるように，GC や SGIT を使用すると，ad-hoc よりも正答率が増加していることがわかる．また，ad-hoc と GC の間では p 値が 0.000400 となり，有意水準 1%の統計的有意差があることがわかった．同様に，ad-hoc と SGIT の間では p 値が 0.000135 となり，有意水準 1%の統計的有意差があることがわかった．

安全なユーザ登録，安全なパスワード通知において構造化された質問項目（指標）の正答率の平均を表 4 に示す．これより，同じ構造化された質問項目（指標）でも SGIT を使用すると正答率が上がる傾向があるということがわかる．

表 4　レビュー方法ごとの構造化された指標の正答率の平均

	安全なユーザ登録		**安全なパスワード通知**	
質問項目 または指標	2	4	**1**	**7**
GC（%）	50	40	100	33.3
SGIT（%）	100	100	100	75.0

5. まとめ

本稿では，質問項目の間に依存関係があるチェックリストの表記方法と指摘欠陥の関係を調査することを目的とし，被験者実験を実施した．被験者は，組み込みソフトウェア開発や Web アプリケーション開発，パッケージ開発に携わる実務経験者 92 人である．レビュー対象は，Web サーバとして実装された書籍販売，貸し出しをする架空のシステムであり，レビュー対象物は，JSP を含む Java ソースコード 4500 行程度と日本語で記述された要件定義書である．

実験では，質問項目の間に依存関係があるチェックリストとして GC と SGIT，特にやり方を指定しない ad-hoc の 3 つの方法で指摘された欠陥の正答数と正答率を比較した．実験の結果，GC を使用すると，他のレビュー方法よりも多くの欠陥を発見することができた．また，SGIT を使用すると，他のレビュー方法よりも指摘された欠陥の正答率が高くなった．

実験終了後のアンケートより，SGIT 利用手順を理解するのにかかった時間は，20 分～45 分であることがわかった．GC の利用手順を理解するのと比較すると，SGIT は学習コストが高いと考えられる．しかし，いったん利用手順を理解してしまえば，SGIT は，正答率を上げるために有効なチェックリストの表記方法であると考えられる．

今後，プログラミング経験等のアンケート結果と合わせて分析を進めることにより，指摘欠陥の正答数や正答率を高めるために必要な経験や知識がどのようなものであるかが明らかになるのではないかと考えている．

謝辞

実験に参加いただいた協力者の皆様，イベント開催に協力いただいた皆様に感謝する．また，本研究の推進にあたって，Frank Elberzhager 氏，保田 裕一朗 氏，服部 京子 氏に多大な支援をいただいた．また，本稿の実験題材（レビュー対象，チェックリストをはじめとするドキュメント）の作成にあたって Frank Elberzhager 氏，Marek Jawurek 氏の支援をいただいた．本研究の一部は，Fraunhofer Institute for Experimental Software Engineering（フラウンホーファ研究機構 実験的ソフトウェア工学研究所）の協力により実施された．また，本稿の分析対象データは IBM Academic Initiative Program（http://www.ibm.com/developerworks/university/academicinitiative/）の支援によって収集された．本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究補助費（若手 B：課題番号 20700028）による助成を受けた．

参考文献

1) Michael E. Fagan: Design and Code Inspection to Reduce Errors in Program Development, IBM Systems Journal, Vol.15, No.3, pp.182-211, 1976
2) Thomas Thelin, Per Runeson and Björn Regnell: Usage-based reading-an experiment to guide reviewers with use cases, Information and Software Technology, Vol.43, No.15, pp.925-938, 2001
3) Adam A. Porter, Lawrence G. Votta, Jr., Victor R. Basili: Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, IEEE Transactions on Software Engineering, Vol.21, No.6, pp.563-575, 1995
4) Victor R. Basili, Scott A. Green, Oliver Laitenberger, Forrest J. Shull, Sivert Sorumgard, Marvin V. Zelkowitz: The Empirical Investigation of Perspective-Based Reading, Empirical Software Engineering, Vol.1, No.2, pp.133-164, 1996
5) Thomas Thelin, Carina Andersson, Per Runeson, Nina D. Fogelstrom: A Replicated Experiment of Usage-Based and Checklist-Based Reading, 10th IEEE International Symposium on Software Metrics, pp.246-256, 2004
6) Thomas Thelin, Per Runeson, Claes Wohlin: An Experimental Comparison of Usage-Based and Checklist-Based Reading, IEEE Transactions on Software Engineering, Vol.29, pp.687-704, 2003
7) Frank Elberzhager, Alexander Klaus, Marek Jawurek: Software Inspections Using Guided Checklists to Ensure Security Goals, 2009 International Conference on Availability, Reliability and Security, pp. 853-858, 2009