データ結合と制御結合の解析技術

The technicalities of data coupling and control coupling

背景

モジュラーソフトウェアは、高品質のソフトウェアの特徴と見なされることがよくありますが、その主張は、モジュールが期待どおりに相互作用する場合にのみ当てはまります。 そのため、DO-178C 標準では、データ結合と制御結合の両方が要件に準拠していることを示す必要があります。

DO-178C のセクション 6.4.4.2.c は、「要件ベースのテストがコードコンポーネント間のデータと制御結合を実行したことを確認するための解析」を要求しています。 この指令は、設計、統合、およびテストの目標が確実に満たされるように設計されています。 本資料では、制御フローとデータフロー解析の組み合わせによって、制御結合とデータの結合の測定をどのように達成できるか、またソフトウェアツールを使用してそのプロセスをどのように効率化できるかについての概要を説明します。

要件ベースのテスト（Requirements based testing）

要件ベースのテストでは、テストケース、条件、およびデータが要件から導き出されます。機能と、パフォーマンス、信頼性、使いやすさなどの非機能属性の両方をテストします。テストプロセス中に検出された欠陥は、欠陥のライフサイクルを経て解決まで追跡され、欠陥統計はプロジェクト進捗状況の尺度を提供します。

一般に、要件ベースのテストには、5つの段階があります。

- **テスト完了基準の定義** – テストは、すべての機能テストと非機能テストが完了したときにのみ完了します。
- **テスト ケースの設計** – 各テストケースには、初期状態または前提条件、データ設定、入力、期待値、および実際の結果の 5 つのパラメーターがあります。
- **テストの実行** – 各テストケースは、テスト対象のシステムに対して実行され、結果を文書化します。
- **結果の検証** – テストに合格するには、期待値が実際の結果と一致する必要があります。
- **テストカバレッジの検証** – 要件の機能的側面と非機能的側面の両方をカバーするテストがあることを確認します。

DO-178C では、システム要件からソフトウェアの高レベル要件へ、ソフトウェアの高レベル要件から低レベルの要件へ、そしてテストケース、テスト手順、テスト結果に至るまで、ライフサイクル全体にわたる双方向の要件トレーサビリティを要求します。 低レベルの要件は、それらが実装されるソースコードにリンクする必要があります。構造カバレッジ解析(コードカバレッジ、データ結合、および制御結合)は、システムのソースコードがテストプロセスによってどの程度実行されたかを定量化します。

これらの作業により、すべてのシステム要件に対処するためのコードが実装されていること、実装されたコードが完全にテストされていること、および要件に起因しないコードが存在しないことを確認できます。従来の方法で双方向のトレーサビリティを実現することは可能ですが、プロセスを自動化することで、労力を軽減すると同時に、エラーが発生しにくくなります。(図 1)

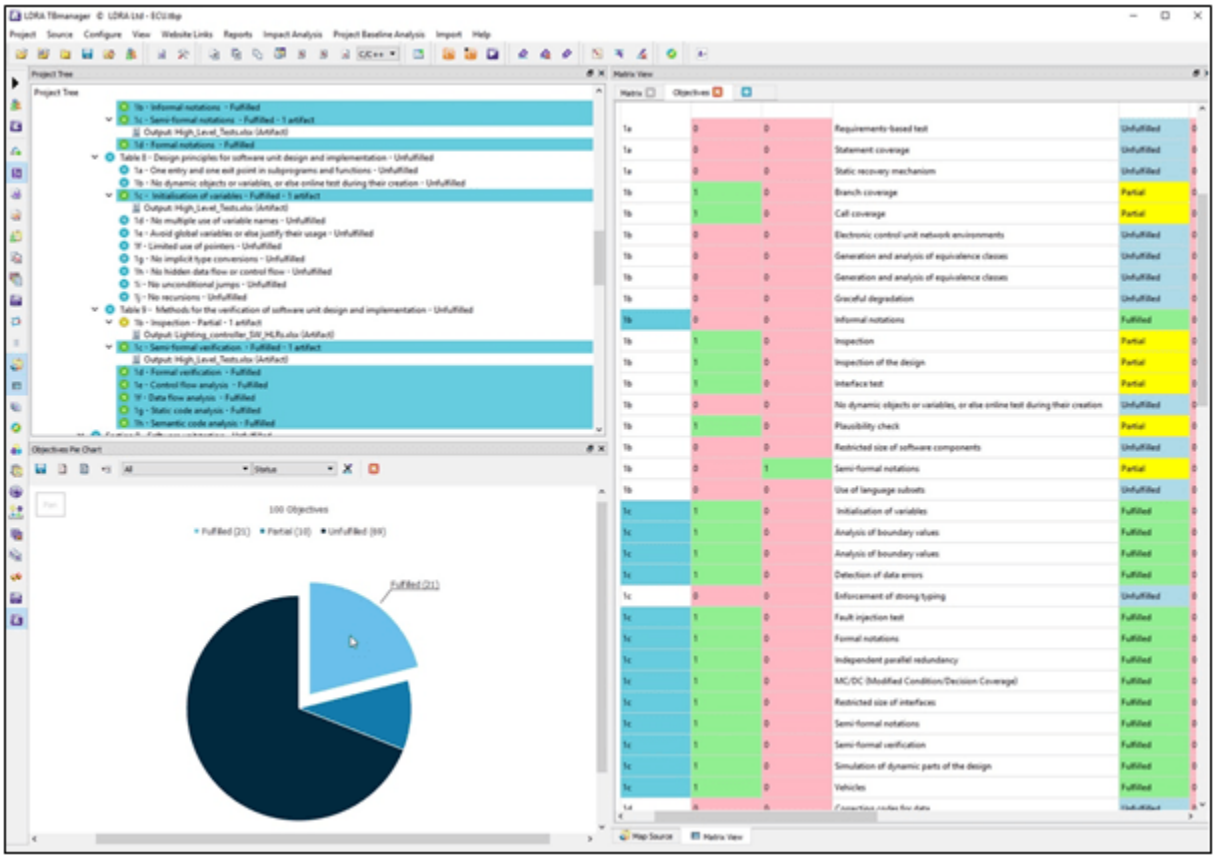

図 1: LDRA ツールスイートの TBmanager を使用した要件トレーサビリティの自動化

データ結合と制御結合の実証(Demonstrating Data Coupling and Control Coupling)

DO-178C セクション 6.4.4.2.c は、「要件ベースのテストがコードコンポーネント間のデータ結合と制御結合を実行したことを確認するための解析」を要求しています。データ結合と制御結合はどちらも実行結果として解析する必要があり、生成される成果物はシステム要件とアーキテクチャに照らしてレビューされます。

制御結合（Control Coupling）

制御結合は、DO-178C によって *「あるソフトウェアコンポーネントが別のソフトウェアコンポーネントの実行に影響を与える方法または程度」* と定義されています。

*第一に、呼び出される可能性のある関数のセットがわかっていることが不可欠であり、第二に、このセットのどのメンバーが実際に呼び出されているかを知っている必要があります。この情報は通常、テスト対象コードの静的解析によって照合されます。要件ベーステストの実行パスの解析により、*関連する目標 A-7.8「ソフトウェア構造のテストカバレッジ(データ結合と制御結合)が達成される」に従って制御結合が実行されたことが示されます。

プロシージャ/関数コールカバレッジの詳細は、多くの場合、認証用の成果物として、必要なレポート形式で提出されますが、グラフィック表現のほうが理解しやすいことから、開発段階では、こちらがより頻繁に使用されます(図 2)。

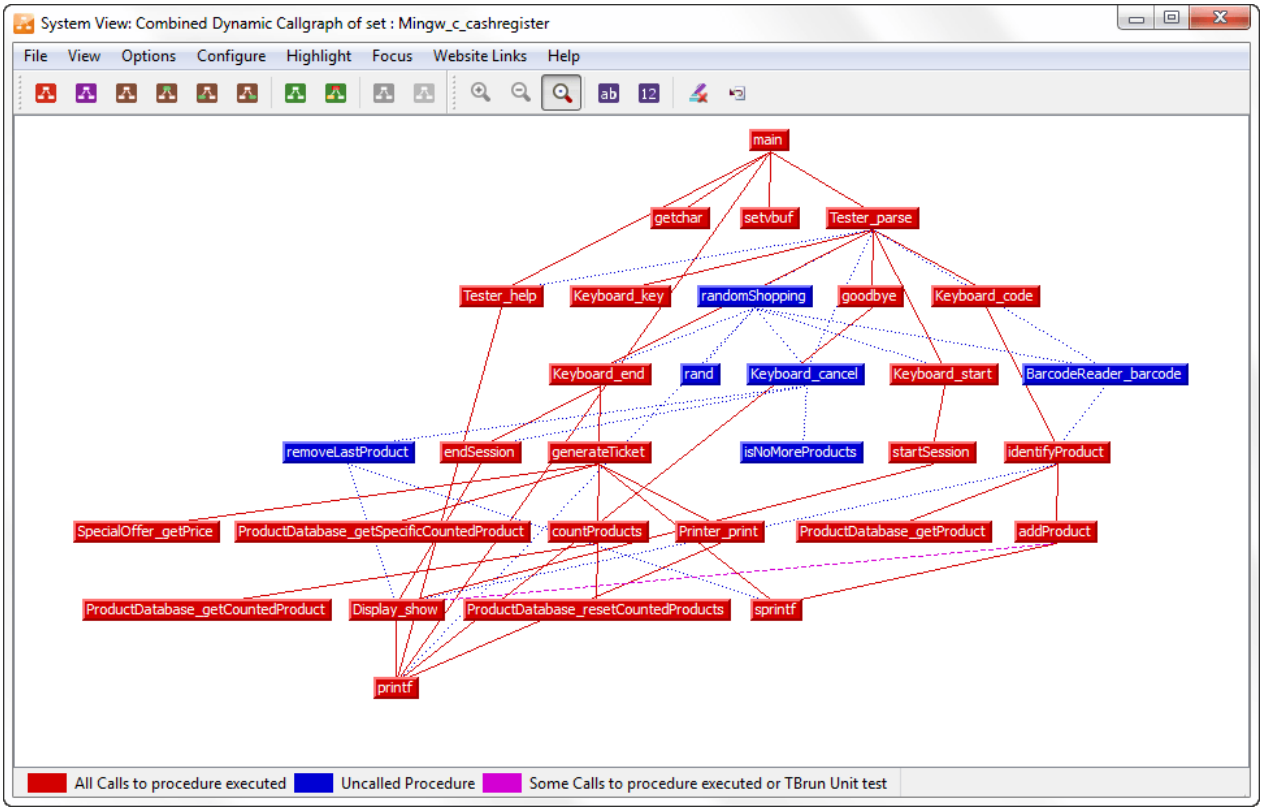

図 2:LDRA ツールスイートによって生成されたコールグラフに見られるプロシージャ/関数コールカバレッジ

制御結合の欠陥（Control coupling defects）

この実行は、DO-178C 等の認証要件を満たすだけでなく、予期しない動作を引き起こす可能性のあるコードの欠陥を明らかにする能力があることに注目ください。

次のモデル（図 3）には 3 つのファイルがあり、それぞれに関数 foo の呼び出しが少なくとも 1 つあります。ただし、2 つのファイルには、関数 foo の同様の定義が含まれています。

リンカは foo の呼出しすべてをファイル A 内の関数定義へのものとして解決すること（ケース 1）もできますし、あるいは、ファイル A 内の foo の呼出しは A 内の定義へ、ファイル B 内の foo の呼出しは B 内の定義へ、そしてファイル C 内の foo の呼出しは A、B いずれかでの定義へのものとして解決すること（ケース 2）もできます

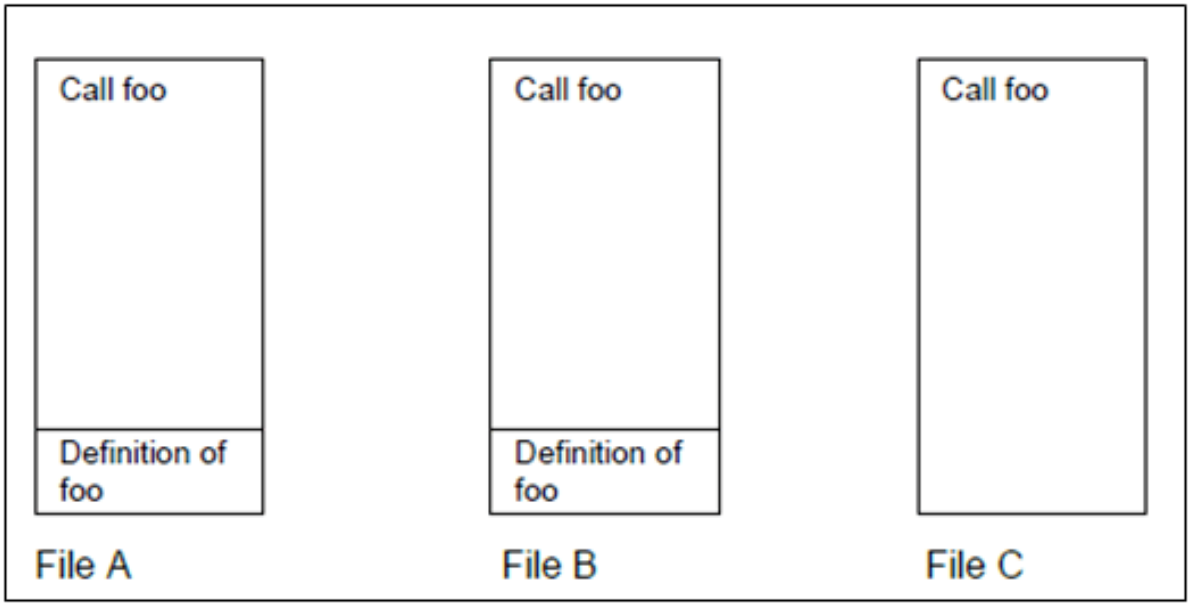

図 3:制御結合の重要性の説明

データ結合（Data Coupling）

データ結合は DO-178C によって「ソフトウェアコンポーネントが、自身の制御下以外にもあるデータに依存すること」と定義されています。 制御結合解析と同様に、データフロー解析は要件ベースのテストの実行から導き出す必要があります。

データ結合解析は、ソフトウェアコンポーネントの境界を越えてデータ要素が設定および使用されるとき（「設定/使用ペア」）の観察と解析に焦点を当て、それらのデータ要素が設計と要件に従って動作することを確認します。 デバッガーを使用してこれらを手動で実行することは、多大な労力がかかり、繰り返すのが難しく、エラーが発生しやすくなります。 この作業を自動化すると、そのオーバーヘッドが大幅に削減されます。

データ結合の欠陥（Data coupling defects）

図 4 の例は DO-248C から複製されており、その実装は右側の関数 *runAirspeedCommand* に示されています。 このソースコードで期待される動作は、最初に対気速度を計算し、次にそれを表示することです。計算して表示するという順序は、「設定/使用」ペアの例です。

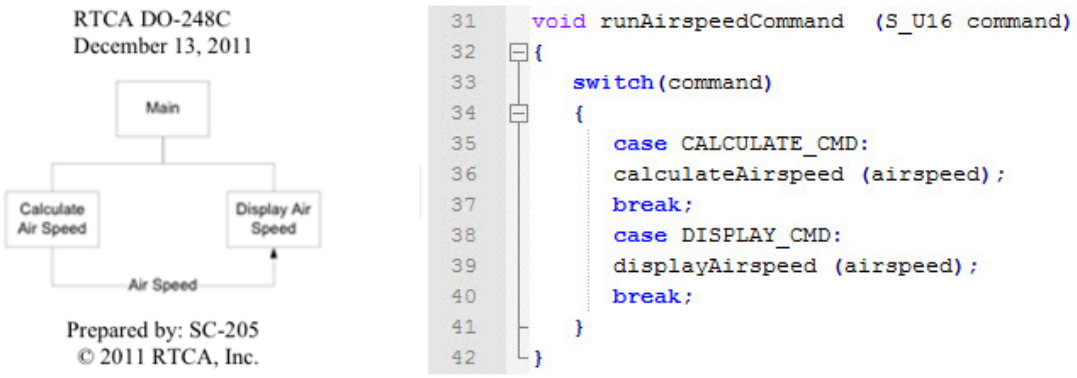

図 4:DO-248C の例

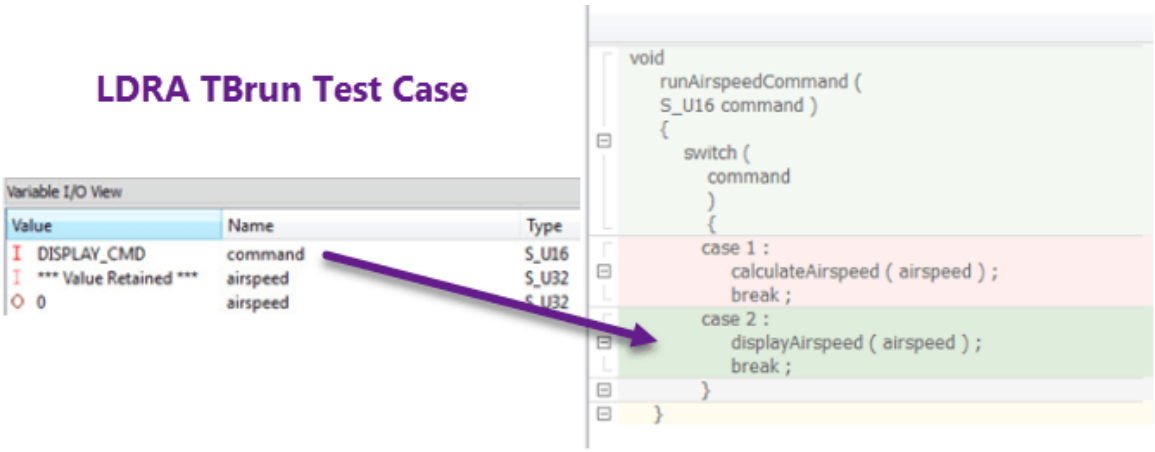

図 5: runAirspeedCommand を実行するテストケースで、結果の制御フローと構造カバレッジが緑色で示されています。LDRA ツールスイートの TBrun の例

図 5 のテストケースは、次の構造カバレッジに反映されているように、switch ステートメントの 2 番目のケースを実行します。 また、対気速度を最新の値に更新（calculateAirspeed）せずに表示（displayAirspeed）が呼び出されることも示しています。追加のテストケースでは、calculateAirspeed コマンドを呼び出す場合がありますが、必ずしも displayAirspeed の呼び出し後に呼び出す必要はありません。

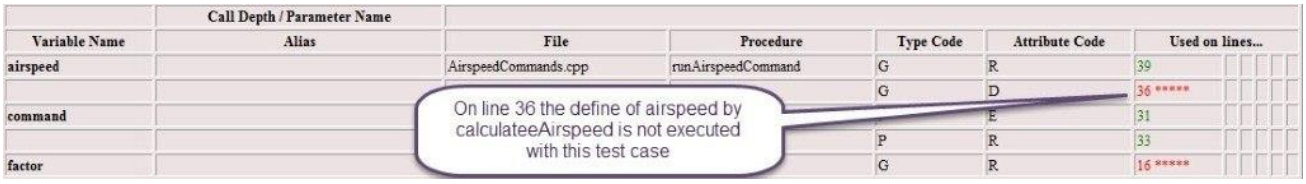

図 6:実行時に参照されていない変数を示す LDRA ツールスイートのレポート。 これらの成果物は、DO-178C 認証の目標 A-7.8 を満たすために使用されます

図 6 のレポートは、上記のテストケースを実行することによって生成されました。 これは動的なデータフロー情報を示しており、対気速度が実際には 36 行目で書き込まれておらず、表示される前に更新されていなかったため、不正確な情報が表示されている可能性があることがわかります。 一般に、観察されたデータフローは、データの相互作用、要件とアーキテクチャ、およびアプリケーションの動作を調整するために必要な情報を提供します。

まとめ

DO-178C の目標 A-7.8 は、「ソフトウェア構造(データ結合と制御結合)のテストカバレッジが達成される」ことを要求しています。

その中の制御結合の側面を満たすには、まず、呼び出される可能性のある関数のセットがわかっていることが不可欠であり、次に、このセットのどのメンバーが実際に呼び出されているかを知る必要があります。この情報は通常、テスト対象のコードの静的解析によって*照合*され、潜在的な制御結合の異常を明らかにすることもできます。

要件ベースのテスト中に実行パスを解析することにより、制御結合が実行されたことが示され、静的解析中に強調表示された潜在的な異常がチェックされます。

データ結合解析は、対気速度などのデータ要素がソフトウェアコンポーネントの境界を越えて設定および使用される(「設定/使用ペア」) ときの観測と解析に焦点を当てています。デバッガーを使用してこれらを手動で実行することは、多大な労力がかかり、繰り返すのが難しく、エラーが発生しやすくなります。 この作業を自動化すると、そのオーバーヘッドが大幅に削減されます。

FUJI SETSUBI

富士設備工業株式会社 電子機器事業部 www.fuji-setsu.co.jp