การศึกษาโมเดล SPII–XP เพื่อเสริมสร้างทักษะการพัฒนาซอฟต์แวร์สำหรับผู้เรียนเฉพาะบุคคล

AN EMPIRICAL STUDY FOR SPII–XP MODEL AS A SOFTWARE DEVELOPMENT SKILL IMPROVEMENT FOR INDIVIDUAL LEARNER.

พรสิริ ชาติปรีชา[1]
Pornsiri Chatpreecha[1]

บทคัดย่อ

งานวิจัยนี้นำเสนอการทดลองการนำรูปแบบใหม่ “Software Process for improvement with Extreme Programming: (SPII-XP)” ของกระบวนการพัฒนาซอฟต์แวร์ส่วนบุคคลไปทดลองใช้กับการปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ขนาดเล็กของนักศึกษาระดับปริญญาตรี จุดเด่นของ SPII-XP คือ การนำเอาข้อดีและประโยชน์ของกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล (Personal Software Process (PSP)) และวิธีการพัฒนาซอฟต์แวร์ของ Extreme Programming (XP) ของสองรูปแบบมาผสมผสานกันเพื่อให้เกิดการสร้างรูปแบบใหม่ของการพัฒนากระบวนการซอฟต์แวร์เฉพาะบุคคล โดยรูปแบบใหม่ของ SPII-XP จะเน้นการวัดกระบวนการส่วนบุคคล กระบวนการจัดการและการประมาณการซอฟต์แวร์ และกระบวนการจัดการด้านคุณภาพ เป้าหมายคือ การปรับให้เหมาะกับการพัฒนาซอฟต์แวร์ส่วนบุคคลซึ่งจะสามารถนำไปปรับใช้กับกระบวนการพัฒนาทีมซอฟต์แวร์ขนาดใหญ่ได้ในภายหลังอย่างมีประสิทธิภาพมากขึ้น

คำสำคัญ: กระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล เอ็กซ์ตรีมโปรแกรมมิ่ง การพัฒนากระบวนการ ซอฟต์แวร์เฉพาะบุคคลโดยรูปแบบใหม่ของเอ็กซ์ตรีมโปรแกรมมิ่ง

Abstract

This article introduces a new experimental approach, the “Software Process for improvement with Extreme Programming (SPII-XP)”, on personal software development process as a test process to enhance the software improvement process for the undergraduate student. The strong point of SPII-XP is to combine the advantage of both the “Personal Software Process (PSP)” and the “Extreme Programming (XP)” process to create a new format for personal software development for individual. This new approach, the “Software Process for improvement with Extreme Programming (SPII-XP)”, will emphasis on: Personal Measurement, Personal Management and Estimating including the Personal Quality Management. The goal is to come up with appropriate personal software development process which can later on adapt to the team development of large scaled software.

[1] อาจารย์ประจำสาขาเทคโนโลยีสารสนเทศ คณะวิศวกรรมศาสตร์และเทคโนโลยี, full time lecturer in Faculty of Engineering and Technology, Panyapiwat Institute of Management, E-mail: pornsiricha@pim.ac.th

ผ่านการรับรองคุณภาพจาก TCI (กลุ่มที่ 1) สาขามนุษยศาสตร์และสังคมศาสตร์

Keywords: Personal Software Process, Extreme Programming, Software Process Improvement for Individual with Extreme Programming

บทนำ

ปัจจุบันองค์กรต่างๆ เน้นมาให้ความสำคัญกับเรื่องของกระบวนการพัฒนาซอฟต์แวร์กันมากขึ้น เนื่องจากเป็นแนวทางที่ช่วยลดความเสี่ยงและเพิ่มคุณภาพของการพัฒนาซอฟต์แวร์ โดยเฉพาะการพัฒนากระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล โดยเฉพาะวิธีการที่ได้รับการยอมรับและถูกนำมาใช้กับกระบวนการพัฒนาซอฟต์แวร์ส่วนบุคคล คือ PSP (Personal Software Process) และ XP (Extreme Programming)

ซึ่งวิธีการของ PSP [1] จะเน้นให้ความสำคัญทางด้านความรับผิดชอบต่อกระบวนการพัฒนาซอฟต์แวร์ของตนเอง การวัดผล การวิเคราะห์การทำงานและการนำสิ่งที่เรียนรู้มาปรับใช้กับวิธีการทำงานของตนเอง ส่วนวิธีการของ XP จะเน้นให้ความสำคัญสำหรับการพัฒนาซอฟต์แวร์ที่ทำให้เกิดความรวดเร็ว ตรงต่อเวลา และการใช้เทคนิควิธีต่างๆ ที่ก่อให้เกิดคุณภาพสำหรับการพัฒนาซอฟต์แวร์ให้ตรงกับความต้องการของผู้ใช้

ด้วยเหตุนี้ ผู้วิจัยจึงเกิดแนวความคิดที่จะนำเอาวิธีการ PSP และ XP ซึ่งเป็นกระบวนการซอฟต์แวร์ที่ช่วยพัฒนาขีดความสามารถของนักพัฒนาซอฟต์แวร์แต่ละคนหรือทีมซอฟต์แวร์ขนาดเล็ก โดยการนำเอาข้อดีและประโยชน์ของ PSP และ XP มาทำการปรับปรุงจนเกิดเป็นกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล คือ SPII-XP (Software Process Improvement for Individual with Extreme Programming)

วัตถุประสงค์ของการวิจัย

ศึกษาถึงผลของการนำรูปแบบใหม่ระหว่างการผสมผสานของกระบวนการทางซอฟต์แวร์เฉพาะบุคคล (PSP: Personal Software Process) และการพัฒนาซอฟต์แวร์ในรูปแบบ Extreme Programming ไปใช้งานได้จริง

ทบทวนวรรณกรรม

สำหรับหลักการและที่มาของ PSP และ XP Personal Software Process (PSP) คือ กระบวนการพัฒนาซอฟต์แวร์ส่วนบุคคลที่เน้นการวางแผนและคุณภาพของระบบ คือ มีการตรวจสอบการออกแบบและการตรวจสอบการเขียนโค้ดแต่ PSP มีข้อจำกัด คือ ไม่มีการจัดเก็บความต้องการของผู้ใช้การวิเคราะห์ระบบ การทดสอบโปรแกรมการบำรุงรักษา การจัดการกับความเสี่ยงที่เกิดขึ้น และมีการทำเอกสารค่อนข้างมาก รวมไปถึงจะต้องมีการเก็บข้อมูลเพื่อมาทำการวัดผลจึงเกิดความล่าช้ากับกระบวนการพัฒนาซอฟต์แวร์ [2]

ในลำดับขั้นของกระบวนการพัฒนาซอฟต์แวร์ส่วนบุคคลจะแบ่งออกเป็น 4 กระบวนการดังรูปที่ 1

PSP 0 คือ จะใช้กระบวนการพัฒนาที่นักพัฒนาโปรแกรมใช้อยู่ในปัจจุบัน นักพัฒนาโปรแกรมจะเริ่มเรียนรู้ฟอร์มและสคริปของพีเอสพี (PSP) และจับเวลาที่ใช้ในการพัฒนาโปรแกรม และวัดจำนวนข้อผิดพลาด (defect) ที่ตรวจพบ

PSP 0.1 คือ มีการใช้มาตรฐานของการเขียนโปรแกรม (Coding Standard) การวัดขนาดของโปรแกรมและพีไอพีฟอร์ม (PIP: Process Improvement Proposal form) ซึ่งจะรวบรวมปัญหา และแนวคิดที่จะใช้ในการพัฒนาโปรแกรมตลอดจนข้อเสนอการปรับปรุงกระบวนการ

PSP 1 คือ มีการวางแผนสำหรับกระบวนการพัฒนาซึ่งการวางแผนนี้จะรวมถึงการประมาณการขนาดโปรแกรมและทรัพยากรที่ใช้ และบันทึกการทดสอบโปรแกรม (Test report) ในพีเอสพี (PSP) การประมาณขนาดและเวลาจะใช้ในการพัฒนาโปรแกรมจะใช้วิธีการของโพรบ (PROBE: Proxy Base Estimate) [3]

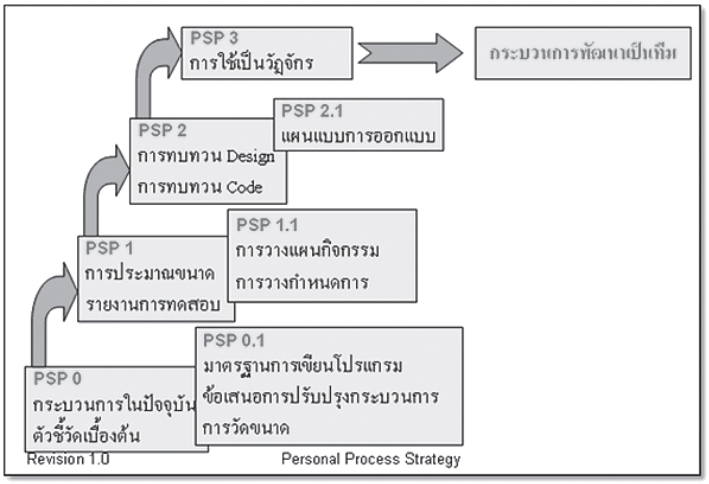
PSP 1.1 คือ มีการวางแผนและกำหนดการงานหรือกิจกรรมต่างๆ ที่ต้องทำสำหรับของการพัฒนาซอฟต์แวร์

PSP 2 คือ มีการจัดการข้อผิดพลาด (defect) เพื่อตรวจสอบและป้องกันข้อผิดพลาด (defect) โดยเพิ่ม

การตรวจสอบโค้ด (Code review) และการทบทวนการออกแบบ (Design review)

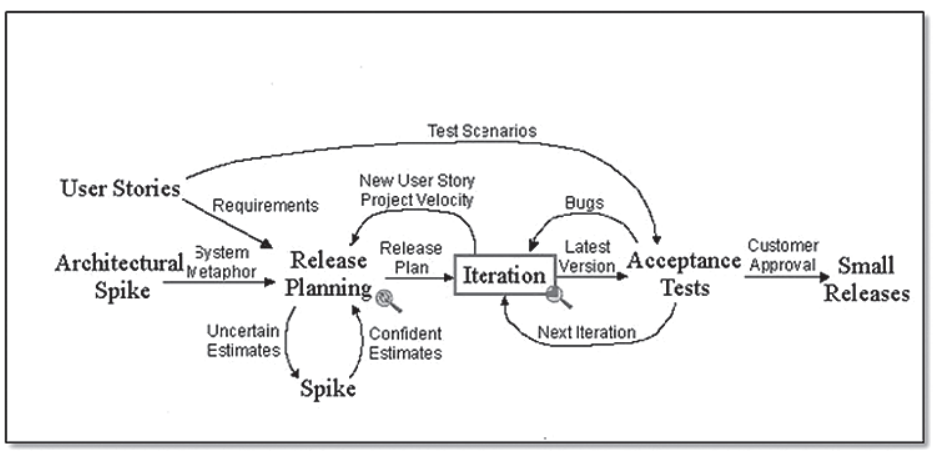
ส่วน PSP 2.1 คือ มีการออกแบบในกระบวนการพัฒนา

PSP 3 คือ การใช้เป็นวัฏจักร ซึ่งมีการปรับกระบวนการสำหรับการพัฒนาโปรแกรมเพื่อรองรับการพัฒนาโปรแกรมขนาดใหญ่ โดยแบ่งโปรแกรมออกเป็นส่วนๆ แล้วพัฒนาแต่ละส่วนของโปรแกรมตาม PSP 2 การพัฒนาแบบนี้สามารถพัฒนาโปรแกรมที่มีขนาดหลายพันบรรทัด



รูปที่ 1 PSP Structure [1]

ส่วน Extreme Programming (XP) คือ กระบวนการพัฒนาซอฟต์แวร์สำหรับทีมขนาดเล็กโดยพยายามให้นักพัฒนามุ่งเน้นไปที่การพัฒนาซอฟต์แวร์ ซึ่งมีการเขียนกรณีทดสอบก่อนการเขียนโปรแกรมทำให้มีการวางแผนสำหรับการทดสอบระบบในขณะเริ่มต้นการพัฒนาระบบ การทดสอบจะถูกพัฒนาคู่ขนานไปกับการวิเคราะห์ความต้องการของผู้ใช้ระบบทั้งหมดตามความต้องการของผู้ใช้ [4]



รูปที่ 2 XP Practice [6]

จากรูปที่ 2 คือ แนวทางปฏิบัติของ XP ประกอบด้วย 12 แนวทางปฏิบัติดังนี้ คือ

1. เกมส์การวางแผน (Planning Game) คือ ลูกค้าจะเลือกขอบเขตและเวลาของแต่ละรอบจากการประมาณโดยนักพัฒนาโปรแกรม จะต้องพัฒนาโปรแกรมตามความต้องการในแต่ละรอบของการปรับปรุงเปลี่ยนแปลง

2. การส่งมอบงานขนาดเล็ก (Small releases) คือ ระบบจะถูกแบ่งออกเป็นส่วนๆ และเมื่อผู้ใช้เห็นว่าระบบมีความสามารถมากพอที่จะนำมาใช้งานได้ ก็จะนำระบบนั้นมาเริ่มใช้งานโดยไม่ต้องรอให้ระบบเสร็จสมบูรณ์ และพัฒนาใหม่ในทุกวันหรือในทุกเดือน

3. เมตาฟอร์ (Metaphor) คือ การแสดงความเข้าใจโดยการอธิบายการใช้ระบบเมตาฟอร์จะถูกใช้โดยลูกค้าและนักพัฒนาโปรแกรมเท่านั้น

4. การออกแบบเน้นความง่าย (Simple design) การออกแบบจะต้องถูกทดสอบทั้งหมด โดยจะไม่มีโค้ดที่ซ้ำกันและมีจำนวนคลาสและเมธอดน้อยที่สุดเท่าที่เป็นไปได้

5. การทดสอบซอฟต์แวร์ (Test) นักพัฒนาจะต้องเขียนกรณีทดสอบอยู่ตลอดเวลา และการทดสอบทั้งหมดจะต้องถูกต้อง ลูกค้าจะกำหนดฟังก์ชันนอลเทสต์ (functional test) สำหรับแต่ละสตอรีในทุกไอเทอร์เรชัน ฟังก์ชันนอลเทสต์นี้จะต้องใช้ทดสอบระบบเช่นกัน ในบางครั้งอาจจะมีการส่งระบบที่ไม่ผ่านการทดสอบทั้งหมดได้เช่นกัน ด้วยเหตุผลทางธุรกิจ

6. การทำงานของนักพัฒนาที่เป็นคู่ (Pair Programming) คือ นักพัฒนาโปรแกรมจะทำการพัฒนาโปรแกรมเป็นคู่โดยใช้เครื่องมือทุกอย่างร่วมกันไม่ว่าจะเป็นคอมพิวเตอร์ ปากกา กระดาษ

7. การบูรณาการอย่างต่อเนื่อง (Continuous integration) คือ โค้ดที่นักพัฒนาโปรแกรมเขียนขึ้นใหม่จะต้องถูกนำไปรวมในระบบอย่างเร็วที่สุด ระบบต้องถูกต้องคอมไพล์ใหม่ทั้งหมดและต้องผ่านการทดสอบไม่เช่นนั้นโค้ดส่วนนั้นจะต้องถูกตัดออกไป

8. คอเล็กทีฟโอนเนอร์ชิพ (Collective ownership) คือ นักพัฒนาโปรแกรมทุกคนมีสิทธิที่จะแก้ไขโปรแกรมส่วนใดส่วนหนึ่งได้ ตราบเท่าที่การแก้ไขนั้นผ่านการทดสอบทั้งหมด

9. ออนไซด์คัสตอมเมอร์ (On-site customer) คือ ลูกค้าจะต้องนั่งทำงานอยู่กับนักพัฒนาโปรแกรมตลอดเวลา

10. การทำงานสัปดาห์ละ 40 ชั่วโมง (40-hour weeks) คือ การที่นักพัฒนาโปรแกรมทำงานหนักมากเกินไป ทำให้ขาดประสิทธิภาพในการทำงาน ดังนั้นจะไม่ใช้เวลามากเกินไปในการทำงาน

11. รีแฟกเตอร์ริง (Refactoring) หมายถึง การออกแบบระบบโดยการเปลี่ยนระบบเดิม โดยที่ระบบใหม่ต้องผ่านการทดสอบทั้งหมด

12. การทำงานในสภาวะเปิด (Open workspace) นักพัฒนาโปรแกรมทุกคนในทีมจะทำงานอยู่ในห้องเดียว

เนื่องจากบางแนวทางปฏิบัติของ XP ยากต่อการนำไปปฏิบัติ เช่น การพัฒนาโปรแกรมเป็นคู่ (Pair Programming) การเขียนโปรแกรมเท่าที่จำเป็น นอกจากนี้ XP ไม่มีตัววัดผลสำหรับกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคลและไม่มีเอกสารรองรับสำหรับกระบวนการพัฒนาซอฟต์แวร์เพราะใช้ source code เป็นเอกสารดังนั้นนักพัฒนาจึงต้องเขียนโค้ดที่อ่านและเข้าใจง่าย สามารถสรุปข้อดีข้อเสียของได้ [3] ดังตารางที่1

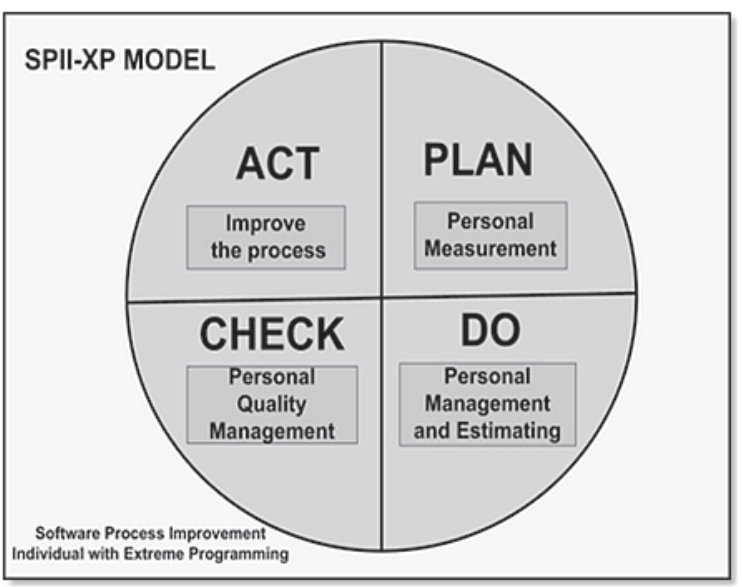
ตารางที่ 1 เปรียบเทียบกระบวนการพัฒนาซอฟต์แวร์ส่วนบุคคลของ PSP XP และ SPII-XP

ขั้นการพัฒนา (PDCA)	PSP	XP	SPII-XP
ขั้นตอนการวางแผน (Plan)	PSP 1.0 จะใช้การวางแผนและเริ่มจากการออกแบบในขั้นต้นแล้วจึงใช้วิธี PROBE ในการประมาณขนาดและเวลาที่ใช้ ถ้าโปรแกรมมีขนาดใหญ่ก็จะแบ่งโปรแกรมเป็นส่วนๆ เพื่อทำการพัฒนา PSP 1.1 จะใช้การจัดแผนงานและตารางเวลาสำหรับการพัฒนาโปรแกรมและจัดทำเป็นเอกสาร	XP ใช้ Planning Game โดยให้ลูกค้าเรียง user story ตามลำดับความสำคัญ แล้วเลือก user story มาตามลำดับโดยแบ่งเป็นรอบ รอบละประมาณ 3 สัปดาห์ การประมาณขนาดของ user story ดูจาก user story ก่อนหน้าที่ใกล้เคียงกันแต่ไม่มีเอกสาร	SPII-XP 2.2 นำรูปแบบของเอกสารใน PSP 1.1 มาใช้แต่ปรับปรุงในส่วนของเอกสารให้สามารถใช้ได้กับการวางแผนพัฒนาโปรแกรมที่มีขนาดเล็ก ส่วนใน XP ได้นำรูปแบบของเอกสารมาปรับใช้เพื่อกำหนดการวางแผน SPII-XP 2.2 นำรูปแบบของเอกสารใน PSP 1.1 มาใช้ ส่วนใน XP ได้นำหลักการของการเขียน Test Case มาใช้ SPII-XP 2.1 เข้ามาปรับปรุงวิธีการประมาณขนาดของ PSP 1.0 โดยใช้ Usecasepoints ส่วนใน XP ซึ่ง SPII-XP 2.1 ได้นำวิธีการประมาณขนาดมาใช้รวมไปถึงรูปแบบเอกสาร
ขั้นตอนการปฏิบัติ (Do)	PSP 0 จะให้นักพัฒนามีการเรียนรู้ฟอร์มและสคริปต์ของ PSP และนักพัฒนาโปรแกรมจะจับเวลาที่ใช้ในการพัฒนาโปรแกรม และบันทึกจำนวนข้อผิดพลาด (defect) ที่ตรวจพบ ซึ่งในส่วนของการพัฒนาโปรแกรมแยกเป็น 3 ขั้นตอน และใน PSP ระดับที่ 3 จะเตรียมกรณีทดสอบก่อนเขียนโปรแกรม PSP 0.1 ใช้มาตรฐานการเขียนโปรแกรมการวัดขนาดของโปรแกรม และข้อเสนอการปรับปรุงกระบวนการ (PIP Form) ซึ่ง PIP ฟอร์มจะรวบรวมปัญหา และแนวคิดที่จะพัฒนากระบวนการที่ใช้ PSP 2.1 มีการเพิ่มการออกแบบแต่ไม่ได้กำหนดวิธีการออกแบบ มีเพียงตัวอย่างการออกแบบโดยใช้เซตของ template และ logic notation	XP มีการส่งมอบงานขนาดเล็ก (Small releases) ระบบจะถูกแบ่งออกเป็นส่วนๆ และเมื่อพัฒนาระบบจนผู้ใช้เห็นว่าระบบมีความสามารถมากพอที่จะนำมาใช้งานได้ ก็จะนำระบบนั้นมาเริ่มใช้งานโดยไม่ต้องรอให้ระบบเสร็จสมบูรณ์ XP มีใช้มาตรฐานการเขียนโปรแกรมเช่นเดียวกับ PSP โดยที่การวัดขนาดของ XP จะใช้การประมาณขนาดซอฟต์แวร์จากการประมาณเวลาอย่างหยาบๆ เป็นสัปดาห์ XP จะใช้การออกแบบเน้นความง่ายโดยใช้เทคนิค explicit design เพื่อทำความเข้าใจ ส่วนใหญ่นักพัฒนาโปรแกรมที่ใช้ UML หรือ CRC card ในการออกแบบแต่เป็นเพียงแค่การออกแบบแค่โครงสร้างเท่านั้น	นำ PSP 0 มาใช้เพื่อเปรียบเทียบข้อมูล 2 อย่าง คือ เวลาและข้อผิดพลาดที่เกิดขึ้น SPII-XP 1.2 นำรูปแบบของเอกสารใน PSP 0.1 มาปรับใช้ใน XP SPII-XP 1.2 ได้นำวิธีการวัดขนาดของ SEI มาปรับใช้ใน PSP 0.1 ส่วน XP ได้นำวิธีการประมาณขนาดและเอกสารไปใช้
ขั้นตอนการตรวจสอบ (Check)	PSP 2 มีการจัดการข้อผิดพลาด (defect) เพื่อตรวจสอบและป้องกันข้อผิดพลาด (defect) โดยเพิ่มการตรวจสอบโค้ดและการตรวจสอบการออกแบบ	XP การทดสอบการเขียนโค้ดโดยการใช้เทคนิคที่ชื่อว่า Test–first design โดยจะเขียนกรณีทดสอบครั้งละ 1 กรณี และเขียนโปรแกรม	SPII-XP 3.1 นำหลักการของ PSP มาใช้แต่จะกำหนดวิธีการตรวจสอบคุณภาพขั้นพื้นฐาน ส่วนใน XP ไม่นำ Pair Programming มาใช้แต่นำวิธีการใน SPII-XP 3.1 มาใช้แทน

ขั้นการพัฒนา (PDCA)	PSP	XP	SPII-XP
		ให้ผ่านกรณีทดสอบนั้น โดยอาจมีการ refactoring เพื่อปรับโครงสร้างของโปรแกรมให้ง่ายต่อการแก้ไขและเขียนกรณีทดสอบถัดไปจนกว่าจะเขียนโปรแกรมเสร็จ XP ใช้เทคนิคของการ refactoring เพื่อแก้ไขการออกแบบ	SPII-XP3.1 นำหลักการของ PSP มาใช้แต่จะกำหนดวิธีการตรวจสอบคุณภาพขั้นพื้นฐาน ส่วนใน XP ได้นำหลักการนี้ไปใช้เพราะจะช่วยให้นักพัฒนาโปรแกรมสามารถที่จะสร้างความเข้าใจสำหรับการอธิบายโปรแกรมที่พัฒนาสร้างขึ้นให้มีประสิทธิภาพมากขึ้น
ขั้นตอนการดำเนินการให้เหมาะสม (Act)	PSP 3 มีการดำเนินการสำหรับการปรับปรุงกระบวนการพัฒนาโปรแกรมให้รองรับการพัฒนาโปรแกรมขนาดใหญ่ โดยแบ่งโปรแกรมออกเป็นส่วนๆ แล้วพัฒนาแต่ละส่วนของโปรแกรมตาม PSP 2 การพัฒนาแบบนี้ทำให้สามารถพัฒนาโปรแกรมที่มีขนาดหลายพันบรรทัดได้	XP ใช้วิธีการ Continuous integration โดยโค้ดที่นักพัฒนาโปรแกรมเขียนขึ้นใหม่จะต้องถูกนำไปรวมในระบบอย่างเร็วที่สุด ระบบจะต้องถูกคอมไพล์ใหม่ทั้งหมดและต้องผ่านการทดสอบ ไม่เช่นนั้นโค้ดส่วนนั้นจะถูกตัดออกไป	SPII-XP 3.2 นำหลักการของ PSP 3 มาใช้แต่ปรับรูปแบบของกระบวนการโดยสร้างรูปแบบตัวอย่างของโปรแกรมที่สามารถนำไปใช้งานได้จริง ส่วนใน XP นำหลักการของการทำ Refactoring มาใช้รวมไปถึงรูปแบบของเอกสาร

SPII-XP MODEL

SPII-XP MODELได้นำหลักการของวงจร PDCA [7] มาประยุกต์ใช้กับกระบวนการพัฒนา ซอฟต์แวร์เฉพาะบุคคลโดยใช้วิธีการของ XP ซึ่งขั้นตอนของ PDCA จะประกอบไปด้วยดังนี้ “การวางแผน” อย่างรอบคอบเพื่อ“การปฏิบัติ” แล้วจึง “ตรวจสอบ” วิธีการปฏิบัติใดมีประสิทธิภาพสูงสุด ก็จัดให้เป็นมาตรฐาน หากไม่สามารถบรรลุเป้าหมายได้ก็ต้องมองหาวิธีการปฏิบัติใหม่หรือใช้ความพยายามให้มากกว่าเดิม ดังรูปที่ 3 [7]



รูปที่ 3 SPII-XP MODEL

จากรูปที่ 1 SPII-XP MODEL จะประกอบไปด้วย 4 ขั้นตอน ดังนี้ คือ

1. ขั้นตอนการวางแผน (Plan) นำมาประยุกต์ใช้กับกระบวนการจัดการและการประมาณการสำหรับกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล

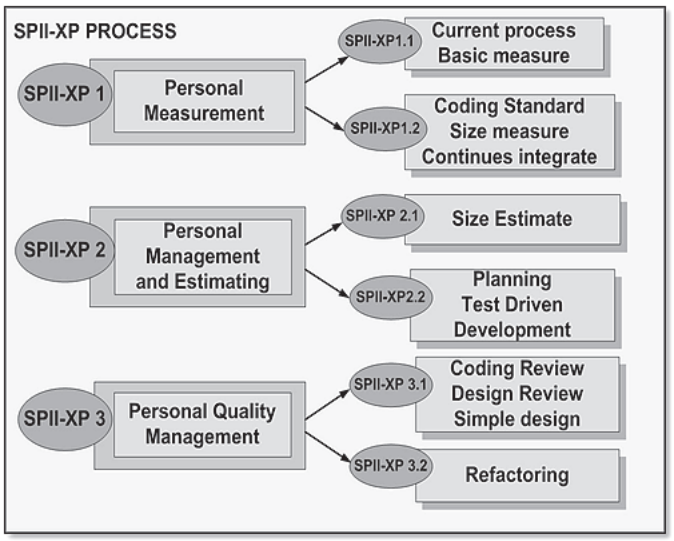
2. ขั้นตอนการปฏิบัติ (Do) ในขั้นนี้จะต้องมีการตรวจสอบระหว่างการปฏิบัติด้วยว่าได้ดำเนินไปในทิศทางที่ตั้งใจไว้หรือไม่ ซึ่งนำมาประยุกต์ใช้กับกระบวนการวัดผลสำหรับนักพัฒนาซอฟต์แวร์เฉพาะบุคคล

3. ขั้นตอนการตรวจสอบ (Check) นำมาประยุกต์ใช้กับกระบวนการจัดการทางด้านคุณภาพขอกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล

4. ขั้นตอนการดำเนินการให้เหมาะสม (Act) นำมาประยุกต์ใช้สำหรับเป็นแนวทางในการนำกระบวนการไปปฏิบัติและปรับปรุงกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคลให้มีคุณภาพมากขึ้น

SPII–XP PROCESS

SPII-XP PROCESS คือ กระบวนการที่นำเอาหลักการของ PSP และแนวทางการปฏิบัติของ XP มาปรับปรุงแล้วได้เป็น 3 กระบวนการ ดังรูปที่ 4



รูปที่ 4 SPII-XP PROCESS

จากรูปที่ 4 SPII-XP PROCESS จะประกอบไปด้วย 3 กระบวนการ คือ SPII-XP 1 คือ การวัดกระบวนการส่วนบุคคลจะแบ่งออกเป็น 2 ขั้นตอน คือ

SPII-XP 1.1 คือ การวัดกระบวนการขั้นพื้นฐาน ซึ่งไม่มีใน XP ผู้เขียนได้นำ PSP 0 มาใช้ โดยไม่มีการปรับปรุง ซึ่งเป็นการสร้างตัวเปรียบเทียบโดยจะเก็บข้อมูลพื้นฐาน 2 อย่าง คือ เวลาและข้อผิดพลาดที่เกิดขึ้น

SPII-XP 1.2 เริ่มนำหลักการของ PSP และ XP มาใช้ คือ มาตรฐานสำหรับการเขียนโค้ด (coding standard) โดยยึดรูปแบบเอกสารของ PSP 0.1 เพราะใน XP ไม่มีรูปแบบของเอกสารส่วนการวัดขนาดของโปรแกรมได้นำหลักการของ SEI [5][6][7] มาใช้ ทั้งใน PSP และ XP เพราะวิธีการวัดขนาดใน PSP เข้าใจยาก ส่วน XP ไม่มีการวัดขนาดของโปรแกรม

ส่วนหลักการของ XP ที่นำมาใช้ในกระบวนการนี้แต่ PSP ไม่ได้กำหนดไว้อย่างชัดเจน คือ การบูรณาการอย่างต่อเนื่อง (Continuous integration) คือ โค้ดที่นักพัฒนาเขียนขึ้นใหม่จะต้องถูกนำไปรวมในระบบอย่างเร็วที่สุด ระบบจะต้องถูกคอมไพล์ใหม่ทั้งหมดและต้องผ่านการทดสอบ ไม่เช่นนั้นโค้ดส่วนนั้นจะถูกตัดออกไปจากระบบ

SPII-XP 2 คือ กระบวนการจัดการสำหรับการวางแผนและการประมาณการส่วนบุคคล โดยแบ่งออกเป็น 2 ขั้นตอนดังนี้ คือ

SPII-XP 2.1 คือ นำเพียงหลักการของ PSP 1.0 มาใช้ แต่ไม่นำวิธีการประมาณขนาดโปรแกรมของ PSP 1.0 มาใช้เพราะวิธีการของ PROBE เป็นวิธีการที่ยุ่งยาก ส่วน XP ใช้การประมาณขนาดจาก User story โดยเปรียบเทียบจาก User story ที่คล้ายกัน จึงได้เพียงค่าหยาบๆ จากการประมาณเท่านั้น ดังนั้น ผู้วิจัยจึงได้นำเอาวิธีการของ Use Case Point มาใช้สำหรับการประมาณขนาดซึ่งเป็นวิธีการที่เข้าใจง่ายและสามารถนำไปปรับใช้ได้กับทุกเครื่องมือที่นำมาใช้ในการพัฒนาโปรแกรม

SPII-XP 2.2 คือ นำหลักการของ PSP และ XP มาใช้ คือ การวางแผน ซึ่งใน XP จะใช้วิธีการของเกมวางแผน (Planning game) โดยให้ลูกค้าเลือกขอบเขต และเวลาของแต่ละรอบจากการประมาณโดยนักพัฒนา แต่ไม่มีการจัดทำเป็นเอกสารสำหรับการพัฒนา ดังนั้นผู้วิจัยจึงนำเอารูปแบบของเอกสารใน PSP 1.1 มาปรับใช้ให้เหมาะสมกับการวางแผนและการจัดแผนการดำเนินงาน ส่วนการทดสอบโปรแกรมได้นำหลักการของ XP มาใช้ คือ การเขียนกรณีทดสอบก่อนการเขียนโปรแกรม และทำการทดสอบโปรแกรมทั้งหมดจนถูกต้องเพื่อลดข้อผิดพลาดที่เกิดจากการพัฒนาโปรแกรม

SPII-XP 3 คือ กระบวนการจัดการทางด้านคุณภาพส่วนบุคคล โดยมี 2 ขั้นตอน คือ

SPII-XP 3.1 คือ การตรวจสอบคุณภาพขั้นพื้นฐาน โดยแยกออกเป็น 2 ส่วน คือ Coding Review และ Design Review โดยปฏิบัติตามหลักการของ PSP ส่วน XP นำหลักการของ Simple design มาใช้สำหรับการออกแบบเพื่อให้นักพัฒนาโปรแกรมสามารถนำหลักการนี้ไปใช้ได้เหมาะสมกับการพัฒนาโปรแกรมให้มีคุณภาพมากขึ้น

SPII-XP 3.2 คือ นำเอาหลักการของการทำ Refactoring ใน XP มาใช้สำหรับการปรับโครงสร้างของโปรแกรมให้อยู่ในรูปแบบที่ง่ายต่อการแก้ไขมีความถูกต้องสมบูรณ์ และมีประสิทธิภาพมากที่สุด รวมไปถึงสามารถนำการออกแบบที่ผ่านการตรวจสอบจาก SPII-XP 3.1 ไปใช้งานได้จริง

วิธีการวิจัย

การศึกษาครั้งนี้ใช้ระเบียบวิธีวิจัยแบบ การทดลอง (Experiment) เพื่อศึกษาถึงการนำวิธีการของ SPII-XP ไปทดลองใช้กับนักศึกษา โดยมีลำดับขั้นดังนี้ กำหนดประชากรและกลุ่มตัวอย่าง เครื่องมือที่ใช้ในการศึกษา การเก็บรวบรวมข้อมูลและสถิติที่ใช้ในการวิเคราะห์ข้อมูล

ผลการทดลอง

ผู้วิจัยได้นำเอา SPII-XP MODEL ไปทดลองใช้กับนักศึกษาระดับปริญญาตรีโดยแบ่งออกเป็น 4 กลุ่ม กลุ่มละ 4 คน โดยทำการทดลอง จำนวน 4 โปรแกรม ในภาษาจาวา คือ กลุ่มที่ 1 คือ กลุ่มที่ไม่ได้ใช้วิธีการใดเลยในการพัฒนาโปรแกรม (CTRL GROUP), กลุ่มที่ 2 ใช้วิธีการของ PSP (PSP GROUP), กลุ่มที่ 3 ใช้วิธีการของ XP (XP GROUP) และกลุ่มสุดท้ายใช้วิธีการของ SPII-XP (SPII-XP GROUP)

1. ผลผลิต (Productivity)

สำหรับการคำนวณหาค่าผลผลิต (Productivity) สามารถคำนวณได้จากขนาดของแรงงาน (effort) หารด้วยขนาด (Size) ของโปรแกรมในหน่วยของ (KLOC) ดังตารางที่ 3

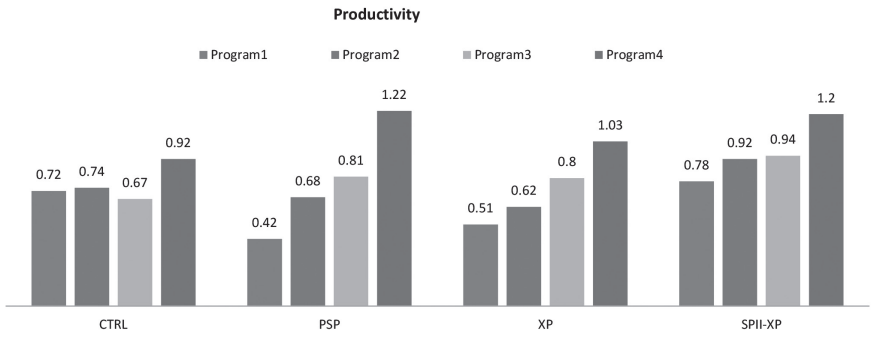
ตารางที่ 3 สูตรการคำนวณหาค่า Productivity [8]

Calculation	$\frac{\textbf{Effort}}{\textbf{size}}$	Work/Time

ส่วนตารางที่ 4 แสดงค่าเฉลี่ยของผลผลิต (Productivity) ซึ่งคำนวณได้จากสูตรในตารางที่ 3

ตารางที่ 4 แสดงค่าเฉลี่ยของผลผลิตที่ได้จากการคำนวณจากสูตรในตารางที่ 3

Group	CTRL	PSP	XP	SPII-XP
Program1	0.72	0.42	0.51	0.78
Program2	0.74	0.68	0.62	0.92
Program3	0.67	0.81	0.80	0.94
Program4	0.92	1.22	1.03	1.20



รูปที่ 5 Productivity by Development Program

จากรูปที่ 5 คือ กราฟแสดงค่าผลผลิตสำหรับการพัฒนาโปรแกรมของนักศึกษาจำนวน 4 กลุ่มสามารถอธิบายดังนี้ คือ จากการทดลอง 4 โปรแกรม พบว่า กลุ่ม SPII-XP จะมีค่า effort สูงกว่ากลุ่มอื่น ซึ่งแสดงให้เห็นว่ากลุ่มที่ใช้วิธีการของ SPII-XP มีการพัฒนาโปรแกรมที่มีค่าของผลผลิต (Productivity) สัมพันธ์กับขนาดของแรงงานที่ใช้ในการพัฒนาโปรแกรม ทำให้ได้โปรแกรมที่มีประสิทธิภาพ

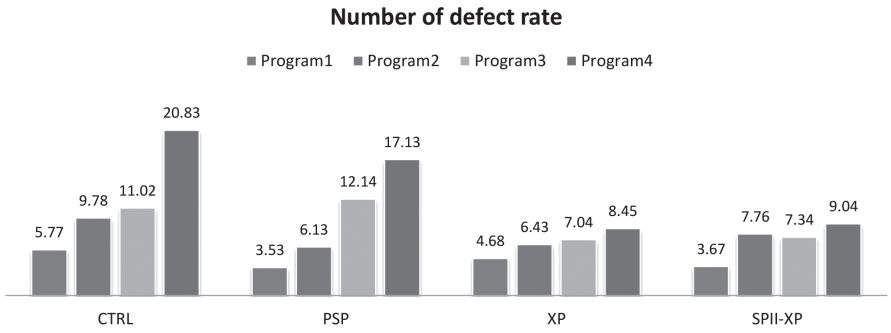
2. อัตราการข้อผิดพลาด (Defect Rate) สำหรับการคำนวณหาเปอร์เซ็นต์ของข้อผิดพลาด (Defect) นั้น สามารถคำนวณได้จาก เวลาที่แก้ไขข้อผิดพลาดที่เกิดขึ้น (Defect) หารด้วยเวลาที่ใช้ในการวางแผนสำหรับการพัฒนาโปรแกรมแล้วคูณด้วย 100 แสดงดังตารางที่ 5

ตารางที่ 5 สูตรการคำนวณหาข้อผิดพลาดที่เกิดจากการพัฒนาโปรแกรม

Calculation	$\left(\frac{\text{Time spent for bug fixing}}{\text{Plan time}}\right)$ X100	Percentage (%)

ตารางที่ 6 แสดงค่าเปอร์เซ็นต์ของข้อผิดพลาด (defect) ที่พบ ซึ่งคำนวณได้จากตารางที่ 5

Group	CTRL	PSP	XP	SPII-XP
Program1	5.77	3.53	4.68	3.67
Program2	9.78	6.13	6.43	7.76
Program3	11.02	12.14	7.04	7.34
Program4	20.83	17.13	8.45	9.04



รูปที่ 6 Number of defect rate

จากรูปที่ 6 คือ กราฟแสดงเปอร์เซ็นต์ข้อผิดพลาดที่เกิดขึ้นในระหว่างการพัฒนาโปรแกรม 4 โปรแกรม คือ โปรแกรมที่ 1 ถึง โปรแกรม 4 กลุ่ม CTRL พบว่ามีอัตราการเกิดข้อผิดพลาดสูงสุด ส่วนระดับที่ 2 จะเป็นกลุ่มที่ใช้วิธีการของ PSP ระดับที่ 3 คือ กลุ่มของ XP ส่วนกลุ่มที่พบข้อผิดพลาดที่เกิดจากการพัฒนาโปรแกรมน้อยที่สุด คือ กลุ่มที่ใช้วิธีการของ SPII-XP

3. ผลต่างของเวลาที่ใช้ในการพัฒนา (Relative Schedule Deviation) สำหรับการคำนวณหาผลต่างของเวลาที่ใช้จริงจากการพัฒนาโปรแกรมสามารถคำนวณได้จากเวลาที่ใช้สำหรับการวางแผน (Plan time) ลบกับเวลาที่ใช้จริง (Actual time) แล้วหารด้วยเวลาที่ใช้สำหรับการวางแผน (Plan time) คูณด้วย 10 ดังตารางที่ 7

ตารางที่ 7 แสดงสูตรการคำนวณหาผลต่างของเวลาที่ใช้ในการพัฒนาโปรแกรม

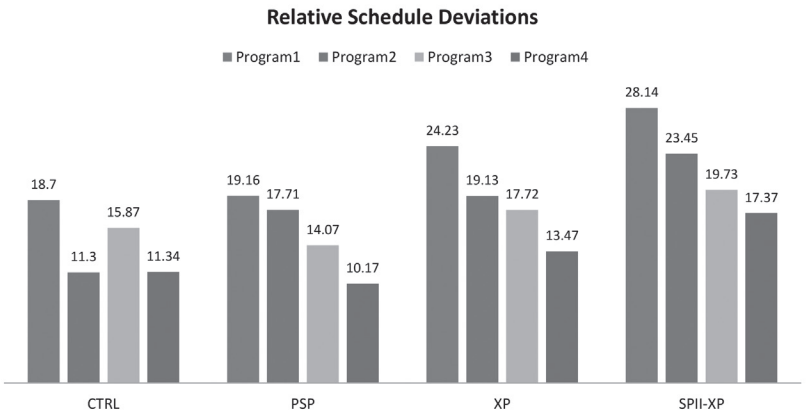
Calculation	$\left(\frac{\text{Planed time-Actual time}}{\text{Planed time}}\right)$ X100	Percentage (%)

ส่วนตารางที่ 8 แสดงค่าที่ได้จาการคำนวณหาผลต่างของเวลาที่ใช้ในการพัฒนาโปรแกรมของแต่ละกลุ่ม

ตารางที่ 8 แสดงค่าผลต่างของเวลาที่ใช้ในการพัฒนาโปรแกรมที่ได้จากการคำนวณในตารางที่ 7

Group	CTRL	PSP	XP	SPII-XP
Program1	18.7	19.16	24.23	28.14
Program2	11.3	17.71	19.13	23.45
Program3	15.87	14.07	17.72	19.73
Program4	11.34	10.17	13.47	17.37

จากข้อมูลในตารางที่ 8 พบว่า กลุ่มที่ใช้วิธีการของ SPII-XP มีเปอร์เซ็นต์ของค่าผลต่างของเวลาสูงกว่ากลุ่มอื่นๆ ส่วนกลุ่มที่ใช้วิธีการของ XP มีเป็นลำดับที่ 2 กลุ่มที่ใช้วิธีการของ PSP มีค่าเป็นลำดับที่ 3 และ ลำดับที่ 4 คือ กลุ่ม CTRL ดังนั้นแสดงให้เห็นว่ากลุ่มที่ใช้วิธีการของ SPII-XP สามารถพัฒนาโปรแกรมได้เสร็จทันภายในเวลาที่วางแผนไว้จึงทำให้มีค่าผลต่างเวลาเหลือมากกว่ากลุ่มอื่น แสดงดังรูปที่ 7



รูปที่ 7 Relative Schedule Deviations

อภิปรายและสรุปผลการวิจัย

จากการศึกษาและนำกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคลโดยใช้วิธีการของ Extreme Programming (SPII-XP: Software Process Improvement for Individual) ซึ่งเกิดจากการผสมผสานระหว่างกระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล (PSP: Personal Software Process) และกระบวนการพัฒนาซอฟต์แวร์แบบ Extreme Programming (XP)

SPII-XP ประกอบไปด้วย 3 กระบวนการ คือ การวัดส่วนบุคคล (Personal Measurement) กระบวนการวางแผนและประมาณการส่วนบุคคล (Personal Management and Estimating) และกระบวนการจัดการทางด้านคุณภาพส่วนบุคคล (Personal Quality Management) ซึ่งกระบวนการทั้งสามนี้เข้ามาช่วยในการปรับปรุงกระบวนการต่างๆ ของกระบวนการพัฒนาซอฟต์แวร์ให้มีคุณภาพสำหรับการนำไปปรับใช้ ดังนี้

1. กระบวนการจัดการและการประมาณการซอฟต์แวร์ ซึ่งนำมาปรับใช้ใน SPII-XP ส่วนของการประมาณขนาดนั้นจะใช้วิธีการของ Use Case point พบว่าเมื่อนำมาใช้กับการประมาณขนาดซอฟต์แวร์ของนักศึกษาแล้วได้ค่าที่ใกล้เคียงกับขนาดซอฟต์แวร์ที่พัฒนาขึ้นจริงและเป็นวิธีการที่เข้าใจง่ายกว่า วิธีการการประมาณขนาดซอฟต์แวร์ของ PSP คือ PROBE พบว่ามีความยุ่งยากต่อการนำมาใช้ ส่วนกลุ่มที่ใช้วิธีการของ XP ไม่มีการประมาณขนาดซอฟต์แวร์ จึงทำให้การพัฒนาซอฟต์แวร์ได้ขนาดใหญ่กว่า กลุ่มที่ใช้วิธีการของ PSP และกลุ่มที่ใช้วิธีการของ SPII-XP

2. การวัดกระบวนการส่วนบุคคล กลุ่มที่ใช้วิธีการของ SPII-XP นำกระบวนการของ PSP มาปรับใช้ในส่วนของวิธีการวัดขนาดและรูปแบบเอกสารแล้วทำให้ง่ายต่อการนำไปใช้งาน ส่วน XP ไม่มีการวัดขนาด และการวัดกระบวนการที่นักพัฒนาทำ ณ ปัจจุบัน รวมไปถึงรูปแบบของเอกสาร ซึ่งเมื่อนำมาปรับใช้ก็จะสามารถปรับปรุงกระบวนของนักพัฒนาให้ดีขึ้น ซึ่งพบว่า กลุ่มที่ใช้วิธีการของ PSP และกลุ่มที่ใช้วิธีการของ SPII-XP มีค่าของเวลาที่ใช้สำหรับการพัฒนาไม่แตกต่างกัน

3. กระบวนการจัดการด้านคุณภาพ กลุ่มที่ใช้วิธีการของ SPII-XP พบว่า เมื่อทำกระบวนการตรวจสอบโค้ดและการออกแบบแล้วจะช่วยให้ลดข้อผิดพลาด ส่วนกลุ่มที่ใช้วิธีการของ PSP พบว่า เมื่อทำกระบวนการตรวจสอบโค้ดและการออกแบบแล้วจะช่วยให้ลดข้อผิดพลาด ส่วนกลุ่มที่ใช้วิธีการของ XP พบว่าข้อผิดพลาดมากกว่ากลุ่มที่ใช้วิธีการของ PSP และ SPII-XP สำหรับการนำ SPII-XP ไปทดสอบใช้กับนักศึกษาพบว่า วิธีการที่นำมาใช้สำหรับการพัฒนาซอฟต์แวร์ที่มีผลต่อการพัฒนาต่อกระบวนการต่างๆ ไม่ว่าจะกระบวนการจัดการทางด้านคุณภาพ ซึ่งพบว่าถ้านักศึกษามีการทำกระบวนการจัดการทางด้านคุณภาพ เช่น การตรวจสอบโค้ดและการตรวจสอบการ

ออกแบบนั้นจะทำให้จำนวนข้อผิดพลาดที่เกิดจากการเขียนโค้ดและการออกแบบลดลง ส่วนกระบวนการจัดการและการประมาณการซอฟต์แวร์พบว่าถ้านักศึกษานำมาปรับใช้แล้วจะสามารถประมาณขนาดได้ใกล้เคียงกับขนาดจริง รวมไปถึงยังช่วยในการประมาณเวลาที่ใช้สำหรับการพัฒนาได้ใกล้เคียงเช่นกัน ส่วนการวัดกระบวนการส่วนบุคคลเมื่อนำมาปรับใช้กับนักศึกษาแล้วพบว่านักศึกษาสามารถเข้าใจกระบวนการต่างๆ ของขั้นตอนการพัฒนา และประเภทของข้อผิดพลาดที่เกิดจากการพัฒนามากขึ้น ซึ่งทำให้นักศึกษาเข้าใจถึงกระบวนการที่นักศึกษาทำการพัฒนาอยู่ ณ ปัจจุบันมากขึ้น

นอกจากการศึกษาและทำการวิจัยในครั้งนี้ พบว่าวิธีการพัฒนาของ SPII-XP ควรจะต้องมีการปรับปรุงบางกระบวนการให้มีความเหมาะสมสำหรับการนำไปปรับใช้กับกระบวนการทำงาน รวมไปถึงรูปแบบของเอกสารและการสร้างความเข้าใจกับผู้นำไปใช้ นอกจากนี้ กระบวนการที่ทำการวัดผลบางกระบวนการไม่สามารถบอกถึงความแตกต่างได้เพราะลักษณะของข้อมูล และกระบวนการที่นำมาใช้แตกต่างกัน

สำหรับ SPII-XP สามารถพัฒนาและนำไปปรับใช้กับการพัฒนาในรูปแบบของทีมพัฒนาซอฟต์แวร์ได้ คือ SPIT-XP (Software Process Team for Individual) เพื่อเพิ่มศักยภาพของกระบวนการพัฒนาซอฟต์แวร์ให้มีประสิทธิภาพและคุณภาพมากยิ่งขึ้น

บรรณานุกรม

สุรเดช จิตประไพกุลศาล. (2555). *กระบวนการพัฒนาซอฟต์แวร์เฉพาะบุคคล*. สืบค้นเมื่อ 30 ตุลาคม 2556, จาก http://www.slideshare.net/suradetj/03-using-psp0-12983756

Chatpreecha, P. (2008). *Software Process Improvement for Individual with Extreme Programming*. JSSE conference International. May 7-9, 2008 Felix River Kwai Resort, Kanchanaburi, Thailand.

Cockburn, A. (2007). *Agile Software Development*. Boston: Addison-Wesley.

Humphrey, W. S. (2005). *The Personal Software Process, Rationale and Status*. the 8th International Software Process Workshop. Germany: Wadern.

Humphrey, W. S. (2009). Using a defined and measured Personal Software Process. *IEEE Software, 13*(3), 77–88.

K. Beck. (2000). *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley.

Park, R. E. (2010). *Software Size Measurement: A Framework for Counting Source Statements*. Technical Report CMU/ SEI-92-TR-020, Software Engineering Institute, 1-242.

Watts, H. S. (2008). The Software Quality Challenge. *Crosstalk*, 21(6), 4-9.

Translated Thai References

Jitprapaikulsarn, S. (2012). *Personal Software Process.* Retrieved October 30, 2012, from http://www.slideshare.net/suradetj/03-using-psp0-12983756

Pornsiri Chatpreecha received her Bachelor Degree of Informatic, major in Computer Science from Sripatum University in 2004. She also received a scholarship as an outstanding student of the university. In 2008, she graduated master degree of Information Technology, major in Software Engineering from Sripatum University. In 2010, she received certificate her Personal Software Process Fundamentals and Personal Software Process Advance from Carnegie Mellon University (Software Engineering Institute) at United States of America. She is currently a full time lecturer in Faculty of Engineering and Technology, Panyapiwat Institute of Management.