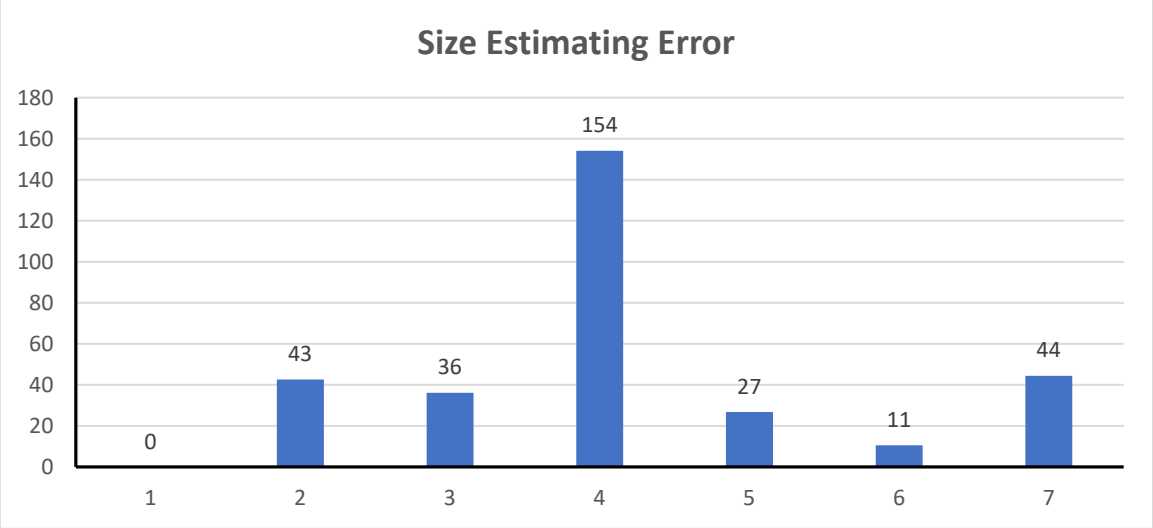
1. ตารางสรุปข้อมูล

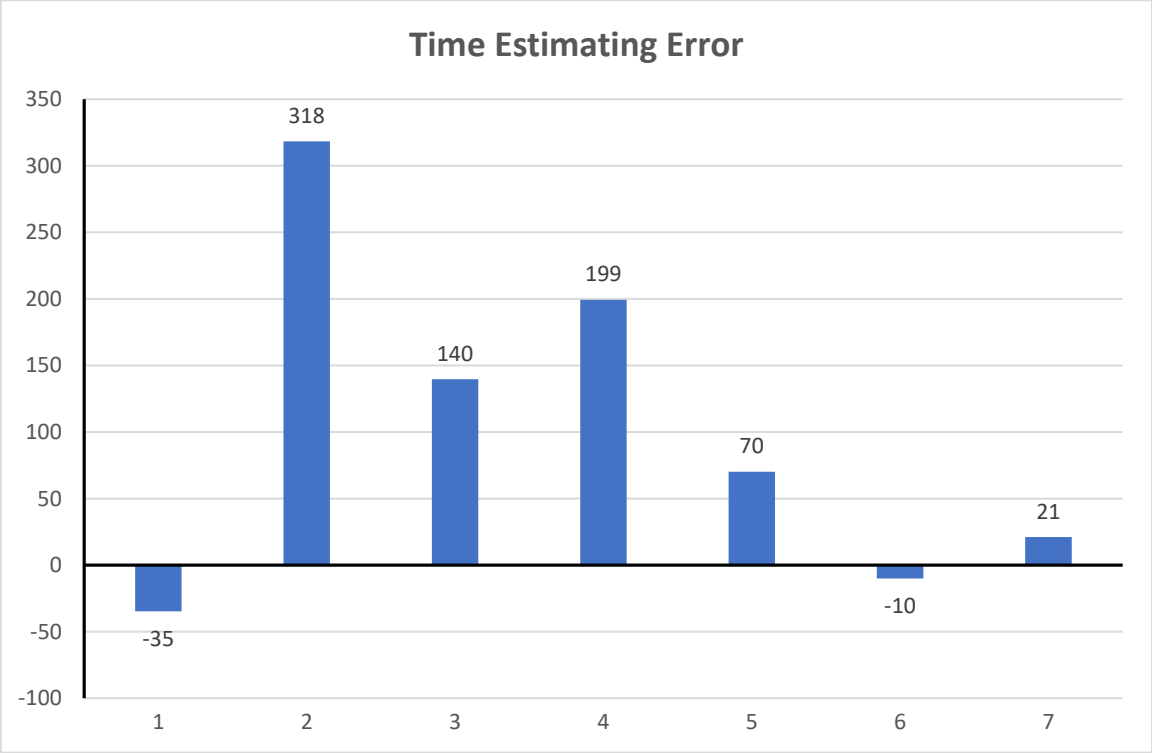
โปรแกรม	Plan			Actual			Error (%) [(A-P)*100/P]			Defects [Defect * 1000 / A.A&M]		Productivity [A.A&M * 60 / A.Time]	Review Rate [A.A&M * 60 / Time Spent in	
	Size	A&M	Time	Size	A&M	Time	Size	A&M	Time	รวม	Total/K LOC		Design	Code
1			60			39			-35	3				
2	192	79	90	225	112	376	17	43	318	8	71	18		
3	72	59	52	95	80	125	32	36	140	3	38	38		
4	37	22	59	72	57	175	92	154	199	5	88	20	428	143
5	68	68	139	86	86	236	27	27	70	9	105	22	397	198
6	103	69	199	110	76	179	7	11	-10	4	53	25	207	217
7	195	91	279	232	132	338	19	44	21	6	45	23	466	273

2.1 แนวโน้มความแม่นยำในการทำนายขนาดของโปรแกรม (Size Estimation)



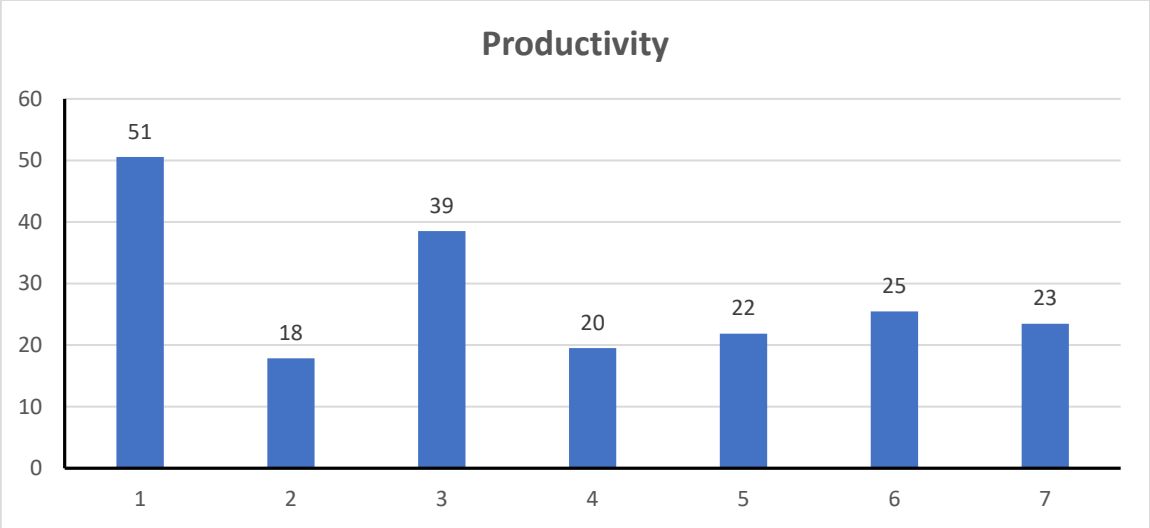
เนื่องจากการประมาณขนาดไม่ได้นับรวมเข้าไปในส่วนของ Base, Added และ Reused ทั้งหมด แต่จะมีส่วนของ main function ที่จำนวนบรรทัดของโค้ดก็ส่งผลกระทบต่อขนาด (total size) ของโปรแกรม ทำให้โปรแกรมที่ 4 ที่มีจำนวนบรรทัดใน main function เยอะที่สุดถูกประมาณขนาดผิดพลาดกระโดดไปจากแนวโน้มโดยรวมของทุกโปรแกรม ส่วนโปรแกรมที่ 2 และ 7 มีความยากกว่าโปรแกรมอื่น ๆ ค่อนข้างมากจึงทำให้การทำนายขนาดจะผิดพลาดมากกว่าโปรแกรมอื่น แต่สุดท้ายแล้วแนวโน้มความแม่นยำในการทำนายขนาดของโปรแกรมก็มีความแม่นยำมากขึ้นเป็นที่น่าพอใจ

2.2 แนวโน้มความแม่นยำในการทำนายเวลาในการพัฒนา (Time Estimation)



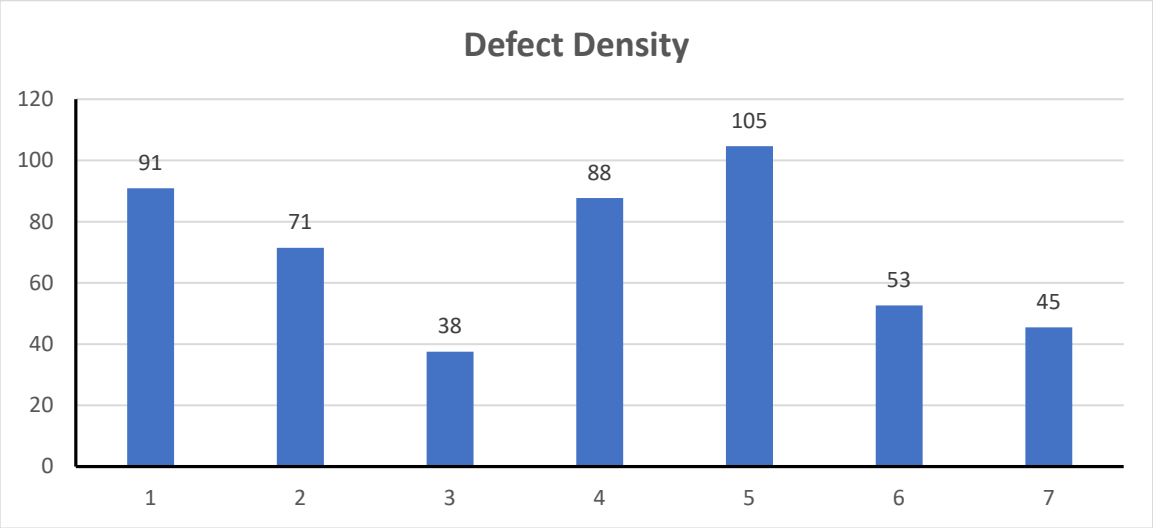
จากกราฟข้างต้นจะสังเกตเห็นได้ว่าในช่วงแรกนั้นมีการทำนายเวลาในการพัฒนาโปรแกรมที่สูงมาก โดยไม่นับโปรแกรมที่ 1 เพราะมีความง่ายในการพัฒนามาก โดยรวมแล้วแนวโน้มในการทำนายเวลามีประสิทธิภาพสูงมากขึ้นเรื่อย ๆ แม้จะมีความยากในการพัฒนาโปรแกรมที่มากขึ้น และอาจเป็นผลเนื่องจากการใช้งาน PROBE A ที่มีการใช้ correlation ของข้อมูลย้อนหลังในการคำนวณเวลาก็สังเกตเห็นได้ว่าในโปรแกรมหลัง ๆ นั้นมีแนวโน้มที่ความถูกต้องของการทำนายเวลานั้นลู่เข้าสู่ 0 ได้ในอนาคตหากมีการปฏิบัติและกระบวนการดังเช่นเดิมต่อไป

3.1 แนวโน้มของผลิตภาพ (Productivity)



ค่าของผลิตภาพในช่วงแรกมีความแปรปรวนสูง แต่ในช่วงหลังค่อนข้างคงที่ คาดว่าผลิตภาพของผมน่าจะอยู่ที่ช่วงประมาณ 20 – 25 LOC/Hour ซึ่งคงไม่เร็วไปกว่านี้เพราะการให้ความสำคัญของการออกแบบและการรีวิวส่วนต่าง ๆ ในกระบวนการมีผลให้ประสิทธิภาพของการพัฒนาโปรแกรมดีขึ้นในแง่ของการไม่สร้างข้อผิดพลาด

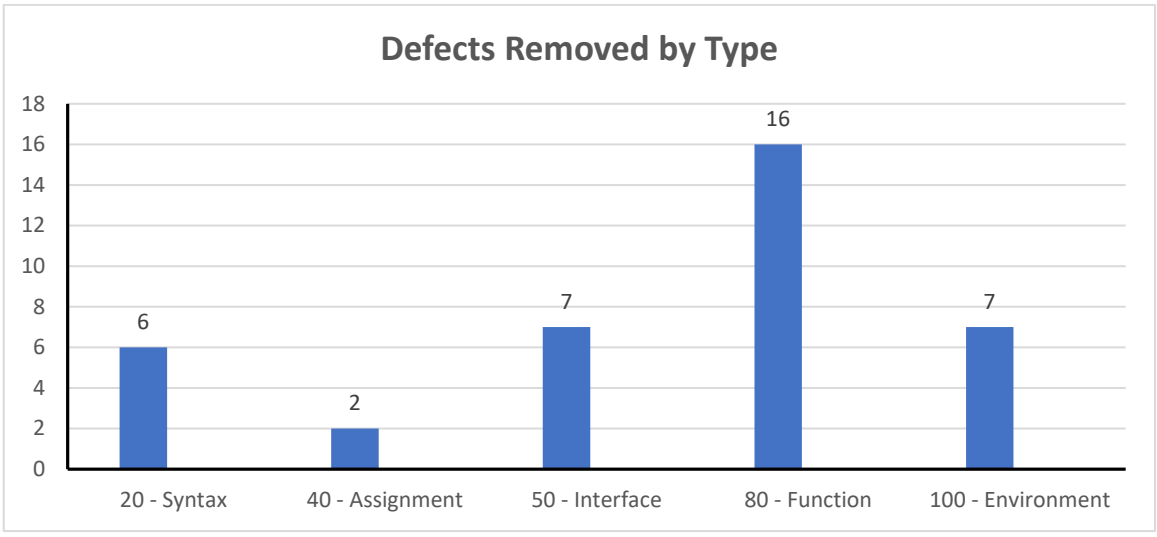
3.2 ความหนาแน่นของข้อผิดพลาด (Defect Density)



จำนวนข้อผิดพลาดที่ถูกสร้างขึ้นในการพัฒนาแต่ละโปรแกรมนั้นไม่ค่อยมีแนวโน้มไปในทางเดียวกัน อาจเนื่องด้วยความยากง่ายหรือส่วนต่าง ๆ ในการพัฒนาของแต่ละโปรแกรมนั้นมีความแตกต่างกัน แต่ก็ถือว่าน่าจะมีความหนาแน่นของข้อผิดพลาดลดลงโดยเฉพาะข้อผิดพลาดเดิม ๆ ที่เคยพบเจอมาก่อน

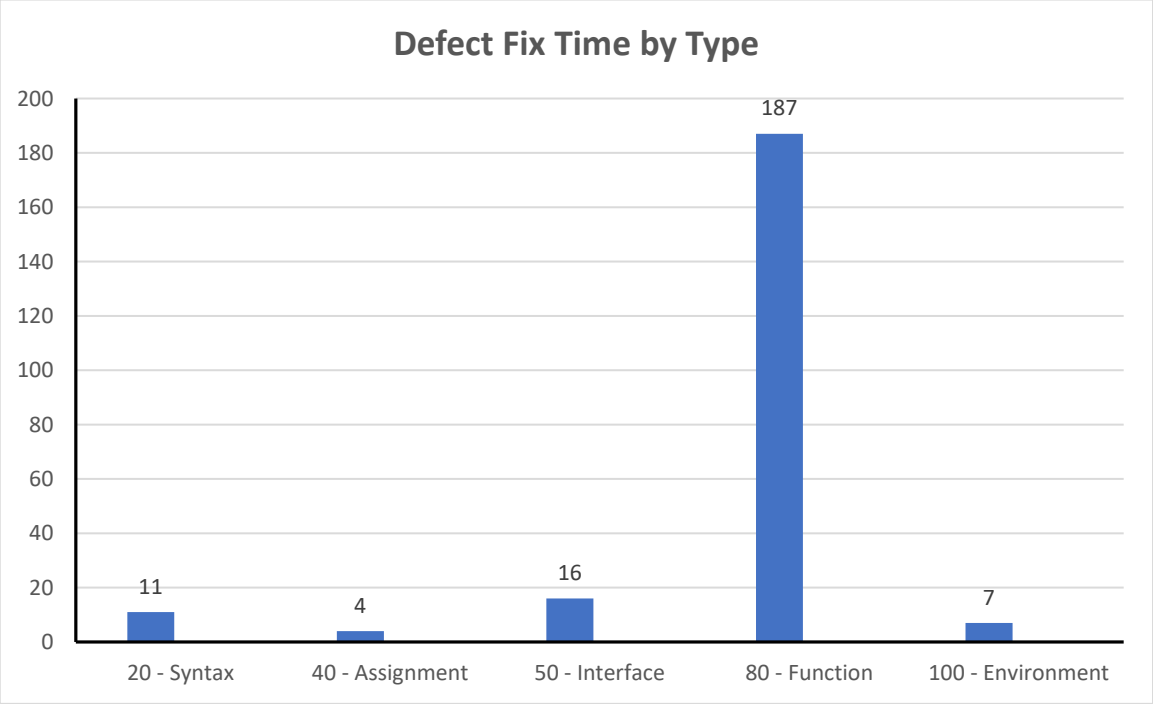
4. แนวโน้มการเกิดและการแก้ไขข้อผิดพลาด (Defects)

4.1 การเกิดข้อผิดพลาดแยกตามชนิดของข้อผิดพลาด



ข้อผิดพลาดชนิดที่เป็น 80 – Function นั้นมีจำนวนมากที่สุดอย่างเห็นได้ชัดเนื่องจากส่วนใหญ่แล้วการพัฒนาโปรแกรมขึ้นมามีความผิดพลาดอยู่ที่ไม่สามารถทำให้การทำงานของโปรแกรมให้ผลลัพธ์ออกมาตามที่คาดหวังไว้ได้ (expected result กับ actual result ไม่ตรงกัน) ตามมาด้วยข้อผิดพลาดชนิด 50 – Interface คือการแสดง format ของข้อความ output ซึ่งเกิดบ่อยที่สุดและแก้ไขยังไม่ได้ และ 100 – Environment ซึ่งเป็นข้อผิดพลาดจากการออกแบบยังไม่รอบคอบเพียงพอ

4.2 เวลาของการแก้ไขข้อผิดพลาดแต่ละชนิด



การแก้ไขข้อผิดพลาดชนิด 80 – Function ล้วนจะเกี่ยวข้องกับลอจิกของโค้ดเป็นหลัก ดังนั้นการพบเจอข้อผิดพลาดชนิดนี้โดยไม่สามารถป้องกันได้นั้นก็จะมีการใช้เวลาในการแก้ไขข้อผิดพลาดนี้ใน UT phase ที่นานมาก ดังนั้นพยายามที่จะออกแบบ (design) ก่อนที่จะเขียนโค้ดจริง ๆ เพื่อลดทั้งเวลาและข้อผิดพลาดที่จะเกิดขึ้นในชนิด 80 – Function

5. แนวทางการพัฒนากระบวนการในอนาคต

- พยายามลดข้อผิดพลาดชนิดของ 80 – Function ต่อไปเพราะเป็นชนิดข้อผิดพลาดที่มีจำนวนการเกิดมากที่สุด และใช้เวลาในการแก้ไขโดยรวมมากที่สุด
- จะทำ Review Checklist เฉพาะกับข้อผิดพลาดที่เกิดขึ้นบ่อยที่สุดและสม่ำเสมอแก้ไขไม่หายโดยง่าย
- จะใช้การออกแบบ (design) โปรแกรมก่อนที่จะทำการเขียนโปรแกรม (coding) ในทุกโปรแกรม
- การทำ design และ design review อาจจะใช้เวลานานต่างกันขึ้นกับขนาดของโปรแกรมที่จะพัฒนา หากโปรแกรมมีขนาดใหญ่จะใช้เวลาในส่วน design มากตาม
- การทำ design จะเน้นไปด้าน Logic Specification เป็นหลัก ในส่วนของ Operational Specification จะไม่ออกแบบอีกต่อไป
- การทำ code review มีประโยชน์มาก จะพึงระลึกไว้เสมอและปฏิบัติให้เป็นนิสัย คือก่อนทำการ compile ในทุกโปรแกรมที่พัฒนาจะทำรีวิวโค้ดอย่างสม่ำเสมอ
- การพัฒนากระบวนการของบุคคล (PSP) จะนำไปสู่การพัฒนากระบวนในระดับของทีม (TSP) และระดับขององค์กร (CMMI) ต่อไป
- จะยึดหลักปฏิบัติตามแบบแผนของตนเองอย่างสม่ำเสมอในการพัฒนาโปรแกรมต่อไป

6. ความเปลี่ยนแปลงที่เกิดขึ้นกับตัวนักศึกษาหลังสิ้นสุดหลักสูตร PSP

- ได้ฝึกการเขียนโปรแกรมด้วยภาษา Java เนื่องจากไม่เคยเขียนมาก่อน
- ได้ทักษะในการออกแบบ (design) และเล็งเห็นความสำคัญของการออกแบบก่อนที่จะทำงานจริง
- ได้ทราบความสามารถในการสร้างข้อผิดพลาดของตัวเอง โดยเฉพาะชนิดของผิดพลาดแต่ละชนิดที่เกิดขึ้น
- ได้ทราบว่าโดยปกติแล้วพฤติกรรมในการพัฒนาโปรแกรมใด ๆ ขึ้นมานั้นใช้เวลาในแต่ละขั้นตอนเป็นอย่างไร
- มีมาตรฐานในการเขียนโค้ด (coding standard) ของตนเองที่ชัดเจนและยึดตามมาตรฐานนี้ในการเขียนโค้ด
- สามารถประเมินเวลาที่ใช้ในการพัฒนาโปรแกรมใหม่ ๆ ได้อย่างแม่นยำมากขึ้นโดยอิงกับข้อมูลที่ถูกเก็บสะสมมาจากการทำ PSP ในช่วงที่ผ่านมา ไม่เหมือนกับตอนก่อนเรียน PSP ซึ่งประเมินเวลาของตัวเองไม่ได้เลยสักขั้นตอน
- สามารถประเมินความสามารถในการทำงานของเราได้พอสมควร
- ไม่เปรียบเทียบกระบวนการส่วนบุคคลของตนเองและผู้อื่นเพราะทุกคนมีความแตกต่างกันอย่างสิ้นเชิง แต่ก็สามารถศึกษาดูกระบวนการของผู้อื่นและทดลองกับตัวเองได้ หากมีประโยชน์ก็นำมาปรับใช้กับของตัวเอง
- กระบวนการในการพัฒนาโปรแกรมมีคุณภาพมากขึ้น!

7. แนวทางการประยุกต์ใช้ PSP กับกิจกรรมอื่น ๆ ในชีวิตประจำวัน

- การจดบันทึกจะทำให้เรามีข้อมูลในอดีตเก็บไว้เพื่อใช้วิเคราะห์แนวโน้มของอนาคตได้
- การป้องกันไม่ให้เกิดข้อผิดพลาดดีกว่าการตามแก้ไขข้อผิดพลาดที่เกิดขึ้นไปแล้ว (ทำให้ดีตั้งแต่เริ่มต้นเลย)
- วางแผน (design or plan) ก่อนที่จะทำงานใด ๆ อย่างเสมอ
- หลังจากทำงานเสร็จ สามารถวิเคราะห์และประเมินกิจกรรมทั้งหมดที่ผ่านมาของงานได้ ซึ่งจะนำไปสู่กระบวนการพัฒนาหรือปรับปรุงกระบวนของเราเอง โดยเริ่มจากการวิเคราะห์กระบวนการที่ใช้ไปว่ามีประสิทธิภาพเพียงใด และหากสามารถปรับปรุงได้ก็เสนอกระบวนการที่น่าจะมีประสิทธิภาพดีขึ้นกว่าเดิม เช่นเดียวกับหลักการของ PIP ใน PSP
- เสริมสร้างลักษณะนิสัยของความสม่ำเสมอในตัวบุคคล แต่หากสามารถปรับปรุงพัฒนาลักษณะนิสัยในการทำสิ่งใดก็ตามให้ดีขึ้นได้ก็ต้องพัฒนา
- ทุกอย่างสามารถพัฒนาได้ ต้องเริ่มที่ตัวเราเอง อยู่ที่เลือกว่าจะละเลยหรือศึกษา วิเคราะห์ และปรับปรุงตัวเอง