ปัจจัยที่ส่งผลต่อความตั้งใจใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

วันที่ได้รับต้นฉบับ: 3 กันยายน 2560
วันที่แก้ไขต้นฉบับ: 13 มีนาคม 2561
วันที่ตอบรับบทความ: 20 เมษายน 2561

*สนิธพรรณ จิตตั้งสมบูรณ์**
*มฑุปายาส ทองมาก***

บทคัดย่อ

ธุรกิจในปัจจุบันมีการเปลี่ยนแปลงอย่างรวดเร็ว ทำให้กระบวนการพัฒนาและส่งมอบซอฟต์แวร์สำหรับธุรกิจนั้น จำเป็นต้องรวดเร็วและคล่องตัวตามไปด้วย ธุรกิจส่วนใหญ่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ในรูปแบบแอจไจล์มากขึ้น ซึ่งระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มุ่งเน้นกระบวนการเพื่อตอบสนองความต้องการของผู้ใช้เป็นหลัก แต่ไม่ได้ครอบคลุมถึงความต่อเนื่องและความเป็นอัตโนมัติด้านการส่งมอบระบบไปยังผู้ใช้งาน ในขณะที่แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ครอบคลุมตั้งแต่การพัฒนาซอฟต์แวร์จนกระทั่งกระบวนการในการส่งมอบซอฟต์แวร์ รวมถึงเน้นย้ำการประสานความสัมพันธ์ระหว่างผู้บริหารโครงการ ทีมพัฒนาซอฟต์แวร์ และทีมปฏิบัติการ แม้ว่าแนวคิด DevOps จะได้รับการสนับสนุนจากภาครัฐและเอกชน แต่แนวคิดยังไม่เป็นที่แพร่หลายในประเทศไทย

งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อศึกษาถึงปัจจัยที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps อันประกอบด้วยมิติด้านโครงการ (ปัจจัยลักษณะของโครงการแบบแอจไจล์) และมิติด้านเทคนิค (ปัจจัยการมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps) ที่ส่งผลต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยี รวมถึงมิติด้านองค์กร (ปัจจัยการสนับสนุนของผู้บริหารระดับสูง) มิติด้านบุคคล (ปัจจัยการรับรู้ความสามารถของตนเองและการรับรู้ประโยชน์) และมิติด้านกระบวนการ (ปัจจัยการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์) ผลการวิจัยพบว่า ปัจจัยการรับรู้ประโยชน์เป็นปัจจัยที่มีอิทธิพลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มากที่สุด รองลงมาคือปัจจัยการรับรู้ความสามารถของตนเอง ความเข้ากันได้ระหว่างงานกับเทคโนโลยี และปัจจัยการสนับสนุนของผู้บริหารระดับสูง ตามลำดับ แต่ปัจจัยการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ไม่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

คำสำคัญ: ระเบียบวิธีการพัฒนาซอฟต์แวร์ DevOps แอจไจล์ ปัจจัยความสำเร็จ ทฤษฎีความเข้ากันได้ระหว่างงานกับเทคโนโลยี ทฤษฎีการยอมรับเทคโนโลยี การรับรู้ความสามารถตนเอง

---

* นักศึกษาภาควิชาระบบสารสนเทศเพื่อการจัดการ คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์
** ผู้ช่วยศาสตราจารย์ประจำคณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์

Factors Affecting the Behavioral Intention to Use the DevOps Concept for Software Development

Received: September 3, 2017
Revised: March 13, 2018
Accepted: April 20, 2018

*Sanithpan Jittangsomboon**
*Mathupayas Thongmak***

Abstract

Nowadays, businesses are rapidly changing, so software development and delivery processes must be fast and flexible. Many businesses have adopted an Agile development methodology, which focuses on the process that meets the user requirements in order to supports the changing of business requirements. However, it does not cover the continuity and automation delivery of the software. Meanwhile, DevOps concept deals with processes start from software development to delivery process which emphasis on collaboration between product manager, software development team, and operations team. Although this concept is supported by public administration and private sectors, it still has yet to be fully explored in Thailand.

The purpose of this research was to study the factors affecting the behavioral intention to use the concept of DevOps software development. The research consists of project dimension (the characteristics of the Agile project factor) and technical dimension (the availability of technology tools that support the DevOps concept factor) that have an influence on the compatibility factor between work and technologies. In addition, organizational dimension (the executive support factor), people dimension (self-efficacy and perceived of usefulness factors) and process dimension (Agile software development usage factor). The results shown that the perceived of usefulness factor was the most influential intention to use DevOps concept. Followed by, self-efficacy factor, compatibility factor between work and technologies, and executive support factor respectively. On the other hand, Agile software development usage factor did not support the intention to use the DevOps concept.

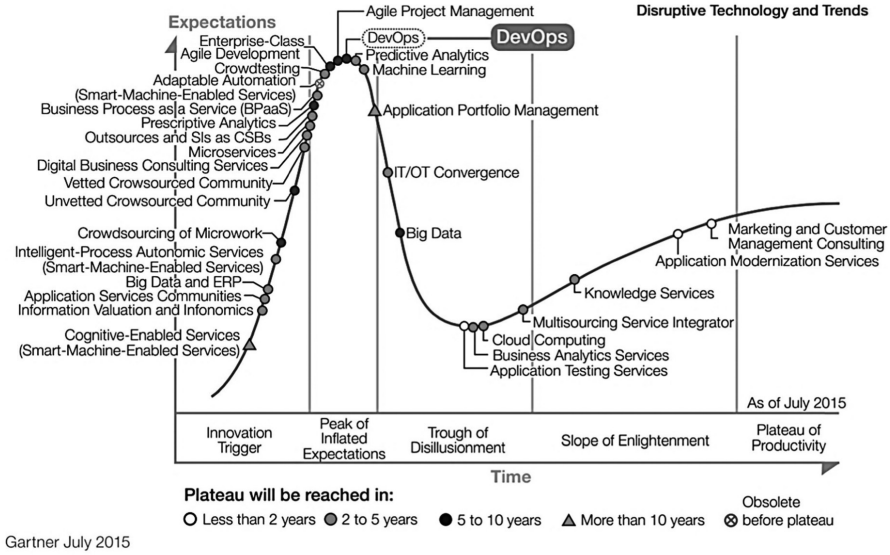
Keywords: Software Development Methodology, DevOps, Agile, Critical Success Factor, Task-Technology-Fit, Technology Acceptance Model, Self-efficacy

* Student, Department of Management Information Systems, Thammasat Business School, Thammasat University.

** Assistant Professor, Thammasat Business School, Thammasat University.

1. บทนำ

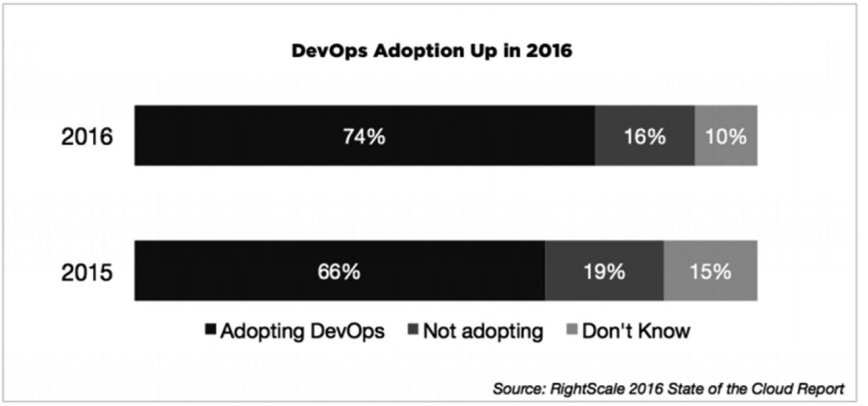
ปัจจุบันสภาพแวดล้อมของธุรกิจเปลี่ยนแปลงอย่างรวดเร็ว ทำให้ธุรกิจต้องเพิ่มความรวดเร็วในการส่งมอบคุณค่าให้แก่ลูกค้า ส่งผลต่อระเบียบวิธีการพัฒนาและกระบวนการส่งมอบซอฟต์แวร์ที่ต้องปรับตัวตาม เพื่อตอบสนองการเปลี่ยนแปลงของธุรกิจ แม้ว่าองค์กรต่างๆ จะเริ่มใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มากขึ้น ทำให้กระบวนการพัฒนาซอฟต์แวร์คล่องตัวและเป็นไปได้อย่างรวดเร็ว (ธนาชัย บูรณะวัฒนากูล, ศรี แก้วงาม, ปัญญา นราพันธ์, และ พระมหาอรรถพล วชิรปญฺโญ, 2555) แต่องค์กรธุรกิจยังคงประสบปัญหา เนื่องด้วยระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มุ่งเน้นกระบวนการเพื่อตอบสนองต่อการเปลี่ยนแปลงความต้องการ (Dingsøy & Lassenius, 2016) แต่ไม่ครอบคลุมความต่อเนื่องถึงการส่งมอบซอฟต์แวร์ไปยังผู้ใช้งาน (Deshpande, 2016) จึงทำให้เกิดปัญหาคอขวดในกระบวนการส่งมอบซอฟต์แวร์ (Michelsen, 2013) นอกจากนี้ยังเกิดปัญหาการทำงานร่วมกันระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการด้วย (Airaj, 2015) ปัจจุบันจึงมีการนำเสนอแนวคิดการใช้เครื่องมือเพื่อสร้างความต่อเนื่องในการพัฒนาซอฟต์แวร์ทำให้มีความอัตโนมัติมากขึ้น โดยเครื่องมือจะเข้ามาช่วยการทำงานระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการปรับปรุงกระบวนการทำงานระหว่างกันให้ราบรื่นเรียกว่า “DevOps” (Mohamed, 2016)



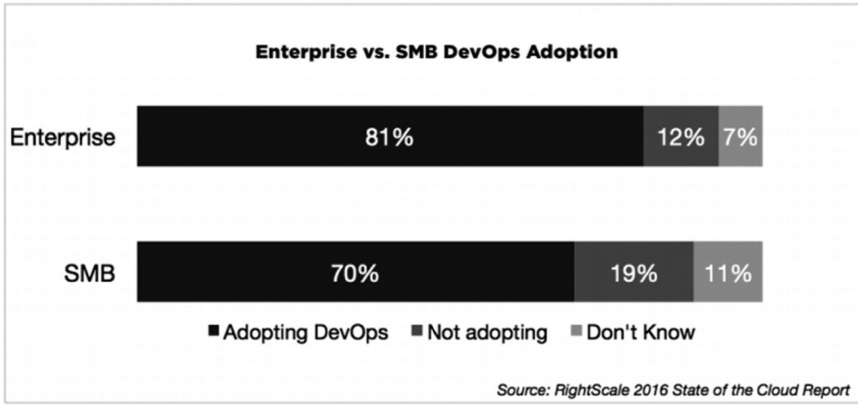
ภาพที่ 1 แนวโน้มเทคโนโลยีของการ์ทเนอร์ ปี ค.ศ. 2015. ดัดแปลงจาก “DevOps - The Future of Application Lifecycle Automation,” A Capgemini Architecture Whitepaper – 2nd Edition, 22. โดย Menzel และ Macaulay (2014)

บทความของการ์ทเนอร์ ดังแสดงในภาพที่ 1 ได้วิเคราะห์ถึงแนวคิดของ DevOps (Menzel & Macaulay, 2014) ว่าเป็นการพัฒนาซอฟต์แวร์ที่เน้นทั้งวงชีพของการพัฒนาโปรแกรมประยุกต์ (Application Life Cycle) (Patwardhan, 2014) ซึ่งถูกพัฒนามาจากระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (Kruis, 2014) ทำให้สามารถดำเนินการพัฒนาซอฟต์แวร์ได้อย่างครบวงจร โดยอาศัยเครื่องมือและเทคโนโลยีที่มีประสิทธิภาพ ทั้งนี้แนวคิดของ DevOps อาศัยการผสมผสานระหว่างบุคคล กระบวนการ และเทคโนโลยี เป็นตัวกระตุ้นการทำงานร่วมกับการสร้างสรรค์นวัตกรรม (Olszewska & Waldén, 2015) เพื่อนำไปสู่การพัฒนาซอฟต์แวร์และกระบวนการดำเนินงานที่มีประสิทธิภาพ (Michelsen, 2013)

แนวคิด DevOps ได้รับความนิยมในต่างประเทศเป็นอย่างมากและมีอัตราการยอมรับการใช้เพิ่มขึ้นอย่างต่อเนื่อง โดยผลสำรวจล่าสุดของบทความ Right Scale ซึ่งมีจำนวนผู้ตอบแบบสอบถาม 1,060 คน ประกอบด้วย ทวีปอเมริกาเหนือ ยุโรป เอเชีย และอื่นๆ พบว่า ในปี ค.ศ. 2016 มีอัตราการใช้งานแนวคิด DevOps เพิ่มขึ้นจากปี ค.ศ. 2015 เป็นร้อยละ 74 ดังภาพที่ 2 จำแนกเป็นระดับวิสาหกิจ (Enterprise) ร้อยละ 81 และธุรกิจขนาดกลางและขนาดเล็ก (SMB) ร้อยละ 70 ดังภาพที่ 3 (Right Scale, 2016) นอกจากนี้บริษัทข้ามชาติที่มีชื่อเสียงอย่าง กูเกิ้ล (google) อะเมซอน (amazon) ทวิตเตอร์ (twitter) ได้ประสบความสำเร็จในการนำแนวคิด DevOps เข้าไปใช้งาน ซึ่งสามารถปรับปรุงซอฟต์แวร์ได้อย่างต่อเนื่องและมีประสิทธิภาพ (ปริวรรต ศักดิ์นิมิตวงศ์, 2560) ดังภาพที่ 4



ภาพที่ 2 อัตราการใช้แนวคิด DevOps ที่เพิ่มขึ้นจากการสำรวจของบทความ Right Scale ระหว่างปี ค.ศ. 2015 กับ 2016 จาก “Right Scale 2016 STATE OF THE CLOUD REPORT: DevOps Trends” http://assets.rightscale.com/uploads/pdfs/rightscale-2016-state-of-the-cloud-report-devops-trends.pdf โดย Right Scale (2016)



ภาพที่ 3 ลักษณะองค์กรที่ปรับนำแนวคิด DevOps เข้ามาใช้งานในปี ค.ศ. 2016 จาก "Right Scale 2016 STATE OF THE CLOUD REPORT: DevOps Trends" http://assets.rightscale.com/uploads/pdfs/rightscale-2016-state-of-the-cloud-report-devops-trends.pdf โดย Right Scale (2016)

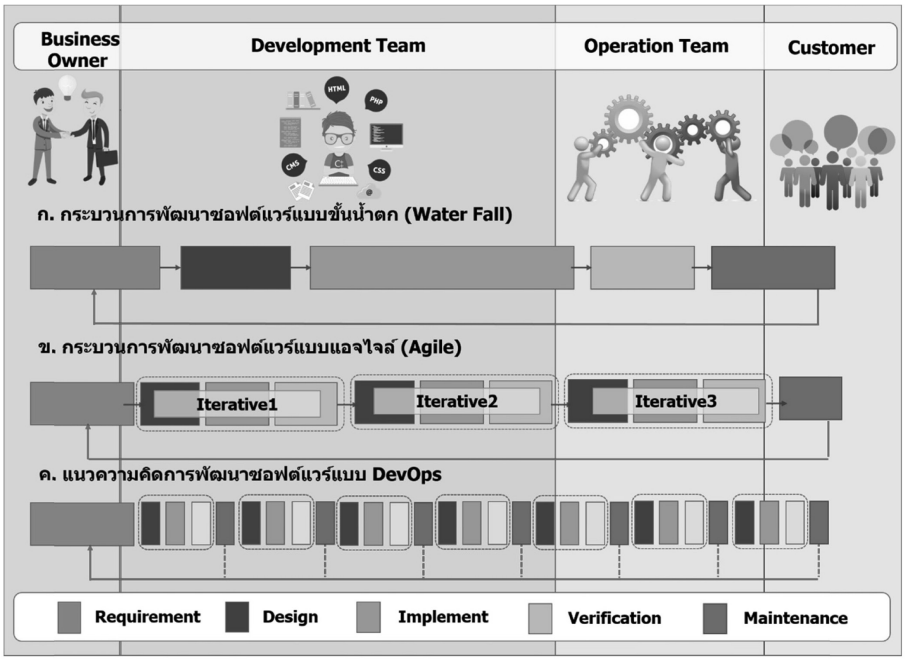
Company	Deploy Frequency	Deploy Lead Time	Reliability	Customer Responsiveness
Amazon	23,000 / day	Minutes	High	High
Google	5,500 / day	Minutes	High	High
Netflix	500 / day	Minutes	High	High
Facebook	1 / day	Hours	High	High
Twitter	3 / week	Hours	High	High
Typical Enterprise	Once every 9 months	Month or Quarters	Low/Med	Low/Med

ภาพที่ 4 อัตราการใช้งานแนวคิด DevOps ของแต่ละบริษัท ซึ่งส่งผลถึงความน่าเชื่อถือและความพึงพอใจของลูกค้า จาก "Learn DevOps ตอนที่ 2 : DevOps คืออะไร ?" https://medium.com/@pariwat_s/learn-devops-ตอนที่-2 - DevOps-คืออะไร-18ac48d73625 โดย ปริวรรต ศักดิ์นิมิตวงศ์ (2560)

สำหรับประเทศไทยได้ให้ความสนใจเกี่ยวกับแนวคิด DevOps เพิ่มขึ้น โดยสำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ (SIPA) ได้จัดอบรมทั้งเชิงทฤษฎีและเชิงปฏิบัติเกี่ยวกับแนวคิด DevOps เพื่อแบ่งปันประสบการณ์จากผู้เชี่ยวชาญ และสร้างความรู้ความเข้าใจเกี่ยวกับแนวคิดดังกล่าว (สำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ, 2559) นอกจากนี้สถาบันอบรมเกี่ยวกับเทคโนโลยี เช่น สถาบันไอเอ็มซี (IMC Institute) ซึ่งเป็นสถาบันวิจัยและสำรวจข้อมูลด้านเทคโนโลยีสารสนเทศต่างๆ ในประเทศไทย และเขตอุตสาหกรรมซอฟต์แวร์ประเทศไทย (Software Park Thailand) ได้จัดอบรมแนวคิดและวิธีปฏิบัติเกี่ยวกับแนวคิด DevOps ขึ้นโดยมีหลักสูตรอบรมอย่างต่อเนื่องตั้งแต่ปี พ.ศ. 2560-2561 (กชพรรณ ออกดีเลิศ, 2560) อีกทั้งรัฐบาลยังส่งเสริมแนวคิด DevOps โดยแผนแม่บทเทคโนโลยีสารสนเทศและการสื่อสารของกรมจัดหางาน พ.ศ. 2558 – 2562 ได้กำหนดแนวโน้มของเทคโนโลยีสารสนเทศในหัวข้อ Web IT Scale สำหรับวิสาหกิจว่า ต้องปรับตัวเพื่อรองรับผู้ใช้งานที่เพิ่มขึ้นอย่างรวดเร็ว โดยให้นำแนวคิด DevOps เข้ามาใช้งานมากขึ้น (กรมการจัดหางาน, 2558) อย่างไรก็ดีแม้ว่าองค์กรในประเทศไทยจะหันมาใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มากขึ้น และบางองค์กรนำแนวคิด DevOps ไปใช้งานบ้างแล้ว แต่แนวคิดดังกล่าวยังไม่เป็นที่แพร่หลาย (คัมภีร์เทพ IT, 2560) ในปัจจุบันองค์กรที่ใช้งานแนวคิด DevOps อยู่ เช่น Agoda เป็นบริษัทผู้ให้บริการเว็บไซต์จองห้องพักที่ได้รับความนิยม Ascend เป็นบริษัทพัฒนาซอฟต์แวร์เกี่ยวกับระบบ E-commerce ภายในเครือเจริญโภคภัณฑ์ (Sukoom2001, 2560) และ Opsta (Thailand) Co., Ltd. เป็นบริษัทให้คำปรึกษาทางด้านซอฟต์แวร์ระบบเปิด (Open Source Consultant) (Winggundamth, 2017) เป็นต้น จากข้อมูลที่กล่าวมาในข้างต้นว่าแนวคิด DevOps ได้ถูกนำไปใช้งานในต่างประเทศจนประสบความสำเร็จ และในประเทศไทยได้ส่งเสริมแนวคิดดังกล่าวจากทั้งภาครัฐและเอกชน รวมถึงองค์กรบางแห่งได้เริ่มนำแนวคิดนี้ไปประยุกต์ใช้งาน จึงเป็นที่น่าสนใจและนำมาสู่การศึกษาถึงปัจจัยที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ในงานวิจัยนี้

2. การทบทวนวรรณกรรมและทฤษฎีที่เกี่ยวข้อง

2.1 แนวคิดการพัฒนาซอฟต์แวร์



ภาพที่ 5 การเปรียบเทียบระหว่างวงจรการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall Model) ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ และแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ดัดแปลงจาก "From Waterfall to Agile to DevOps – A Cultural and Technological Shift," http://blog.spec-india.com/from-waterfall-to-agile-to-devops-a-cultural-and-technological-shift/ โดย Spec-India (2015)

2.1.1 กระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก

กระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall Model) เป็นที่รู้จักกันโดยทั่วไปในการพัฒนาซอฟต์แวร์ ประกอบด้วยกิจกรรมพื้นฐานของกระบวนการพัฒนาซอฟต์แวร์ 5 ขั้นตอนพื้นฐาน ดังนี้ 1) ขั้นเก็บรวบรวมและวิเคราะห์ความต้องการ (Requirement) ซึ่งจะช่วยให้เข้าใจปัญหามากขึ้น 2) ขั้นออกแบบ (Design) เป็นขั้นตอนการวางแผนเพื่อแก้ไขปัญหา 3) ขั้นพัฒนาซอฟต์แวร์ (Implement) เป็นการพัฒนาซอฟต์แวร์ด้วยการเขียนโปรแกรม 4) ขั้นตอนทดสอบโปรแกรม (Verification) เป็นการทดสอบผลลัพธ์ที่ได้จากการเขียนโปรแกรม และ 5) ขั้นบำรุงรักษา (Maintenance) เป็นขั้นตอนการใช้

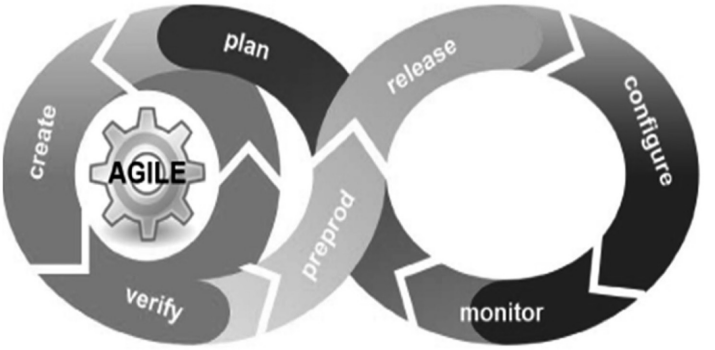
งานจริงและบำรุงรักษาซอฟต์แวร์ ดังภาพที่ 5 ก. ด้วยลักษณะของแนวคิดแบบขั้นน้ำตกที่เป็นลำดับขั้นตอน ทำให้ง่ายต่อการวางแผน แต่ยากต่อการเปลี่ยนแปลง รวมทั้งยังส่งผลถึงความเสี่ยงต่อการล้มเหลวของโครงการ (Cois, Yankel, & Connell, 2014) จึงเกิดกระบวนการพัฒนาซอฟต์แวร์รูปแบบใหม่ขึ้นมาเพื่อปรับปรุงข้อเสียดังกล่าว

2.1.2 ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์

เป็นระเบียบวิธีการพัฒนาซอฟต์แวร์ที่แก้ไขข้อบกพร่องของกระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก โดยนำกระบวนงานมาแบ่งเป็นส่วนเล็กเพื่อการทำรอบซ้ำๆ (Iterative) และเพิ่มจำนวนการพัฒนาแต่ละรอบให้มากขึ้น (Incremental Development) ดังภาพที่ 5 ข. ที่เน้นถึงการติดต่อสื่อสารระหว่างผู้มีส่วนได้ส่วนเสีย (Stakeholder) การปรับปรุงความต้องการและวางแผนงาน (Plan) เพื่อให้สามารถส่งมอบซอฟต์แวร์ไปยังลูกค้าให้เร็วที่สุด (Schaller, 2016) โดยผู้ร่วมโครงการเชื่อว่ากระบวนการออกแบบต้องพร้อมที่จะปรับปรุงอยู่เสมอ ด้วยข้อมูลใหม่ที่มีประสิทธิภาพและช่วยลดความเสี่ยงในโครงการที่จะเกิดขึ้น รวมถึงเพิ่มความสำคัญของผลลัพธ์ที่จะได้จากโครงการ (Cois et al., 2014) ถึงแม้ว่าระเบียบวิธีการพัฒนาซอฟต์แวร์แอจไจล์จะช่วยให้การส่งมอบซอฟต์แวร์ไปสู่ลูกค้าได้เร็วขึ้น แต่ก็ยังพบข้อจำกัดในกระบวนการบางอย่างที่ไม่สอดคล้องกับรูปแบบธุรกิจ เช่น การติดตามปัญหาและแก้ปัญหาที่ผู้ใช้รายงานมา การเพิ่มความสามารถของซอฟต์แวร์จากฝ่ายบริหาร การส่งมอบที่ต้องปรับแต่งตลอดเวลา ซึ่งข้อจำกัดเหล่านี้ยังขาดกระบวนการที่ดีและความน่าเชื่อถือในการติดตามงาน (พงศกร ภูแสนคำ, 2559)

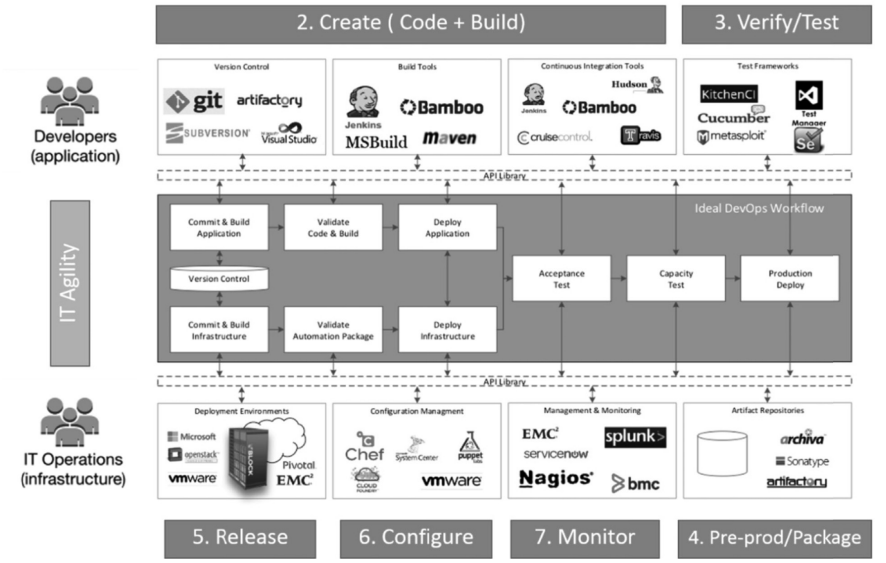
2.1.3 แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

แนวคิด DevOps อยู่บนพื้นฐานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ โดยช่วยเติมเต็มช่องว่างของระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ ที่มุ่งเน้นเพียงด้านธุรกิจ ด้วยการประสานความสัมพันธ์ระหว่างผู้จัดการโครงการ (Project Manager) และนักพัฒนาซอฟต์แวร์ (Developer) โดยแนวคิด DevOps จะช่วยให้ทีมปฏิบัติการ (Operation team) สามารถส่งมอบคุณค่าของซอฟต์แวร์ไปยังผู้ใช้งาน (User) ได้ต่อเนื่องมากขึ้น ดังภาพที่ 5 ค. (Mueller, 2016; Deshpande, 2016; Mohamed, 2016) สาระสำคัญเบื้องหลังแนวคิด DevOps (DevOps Principle) คือ การสร้างความร่วมมือที่ดีระหว่างทีมนักพัฒนาซอฟต์แวร์และทีมปฏิบัติการ เพื่อให้แต่ละทีมสามารถดำเนินงานร่วมกันได้อย่างราบรื่น ทำให้ระเบียบวิธีการพัฒนาซอฟต์แวร์เกิดความต่อเนื่องมากขึ้นจากความเป็นอัตโนมัติ ส่งผลให้ซอฟต์แวร์มีคุณภาพ และรองรับการเปลี่ยนแปลงอย่างรวดเร็วตามความต้องการ ก่อให้เกิดความน่าเชื่อถือในการส่งมอบซอฟต์แวร์ (Babar, Lapouchnian, & Yu, 2011) และยังครอบคลุมไปถึงการเตรียมซอฟต์แวร์เพื่อนำไปสู่การใช้งานจริง (Deployment) (Fitzgerald & Stol, 2014) กิจกรรมที่เกิดขึ้นตามแนวคิด DevOps จะมีลักษณะสอดประสานอย่างต่อเนื่องเป็นลูป (Loop) ดังแสดงในภาพที่ 6



ภาพที่ 6 แนวคิดการพัฒนาซอฟต์แวร์แบบ จาก "How Continuous Requirements Are Key to Gartner's DevOps Toolchain." Http: //www.blueprintsys.com/blueprint-devops โดย Higgins (2016)

นอกจากนี้ชุดเครื่องมือ ดังแสดงในภาพที่ 7 นับเป็นสิ่งสำคัญสำหรับแนวคิด DevOps ชุดเครื่องมือที่ใช้จัดการกระบวนการต่างๆ มีความครอบคลุม ตั้งแต่ขั้นตอนการพัฒนาซอฟต์แวร์ จนถึงส่งมอบซอฟต์แวร์ ตลอดจนติดตามผลลัพธ์ที่เกิดขึ้น (Gartner, 2015) แต่ละกระบวนการจะมีเครื่องมือที่รองรับแตกต่างกัน ทำให้สามารถตอบสนองความต้องการได้อย่างครบถ้วน (Gartner, 2015; Higgins, 2016; Menzel & Macaulay, 2014)



ภาพที่ 7 เครื่องมือสนับสนุนการดำเนินงานแต่ละขั้นตอน ดัดแปลงจาก "Common DevOps Tool Chains Pitfalls," https://infocus.emc.com/bart_driscoll/common-devops-tool-chains-pitfalls โดย Driscoll (2015)

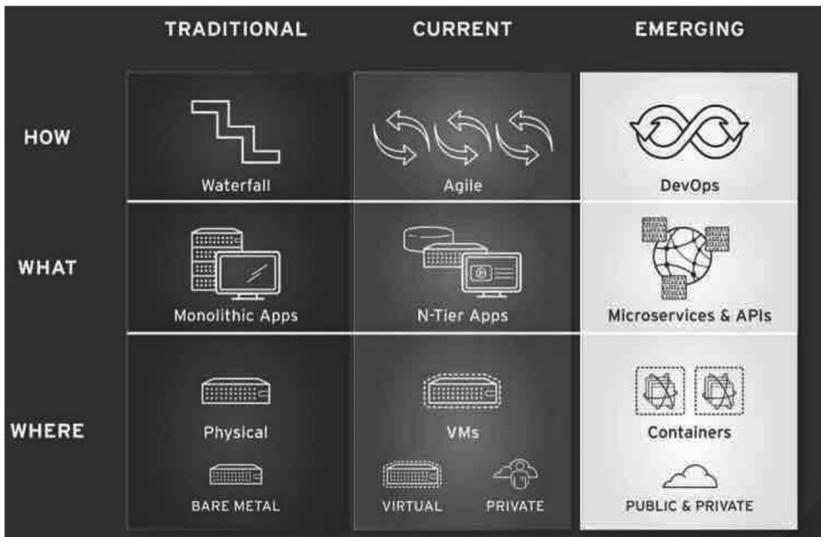
นอกจากนี้ DuMoulin (2017) ได้นำเสนออัตราการใช้กระบวนการพัฒนาซอฟต์แวร์แต่ละรูปแบบในงานวิจัย เกี่ยวกับปัจจัยความสำเร็จในการใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ของบริษัท Pink Think Tank สำรวจการใช้กรอบวิธีปฏิบัติในงานต่างๆ จาก 160 องค์กร ที่มีรูปแบบของธุรกิจที่แตกต่างกันเปรียบเทียบระหว่างปี ค.ศ. 2016 กับ 2017 โดยผลการสำรวจจากกลุ่มตัวอย่างพบว่ามีการใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มีอัตราเพิ่มขึ้น 7% เมื่อเทียบกับระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์เพิ่มขึ้นเพียง 1% เท่านั้น นอกจากนี้กระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตกมีการใช้อย่างลดลงถึง 12% ดังตารางที่ 1

ตารางที่ 1 การเปรียบเทียบการกรอบวิธีปฏิบัติงานของกลุ่มตัวอย่างระหว่างปี ค.ศ. 2016-2017 จากงานวิจัยของบริษัท Pink Think Tank

วิธีการที่ใช้	สำรวจปี ค.ศ. 2016	สำรวจปี ค.ศ. 2017	ผลต่าง
IT Service Management	27%	28%	เพิ่มขึ้น 1%
Waterfall Project Management	18%	6%	ลดลง 12%
Agile Development	22%	23%	เพิ่มขึ้น 1%
Six Sigma	7%	4%	ลดลง 3%
DevOps	13%	20%	เพิ่มขึ้น 7%
Lean IT	13%	19%	เพิ่มขึ้น 6%

หมายเหตุ. จาก “Critical Success Factors For DevOps” โดย DuMoulin (2017)

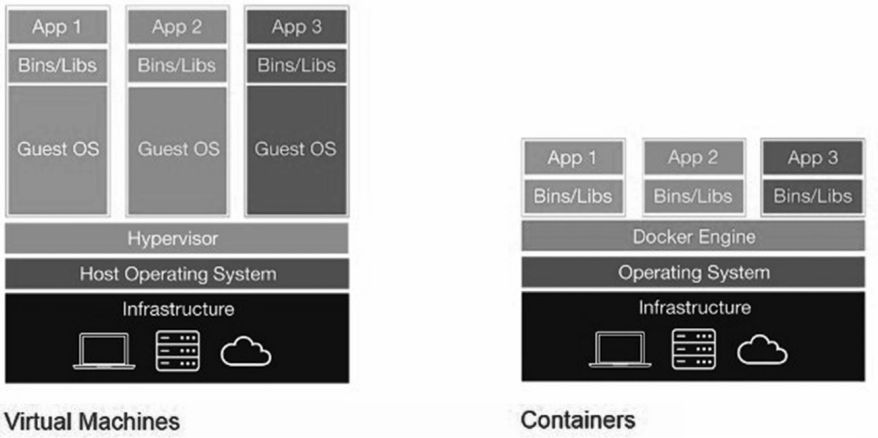
2.1.4 ความแตกต่างระหว่างกระบวนการพัฒนาซอฟต์แวร์แต่ละรูปแบบ



ภาพที่ 8 ลักษณะการทำงานและการใช้งานของกระบวนการพัฒนาซอฟต์แวร์ที่แตกต่างกัน จาก “emerginttechtrends” https://www.extremeuncertainty.com/emergint-techtrends/ โดย Tranter (2016)

จากกระบวนการพัฒนาซอฟต์แวร์ทั้งหมดที่กล่าวมาข้างต้น ผู้วิจัยได้สรุปโครงสร้างสถาปัตยกรรมทางซอฟต์แวร์และรูปแบบการทำงานของเซิร์ฟเวอร์ที่เหมาะสมต่อกระบวนการพัฒนาซอฟต์แวร์แต่ละรูปแบบไว้ดังภาพที่ 8 ด้านสถาปัตยกรรมซอฟต์แวร์ของกระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตกจะเหมาะสมกับการออกแบบโปรแกรมที่ทำงานแบบรวมกับทุกฟังก์ชันมาไว้ที่เดียว (Monolithic Architecture) ทำให้สะดวกต่อการทำงาน แต่หากเกิดการแก้ไขเปลี่ยนแปลงก็จำเป็นต้องบิ้ว (Build) และส่งมอบโปรแกรมใหม่ทั้งหมด จะส่งผลต่อการปรับเพิ่มลด (Scale) ในอนาคต ส่วนของระเบียบวิจัยแบบแอจไจล์จะรองรับสถาปัตยกรรมแบบระดับชั้น (Tier Architecture) โดยแยกการทำงานของแต่ละองค์ประกอบอย่างชัดเจน อาทิ แบบสามระดับชั้น (3- tiers) ประกอบด้วยส่วนหน้าจอติดต่อกับผู้ใช้งาน (Present Tier) ส่วนการประมวลผล (Application Tier) และส่วนฐานข้อมูล (Data Tier) เป็นต้น (Techtalkthai, 2559) ทำให้ง่ายต่อการปรับปรุงและบำรุงรักษา นอกจากนี้ยังสามารถนำองค์ประกอบอื่นๆ ไปใช้ร่วมกับโปรแกรมได้อีก (Krome, 2017) และสถาปัตยกรรมซอฟต์แวร์ของแนวคิด DevOps นั้นจะสนับสนุนการทำงานรูปแบบไมโครเซอร์วิส (Micro Service) และเอพีไอ (API) โดยแต่ละเซอร์วิสจะมีขนาดเล็ก (Atomic) ที่มีอิสระต่อกัน ทั้งนี้แต่ละฟังก์ชันจะประกอบด้วยเซอร์วิสขนาดเล็กๆ รวมกัน (Meresca, 2015) โดยจำแนกตามความต้องการของผู้ใช้งาน (Requirement/Business Capability) ซึ่งแตกต่างจากระดับชั้นที่จำแนกตามหน้าที่ในแต่ละระดับชั้น (Technical Responsibility) (Techtalkthai, 2559)

ส่วนรูปแบบการทำงานของเซิร์ฟเวอร์ของแต่ละกระบวนการพัฒนาซอฟต์แวร์นั้น โดยในยุคแรกของการพัฒนาซอฟต์แวร์แบบขั้นน้ำตกใช้เซิร์ฟเวอร์เชิงกายภาพ (Physical Server) ผู้ใช้จะต้องติดตั้งระบบปฏิบัติการของเซิร์ฟเวอร์และดูแลรักษาด้วยตนเอง ทำให้เกิดความยุ่งยากในการบริหารจัดการและมีค่าใช้จ่ายสูง (Rosehosting, 2016) ส่วนของระเบียบวิจัยแอจแบบแอจไจล์จะอยู่ในยุคของการใช้เครื่องเสมือน (Virtual Machine) ที่ช่วยในการบริหารจัดการทรัพยากรที่มีอยู่บนเซิร์ฟเวอร์ให้มีหลายเครื่อง แต่ทำงานอยู่ภายใต้เครื่องเดียวกัน ซึ่งจะมีความแตกต่างของทรัพยากร (Resource) ประสิทธิภาพและระบบปฏิบัติการ (Operating System) ที่ใช้งานดังภาพที่ 9 (ซ้ายมือ) และเริ่มหันมาใช้บริการกลุ่มเมฆแบบส่วนตัว (Private Cloud) โดยจะบริหารจัดการภายในองค์กรเอง เนื่องจากความกังวลด้านความปลอดภัย ทั้งนี้จะช่วยลดความยุ่งยากในการดูแลรักษาลงไป (White, 2016) และแนวคิด DevOps จะใช้คอนเทนเนอร์ (Container) เป็นการจำลองเพื่อควบคุมสภาพแวดล้อมสำหรับการทำงานเฉพาะเซอร์วิส โดยจะบรรจุสภาพแวดล้อมนั้นลงในคอนเทนเนอร์ ก่อให้เกิดการใช้ทรัพยากรเท่าที่จำเป็น การทำงานจะอยู่ในระดับระบบปฏิบัติการเท่านั้นดังภาพที่ 9 (ขวามือ) แตกต่างจากการจำลองเซิร์ฟเวอร์เสมือนที่ใช้สภาพแวดล้อมในระบบปฏิบัติการทั้งหมดเพื่อสร้างเซิร์ฟเวอร์หนึ่งเครื่องที่มีหลายๆ เซอร์วิสทำงานร่วมกัน ทั้งนี้คอนเทนเนอร์จะมีด๊อกเกอร์ (Dockers) ถือว่าเป็นเครื่องมือที่ได้รับความนิยมในการเข้ามาจัดการคอนเทนเนอร์ ทำให้สามารถใช้งานได้ง่ายและยังสนับสนุนการใช้เทคโนโลยีกลุ่มเมฆทั้งแบบส่วนตัวและสาธารณะ โดยแบบสาธารณะนั้นองค์กรจะไม่สามารถบริหารจัดการได้เองทั้งหมด แต่จะมีผู้ให้บริการเป็นคนบริหารจัดการให้ (White, 2016)



ภาพที่ 9 ลักษณะการทำงานของเครื่องเสมือน (Virtual Machine) กับคอนเทนเนอร์ (containers) จาก "ทำความรู้จัก Docker และ Software Container" โดย พัฒนพงษ์ เชิดทอง (2559)

นอกจากนี้ Kapadia (2015) ได้กล่าวว่าวงจรการพัฒนาซอฟต์แวร์ที่สามารถสร้างมูลค่า (Value) มีสองขั้นตอน ขั้นแรกคือ การพัฒนาซอฟต์แวร์ และขั้นที่สองคือ การส่งมอบซอฟต์แวร์ไปยังผู้ใช้งานจนกระทั่งได้ผลตอบรับกลับมา การส่งมอบซอฟต์แวร์ไปยังผู้ใช้งานได้อย่างรวดเร็วจะส่งผลต่อการรับรู้ถึงคุณค่ามากยิ่งขึ้น แต่ในการส่งมอบซอฟต์แวร์ยังมีกิจกรรมที่เกิดขึ้นอีก อาทิ การทดสอบ (Testing) การตรวจสอบคุณภาพ (Quality Assurance) และการเตรียมจากสภาพแวดล้อมของการทดสอบไปยังสภาพแวดล้อมการใช้งานจริง (Staging) เป็นต้น แม้ว่ากระบวนการเหล่านี้จะไม่สร้างมูลค่าให้เกิดขึ้น แต่กระบวนการดังกล่าวช่วยลดความเสี่ยงของข้อผิดพลาดที่จะเกิดขึ้นก่อนที่ซอฟต์แวร์จะส่งไปยังผู้ใช้งาน ทั้งนี้องค์กรที่นำแนวคิด DevOps ไปใช้งานเข้าใจถึงคุณค่าที่เกิดขึ้น โดยมุ่งเน้นไปยังขั้นตอนดังกล่าวให้ดีที่สุดและลดความเสี่ยงจะเกิดขึ้นให้น้อยที่สุด โดยได้สรุปเปรียบเทียบกระบวนการพัฒนาซอฟต์แวร์แบบดั้งเดิมกับแนวคิด DevOps ด้วยมิติต่างๆ ประกอบด้วย ด้านการวางแผนและลักษณะขององค์กร ได้แก่ ขนาดของโครงการ (Batch Size) องค์กร (Organization) และตารางการทำงาน (Scheduling) ด้านประสิทธิภาพและวัฒนธรรมการทำงาน ได้แก่ การส่งมอบซอฟต์แวร์ (Release) ข้อมูล (Information) และวัฒนธรรม (Culture) และด้านการประเมินผล ได้แก่ มาตรวัด (Measure) และตัวกำหนดความสำเร็จ (Define "done") ดังตารางที่ 2

ตารางที่ 2 การเปรียบเทียบความแตกต่างระหว่างกระบวนการพัฒนาซอฟต์แวร์แบบดั้งเดิมกับแนวคิด DevOps

	มิติที่วัดผล	การพัฒนาซอฟต์แวร์แบบดั้งเดิม (Traditional IT)	แนวคิด DevOps
การวางแผนและลักษณะขององค์กร	ขนาดโครงการ	**โครงการขนาดใหญ่ (Big):** โครงการขนาดใหญ่จะมีโค้ดจำนวนมาก ทำให้ใช้ระยะเวลาดำเนินการนาน ส่งผลต่อค่าใช้จ่ายที่สูงขึ้นและยากต่อการควบคุม รวมถึงปัญหาระหว่างทีมปฏิบัติการที่อนุญาตให้ทีมพัฒนาซอฟต์แวร์เปลี่ยนแปลงซอฟต์แวร์ได้จำนวนน้อยครั้งต่อปี เนื่องจากกังวลต่อการเปลี่ยนแปลงที่เกิดขึ้นว่าจะส่งผลต่อเสถียรภาพของระบบ	**หน่วยเล็กขนาดไมโคร (Micro):** โครงการขนาดเล็ก ทำให้ง่ายต่อการเข้าใจ การทดสอบและความเสี่ยงน้อยกว่า หากเกิดข้อผิดพลาดก็จะส่งผลกระทบเพียงเล็กน้อย ง่ายและรวดเร็วต่อการแก้ไขปัญหา นอกจากนี้ยังทำให้องค์กรขนาดเล็กสามารถส่งมอบซอฟต์แวร์ได้ถี่มากขึ้น ซึ่งสามารถตอบสนองต่อความต้องการของผู้ใช้ได้ดียิ่งขึ้น
	องค์กร	**ความสามารถและความชำนาญ ยึดติดตามหน่วยงาน (Skill Centric Silos) :** โดยส่วนใหญ่ทีมเดียวกันจะใช้ทักษะที่เหมือนกันเพื่อให้เกิดประโยชน์สูงสุด ส่งผลต่อการสร้างค่าใช้จ่ายที่สูงขึ้น เนื่องจากฝ่ายไอทีจะประกอบด้วยอีก 3-4 ทีมย่อย ซึ่งจะใช้เวลาในการติดต่อสื่อสารระหว่างทีมมากถึง 80 %	**แตกย่อยเป็นขนาดเล็ก (Dedicated Cells):** หน่วยเล็กๆ ที่ทำงานร่วมกันและมีเป้าหมายเดียวกันคือ แอปพลิเคชัน โดยสร้างจากความสามารถของทีมพัฒนาซอฟต์แวร์ ทดสอบระบบ วิเคราะห์ธุรกิจ และดำเนินการส่งมอบซอฟต์แวร์ ซึ่งแนวคิด DevOps นี้ก่อให้เกิดความต่อเนื่องในแต่ละขั้นตอนได้โดยปราศจากการปัดความรับผิดชอบ แต่จะสนับสนุนให้เกิดการอบรมและสร้างความเข้าใจในการทำงานร่วมกัน ส่งผลให้เกิดประสิทธิภาพในการดำเนินงาน
	ตารางการทำงาน	**รวมศูนย์ขึ้นกับหน่วยงานรับผิดชอบ (Centralized):** ประสิทธิภาพในการจัดการตารางเวลาถือเป็นหัวใจของแบบดั้งเดิม ซึ่งมีการจัดการทรัพยากรแบบรวมกลุ่ม (Pooled) เมื่อเริ่มต้นโครงการจำเป็นต้องวางแผนอย่างเป็นระบบโดยอาศัยผู้มีประสบการณ์ ในขณะที่ระบบค่อนข้างมีความอ่อนไหวต่อการเปลี่ยนแปลงที่เกิดขึ้น ซึ่งจะใช้เวลาค่อนข้างนานในการจัดการปัญหาที่เกิดขึ้นและกลายมาเป็นปัญหาคอขวด ทั้งนี้การแก้ไขปัญหาเป็นเพียงการบรรเทาผลกระทบที่เกิดขึ้น	**กระจายศูนย์และสร้างความต่อเนื่องในกระบวนการทำงาน (Decentralized & Continuous):** แนวคิด DevOps จะเน้นการทำงานภายในทีมระดับเป็นหน่วยเล็กๆ ทำให้เกิดความช่วยเหลือกัน รวมถึงสร้างความเป็นอัตโนมัติในกระบวนการทำงานเพื่อทำให้การดำเนินการได้อย่างสะดวก

ตารางที่ 2 การเปรียบเทียบความแตกต่างระหว่างกระบวนการพัฒนาซอฟต์แวร์แบบดั้งเดิมกับแนวคิด DevOps (ต่อ)

	มิติที่วัดผล	การพัฒนาซอฟต์แวร์แบบดั้งเดิม (Traditional IT)	แนวคิด DevOps
ประสิทธิภาพและวัฒนธรรมการทำงาน	การส่งมอบซอฟต์แวร์	**มีความเสี่ยงสูงในแต่ละครั้งที่ส่งมอบ (High Risk Event):** การดำเนินงานส่งมอบซอฟต์แวร์มีความเสี่ยงสูงและเต็มไปด้วยปัญหา การบริหารจัดการค่อนข้างยาก เนื่องจากถูกควบคุมดูแลจากผู้บริหารระดับสูง อีกทั้งความต้องการที่เกิดขึ้นมาต้องได้รับความเห็นชอบจากทุกฝ่ายขององค์กร ซึ่งเป็นไปได้ยากที่ฝ่ายไอทีจะดูแลได้ตลอดเวลาทั้งก่อนและหลังการส่งมอบซอฟต์แวร์	**ไม่เป็นกิจกรรม (Nonevent):** แนวคิด DevOps จะไม่มองการส่งมอบซอฟต์แวร์เป็นกิจกรรม แต่พยายามที่จะลดความเสี่ยงที่จะเกิดขึ้นแบบรายวันเมื่อเกิดการเปลี่ยนแปลงโค้ด รวมถึงสร้างการทดสอบแบบอัตโนมัติเพื่อให้เกิดความมั่นใจในทุกสภาพแวดล้อมที่ให้บริการ รวมถึงสามารถเพิ่มฟังก์ชันใหม่ๆ ได้อย่างรวดเร็ว
	ข้อมูล	**ข้อมูลกระจัดกระจาย (Disseminated):** แบบดั้งเดิมข้อมูลถูกสร้างโดยผู้ที่มีหน้าที่รับผิดชอบ จากนั้นก็จะรวบรวมข้อมูลจากแหล่งอื่นๆ เพื่อสร้างเป็นรายงาน หลังจากนั้นเมื่อผ่านการอนุมัติก็จะนำข้อมูลแบ่งปันให้กับทีมอื่นๆ เช่น นักทดสอบระบบ นักพัฒนาซอฟต์แวร์ เป็นต้น โดยปกติรายงานเหล่านี้จะไม่ถูกใช้งานเนื่องจากมีข้อมูลจำนวนมาก ไม่มีเวลาเพียงพอ หรือข้อมูลดังกล่าวไม่ถูกต้อง	**สามารถจัดการได้ (Actionable):** แนวคิด DevOps จะทำให้หน่วยงานแต่ละทีมทำงานร่วมกันโดยสร้างข้อมูลร่วมกัน ซึ่งข้อมูลจะถูกใช้งานเนื่องจากการเก็บบันทึกในแหล่งเดียวกันและรวบรวมเฉพาะข้อมูลที่จำเป็นโดยอัตโนมัติ ข้อมูลดังกล่าวจะถูกประมวลผลภายในทีม เพื่อลดระยะเวลาในการสร้างรายงาน การอนุมัติ และการรอเป็นเวลานาน โดยผลลัพธ์ที่เกิดขึ้นทำให้ทีมงานสามารถตอบสนองต่อข้อมูลที่เกิดขึ้นได้อย่างรวดเร็ว
	วัฒนธรรม	**ไม่เกิดความล้มเหลว (Do Not Fail):** การพัฒนาซอฟต์แวร์แบบดั้งเดิมจะมีพื้นฐานที่ไม่กระทำการที่ก่อให้เกิดความเสี่ยงต่อธุรกิจ ส่งผลให้กระบวนการที่เกิดขึ้นในฝ่ายไอทีเป็นไปอย่างอย่างล่าช้า ซึ่งกระบวนการเหล่านี้จะป้องกันความล้มเหลวที่อาจเกิดขึ้น	**ความล้มเหลวในช่วงเริ่มต้น (Fair Early):** แนวคิด DevOps เข้าใจว่าความล้มเหลวเป็นสิ่งที่หลีกเลี่ยงไม่ได้ จึงเลือกที่จะพยายามกำจัดความล้มเหลวออกไปโดยเลือกวิธีการและช่วงเวลาที่เหมาะสม แม้ว่าจะมีล้มเหลวแต่ก็เป็นเพียงขนาดเล็กในช่วงต้นของโครงการและสามารถกู้กลับมาได้อย่างรวดเร็ว

ตารางที่ 2 การเปรียบเทียบความแตกต่างระหว่างกระบวนการพัฒนาซอฟต์แวร์แบบดั้งเดิมกับแนวคิด DevOps (ต่อ)

	มิติที่วัดผล	การพัฒนาซอฟต์แวร์แบบดั้งเดิม (Traditional IT)	แนวคิด DevOps
การประเมินผล	มาตรวัด	**ค่าใช้จ่ายและความสำเร็จ (Cost & Capacity):** การประเมินผลแบบดั้งเดิมในด้านค่าใช้จ่ายและความสามารถ โดยฝ่ายไอทีจะประเมินผลความสำเร็จที่เกิดขึ้นจากการใช้ค่าใช้จ่ายที่น้อยที่สุด จึงพยายามที่จะลดต้นทุนในขณะที่คงความสำเร็จไว้เท่าเดิม ซึ่งทำให้แนวคิดการจัดจ้างบุคคลภายนอก (Outsource) กลายมาเป็นที่นิยม และมีความเหมาะสมอย่างมากต่อระบบแบบเก่า	**ค่าใช้จ่าย ความสำเร็จ และเวลาในการดำเนินงาน (Cost , Capacity and Flow (Time)):** ในยุคปัจจุบันเน้นการทำงานร่วมกันของแต่ละระบบ โดยมีมาตรวัดที่เพิ่มขึ้นมา คือ เวลา ซึ่งแนวคิด DevOps มีความเข้าใจในส่วนนี้ ทำให้วางแผนระบบตั้งแต่เริ่มต้นจนกระทั่งสิ้นสุดวงจรการพัฒนาซอฟต์แวร์ เพื่อลดความสูญเปล่าของเวลา เพิ่มประสิทธิผล ประสิทธิภาพ และคุณภาพ รวมถึงมุ่งเป้าไปยังกิจกรรมที่ก่อให้เกิดมูลค่า
	ตัวกำหนดความสำเร็จ	**“ฉันทำงานเสร็จแล้ว” (I Did my Job):** การพัฒนาซอฟต์แวร์แบบดั้งเดิมผู้รับผิดชอบจะทำเพียงแต่ส่วนที่ตนเองได้รับผิดชอบเท่านั้น เพื่อให้ทันต่อเวลาส่งมอบงาน ทำให้มองข้ามถึงเรื่องคุณภาพ และก่อให้เกิดกระบวนการทำงานที่ไม่เหมาะสม	**“ฉันพร้อมที่จะส่งมอบซอฟต์แวร์แล้ว” (Its Ready to Deploy):** เป็นการสร้างทีมข้ามสายงาน ซึ่งเกิดจากสมาชิกหลายทีมรวมกัน แต่มีสิ่งที่รับผิดชอบเป็นเป้าหมายเดียวกัน คือ เพื่อนำซอฟต์แวร์ที่มีคุณภาพเข้าสู่ตลาด

หมายเหตุ. จาก “Comparing DevOps to traditional IT: Eight key differences” โดย Kapadia (2015)

อีกทั้ง Mohamed (2016) ได้เปรียบเทียบผลลัพธ์การดำเนินงานของแนวคิด DevOps กับวิธีการพัฒนาซอฟต์แวร์อื่นๆ อาทิ มูลค่าที่เกิดขึ้นจากการใช้แนวคิด DevOps ถึง 7 เท่า อัตราการเปลี่ยนแปลงที่เกิดถี่ขึ้นเมื่อใช้แนวคิด DevOps มากถึง 14 เท่า โดยครึ่งหนึ่งของอัตราการเปลี่ยนแปลงที่ผิดพลาดนั้นสามารถแก้ไขได้อย่างรวดเร็วมากกว่าถึง 4 เท่า และใช้ระยะเวลาดำเนินการน้อยกว่า 10 เท่า นอกจากนี้ยังรองรับการเปลี่ยนแปลงของซอฟต์แวร์ถึง 600 ครั้ง/สัปดาห์ โดยมีอัตราความสำเร็จที่เกิดขึ้นในการเปลี่ยนแปลงถึง 99.5 % ดังตารางที่ 3

ตารางที่ 3 การเปรียบเทียบผลลัพธ์การดำเนินงานของแนวคิด DevOps กับวิธีการพัฒนาซอฟต์แวร์อื่นๆ

เงื่อนไข	Waterfall/ Ad-hoc	Agile	Propose DevOps Continuous Delivery
อัตราการส่งมอบซอฟต์แวร์โดยเฉลี่ยต่อเดือน	ทุกๆ 6 เดือน	ทุกๆ 3 สัปดาห์	ทุกวัน
จำนวนข้อผิดพลาด	10	8	2
หน่วยเวลาที่ใช้ในการเปลี่ยนแปลง	เดือน	วัน	นาที
อัตราเฉลี่ยระยะเวลาในความล้มเหลว (Mean Time Between Failure : MTBF)	6 เดือน	16 ชั่วโมง	4 ชั่วโมง
ระยะเวลาในการแก้ไขความล้มเหลว (Mean Time To Repair : MTTR)	6 เดือน	24 ชั่วโมง	6 ชั่วโมง
ระยะเวลาที่ขัดข้อง (Outages Time)	X	น้อยกว่า 5X	น้อยกว่า 10X
ทรัพยากรที่เกิดขึ้นต่อหนึ่งหน่วยเวลา (Resources Productive Time)	X	3X	7X
การเปลี่ยนแปลง	X	5X	14X
อัตราความสำเร็จในการเปลี่ยนแปลง	X	80% X	99.5% X

หมายเหตุ. จาก “Software Release Management Evolution -Comparative Analysis across Agile and DevOps Continuous Delivery” โดย Mohamed (2016)

2.2 ปัจจัยความสำเร็จในโครงการพัฒนาซอฟต์แวร์แบบแอจไจล์ (Critical Success Factor in Agile Software Projects)

ปัจจัยความสำเร็จเป็นวิธีปฏิบัติที่ถูกพิสูจน์และเป็นมาตรวัดด้านประสิทธิภาพภายในองค์กร โดยจะที่ใช้วัดความพึงพอใจ เพื่อให้มั่นใจได้ว่าองค์กรผู้พัฒนาซอฟต์แวร์ด้วยแนวคิดแอจไจล์ ประสบความสำเร็จในการแข่งขันด้านประสิทธิภาพของบุคคล แผนก และองค์กร ทั้งนี้ปัจจัยความสำเร็จในการพัฒนาซอฟต์แวร์จะสัมพันธ์กับเทคนิคการบริหารโครงการ (Project Management) รวมถึงวิศวกรรมซอฟต์แวร์ (Software Engineering) และกลยุทธ์ขององค์กร Chow และ Cao (2008) ได้นำเสนอปัจจัยในรูปแบบของมิติ ประกอบด้วย 5 มิติ ที่ใช้ในการพิจารณา ทั้งนี้ในแต่ละมิติจะมีปัจจัยเป็นส่วนย่อยของมิติเหล่านั้น ได้แก่ 1) มิติด้านองค์กร (Organizational Dimension) เป็นส่วนแรกที่เกิดขึ้น เมื่อ

องค์กรนำแนวคิดแอจไจล์มาใช้ ผู้บริหารจำเป็นต้องให้การสนับสนุนที่ดี องค์กรต้องมีการปรับวัฒนธรรมองค์กร และแนวคิดของบุคลากรภายในองค์กร 2) มิติด้านบุคคล (People Dimension) คือ บุคลากรผู้มีบทบาทในโครงการพัฒนาซอฟต์แวร์ รวมถึงความรู้เชิงเทคนิคของบุคลากรเหล่านั้น ที่จะช่วยให้โครงการพัฒนาซอฟต์แวร์แบบแอจไจล์ประสบความสำเร็จ 3) มิติด้านกระบวนการ (Process Dimension) เป็นกระบวนการต่างๆ ที่เกิดขึ้นจากการใช้แนวคิดแอจไจล์ เช่น SCRUM, Test Drive Development (TDD), Extreme Programming (XP) เป็นต้น 4) มิติด้านเทคนิค (Technical Dimension) เป็นเทคนิคต่างๆ ที่นำไปปฏิบัติ เช่น การบูรณาการอย่างต่อเนื่อง (Continuous Integration) แพร์โปรแกรมมิง (Pair Programing) รวมถึงการใช้เครื่องมือต่างๆ ในการดำเนินงาน (Hameed, Latif, & Kholief, 2016) 5) มิติด้านโครงการ (Project Dimension) คือ วิธีการต่างๆ ของแนวคิดแอจไจล์ที่เหมาะสมกับโครงการ เช่น ลักษณะความต้องการของโครงการที่ไม่ชัดเจน และเปลี่ยนแปลงตลอดเวลา เป็นต้น (Hameed et al., 2016) ต่อมา Hameed Latif และ Kholief (2016) ได้ปรับปรุงกรอบแนวคิดของ Chow และ Cao (2008) โดยนำเสนอปัจจัยความสำเร็จในรูปแบบมิติเช่นเดียวกัน ซึ่งเกิดจากการจัดกลุ่มของข้อมูลที่ถูกรวบรวมจากการศึกษางานวิจัยเชิงปริมาณและเชิงคุณภาพ โดยปัจจัยที่มีลักษณะคล้ายกันจะถูกรวมเป็นกลุ่มเดียวกัน

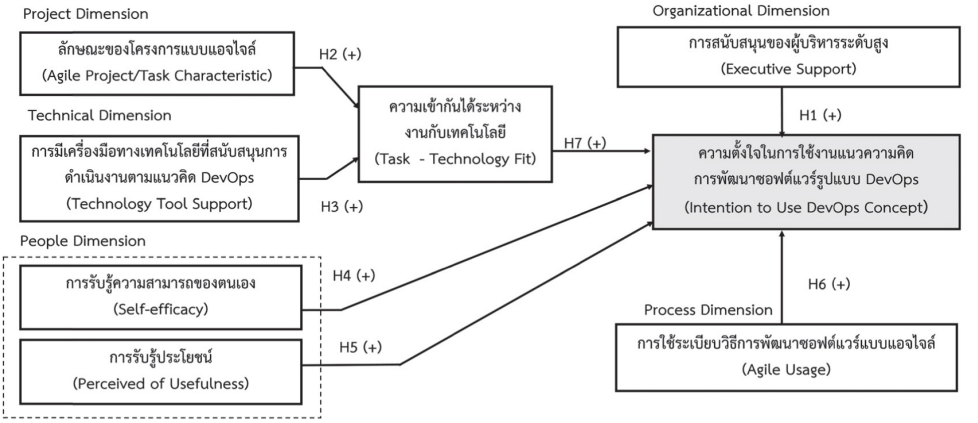
แม้ว่างานวิจัยฉบับนี้จะอ้างอิงปัจจัยความสำเร็จในโครงการพัฒนาซอฟต์แวร์แบบแอจไจล์มาเพื่อทดสอบในบริบทของแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps เนื่องจากแนวคิด DevOps เป็นส่วนขยายที่เพิ่มเติมจากกระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์ขึ้นมา โดยสอดคล้องกับภาพที่ 6 ซึ่งแสดงให้เห็นถึงความสัมพันธ์ระหว่างวิธีการพัฒนาซอฟต์แวร์ทั้งสองแบบ ซึ่งแอจไจล์ได้ประสานการทำงานร่วมกันของทีมออกแบบ ทีมทดสอบระบบ และทีมพัฒนาซอฟต์แวร์ ในขณะที่แนวคิด DevOps ได้ก้าวข้ามขึ้นไปอีกขั้นโดยได้เชื่อมโยงการทำงานไปถึงทีมปฏิบัติการ ซึ่งเป็นผู้ที่ส่งมอบซอฟต์แวร์ไปยังผู้ใช้งานอีกด้วย (Watts, 2017) ทั้งนี้วิธีการพัฒนาซอฟต์แวร์ทั้งสองแบบมีเป้าหมายเพื่อเพิ่มประสิทธิภาพของฝ่ายไอทีในการส่งมอบซอฟต์แวร์ ทำให้ทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการสามารถทำงานตอบสนองต่อระเบียบวิธีปฏิบัติแบบแอจไจล์ได้อย่างรวดเร็ว และลดปัญหาการสื่อสารระหว่างสองทีม (Riley, 2014) อีกทั้งงานวิจัยของ Menard Sweeney และ Shropshire (2017) ได้กล่าวถึงองค์ประกอบที่มีต่องานด้านนวัตกรรมในบริบทของแนวคิด DevOps ซึ่งมีบางมิติในงานวิจัยที่สอดคล้องกับมิติที่ผู้วิจัยใช้ในงานวิจัยฉบับนี้ เช่น ประโยชน์ที่ได้จากนวัตกรรม (Relative Advantage) ความเข้ากันได้ระหว่างงานกับเทคโนโลยี (Compatibility) เป็นต้น โดยกล่าวว่า DevOps เกิดจากการผสมผสานกันระหว่างบุคคล กระบวนการและการจัดเตรียมความต่อเนื่องของกระบวนการทำงานเพื่อส่งมอบไปยังลูกค้าโดยอาศัยเครื่องมือเพื่อสร้างความราบรื่นในการดำเนินงาน ทั้งนี้แนวคิด DevOps ยังสามารถทำงานร่วมกันได้ดีกับวิธีปฏิบัติของกระบวนการพัฒนาซอฟต์แวร์แบบแอจไจล์อีกด้วย

2.3 ความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

Ajzen (1985) กล่าวว่าความตั้งใจเป็นสิ่งที่เกิดก่อนพฤติกรรม โดยความตั้งใจสัมพันธ์กับทัศนคติและความเชื่อ ความตั้งใจจะส่งผลให้เกิดความพยายามลองกระทำบางอย่าง ที่สามารถทำนายถึงพฤติกรรมได้ ในบริบทของการพัฒนาซอฟต์แวร์ Diéguez Sepúlveda และ Cachero (2012) ได้ศึกษาถึงเครื่องมือในการประเมินความตั้งใจในการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ พบว่า ความตั้งใจในการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์จะเป็นตัวทำนายการใช้งานและการยอมรับการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์ดังกล่าว แต่จะต้องคำนึงถึงคุณลักษณะ (Characteristic) ของระเบียบวิธีการพัฒนาซอฟต์แวร์ด้วย ซึ่งมีอิทธิพลอย่างมากต่อการยอมรับการใช้งาน โดยสอดคล้องกับบริบทของความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ในงานวิจัยของ Menard และคณะ (2017) กล่าวว่าก่อนที่องค์กรจะนำแนวคิด DevOps เข้ามาใช้งาน ควรจะปรับทัศนคติของพนักงานให้มีทัศนคติเชิงบวกต่อสิ่งแนวคิดนั้น หากมิเช่นนั้นแล้วเมื่อนำไปใช้งานก็จะเกิดความล้มเหลวขึ้น โดยข้อค้นพบที่สำคัญในงานวิจัยพบว่า บางคุณสมบัติของนวัตกรรม เช่น ความเข้ากันได้ระหว่างงานกับเทคโนโลยีและประโยชน์ที่ได้จากนวัตกรรม ส่งผลต่อการความตั้งใจในการใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps แม้ว่าความง่ายต่อการใช้งานจะมีผลต่อการริเริ่มนำแนวคิด DevOps ไปใช้ แต่ในทางกลับกันความง่ายต่อการใช้งานนั้นไม่มีผลต่อการยอมรับการใช้งานเลย

3. สมมติฐานงานวิจัย

จากการทบทวนวรรณข้างต้น ผู้วิจัยต้องการนำปัจจัยความสำเร็จนำเสนอให้ครบทุกมิติ ดังนั้นจึงได้นำเสนอปัจจัยในแต่ละมิติ ดังภาพที่ 10 ซึ่งประกอบด้วย มิติด้านองค์กรเกี่ยวกับปัจจัยการสนับสนุนของผู้บริหารระดับสูง มิติด้านโครงการเกี่ยวกับปัจจัยลักษณะของโครงการแบบแอจไจล์ มิติด้านเทคนิคเกี่ยวกับปัจจัยการมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps มิติด้านกระบวนการเกี่ยวกับปัจจัยการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ และความเข้ากันได้ระหว่างงานกับเทคโนโลยี ทั้งนี้ในงานวิจัยของ Hameed Latif และ Kholief (2016) ได้กล่าวว่ามิติด้านบุคคลถือว่าเป็นบทบาทที่สำคัญในโครงการพัฒนาซอฟต์แวร์ ทั้งนี้ในงานวิจัยดังกล่าวได้ให้ความสำคัญแก่มิติด้านบุคคลสูงกว่ามิติอื่นๆ ซึ่งผลการวิจัยแสดงให้เห็นว่ามิติด้านบุคคลเป็นปัจจัยที่ได้ค่าน้ำหนักสูงที่สุดเมื่อเทียบกับมิติด้านอื่น ทำให้ผู้วิจัยพิจารณาระบุปัจจัยของมิติด้านบุคคลในจำนวนที่มากกว่ามิติอื่นเช่นกัน โดยเลือกปัจจัยด้านการรับรู้ประโยชน์และการรับรู้ความสามารถของตนเองในงานวิจัยครั้งนี้



ภาพที่ 10 กรอบแนวคิดการวิจัย

3.1 นิยามศัพท์

การสนับสนุนของผู้บริหารระดับสูง (Executive Support: ES)

การสนับสนุนของผู้บริหารระดับสูง คือ การสนับสนุนที่ดีจากผู้บริหารตั้งแต่เริ่มต้นโครงการและระหว่างการดำเนินโครงการอย่างต่อเนื่อง (Senapathi & Srinivasan, 2011) รวมถึงการเตรียมสิ่งอำนวยความสะดวกในการทำงาน อาทิ สถานที่ทำงาน เป็นต้น (Chow & Cao, 2008)

ลักษณะของโครงการแบบแอจไจล์ (Agile Project/task Characteristic: APC)

ลักษณะของโครงการแบบแอจไจล์ คือ ลักษณะโครงการที่มีการแบ่งส่วนของซอฟต์แวร์ออกเป็นส่วนย่อยๆ เพื่อลดระยะเวลาดำเนินการ ตอบสนองความต้องการได้ทันเวลา และปรับเปลี่ยนได้ตามความต้องการที่เปลี่ยนไป โดยเน้นให้ผู้ใช้งานมีส่วนร่วม (Young, 2013)

การมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานในแนวความคิด DevOps (Technology Tool Support: TS)

การมีเครื่องมือทางเทคโนโลยีที่สนับสนุน คือ ชุดของเครื่องมือที่ใช้รองรับการทำงานตามแนวคิด DevOps ซึ่งเป็นสิ่งอำนวยความสะดวกในการดำเนินงานในแต่ละขั้นตอน จากต้นทางสู่ปลายทางตามกระบวนการพัฒนาซอฟต์แวร์ ดังแสดงในภาพที่ 4 (Mohamed, 2016)

การรับรู้ความสามารถของตนเอง (Self-efficacy: PSE)

การรับรู้ความสามารถของตนเอง คือ ความเชื่อมั่นและความมั่นใจในความสามารถของตนเองเกี่ยวกับการเรียนรู้แนวคิด DevOps ที่ต้องมีการแลกเปลี่ยนความรู้และประสบการณ์กับทีมงานที่เกี่ยวข้อง รวมถึงความสามารถในการใช้เครื่องมือต่างๆ ที่สนับสนุนการทำงาน

การรับรู้ประโยชน์ (Perceived of Usefulness: PU)

การรับรู้ประโยชน์ คือ การรับรู้ว่า แนวคิด DevOps เป็นกระบวนการที่ทำให้การพัฒนาซอฟต์แวร์เกิดความต่อเนื่องและเป็นอัตโนมัติมากขึ้น ก่อให้เกิดความน่าเชื่อถือในการส่งมอบซอฟต์แวร์ (Babar et al., 2011) รวมถึงลดความขัดแย้งในการดำเนินงานระหว่างทีมนักพัฒนาซอฟต์แวร์ และทีมปฏิบัติการ (Mohamed, 2016)

การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (Agile Usage: AU)

การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์แบ่งออกเป็น 2 มิติ คือ เชิงกว้าง (Horizontal) เช่น อัตราการใช้งานแนวคิดแอจไจล์ในโครงการต่างๆ ระยะเวลาในการใช้งาน (Senapathi & Srinivasan, 2011) และเชิงลึก (Vertical) คือ การเลือกใช้วิธีการ วิธีปฏิบัติ และเครื่องมือในกระบวนการพัฒนาซอฟต์แวร์ (Rodríguez, Markkula, Oivo, & Turula, 2012)

ความเข้ากันได้ระหว่างงานกับเทคโนโลยี (Task - Technology Fit: TTF)

ความเข้ากันได้ระหว่างงานกับเทคโนโลยี คือ การเลือกใช้วิธีการและเครื่องมือให้เหมาะสมกับลักษณะโครงการ (Senapathi & Srinivasan, 2011) เพื่อสนับสนุนการดำเนินงานให้มีประสิทธิภาพ (Goodhue & Thomspson, 1995) รวมถึงตรวจสอบระดับความเข้ากันได้กับระบบที่มีอยู่ในปัจจุบัน (Park & Raven, 2015)

ความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps (Intention to Use DevOps Concept: ICD)

ความตั้งใจที่ผู้ที่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ตัดสินใจที่จะยอมรับการใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ซึ่งเกิดจากความเชื่อและการมีทัศนคติที่ดีต่อแนวคิดดังกล่าว (Menard et al., 2017)

3.2 การตั้งสมมติฐาน

มิติด้านองค์กร เกี่ยวกับปัจจัยการสนับสนุนของผู้บริหารระดับสูง

การสนับสนุนของผู้บริหารในการเริ่มต้นโครงการถือเป็นสิ่งสำคัญที่ทำให้โครงการต่างๆ เกิดขึ้นได้ การสนับสนุนอย่างต่อเนื่องแสดงถึงความใส่ใจต่อพนักงาน (Chan & Thong, 2008) ทั้งนี้พนักงานที่ได้รับการสนับสนุนที่เพียงพอจะส่งผลให้เกิดการใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์อย่างเหมาะสม (Senapathi & Srinivasan, 2011) นอกจากนี้การสนับสนุนจากผู้บริหารระดับสูงเป็นปัจจัยที่ผลักดันให้การใช้กระบวนการพัฒนาซอฟต์แวร์ใหม่หรือที่เปลี่ยนแปลงประสบความสำเร็จ (Russo, Shams, & Fitzgerald, 2013) ส่งผลให้เกิดประโยชน์ในแง่คุณภาพงาน (Chow & Cao, 2008) ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 1 (H1) การสนับสนุนของผู้บริหารระดับสูง ส่งผลเชิงบวกต่อความตั้งใจใช้งานแนวคิด DevOps

มิติด้านด้านโครงการ เกี่ยวกับปัจจัยลักษณะของโครงการแบบแอจไจล์

การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์กับโครงการ ทำให้การควบคุมและการจัดการความเปลี่ยนแปลงที่เกิดจากลูกค้าได้ดี จึงมีความเหมาะสมที่จะนำแนวคิดแอจไจล์เข้ามาใช้งาน (McAvoy, Sammon, & Owens, 2007) อีกทั้งวิธีปฏิบัติในระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มีการวนซ้ำ (Iteration) ในแต่ละรอบของการพัฒนา และมีการปรับปรุงคุณภาพของโค้ด (Refactor Code) ให้ดีขึ้น ซึ่งเป็นการเพิ่มความสามารถในการพัฒนาซอฟต์แวร์ นอกจากนี้ แนวคิดของ DevOps ที่ถูกพัฒนาต่อยอดมาจากแนวคิดแอจไจล์ ยังเพิ่มขีดความสามารถของนักพัฒนาซอฟต์แวร์ ในด้านการส่งมอบซอฟต์แวร์แก่ผู้ใช้งานได้ โดยแนวคิด DevOps สามารถรองรับการพัฒนาซอฟต์แวร์ที่ทำงานอยู่บนพื้นฐานเทคโนโลยีการประมวลผลแบบกลุ่มเมฆ เช่น แพลตฟอร์มเสมือนบริการ (Platform-as-a-service: PAAS) ได้ทำให้พนักงานสามารถเข้าถึงหรือใช้งานโปรแกรมประยุกต์เพื่อทำงานได้อย่างสะดวก (Farroha & Farroha, 2014) เมื่อนำเอาแนวคิดแอจไจล์มาใช้งานร่วมกับแนวคิด DevOps จะช่วยการส่งมอบซอฟต์แวร์ทำได้อย่างต่อเนื่อง และเป็นอัตโนมัติมากขึ้น (Mohamed, 2016) ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 2 (H2) ลักษณะของโครงการแบบแอจไจล์ ส่งผลเชิงบวกต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยี

มิติด้านเทคนิค เกี่ยวกับปัจจัยการมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps

การมีเครื่องมือสนับสนุนการทำงานจะช่วยเติมเต็มกระบวนการพัฒนาซอฟต์แวร์ เช่น แนวคิดแอจไจล์ ให้มีประสิทธิภาพ (Hovsepyan & Landuyt, 2015) โดยการสนับสนุนการดำเนินงานในแต่ละขั้นตอนช่วยให้พนักงานเกิดความพึงพอใจ (Senapathi & Srinivasan, 2011) ส่งผลให้การใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์ประสบความสำเร็จ (Tripp & Armstrong, 2014) ทั้งในฝั่งลูกค้าและฝั่งนักพัฒนาซอฟต์แวร์ (Russo et al., 2013) นอกจากนี้ในบริบทของแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps เครื่องมือที่สนับสนุนการทำงานแต่ละขั้นตอนสามารถทำงานร่วมกันได้อย่างต่อเนื่องและเป็นอัตโนมัติ และช่วยตรวจสอบความถูกต้อง เพื่อสร้างความมั่นใจในการส่งมอบซอฟต์แวร์แก่ลูกค้ามากขึ้นได้ (Lwakatare et al., 2016) ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 3 (H3) เครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ส่งผลเชิงบวกต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยี

มิติด้านบุคคล เกี่ยวกับปัจจัยการรับรู้ความสามารถของตนเอง

การรับรู้ความสามารถของตนเองเป็นสิ่งสำคัญในการยอมรับการใช้งาน ระดับการรับรู้ความสามารถของตนเองที่สูงจะยิ่งส่งผลต่อความตั้งใจใช้งานระบบมากยิ่งขึ้น (Kim, Suh, Lee, & Choi, 2010) ทั้งนี้ผู้ที่รับรู้ความสามารถของตนเองในระดับที่สูงกว่าจะมีความมั่นใจในการใช้ทักษะที่มีของตนมากกว่าผู้ที่รับรู้ความสามารถตนเองในระดับต่ำ (Fathema, Shannon, & Ross, 2015) ความมั่นใจในการใช้เทคโนโลยีที่เพียงพอ ยังช่วยขับเคลื่อนการใช้นวัตกรรมที่ซับซ้อนได้ ในทางกลับกันหากปราศจากความมั่นใจในการใช้เทคโนโลยีก็จะกลายเป็นอุปสรรคต่อการปฏิบัติงาน (Wang, Li, & Hsieh, 2013) ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 4 (H4) การรับรู้ความสามารถตนเอง ส่งผลเชิงบวกต่อความตั้งใจใช้งานแนวคิด DevOps

มิติด้านบุคคล เกี่ยวกับปัจจัยการรับรู้ประโยชน์

การแสดงให้ผู้ใช้งานเห็นถึงผลการดำเนินงานที่ดี จะช่วยเพิ่มประสิทธิภาพในการทำงาน และส่งผลต่อความตั้งใจในการยอมรับระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (Chan & Thong, 2008) การรับรู้ประโยชน์ส่งผลเชิงบวกต่อการยอมรับการใช้แนวคิดแอจไจล์ (Vijayasarathy & Turk, 2010) นอกจากนี้ Fiampolis และ Groll (2016) ยังกล่าวว่า ผลประโยชน์ที่เกิดขึ้นทั้งเชิงบุคคลและเชิงองค์กรส่งผลให้เกิดการนำแนวคิด DevOps มาใช้งาน เพื่อเพิ่มความรวดเร็วในการดำเนินงาน ตอบสนองความต้องการของธุรกิจได้ทัน ขณะที่ยังคงเสถียรภาพ ความน่าเชื่อถือ และความมั่นคงของระบบไว้ได้ ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 5 (H5) การรับรู้ประโยชน์ ส่งผลเชิงบวกต่อความตั้งใจใช้งานแนวคิด DevOps

มิติด้านกระบวนการ เกี่ยวกับปัจจัยการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์

แนวคิด DevOps กลายมาเป็นส่วนขยายของแนวคิดแอจไจล์ ที่ช่วยเติมเต็มในส่วนการส่งมอบซอฟต์แวร์ และรองรับการเปลี่ยนแปลงฟังก์ชันการทำงานตามความต้องการของผู้ใช้งาน (Airaj, 2015) รวมถึงเน้นด้านการทดสอบ และตรวจสอบคุณภาพซอฟต์แวร์ (Deshpande, 2016) ทั้งนี้องค์กรที่มีการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์และกระบวนการส่งมอบซอฟต์แวร์อย่างต่อเนื่อง มีแนวโน้มที่จะปรับตัวเพื่อใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มากขึ้น (Lwakatare et al., 2016) โดยแนวคิดนี้จะเข้ามาผนวกระหว่างกระบวนการพัฒนาซอฟต์แวร์และการดำเนินการส่งมอบซอฟต์แวร์ รวมถึงการดำเนินการติดตาม ดูแลรักษาระบบภายในวงชีพการพัฒนาซอฟต์แวร์ด้วย (Engel & Schweimler, 2016) ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 6 (H6) การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ ส่งผลเชิงบวกต่อความตั้งใจใช้งานแนวคิด DevOps

ความเข้ากันได้ระหว่างงานกับเทคโนโลยี

การนำเทคโนโลยีไปใช้เพื่อให้งานประสบความสำเร็จมีความเชื่อว่า ระบบที่จะก่อให้เกิดประโยชน์นั้นต้องมีความเข้ากันได้ ระหว่างงานกับเทคโนโลยี (Goodhue & Thomspson, 1995) ทั้งนี้กรอบวิธีปฏิบัติของแนวคิด DevOps มีพื้นฐานมาจากระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ โดยความรับผิดชอบของทีมนักพัฒนาซอฟต์แวร์จะเป็นการติดต่อกับผู้ใช้งาน เพื่อวิเคราะห์ความต้องการ ส่วนทีมปฏิบัติการจะติดต่อประสานงานกับผู้ใช้งาน เพื่อนำซอฟต์แวร์ไปติดตั้งในสภาพแวดล้อมต่างๆ ของระบบ (Farroha & Farroha, 2014) นอกจากนี้ทั้งสองทีมยังต้องเลือกใช้ชุดเครื่องมือในการทำโครงการให้เหมาะสมกับงานที่ทำ (Mohamed, 2016) ซึ่งการใช้งานแนวคิด DevOps ตลอดจนเครื่องมือที่เกี่ยวข้องจะช่วยปรับปรุงการพัฒนาซอฟต์แวร์ด้านการประกันคุณภาพของซอฟต์แวร์ และสร้างความต่อเนื่องในการส่งมอบบริการแก่ลูกค้า ซึ่งเป็นส่วนสำคัญที่ทำให้การพัฒนาและส่งมอบซอฟต์แวร์ประสบความสำเร็จ (Olszewska & Waldén, 2015) ดังนั้นจึงตั้งสมมติฐานได้ว่า

สมมติฐานที่ 7 (H7) ความเข้ากันได้ระหว่างงานกับเทคโนโลยี ส่งผลเชิงบวกต่อความตั้งใจใช้งานแนวคิด DevOps

4. วิธีการวิจัย

4.1 ประชากร กลุ่มตัวอย่าง

ประชากรในการวิจัยคือ บุคลากรในบริษัทผู้ผลิตซอฟต์แวร์หรือหน่วยงานที่พัฒนาระบบขององค์กรที่มีอำนาจในการตัดสินใจ มีส่วมร่วมในการตัดสินใจ หรือสามารถให้คำแนะนำในการนำแนวคิดหรือระเบียบวิธีการพัฒนาซอฟต์แวร์รูปแบบใหม่มาใช้งานในองค์กรได้ โดยองค์กรเหล่านี้มีการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์เป็นระเบียบวิธีหนึ่งด้วย ทั้งนี้กลุ่มตัวอย่างในงานวิจัยหากลุ่มตัวอย่างแบบ (Snowball Sampling) (Šumak & Šorgo, 2016) โดยการสร้างเครือข่ายข้อมูลโดยใช้กลุ่มตัวอย่างกลุ่มแรก ซึ่งจะใช้วิธีหากลุ่มตัวอย่างแบบที่ไม่ใช้ความน่าจะเป็นแบบเจาะจง (Purposive Sampling) แล้วให้นำเสนอบุคคลที่มีลักษณะใกล้เคียงต่อไป โดยงานวิจัยนี้ใช้การคำนวณหาขนาดกลุ่มตัวอย่างโดยใช้ 5 เท่า ของข้อคำถามต่อตัวแปร ในงานวิจัยได้ยึดข้อคำถามเป็นจำนวนตัวแปร ดังนั้นจึงมีตัวแปรจำนวน 37 ตัวแปร (Hair, Black, Babin, & Anderson, 2010) มีการคำนวณดังนี้ 5 x 37 = 185 โดยผู้วิจัยได้ตั้งเป้าหมายของกลุ่มตัวอย่างไว้ที่จำนวน 200 ตัวอย่าง

4.2 การจัดสร้างเครื่องมือเพื่อการวิจัย และการตรวจสอบคุณภาพของเครื่องมือ

แบบสอบถามประกอบด้วย 5 ส่วน ได้แก่ ส่วนที่ 1 คำถามเพื่อคัดกรองกลุ่มเป้าหมายให้ตรงตามประชากรที่กำหนด ส่วนที่ 2 การให้ความรู้เบื้องต้นเกี่ยวกับแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps เพื่อให้เกิดความเข้าใจที่ตรงกันเกี่ยวกับแนวคิด DevOps และคำถามเพื่อคัดกรองกลุ่มเป้าหมายที่มีความรู้ความเข้าใจตรงกันต่อแนวคิดดังกล่าว ซึ่งผู้ตอบแบบสอบถามต้องตอบถูกต้อง 2 ใน 3 ข้อขึ้นไป ส่วนที่ 3 คำถามเกี่ยวกับเป็นปัจจัยที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ซึ่งดัดแปลงจากงานวิจัยของ Zwikael (2008) Thakurta และ Roy (2012) Lambert (2011) Kornecki (2007) Park และ Raven (2015) Awad (2005) Rodríguez และคณะ (2012) Baleghi-Zadeh Ayub Mahmud และ Daud (2014) ชาญชัย อรรคผาติ. (2557) Soto-Acosta Ramayah และ Popa (2013) สุชาดา เกยุระ (2553) Fathema และคณะ (2015) Sarıtaş Yıldız และ ŞENEL (2015) Hoehle (2011) ที่ใช้การวัดผลในรูปแบบของลิเคิร์ทสเกล (Likert Scale) 5 ระดับ ตั้งแต่เห็นด้วยอย่างยิ่ง (5) จนถึงไม่เห็นด้วยอย่างยิ่ง (1) ส่วนที่ 4 คำถามเกี่ยวกับการใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ เก็บข้อมูลด้วยมาตราแบบอัตราส่วน (Ratio Scale) รวมถึงคำถามเกี่ยวกับชุดเครื่องมือที่ใช้สำหรับผู้ที่ใช้งานแนวคิด DevOps อยู่แล้ว ส่วนที่ 5 ข้อมูลทั่วไปเกี่ยวกับผู้ตอบแบบสอบถาม เช่น เพศ ลักษณะขององค์กรที่ทำงาน ตำแหน่งงาน เก็บข้อมูลด้วยมาตรานามบัญญัติ (Nominal Scale) รวมถึงข้อมูลอายุ ระดับการศึกษาสูงสุด และประสบการณ์ในการทำงานด้วยมาตราแบบอัตราส่วน ซึ่งแบบสอบถามแบบกระดาษและแบบออนไลน์ตามลิงก์ (Link) และ QR Code แสดงอยู่ที่ภาคผนวกท้ายบทความ

ผู้วิจัยได้ประมวลผลด้วยสถิติเชิงพรรณนา ได้แก่ ค่าร้อยละ ค่าเฉลี่ย และส่วนเบี่ยงเบนมาตรฐาน และทดสอบสมมมติฐานทางสถิติ โดยตรวจสอบความสมเหตุสมผลของเครื่องมือวัด (Construct Validity) ด้วยการวิเคราะห์องค์ประกอบ (Factor Analysis) เพื่อจัดกลุ่มข้อคำถามของปัจจัยเดียวกัน ซึ่งต้องมีความสัมพันธ์กันสูง แต่ในปัจจัยต่างกันต้องมีความสัมพันธ์กันน้อยมากหรือไม่มีเลย (กัลยา วานิชย์บัญชา, 2551) โดยเกณฑ์ที่ใช้คือ ค่าน้ำหนักขององค์ประกอบ (Factor Loading) มากกว่า 0.5 และใช้ค่าบ่งบอกความสามารถขององค์ประกอบ (Eigenvalue) มากกว่า 1 ด้วยวิธีการหมุนแกนแบบ Varimax เนื่องจากเทคนิคดังกล่าวได้รับความนิยมในการทำงานวิจัย และเป็นเทคนิคที่ทำให้มีจำนวนตัวแปรน้อยที่สุด แต่มีค่าน้ำหนักองค์ประกอบมากในแต่ละปัจจัย (รสริน ศรีริกานนท์, 2552) จึงทำให้ผู้วิจัยพิจารณาเทคนิคดังกล่าวในงานวิจัยครั้งนี้ รวมถึงการตรวจสอบความเหมาะสมของกลุ่มตัวอย่างด้วยการหา KMO (Kaiser-Meyer-Olkin) ที่มากกว่า 0.5 (Hair et al., 2010) ส่วนการวิเคราะห์ความเที่ยงของเครื่องมือ โดยการใช้ค่าสัมประสิทธิ์ครอนบาคแอลฟาด้วยเกณฑ์มากกว่า 0.7 ซึ่งถือว่าเป็นระดับที่เชื่อถือได้ในงานวิจัย (Schmitt, 1996)

สำหรับขั้นตอนการทดสอบสมมติฐานผู้วิจัยเลือกใช้การวิเคราะห์การถดถอยแบบพหุคูณ (Multiple Regression Analysis) โดยใช้ค่า p-value ที่น้อยกว่าหรือเท่ากับ 0.05 เป็นเกณฑ์ และเมื่อพิจารณาจากกรอบงานวิจัยแล้วพบว่า ตัวแปรในงานวิจัยไม่มีตัวแปรปรับ/กำกับ (Moderator) อีกทั้งตัวแปรแต่ละมิติไม่มีตัวแปรแฝง (Latent Variable) ซึ่งสามารถสังเกตและวัดได้โดยตรง (นงลักษณ์ วิรัชชัย, 2555) ดังนั้น การวิเคราะห์การถดถอยจึงเพียงพอในการตอบคำถามงานวิจัยที่ศึกษาในครั้งนี้

5. ผลการศึกษา

ผู้วิจัยได้เก็บรวบรวมข้อมูลจากกลุ่มตัวอย่างจำนวน 275 ชุด ผ่านการแจกแบบสอบถามออนไลน์ไปยังองค์กรต่างๆ และแบบสอบถามแบบกระดาษด้วยการลงพื้นที่ในงานสัมมนาของกลุ่ม Agile66 ที่จัดขึ้นที่ตลาดหลักทรัพย์แห่งประเทศไทย เมื่อวันที่ 14 พฤษภาคม พ.ศ. 2560 ด้วยระยะเวลาในการเก็บแบบสอบถาม ตั้งแต่วันที่ 1-31 พฤษภาคม พ.ศ. 2560 โดยมีจำนวนแบบสอบถามที่สามารถใช้งานได้จำนวน 208 ชุด ที่เป็นไปตามเงื่อนไขของประชากร

5.1 การวิเคราะห์ตัวแปรและการประเมินความเที่ยงของเครื่องมือ

การประมวลผลจากสถิติข้างต้น ผู้วิจัยได้ตรวจสอบค่า KMO ดังแสดงผลการวิเคราะห์องค์ประกอบในตารางที่ 4 ซึ่งผู้วิจัยได้จำแนกการวิเคราะห์เป็น 3 กลุ่ม ตามลักษณะของทฤษฎีที่สนับสนุนกรอบแนวคิดและความสัมพันธ์ระหว่างปัจจัยประกอบด้วย กลุ่มที่หนึ่ง (การรับรู้ประโยชน์ (PU) การสนับสนุนของผู้บริหารระดับสูง (ES) การรับรู้ความสามารถของตนเอง (PSE) การมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps (TS) ลักษณะของโครงการแบบแอจไจล์

(APC) และการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (AU)) กลุ่มที่สอง (ปัจจัยความเข้ากันได้ระหว่างงานกับเทคโนโลยี (TTF)) และกลุ่มที่สาม (ความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps (ICD))

ตารางที่ 4 ค่า KMO และ Total % Variance

กลุ่มตัวแปร	ค่า KMO	Total % Variance
กลุ่มที่หนึ่ง	0.845	62.394
กลุ่มที่สอง	0.766	52.023
กลุ่มที่สาม	0.771	70.126

หลังจากผู้วิจัยได้วิเคราะห์องค์ประกอบ และตัดตัวแปรเป็นที่เรียบร้อยแล้ว จึงได้วิเคราะห์ความเที่ยงของเครื่องมือด้วยค่าสัมประสิทธิ์ครอนบาคแอลฟา ดังแสดงในตารางที่ 5

ตารางที่ 5 ค่าสัมประสิทธิ์ครอนแบคแอลฟา

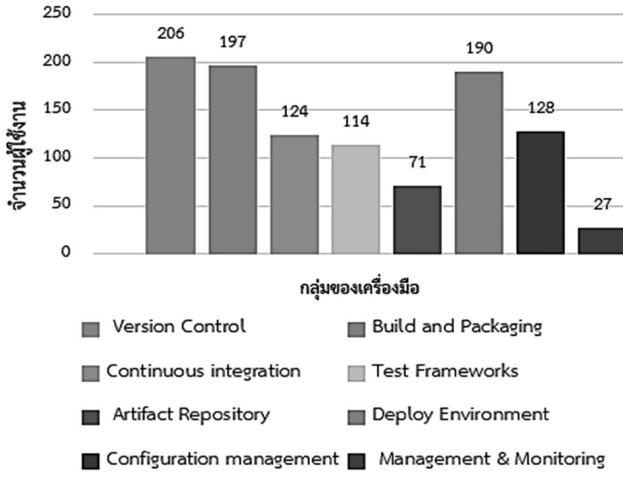
ตัวแปร	ครอนบาคแอลฟา	ข้อคำถามจำนวน
ความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps (ICD)	0.855	4
การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (AU)	0.790	3
การรับรู้ความสามารถของตนเอง (PSE)	0.856	4
การรับรู้ประโยชน์ (PU)	0.830	5
ปัจจัยความเข้ากันได้ระหว่างงานกับเทคโนโลยี (TTF)	0.768	5
การมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps (TS)	0.817	5
ลักษณะของโครงการแบบแอจไจล์ (APC)	0.725	4
การสนับสนุนของผู้บริหารระดับสูง (ES)	0.787	5

5.2 การวิเคราะห์ทางประชากรศาสตร์ของกลุ่มตัวอย่าง

ข้อมูลทั่วไปของกลุ่มตัวอย่าง ส่วนใหญ่อยู่ในองค์กรที่มีพนักงานจำนวน 200 คนขึ้นไป คิดเป็นร้อยละ 63.94 องค์กรประกอบธุรกิจผู้ผลิตหรือพัฒนาซอฟต์แวร์ คิดเป็นร้อยละ 61.06 รองลงมาเป็นธนาคารและสถาบันการเงิน คิดเป็นร้อยละ15.87 ผู้ตอบแบบสอบถามส่วนใหญ่มีประสบการณ์ทำงานในองค์กรที่ใช้แนวคิดการพัฒนาซอฟต์แวร์แบบแอจไจล์มาก่อน เป็นระยะเวลา 5 -10 ปี คิดเป็นร้อยละ 55 เป็นระดับหัวหน้าที่มีอำนาจในการตัดสินใจสูง คิดเป็นร้อยละ 40.87 รองลงมาคือ ระดับปฏิบัติการที่สามารถให้คำแนะนำในการตัดสินใจได้ คิดเป็นร้อยละ 39.90

ด้านการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ ผู้ตอบแบบสอบถามส่วนใหญ่มีประสบการณ์ในการทำโครงการน้อยกว่า 10 โครงการ คิดเป็นร้อยละ 60.10 ใช้งานแนวคิดแอจไจล์อยู่ประมาณ 1-20% ของโครงการที่รับผิดชอบทั้งหมด องค์กรส่วนใหญ่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มาแล้ว 1-2 ปี คิดเป็นร้อยละ 37.02 เมื่อพิจารณาลงไปในรายละเอียดเกี่ยวกับระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ที่ใช้ ผู้ตอบแบบสอบถามใช้วิธีการ (Method) ส่วนใหญ่คือ Scrum คิดเป็นร้อยละ 91.35 ในส่วนของวิธีปฏิบัติงาน ส่วนใหญ่ใช้วิธีปฏิบัติคือ Daily Stand-up Meetings ร้อยละ 85.10 รองลงมาคือ Iteration/sprint Planning คิดเป็นร้อยละ 78.85 และชุดเครื่องมือที่นิยมใช้ได้แก่ Task Board และ Unit Test Tool คิดเป็นร้อยละ 66.83 และ 48.56 ตามลำดับ

ด้านการใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ผู้ตอบแบบสอบถามส่วนใหญ่เคยใช้งานแนวคิด DevOps เพียง 1-20 % ของโครงการทั้งหมด คิดเป็นร้อยละ 44.71 และรองลงมาคือ ผู้ที่ไม่ได้ใช้งานแนวคิดดังกล่าว คิดเป็นร้อยละ 28.37 องค์กรโดยส่วนใหญ่นำแนวคิดดังกล่าวเข้ามาใช้เป็นระยะเวลาน้อยกว่า 1 ปี คิดเป็นร้อยละ 34.62 และรองลงมาคือ องค์กรไม่เคยใช้งานแนวคิด DevOps คิดเป็นร้อยละ 28.37 ทั้งนี้ชุดเครื่องมือตามแนวคิด DevOps ที่มีการใช้งานอยู่ โดยส่วนใหญ่ จำแนกตามทีมพัฒนาซอฟต์แวร์ ได้แก่ กลุ่มของ Version Control คิดเป็นร้อยละ 99.04 และจำแนกตามกลุ่มทีมปฏิบัติการ ได้แก่ กลุ่มของ Deploy Environment คิดเป็นร้อยละ 91.35 ดังแสดงในภาพที่ 11



ภาพที่ 11 แผนภูมิการใช้งานชุดเครื่องมือตามแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

5.3 การทดสอบสมมติฐาน

การทดสอบสมมติฐานโดยใช้วิเคราะห์การถดถอยแบบพหุคูณ แบ่งการวิเคราะห์ออกเป็น 2 ส่วน ได้แก่ ส่วนที่หนึ่งคือ การวิเคราะห์อิทธิพลของลักษณะของโครงการแบบแอจไจล์ และการมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ที่มีต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยี ซึ่งตัวแปรอิสระ อธิบายความแปรปรวนของตัวแปรตามได้ร้อยละ 23.9 (R^2) ดังแสดงในตารางที่ 6

ตารางที่ 6 ผลการวิเคราะห์การถดถอยแบบปกติสำหรับปัจจัยที่ส่งผลต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยี

Model		Unstandardized Coefficients	Standardized Coefficients	*t*	Sig.
		β	Beta		
1	(Constant)	4.658E-16		0.000	1.000
	การมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานในแนวความคิด DevOps (TS)	0.380	0.380	6.233	0.000**
	ลักษณะของโครงการแบบแอจไจล์ (APC)	0.308	0.308	5.054	0.000**

หมายเหตุ $^{**}p < 0.01$

การทดสอบสมมติฐานที่ 2 พบว่า ลักษณะของโครงการแบบแอจไจล์ ส่งผลต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยีที่ระดับนัยสำคัญ p = 0.000 โดยมีค่าสัมประสิทธิ์ถดถอย (β) คือ 0.308 และ**การทดสอบสมมติฐานที่ 3** พบว่า การมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ส่งผลต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยีที่ระดับนัยสำคัญ p = 0.000 โดยมีค่าสัมประสิทธิ์ถดถอย คือ 0.380

ส่วนที่สองเป็นการวิเคราะห์ความสัมพันธ์ระหว่างการรับรู้ประโยชน์ ความเข้ากันได้ระหว่างงานกับเทคโนโลยี การรับรู้ความสามารถของตนเอง การสนับสนุนของผู้บริหารระดับสูง และการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ กับความตั้งใจใช้งานแนวคิด DevOps ซึ่งปัจจัยเหล่านี้ อธิบายความแปรปรวนของตัวแปรตามได้ร้อยละ 48.3 (R^2) ดังแสดงในตารางที่ 7

ตารางที่ 7 ผลการวิเคราะห์การถดถอยแบบปกติสำหรับปัจจัยที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

Model		Unstandardized Coefficients	Standardized Coefficients	*t*	Sig.
		β	Beta		
1	(Constant)	-4.306E-16		0.000	1.000
	การรับรู้ประโยชน์ (PU)	0.358	0.358	5.685	0.000**
	การรับรู้ความสามารถตนเอง (PSE)	0.330	0.330	6.201	0.000**
	การสนับสนุนของผู้บริหารระดับสูง (ES)	0.104	0.104	1.992	0.048*
	การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (AU)	-0.009	-0.009	-.0170	0.865*
	ความเข้ากันได้ระหว่างงานกับเทคโนโลยี (TTF)	0.268	0.268	4.033	0.000**

หมายเหตุ $*p < 0.05$, $**p < 0.01$

การทดสอบสมมติฐานที่ 1 พบว่า การสนับสนุนของผู้บริหารระดับสูงส่งผลต่อความตั้งใจยอมรับแนวคิด DevOps ที่ระดับนัยสำคัญ $p = 0.048$ โดยมีค่าสัมประสิทธิ์ถดถอย คือ 0.104 **การทดสอบสมมติฐานที่ 4** พบว่า การรับรู้ความสามารถตนเองส่งผลต่อความตั้งใจยอมรับแนวคิด DevOps ที่ระดับนัยสำคัญ $p = 0.000$ โดยมีค่าสัมประสิทธิ์ถดถอย คือ 0.330 **การทดสอบสมมติฐานที่ 5** พบว่า การรับรู้ประโยชน์ส่งผลต่อความตั้งใจยอมรับแนวคิด DevOps ที่ระดับนัยสำคัญ $p = 0.000$ โดยมีค่าสัมประสิทธิ์ถดถอย คือ 0.358 **การทดสอบสมมติฐานที่ 6** พบว่า การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ไม่ส่งผลต่อความตั้งใจยอมรับแนวคิด DevOps ที่ระดับนัยสำคัญ $p = 0.865$ และ**การทดสอบสมมติฐานที่ 7** พบว่า ความเข้ากันได้ระหว่างงานกับเทคโนโลยีส่งผลต่อความตั้งใจยอมรับแนวคิด DevOps อย่างมีนัยสำคัญที่ $p = 0.000$ โดยมีค่าสัมประสิทธิ์ถดถอย คือ 0.268 จึงการสรุปผลการทดสอบสมมติฐานดังแสดงในตารางที่ 8

ตารางที่ 8 สรุปผลการทดสอบสมมติฐานทางสถิติโดยยึดหลักสมมติฐานทางเลือก (Alternative Hypothesis) ที่ระดับนัยสำคัญ 0.05

สมมติฐานทางงานวิจัย	สรุปผลการทดสอบสมมติฐานทางสถิติ
H1	มีนัยสำคัญ
H2	มีนัยสำคัญ
H3	มีนัยสำคัญ
H4	มีนัยสำคัญ
H5	มีนัยสำคัญ
H6	ไม่มีนัยสำคัญ
H7	มีนัยสำคัญ

6. การอภิปรายผลการวิจัย

จากผลวิจัยข้างต้นสรุปได้ว่า การมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps และลักษณะของโครงการแบบแอจไจล์เป็นตัวกำหนดความเข้ากันได้ระหว่างงานกับเทคโนโลยี นอกจากนี้ยังมีตัวแปรที่มีอิทธิพลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps อย่างมีนัยสำคัญประกอบด้วย การสนับสนุนของผู้บริหารระดับสูง การรับรู้ความสามารถของตนเอง การรับรู้ประโยชน์ และความเข้ากันได้ระหว่างงานกับเทคโนโลยี

เมื่อพิจารณาค่าสัมประสิทธิ์ถดถอยจากตารางที่ 6 พบว่า มิติด้านเทคนิคเกี่ยวกับปัจจัยการมีเครื่องมือทางเทคโนโลยีที่สนับสนุนการดำเนินงานตามแนวคิด DevOps มีอิทธิพลต่อความเข้ากันได้ระหว่างงานกับเทคโนโลยีมากกว่ามิติด้านโครงการ เกี่ยวกับปัจจัยลักษณะของโครงการแบบแอจไจล์ และเมื่อพิจารณาค่าสัมประสิทธิ์ถดถอยในตารางที่ 7 แสดงให้เห็นว่า มิติด้านบุคคล เกี่ยวกับปัจจัยการรับรู้ประโยชน์ ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มากที่สุด รองลงมาคือ การรับรู้ความสามารถของตนเอง ความเข้ากันได้ระหว่างงานกับเทคโนโลยี และการสนับสนุนของผู้บริหารระดับสูง ตามลำดับ

สำหรับมิติด้านกระบวนการ เกี่ยวกับปัจจัยการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์เป็นปัจจัยที่ไม่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps อธิบายได้จากข้อมูลเชิงพรรณนาของผู้ตอบแบบสอบถาม ที่สะท้อนว่ากลุ่มตัวอย่างใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ส่วนใหญ่เริ่มใช้งานเพียง 1-2 ปี เท่านั้น คิดเป็นร้อยละ 38.46 ดังตารางที่ 9

อีกทั้งองค์กรโดยส่วนใหญ่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ในอัตราเฉลี่ย 1-20% เมื่อเทียบกับการใช้กระบวนการพัฒนาซอฟต์แวร์ทั้งองค์กร คิดเป็นร้อยละ 37.02 ดังตารางที่ 10 รวมถึงผู้ตอบแบบสอบถามส่วนใหญ่นำแนวคิด DevOps มาใช้งานน้อยกว่า 1 ปี และบางส่วนก็ไม่ได้นำแนวคิดดังกล่าวมาใช้งานเลย จึงอาจยังไม่เล็งเห็นถึงความเกี่ยวข้องของระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ที่มีต่อแนวคิด DevOps และยังไม่เห็นถึงความสำคัญของการนำแนวคิด DevOps มาประยุกต์ใช้งาน ดังตารางที่ 11 ทั้งนี้ผลการวิจัยในหัวข้อดังกล่าวสอดคล้องกับผลงานวิจัยของบริษัท CA Technology (2560) ที่สำรวจบริษัทพัฒนาซอฟต์แวร์ในประเทศไทยนำเสนอว่า ภาคธุรกิจในประเทศไทยมีความก้าวหน้าในการใช้งานแนวคิด DevOps ในระดับสูง มากกว่าประเทศอื่นในภูมิภาคเอเชียแปซิฟิกและญี่ปุ่น โดยคิดเป็น 6 ใน 10 บริษัท เมื่อเทียบกับบริษัทอื่นในภูมิภาคเพียงร้อยละ 38 แต่บริษัทและองค์กรในประเทศไทยได้นำระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มาใช้งานเพียงร้อยละ 35 เท่านั้น ซึ่งแสดงให้เห็นว่า องค์กรในประเทศไทยได้เริ่มนำแนวคิดของ DevOps เข้ามาใช้งานแม้ว่าองค์กรจะไม่ได้ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ก็ตาม อีกทั้งผู้ตอบแบบสอบถามได้เสนอแนะว่าการนำแนวคิด DevOps มาใช้งานไม่จำเป็นต้องใช้งานแอจไจล์มาก่อน ขึ้นอยู่ว่าต้องการนำแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ไปเติมเต็มในส่วนใดของการพัฒนาซอฟต์แวร์ นอกจากนี้การนำแนวคิดดังกล่าวไปใช้งานนอกจากปัจจัยที่ศึกษาในข้างต้นแล้ว จำเป็นต้องปรับแนวคิดของทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการในการทำงานร่วมกันอีกด้วย

ตารางที่ 9 การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มุมมององค์กร

ประเภทข้อมูล	จำนวน	ร้อยละ
ระยะเวลาที่องค์กรใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ตั้งแต่เริ่มต้นจนถึงปัจจุบัน		
< 1 ปี	51	24.52
1-2 ปี	80	38.46
2-3 ปี	33	15.87
3-4 ปี	14	6.73
>5 ปี	30	14.42
รวม	208	100.00

ตารางที่ 10 การใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มุมมององค์กร

ประเภทข้อมูล	จำนวน	ร้อยละ
อัตราเฉลี่ยจำนวนโครงการในองค์กรทั้งหมด ที่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์		
1-20%	77	37.02
21-40%	27	12.98
41-60%	29	13.94
60-80%	36	17.31
80-100%	39	18.75
รวม	208	100.00

ตารางที่ 11 การใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ในมุมมององค์กร

ประเภทข้อมูล	จำนวน	ร้อยละ
ระยะเวลาที่องค์กรใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ตั้งแต่เริ่มต้นจนถึงปัจจุบัน		
ไม่ใช้เลย	59	28.37
< 1 ปี	72	34.62
1-2 ปี	41	19.71
2-3 ปี	36	17.31
3-4 ปี	-	-
>5 ปี	-	-
รวม	208	100.00

7. การนำไปใช้ประโยชน์

งานวิจัยนี้สามารถนำไปประยุกต์ในองค์กรที่มีการพัฒนาซอฟต์แวร์ ที่ต้องการนำแนวคิด DevOps เข้าไปใช้เพื่อเพิ่มความต่อเนื่องให้แก่การดำเนินงานพัฒนาและส่งมอบซอฟต์แวร์ หากพิจารณาจากปัจจัยที่ส่งผลต่อความตั้งใจนำแนวคิด DevOps ไปใช้พบว่า มิติด้านบุคคล เกี่ยวกับปัจจัยการรับรู้ประโยชน์มีอิทธิพลต่อความตั้งใจใช้งานแนวคิด DevOps มากที่สุด องค์กรจึงควรชี้ให้บุคลากรเห็นถึงประโยชน์ของแนวคิด DevOps ซึ่งจะช่วยเพิ่มศักยภาพในการทำงานด้านการส่งมอบซอฟต์แวร์ให้รวดเร็วขึ้นและตรงความต้องการผู้ใช้ได้มากขึ้น อีกทั้งยังช่วยสร้างความต่อเนื่องในกระบวนการพัฒนาซอฟต์แวร์ให้ได้มากขึ้น รวมถึงประสานการทำงานระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการให้ราบรื่นมากยิ่งขึ้น นอกจากนี้องค์กรควรเพิ่มการรับรู้ความสามารถในการเรียนรู้และสร้างความเข้าใจเกี่ยวกับแนวคิดดังกล่าวให้แก่บุคลากร เพื่อให้บุคลากรมีความมั่นใจที่จะศึกษาและนำแนวคิดดังกล่าวเข้ามาประยุกต์ใช้งานได้

ทั้งนี้กลุ่มตัวอย่างส่วนใหญ่มีความเห็นว่า ความเข้ากันได้ระหว่างงานกับเทคโนโลยีเป็นสิ่งที่สำคัญ โดยควรเลือกชุดเครื่องมือในการดำเนินงานให้สอดคล้องกับกระบวนการทำงานด้วย ซึ่งเครื่องมือตามแนวคิดของ DevOps จะช่วยให้แต่ละทีมสามารถทำงานได้อย่างสะดวก และเป็นอัตโนมัติขึ้น ก่อให้เกิดความต่อเนื่องในการดำเนินโครงการ รวมถึงเครื่องมือดังกล่าวที่หลากหลายสามารถทำงานร่วมกับระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ ซึ่งองค์กรใช้งานอยู่แล้วเดิมได้ดี เนื่องจากรองรับลักษณะโครงการที่มีการเปลี่ยนแปลงบ่อยและตอบสนองต่อความต้องการทางธุรกิจในปัจจุบันได้อย่างรวดเร็ว

แม้ว่ามิติด้านองค์กร เกี่ยวกับปัจจัยการสนับสนุนของผู้บริหารจะไม่ใช่ปัจจัยที่มีอิทธิพลมากที่สุด แต่เป็นปัจจัยที่ผู้ตอบแบบสอบถามส่วนใหญ่ให้ความเห็นตรงกันว่า เป็นสิ่งที่ขาดไม่ได้ในการเริ่มต้นโครงการ เนื่องจากการจัดการโครงการนั้น มีผู้เกี่ยวข้องหลายฝ่าย ตั้งแต่ทีมพัฒนาซอฟต์แวร์ ทีมทดสอบระบบ ทีมประกันคุณภาพ ทีมปฏิบัติการ ตลอดจนผู้ใช้งาน ผู้บริหารจึงควรเป็นผู้สื่อสาร และสร้างความเข้าใจที่ดีแก่ทีมงานทุกฝ่าย และควรจัดสรรทรัพยากรในด้านต่างๆ เช่น บุคลากร เวลา เครื่องมือ และการฝึกอบรม ซึ่งจะช่วยให้การนำแนวคิดใหม่ๆ อาทิ DevOps มาใช้งานในองค์กรได้ประสบความสำเร็จ

8. สรุปผลการวิจัย และงานวิจัยในอนาคต

งานวิจัยนี้ศึกษาถึงอิทธิพลของปัจจัยในมิติต่างๆ ที่มีผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps โดยใช้วิธีการศึกษาเชิงปริมาณ ใช้แบบสอบถามเป็นเครื่องมือ ผลการวิจัยพบว่า มิติด้านบุคคล เกี่ยวกับปัจจัยการรับรู้ประโยชน์ส่งผลต่อความตั้งใจใช้งานแนวคิด DevOps มากที่สุด รองลงมาคือ การรับรู้ความสามารถของตนเอง ความเข้ากันได้ระหว่างงานกับเทคโนโลยี และการสนับสนุนของผู้บริหารระดับสูง ตามลำดับ ทั้งนี้ความเข้ากันได้ระหว่างงานกับเทคโนโลยีได้รับ

อิทธิพลมาจากมิติด้านโครงการเกี่ยวกับปัจจัยลักษณะโครงการแบบแอจไจล์ และมิติด้านเทคนิคเกี่ยวกับปัจจัยการมีเครื่องมือทางเทคโนโลยีที่สนับสนุน แต่มิติด้านกระบวนการ เกี่ยวกับปัจจัยการใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์นั้น ไม่ส่งผลต่อความตั้งใจในการใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

ข้อจำกัดของงานวิจัยนี้คือ ความเข้าใจที่ตรงกันต่อแนวคิด DevOps เนื่องจาก DevOps เป็นแนวคิดที่ค่อนข้างใหม่ ทำให้ผู้วิจัยจำเป็นต้องใส่การอธิบายจำนวนมากในแบบสอบถาม และต้องมีคำถามเพื่อตรวจสอบความเข้าใจเกี่ยวกับแนวคิด ซึ่งอาจส่งผลต่อสมาธิของผู้ตอบ แม้ว่าผู้วิจัยจะอาศัยปัจจัยความสำเร็จในออกแบบกรอบงานวิจัยครั้งนี้ แต่ผู้วิจัยศึกษาเพียงความตั้งใจใช้งานแนวคิด DevOps เท่านั้น จากเหตุผลข้างต้นที่ว่า แนวคิดดังกล่าวยังไม่ค่อยแพร่หลายในวงการพัฒนาซอฟต์แวร์หรือมีการเริ่มต้นใช้งานเพียงบางองค์กรในประเทศไทย ทำให้ยังไม่สามารถวัดถึงความสำเร็จของโครงการได้ ส่วนด้านกลุ่มตัวอย่างของงานวิจัยนี้เก็บข้อมูลจากคนไทยเท่านั้น อีกทั้งองค์กรของผู้ตอบแบบสอบถามมีความหลากหลาย มีทั้งใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ทั้งองค์กร และเพียงบางโครงการ จึงทำให้ประสบการณ์ ความรู้ ความเข้าใจ ที่เกี่ยวข้องกับแนวคิดแอจไจล์และ DevOps แตกต่างกัน

งานวิจัยในอนาคต ควรกำหนดระยะเวลาขั้นต่ำที่ผู้ตอบแบบสอบถามเคยใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์มาก่อน หรือเพิ่มเติมกลุ่มตัวอย่างผู้ที่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ และผู้ที่ไม่ได้ใช้ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์หรือใช้แบบอื่น เพื่อศึกษาเปรียบเทียบ รวมถึงสามารถขยายต่อไปถึงการใช้งานจริงหรือวัดความสำเร็จของโครงการที่เมื่อได้นำแนวคิด DevOps มาใช้งานได้อีกด้วย อีกทั้งยังสามารถเพิ่มเติมปัจจัยในมิติอื่นๆ ตามกรอบแนวคิดการวิจัย เช่น ปัจจัยขัดขวาง ซึ่งจะช่วยเพิ่มความน่าสนใจและสะท้อนความคิดของผู้ปฏิบัติงานจริงได้ดียิ่งขึ้น

เอกสารอ้างอิง

กชพรรณ ออกดีเลิศ. (2560). DevOps. จาก http://www.swpark.or.th/index.php?option=com_training&view=course&id=253&format=raw&Itemid=3

กรมการจัดหางาน. (2558). แนวโน้มการพัฒนาเทคโนโลยีสารสนเทศและการสื่อสารในอนาคต. ***แผนแม่บทเทคโนโลยีสารสนเทศและการสื่อสารของกรมการจัดหางาน พ.ศ. 2558-2562,*** 89.

กัลยา วานิชย์บัญชา. (2551). ***การวิเคราะห์สถิติขั้นสูงด้วย SPSS for Windows.*** กรุงเทพฯ: จุฬาลงกรณ์มหาวิทยาลัย.

คัมภีร์เทพ IT. (2560). รู้จัก “DevOps” ให้มากขึ้น “ตำแหน่ง” ที่องค์กรต่างมองหา. จาก https://www.techstarthailand.com/blog/detail/why-DevOps-skills-continue-to-be-in-high-demand/127

ชาญชัย อรรคผาติ. (2557). ***ปัจจัยที่ส่งผลต่อทัศนคติในการยอมรับในเทคโนโลยีคลาวด์คอมพิวติ้งเพื่อประยุกต์ใช้ในการให้บริการระบบบัญชีออนไลน์ สำหรับวิสาหกิจขนาดกลางและขนาดย่อมในมุมมองของผู้ทำบัญชี.*** (วิทยานิพนธ์ปริญญามหาบัณฑิต). มหาวิทยาลัยธุรกิจบัณฑิตย์, คณะการบัญชี.

ธนาชัย บูรณะวัฒนากูล, ศรี แก้วงาม,ปัญญา นราพันธ์, และ พระมหาอรรถพล วชิรปญฺโญ. (2555). Agile Method: วิธีการพัฒนาซอฟท์แวร์สำหรับปัจจุบันและอนาคต. จาก https://reg7.pwa.co.th/kmr7/?p=453

นงลักษณ์ วิรัชชัย. (2555). การกำหนดขนาดตัวอย่างและสถิติวิเคราะห์ใหม่ๆ ที่น่าสนใจ. ใน ***การนำเสนอผลงานวิจัยแห่งชาติ 2555.*** จาก http://edserv.nida.ac.th/main/images/download/3.pdf

ปริวรรต ศักดิ์นิมิตวงศ์. (2560). Learn DevOps ตอนที่ 2: DevOps คืออะไร ? จาก https://medium.com/@pariwat_s/learn-devops-ตอนที่-2-devops-คืออะไร-18ac48d73625

พงศกร ภูแสนคำ. (2559). DEVOPS (ภาคต่อของ LEAN CANVAS) สำหรับ TECH STARTUP ที่ต้องรู้. จาก http://coder.in.th/?p=135

พัฒนพงษ์ เชิดทอง. (2559). ทำความรู้จัก Docker และ Software Container. จาก https://medium.com/thothzocial-engineering/ทำความรู้จัก-docker-และ-software-container-c6338629da11

รสริน ศรีริกานนท์. (2552). การวิเคราะห์องค์ประกอบ (Factor Analysis). จาก http://www.saruthipong.com/port/document/299-705/299-705-8.pdf

สำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ. (2559). Workshop#1: DevOps and Tools. จาก http://sipatechmeetup.com/events/e/workshop1-devops-and-tools/

สุชาดา เกยุระ. (2553). ***การประยุกต์ใช้ Technology Acceptance Model และ Task – Technology Fit ใน ELearning.*** มหาวิทยาลัยธรรมศาสตร์, วิทยาลัยนวัตกรรม.

Sukoom2001. (2560). สรุป AWS 2017 KEYNOTE: ตลาด CLOUD เติบโตอย่างรวดเร็วในไทย กับการใช้ CLOUD ในรูปแบบใหม่ๆ ที่หลากหลาย. จาก https://thaitechnewsfeed.com/2017/05/30/สรุป-aws-2017-keynote-ตลาด-cloud-เติบโตอย่าง

Techtalkthai. (2559). ทำความรู้จักกับ Microservices สถาปัตยกรรมระบบที่ทั้งนักพัฒนาและผู้ดูแลระบบควรรู้จัก. จาก https://www.techtalkthai.com/introduction-to-microservices-architecture/

Winggundamth. (2560). บริษัท Opsta ประกาศรับสมัครขยายทีม OpenStack และ DevOps. จากhttps://www.blognone.com/node/96191

References

Airaj, M. (2015). Enable cloud DevOps approach for industry and higher education. ***WILEY.***

Ajzen, I. (1985). From intentions to actions: A theory of planned behavior. 11-39.

Aokdeelert, K. (2017). DevOps. Retrieved from http://www.swpark.or.th/index.php?option=com_training&view=course&id=253&format=raw&Itemid=3 (in Thai)

Arkaphati, C. (2014). Factor Affecting the Attitude in Adoption of Cloud Computing Technology Application for Online Accounting System Services in Small and Medium Enterprise (SMEs) from Accountants' perspective. (Master's thesis). Dhurakij Pundit University, Accountancy Program Faculty. (in Thai)

Awad, M. A. (2005). ***A Comparison between Agile and Traditional Software Development Methodologies.***

Babar, Z., Lapouchnian, A., & Yu, E. (2011). Modeling DevOps Deployment Choices using Process Architecture Design Dimensions.

Baleghi-Zadeh, S., Ayub, A. F. M., Mahmud, R., & Daud, S. M. (2014). An Assessment of Task-Technology Fit, Subjective Norm and Internet Experience of Learning Management System in Views of Malaysian Higher Education Students. ***International Journal of Information and Communication Technology Research,*** 4(4), 142-146.

Buranawattanakul, T., Keawngam, S., Narapan, P., & Wachirapanyo, A. (2012). Agile Method: How to develop software for present and future. Retrieved from https://reg7.pwa.co.th/kmr7/?p=453 (in Thai)

Chan, F. K. Y., & Thong, J. Y. L. (2008). Acceptance of agile methodologies: A critical review and conceptual framework. Acceptance of agile methodologies: ***A critical review and conceptual framework, Decision Support Systems,*** 803–814.

Cherthong, P. (2016). Getting to Know the Docker and Software Container. Retrieved from https://medium.com/thothzocial-ngineering/%E0%B8%97%E0%B8%B3%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-docker-%E0%B9%81%E0%B8%A5%E0%B8%B0-software-container-c6338629da11 (in Thai)

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. ***Journal of Systems and Software,*** 81(6), 961-971. doi:10.1016/j.jss.2007.08.020.

Cois, C. A., Yankel, J., & Connell, A. (2014). Modern DevOps: Optimizing Software Development Through Effective System Interactions. ***IEE computer Society.***

Department of Employment. (2015). Technology Trends. ***Information and Communications Technology Master Plan of the Department of Employment,*** 2015-2019, 89. (in Thai)

Deshpande, A. (2016). DevOps" an Extension of Agile Methodology–How It will Impact QA? Retrieved from http://www.softwaretestinghelp.com/devops-and-software-testing/

Diéguez, M., Sepúlveda, S., & Cachero, C. (2012). UMAM-Q: An Instrument to Assess the Intention to Use Software Development Methodologies.

Dingsøy, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. ***Information and Software Technology,*** 56-60.

DuMoulin, T. (2017). Critical Success Factors for DevOps. ***Pink Elephant.***

Engel, P., & Schweimler, B. (2016). Design and Implementation of a Modern Automatic Deformation Monitoring System. ***Journal of Applied Geodesy,*** 10(1), 79-85. doi:10.1515/jag-2015-0024

Farroha, B., & Farroha, D. (2014). ***A framework for managing mission needs, compliance, and trust in the devops environment.*** Paper presented at the Military Communications Conference (MILCOM), 2014 IEEE.

Fathema, N., Shannon, D., & Ross, M. (2015). Expanding the Technology Acceptance Model (TAM) to Examine Faculty Use of Learning Management Systems (LMSs) In Higher Education Institutions. ***MERLOT Journal of Online Learning and Teaching,*** 11(210-232).

Fiampolis, P., & Groll, J. (2016). Bringing together Dev and Ops. ***Training Journal,*** 13-15.

Fitzgerald, B., & Stol, K.-J. (2014). Continuous Software Engineering and Beyond: Trends and Challenges. ***ACM/IEEE.***

Gartner. (2015). Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations. Retrieved from http://www.gartner.com/newsroom/id/2999017

Goodhue, D. L., & Thomspson, R. L. (1995). Task-Technology Fit and Indivual Performance. ***MIS Quarterly,*** 213 - 236.

Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2010). Multivariate Data Analysis: New Jersey: Pearson Prentice Hall.

Hameed, T. A. E., Latif, M. A. E., & Kholief, S. (2016). Identify and Classify Critical Success Factor of Agile Software Development Methodology Using Mind Map***. (IJACSA) International Journal of Advanced Computer Science and Applications,*** 7.

Harb, Y., Noteboom, C., & Sarnikar, S. (2014). Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case. ***Journal of the Midwest Association for information Systems,*** 2015 (1).

Higgins, T. (2016). How Continuous Requirements Are Key to Gartner's DevOps Toolchain. Retrieved from http://www.blueprintsys.com/blueprint-devops/

Hoehle, H. (2011). ***CONSUMER INTENTIONS TO USE ELECTRONIC BANKING CHANNELS: THE ROLE OF TASK-CHANNEL FIT.*** (Doctor), Victoria University of Wellington.

Hovsepyan, A., & Landuyt, D. V. (2015). Prototizer: Agile on Steroids. ***ACM/IEEE,*** 51-60.

Kampeetep IT. (2017). Know the "DevOps" for more "positions" that organizations look for these. Retrieved from https://www.techstarthailand.com/blog/detail/why-DevOps-skills-continue-to-be-in-high-demand/127 (in Thai)

Kapadia, M. (2015). Comparing DevOps to traditional IT: Eight key differences. Retrieved from https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/comparing_devops_to_traditional_it_eight_key_differences?lang=en

Keyura, S. (2010). Extending the Technology Acceptance Model and the Task-Technology Fit in E-learning. Thammasat University, Technology Management College of Innovation. (in Thai)

Kim, T. T., Suh, Y. K., Lee, G., & Choi, B. G. (2010). Modelling roles of task-technology fit and self-efficacy in hotel employees' usage behaviours of hotel information systems. ***INTERNATIONAL JOURNAL OF TOURISM RESEARCH,*** 12(6), 709-725. doi:10.1002/jtr.787

Kornecki, A. J. (2007). ***SOFTWARE DEVELOPMENT TOOLS FOR SAFETY-CRITICAL, REAL-TIME SYSTEMS HANDBOOK*** (DOT/FAA/AR-06/35)

Krome, P. (2017). What is the difference between Monolith and n Layer? Retrieved from https://stackoverflow.com/questions/45661006/what-is-the-difference-between-monolith-and-n-layer

Kruis, S. (2014). ***Designing a metrics model for DevOps at Philips IT.*** (Master's thesis). Eindhoven University of Technology. Retrieved from https://pure.tue.nl/ws/files/47007244/785841-1.pdf

Lambert, T. (2011). ***CROSS SECTIONAL STUDY OF AGILE SOFTWARE DEVELOPMENT METHODS AND PROJECT PERFORMANCE.*** (DOCTOR BUSINESS ADMINISTRATION), Nova Southeastern University.

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo1, M. (2016). Towards DevOps in the Embedded Systems Domain: Why is It so Hard? ***Hawaii International Conference on System Sciences.***

McAvoy, J., Sammon, D., & Owens, I. (2007). A Simple Tool to Assist in Agile Methodology Adoption Decisions. ***Journal of Decision Systems,*** 16(4), 451-468. doi:10.3166/jds.16.451-468

Menard, P., Sweeney, B., & Shropshire, J. (2017). DevOps: An Innovation Attributes Perspective.

Menzel, G., & Macaulay, A. (2014). DevOps - The Future of Application Lifecycle Automation. A Capgemini Architecture Whitepaper – 2nd Edition, 22.

Meresca, P. (2015). From Monolithic Three-tiers Architectures to SOA Vs Microservices. Retrieved from https://thetechsolo.wordpress.com/2015/07/05/from-monolith-three-tiers-architectures-to-soa-vs-microservices/

Michelsen, J. (2013). Dysfunction Junction: A Pragmatic Guide to Getting Started with DevOps. ***CA Technologies.***

Mohamed, S. I. (2016). Software Release Management Evolution -Comparative Analysis across Agile and DevOps Continuous Delivery. ***International Journal of Advanced Engineering Research and Science (IJAERS),*** 3(6).

Mueller, E. (2016). What Is DevOps? Retrieved from https://theagileadmin.com/what-is-devops/

Olszewska, M., & Waldén, M. (2015). DevOps Meets Formal Modelling in High-Criticality Complex Systems. QUDOS.

Park, C., & Raven, A. (2015). INFORMATION QUALITY AS A DETERMINANT OF TASK-TECHNOLOGY FIT IN USING COMMUNICATION TECHNOLOGY FOR SIMPLE TASK. ***Issues in Information Systems,*** 16(1), 189-199.

Patwardhan, B. (2014). An Overview of DevOps. In D. D. Jana & D. A. Nair (Eds.), ***CSI communication knowledge Digest for IT Community*** (Vol. 38, pp. 14): Executive Secretary Mr. Suchit Gogwekar for Computer Society of India.

Poosankam, P. (2016). DEVOPS (The next step from LEAN CANVAS) for TECH STARTUP to know. Retrieved from http://coder.in.th/?p=135 (in Thai)

Right Scale. (2016). Right Scale 2016 STATE OF THE CLOUD REPORT: DevOps Trends. Retrieved from http://assets.rightscale.com/uploads/pdfs/rightscale-2016-state-of-the-cloud-report-devops-trends.pdf

Riley, C. (2014). Waterfall to Agile to DevOps: The State of Stagnant Evolution. Retrieved from https://devops.com/waterfall-agile-devops-state-stagnant-evolution/

Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on agile and lean usage in finish software industry. ***Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement,*** 139. doi:10.1145/2372251.2372275

Rosehosting. (2016). Physical server vs Virtual server: All you need to know. Retrieved from https://www.rosehosting.com/blog/physical-server-vs-virtual-server-all-you-need-to-know/#comments

Russo, N. L., Shams, S., & Fitzgerald, G. (2013). Exploring Adoption and Use of Agile Methods: A Comparative Case Study. ***Proceedings of the Nineteenth Americas Conference on Information Systems.***

Saknimitwong, P. (2017). Learn DevOps Chapter 2 : What's DevOps? Retrieved from https://medium.com/@pariwat_s/learn-devops-%E0%B8%95%E0%B8%AD%E0%B8%99%E0%B8%97%E0%B8%B5%E0%B9%88-2-devops-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-18ac48d73625 (in Thai)

Sarıtaş, M. T., Yıldız, E., & ŞENEL, H. C. (2015). Examining the Attitudes and Intention to Use Synchronous Distance Learning Technology among Pre-service Teachers: A Qualitative Perspective of Technology Acceptance Model. ***American Journal of Educational Research,*** 3, 17-25.

Schaller, A. E. (2016). DEVOPS TRANSFORMATION CHALLENGES FACING LARGE SCALE LEGACY SYSTEMS. ***ProQuest.***

Schmitt, N. (1996). Uses and Abuses of Coefficient Alpha ***Psychological Assessment,*** 8, 350-353.

Senapathi, M., & Srinivasan, A. (2011). Understanding Post-Adoptive Agile Usage-an Exploratory Cross-Case analysis. ***IEE computer Society.***

Software Industry Promotion Agency (Public Organization). (2016). Workshop#1: DevOps and Tools. Retrieved from http://sipatechmeetup.com/events/e/workshop1-devops-and-tools/ (in Thai)

Soto-Acosta, P., Ramayah, T., & Popa, S. (2013). EXPLAINING INTENTION TO USE AN ENTERPRISE RESOURCE PLANNING SYSTEM: A REPLICATION AND EXTENSION. ***Objašnjenje namjere primjene sustava za planiranje resursa poduzeća: ponavljanje i proširenje,*** 397-405.

Spec-India. (2015). From Waterfall to Agile to DevOps – A Cultural and Technological Shift. Retrieved from http://blog.spec-india.com/from-waterfall-to-agile-to-devops-a-cultural-and-technological-shift/

Sririkanon, R. (2009). Factor Analysis. Retrieved from http://www.saruthipong.com/port/document/299-705/299-705-8.pdf (in Thai)

Sukoom2001. (2017). AWS 2017 summary KEYNOTE: CLOUD markets are rapidly growing in Thailand, with the adoption of new CLOUD. Retrieved from https://thaitechnewsfeed.com/2017/05/30/%E0%B8%AA%0E0%B8%A3%E0%B8%B8%E0%B8%9B-aws-2017-keynote-%E0%B8%95%E0%B8%A5%E0%B8%B2%E0%B8%94-cloud-%E0%B9%80%E0%B8%95%E0%B8%B4%E0%B8%9A%E0%B9%82%E0%B8%95%E0%B8%AD%E0%B8%A2%E0%B9%88%E0%B8%B2%E0%B8%87 (in Thai)

Šumak, B., & Šorgo, A. (2016). The acceptance and use of interactive whiteboards among teachers: Differences in UTAUT determinants between pre- and post-adopters. ***Computers in Human Behavior,*** 64, 602-620. doi:10.1016/j.chb.2016.07.037

Techtalkthai. (2016). Get to know Microservices System architecture that both developers and administrator should know. Retrieved from https://www.techtalkthai.com/introduction-to-microservices-architecture (in Thai)

Thakurta, R., & Roy, R. (2012). Determinants of User Involvement in Software Projects. ***Hawaii International Conference on System Sciences.***

Tranter, L. (2016). emerginttechtrends. Retrieved from https://www.extremeuncertainty.com/emerginttechtrends/

Tripp, J. F., & Armstrong, D. J. (2014). Exploring the Relationship between Organizational Adoption Motives and the Tailoring of Agile Methods. ***Hawaii International Conference on System Science,*** 4799-4806.

Vanichbuncha, K. (2008). Advanced statistical analysis with SPSS for Windows. Bangkok: Chulalongkorn University. (in Thai)

Vijayasarathy, L., & Turk, D. (2010). Drivers of agile software development use: Dialectic interplay between benefits and hindrances. ***Information and Software Technology.***

Wang, W., Li, X., & Hsieh, J. J. P.-A. (2013). The Contingent Effect of Personal IT Innovativeness and IT Self-Efficacy on Innovative Use of Complex IT. ***Computer Information Systems Faculty Publications.***

Watts, S. (2017). DevOps vs Agile: What's the Difference and How Are They Related? Retrieved from http://www.bmc.com/blogs/devops-vs-agile-whats-the-difference-and-how-are-they-related/

White, J. (2016). Private vs. Public Cloud: What's the Difference? Retrieved from https://www.expedient.com/blog/private-vs-public-cloud-whats-difference/

Winggundamth. (2017). Opsta (Thailand) Announces expansion of OpenStack and DevOps Teams. Retrieved from https://www.blognone.com/node/96191 (in Thai)

Wiratchai, N. (2012). Sample size and new analytical statistics. In Thailand Research EXPO 2012. Retrieved from http://edserv.nida.ac.th/main/images/download/3.pdf (in Thai)

Young, D. K. (2013). ***PROJECT-METHOD FIT: EXPLORING FACTORS THAT INFLUENCE AGILE METHOD USE*** (Degree of DOCTOR OF PHILOSOPHY IN BUSINESS ADMINISTRATION), THE UNIVERSITY OF TEXAS AT SAN ANTONIO

Zwikael, O. (2008). Top management involvement in project management. A cross country study of the software industry.

ภาคผนวก

แบบสอบถามที่ใช้ในงานวิจัยแบบออนไลน์

https://docs.google.com/forms/d/e/1FAIpQLSdOnSGROsg-y0EjFcm1IWB6MHeavoV7_oyDXYn-4R3vznsmew/viewform?c=0&w=1

แบบสอบถามที่ใช้ในงานวิจัยแบบกระดาษ

THAMMASAT BUSINESS SCHOOL

MSMIS

MASTER OF SCIENCE PROGRAM
IN MANAGEMENT INFORMATION SYSTEMS

(QR Code สำหรับทำแบบ online ค่ะ)

โครงการปริญญาโทสาขาวิชาระบบสารสนเทศเพื่อการจัดการ
คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์

เรียน ท่านผู้ตอบแบบสอบถาม

แบบสอบถามนี้เป็นแบบสอบถามที่ใช้ในการศึกษาเรื่อง *"ปัจจัยที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps"* โดยผู้วิจัยขอความอนุเคราะห์จากท่านในการแสดงความคิดเห็นในด้านต่างๆ ที่เกี่ยวข้องกับปัจจัยในด้านต่างๆ ที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ข้อมูลที่ได้จากแบบสอบถามชุดนี้ จะใช้เพื่อการศึกษาค้นคว้าเท่านั้นคำตอบจะถือเป็นความลับ ที่ผู้วิจัยจะนำเสนอในภาพรวมเท่านั้น ซึ่งแบบสอบถามจะประกอบด้วย 6 ส่วน ดังนี้

ส่วนที่ 1: คำถามคัดกรองกลุ่มเป้าหมายที่เป็นบุคลากรในบริษัทพัฒนาซอฟต์แวร์ หรือหน่วยงานพัฒนาระบบขององค์กรระดับผู้จัดการขึ้นไปที่มี *"สิทธิ์ตัดสินใจ"* หรือ *"ส่วนร่วมตัดสินใจ"* หรือ "ให้คำแนะนำ" ในการนำการพัฒนาซอฟต์แวร์รูปแบบใหม่มาใช้งาน หรือ เลือกใช้เครื่องมือเข้ามาช่วยในการพัฒนาซอฟต์แวร์หรือไม่

ส่วนที่ 2: *การให้ความรู้เบื้องต้นเกี่ยวกับแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps และมีคำถามคัดกรองผู้ที่กลุ่มเป้าหมายเกี่ยวกับความรู้ความเข้าใจที่มีต่อแนวความคิดดังกล่าว **รบกวนผู้ตอบแบบสอบถามสละเวลาอ่าน เพื่อทำความเข้าใจต่อแนวคิดดังกล่าว เนื่องจากการตอบคำถามมีผลต่อการนำข้อมูลไปวิเคราะห์ค่ะ **

ส่วนที่ 3: คำถามเกี่ยวกับความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

ส่วนที่ 4: คำถามเกี่ยวกับการใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (Agile)

ส่วนที่ 5: คำถามเกี่ยวกับการใช้กลุ่มของเครื่องมือสำหรับผู้ที่ใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps (กรณีมีการใช้งานแนวคิดของ DevOps)

ส่วนที่ 6: ข้อมูลทั่วไปและข้อมูลเกี่ยวกับด้านความรับผิดชอบของงานของผู้ตอบแบบสอบถาม

ทางผู้วิจัยขอขอบพระคุณผู้ตอบแบบสอบถามทุกท่านที่สละเวลาในการตอบคำถาม ซึ่งจะเป็นส่วนสำคัญในการทำให้งานวิจัยครั้งนี้ประสบความสำเร็จได้ด้วยดี

แบบสอบถามการศึกษางานวิจัยเรื่อง ปัจจัยที่ส่งผลต่อความตั้งใจใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

ส่วนที่ 1: คำถามคัดกรองกลุ่มเป้าหมาย

1. องค์กรที่ท่านทำงานอยู่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Agile ใช่หรือไม่

☐ ใช่	☐ ไม่ใช่ (จบแบบสอบถาม)

2. ตำแหน่งงานหรือหน้าที่ที่รับผิดชอบของท่าน “มีสิทธิ์ตัดสินใจ” หรือ “มีส่วนร่วมในการตัดสินใจ” หรือ “ให้คำแนะนำ” ในการเลือกใช้การพัฒนาซอฟต์แวร์แบบใหม่ หรือ เลือกใช้เครื่องมือเข้ามาช่วยในการพัฒนาซอฟต์แวร์หรือไม่

☐ มี	☐ ไม่มี (จบแบบสอบถาม)

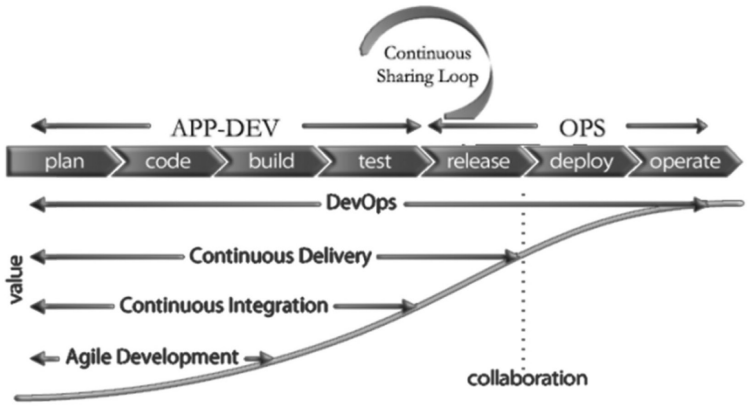
ส่วนที่ 2: เนื้อหาเกี่ยวกับแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

กระบวนการพัฒนาซอฟต์แวร์แบบดั้งเดิมที่เป็นที่รู้จักกันทั่วไป คือกระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall) เป็นกระบวนการที่ต้องทำแต่ละขั้นตอนให้สำเร็จก่อนจะทำในขั้นตอนต่อไป ประกอบด้วย ขั้นตอนการเก็บความต้องการ (Requirement) => ออกแบบ (Design) => พัฒนาระบบ (Code & Implement) => ทดสอบคุณภาพและใช้งานจริง (Verification) => ขั้นการบำรุงรักษาอย่างต่อเนื่อง (Maintenance) แต่เมื่อเทคโนโลยีและธุรกิจที่เปลี่ยนไปทำให้กระบวนการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก ไม่สามารถตอบสนองความต้องการของผู้ใช้งานหรือลูกค้าได้ทันต่อความต้องการ จึงมีการปรับปรุงกระบวนการพัฒนาซอฟต์แวร์ โดยจะแบ่งเป็นกระบวนงานย่อยๆ เพื่อความรวดเร็ว เพิ่มคุณภาพและการส่งมอบงานที่ดีขึ้น ในขั้นตอนการเก็บความต้องการ (Requirement) รวมถึงปรับปรุงความต้องการ การออกแบบ (Design) และพัฒนาระบบ (Code & Implement) ซึ่งเป็นขั้นตอนในส่วน ***“ต้นทาง”*** ในฝั่ง*ทีมพัฒนาซอฟต์แวร์* และเน้นสร้างการมีส่วนร่วมให้กับ

ผู้ใช้งานให้มากขึ้น ที่เรียกว่า **แอจไจล์ (Agile)** แต่อย่างไรก็ตามแอจไจล์ไม่สามารถตอบสนองต่อการเปลี่ยนแปลงที่เกิดขึ้นได้ทันตามความต้องการในขั้นตอน ***"ปลายทาง"*** ในฝั่ง*ทีมปฏิบัติการ* ซึ่งเป็นการส่งมอบซอฟต์แวร์ไปยังผู้ใช้งาน ให้ประสบปัญหาคอขวด ดังภาพที่ 1

ภาพที่ 1 ปัญหาคอขวดในฝั่งปฏิบัติการเมื่อนำระเบียบวิธีการพัฒนาซอฟแวร์แบบแอจไจล์มาใช้งาน
ที่มา: http://www.slideshare.net/CloudOps/agile-itil-cloud-mit-devops-in-die-zukunft

รวมถึงการเปลี่ยนแปลงที่เกิดขึ้นจะส่งผลกระทบต่อเสถียรภาพของซอฟต์แวร์และฮาร์ดแวร์ที่ทำงานอยู่เดิม ทำให้ปัจจุบันเกิดแนวคิดในการพัฒนาซอฟต์แวร์ที่เข้ามาเป็น***ส่วนเพิ่มเติม***จาก Agile ดังภาพที่ 2

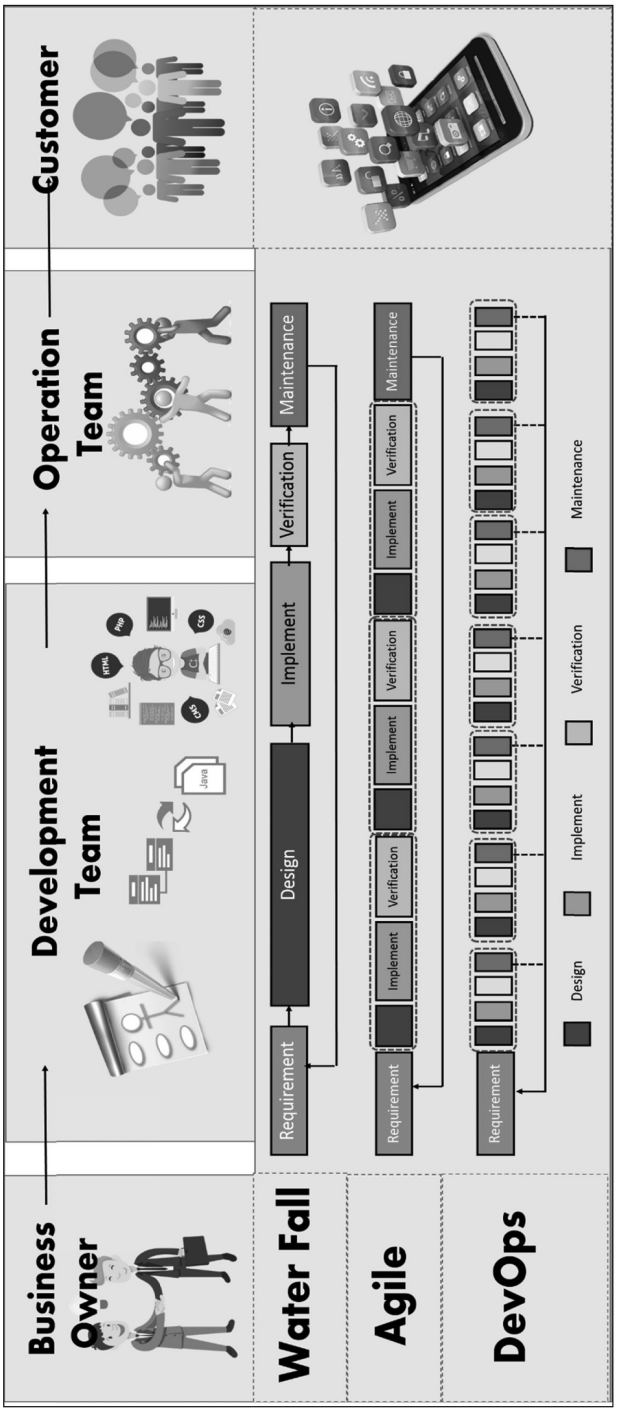


ภาพที่ 2 ขั้นตอนในแนวคิดการพัฒนาซอฟแวร์แบบ DevOps ที่เพิ่มเติมจาก Agile
ที่มา: https://www.collab.net/solutions/development-process.

ที่สามารถรองรับผลกระทบที่เกิดจากการเปลี่ยนแปลง และทำให้ส่งมอบซอฟต์แวร์ไปยังผู้ใช้งานปลายทางได้อย่างมีประสิทธิภาพและรวดเร็วขึ้น โดยจะนำเครื่องมือมาช่วยให้กระบวนการมีความเป็นอัตโนมัติมากขึ้น ซึ่งสร้างความต่อเนื่องในการทำงานให้แก่ทีมนักพัฒนาระบบ (Development Team) และทีมปฏิบัติการ (Operation Team) ที่เรียกว่า **DevOps** ทั้งนี้ประโยชน์ที่เกิดขึ้นจากแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ได้แก่

- การเพิ่มการติดต่อสื่อสารและการทำงานร่วมกันระหว่างทีมพัฒนาซอฟต์แวร์และทีมปฏิบัติการ
- ปรับปรุงคุณภาพในการทดสอบภายในสภาพแวดล้อมต่างๆ ก่อนการส่งมอบพัฒนาซอฟต์แวร์ (Deployment)
- การเพิ่มความถี่ในการส่งมอบ (Release) ซอฟต์แวร์
- ตอบสนองต่อความต้องการทางธุรกิจได้มากขึ้น
- เพิ่มความยืดหยุ่นในการดำเนินการตามความต้องการของลูกค้าตลอดกระบวนการพัฒนาซอฟต์แวร์
- ปรับปรุงความน่าเชื่อถือของคุณภาพซอฟต์แวร์ เป็นต้น

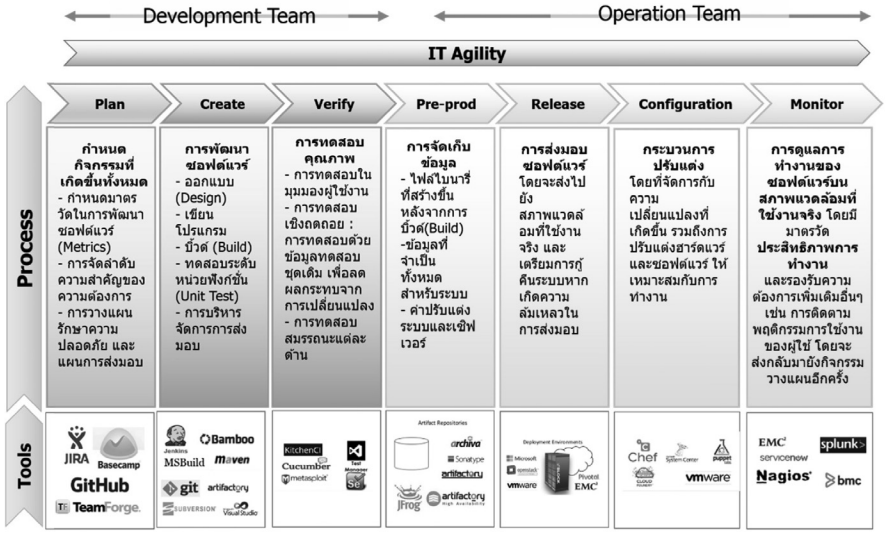
จากที่กล่าวมาข้างต้นจะแสดงให้เห็นถึงความแตกต่างกันระหว่าง 3 กระบวนการพัฒนาซอฟต์แวร์ ดังภาพที่ 3



ภาพที่ 3 การเปรียบเทียบระหว่าง วงจรการพัฒนาซอฟต์แวร์แบบขั้นน้ำตก (Waterfall Model) ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ (Agile) และ แนวความคิดการพัฒนาซอฟต์แวร์แบบ DevOps

ที่มา: ดัดแปลงจาก http://blog.spec-india.com/from-waterfall-to-agile-to-devops-a-cultural-and-technological-shift/

จากที่กล่าวในข้างต้นว่าแนวคิด DevOps เป็นส่วนที่เพิ่มเติมมาจากระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Agile ซึ่งได้เน้นการทำงานในส่วนของทีมพัฒนาซอฟต์แวร์ที่ติดต่อกับผู้ใช้งาน จึงมีกระบวนการที่สนับสนุนการเปลี่ยนแปลงในส่วนของความต้องการได้อย่างรวดเร็ว ในกระบวนการของการวางแผน (Plan) การพัฒนาซอฟต์แวร์ (Create) การทดสอบในมิติของนักพัฒนาซอฟต์แวร์ (Verify) แต่แนวคิดการพัฒนาซอฟแวร์แบบ DevOps จะสร้างความคล่องตัวให้แก่การดำเนินการทางเทคโนโลยี (IT Agility) ที่ครอบคลุมตั้งแต่ฝั่งซ้าย ที่รวมระเบียบวิธีการพัฒนาซอฟต์แวร์แอจไจล์และให้ความสำคัญต่อเนื่องไปถึงการเตรียมสภาพแวดล้อมต่างๆ (Pre-prod) เช่น UAT, Production เพื่อนำไปสู่การใช้งานจริง (Release) และปรับแต่งค่าให้เหมะสม (Configuration) รวมถึงติดตามผลการดำเนินงาน (Monitor) ประกอบด้วยกิจกรรมดังภาพที่ 4



ภาพที่ 4 แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps โดยสัมพันธ์กับกลุ่มของเครื่องมือสนับสนุนการสร้างความต่อเนื่องในกระบวนการทำงาน (DevOps Tool Chain)

ที่มา: https://infocus.emc.com/bart_driscoll/common-devops-tool-chains-pitfall

จากกระบวนการข้างต้นจะเห็นได้ว่า แนวคิดดังกล่าวว่ามีความต่อเนื่องของขั้นตอนต่างๆ ด้วยความเป็นอัตโนมัติ ส่งผลให้ซอฟต์แวร์มีคุณภาพ รวมถึงสามารถรองรับการเปลี่ยนแปลงได้อย่างรวดเร็วตามความต้องการ ก่อให้เกิดความน่าเชื่อถือในการส่งมอบซอฟต์แวร์ (Delivery) ทั้งนี้แต่ละขั้นตอนจะมีเครื่องมือเข้ามาสนับสนุนการทำงานที่เรียกว่า DevOps Tool Chain

จากข้อมูลข้างต้น เพื่อทดสอบความเข้าใจเบื้องต้นเกี่ยวกับแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

คำชี้แจง ให้ทำเครื่องหมาย/หน้าคำตอบในแต่ละข้อ เพียงคำตอบใดคำตอบหนึ่งเท่านั้น

1. แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะเข้ามาแทนที่การพัฒนาซอฟต์แวร์แบบ Agile ใช่หรือไม่

☐ ใช่	☐ ไม่ใช่

2. แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะช่วยให้ทีมหรือบุคคลใด ทำงานได้ราบรื่นขึ้น

☐ ผู้ใช้งาน (User) กับ นักวิเคราะห์ออกแบบ (System Analysis)
☐ ทีมพัฒนาระบบ (Development Team) กับ ทีมปฏิบัติการ (Operation Team)
☐ ทีมทดสอบระบบ (Tester) กับ ทีมประกันคุณภาพ (Quality Assurance)
☐ ทีมปฏิบัติการ (Operation Team) กับ ผู้ใช้งาน (User)

3. ข้อใด “ไม่ใช่” ประโยชน์ที่ได้จากการนำ DevOps เข้ามาใช้ในกระบวนการพัฒนาซอฟต์แวร์

☐ องค์กรสามารถผลิตซอฟต์แวร์ให้ตรงตามความต้องการผู้ใช้งานได้มากขึ้น
☐ ช่วยปรับปรุงคุณภาพในการทดสอบภายในสภาพแวดล้อมต่างๆ ก่อนการส่งมอบพัฒนาซอฟต์แวร์
☐ ช่วยให้เกิดการประสานงานที่ดีระหว่างทีมพัฒนาซอฟต์แวร์ (Development Team) กับ ทีมปฏิบัติการ (Operation Team)
☐ ทำให้ผู้ใช้งานมีการเปลี่ยนแปลงความต้องการ (Requirement) ได้น้อยลง

ส่วนที่ 3: เลือกระดับคะแนนที่ท่านเห็นด้วยมากที่สุด โดยทำเครื่องหมาย / ลงในช่องดังกล่าว ซึ่งลำดับคะแนนประกอบด้วย

ระดับคะแนน 5 หมายถึง เห็นด้วยอย่างยิ่ง

ระดับคะแนน 4 หมายถึง เห็นด้วย

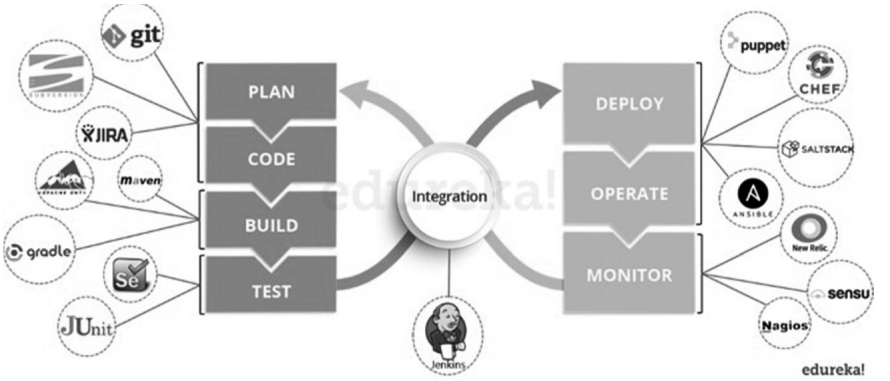
ระดับคะแนน 3 หมายถึง ปานกลาง

ระดับคะแนน 2 หมายถึง ไม่เห็นด้วย

ระดับคะแนน 1 หมายถึง ไม่เห็นด้วยอย่างยิ่ง

ข้อ	คำถาม	ระดับความสำคัญ				
		1	2	3	4	5
1	ผู้บริหารระดับสูงควรมีการกำหนดเป้าหมายที่แน่ชัด ในการเริ่มต้นนำแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มาใช้ในองค์กร					
2	เมื่อเริ่มต้นการนำแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มาใช้งาน ผู้บริหารระดับสูงควรสื่อสารไปทุกฝ่ายที่เกี่ยวข้อง (Stakeholder) เพื่อสร้างความเข้าใจที่ตรงกัน					
3	ผู้บริหารระดับสูงควรจัดสรรทรัพยากรด้านเวลา บุคลากร และเครื่องมือที่พร้อมต่อการนำแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps มาใช้งาน					
4	ผู้บริหารระดับสูงควรสนับสนุนให้มีการฝึกอบรมและให้ความรู้แก่พนักงานเพื่อเรียนรู้เกี่ยวกับแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps					
5	ผู้บริหารระดับสูงควรให้ความช่วยเหลือในการแก้ไขปัญหา เพื่อทำให้การนำแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ประสบความสำเร็จ					
6	ลักษณะโครงการพัฒนาซอฟต์แวร์ที่ท่านรับผิดชอบโดยส่วนใหญ่ต้องการให้รองรับการเปลี่ยนแปลงของผู้ใช้งานได้มากขึ้น					
7	ลักษณะโครงการพัฒนาซอฟต์แวร์ที่ท่านรับผิดชอบโดยส่วนใหญ่สร้างความพึงพอใจให้แก่ผู้ใช้งานได้มากขึ้น					
8	ลักษณะโครงการพัฒนาซอฟต์แวร์ที่ท่านรับผิดชอบโดยส่วนใหญ่เน้นการมีส่วนร่วมกับผู้ใช้งานมากขึ้น					
9	ลักษณะโครงการพัฒนาซอฟต์แวร์ที่ท่านรับผิดชอบโดยส่วนใหญ่ต้องการให้สามารถส่งมอบซอฟต์แวร์ (Release) ได้รวดเร็วขึ้น					
10	ลักษณะโครงการพัฒนาซอฟต์แวร์ที่ท่านรับผิดชอบโดยส่วนใหญ่มีอัตราการประสบความสำเร็จในโครงการที่มากขึ้น					

คำถามข้อ 11-14 เกี่ยวกับเครื่องมือ แนะนำให้ดูภาพที่ 4 ประกอบ เพื่อความเข้าใจมากยิ่งขึ้น



ข้อ	คำถาม	ระดับความสำคัญ				
		1	2	3	4	5
11	ท่านคิดว่าเครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps สามารถสนับสนุนการพัฒนาซอฟต์แวร์แต่ละขั้นตอนได้อย่างครอบคลุมทุกขั้นตอนตั้งแต่วางแผน (Plan) จนถึงติดตามผล (Monitor)					
12	ท่านคิดว่าความหลากหลายของเครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps (ดังภาพข้างต้น) ทำให้ผู้ใช้สามารถประเมินความเหมาะสมในการใช้เครื่องมือได้มากขึ้น					
13	ท่านคิดว่าการมีเครื่องมือแต่ละขั้นตอนในกระบวนการตามแนวความคิดแบบ DevOps มีส่วนช่วยสนับสนุนให้เกิดความต่อเนื่องในการดำเนินโครงการ					
14	ท่านคิดว่าการมีเครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ที่เพียงพอ จะช่วยให้ท่านสามารถทำงานได้รวดเร็วขึ้น					
15	ท่านคิดว่าการมีเครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ที่เพียงพอ จะช่วยให้ท่านสามารถทำงานได้สะดวกขึ้น					
16	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps สอดคล้องกับความต้องการทางธุรกิจในปัจจุบันที่เปลี่ยนแปลงอย่างรวดเร็ว					
17	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps สนับสนุนลักษณะโครงการที่ต้องการส่งมอบ (Release) อย่างรวดเร็ว					
18	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps สอดคล้องกับลักษณะโครงการที่มีการเปลี่ยนแปลงบ่อย					
19	ท่านคิดว่าการเลือกใช้เครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ให้สอดคล้องกับกระบวนการทำงานเป็นสิ่งที่สำคัญ					
20	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ทำงานร่วมกับกระบวนการพัฒนาซอฟต์แวร์แบบ Agile ได้ดี					

ข้อ	คำถาม	ระดับความสำคัญ				
		1	2	3	4	5
21	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะช่วยเพิ่มศักยภาพในการทำงานด้านการส่งมอบซอฟต์แวร์ได้รวดเร็วขึ้น					
22	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะช่วยให้สามารถตอบสนองความต้องการทางธุรกิจได้มากขึ้น					
23	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะช่วยส่งมอบ (Release) ซอฟต์แวร์ได้ตรงตามความต้องการของผู้ใช้งานได้มากขึ้น					
24	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะสร้างความต่อเนื่องในกระบวนการพัฒนาซอฟต์แวร์ให้มากขึ้น					
25	ท่านคิดว่าแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps จะช่วยให้การทำงานระหว่างทีมนักพัฒนาซอฟต์แวร์ (Development Team) และทีมปฏิบัติการ (Operation Team) ราบรื่นมากขึ้น					
26	ท่านคิดว่าท่านมีความสามารถในการเรียนรู้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ได้					
27	ท่านคิดว่าท่านมีความสามารถในการใช้เครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ในแต่ละขั้นตอนการพัฒนาซอฟต์แวร์					
28	ท่านคิดว่าท่านมีความมั่นใจในการเรียนรู้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps					
29	ท่านคิดว่าท่านมีความมั่นใจหากต้องศึกษาการใช้เครื่องมือที่สนับสนุนการดำเนินงานตามแนวคิด DevOps ตัวใหม่ในการทำงาน					
30	ท่านคาดการณ์ว่าในอนาคตองค์กรจะมีแนวโน้มนำแนวความคิดพัฒนาซอฟต์แวร์แบบ DevOps นี้มาใช้งาน					
31	ท่านอยากจะทดลองใช้เครื่องมือที่ช่วยในการพัฒนาซอฟต์แวร์เพื่อสร้างความเป็นอัตโนมัติ (Automate Tools) ในการทำงานให้มากขึ้น					
32	ท่านคิดว่าอยากจะลองใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps					
33	ท่านมีความตั้งใจที่จะใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps					

ส่วนที่ 4: ข้อ A, B, C, D เป็นคำถามเกี่ยวกับการใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Agile (***จำเป็นต้องตอบค่ะ)

A) ให้เลือกคำตอบที่ตรงกับการทำงานของท่านมากที่สุดเพียง 1 คำตอบ ในแต่ละข้อคำถาม

	คำถาม					
1	อัตราเฉลี่ยจำนวนโครงการ (Project) ทั้งหมดที่ท่านเคยรับผิดชอบ	< 10 โครงการ	10-20 โครงการ	21-30 โครงการ	31-40 โครงการ	>40 โครงการ
2	อัตราเฉลี่ยจำนวนโครงการ (Project) ทั้งหมดที่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ที่ท่านเคยรับผิดชอบ	1-20 %	21-40%	41-60%	61-80%	81-100%
3	ระยะเวลาที่องค์กรใช้งานระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์ตั้งแต่เริ่มต้นจนถึงปัจจุบัน	< 1 ปี	1-2ปี	2-3ปี	3-4 ปี	>5ปี
4	อัตราเฉลี่ยจำนวนโครงการในองค์กรทั้งหมดที่ใช้ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบแอจไจล์	1-20 %	21-40%	41-60%	60-80%	80-100%

B) เลือกวิธีปฏิบัติที่ใช้งาน (Practices) สามารถเลือกได้มากกว่า 1 ข้อ

วิธีปฏิบัติที่ใช้งาน (Practices)			
☐	Daily Stand-up Meetings	☐	Prioritized Backlogs
☐	Iteration/sprint Planning	☐	Short Iterations
☐	Retrospectives	☐	Continuous Integration
☐	อื่นๆ โปรดระบุ		

C) เลือก Agile Method สามารถเลือกได้มากกว่า 1 ข้อ

Agile Method			
☐	Scrum	☐	Adaptive Software Development
☐	Extreme Programming (XP)	☐	Feature-Driven Development (FDD)
☐	Lean Development	☐	อื่นๆ โปรดระบุ.....

D) เลือกเครื่องมือที่ใชในระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Agile สามารถเลือกได้มากกว่า 1 ข้อ

เครื่องมือที่ใช้ในระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Agile			
☐	Task Board	☐	Agile Project Management Tool
☐	Unit Test Tool	☐	Automated Build Tool
☐	Spreadsheet (ex. MS Excel)	☐	Project & Portfolio Management (PPM) Tool
☐	Microsoft Project	☐	อื่นๆ โปรดระบุ .....

ส่วนที่ 5: ข้อ E, F คำถามเกี่ยวกับการใช้กลุ่มของเครื่องมือสำหรับผู้ที่ใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps

(กรณีมีการใช้งานแนวคิดของ DevOps ถ้าไม่ใช้งานสามารถข้ามส่วนนี้ได้)

E) การใช้งานเครื่องมือในแนวคิด Dev Ops สามารถเลือกตอบได้มากกว่า 1 ข้อ ในแต่ละข้อคำถาม

Version Control	Build and Packaging	Continuous integration	Test Frameworks
☐ Subversion	☐ Maven	☐ Jenkins	☐ Kitchen CI
☐ Git/GitHub	☐ Ms Build	☐ Hudson	☐ Test Manager
☐ BitBucket	☐ Jenkins	☐ Bamboo	☐ Cucumber
☐ Mercurial	☐ Bamboo	☐ Travis	☐ Metasploit
☐ Microsoft Team Foundation	☐ Gradle	☐ Cruise Control	☐ JUnit
☐ อื่นๆ ระบุ.....	☐ อื่นๆ ระบุ.....	☐ อื่นๆ ระบุ.....	☐ อื่นๆ ระบุ.....
Artifact Repository	**Deploy Environment**	**Configuration management**	**Management & Monitoring**
☐ Apache Archiva	☐ Cloud formation	☐ Chef	☐ Stats
☐ Jfrog's Artifactory	☐ Packer	☐ Puppet	☐ jmxtrans
☐ Inedo's ProGet	☐ Dockers	☐ VmWare	☐ Metrics
☐ Sonatype Nexus	☐ VMware	☐ SaltStack	☐ Esper
☐ Maven Repository	☐ AWS	☐ Ansible	☐ Ganglia
☐ อื่นๆ ระบุ.....	☐ อื่นๆ ระบุ.....	☐ อื่นๆ ระบุ.....	☐ อื่นๆ ระบุ.....

F) ให้เลือกคำตอบที่ตรงกับการทำงานของท่านมากที่สุด**เพียง 1 คำตอบ** ในแต่ละข้อคำถาม

	คำถาม					
1	อัตราเฉลี่ยจำนวนโครงการ (Project) ทั้งหมดที่ใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ที่ท่านเคยรับผิดชอบ	1-20%	21-40%	41-60%	61-80%	81-100%
2	ระยะเวลาที่องค์กรใช้งานแนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps ตั้งแต่เริ่มต้นจนถึงปัจจุบัน	< 1 ปี	1-2 ปี	2-3 ปี	3-4 ปี	>5 ปี

ส่วนที่ 6: ข้อมูลทั่วไป ได้แก่ เพศ อายุ และข้อมูลเกี่ยวกับด้านความรับผิดชอบของงานของผู้ตอบแบบสอบถาม

คำชี้แจง: ให้ทำเครื่องหมาย/หน้าคำตอบในแต่ละข้อ เพียงคำตอบใดคำตอบหนึ่งเท่านั้น

1. เพศ	☐ ชาย	☐ หญิง
2. อายุ	☐ 21-25 ปี	☐ 26-30 ปี
	☐ 31-35 ปี	☐ 36-40 ปี
	☐ 41-45 ปี	☐ 46 ปีขึ้นไป
3. ประสบการณ์การทำงานในที่ทำงานปัจจุบัน	☐ < 1 ปี	☐ 1-3 ปี
	☐ 3-5 ปี	☐ 5-10 ปี
	☐ 10-15 ปี	☐ 15 ปีขึ้นไป
4. ตำแหน่งงาน	☐ หัวหน้าโครงการ (Project Manager)	☐ นักวิเคราะห์ระบบ (System Analysis)
	☐ หัวหน้าทีมพัฒนาซอฟต์แวร์ (Development Manager)	☐ นักวิเคราะห์ธุรกิจ (Business Analysis)
	☐ หัวหน้าทีมประกันคุณภาพ (Quality Assurance manager)	☐ หัวหน้าทีมปฏิบัติการ (Operation Manager)
	☐ Scrum Master	☐ Scrum Team
	☐ Product Owner	☐ อื่นๆ โปรดระบุ .....
5. ระดับการศึกษาสูงสุด	☐ ต่ำกว่า ปริญญาตรี	☐ ปริญญาตรี หรือ เทียบเท่า
	☐ ปริญญาโท	☐ ปริญญาเอก

6. ลักษณะองค์กรที่ท่านทำงานอยู่	☐ ธุรกิจพัฒนาซอฟต์แวร์	☐ ธุรกิจอาหาร
	☐ ธุรกิจการสื่อสารและโทรคมนาคม	☐ ธุรกิจเทคโนโลยี
	☐ ราชการ	☐ รัฐวิสาหกิจ
	☐ ธนาคารและสถาบันการเงิน	☐ ธุรกิจท่องเที่ยวและบริการ
	☐ อื่นๆ โปรดระบุ.....	
7. ขนาดองค์กร	☐ น้อยกว่า 50 คน	☐ 51-200 คน
	☐ มากกว่า 200 คน	

ขอบพระคุณที่ร่วมตอบแบบสอบถาม