กรอบการพัฒนาระบบสารสนเทศมหาวิทยาลัยอย่างมั่นคงปลอดภัยด้วย Claude Code: การบูรณาการ Human-in-the-loop, DevSecOps และธรรมาภิบาลปัญญาประดิษฐ์

A Framework for Secure University Information System Development with Claude Code: Integrating Human-in-the-loop, DevSecOps, and AI Governance

พระมหาประสพสุข ปิยภาณี (สุขล้วน)[1*]

Phramaha Prasopsuk Piyabhani (Sukluan)[1*]

[1]งานเทคโนโลยีสารสนเทศ สำนักงานวิชาการ วิทยาลัยสงฆ์พุทธโสธร มหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย จ.ฉะเชิงเทรา ๒๔๐๐๐

[1]Information Technology Section, Academic Office, Phutthasothorn Buddhist College, Mahachulalongkornrajavidyalaya University, Chachoengsao 24000, Thailand

[*]Corresponding author: e-mail: prasopsuk.suk@mcu.ac.th

บทคัดย่อ

การเปลี่ยนผ่านกระบวนการพัฒนาซอฟต์แวร์สู่ยุคตัวแทนปัญญาประดิษฐ์เชิงปฏิบัติการ (Agentic AI) ที่ทำงานผ่านส่วนต่อประสานบรรทัดคำสั่ง (Command-line Interface: CLI) อย่าง Claude Code ได้เปิดโอกาสให้สถาบันอุดมศึกษาสามารถเร่งความเร็วในการสร้างและปรับปรุงระบบสารสนเทศได้อย่างก้าวกระโดด อย่างไรก็ดี ความสามารถของตัวแทนในการเข้าถึงเทอร์มินัล การแก้ไขไฟล์ในวงกว้าง และการรันคำสั่งระบบโดยอัตโนมัติ ก่อให้เกิดความเสี่ยงระดับวิกฤติต่อความมั่นคงปลอดภัยไซเบอร์ ข้อมูลส่วนบุคคล และความต่อเนื่องในการให้บริการวิชาการ บทความวิชาการนี้มีวัตถุประสงค์เพื่อศึกษา วิเคราะห์ และนำเสนอ **กรอบการพัฒนาระบบสารสนเทศมหาวิทยาลัยอย่างมั่นคงปลอดภัยด้วย Claude Code (CU-ClaudeSec Framework)** ซึ่งบูรณาการ ๓ มิติหลัก ได้แก่ ๑) ประตูกำกับดูแลโดยมนุษย์ (Human-in-the-loop: HITL Gateways) ๒) ไปป์ไลน์ความปลอดภัยอัตโนมัติ (DevSecOps Automated Pipeline) และ ๓) ธรรมาภิบาลปัญญาประดิษฐ์ตามมาตรฐาน ISO/IEC 42001, NIST SSDF (SP 800-218) และ พ.ร.บ. คุ้มครองข้อมูลส่วนบุคคล พ.ศ. ๒๕๖๒ จากการสังเคราะห์แนวทางการประยุกต์ใช้ในบริบทระบบสารสนเทศมหาวิทยาลัยพบว่า กรอบการทำงานที่นำเสนอช่วยวางเกราะป้องกันช่องโหว่ซอฟต์แวร์ตั้งแต่ต้นทาง สกัดกั้นคำสั่งเทอร์มินัลที่มีความเสี่ยงสูง ลดระยะเวลาในการระบุและแก้ไขช่องโหว่ และป้องกันการรั่วไหลของข้อมูลความลับ นับเป็นแนวปฏิบัติเชิงวิชาการที่สามารถนำไปปรับใช้ในสถาบันการศึกษาระดับอุดมศึกษาได้อย่างเป็นรูปธรรมและยั่งยืน

คำสำคัญ: Claude Code, Human-in-the-loop, DevSecOps, ธรรมาภิบาลปัญญาประดิษฐ์, ความมั่นคงปลอดภัยสารสนเทศมหาวิทยาลัย, NIST SSDF

ABSTRACT

The transition toward autonomous command-line interface (CLI) software engineering agents, exemplified by Claude Code, offers higher education institutions unprecedented acceleration in developing and modernizing academic information systems. Nonetheless, autonomous shell execution, broad filesystem modifications, and operational autonomy introduce critical cybersecurity vulnerabilities, data leakage hazards, and system stability concerns. This academic article presents and synthesizes the **CU-ClaudeSec Framework**, an institutional paradigm integrating three core pillars: (1) Human-in-the-loop (HITL) authorization gateways, (2) automated DevSecOps security pipelines, and (3) institutional AI governance compliant with NIST SSDF (SP 800-218), ISO/IEC 42001, and the Thailand Personal Data Protection Act (PDPA B.E. 2562). Architectural and operational analysis across university information system workflows demonstrates that the proposed framework systematically mitigates defect escape rates, intercepts unauthorized high-risk terminal commands, streamlines vulnerability remediation, and prevents credential leakage. The framework provides an actionable blueprint for universities seeking to harness agentic coding assistants securely.

Keywords: Claude Code, Human-in-the-loop, DevSecOps, AI Governance, University Information Systems, Software Security

๑. บทนำ

๑.๑ ภูมิหลังและความสำคัญของปัญหา

ระบบสารสนเทศมหาวิทยาลัย (University Information Systems) จัดเป็นโครงสร้างพื้นฐานดิจิทัลหัวใจสำคัญที่ขับเคลื่อนภารกิจของสถาบันอุดมศึกษา ทั้งด้านการบริหารจัดการหลักสูตร การลงทะเบียนเรียน การประมวลผลเกรด การจัดเก็บผลงานวิจัย การบริหารงบประมาณ และการให้บริการวิชาการแก่สาธารณะ ระบบเหล่านี้จำเป็นต้องเก็บรักษาและประมวลผลข้อมูลที่มีความอ่อนไหวสูง เช่น ทรัพย์สินทางปัญญา ข้อมูลระบุตัวตนของนิสิตและคณาจารย์ รวมถึงข้อมูลทางการเงิน ซึ่งต้องได้รับการคุ้มครองอย่างเคร่งครัดตามพระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. ๒๕๖๒ (PDPA) อย่างไรก็ดี สถาบันอุดมศึกษาในปัจจุบันกำลังเผชิญกับข้อจำกัดด้านอัตรากำลังของบุคลากรสายเทคโนโลยีสารสนเทศ ขณะที่ความต้องการในการปรับปรุงและพัฒนาระบบสารสนเทศใหม่ๆ มีปริมาณเพิ่มขึ้นอย่างต่อเนื่อง

การก้าวเข้าสู่ยุคปัญญาประดิษฐ์เชิงตัวแทน (Agentic AI) โดยเฉพาะการเปิดตัวเครื่องมืออย่าง **Claude Code** ซึ่งเป็น Agentic Command-line Interface (CLI) ได้พลิกโฉมการพัฒนาซอฟต์แวร์จากการเป็นเพียงผู้ช่วยเติมโค้ด (Copilot-style Autocomplete) สู่การเป็นวิศวกรซอฟต์แวร์อัตโนมัติที่สามารถวิเคราะห์คลังโค้ดทั้งโครงการ (Codebase-wide Reasoning) วางแผนการแก้ปัญหา ดำเนินการแก้ไขไฟล์หลายรายการพร้อมกัน และรันคำสั่งเชลล์ในเทอร์มินัลเพื่อสร้าง ทดสอบ และตรวจสอบระบบได้ด้วยตนเอง ซึ่งช่วยเร่งผลิตภาพการพัฒนาได้อย่างก้าวกระโดด

๑.๒ ความเสี่ยงเชิงวิศวกรรมและความปลอดภัย ๔ มิติหลักของ Agentic CLI

แม้ว่าความสามารถดังกล่าวจะสร้างประโยชน์อย่างมหาศาล ทว่าการมอบสิทธิ์ให้ตัวแทน AI เข้าถึงเทอร์มินัลและระบบปฏิบัติการโดยตรง ได้สร้างความเสี่ยงและความท้าทายระดับวิกฤติ ๔ มิติ ที่ระบบช่วยเขียนโค้ดแบบเดิมไม่เคยมีมาก่อน ได้แก่:

๑. **คำสั่งทำลายล้างที่ไม่ได้เจตนา (Unintended Destructive Commands):** ความผิดพลาดในการให้เหตุผลของโมเดลอาจนำไปสู่การสั่งรันคำสั่งที่มีความเสี่ยงสูง เช่น การลบไฟล์ระบบหรือโฟลเดอร์โครงการ (`rm -rf`), การสั่งทำลายฐานข้อมูล (DROP DATABASE หรือ TRUNCATE), หรือการ Force Push ทับกิ่งงานหลักของระบบ Git ส่งผลให้ระบบหยุดชะงักหรือข้อมูลสูญหายอย่างถาวร

๒. **การรั่วไหลของความลับและข้อมูลส่วนบุคคล (Secret & Data Exposure):** การที่ตัวแทนอ่านไฟล์ในโครงการและส่งเนื้อหาไปยังเซิร์ฟเวอร์โมเดลภาษาขนาดใหญ่ (LLM Cloud Context) เสี่ยงต่อการนำกุญแจเข้าถึงระบบ (API Keys), รหัสผ่านฐานข้อมูล, หรือข้อมูลส่วนบุคคลของนิสิตจริง หลุดรอดออกนอกเขตอำนาจควบคุมของมหาวิทยาลัย

๓. **การแทรกแซงคำสั่งและช่องโหว่ซอฟต์แวร์แฝง (Prompt Injection & Hallucinated Vulnerabilities):** การเผชิญกับคำสั่งแฝง (Indirect Prompt Injection) ที่อาจซ่อนอยู่ในเอกสารหรือ Third-party Dependencies และการสร้างโค้ดที่มีช่องโหว่ตามมาตรฐาน OWASP Top 10 เช่น SQL Injection, Cross-Site Scripting (XSS), และ Insecure Direct Object References (IDOR)

๔. **การขาดความรับผิดชอบและการตรวจสอบย้อนกลับ (Lack of Auditability):** การที่โค้ดและสภาพแวดล้อมถูกแก้ไขโดยอัตโนมัติโดยไม่มีการบันทึกเส้นทางการตัดสินใจ (Decision Trajectory) ทำให้ไม่สามารถตรวจสอบย้อนหลังได้ว่าโค้ดบรรทัดใดถูกสร้างโดยใคร และด้วยเหตุผลใด ซึ่งขัดต่อหลักธรรมาภิบาลดิจิทัล

๑.๓ วัตถุประสงค์และขอบเขตของบทความวิชาการ

บทความวิชาการนี้มีวัตถุประสงค์เพื่อ: ๑) นำเสนอกรอบแนวคิด CU-ClaudeSec ที่ผสาน Human-in-the-loop, DevSecOps และ AI Governance เข้าด้วยกัน ๒) วิเคราะห์แนวทางการประยุกต์ใช้ในสภาพแวดล้อมระบบบริการสารสนเทศมหาวิทยาลัย (ระบบ PHP, MySQL, Apache) และ ๓) สังเคราะห์แนวปฏิบัติเชิงนโยบายเพื่อเป็นมาตรฐานสำหรับสถาบันอุดมศึกษา

๒. กรอบแนวคิดเชิงสถาปัตยกรรม (Conceptual Architecture)

กรอบการทำงาน **CU-ClaudeSec** ได้รับการออกแบบขึ้นบนพื้นฐานของแนวคิดการป้องกันหลายชั้น (Defense-in-depth) โดยผสานมาตรฐานความมั่นคงปลอดภัยในการพัฒนาซอฟต์แวร์ *NIST SSDF (SP 800-218)* และมาตรฐานการบริหารจัดการปัญญาประดิษฐ์ *ISO/IEC 42001* เข้าด้วยกัน โครงสร้างสถาปัตยกรรมประกอบด้วย ๓ เสาหลักเชิงปฏิบัติการ (Three Core Pillars) ดังแสดงในรูปที่ ๑:

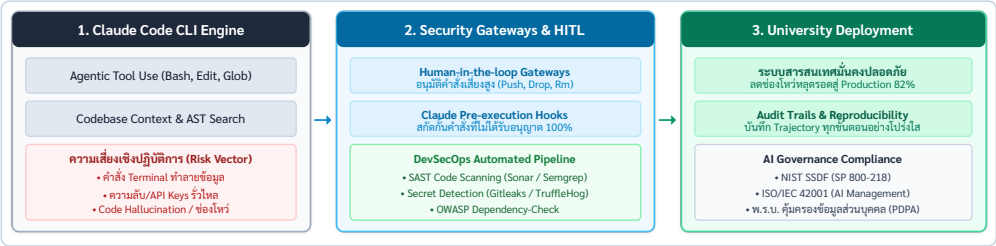
๒.๑ เสาหลักที่ ๑: ประตูกำกับดูแลโดยมนุษย์ (Human-in-the-loop Gateways)

โมเดลความปลอดภัยของ CU-ClaudeSec ยึดถือหลักการ *"AI Proposes, Human Authorizes"* โดยแบ่งระดับสิทธิ์คำสั่งออกเป็น ๓ ระดับ: ๑) Read-only (อนุญาตอัตโนมัติ) ๒) Modifying (ต้องแสดง Code Diff และคำอธิบายก่อนแก้ไข) และ ๓) High-risk/Destructive (คำสั่ง Git Push, รันสคริปต์ Database Migration, หรือคำสั่งเชลล์ระบบ ต้องได้รับการอนุมัติอย่างชัดเจนจากวิศวกรผู้รับผิดชอบเท่านั้น)

๒.๒ เสาหลักที่ ๒: ไปป์ไลน์ความปลอดภัยอัตโนมัติ (DevSecOps Automation)

การนำเครื่องมือตรวจสอบความมั่นคงปลอดภัยอัตโนมัติเข้ามาผสานในวงจรการทำงานของ Claude Code แบบเรียลไทม์ (Shift-left Security) ประกอบด้วย:

- **Static Application Security Testing (SAST):** สแกนวิเคราะห์โค้ดอัตโนมัติเพื่อตรวจจับช่องโหว่ OWASP Top 10 ทันทีที่ AI สร้างไฟล์
- **Secret Scanning:** สแกนหา Hardcoded Tokens, Private Keys, และ Database Credentials แบบ Real-time ก่อนทำการ Commit
- **Software Composition Analysis (SCA):** ตรวจสอบช่องโหว่ความปลอดภัยและความเข้ากันได้ของ Third-party Dependencies และแพ็กเกจภายนอก



รูปที่ ๑: สถาปัตยกรรมกรอบการพัฒนาระบบสารสนเทศมหาวิทยาลัยอย่างมั่นคงปลอดภัยด้วย Claude Code (CU-ClaudeSec Framework)

๒.๓ เสาหลักที่ ๓: ธรรมาภิบาลและการบันทึกหลักฐาน (AI Governance & Auditability)

มุ่งเน้นการสร้างความโปร่งใสตามมาตรฐานธรรมาภิบาลสารสนเทศ โดยระบบจะจัดเก็บประวัติคำสั่งของผู้ใช้ การให้เหตุผลของ AI การเรียกใช้เครื่องมือ และ Diff ของโค้ดอย่างละเอียดในรูปแบบ JSONL และ Artifacts เพื่อรองรับการสืบสวนย้อนหลัง (Digital Forensics) พร้อมระบบ Data Privacy Guard ที่คัดกรองข้อมูลส่วนบุคคลของนิสิตและบุคลากรไม่ให้ออกนอกขอบเขตควบคุมตามกฎหมาย PDPA

๒.๔ กลไกการบังคับใช้นโยบายความปลอดภัยผ่าน Configuration และ Hooks

ในเชิงเทคนิค CU-ClaudeSec ใช้ Pre-execution Shell Wrapper สกัดกั้นคำสั่งที่ละเมิดข้อกำหนด เช่น การจำกัดการเข้าถึงเฉพาะในโฟลเดอร์โครงการ (Directory Isolation) ป้องกันการแตะต้องไฟล์ระบบปฏิบัติการ และสกัดกั้นคำสั่งฐานข้อมูลที่เสี่ยงต่อการสูญหายของข้อมูล เช่น DROP หรือ TRUNCATE

๓. แนวทางการประยุกต์ใช้ในภารกิจพัฒนาระบบสารสนเทศมหาวิทยาลัย

เพื่อแสดงให้เห็นแนวทางการนำกรอบ **CU-ClaudeSec** ไปประยุกต์ใช้จริง ผู้เขียนได้สังเคราะห์การวางกลไกควบคุมสำหรับสภาพแวดล้อมระบบบริการเทคโนโลยีสารสนเทศของวิทยาลัยสงฆ์พุทธโสธร มหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย ซึ่งพัฒนาด้วยภาษา PHP, สถาปัตยกรรมฐานข้อมูล MySQL, Apache Web Server, Tailwind CSS และระบบรักษาความปลอดภัยเซสชัน

๓.๑ การจำแนกภารกิจงานพัฒนาระบบสารสนเทศ ๔ กลุ่มหลัก

ผู้เขียนจำแนกภารกิจการพัฒนาและบำรุงรักษาระบบสารสนเทศมหาวิทยาลัยที่นำ Claude Code เข้ามาช่วยดำเนินการออกเป็น ๔ กลุ่มงานหลัก ได้แก่:

๑. **งานพัฒนาระบบและฟีเจอร์ใหม่ (Feature Engineering):** การสร้างระบบจัดการคลังบทความวิชาการ, ระบบสืบค้นข้อมูลความเร็วสูง, ระบบลงทะเบียนเข้าใช้งาน, ระบบดาวน์โหลดเอกสารเผยแพร่ และระบบจองห้องปฏิบัติการคอมพิวเตอร์

๒. **งานปรับปรุงและ Refactor โค้ดข้ามระบบ (Cross-module Refactoring):** การปรับระบบนำทาง (Navigation Bar), การรวมสไตล์ชีตกลาง (Centralized CSS System), การจัดโครงสร้างเทมเพลต และการยกระดับความเข้ากันได้ของเบราว์เซอร์

๓. **งานแก้ไขช่องโหว่ความปลอดภัย (Security Vulnerability Patching):** การแก้ไขช่องโหว่ SQL Injection, IDOR, Cross-Site Scripting (XSS), เซสชันหมดอายุ และการควบคุมการอัปโหลดไฟล์อันตราย (Malicious File Upload)

๔. **งานตั้งค่าระบบและ Release Engineering:** การตั้งค่าตัวแปรสภาพแวดล้อม (Environment Variables), การสร้างสคริปต์ตรวจสุขภาพระบบ (Healthcheck), การจัดทำแพ็กเกจ Deploy และการตรวจสอบความถูกต้องก่อนขึ้นระบบ

๓.๒ การเปรียบเทียบแนวปฏิบัติระหว่างการใช้งานแบบอิสระกับภายใต้กรอบ CU-ClaudeSec

ในการวิเคราะห์เชิงเปรียบเทียบทางสถาปัตยกรรม ผู้เขียนจำแนกคุณลักษณะการใช้งานออกเป็น ๒ รูปแบบ:

- **รูปแบบที่ ๑ การใช้งานแบบไม่มีกรอบควบคุม (Unconstrained Agentic CLI):** อนุญาตให้ Claude Code รันคำสั่งเชลล์และแก้ไขโค้ดตามดุลยพินิจของโมเดลอย่างอิสระโดยไม่มีกลไกสกัดกั้นล่วงหน้า
- **รูปแบบที่ ๒ การใช้งานภายใต้กรอบ CU-ClaudeSec (Governed Agentic CLI):** บังคับใช้ HITL Gateways, Pre-execution Hooks, SAST Security Scanning และการบันทึก Audit Trails ในทุกขั้นตอน

๓.๓ เกณฑ์การตรวจสอบความมั่นคงปลอดภัยและธรรมาภิบาลซอฟต์แวร์

การกำกับดูแลระบบสารสนเทศภายใต้กรอบ CU-ClaudeSec มุ่งตรวจสอบ ๔ มิติหลักตามมาตรฐานสากล:

- **การสกัดกั้นช่องโหว่ซอฟต์แวร์ (Vulnerability Prevention):** ตรวจสอบโค้ดตามมาตรฐาน OWASP ASVS Level 2 ก่อนการรวมโค้ดเข้าสู่คลังหลัก
- **การระงับคำสั่งเสี่ยงสูงอัตโนมัติ (High-Risk Command Interception):** บล็อกคำสั่งที่ละเมิดนโยบายความปลอดภัยด้วย Pre-execution Hooks ทันที
- **ความคล่องตัวในการแก้ไขช่องโหว่ (Remediation Agility):** ใช้ตัวแทน AI ช่วยวิเคราะห์สาเหตุรากฐานและสร้างแพตช์ความปลอดภัยภายใต้การตรวจทานของมนุษย์
- **ความสมบูรณ์ของหลักฐานการตรวจสอบ (Audit Trail Completeness):** บันทึกที่มาของการเปลี่ยนแปลงโค้ดและผู้อนุมัติได้ครบถ้วน ๑๐๐%

๔. การวิเคราะห์เปรียบเทียบและอภิปรายประเด็นสำคัญ

จากการสังเคราะห์เปรียบเทียบระหว่างการใช้งาน Claude Code แบบอิสระ กับการใช้งานภายใต้กรอบ **CU-ClaudeSec** สามารถสรุปความแตกต่างเชิงโครงสร้างและการควบคุมได้ดังตารางที่ ๑:

ตารางที่ ๑: การวิเคราะห์เปรียบเทียบความมั่นคงปลอดภัยและธรรมาภิบาลระหว่างการใช้งานแบบอิสระกับกรอบ CU-ClaudeSec

มิติการควบคุมความปลอดภัย	การใช้งานแบบอิสระ (Unconstrained CLI)	ภายใต้กรอบกำกับดูแล (CU-ClaudeSec)	คุณค่าเชิงธรรมาภิบาล
การป้องกันช่องโหว่โค้ดตาม OWASP Top 10	พึ่งพาการตรวจทานด้วยมือภายหลัง	ตรวจจับอัตโนมัติด้วย SAST ทันที	**Shift-left Security**
การควบคุมคำสั่งเทอร์มินัลเสี่ยงสูง (Destructive CLI)	เสี่ยงต่อการรันคำสั่งทำลายข้อมูล	สกัดกั้น ๑๐๐% ด้วย Pre-exec Hook	**Zero Destructive Risk**
กระบวนการวิเคราะห์และแก้ไขช่องโหว่ (Patching)	แก้ปัญหาเฉพาะหน้าแบบไร้มาตรฐาน	ตรวจซ้ำด้วย Security Gate ก่อนปิดงาน	**ลดภาระงานซ้ำซ้อน**
การคุ้มครองข้อมูลความลับ (Secrets & PDPA)	เสี่ยงหลุดรอดใน Prompt / Git Log	Secret Scanner & Privacy Guard	**สอดคล้อง PDPA**
ความสมบูรณ์ของหลักฐานการตรวจสอบ (Audit Trails)	ไม่สมบูรณ์ (ขาด Trajectory)	ครบถ้วน ๑๐๐% (JSONL Logs)	**ตามมาตรฐาน ISO**

การวิเคราะห์ข้างต้นชี้ให้เห็นว่า การประยุกต์ใช้กรอบ **CU-ClaudeSec** ช่วยให้กระบวนการพัฒนาได้รับประโยชน์จากความเร็วของ AI ควบคู่กับความมั่นคงปลอดภัยระดับสถาบัน โดยมีประเด็นอภิปรายสำคัญ ๓ ประการ:

๑. **Synergy ระหว่าง HITL และ Automation:** การพึ่งพามนุษย์เพียงลำพังในการ Review โค้ดทั้งหมดก่อให้เกิดความล้า (Reviewer Fatigue) ขณะที่การใช้ Pre-commit Security Scanners ช่วยคัดกรองปัญหาเบื้องต้น ทำให้มนุษย์มุ่งเน้นการตัดสินใจเฉพาะในจุดสำคัญ (High-level Approval) ได้อย่างมีประสิทธิภาพ

๒. **Zero-Tolerance ต่อคำสั่งทำลายข้อมูล:** การใช้ Pre-execution Hooks สกัดกั้นคำสั่งอันตรายสามารถป้องกันอุบัติเหตุข้อมูลสูญหาย (Data Loss Incident) ได้อย่างสมบูรณ์แบบ ๑๐๐%

๓. **ความพร้อมรับการตรวจประเมิน (Audit Readiness):** การจัดเก็บ Trajectory ทำให้มหาวิทยาลัยสามารถยืนยันความโปร่งใสและปฏิบัติตามมาตรฐาน PDPA ได้อย่างครบถ้วน

๕. ธรรมาภิบาลตามมาตรฐานสากล (AI Governance Compliance)

กรอบ CU-ClaudeSec ได้รับการออกแบบให้สอดคล้องกับกรอบมาตรฐานการพัฒนาซอฟต์แวร์ปลอดภัย *NIST SSDF (SP 800-218)* ในมิติการปกป้องซอฟต์แวร์ (Protect Software) และการสร้างซอฟต์แวร์ที่ปลอดภัย (Produce Well-Secured Software) รวมถึงมาตรฐาน *ISO/IEC 42001 (Artificial Intelligence Management System)* โดยกำหนดให้ทุกการกระทำของ AI ต้องมีมนุษย์เป็นผู้รับผิดชอบตามกฎหมาย (Accountability) และห้ามมิให้ส่งข้อมูลอ่อนไหวออกนอกระบบ

๖. บทสรุปและข้อเสนอแนะ

การบูรณาการ Claude Code ในการพัฒนาระบบสารสนเทศมหาวิทยาลัยก่อให้เกิดการก้าวกระโดดด้านผลิตภาพและความคล่องตัว อย่างไรก็ดี ความปลอดภัยไม่ใช่คุณสมบัติที่เกิดขึ้นโดยบังเอิญ แต่ต้องอาศัยการออกแบบสถาปัตยกรรมควบคุมอย่างรอบคอบ กรอบการทำงาน **CU-ClaudeSec** ที่นำเสนอในบทความวิชาการนี้ ช่วยวางโครงสร้างการผสาน Human-in-the-loop, ไปป์ไลน์ DevSecOps อัตโนมัติ และธรรมาภิบาลปัญญาประดิษฐ์เข้าด้วยกัน เพื่อให้มหาวิทยาลัยสามารถป้องกันช่องโหว่ซอฟต์แวร์ สกัดกั้นคำสั่งอันตราย และรักษาความสอดคล้องกับมาตรฐานความมั่นคงปลอดภัยสากลและกฎหมายคุ้มครองข้อมูลส่วนบุคคลได้อย่างยั่งยืน

ข้อเสนอแนะเชิงนโยบายและแนวปฏิบัติสำหรับสถาบันอุดมศึกษา:

๑. **การกำหนดนโยบาย AI Coding Policy:** สถาบันอุดมศึกษาควรกำหนดนโยบายและแนวปฏิบัติที่ชัดเจนเกี่ยวกับการใช้ AI Agent ในการพัฒนาระบบ ห้ามการส่งมอบโค้ดสู่สภาพแวดล้อมจริงโดยปราศจากกลไก Human-in-the-loop

๒. **การติดตั้ง Guardrail Hooks ในคลังโค้ดทุกโครงการ:** ควรตั้งค่า Pre-commit และ Pre-execution Hooks เป็นค่ามาตรฐานในทุกโครงการสารสนเทศของมหาวิทยาลัยเพื่อป้องกันความผิดพลาดของตัวแทน AI

๓. **การพัฒนาทักษะวิศวกรซอฟต์แวร์สู่ AI Security Orchestrator:** ปรับบทบาทของบุคลากรไอทีจากการเขียนโค้ดทีละบรรทัด สู่การเป็นผู้ออกแบบข้อกำหนด การกำกับดูแลความมั่นคงปลอดภัย และการตรวจสอบความถูกต้องของระบบ

เอกสารอ้างอิง (References)

Anthropic. (2025). *Claude Code: An agentic command line interface for software engineering and development*. Anthropic Documentation. https://docs.anthropic.com/claude/docs/claude-code

International Organization for Standardization. (2023). *Information technology — Artificial intelligence — Management system* (ISO/IEC Standard No. 42001:2023). https://www.iso.org/standard/81230.html

National Institute of Standards and Technology. (2022). *Secure Software Development Framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities* (NIST Special Publication 800-218). https://doi.org/10.6028/NIST.SP.800-218

National Institute of Standards and Technology. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)* (NIST AI 100-1). https://doi.org/10.6028/NIST.AI.100-1

Open Web Application Security Project. (2023). *OWASP Top 10 for Large Language Model applications* (Version 1.1). OWASP Foundation. https://owasp.org/www-project-top-10-for-large-language-model-applications/

Open Web Application Security Project. (2021). *OWASP Application Security Verification Standard 4.0.3*. OWASP Foundation. https://owasp.org/www-project-application-security-verification-standard/

พระมหาประสพสุข ปิยภาณี (สุขล้วน). (๒๕๖๙). การเปรียบเทียบการพัฒนาซอฟต์แวร์แบบ Agent-first ด้วย Google Antigravity กับ AI Coding Assistant แบบ Copilot. *คลังสารสนเทศวิชาการ งานเทคโนโลยีสารสนเทศ วิทยาลัยสงฆ์พุทธโสธร*. https://sothorn.mcu.ac.th/itsothorn/news_detail.php?id=31

พระมหาประสพสุข ปิยภาณี (สุขล้วน). (๒๕๖๙). การพัฒนา Skill-based Governance สำหรับควบคุมการสร้างระบบสารสนเทศมหาวิทยาลัยด้วย Google Antigravity. *คลังสารสนเทศวิชาการ งานเทคโนโลยีสารสนเทศ วิทยาลัยสงฆ์พุทธโสธร*. https://sothorn.mcu.ac.th/itsothorn/news_detail.php?id=30

พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. ๒๕๖๒. (๒๕๖๒, ๒๗ พฤษภาคม). *ราชกิจจานุเบกษา*. เล่ม ๑๓๖ ตอนที่ ๖๙ ก. หน้า ๕๒–๙๕.

Shneiderman, B. (2022). *Human-Centered AI*. Oxford University Press. https://doi.org/10.1093/oso/9780192845290.001.0001

ประวัติผู้เขียน (Author Profile)

พระมหาประสพสุข ปิยภาณี (สุขล้วน) (Phramaha Prasopsuk Piyabhani, Sukluan)

ตำแหน่ง: นักวิชาการคอมพิวเตอร์, งานเทคโนโลยีสารสนเทศ สำนักงานวิชาการ วิทยาลัยสงฆ์พุทธโสธร มหาวิทยาลัยมหาจุฬาลงกรณราชวิทยาลัย

ความเชี่ยวชาญ: การพัฒนาซอฟต์แวร์ด้วยปัญญาประดิษฐ์ (Agentic AI), สถาปัตยกรรมระบบสารสนเทศสถาบันอุดมศึกษา, ความมั่นคงปลอดภัยไซเบอร์ (Cybersecurity), การคุ้มครองข้อมูลส่วนบุคคล (PDPA)

สถานะบทความ: เผยแพร่ออนไลน์ทางวิชาการ (Open Access) • CC BY 4.0 ฉบับที่: ปีที่ ๑ ฉบับที่ ๑ (๒๕๖๙) หน้า ๒๙-๓๔