SGIT(Security Goal Indicator Trees)を使ったインスペクション

本文書は、SGIT(Security Goal Indicator Trees)を使ってインスペクションをする方法を説明するものです。

準備

ある要件を満たすためのインスペクションの方法が記述された SGIT を使用することで、成果物（要件定義書、ソースコード）がその要件を満たしているかどうかを評価できます。本ハンズオンでは、以下のドキュメントが配布されます。本文書も配布されます。

- 2 つの SGIT
- インスペクション対象物（要件定義書、ソースコード）
- 参考文書（設計書）
- 空の欠陥リスト
- アンケート

ハンズオンの目的： SGIT が表す階層的な情報を使いながら、セキュリティ要件が満たされているかどうか評価すること

本文書では、SGIT を使ってソースコードと設計書をインスペクションする方法について具体例をまじえて説明します。SGIT は要件に応じて作成します。本文書で提示する SGIT の例は、保守性が十分であるかを確認するためのものです。ハンズオン当日には、セキュリティ要件を満たすかどうかを確認するための SGIT を配布します。

設計書とソースコードへの SGIT の適用例

インスペクタである Bob さんには、以下の二つのシステムの中間成果物が与えられています。

- UML 図（設計書）（図 1）
- ソースコード（図 2）

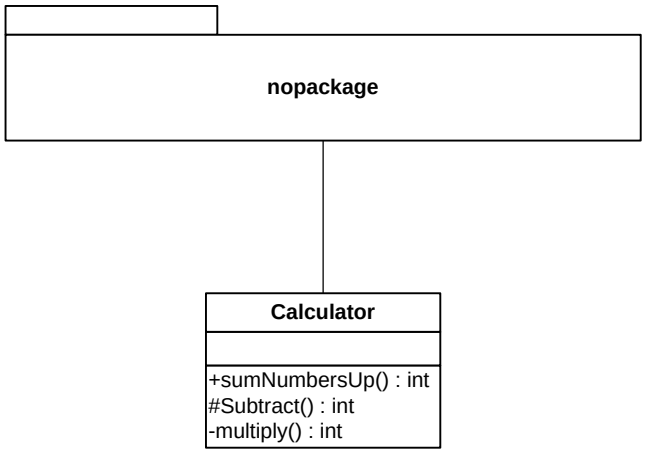

図 1　Calculator クラスの UML 図（設計書）

© Fraunhofer IESE, 2009

```
1.    package nopackage;
2.    /**
3.    *This clas represents a simple Calculator,
4.    *which performs some actions right after its start.
5.    *@author Charlie
6.    */
7.    public class Calculator {
8.    /**
9.    *This is the standard way to start the Calculator.
10.   *@param argv
11.   */
12.   public static void main(String argv[]){
13.     Calculator calc = new Calculator();
14.     System.out.println("Welcome to Calculator 2000");
15.     StringBuffer o = new StringBuffer();
16.     // hi Bob, how's the inspection?
17.     o.append("2+2*(5-1)="+calc.sumNumbersUp(2,
18.       calc.multiply(2,calc.Subtract(5,1))));
19.     // calculation of 5-4*(3+3) and adding
20.     // the result to the StringBuffer of results
21.     o.append("5-4*(3+3)="+calc.Subtract(5,
22.       calc.multiply(4,calc.sumNumbersUp(3,3))));
23.     // calculation of 1+2+3+4 and adding the result
24.     // to the StringBuffer of results
25.     o.append("1+2+3+4="+calc.sumNumbersUp(1,
26.       calc.sumNumbersUp(2, calc.sumNumbersUp(3,4))));
27.     System.out.println(o.toString());
28.   }
29.   public int sumNumbersUp(int a, int b){return a+b;}
30.   private int Subtract(int a, int b){return a-b;}
31.   protected int multiply(int a, int b){return a*b;}
32.   }
```

図 2　Calculator クラスのソースコード

Bob さんには、上記の 2 つの中間成果物に加えて、図 3 に示される SGIT も与えられています。

図 3 の SGIT でチェックできる要件は『保守性』です。図中の要件「良い保守性」の下には、木構造で示された 1～8 の指標があります。個々の指標は六角形で示され、指標どうしは木構造で結合されます。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（図中「AND」で結合されている指標）と、どれかを確認すればよい OR（図中「OR」で結合されている指標）があります。たとえば、指標 6 と 7 は OR で結合されているので、要件が満たされているかどうかを確認するという点では、いずれかを確認するだけで構いません。指標が満たされているかどうかを判断できます。一方、指標 1～5 は AND で結合されているので、全て確認する必要があります。

SGIT の指標を 1 つずつ確認する前に、要件『保守性』を大まかに把握するために全ての指標に一通り目を通します。また、システムの理解を深めるために、設計書にも簡単に目を通します（ここで紹介する例では、インスペクションの対象は設計書とソースコードです）。

指標にはポジティブ（肯定的）なものとネガティブ（否定的）なものがあり、これを意識する必要があります。SGIT では、ポジティブな指標は指標の表記の左側に「+」が記入され、指標の内容は縦縞模様の六角形に記入されます（図 3 中の指標 2, 3, 4, 5, 6, 7）。またネガティブな指標には「-」が記入され、指標の内容は横縞模様の六角形に記入されます（図 3 中の指標 1, 8）。

- ポジティブな指標は要件を満たすために必要な項目です。図 3 の『保守性』の例では、指標 2「D: メソッドの名前は一貫している」が満たされていることが確認されれば、保守性が高くなっていることを意味します。この場合、また、図中の「検出/有効」欄に

© Fraunhofer IESE, 2009

「レ（Yes）」を記入します。指標を満たさない箇所が検出されれば、図中の「検出/有効」欄に「×（No）」を記入します。

- 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には、その指標に該当する箇所が成果物から除外されれば要件を満たすことになります。図 3 の『保守性』の例では、指標 1「D/S: メソッドの可視性に一貫性がない」に該当する箇所が成果物から除外されれば、保守性が高くなることを意味します。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）」を記入します。

- 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 3 では指標 4 以外）の場合は、指標の内容が確認された時点で検出は終了です。そのため、サブツリーのない指標では、指標の「検出」と「有効」が一致します。

- 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 3 では指標 4）の場合は、その指標の内容が有効であると確認され、かつ、サブツリー以下の指標の内容が有効であれば、有効であるとみなします。そのため、サブツリーのある指標では、指標の「検出」と「有効」が別になります。（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）

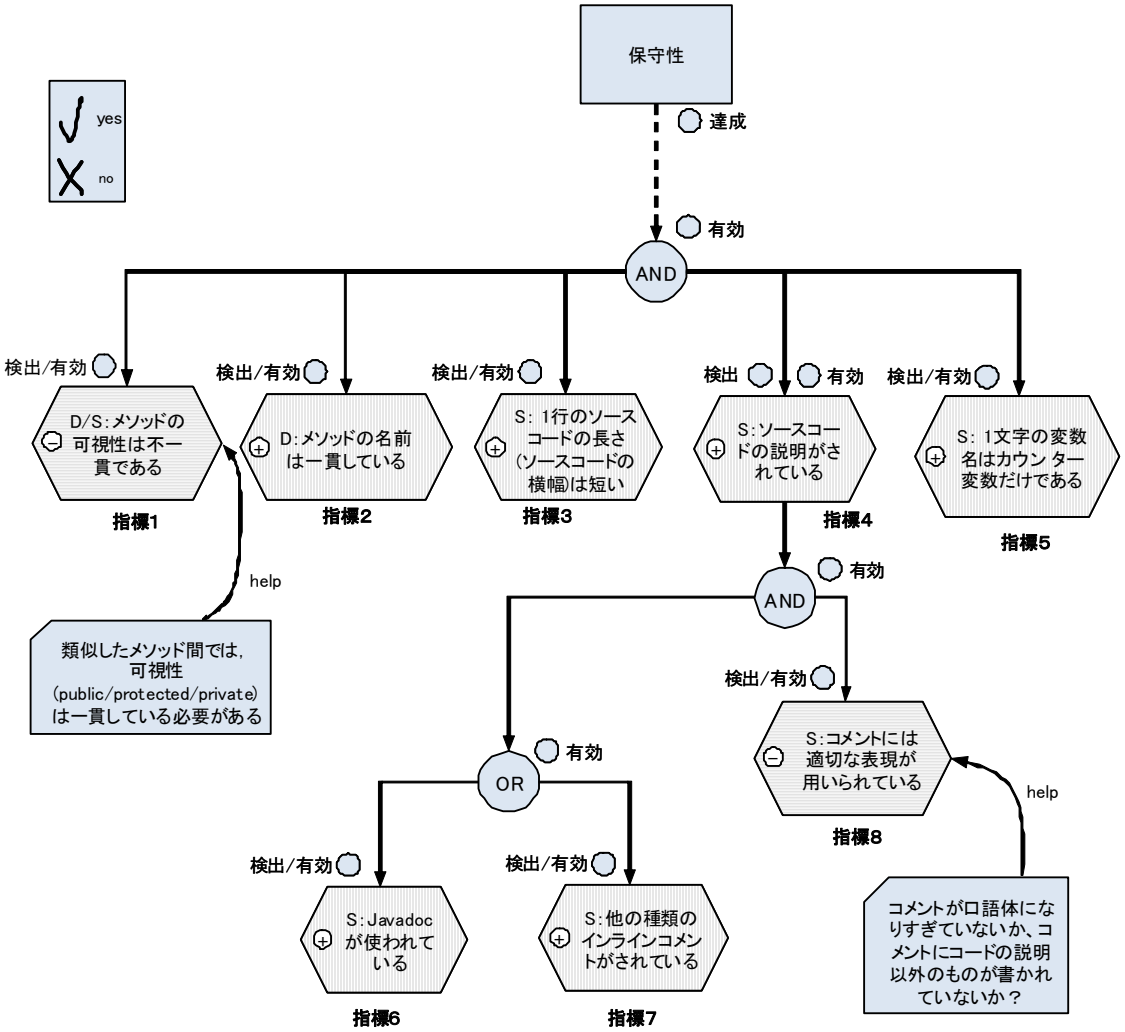

図 3　要件『保守性』の SGIT の例

© Fraunhofer IESE, 2009

では、具体的に SGIT を用いたインスペクションを例を使いながら説明していきます。

Bob さんはインスペクションを開始します。

1. Bob さんは、指標 1 を確認します。指標 1 は、六角形の左側に「-」が記され、横縞模様で示されており、ネガティブな指標です。指標の内容は『D/S：メソッドの可視性は不一貫である』です。 全ての指標には、文頭に D か S の英字が付いており、続いて指標の内容が書かれています。英字は指標内容の達成に向けて参照する文書の頭文字を表しています。D は設計書（**D**esign）、S はソースコード（**S**ource Code）を表しています。

   指標 1 はネガティブな指標なので、インスペクタは指標に該当する箇所を除外しなければなりません。つまり、メソッドの可視性に一貫性がない箇所を除外しなければなりません。もし、Bob さんによって可視性に一貫性がないと判断された場合は、指標内容に該当する箇所が検出されていることになります。

   『help』の線を通して指標につながっている箱は、指標内容に対しての追加の説明やヒントを表しています。指標 1 につながっている help を読んだ後、インスペクタは、インスペクション対象の設計書である UML 図（図 1）に目を通します。そこで、Calculator クラスが、足し算（sumNumbersUp）、引き算（Subtract）、掛け算（multiply）の 3 つのメソッドから構成されていることがわかります。3 つのメソッドは全て異なる可視性（+ public、 # protected、 - private）を持ちます。明らかにメソッドの可視性に一貫性がないため、この指標内容は確認できません。したがって、指標 1 は「確認できない」となります。確認できないので、『検出/有効』のチェックボックスに×印を付け、欠陥リストに以下のように記述します。

指標番号 / 質問番号（SGIT、Guided Checklist の方のみ）: 指標 1

Java のクラス名、行 / ページ: UML 図、Calculator クラス

指摘内容: メソッドの可視性が矛盾している！ Calculator クラスの全てのメソッドが異なる可視性を持っている。

   この時点で、AND で結合された指標のうちの 1 つがすでに確認できず、有効でないため、要件『保守性』を達成できないことがわかります。ここでは、本要件に関する全ての欠陥を発見することを目指し、Bob さんはインスペクションを続けます。

2. 次に、Bob さんは指標 2:『D: メソッドの名前は一貫している』に従って、インスペクションを続けます。指標 2 は六角形の左側に「+」が記され、たて縞模様で示されており、ポジティブな指標です。Bob さんは指標の内容が満たされていることを示さなければなりません。また、指標の内容の先頭に、英字 D が表示されているため、指標 2 の対象は設計書です。Bob さんは設計書をインスペクションします。UML 図より、メソッドが sumNumbersUp、 Subtract、 multiply であることがわかります。メソッドは 1 単語から成るわけでも、小文字/大文字で一貫しているわけでもないので、指標内容『メソッドは一貫している』は、このシステムでは有効ではありません。したがって、指標 2 を検出できなかったので、『検出/有効』のチェックボックスに×印を付け、欠陥リストに以下のように記述します。

指標番号 / 質問番号（SGIT、Guided Checklist の方のみ）: 指標 2

Java のクラス名、行 / ページ: UML 図、Calculator クラス

指摘内容: Calculator クラスの全てのメソッドの名前は一貫していない。

© Fraunhofer IESE, 2009

3. 続いて、指標 3「S:1 行のソースコードの長さ」です。指標 3 はポジティブな指標です。指標内容の先頭文字が「S」なので、欠陥はソースコードを見ることで発見することができます。Bob さんはソースコードに目を通し、短くする必要のある 3 つの長い行（17-18、21-22、25-26 行目）があることを確認します。したがって、『検出/有効』のチェックボックスに×印を付け、欠陥リストに以下のように記述します（長い行は画面の幅に収まるように、途中で改行する等します）。

指標番号 / 質問番号（SGIT、Guided Checklist の方のみ）: 指標 3
Java のクラス名、行 / ページ: Calculator クラス、17-18、21-22、25-26 行目
指摘内容: これらの行は長すぎるため、適当なメソッドの引数区切りで行を折り返し、短くするべきである。

4. Bob さんはサブツリーを持つ指標 4「S: ソースコードの説明がある」に進みます。Bob さんは指標に反する箇所をみつけようとします。ソースコード中にあるコメントを調べることで、クラス定義の前、メソッド定義の前、メソッドの中にコメントがあることを確認し、その内容が説明として十分であることを確認しました。そこで、Bob さんは、指標 4 は満たされたと判断し『検出』のチェックボックスに「レ（Yes）」を記入します。次に、指標 4 を始点とする矢印がつながっている先のサブツリーを調べます。サブツリーが有効である場合のみ、指標 4 が検出できたといえます。（指標 4 自体が検出されない場合（チェックボックスが「×」の場合）は、Bob さんはサブツリーを見る必要はありません。通常サブツリーの指標は、その親の指標よりも具体的な内容になります。）

5. Bob さんは、まず、指標 4 の下の AND で結合された指標とサブツリーに進みます。その先には OR で結合されたサブツリーがあり、その先にはポジティブな指標 6:「S: JavaDoc が使われている」があります。指標はすぐに検出できるので、『検出/有効』のチェックボックスに「レ（Yes）」にチェックをします。OR ノードにはもう一つのポジティブな指標 7 があります。要件が達成されているかを知るためには指標 7 を確認する必要はありませんが、ここでは、指標 7 もチェックすることにします。

6. Bob さんはポジティブな指標 7「S: 他の種類のインラインコメントが使われている」に目を通します。他のインラインコメントがあり、クラスについてのコメントが十分であると Bob さんは判断し、指標 7 が検出されたとしました。「検出/有効」のチェックボックスに「レ（Yes）」を記入します。

7. OR で結合されたサブツリーの指標を全てチェックしたので、Bob さんは AND ノードで結合された右のサブツリーに移動し、ネガティブな指標 8: 「S: コメントには適切な表現が用いられている」を調べます。指標内容を確認するために、Bob さんはソースコード中にある適切でないコメントがないかを探します。「help」というラベルがついた指標 8 の補足説明も読みます。Bob さんはソースコードのコメントに一通り目を通したときにソースコードの 16 行目に Bob さん宛のコメント「やぁボブ、インスペクションはどうだい？」を発見します。面白いコミュニケーションですが、これはコメントとしては明らかに適切ではありません。このコメントは Bob さんの友人の開発者 Charlie さんが書いたものであると考えられます。Charlie さんは Bob さんがコードインスペクションすることを事前に知っていたからでしょう。Bob さんは「検出/有効」のチェックボックスに「×印（No）」を付け、欠陥リストに以下のように記述します。

指標番号 / 質問番号（SGIT、Guided Checklist の方のみ）: 指標 8

© Fraunhofer IESE, 2009

Java のクラス名、行 / ページ: Calculator クラス、16 行目

指摘内容: Charlie はソースコードのコメントを通して私と連絡をとりたがる。

8. 次に、Bob さんはポジティブな指標 5: 「S: 1 文字の変数名はカウンター変数だけである」に進みます。StringBuffer o という名前の変数名が 1 文字のものがあることに気づきます。o はカウンター変数ではないので、『検出/有効』のチェックボックスに「×印（No）」を付け、欠陥リストに以下のように記述します。

指標番号 / 質問番号（SGIT、Guided Checklist の方のみ）: 指標 5

Java のクラス名、行 / ページ: Calculator クラス、15 行目

指摘内容: StringBuffer 変数 o は、カウンター変数でないのにもかかわらず、変数の中身を適切に表していない。

これで、Bob さんは全ての指標を調べ、インスペクション対象が要件『保守性』を達成しているか把握できました。図 4，5 は、Bob さんが全ての指標を調べた後の、SGIT の状態（「レ」と「×」の記入結果）を示しています。

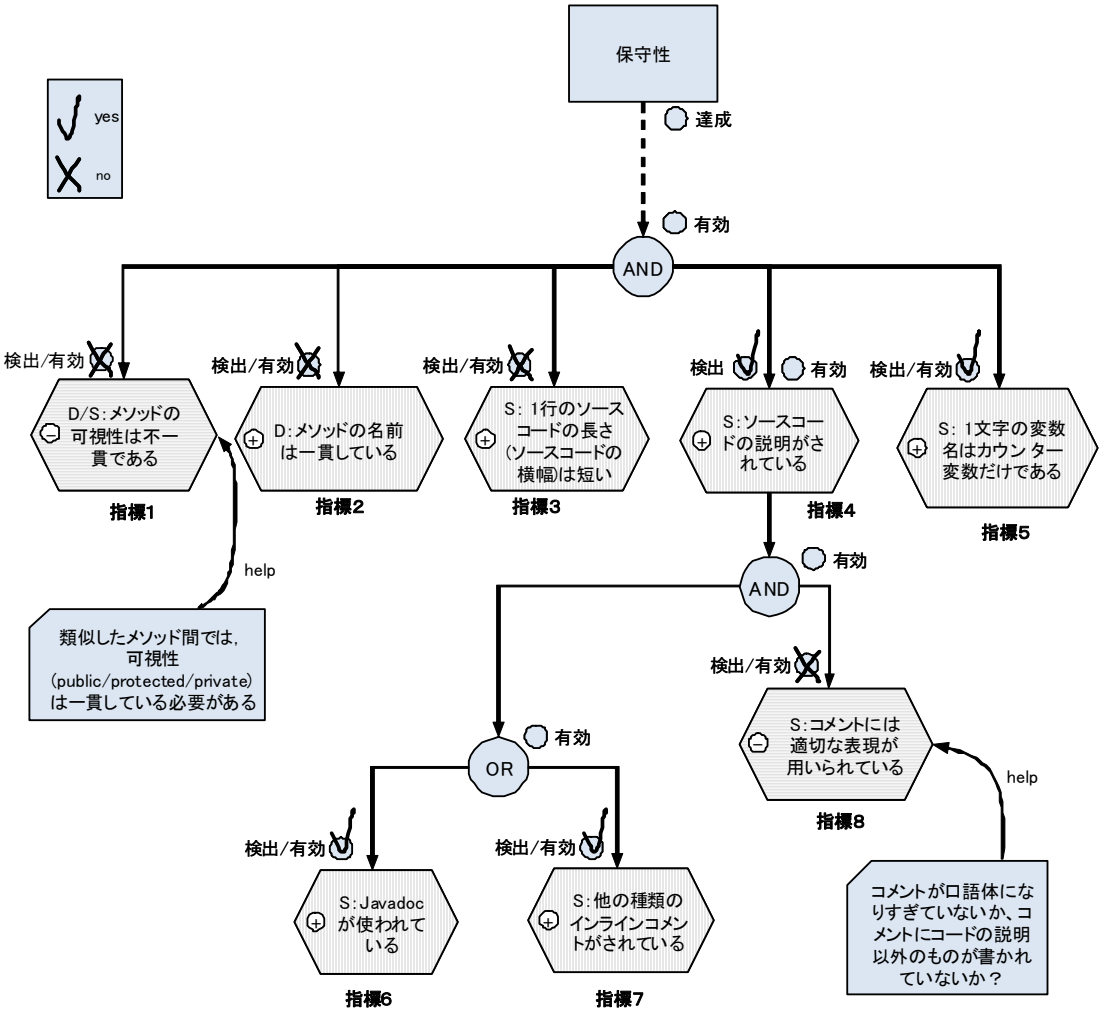

図 4　全指標の調査終了時における SGIT

© Fraunhofer IESE, 2009

Bob さんは、全ての検出できない/有効でない指標を調べ、全ての検出可能/有効である指標をチェックしました。さらに最終目標「保守性」や AND や OR で結合されたサブツリーの「有効」の指標を集めます。

1. 指標 6 と指標 7 では、2 つとも「検出/有効」に「レ（Yes）」がチェックされたので、OR のとなりにある「有効」のチェックボックスに「レ（Yes）」を記入します。

2. 次に、指標 8 の上にある AND のサブツリーを調べます。ここでは、左側のサブツリーは有効ですが、右のサブツリーは有効ではありません（指標 8 の「検出/有効」が「×（No）」であるため）。したがって、この AND で結合されたサブツリーは有効ではないので、AND の右上の「有効」のチェックボックスに×印を付けます。

3. SGIT の最上段の AND で結合されたサブツリーを調べます。一番右側のサブツリー以外は有効（「検出/有効」または「有効」に「×（No）」が記入されている）ではありません。したがって、この AND で結合されたサブツリーは有効ではないので、AND の右上のチェックボックスに「×（No）」を記入します。

4. 結果、「達成」のチェックボックスには×印が付けられます。結果として要件は満たされていないことになります。

要件が達成されているかを調べるためには、必ずしも全ての AND/OR サブツリーや指標を調べる必要はありません。指標 1 が検出できなかった段階で、最上段の AND で結合されたサブツリー（指標 1～5 を子にもつサブツリー）は有効ではないことがわかります。最上段の指標を調べただけで、要件が達成されているか、あるいは、達成されていないかということを事前に知ることができますが、要件を達成するには全ての指標を満たしておくことが必要になりますので、他の指標も調べました。
図 5 は、サブツリーを含め全ての「検出/有効」の評価を行った後の SGIT の様子です。

Bob さんのインスペクションの成果物は、印が記入された SGIT と欠陥リストであり、次回のインスペクションミーティングの議題となります。

© Fraunhofer IESE, 2009

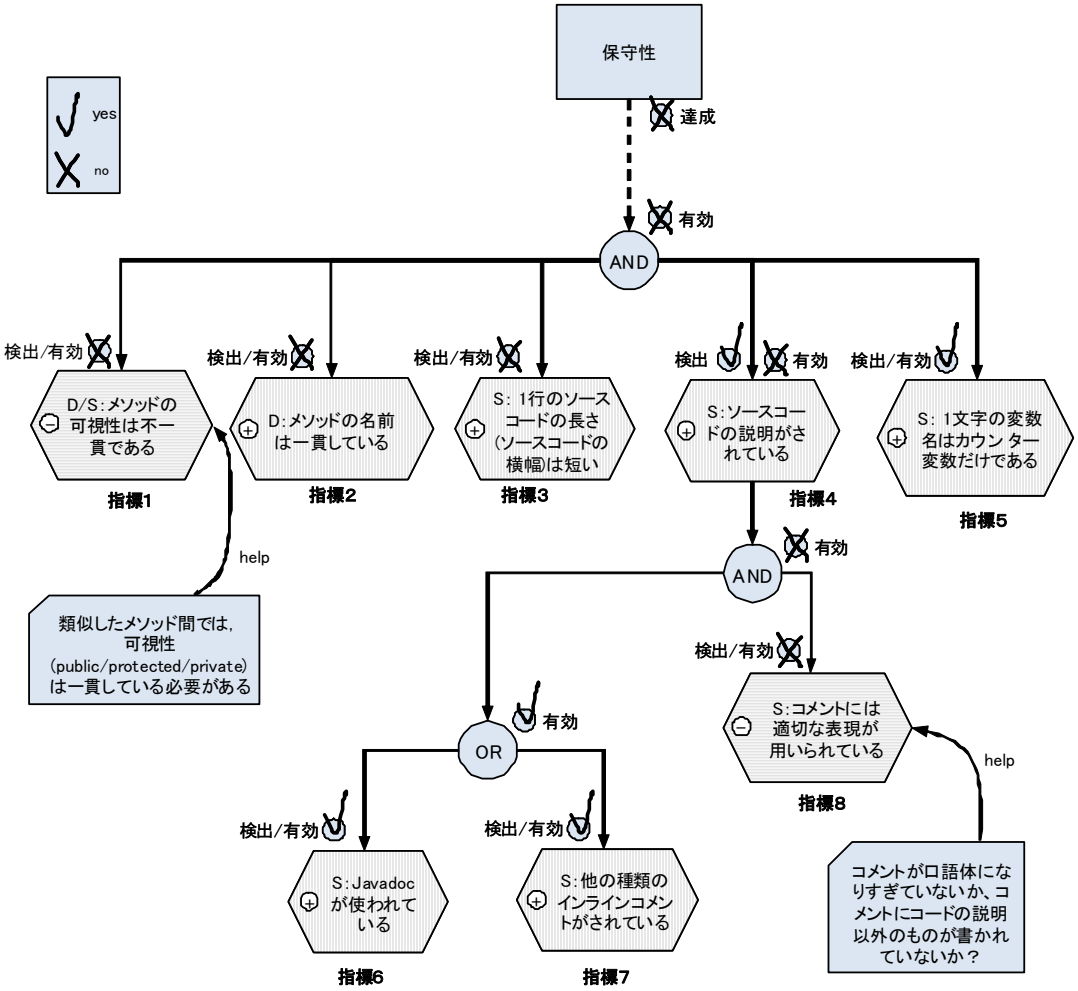

図 5　インスペクション終了時の SGIT

© Fraunhofer IESE, 2009