

Java アプリケーション開発の脆弱性検知方法の検証及び

新人向け開発研修方法の検討

○伊藤康広、原祐一、福井清悟

工学系技術支援室 情報通信技術系

概要

アメリカ国防省によれば、サイバー空間は「第 5 の戦場」と定義されている。事実、クラッカー（悪意を持ってネットワークの不正侵入やシステムの破壊・改竄を行う者）の攻撃により、日本でも多くの組織で情報漏えいの被害が発生している。セキュリティ対策は大学のみならず、世界中の組織における課題となっている。しかしながら、昔から稼働している Web アプリケーションは今ほどセキュリティを考慮していないコーディングをされている可能性があり、危険な状態で運用されている恐れがある。サーバを安全に運用するためには、OS などを最新の状態に保つだけでなく、コードの品質の向上が欠かせない。

そこで、書かれたコードに複数人が目を通すソースコードのチェック体制を開発の途中に組み込むことにより、バグを含むコードをリリースすることを防止するだけでなく、周囲からの指導を受けやすくすることで能力の底上げができないかと考えた。今回はコードの品質を上げるため、いくつかのツールを開発に導入したので、実際に利用した結果について報告する。

1 検証のための題材について

工学部技術部で利用されている会議室予約システムをリプレースしながら、各種ツールの使い勝手を検証することにした。現在の予約システムの開発元は新バージョンの開発を長期間停止しており、セキュリティ対策として、新しく開発した方が適切であるという状況になっているためである。

2 開発に利用したツール

ここでは、今回の開発プロジェクトで新たに利用を開始した主要なツールについて列挙する。

2.1 Redmine

Redmine はプロジェクト管理ツールである。これまでの進捗管理は Excel シートを用いて行うことはあったが、専用のツールを用いることで、進捗管理を容易にするために用いた。本年度は新人研修に適用するため、他のツールよりも先行して利用を開始し、遠隔地の研修管理者でも研修の状況を容易に把握できるということを確認した。

Redmine の特徴はチケットを利用した進捗管理である。チケットは 1 つの課題に相当する。複数のチケットの状況をガントチャートで可視化することにより、進捗状況の把握を容易にしている (図 1)。

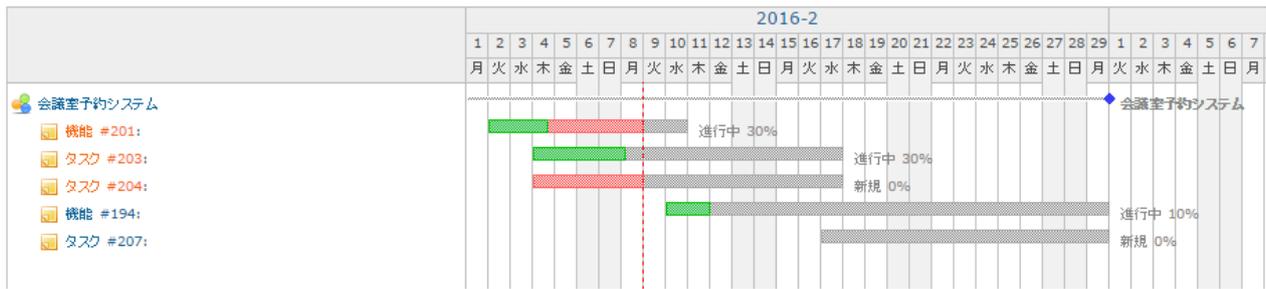


図 1. Redmine におけるガントチャートの表示（注：機能やタスクの名前は伏せています）

2.2 FindBugs

バグになる可能性のあるコードを検出するためのツールである。Eclipse 4.5 (Pleiades All in One) を用いて Java で開発する場合、Findbugs はプラグインとして最初から組み込まれている。Eclipse は文法エラーについては指摘してくれるが、文法上は正しくても内容的に正しくないコードまでは指摘してくれない。Findbugs は登録された検出ルールに基づいてバグの可能性のあるコードを指摘してくれるため、その内容を検討、反映することで、より安全なコーディングができるようになる。

2.3 Git

Git はバージョン管理ツールである。バージョン管理ツールを使わないでコードを管理する場合、開発中のコードがあるディレクトリに対し、日付などをつけて別名で保存するといった方法が考えられるが、それでは何を変更したのかということが一目で分からないという問題が発生する。バージョン管理の専用ツールを使うことで、いつ誰がどのような修正を行ったかを可視化することができ、バグの追跡やコード一式をある時点に戻すといったことが容易になる。

3 開発・運用の環境

今回の開発では主に以下のプロダクトを用いている。

表 1. 開発で採用した主要プロダクト一覧

製品	バージョン
開発環境	Windows7 + Eclipse 4.5
Java	8 系
Tomcat	8 系
Spring Framework	4.2 系
MariaDB (データベース)	5.5 系
運用環境	CentOS 7 系

4 開発プロセスへのツール導入

初めて導入するツールについて、それぞれどのように導入をしたかということについて説明する。

4.1 FindBugs の定期的な利用

FindBugs は Eclipse に初めからインストールされており、単に実行するだけであれば容易である。Eclipse でのプロジェクト作成後、プロジェクト・エクスプローラで、作成したプロジェクトを右クリックし、バグを検索とすれば FindBugs の実行が始まる（図 2）。

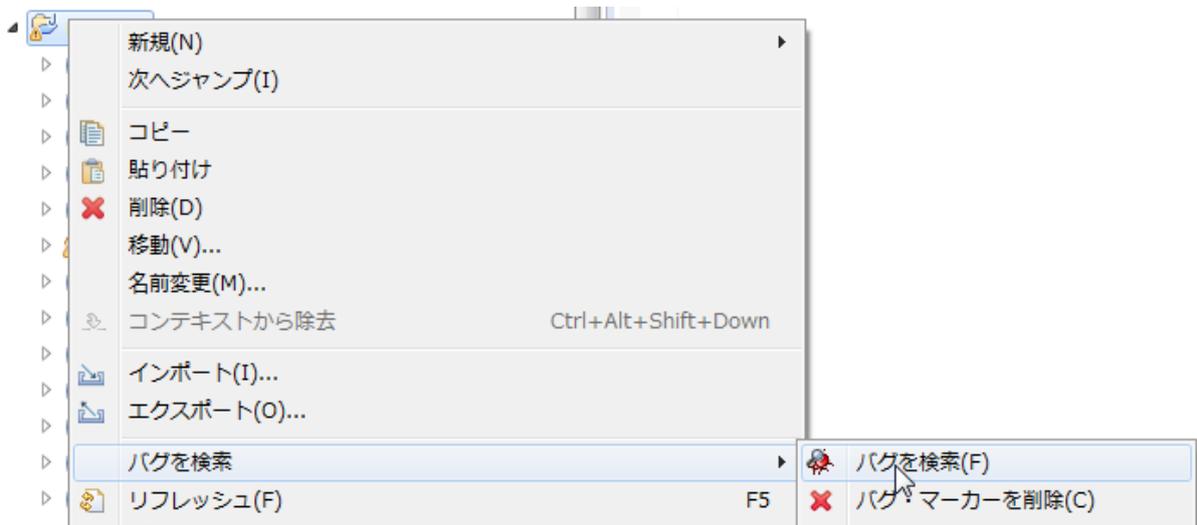


図 2. FindBugs の実行方法

開発中のプロジェクトでは約 1 分で FindBugs の実行結果が得られた。実行後に問題と判断されたコードがあれば、該当するコードの行に対して虫のマークを出すようになっている（図 3 の丸で囲った箇所）。インタフェースから関数を自動生成すると、未実装の初期状態では null を返すようになっている。本来は論理値 (true か false) を返すべき関数であるが、明示的に null を返すのはバグの原因になるということで、良くない習慣であるという結果が返されてくる。FindBugs を導入することで、誤りの少ないコードを書くための指針を得ることができる。開発中は定期的に行うことで問題になりうる箇所を検出し、コードの品質を保つようにした。なお、開発ルールに従って独自にバグ検出の基準をカスタマイズすることも可能であるが、本開発では、既定値で進めることとした。

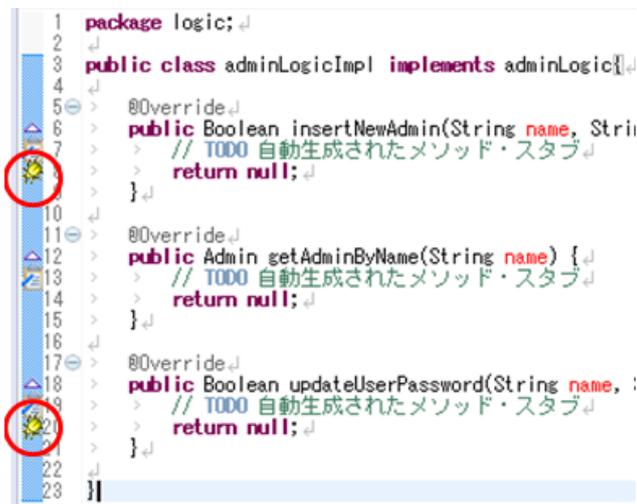


図 3. FindBugs の実行結果例

4.2 Redmine での定期的な報告

今回は個人での開発となっているが、そのような場合であっても Redmine はメモの代わりとして役立つ。特に翌日に作業を持ち越す場合や、バグを見つけた場合など、対処しなければならないことを Redmine に登録することによって、作業の漏れを防ぐことができた。

4.3 Git を用いたソースコード管理

Git はリポジトリ（クライアント側はローカルリポジトリ、サーバ側はリモートリポジトリ）を作るためのツールであるが、クライアントと Git サーバを連携させるためのツールも同時に導入する必要がある。Eclipse

のプラグインとしてEGitというツールもあるが、プロジェクト・エクスプローラの表示が複雑になることと、GUIで表示されたリポジトリの情報が欲しいことから、SourceTreeを用いることにした（図4）。



図4. SourceTreeにおけるブランチの表示（注：説明は開発内容を含むため消しています）

4.4 Gitのブランチ戦略

Gitの運用方法は自由に決められるが、リポジトリに大きく分けて3種類のブランチを用意する方針とした。master ブランチ、release ブランチ、develop ブランチである。develop ブランチは普段の開発を進めるブランチである。release ブランチは、ある時点の develop ブランチをマージしたもので、ここに置かれたリポジトリの内容をリリース対象とする。master ブランチは develop ブランチをマージするだけで開発を行わないブランチとし、バージョン番号を表すタグを置くようにする。develop ブランチはある程度の形になるまでは1本で運用し、開発中盤以降は開発箇所ごとにブランチを派生させて、開発完了後に develop ブランチに統合する方針とした。

4.5 Redmine と Git の連携

Redmine はソフトウェア開発のプロジェクト管理を前提にして作られているので、Git のようなバージョン管理ツールとの連携も行うことができるように作られている。Redmine のプロジェクトの設定にリポジトリを登録する項目があるので、その設定を行った（図5）。

図5. リポジトリ設定画面

Redmine と Git を連携させるには、リポジトリが Redmine と同じサーバにあることを前提としている。今回の開発では Redmine のみを新人研修のために先行して作成し、Git サーバは別サーバで後から構築していた。そのため、Git サーバのディレクトリを Redmine がマウントすることによって、リポジトリを認識させることにした。Redmine の側からリポジトリの内容が閲覧できるよう、リポジトリが更新された後に実行されるフ

ックスクリプト `post-update` を書いて権限の設定を行った。

Redmine と Git が連携すると、ブラウザ上でリポジトリの 2 つの時点における差分を確認することが簡単にできる (図 6)。これにより、プロジェクト関係者によるコードチェックが容易になる。

The screenshot shows the Redmine interface for a repository named 'roomreserve' on the 'develop' branch. On the left, there is a file browser showing 'roomreserve' and '.gitignore'. Below it is a table of recent revisions. On the right, a diff view for the file 'roomreserve/css/common.css' is shown, highlighting changes in CSS rules for link and hover states.

#	日付	作成者	
5146664e	2016/02/03 10:21	Yasuhiro Ito	[add] 予約
52941aa7	2016/02/03 10:21	Yasuhiro Ito	[modify]
1003546f	2016/02/03 10:19	Yasuhiro Ito	[modify] - 予約表示 - 1pmを1
f54c30de	2016/02/03 10:16	Yasuhiro Ito	[add] 予約
2c1f277c	2016/02/03 10:15	Yasuhiro Ito	[modify]
05d03a77	2016/02/03 10:14	Yasuhiro Ito	[modify]
1165c38b	2016/02/03 10:13	Yasuhiro Ito	[modify] テーブルの
5e935f2b	2016/02/01 16:22	Yasuhiro Ito	[modify]
31328467	2016/02/01 16:21	Yasuhiro Ito	[modify]
b739c8f4	2016/02/01 16:20	Yasuhiro Ito	[modify]

```
129 129      /* height: 15px; */
130 130 }
131 131
132 132 .purpose a {
133 133     text-decoration: none;
134 134 }
135 135
136 136 .purpose a:link, a:visited, a:active {
137 137     color: #404040;
138 138 }
139 139
140 140 .purpose a:hover {
141 141     color: #ff8000;
142 142 }
143 143
144 144 .edit a:link, a:visited, a:active {
145 145     color: #0000ff;
146 146 }
147 147
148 148 .edit a:hover {
149 149     color: #ff8000;
150 150 }
151 151
132 152 /* 予約画面 */
133 153 #reserve_article {
134 154     width: 500px;
```

図 6. Redmine 上でのリポジトリの表示 (注: コメントは開発内容に関わるため伏せています)

ローカルリポジトリにコミットする際に、Redmine で割り振ったチケットの番号をコミットメッセージにつけることで、開発内容と Redmine のチケットを関連づけることもできる。図 7にある「refs #186」(186 はチケット作成時に Redmine によって自動的につけられた番号) というキーワードをコミット時につけることで、チケットと関連付けられるようになる。

関係しているリビジョン

The screenshot shows a commit message in Redmine. It includes the revision ID '675b3767', the author 'Yasuhiro Ito', and the commit message '[modify] 会議室に関する修正'. Below the message, it says 'DBから会議室一覧を取得するよう、SQL、XML、JSPなどを修正した。' and 'refs #186'.

図 7. 開発内容のチケットへの反映

5 カレンダーについて

予約システムで欠かせないものに、現在の予約状況を把握するためのカレンダーが挙げられる。カレンダーの表示のために 2 つのライブラリを用いたので、それらの概要について紹介する。

5.1 AJD4JP

Java でカレンダー情報を扱うためのライブラリである。このライブラリの特徴は日本の祝日情報が用意されていることである。例えば、「2016 年 8 月 11 日は山の日」のような情報を取り出すことも可能である。本システムでは、毎月の祝日名を表示するために用いた (図 8)。

自動的に祝日情報を計算してくれるのは便利であるが、祝日の追加や変更が生じた場合、ライブラリを差し替えるためのメンテナンスが必要となる。

2016年3月						
1年前	1月前	今月	1月後	1年後		
日	月	火	水	木	金	土
		1 08:30~17:15 名大研修会	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20 春分の日	21 振替休日	22	23	24	25	26
27	28	29	30	31		

図 8. AJD4JP を用いたカレンダーの例

5.2 FullCalendar

FullCalendar は jQuery と連携して動作する Javascript のライブラリである。FullCalendar を用いると、プログラマ側でデザインを作成することなく、商用 Web サービスのカレンダーのような表示を実現することができる。本システムでは予約時に参照する 1 日の予約状況を表示するために用いた (図 9)。

[カレンダーに戻る](#)

*は必須項目です。

予約日	2016年 3月 1 日
会議室名	工7号館
開始予定時刻	08 時 30 分
終了予定時刻	17 時 15 分
会議目的*	
会議詳細 (100字以内)	
登録者*	
連続予約	<input checked="" type="radio"/> しない(今回限り)
	<input type="radio"/> 予約日を含め連日同一時間帯で 2 回
	<input type="radio"/> 予約日を含め毎週同一時間帯で 2 回
予約削除キー*	(半角数字4桁)

3/1(火)の予約状況

図 9. FullCalendar を用いた予約画面の例 (注: 会議室名は一部伏せています)

6 開発ツールの新人研修への適用

既に Redmine は新人研修に利用してきたが、Git を新たに導入することにより、新人研修の形式を工夫することができると思われる。例えば、新人研修では静的な Web ページを作るという課題があるので、それに Git と連携した Redmine を適用することで、成果物である HTML のコードがブラウザ上で閲覧でき、より正確に進捗状況を可視化できるようになる。新人が研修で複数のデザイン案を作成した場合には、案ごとに Git

のブランチを用意してもらえれば、研修を指導する側はリポジトリからコードを取得して、ブランチを切り替えながら手元のブラウザで成果物を確認することができる。また、Eclipse を用いて Web ページを作成するように指導すれば、Java を用いた Web アプリケーション開発の準備にもなる。

7 まとめ

開発にさまざまなツールを導入したことにより、開発の進捗とコードの状況を把握しやすくするとともに、書かれたソースコードをチェックするための体制づくりを行うことができた。特に Git を導入したことによって、普段利用している PC とは別の箇所にバックアップとしておけるので、開発する上での安心感が高まった。ただし、Git は個人での利用にとどまったので、複数人で開発した際に発生する可能性があるトラブルが起きていない。そのような場合のトラブル解決については今後の課題としたい。

参考文献

- [1] 前田剛 著、入門 Redmine 第 4 版、秀和システム
- [2] 松下雅和ほか著、開発効率を UP する Git 逆引き入門、C&R 研究所