

大ゼミでのコメント、先生のコメントより、FiindBugsのエラー=TDとするのは無理があると判断した。FindBugsはあくまでコードチェッカーであり、依存関係などを取得することはできない。FnidBugsで検出されるエラーは幅広く、単純な入力ミスからマルチスレッドまで多様である。また、FindBugsで検出できるエラーはCode Smellのほうが意味合いとしては適切である。したがって、TDを分析するためには、依存関係も考慮する必要がある。

上記の理由により、TDの検出をSpotBugsとUnderStandを用いて行う。

SpotBugsはFindBugsの後継プロジェクトである。FindBugsと比較するとより精緻にエラーを検出することが可能となる。

UnderStandはメトリクス解析ツールであり、コード間の依存関係、コードの複雑度を検出するために使用する。最終的には、複数プロジェクトでUnderStandを実行し、解析結果を元にモデルを作成する際の重み付けとなる指標を発見する。検出の手順は次の通りである。

Step1. 対象プロジェクト(Java)からリビジョンIDのリストを取得する。これはSpotBugs, UnderStandをリビジョン毎に実行するためである。

- Step1-1 対象プロジェクトの全コミットIDを取得。

- Step1-2 対象プロジェクトの全コミットのソースフォルダを取得。これはUnderStandを実行する際にソースコードが必要となるため収集する。

Step2. SpotBugsをリビジョン毎に実行する。リビジョンIDはStep1で取得されたものを用いる。制約として、Javaプロジェクトを対象とすること、またビルドツールとしてMavenを使用するツールを対象とする必要がある。

Step3. UnderStandをリビジョン毎に実行する。リビジョンIDはStep1で取得されたものを用いる。

- Step3-1 Step1-2で取得したソースフォルダに対してUnderStandを実行。

Step4. リビジョンIDを用いてSpotBugsで検出されたエラーと依存関係にあるクラスを特定する。

Step5. 依存関係先でTDが波及されていることが確認された場合、そのエラー(TD)は累積したものであると判断する。

Step6. Step5で取得したTDについて、それぞれのメトリクス結果から重み付けを行う。(詳細は下記を参照)

メトリクスから重み付けを行う方法

モデルとして、ソフトウェア信頼度曲線モデルを使用することを試みる。

ソフトウェア信頼度曲線モデルはフォールト(欠陥)の検出数や故障の発生回数を時間の経過とともに観察し、その変化を成長曲線をしてモデル化する手法である。フォールトの検出数(あるいは故障発生回数)の予測や平均故障発生時間間隔(MTBF)の算出といった信頼度の定量的評価に利用できる。

このモデルをTDの定量化モデルに応用することを考える。フォールトの検出数を"TDの検出数"、故障の発生回数を"TDの発生回数"と置き換える。TDの検出事象を1つの確率事象とみなした場合、単位時間に一定のペースでTDを検出するという状況はソフトウェア開発では考えにくい。ため、検出ペースを時間によって変化させる非同時ポアソン過程(NHPP)モデルを使用する。

NHPPの最も基本的なモデルとして、時刻 t の時点で検出済みのフォールト数(期待値)を $H(t)$ 、総フォールト数を a として、単位時間あたりに検出されるフォールト数がその時点での残存フォールトに比例する、すなわち

$$\frac{dH(t)}{td} = b(a - H(t))$$

という微分方程式を解いて得られる指数型ソフトウェア信頼度成長モデル

$$H(t) = a(1 - e^{-bt})$$

がある(bは1個あたりのフォールト検出率)。また、フォールト間の依存関係を考慮してフォールトの検出能力が時間とともに変化する概念を取り入れた、より複合的な習熟S次形ソフトウェア信頼度モデル

$$H(t) = \frac{a(1 - e^{-bt})}{1 + ce^{-bt}}$$

といったものである(cはテスト能力の習熟傾向を表すパラメータ)。近年では、これまでの代表的なソフトウェアモデルを包括した、統合モデルや一般化ガンマソフトウェア信頼性モデル等も研究されている。