บทที่ 5

บทสรุป

การวิจัยครั้งนี้มีวัตถุประสงเพื่อเปรียบเทียบประสิทธิภาพการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) และไม่ใช่อาไจล์ (Non-Agile) โดยศึกษาเป็นรายกรณี (Case Study) เพื่อมุ่งศึกษารายละเอียดของแต่ละกรณีศึกษาเพื่อหาข้อสรุป ซึ่งการเลือกกลุ่มตัวอย่างใช้วิธีการเลือกโดยวิธีเจาะจง (Purposive Sampling) ด้วยวิธีการแนะนำต่อๆ กันไป สำหรับการเลือกกลุ่มตัวอย่างที่ประยุกต์ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ผู้วิจัยเลือกทีมพัฒนาที่มีประสบการณ์ด้านการพัฒนาซอฟต์แวร์ด้วยระเบียบวิธีการพัฒนาตามแนวคิดแบบ อาไจล์ (Agile) 4 กรณีศึกษา จากนั้นเลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบไม่ใช่วิธีการ อาไจล์ (Non-Agile) ผู้วิจัยเลือกกลุ่มตัวอย่างโดยอ้างอิงคุณลักษณะของเป้าหมายทางธุรกิจ หรือเป้าหมายสำหรับการทำงานของซอฟต์แวร์ของกรณีศึกษาที่มีลักษณะคล้ายคลึงกับกลุ่มตัวอย่างที่เลือกใช้วิธีการพัฒนาซอฟต์แวร์ตามแนวคิดอาไจล์ (Agile) 4 กรณีศึกษาซึ่งสรุปผลการศึกษาได้ดังนี้

สรุปผลการวิจัย

จากคำถามของการวิจัยที่ว่า "ระเบียบวิธีการการพัฒนาซอฟต์แวร์ด้วยวิธีการ อาไจล์ (Agile) ช่วยส่งผลให้ซอฟต์แวร์ที่พัฒนาขึ้นมีประสิทธิภาพมากกว่าการวิเคราะห์ และออกแบบระบบตามวิธีการ ไม่ใช่หลักการตามแบบอาไจล์ (Non-Agile)" จากการวิเคราะห์ข้อมูลด้วยสถิติพรรณนาพบว่าระเบียบวิธีการการพัฒนาซอฟต์แวร์ด้วยวิธีการ อาไจล์ (Agile) ช่วยส่งผลให้ซอฟต์แวร์ที่พัฒนาขึ้นมีประสิทธิภาพมากกว่าการวิเคราะห์ และ ออกแบบระบบตามวิธีการ ไม่ใช่หลักการตามแบบอาไจล์ (Non-Agile) ทั้ง 4 มิติดังนี้ 1) มิติด้านการบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) 2) มิติด้านการบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) 3) มิติด้านคุณภาพของกระบวนการ (Quality of Process) และ 4) มิติด้านการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) ซึ่งสอดคล้องกับคำถามของการวิจัยที่ตั้งไว้ และจากการค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) ของสิ่งแวดล้อมที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ และแนวปฏิบัติหลักกรณีศึกษาประยุกต์ใช้กับโครงการพัฒนาซอฟต์แวร์ มีความสัมพันธ์กับประสิทธิผลของโครงการพัฒนาซอฟต์แวร์อย่างมี

นัยสำคัญทางสถิติที่ระดับ 0.05 และ 0.01 ซึ่ง สอดคล้องกับคำถามของการวิจัยที่ตั้งไว้ แสดงให้เห็นว่าตัวแปรด้านสิ่งแวดล้อมของโครงการ ปัจจัยนำเข้าของโครงการ แนวปฏิบัติหลักที่ประยุกต์ใช้เป็นปัจจัยที่มีความสัมพันธ์กับประสิทธิผลของโครงการพัฒนาซอฟต์แวร์

การอภิปรายผล

1. จากคำถามของการวิจัยที่ว่า "การพัฒนาซอฟต์แวร์ด้วยวิธีการ อาไจล์ (Agile) ส่งผลให้ผู้บริหารจัดการโครงการสามารถบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) ตามระยะเวลาการพัฒนาซอฟต์แวร์ในแต่ละขั้นตอนของการพัฒนาตามหลักการของวงจรการพัฒนาซอฟต์แวร์แตกต่างจากการพัฒนาซอฟต์แวร์ด้วยวิธีการที่ไม่ใช่หลักการตามแบบอาไจล์ (Non-Agile) หรือไม่" จากการวิเคราะห์ข้อมูลตารางที่ 17 พบว่ากรณีศึกษาที่มีการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการ อาไจล์ (Agile) มากกว่ามีแนวโน้มที่จะสามารถบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) มากขึ้น โดยกรณีศึกษาที่เลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ร้อยละ 50 สามารถบริหารจัดการแผนการดำเนินงานได้เร็วกว่าการประมาณการ ซึ่งสอดคล้องกับผลจากการแสดงข้อมูลและกระจายตัว และการหาค่าสหสัมพันธ์ของสิ่งแวดล้อม โดยอธิบายได้ดังนี้

1.1 มาตรฐานขององค์กรแสดงการกระจายตัวของข้อมูลจากตาราง 25 และภาพ 20 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 26 สรุปได้ว่ากรณีศึกษาที่มีการกำหนดมาตรฐานขององค์กรจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการกำหนดมาตรฐานขององค์กรมีความสัมพันธ์ทางลบกับการบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการกำหนดมาตรฐานขององค์กร และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) มีประสิทธิภาพดีขึ้น

1.2 ความกดดันในเรื่องของงบประมาณแสดงการกระจายตัวของข้อมูลจากตาราง 27 และภาพ 21 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 28 สรุปได้ว่ากรณีศึกษาที่มีความกดดันในเรื่องของงบประมาณน้อยกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งความกดดันในเรื่องของงบประมาณมีความสัมพันธ์ทางลบกับการบริหารจัดการแผนการดำเนินงานอย่าง

มีประสิทธิภาพ (Effective Time Management) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งความกดดันในเรื่องของงบประมาณ และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) มีประสิทธิภาพ ดีขึ้น

1.3 ประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) แสดงการกระจายตัวของข้อมูลจากตาราง 39 และภาพ 27 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 40 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) มากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) มีความสัมพันธ์ทางลบกับการบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) มีประสิทธิภาพดีขึ้น

1.4 ประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบแสดงการกระจายตัวของข้อมูลจากตาราง 41 และภาพ 28 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 42 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบมากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบมีความสัมพันธ์ทางลบกับการบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการแผนการดำเนินงานอย่างมีประสิทธิภาพ (Effective Time Management) มีประสิทธิภาพดีขึ้น

2. จากคำถามของการวิจัยที่ว่า "การพัฒนาซอฟต์แวร์ด้วยวิธีการ อาไจล์ (Agile) ส่งผลให้ผู้บริหารจัดการโครงการสามารถบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) ภายใต้เงื่อนไขของงบประมาณแตกต่างจากการพัฒนาซอฟต์แวร์ด้วยวิธีการที่ไม่ใช่หลักการตามแบบอาไจล์ (Non-Agile) หรือไม่" จากการวิเคราะห์ข้อมูลตาราง 18 พบว่า

กรณีศึกษาที่มีการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการ อาไจล์ (Agile) มากกว่ามีแนวโน้มที่จะสามารถเพิ่มการจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) มากขึ้น โดยกรณีศึกษาที่เลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ร้อยละ 50 สามารถบริหารจัดการงบประมาณต่ำกว่าการประมาณการ ซึ่งสอดคล้องกับผลจากการแสดงข้อมูลและกระจายตัว และการหาค่าสหสัมพันธ์ของสิ่งแวดล้อม โดยอธิบายได้ดังนี้

2.1 มาตรฐานขององค์กรแสดงการกระจายตัวของข้อมูลจากตาราง 25 และภาพ 20 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 26 สรุปได้ว่า กรณีศึกษาที่มีการกำหนดมาตรฐานขององค์กรจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการกำหนดมาตรฐานขององค์กรมีความสัมพันธ์ทางลบกับการบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการกำหนดมาตรฐานขององค์กร และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) มีประสิทธิภาพดีขึ้น

2.2 ความกดดันในเรื่องของงบประมาณแสดงการกระจายตัวของข้อมูลจากตาราง 27 และภาพ 21 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 28 สรุปได้ว่ากรณีศึกษาที่มีความกดดันในเรื่องของงบประมาณน้อยกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งความกดดันในเรื่องของงบประมาณมีความสัมพันธ์ทางลบกับการบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งความกดดันในเรื่องของงบประมาณ และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) มีประสิทธิภาพดีขึ้น

2.3 ประสบการณ์การทำงานและความคุ้นเคยเกี่ยวกับภาษา แสดงการกระจายตัวของข้อมูลจากตาราง 39 และภาพ 27 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 40 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์การทำงานและความคุ้นเคยเกี่ยวกับภาษามากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์การทำงานและความคุ้นเคย

เกี่ยวกับภาษามีความสัมพันธ์ทางลบกับการบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งประสบการณ์การทำงานและความคุ้นเคยเกี่ยวกับภาษา และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) มีประสิทธิภาพดีขึ้น

2.4 ประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) แสดงการกระจายตัวของข้อมูลจากตาราง 39 และภาพ 27 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 40 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) มากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) มีความสัมพันธ์ทางลบกับการบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) มีประสิทธิภาพดีขึ้น

2.5 ประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบแสดงการกระจายตัวของข้อมูลจากตาราง 41 และภาพ 28 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 42 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบมากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบมีความสัมพันธ์ทางลบกับการบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การบริหารจัดการงบประมาณอย่างมีประสิทธิภาพ (Effective Cost Management) มีประสิทธิภาพดีขึ้น

3. จากคำถามของการวิจัยที่ว่า "การพัฒนาซอฟต์แวร์ด้วยวิธีการ อาไจล์ (Agile) ส่งผลให้ประสิทธิภาพการบริหารจัดการคุณภาพของกระบวนการ (Quality of Process) แตกต่างจากการพัฒนาซอฟต์แวร์ด้วยวิธีการ ไม่ใช่หลักการตามแบบอาไจล์ (Non-Agile) หรือไม่" จากการวิเคราะห์ข้อมูลตาราง 19 พบว่า กรณีศึกษาที่มีการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการ

พัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) มากกว่ามีแนวโน้มที่จะสามารถเพิ่มคุณภาพของกระบวนการ (Quality of Process) ได้มากขึ้น โดยกรณีศึกษาที่เลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) มีค่าเฉลี่ยของความพึงพอใจด้านคุณภาพของกระบวนการพัฒนาซอฟต์แวร์มากกว่ากรณีศึกษาที่เลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการไม่ใช่อาไจล์ (Non-Agile) ในทุกตัวชี้วัด ซึ่งสอดคล้องกับผลจากการแสดงข้อมูลและกระจายตัว และการหาค่าสหสัมพันธ์ของสิ่งแวดล้อม โดยอธิบายได้ดังนี้

3.1 ระยะเวลาการประกอบการขององค์กร/ หน่วยงานแสดงการกระจายตัวของข้อมูลจากตาราง 21 และภาพ 18 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 22 สรุปได้ว่ากรณีศึกษาที่มีระยะเวลาการประกอบการขององค์กร/ หน่วยงานมากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งระยะเวลาการประกอบการขององค์กร/ หน่วยงานมีความสัมพันธ์ทางบวกกับคุณภาพของกระบวนการ (Quality of Process) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งระยะเวลาการประกอบการขององค์กร/ หน่วยงาน และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้คุณภาพของกระบวนการ (Quality of Process) มีประสิทธิภาพดีขึ้น

3.2 รูปแบบวัฒนธรรมองค์กร (Business culture) ในการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นแสดงการกระจายตัวของข้อมูลจากตาราง 23 และภาพ 19 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 24 สรุปได้ว่ากรณีศึกษาที่มีการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นมีความสัมพันธ์ทางลบกับคุณภาพของกระบวนการ (Quality of Process) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการ อาไจล์ (Agile) ส่งผลให้คุณภาพของกระบวนการ (Quality of Process) มีประสิทธิภาพดีขึ้น

3.3 มาตรฐานขององค์กรแสดงการกระจายตัวของข้อมูลจากตาราง 25 และภาพ 20 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 26 สรุปได้ว่ากรณีศึกษาที่มีการกำหนดมาตรฐานขององค์กรจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการกำหนดมาตรฐานของ

องค์กรมีความสัมพันธ์ทางลบกับคุณภาพของกระบวนการ (Quality of Process) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการกำหนดมาตรฐานขององค์กร และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้คุณภาพของกระบวนการ (Quality of Process) มีประสิทธิภาพดีขึ้น

3.4 ความกดดันในเรื่องของงบประมาณแสดงการกระจายตัวของข้อมูลของข้อมูลจากตาราง 27 และภาพ 21 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 28 สรุปได้ว่ากรณีศึกษาที่มีความกดดันในเรื่องของงบประมาณน้อยกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งความกดดันในเรื่องของงบประมาณมีความสัมพันธ์ทางลบกับคุณภาพของกระบวนการ (Quality of Process) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งความกดดันในเรื่องของงบประมาณ และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้คุณภาพของกระบวนการ (Quality of Process) มีประสิทธิภาพดีขึ้น

3.5 ความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์แสดงการกระจายตัวของข้อมูลจากตาราง 31 และภาพ 23 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 32 สรุปได้ว่ากรณีศึกษาที่มีการให้ความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์มากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการให้ความสำคัญกับความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์ทางบวกกับคุณภาพของกระบวนการ (Quality of Process) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้คุณภาพของกระบวนการ (Quality of Process) มีประสิทธิภาพ ดีขึ้น

3.6 การแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่แสดงการกระจายตัวของข้อมูลจากตาราง 33 และภาพ 24 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 34 สรุปได้ว่ากรณีศึกษาที่มีการแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่มากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่มีความสัมพันธ์ทางบวกกับคุณภาพของกระบวนการ (Quality of Process) อย่างมีนัยสำคัญ

ทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งการแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้คุณภาพของกระบวนการ (Quality of Process) มีประสิทธิภาพดีขึ้น

4. จากคำถามของการวิจัยที่ว่า "การพัฒนาซอฟต์แวร์ด้วยวิธีการ อาไจล์ (Agile) ส่งผลให้ประสิทธิภาพด้านการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) แตกต่างจากการพัฒนาซอฟต์แวร์ด้วยวิธีการที่ไม่ใช่หลักการตามแบบอาไจล์ (Non-Agile) หรือไม่" จากการวิเคราะห์ข้อมูลตาราง 20 พบว่ากรณีศึกษาที่มีการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) มากกว่ามีแนวโน้มที่จะสามารถเพิ่มประสิทธิภาพด้านการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มากขึ้น โดยกรณีศึกษาที่เลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) มีค่าเฉลี่ยของความพึงพอใจด้านการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มากกว่ากรณีศึกษาที่เลือกใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการไม่ใช่อาไจล์ (Non-Agile) ในทุกตัวชี้วัด ซึ่งสอดคล้องกับผลจากการแสดงข้อมูลและกระจายตัว และการหาค่าสหสัมพันธ์ของสิ่งแวดล้อม โดยอธิบายได้ดังนี้

4.1 ระยะเวลาการประกอบการขององค์กร/ หน่วยงานแสดงการกระจายตัวของข้อมูลจากตาราง 21 และภาพ 18 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 22 สรุปได้ว่ากรณีศึกษาที่มีระยะเวลาการประกอบการขององค์กร/ หน่วยงานมากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งระยะเวลาการประกอบการขององค์กร/ หน่วยงานมีความสัมพันธ์ทางบวกกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการกำหนดมาตรฐานขององค์กร และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.2 รูปแบบวัฒนธรรมองค์กร (Business culture) ในการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นแสดงการกระจายตัวของข้อมูลจากตาราง 23 และภาพ 19 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 24 สรุป

ได้ว่ากรณีศึกษาที่มีการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นมีความสัมพันธ์ทางลบกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการยอมรับเทคโนโลยี (Technology) หรือวิธีการ (Method) ใหม่เกิดขึ้นและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการ อาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.3 มาตรฐานขององค์กรแสดงการกระจายตัวของข้อมูลจากตาราง 25 และภาพ 20 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 26 สรุปได้ว่ากรณีศึกษาที่มีการกำหนดมาตรฐานขององค์กรจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการกำหนดมาตรฐานขององค์กรมีความสัมพันธ์ทางลบกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งการกำหนดมาตรฐานขององค์กร และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.4 ความกดดันในเรื่องของงบประมาณแสดงการกระจายตัวของข้อมูลจากตาราง 27 และภาพ 21 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 28 สรุปได้ว่ากรณีศึกษาที่มีความกดดันในเรื่องของงบประมาณน้อยกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งความกดดันในเรื่องของงบประมาณมีความสัมพันธ์ทางลบกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งความกดดันในเรื่องของงบประมาณ และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.5 การให้ความสำคัญกับการพัฒนาทรัพยากรมนุษย์ การทำงานเป็นทีม และการมอบหมายความไว้วางใจแก่พนักงานแสดงการกระจายตัวของข้อมูลจากตาราง 27 และภาพ 21 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 28 สรุปได้ว่ากรณีศึกษาที่มีการให้ความสำคัญกับการพัฒนาทรัพยากรมนุษย์ การทำงานเป็นทีม และการมอบหมายความไว้วางใจแก่พนักงานมากว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการให้ความสำคัญกับการพัฒนาทรัพยากรมนุษย์ การทำงานเป็นทีม และการมอบหมายความไว้วางใจแก่พนักงานมีความสัมพันธ์ทางบวกกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งการให้ความสำคัญกับการพัฒนาทรัพยากรมนุษย์ การทำงานเป็นทีม และการมอบหมายความไว้วางใจแก่พนักงานและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.6 ความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรมและผลิตภัณฑ์แสดงการกระจายตัวของข้อมูลจากตาราง 31 และภาพ 23 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 32 สรุปได้ว่ากรณีศึกษาที่มีการให้ความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์มากว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการให้ความสำคัญกับความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์ทางบวกกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งความเป็นเอกลักษณ์ ทันสมัยของผลิตภัณฑ์ โดยผลักดันให้เป็นผู้นำด้านนวัตกรรม และผลิตภัณฑ์และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.7 การแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่แสดงการกระจายตัวของข้อมูลจากตาราง 33 และภาพ 24 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 34 สรุปได้ว่ากรณีศึกษาที่มีการแลกเปลี่ยนความรู้ นวัตกรรม และ

เทคโนโลยีใหม่มากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งการแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่มีความสัมพันธ์ทางบวกกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งการแลกเปลี่ยนความรู้ นวัตกรรม และเทคโนโลยีใหม่และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.8 ความคล้ายคลึงกับโครงการเดิม หรือมีแบบอย่างมาก่อน เมื่อเทียบกับโครงการปัจจุบันแสดงการกระจายตัวของข้อมูลจากตาราง 33 และภาพ 24 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 34 สรุปได้ว่ากรณีศึกษาที่มีความคล้ายคลึงกับโครงการเดิม หรือมีแบบอย่างมาก่อน เมื่อเทียบกับโครงการปัจจุบันสูงกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งความคล้ายคลึงกับโครงการเดิม หรือมีแบบอย่างมาก่อน เมื่อเทียบกับโครงการปัจจุบันมีความสัมพันธ์ทางบวกกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) แปลความหมายได้ว่าทั้งความคล้ายคลึงกับโครงการเดิม หรือมีแบบอย่างมาก่อน เมื่อเทียบกับโครงการปัจจุบันและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.9 ประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) แสดงการกระจายตัวของข้อมูลจากตาราง 39 และภาพ 27 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 40 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) มากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) มีความสัมพันธ์ทางลบกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.05 แปลความหมายได้ว่าทั้งประสบการณ์การทำงานที่มีต่อระบบการทำงาน (Platform) และการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์

(Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

4.10 ประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบแสดงการกระจายตัวของข้อมูลจากตาราง 41 และภาพ 28 และผลจากการหาค่าสหสัมพันธ์อย่างง่าย (Simple Correlation) จากตาราง 42 สรุปได้ว่ากรณีศึกษาที่มีประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบมากกว่าจะมีแนวโน้มประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ที่สูงขึ้น ซึ่งประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบมีความสัมพันธ์ทางลบกับการทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) อย่างมีนัยสำคัญทางสถิติที่ 0.01 แปลความหมายได้ว่าทั้งประสบการณ์ของทีมในการทำงานกับระเบียบวิธีการพัฒนาระบบและการประยุกต์ใช้แนวปฏิบัติหลักตามระเบียบวิธีการพัฒนาซอฟต์แวร์ตามหลักการอาไจล์ (Agile) ส่งผลให้การทำงานร่วมกันของผู้ที่เกี่ยวข้องกับโครงการพัฒนาซอฟต์แวร์ (Inter-supplier performance and Customer satisfactions) มีประสิทธิภาพดีขึ้น

ทัศนคติของกรณีศึกษาที่มีต่อแนวปฏิบัติหลักที่ประยุกต์ใช้ในโครงการพัฒนาซอฟต์แวร์

ตาราง 70 แสดงสรุปผลความพึงพอใจจากการประยุกต์ใช้แนวปฏิบัติหลักของกรณีศึกษา

		Cycle time	Cost and benefit Effective	quality of process	inter-supplier performance
Techniques	n	Median	Median	Median	Median
Short release	4	+	0	+	+++
Metaphor	4	+++	+	++	+++
Simple Design	5	++	+	+	++
Refactoring	4	+++	0	+++	++
Pair programming	2	++	+	+++	++
Collective ownership	5	+	0	++	+++
Continuous integration	3	+	0	+	0
On-site customer	5	++	0	+++	+++
40-h week	4	-	0	0	0
Test first Development	3	0	0	++	++
Coding standards	3	++	+	+++	++
Scrum teams	2	++	+	+++	+++
Product backlog	3	++	0	+	+
Sprint	3	+	+	+	++
Sprint review	3	-	0	+++	+++
Individual class ownership	1	+++	+	+++	+++
Feature teams	4	+++	++	+++	+++

ตาราง 70 (ต่อ)

Techniques	n	Cycle time Median	Cost and benefit Effective Median	quality of process Median	inter-supplier performance Median
Developing by components	4	+++	+	++	+
Software inspection	2	-	-	+++	+++
Iterative and incremental development	2	+	0	++	++
Requirements are baselined at a high level	1	+++	+++	+++	++
The planning game	2	+	0	++	++
Scrum master	2	+	0	++	++
Sprint Planning meeting	3	+++	++	+	++
Daily scrum meeting	2	+	0	++	+++
Configuration management	1	0	0	+	+

วงจรเวลาในการพัฒนา (cycle time)

1. Short release: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำเสนอผลการพัฒนาทีละน้อยอย่างต่อเนื่อง โดยพัฒนาเป็นวงรอบสั้นๆ เพื่อให้ผู้ที่เกี่ยวข้องเห็นภาพของระบบที่กำลังดำเนินการพัฒนา (Small Releases of Software Product) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

2. Metaphor: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบสร้างวิสัยทัศน์การออกแบบร่วมกันเพื่อให้ผู้ที่เกี่ยวข้องเข้าใจการทำงานของระบบ (System metaphor development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

3. Simple Design: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการออกแบบด้วยโครงสร้างง่ายๆ ไม่ซับซ้อนสามารถแก้ไข หรือเปลี่ยนแปลงง่าย และคล่องตัวขึ้น (Design is kept as simple as possible) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

4. Refactoring: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการปรับเปลี่ยนโครงสร้างของโค้ช (code) เพื่อให้มันสามารถจัดการได้ง่ายขึ้น และทำให้เราสามารถรับมือกับการเปลี่ยนแปลงได้ง่ายขึ้น มีความยืดหยุ่น และรองรับการเปลี่ยนแปลงได้ตลอดเวลา (Refactoring of code) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

5. Pair programming: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาโปรแกรมเป็นคู่ สลับหน้าที่กันระหว่างการพัฒนาโปรแกรม และการตรวจสอบ เสมือนใช้คอมพิวเตอร์เครื่องเดียวกัน เพื่อให้ซอฟต์แวร์ที่พัฒนาขึ้นมีคุณภาพ เนื่องจากมีการตรวจสอบอยู่ตลอดเวลา (pair programming) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

6. Collective ownership: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการสร้างความมีส่วนร่วมในทุกส่วนของโปรแกรม ทีมพัฒนาทุกคนสามารถเข้าไปศึกษา หรือแม้แต่ปรับปรุงพัฒนาโค้ช (Code) ได้ (Collective ownership of code) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

7. Continuous integration: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการรวมการทำงานของซอฟต์แวร์อย่างต่อเนื่อง เพื่อตรวจสอบความสามารถของการทำงานของซอฟต์แวร์ (Continue Integration) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

8. On-site customer: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการยินยอมให้ลูกค้าเข้ามามีส่วนเกี่ยวข้องกับการพัฒนาระบบ เช่น กระบวนการทดสอบ หรือการกำหนดผลยอมรับของการทดสอบ (Customer on-site) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

9. 40-h week: - บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบทำงาน 40 ชั่วโมงต่อสัปดาห์ (40 Hour week) ส่งผลในทางตรงกันข้ามต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

10. Test first Development: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำการทดสอบและผลการทดสอบมาผลักดันโครงการ หรือนำผลการทดสอบมาใช้ในการวางแผนการทำงานวงรอบถัดไป (Test first Development) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

11. Coding standards: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำมาตรฐานการเขียนโปรแกรม เพื่อให้ทีมพัฒนามีความเข้าใจร่วมกันในการพัฒนาโปรแกรม (Coding to an agree standard) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

12. Scrum teams: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการระดมความคิดในเชิงการพัฒนาซอฟต์แวร์ เพื่อให้ผู้ที่เกี่ยวข้องมีส่วนร่วมในการทำงาน และเกิดการสื่อสารระหว่างทีม และ

เกิดการกระจายสามรู้ในทีม (Scrum Team) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

13. Product backlog: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำรายการความต้องการที่ถูกจัดแยกสิ่งที่ต้องพัฒนาในแต่ละวงรอบ (Product backlog) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

14. Sprint: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทำงานแต่ละวงรอบของทีม โดยยินยอมให้ผู้ที่มีส่วนเกี่ยวข้องทุกฝ่ายเข้ามามีส่วนร่วมสำหรับการพัฒนา พิจารณาความก้าวหน้าและจัดส่งซอฟต์แวร์ที่สามารถทำงานได้ในแต่ละวงรอบ เพื่อผลักดันให้เกิดการพัฒนาต่อไปได้อย่างรวดเร็ว (Sprint) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

15. Sprint review: - บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทบทวนการดำเนินงานในแต่ละวงรอบ (Sprint review) ส่งผลในทางตรงกันข้ามต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

16. Individual class ownership: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการแบ่งระดับการดูแล และสิทธิในการเข้าถึงโค้ช (Code) เพื่อให้การพัฒนาเป็นไปอย่างคล่องตัวขึ้น สามารถเรียกใช้คลาส (Class) อย่างรวดเร็ว (Individual class ownership) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

17. Feature teams: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดตั้งทีมที่มีขนาดเล็ก มีระดับความสามารถสูง ความคล่องตัวสูง และเต็มไปด้วยความคิดสร้างสรรค์ สมาชิกในทีมอาจจะต้องทำหน้าที่อื่นนอกเหนือจากทักษะที่ตนเองชำนาญ เช่น นักวิเคราะห์ระบบอาจจะต้องทำหน้าที่เป็นผู้ทดสอบ สนับสนุนการพัฒนาที่ขับเคลื่อนด้วยคุณลักษณะของซอฟต์แวร์ และการจัดการโครงการที่มีรายละเอียดค่อนข้างมากกว่าโครงการอื่นๆ ทีมจะมุ่งพัฒนาคุณลักษณะที่สามารถทำงานได้ภายในเวลาทุกๆ 2 สัปดาห์ เมื่อโครงการมีขนาดและความซับซ้อนมากขึ้น นักพัฒนา ผู้บริหารโครงการ และลูกค้าจำเป็นต้อง เข้าใจสถานการณ์ของคุณลักษณะของซอฟต์แวร์ทั้งหมด (Feature teams) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

18. Developing by components: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาจากคอมโพเนนท์ (Component Development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

19. Software inspection: - บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการตรวจสอบข้อผิดพลาดของซอฟต์แวร์ โดยมุ่งไปที่ข้อผิดพลาดของการทำงานแต่ละส่วน (Software inspections) ส่งผลในทางตรงกันข้ามต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

20. Iterative and incremental development: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาเป็นวงรอบ โดยอาศัยการเพิ่มเติมส่วนงานเข้าไปในงานเดิมเรื่อยๆ เพื่อให้มีความสมบูรณ์มากขึ้น (Iterative and incremental development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

21. Requirements are base lined at a high level: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดเตรียมรายการความต้องการที่มีความคงที่สูง (Requirements are baseline at a high level) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

22. The planning game: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบวางแผนร่วมกันระหว่าง ผู้ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ก่อนจะเริ่มการวนรอบ โดยมีการกำหนดรายละเอียด และขอบเขตการทำงานแต่ละวนรอบ (Planning Game) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

23. Scrum master: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบผู้ที่ทำหน้าที่ดูแลทีมงาน เป็นโค้ชของทีมงาน เป็นคนที่รับ ผิดชอบคุณภาพของผลงาน จัดลำดับความสำคัญของงาน (Scrum master) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

24. Sprint Planning meeting: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการประชุมวางแผนการวนรอบถึง เป้าหมาย และการพัฒนาเพิ่ม (Sprint Planning meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

25. Daily scrum meeting: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบจัดประชุมทีมทุกวัน เพื่อติดตามความคืบหน้า และช่วยสำหรับการวางแผน และปรับปรุงกระบวนการพัฒนา (Daily scrum meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

26. Configuration management: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการบริหารจัดการเกี่ยวกับการกำหนดคุณสมบัติของคอมพิวเตอร์ อุปกรณ์หรือโปรแกรมใดๆ ที่จะนำมาใช้กับคอมพิวเตอร์เพื่อให้ทำงานมีประสิทธิภาพ เหมาะสมกับงานที่ต้องการ (Configuration management) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

การบริหารจัดการงบประมาณ (Cost and benefit Effective)

1. Short release: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำเสนอผลการพัฒนาทีละน้อยอย่างต่อเนื่อง โดยพัฒนาเป็นวงรอบสั้นๆ เพื่อให้ผู้ที่เกี่ยวข้องเห็นภาพของระบบที่กำลังดำเนินการพัฒนา (Small Releases of Software Product) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

2. Metaphor: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบสร้างวิสัยทัศน์การออกแบบร่วมกันเพื่อให้ผู้ที่เกี่ยวข้องเข้าใจการทำงานของระบบ (System metaphor development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

3. Simple Design: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการออกแบบด้วยโครงสร้างง่ายๆ ไม่ซับซ้อนสามารถแก้ไข หรือเปลี่ยนแปลงง่าย และคล่องตัวขึ้น (Design is kept as simple as possible) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

4. Refactoring: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการปรับเปลี่ยนโครงสร้างของโค้ช (code) เพื่อให้มันสามารถจัดการได้ง่ายขึ้น และทำให้เราสามารถรับมือกับการเปลี่ยนแปลงได้ง่ายขึ้น มีความยืดหยุ่น และรองรับการเปลี่ยนแปลงได้ตลอดเวลา (Refactoring of code) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

5. Pair programming: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาโปรแกรมเป็นคู่สลับหน้าที่กันระหว่างการพัฒนาโปรแกรม และการตรวจสอบ เสมือนใช้คอมพิวเตอร์เครื่องเดียวกัน เพื่อให้ซอฟต์แวร์ที่พัฒนาขึ้นมีคุณภาพ เนื่องจากมีการตรวจสอบอยู่ตลอดเวลา (pair programming) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

6. Collective ownership: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการสร้างความมีส่วนร่วมในทุกส่วนของโปรแกรม ทีมพัฒนาทุกคนสามารถเข้าไปศึกษา หรือแม้แต่ปรับปรุงพัฒนาโค้ช (Code) ได้ (Collective ownership of code) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

7. Continuous integration: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการรวมการทำงานของซอฟต์แวร์อย่างต่อเนื่อง เพื่อตรวจสอบความสามารถของการทำงานของซอฟต์แวร์ (Continue Integration) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

8. On-site customer: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการยินยอมให้ลูกค้าเข้ามามีส่วนเกี่ยวข้องกับการพัฒนาระบบ เช่น กระบวนการทดสอบ หรือการกำหนดผลยอมรับของการทดสอบ (Customer on-site) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

9. 40-h week: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบทำงาน 40 ชั่วโมงต่อสัปดาห์ (40 Hour week) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

10. Test first Development: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำการทดสอบและผลการทดสอบมาผลักดันโครงการ หรือนำผลการทดสอบมาใช้ในการวางแผนการทำงานวงรอบถัดไป (Test first Development) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

11. Coding standards: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำมาตรฐานการเขียนโปรแกรม เพื่อให้ทีมพัฒนามีความเข้าใจร่วมกันในการพัฒนาโปรแกรม (Coding to an agree standard) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบระบบบางส่วน

12. Scrum teams: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการระดมความคิดในเชิงการพัฒนาซอฟต์แวร์ เพื่อให้ผู้ที่เกี่ยวข้องมีส่วนร่วมในการทำงาน และเกิดการสื่อสารระหว่างทีม และเกิดการกระจายสามรู้ในทีม (Scrum Team) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

13. Product backlog: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำรายการความต้องการที่ถูกจัดแยกสิ่งที่ต้องพัฒนาในแต่ละวงรอบ (Product backlog) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

14. Sprint: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทำงานแต่ละวงรอบของทีม โดยยินยอมให้ผู้ที่มีส่วนเกี่ยวข้องทุกฝ่ายเข้ามามีส่วนร่วมสำหรับการพัฒนา พิจารณาความก้าวหน้าและจัดส่งซอฟต์แวร์ที่สามารถทำงานได้ในแต่ละวงรอบ เพื่อผลักดันให้เกิดการพัฒนาต่อไปได้อย่างรวดเร็ว (Sprint) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

15. Sprint review: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทบทวนการดำเนินงานในแต่ละวงรอบ (Sprint review) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

16. Individual class ownership: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการแบ่งระดับการดูแล และสิทธิในการเข้าถึงโค้ซ (Code) เพื่อให้การพัฒนาเป็นไปอย่างคล่องตัวขึ้น สามารถเรียกใช้คลาส (Class) อย่างวดเร็ว (Individual class ownership) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

17. Feature teams: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดตั้งทีมที่มีขนาดเล็ก มีระดับความสามารถสูง ความคล่องตัวสูง และเต็มไปด้วยความคิดสร้างสรรค์ สมาชิกในทีมอาจจะต้องทำหน้าที่อื่นนอกเหนือจากทักษะที่ตนเองชำนาญ เช่น นักวิเคราะห์ระบบอาจจะต้องทำหน้าที่เป็นผู้ทดสอบ สนับสนุนการพัฒนาที่ขับเคลื่อนด้วยคุณลักษณะของซอฟต์แวร์ และการจัดการ

โครงการที่มีรายละเอียดค่อนข้างมากกว่าโครงการอื่นๆ ทีมจะมุ่งพัฒนาคุณลักษณะที่สามารถทำงานได้ภายในเวลาทุกๆ 2 สัปดาห์ เมื่อ โครงการมีขนาดและความซับซ้อนมากขึ้น นักพัฒนา ผู้บริหารโครงการ และลูกค้าจำเป็นต้อง เข้าใจสถานการณ์ของคุณลักษณะของซอฟต์แวร์ทั้งหมด (Feature teams) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

18. Developing by components: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาจากคอมโพเนนท์ (Component Development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

19. Software inspection: - บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการตรวจสอบข้อผิดพลาดของซอฟต์แวร์ โดยมุ่งไปที่ข้อผิดพลาดของการทำงานแต่ละส่วน (Software inspections) ส่งผลในทางตรงกันข้ามต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

20. Iterative and incremental development: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาเป็นวงรอบ โดยอาศัยการเพิ่มเติมส่วนงานเข้าไปในงานเดิมเรื่อยๆ เพื่อให้มีความสมบูรณ์มากขึ้น (Iterative and incremental development) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

21. Requirements are base lined at a high level: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดเตรียมรายการความต้องการที่มีความคงที่สูง (Requirements are baseline at a high level) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

22. The planning game: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบวางแผนร่วมกันระหว่าง ผู้ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ก่อนจะเริ่มการวนรอบ โดยมีการกำหนดรายละเอียด และขอบเขตการทำงานแต่ละวนรอบ (Planning Game) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

23. Scrum master: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบผู้ที่ทำหน้าที่ดูแลทีมงาน เป็นโค้ชของทีมงาน เป็นคนที่รับ ผิดชอบคุณภาพของผลงาน จัดลำดับความสำคัญของงาน (Scrum master) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

24. Sprint Planning meeting: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการประชุมวางแผนการวนรอบถึงเป้าหมาย และการพัฒนาเพิ่ม (Sprint Planning meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

25. Daily scrum meeting: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบจัดประชุมทีมทุกวัน เพื่อติดตามความคืบหน้า และช่วยสำหรับการวางแผน และปรับปรุงกระบวนการพัฒนา (Daily scrum meeting) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

26.Configuration management: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการบริหารจัดการเกี่ยวกับการกำหนดคุณสมบัติของคอมพิวเตอร์ อุปกรณ์หรือโปรแกรมใดๆ ที่จะนำมาใช้กับคอมพิวเตอร์ เพื่อให้ทำงานมีประสิทธิภาพ เหมาะสมกับงานที่ต้องการ (Configuration management) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

คุณภาพของกระบวนการ (quality of process)

1. Short release: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำเสนอผลการพัฒนาทีละน้อยอย่างต่อเนื่อง โดยพัฒนาเป็นวงรอบสั้นๆ เพื่อให้ผู้ที่เกี่ยวข้องเห็นภาพของระบบที่กำลังดำเนินการพัฒนา (Small Releases of Software Product) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

2. Metaphor: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบสร้างวิสัยทัศน์การออกแบบร่วมกันเพื่อให้ผู้ที่เกี่ยวข้องเข้าใจการทำงานของระบบ (System metaphor development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

3. Simple Design: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการออกแบบด้วยโครงสร้างง่ายๆ ไม่ซับซ้อนสามารถแก้ไข หรือเปลี่ยนแปลงง่าย และคล่องตัวขึ้น (Design is kept as simple as possible) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

4. Refactoring: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการปรับเปลี่ยนโครงสร้างของโค้ช (code) เพื่อให้มันสามารถจัดการได้ง่ายขึ้น และทำให้เราสามารถรับมือกับการเปลี่ยนแปลงได้ง่ายขึ้น มีความยืดหยุ่น และรองรับการเปลี่ยนแปลงได้ตลอดเวลา (Refactoring of code) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

5. Pair programming: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาโปรแกรมเป็นคู่สลับหน้าที่กันระหว่างการพัฒนาโปรแกรม และการตรวจสอบ เสมือนใช้คอมพิวเตอร์เครื่องเดียวกัน เพื่อให้ซอฟต์แวร์ที่พัฒนาขึ้นมีคุณภาพ เนื่องจากมีการตรวจสอบอยู่ตลอดเวลา (pair programming) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

6. Collective ownership: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการสร้างความมีส่วนร่วมในทุกส่วนของโปรแกรม ทีมพัฒนาทุกคนสามารถเข้าไปศึกษา หรือแม้แต่ปรับปรุงพัฒนาโค้ช

(Code) ได้ (Collective ownership of code) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

7. Continuous integration: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการรวมการทำงานของซอฟต์แวร์อย่างต่อเนื่อง เพื่อตรวจสอบความสามารถของการทำงานของซอฟต์แวร์ (Continue Integration) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

8. On-site customer: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการยินยอมให้ลูกค้าเข้ามามีส่วนเกี่ยวข้องกับการพัฒนาระบบ เช่น กระบวนการทดสอบ หรือการกำหนดผลยอมรับของการทดสอบ (Customer on-site) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

9. 40-h week: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบทำงาน 40 ชั่วโมงต่อสัปดาห์ (40 Hour week) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

10. Test first Development: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำการทดสอบและผลการทดสอบมาผลักดันโครงการ หรือ นำผลการทดสอบมาใช้ในการวางแผนการทำงานวงรอบถัดไป (Test first Development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

11. Coding standards: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำมาตรฐานการเขียนโปรแกรม เพื่อให้ทีมพัฒนามีความเข้าใจร่วมกันในการพัฒนาโปรแกรม (Coding to an agree standard) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

12. Scrum teams: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการระดมความคิดในเชิงการพัฒนาซอฟต์แวร์ เพื่อให้ผู้ที่เกี่ยวข้องมีส่วนร่วมในการทำงาน และเกิดการสื่อสารระหว่างทีม และเกิดการกระจายสามรู้ในทีม (Scrum Team) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

13. Product backlog: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำรายการความต้องการที่ถูกจัดแยกสิ่งที่ต้องพัฒนาในแต่ละวงรอบ (Product backlog) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

14. Sprint: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทำงานแต่ละวงรอบของทีม โดยยินยอมให้ผู้ที่มีส่วนเกี่ยวข้องทุกฝ่ายเข้ามามีส่วนร่วมสำหรับการพัฒนา พิจารณาความก้าวหน้าและจัดส่งซอฟต์แวร์ที่สามารถทำงานได้ในแต่ละวงรอบ เพื่อผลักดันให้เกิดการพัฒนาต่อไปได้อย่างรวดเร็ว (Sprint) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

15. Sprint review: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทบทวนการดำเนินงานในแต่ละวงรอบ (Sprint review) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

16. Individual class ownership: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการแบ่งระดับการดูแล และสิทธิในการเข้าถึงโค้ช (Code) เพื่อให้การพัฒนาเป็นไปอย่างคล่องตัวขึ้น สามารถเรียกใช้คลาส (Class) อย่างวดเร็ว (Individual class ownership) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

17. Feature teams: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดตั้งทีมที่มีขนาดเล็ก มีระดับความสามารถสูง ความคล่องตัวสูง และเต็มไปด้วยความคิดสร้างสรรค์ สมาชิกในทีมอาจจะต้องทำหน้าที่อื่นนอกเหนือจากทักษะที่ตนเองชำนาญ เช่น นักวิเคราะห์ระบบอาจจะต้องทำหน้าที่เป็นผู้ทดสอบ สนับสนุนการพัฒนาที่ขับเคลื่อนด้วยคุณลักษณะของซอฟต์แวร์ และการจัดการโครงการที่มีรายละเอียดค่อนข้างมากกว่าโครงการอื่นๆ ทีมจะมุ่งพัฒนาคุณลักษณะที่สามารถทำงานได้ภายในเวลาทุกๆ 2 สัปดาห์ เมื่อโครงการมีขนาดและความซับซ้อนมากขึ้น นักพัฒนา ผู้บริหารโครงการ และลูกค้าจำเป็นต้องเข้าใจสถานการณ์ของคุณลักษณะของซอฟต์แวร์ทั้งหมด (Feature teams) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

18. Developing by components: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาจากคอมโพเนนท์ (Component Development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

19. Software inspection: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการตรวจสอบข้อผิดพลาดของซอฟต์แวร์ โดยมุ่งไปที่ข้อผิดพลาดของการทำงานแต่ละส่วน (Software inspections) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

20. Iterative and incremental development: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาเป็นวงรอบ โดยอาศัยการเพิ่มเติมส่วนงานเข้าไปในงานเดิมเรื่อยๆ เพื่อให้มีความสมบูรณ์มากขึ้น (Iterative and incremental development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

21. Requirements are base lined at a high level: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดเตรียมรายการความต้องการที่มีความคงที่สูง (Requirements are baseline at a high level) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

22. The planning game: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบวางแผนร่วมกันระหว่างผู้ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ก่อนจะเริ่มการวนรอบ โดยมีการกำหนดรายละเอียด และขอบเขตการทำงานแต่ละวนรอบ (Planning Game) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

23. Scrum master: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบผู้ที่ทำหน้าที่ดูแลทีมงาน เป็นโค้ชของทีมงาน เป็นคนที่รับ ผิดชอบคุณภาพของผลงาน จัดลำดับความสำคัญของงาน (Scrum master) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

24. Sprint Planning meeting: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการประชุมวางแผนการวนรอบถึง เป้าหมาย และการพัฒนาเพิ่ม (Sprint Planning meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

25. Daily scrum meeting: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบจัดประชุมที่มทุกวันเพื่อติดตามความคืบหน้า และช่วยสำหรับการวางแผน และปรับปรุงกระบวนการพัฒนา (Daily scrum meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

26. Configuration management: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการบริหารจัดการเกี่ยวกับการกำหนดคุณสมบัติของคอมพิวเตอร์ อุปกรณ์หรือโปรแกรมใดๆ ที่จะนำมาใช้กับคอมพิวเตอร์ เพื่อให้ทำงานมีประสิทธิภาพ เหมาะสมกับงานที่ต้องการ (Configuration management) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

การทำงานของผู้ที่เกี่ยวข้อง (inter-supplier performance)

1. Short release: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำเสนอผลการพัฒนาทีละน้อยอย่างต่อเนื่อง โดยพัฒนาเป็นวงรอบสั้นๆ เพื่อให้ผู้ที่เกี่ยวข้องเห็นภาพของระบบที่กำลังดำเนินการพัฒนา (Small Releases of Software Product) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

2. Metaphor: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบสร้างวิสัยทัศน์การออกแบบร่วมกันเพื่อให้ผู้ที่เกี่ยวข้องเข้าใจการทำงานของระบบ (System metaphor development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

3. Simple Design: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการออกแบบด้วยโครงสร้างง่ายๆ ไม่ซับซ้อนสามารถแก้ไข หรือเปลี่ยนแปลงง่าย และคล่องตัวขึ้น (Design is kept as simple as possible) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

4. Refactoring: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการปรับเปลี่ยนโครงสร้างของโค้ช (code) เพื่อให้มันสามารถจัดการได้ง่ายขึ้น และทำให้เราสามารถรับมือกับการเปลี่ยนแปลงได้ง่ายขึ้น มีความยืดหยุ่น และรองรับการเปลี่ยนแปลงได้ตลอดเวลา (Refactoring of code) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

5. Pair programming: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาโปรแกรมเป็นคู่ สลับหน้าที่กันระหว่างการพัฒนาโปรแกรม และการตรวจสอบ เสมือนใช้คอมพิวเตอร์เครื่องเดียวกัน เพื่อให้ซอฟต์แวร์ที่พัฒนาขึ้นมีคุณภาพ เนื่องจากมีการตรวจสอบอยู่ตลอดเวลา (pair programming) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

6. Collective ownership: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการสร้างความมีส่วนร่วมในทุกส่วนของโปรแกรม ทีมพัฒนาทุกคนสามารถเข้าไปศึกษา หรือแม้แต่ปรับปรุงพัฒนาโค้ช (Code) ได้ (Collective ownership of code) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

7. Continuous integration: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการรวมการทำงานของซอฟต์แวร์อย่างต่อเนื่อง เพื่อตรวจสอบความสามารถของการทำงานของซอฟต์แวร์ (Continue Integration) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

8. On-site customer: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการยินยอมให้ลูกค้าเข้ามามีส่วนเกี่ยวข้องกับการพัฒนาระบบ เช่น กระบวนการทดสอบ หรือการกำหนดผลยอมรับของการทดสอบ (Customer on-site) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

9. 40-h week: 0 บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบทำงาน 40 ชั่วโมงต่อสัปดาห์ (40 Hour week) ไม่ส่งผลกระทบต่อประสิทธิภาพของการพัฒนาระบบ

10. Test first Development: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการนำการทดสอบและผลการทดสอบมาผลักดันโครงการ หรือนำผลการทดสอบมาใช้ในการวางแผนการทำงานวงรอบถัดไป (Test first Development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

11. Coding standards: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำมาตรฐานการเขียนโปรแกรม เพื่อให้ทีมพัฒนามีความเข้าใจร่วมกันในการพัฒนาโปรแกรม (Coding to an agree standard) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

12. Scrum teams: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการระดมความคิดในเชิงการพัฒนาซอฟต์แวร์ เพื่อให้ผู้ที่เกี่ยวข้องมีส่วนร่วมในการทำงาน และเกิดการสื่อสารระหว่างทีม และเกิดการกระจายสามรู้ในทีม (Scrum Team) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

13. Product backlog: +บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดทำรายการความต้องการที่ถูกจัดแยกสิ่งที่ต้องพัฒนาในแต่ละวงรอบ (Product backlog) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

14. Sprint: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทำงานแต่ละวงรอบของทีม โดยยินยอมให้ผู้ที่มีส่วนเกี่ยวข้องทุกฝ่ายเข้ามามีส่วนร่วมสำหรับการพัฒนา พิจารณาความก้าวหน้าและจัดส่งซอฟต์แวร์ที่สามารถทำงานได้ในแต่ละวงรอบ เพื่อผลักดันให้เกิดการพัฒนาต่อไปได้อย่างรวดเร็ว (Sprint) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

15. Sprint review: +++บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการทบทวนการดำเนินงานในแต่ละวงรอบ (Sprint review) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

16. Individual class ownership: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการแบ่งระดับการดูแล และสิทธิในการเข้าถึงโค้ช (Code) เพื่อให้การพัฒนาเป็นไปอย่างคล่องตัวขึ้น สามารถเรียกใช้คลาส (Class) อย่างวดเร็ว (Individual class ownership) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

17. Feature teams: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดตั้งทีมที่มีขนาดเล็ก มีระดับความสามารถสูง ความคล่องตัวสูง และเต็มไปด้วยความคิดสร้างสรรค์ สมาชิกในทีมอาจจะต้องทำหน้าที่อื่นนอกเหนือจากทักษะที่ตนเองชำนาญ เช่น นักวิเคราะห์ระบบอาจจะต้องทำหน้าที่เป็นผู้ทดสอบ สนับสนุนการพัฒนาที่ขับเคลื่อนด้วยคุณลักษณะของซอฟต์แวร์ และการจัดการโครงการที่มีรายละเอียดค่อนข้างมากกว่าโครงการอื่นๆ ทีมจะมุ่งพัฒนาคุณลักษณะที่สามารถทำงานได้ภายในเวลาทุกๆ 2 สัปดาห์ เมื่อโครงการมีขนาดและความซับซ้อนมากขึ้น นักพัฒนา ผู้บริหารโครงการ และลูกค้าจำเป็นต้อง เข้าใจสถานการณ์ของคุณลักษณะของซอฟต์แวร์ทั้งหมด (Feature teams) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

18. Developing by components: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาจากคอมโพเนนท์ (Component Development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

19. Software inspection: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการตรวจสอบข้อผิดพลาดของซอฟต์แวร์ โดยมุ่งไปที่ข้อผิดพลาดของการทำงานแต่ละส่วน (Software inspections) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

20. Iterative and incremental development: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการพัฒนาเป็นวงรอบ โดยอาศัยการเพิ่มเติมส่วนงานเข้าไปในงานเดิมเรื่อยๆ เพื่อให้มีความสมบูรณ์มากขึ้น (Iterative and incremental development) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

21. Requirements are base lined at a high level: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการจัดเตรียมรายการความต้องการที่มีความคงที่สูง (Requirements are baseline at a high level) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

22. The planning game: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบวางแผนร่วมกันระหว่างผู้ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ก่อนจะเริ่มการวนรอบ โดยมีการกำหนดรายละเอียด และขอบเขตการทำงานแต่ละวนรอบ (Planning Game) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

23. Scrum master: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบผู้ที่ทำหน้าที่ดูแลทีมงาน เป็นโค้ชของทีมงาน เป็นคนที่รับผิดชอบคุณภาพของผลงาน จัดลำดับความสำคัญของงาน (Scrum master) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

24. Sprint Planning meeting: ++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการประชุมวางแผนการวนรอบถึงเป้าหมาย และการพัฒนาเพิ่ม (Sprint Planning meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบอย่างมาก

25. Daily scrum meeting: +++ บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบจัดประชุมทีมทุกวันเพื่อติดตามความคืบหน้า และช่วยสำหรับการวางแผน และปรับปรุงกระบวนการพัฒนา (Daily scrum meeting) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบมากที่สุด

26. Configuration management: + บ่งชี้ว่าเทคนิคหรือวิธีปฏิบัติแบบการบริหารจัดการเกี่ยวกับการกำหนดคุณสมบัติของคอมพิวเตอร์ อุปกรณ์หรือโปรแกรมใดๆ ที่จะนำมาใช้กับคอมพิวเตอร์ เพื่อให้ทำงานมีประสิทธิภาพ เหมาะสมกับงานที่ต้องการ (Configuration management) ส่งผลกระทบต่อการปรับปรุงประสิทธิภาพของการพัฒนาระบบบางส่วน

ข้อเสนอแนะ

จากการวิจัยครั้งนี้ ผู้วิจัยได้พบข้อสังเกตบางประการ ซึ่งจะขอเสนอแนะดังต่อไปนี้

ข้อเสนอแนะในการนำไปใช้

1. การประยุกต์ใช้แนวปฏิบัติหลักให้สอดคล้องกับสิ่งแวดล้อม และทรัพยากรมีความสำคัญอย่างมากต่อความสำเร็จของโครงการในทุกด้าน

2. โครงการพัฒนาซอฟต์แวร์ที่มีโครงสร้างขององค์กรที่มีความซับซ้อน เช่น เกี่ยวข้องกับผู้ใช้จำนวนมาก ส่งผลให้การสื่อสารระหว่างผู้ที่เกี่ยวข้องสามารถทำได้ยาก ควรเข้าถึงข้อตกลงที่มีความชัดเจนสำหรับการพัฒนาซอฟต์แวร์

ข้อเสนอแนะในการวิจัยครั้งต่อไป

1. ควรควบคุมปัจจัยด้านสภาวะแวดล้อมขององค์กรให้มีความคล้ายคลึงกัน โดยเลือกองค์กรเดียวกันเพื่อสำหรับการเปรียบเทียบ

2. ควรเลือกโครงการพัฒนาซอฟต์แวร์ที่มีขนาดของโครงการพัฒนาซอฟต์แวร์ให้มีความใกล้เคียงกัน โดยอาจจะเลือกการพัฒนาแบบคู่ขนาดสำหรับโครงการพัฒนาซอฟต์แวร์แบบอาไจล์ (Agile) และ ไม่ใช่อาไจล์ (Non-Agile)

3. ควรศึกษาพฤติกรรมในช่วงเวลาการพัฒนาซอฟต์แวร์ โดยศึกษาความถี่ที่เกิดขึ้นจากแต่ละแนวปฏิบัติหลักที่เลือกใช้