

การวิเคราะห์และบทสรุปในการเลือก
Open-source software project

โดย

633020556-0 นายนรบดี เดชขันธุ์
633021009-4 นายอัครเดช วิทยาอุฒิรัตน์

เสนอ

อาจารย์ที่ผศ.ดร.ชิตสุธา สุ่มเล็ก

รายงานนี้เป็นส่วนหนึ่งของวิชาศึกษาวิชา SC313504 Software Quality Assurance

ภาคเรียนที่ 1 ปีการศึกษา 2565

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาการคอมพิวเตอร์

มหาวิทยาลัยขอนแก่น

คำนำ

รายงานเล่มนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของรายวิชา SC313504 SOFTWARE QUALITY ASSURANCE เพื่อศึกษาหาความรู้ในเรื่อง การวิเคราะห์คุณภาพของซอฟต์แวร์ด้วย Software metric/Code metric โดยใช้เครื่องมือ ประเภท Static หรือ Dynamic analysis tools และได้ศึกษาอย่างเข้าใจเพื่อเป็นประโยชน์กับการเรียน ผู้จัดทำหวังว่า รายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักเรียน นักศึกษาที่กำลังหาข้อมูลเรื่องนี้อยู่ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้และขออภัย ณ ที่นี้ด้วย

ผู้จัดทำ

นายอัครเดช วิทยาธุริรัตน์

นายนรบดี เดชขันธ

สารบัญ

เรื่อง	หน้า
Purpose	1
Application Overview	1
Testing Scope	1
Test Environments and Tools	1
Conclusion	2
Functional Testing (Junit)	2
Cyclomatic Complexity	3
Source Lines of Code	4
Lack of Cohesion of Methods	5
Class Response	6
Quality Gates	7
อ้างอิง	8

1. Purpose

จุดประสงค์เพื่อทดสอบการทำงานของ Source Code ว่าทำงานถูกต้องตามที่ควรจะเป็นหรือไม่ และ ทดสอบคุณภาพของ Source Code รวมถึงตรวจสอบความซับซ้อนของ Method ต่าง ๆ ที่อยู่ภายในตัวโปรแกรม

2. Application Overview

“BlackJack” เป็นเกมไพ่รูปแบบหนึ่งที่ผู้เล่นต้องวางเดิมพัน Dealer ก็จะเริ่มแจกไพ่ให้ฝั่งละ 2 ใบ หน้าที่ต่อไปคือการทำการจั่วไพ่เพิ่มเติมไปเรื่อย ๆ จนกว่าแต้มของคุณจะใกล้เคียงหรือเท่ากับ 21 แต้ม แต่หากเกิน 21 แต้ม เมื่อไร จะถือว่าแพ้ทันที

3. Testing Scope

- Functional Testing
- Cyclomatic Complexity Testing
- Source Lines of Code Testing
- Lack of Cohesion of Methods Testing
- Class Response Testing

4. Test Environment & Tools

- Eclipse IDE for Java Developers
- JavaSE-18
- Junit 4
- CodeMR
- JArchitect

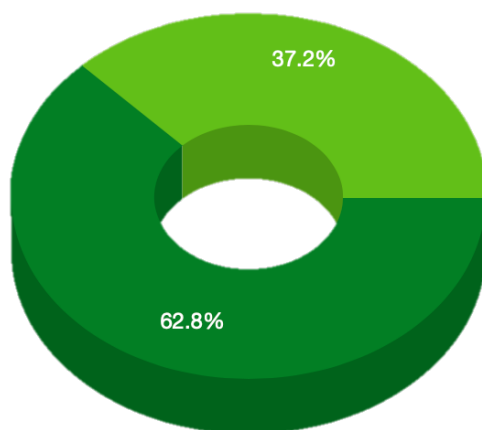
5. Conclusion

- Functional Testing (JUnit)

Test Case Design and Test Results								
Project Name: Test of getWin method, of class Game.			Project ID: 1					
Test Strategy:			Designer: Akkaradet Wittayawutitrat, Narabodee Dachkan					
Test Environment:			Eclipse IDE 2020 (java-SE 18), Junit 4					
Scenario ID	Test Scenario	Pre-requisite	Step No.	Description	Expected Result	Actual Res	Status (Pass/Fail/No run)	Remark/Defect ID
TS001	Player Win	-	1	PlayerScore more than Dealer Score ,PlayerScore less than 21	Player win	Player win	Pass	
			2	Dealer Score more than 21, PlayerScore less than 21	Player win	Player win	Pass	
			3	PlayerScore equal 21 (BlackJack), DealerScore less than 21	Player win	Player win	Pass	
			4	PlayerScore equal 21 (BlackJack), DealerScore more than 21	Player win	Player win	Pass	
TS002	Player Lose	-	1	DealScore more than Player Score , DealerScore less than 21	Player lose	Player lose	Pass	
			2	Player Score more than 21, DealerScore less than 21	Player lose	Player lose	Pass	
			3	DealerScore equal 21 (BlackJack), PlayerScore less than 21	Player lose	Player lose	Pass	
			4	DealerScore equal 21 (BlackJack), PlayerScore more than 21	Player lose	Player lose	Pass	
TS003	Draw	-	1	DealerScore equal PlayerScore, Both score are less than 21	Player lose	Player lose	Pass	
			2	DealerScore equal PlayerScore, Both score are more than 21	Player lose	Player lose	Pass	
			3	Both BackJack(Both Score equal 21)	Player lose	Player lose	Pass	

การทดสอบเฉพาะหน่วย (Unit Testing) ผู้ทดสอบได้แบ่ง Test Scenario เป็น 3 ส่วน โดยหลักการสร้าง Test Scenario ยึดหลักกติกาเกม Black Jack ที่ผู้พัฒนาได้สร้างขึ้น ซึ่งเหตุการณ์ที่เกิดขึ้น สามารถแบ่งได้หลัก ๆ 3 เหตุการณ์ คือ เหตุการณ์ที่ Player ชนะ, Player แพ้ และ เหตุการณ์ที่แต้มไพ่ของทั้ง Player และ Dealer เท่ากัน

จากการทดสอบสามารถสรุปได้ว่า โปรแกรมเกม BGS มีการทำงานที่ในส่วนของการประมวลผลผู้ชนะในเกมถูกต้อง 100% ซึ่งผู้ทดสอบได้ใส่ค่าเข้าไปใน Method whoWon() และ Method ได้ Return ค่าออกมาตรงตามความคาดหวังของผู้ทดสอบทั้ง 11 Testcase ซึ่งผลลัพธ์ที่คาดหวัง ยึดหลักความถูกต้องตามกติกาของเกมทั้งหมด



Cyclomatic Complexity

- Cyclomatic complexity

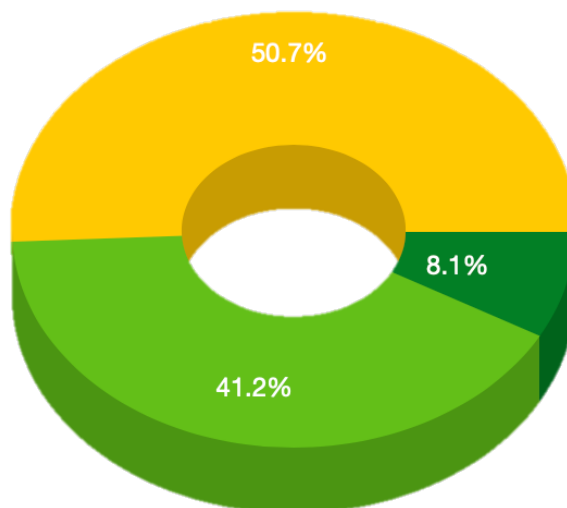
ความซับซ้อนในการเขียนโปรแกรม(หรือความซับซ้อนของซอฟต์แวร์) เป็นคุณสมบัติอย่างหนึ่งของซอฟต์แวร์ ซึ่งทั้งหมดนี้ส่งผลต่อการโต้ตอบภายในตัวโปรแกรมโดยตรงความซับซ้อนอธิบายการได้จากการที่มีเอนทิตีตัวหนึ่งมีความเชื่อมโยงกันเอนทิตีตัวอื่น ซึ่งอาจจะอยู่คนละคลาสกัน หรือ ต่าง Method เมื่อจำนวนของเอนทิตีเพิ่มขึ้นจำนวนของการโต้ตอบและการทำงานของโปรแกรมก็จะเพิ่มขึ้นแบบทวีคูณระดับความซับซ้อนที่สูงขึ้นในซอฟต์แวร์จะเพิ่มความเสี่ยงที่จะเกิดความผิดพลาดในการประมวลผลได้

จากการวิเคราะห์พบว่าในตัวโปรแกรม BGS มีค่า Cyclomatic Complexity อยู่ในระดับต่ำ ถึง ปานกลาง สามารถสรุปได้ว่าตัวโปรแกรม BGS นั้นมีโอกาสเกิดข้อผิดพลาดจากการทำงานของ Method อยู่ในระดับต่ำ แต่อย่างไรก็ตาม Method ที่ไม่มีเงื่อนไข และ loop จะมีค่าความซับซ้อนเท่ากับ 1 ถ้า method นั้นมี 1 เงื่อนไข จะมีความซับซ้อนเท่ากับ 2 เนื่องจากมีเส้นทางการทำงาน 2 ทางผู้ทดสอบก็ควรเขียน Unit test ก่อน เพื่อให้ครอบคลุมในทุกๆเส้นทาง เนื่องจากสิ่งที่คุณพัฒนาสร้างมีค่า Cyclomatic Complexity ในระดับปานกลาง เพื่อหลีกเลี่ยงการทำงานผิดพลาดหรือการทำงานที่ไม่ครอบคลุมของ Software ทางผู้ทดสอบควิเขียน Unit test ทุกครั้งที่มีการ สร้าง Method ขึ้นมา

Total 26 classes

37.2 % 2 classes with low-medium Complexity

62.8 % 24 classes with low Complexity



Source Lines of Code

- Source Lines of code (SLOC)

เป็นวิธีการที่นิยมใช้ในการประมาณซอฟต์แวร์ใช้ในการวัดขนาดของโปรแกรมคอมพิวเตอร์ โดยการนับจำนวนบรรทัดนั้นโดยจะไม่นับรวมบรรทัดว่าง, บรรทัดที่มีข้อความอธิบาย โดยทั่วไปแล้วจะใช้เพื่อคาดการณ์ปริมาณความจำเป็น เช่น โปรแกรมเมอร์ใช้กี่คนใช้เวลานานประมาณเท่าไร

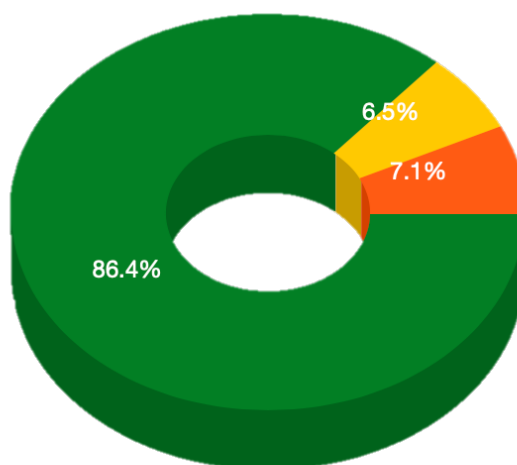
จากการทดสอบโปรแกรม BGS ด้วยเครื่องมือ CodeMR ทำให้ทราบถึงจำนวนบรรทัดทั้งหมดของโปรแกรม และ จำแนกออกมาเป็นจำนวนบรรทัดของแต่ละ Method สามารถแสดงเป็นเปอร์เซ็นต์ได้ดังกราฟ ซึ่งสามารถสรุปได้ว่า โปรแกรม BGS นั้นออกแบบมาได้ค่อนข้างดี เพราะจำนวนบรรทัดไม่เยอะมาก และสามารถทำงานตามวัตถุประสงค์ได้ ส่งผลดีต่อการ Maintenance ตัวโปรแกรมในอนาคต รวมถึง ช่วยลดแนวโน้มที่จะเกิดข้อผิดพลาดของการทำงานภายในตัวโปรแกรมอีกด้วย

Total 26 classes

50.7 % 2 classes with medium-high Class Lines of Code

41.2 % 10 classes with low-medium Lines of Code

8.1 % 14 classes with low Class Lines of Code



Lack of Cohesion of Methods

- Lack of Cohesion of Methods

การขาดการประสานกันในเมธอดเป็นการวัดจำนวนคู่ของเมธอดที่ไม่ได้เชื่อมกันภายในคลาสที่แสดงขึ้นส่วนอิสระที่ไม่มีการติดต่อกัน แสดงถึงความแตกต่างระหว่างจำนวนของคู่เมธอดที่ไม่มีตัวแปรเหมือนกัน และจำนวนของคู่เมธอดที่มีตัวแปรเหมือนกัน

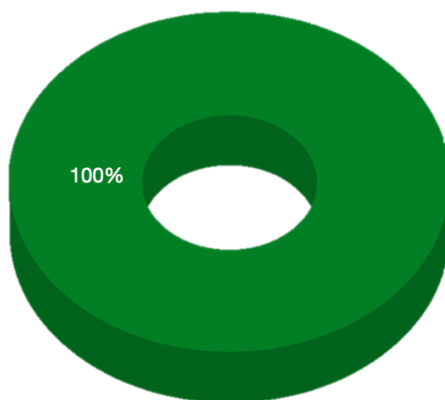
จากการทดสอบโปรแกรม BGS ด้วยเครื่องมือ CodeMR ทำให้ทราบว่าภายในตัวโปรแกรม BGS มีคลาสที่เชื่อมต่อกัน และ มีการสืบทอดตัวแปร หรือ ฟังก์ชันต่อกันเป็นส่วนมาก ซึ่งเป็นผลดีเพราะ ทำให้ Developer ใหม่ทำงานได้เร็วขึ้น – ถ้า Code เป็น Structure อยู่แล้ว ทำให้คนใหม่ที่เข้ามาก็สามารถเข้าใจ Code ได้ง่ายขึ้น ลดความผิดพลาดในการทำงาน ยังรวมถึง ความปลอดภัยที่มากขึ้น – เนื่องจาก OOP นั้นเราสามารถกำหนดได้ว่าจะให้ใครบ้างที่สามารถเข้าถึง Attribute หรือ Method นั้นได้บ้าง ช่วยลดโอกาสที่ผิดพลาดจากการให้ Object อื่นภายนอกที่ไม่ได้รับอนุญาตให้เข้าถึงข้อมูล

Total 26 classes

7.1 % 2 classes with high Lack of Cohesion of Methods

6.5 % 2 classes with medium-high Lack of Cohesion of Methods

86.4 % 22 classes with low-medium Lack of Cohesion of Methods
















Class Response

- Class Response

คือการทดสอบการตอบสนองของคลาสแต่ละคลาสว่ามีการเรียกใช้งานไปแล้วสามารถตอบสนองส่งค่ากลับออกมา หรือสามารถทำงานได้ตามความต้องการของผู้ใช้ได้หรือไม่ โดยคลาสที่มีการตอบสนองดี จะส่งผลให้ Software มีความลื่นไหล และ มีโอกาสเกิดข้อผิดพลาดได้น้อย ซึ่งการทดสอบนั้นจะทำการเรียกคลาสขึ้นมา และ ทดลองสร้าง Model หรือ Object แล้วเรียกใช้ Method ดูการตอบสนองว่ามีการตอบสนองอยู่ในระดับใด โดยอาจวัดได้หลายด้าน เช่น ความเร็วในการตอบสนอง หรือ ความถูกต้องของการตอบสนอง

จากการทดสอบโปรแกรม BGS ด้วยเครื่องมือ CodeMR ทำให้ทราบว่า ภายในตัวโปรแกรมนั้น คลาสในแต่ละคลาสมีการตอบสนองอยู่ในระดับดีมาก โดยผลการทดสอบพบว่าทุกคลาสในโปรแกรม BGS ตอบสนองทุกคลาสเมื่อใช้เครื่องมือทดสอบ ได้ผลออกมาเป็น 100% Class Response ซึ่งสรุปได้ว่า ตัวโปรแกรมนั้นถูกออกแบบมาได้ดี ทุกคลาสมีการเรียกใช้ และ ถูกสร้างขึ้นมามีใช้งานภายในตัวโปรแกรม

5 quality gates	Value	Status
5 quality gates matched		
 Blocker Issues	0 issues	 Pass
 Critical Issues	0 issues	 Pass
 Critical Rules Violated	1 rules	 Fail
 Percentage Debt	7.39 %	 Pass
 Debt Rating per Package	1 packages	 Fail

Quality Gates		
	Fail	2
	Warn	0
	Pass	3

- Quality Gates

คือการทดสอบคุณภาพของโปรแกรมก่อนการส่งต่อถึงผู้ใช้ Quality Gates ใช้วัดการทดสอบซึ่งจะส่งผลกลับมาออกเป็น Pass, Warn และ Fail ซึ่งจะมีการตั้งมาตรฐานของแต่ละGate เป็นเปอร์เซ็นต์แล้วทดสอบออกมาและแจ้งรายละเอียดว่ามีกี่ Gates ที่ทดสอบผ่าน

จากการทดสอบโปรแกรม BGS ด้วยเครื่องมือ JArchitect ทำให้ทราบว่า มีจำนวน 2 Gates ที่ไม่ผ่าน Quality Gates และ 3 Gates ที่ผ่าน Quality Gates จึงสรุปได้ว่าซึ่งสรุปได้ว่า ตัวโปรแกรมนั้นมีคุณภาพของ Software ที่ยังมีข้อบกพร่องอยู่ ควรทำการแก้ไขในส่วนที่ทำให้ไม่ผ่าน Quality Gates ก่อนทำการส่งชิ้นงานถึงมือลูกค้า

อ้างอิง

Hamilton, T. (2022). *Test Environment for Software Testing*. Guru99.

<https://www.guru99.com/test-environment-software-testing.html>

O. (2021). *Software Tester (Test script | Test case | Test scenario)*. medium.

<https://medium.com/@srivichai29/test-ee39fb6772cb>

(n.d.). *Development and Test Environments: Understanding the Different Types of Environments*. unitrends. <https://www.unitrends.com/blog/development-test-environments>

(2022). *วิธีเล่นไฟแบล็คแจ็คสำหรับมือใหม่ 2022*. aviancethailand. <https://www.aviancethailand.com/วิธีเล่นไฟแบล็คแจ็คสำ/>

(2022). *How To Write An Effective Test Summary Report*. softwaretestinghelp.

<https://www.softwaretestinghelp.com/test-summary-report-template-download-sample/>