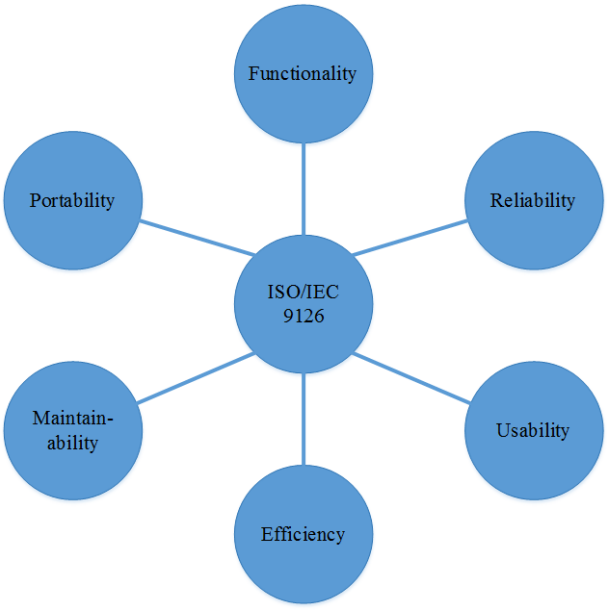
การประกันคุณภาพซอฟต์แวร์

ฝ่ายพัฒนาระบบงาน กองพัฒนาระบบงานคอมพิวเตอร์ เพชรรัตน์ ปัญญาภาณุวัฒน์

"คุณภาพ" เป็นสิ่งสำคัญที่สุดในอุตสาหกรรมการผลิตซอฟต์แวร์ โดยทั่วไปคำว่า "คุณภาพ" หมายถึง การที่ผลิตภัณฑ์นั้นสามารถตอบสนองความต้องการของผู้ใช้ แต่สำหรับผลิตภัณฑ์ซอฟต์แวร์แล้ว "คุณภาพ" ยังรวมไปถึงคุณลักษณะในด้านอื่นๆ ของซอฟต์แวร์อีกด้วย ได้แก่ ความสามารถในการบำรุงรักษาได้ง่าย ความน่าเชื่อถือ หรือประสิทธิภาพในการทำงาน เป็นต้น คุณลักษณะดังกล่าวไม่สามารถเขียนเป็นข้อกำหนดถึงวิธีปฏิบัติได้อย่างชัดเจนในเบื้องต้น ดังนั้นจึงเป็นหน้าที่ของวิศวกรซอฟต์แวร์ที่จะต้องกำหนดวิธีการสร้างคุณภาพให้กับซอฟต์แวร์ในแต่ละขั้นตอนของการผลิต โดยอาศัยกระบวนการจัดการคุณภาพซอฟต์แวร์ (Software Quality Management) เพื่อให้บรรลุวัตถุประสงค์ดังกล่าว

โดยปราศจากกระบวนการจัดการคุณภาพที่ดี อาจได้ผลิตภัณฑ์ที่ไม่ตรงตามความต้องการ

การประกันคุณภาพซอฟต์แวร์หรือการรับรองคุณภาพซอฟต์แวร์ (Software Quality Assurance) เป็นกิจกรรมหนึ่งของกระบวนการจัดการคุณภาพซอฟต์แวร์ ซึ่งควรมีการวางแผนอย่างเป็นระบบตั้งแต่ในระยะเริ่มแรกของกิจกรรมการออกแบบ นอกเหนือไปจากการให้ความสำคัญกับตัวผลิตภัณฑ์เพื่อรับประกันว่าระบบมีลักษณะเป็นไปตามมาตรฐานคุณภาพที่กำหนดไว้



มาตรฐานคุณภาพ ISO/IEC 9126

ซอฟต์แวร์มีลักษณะทางคุณภาพทั้งภายในและภายนอก โดยลักษณะภายนอกเป็นลักษณะที่ผู้ใช้ซอฟต์แวร์สนใจ ได้แก่

- **ความถูกต้อง (Correctness)** หมายถึง ระดับของการปราศจากข้อบกพร่องในข้อกำหนดการออกแบบ และการสร้างเป็นรูปธรรม
- **การใช้งาน (Usability)** หมายถึง ความง่ายในการเรียนรู้และใช้งานระบบ
- **ประสิทธิภาพ (Efficiency)** หมายถึง การใช้ทรัพยากรระบบน้อยที่สุด รวมทั้งหน่วยความจำและเวลาในการประมวลผล
- **ความน่าเชื่อถือ (Reliability)** หมายถึง ความสามารถของระบบในการทำหน้าที่ภายใต้เงื่อนไขที่ต้องการ การมีเวลาเฉลี่ยของความบกพร่องที่ยาวนาน
- **ความสมบูรณ์ (Integrity)** หมายถึง ระดับการป้องกันระบบจากผู้ไม่ได้รับอนุญาต หรือจากการเข้าถึงโปรแกรมและข้อมูลอย่างไม่เหมาะสม ความสมบูรณ์นี้รวมถึงการรับประกันว่าข้อมูลถูกเข้าถึงอย่างเหมาะสม กล่าวคือ ตารางที่มีข้อมูลเกี่ยวข้องกันถูกแก้ไขไปด้วยกัน ฟิลด์วันที่เก็บข้อมูลเฉพาะวันที่ที่มีรูปแบบถูกต้อง เป็นต้น
- **การปรับตัวได้ (Adaptability)** หมายถึง ขอบเขตที่สามารถนำระบบไปใช้โดยไม่ต้องดัดแปลงแก้ไขแอปพลิเคชัน หรือสภาวะแวดล้อมอื่นนอกจากที่ได้รับการออกแบบมาโดยเฉพาะ
- **ความแม่นยำ (Accuracy)** หมายถึง ระดับของการปราศจากข้อผิดพลาดของระบบเมื่อถูกสร้างขึ้น โดยเฉพาะอย่างยิ่งเอาต์พุตที่ได้มาจากการคำนวณเชิงตัวเลข ความแม่นยำต่างจากความถูกต้อง กล่าวคือเป็นการกำหนดว่าระบบทำงานได้ดีเพียงใดมากกว่าการพิจารณาว่าระบบถูกสร้างมาอย่างถูกต้องหรือไม่
- **เสถียรภาพ (Robustness)** หมายถึง ระดับของการทำหน้าที่อย่างต่อเนื่องของระบบในกรณีที่อินพุตไม่ถูกต้อง หรือเงื่อนไขแวดล้อมที่บีบคั้น

โดยปราศจากการวางแผนและการควบคุมคุณภาพ อาจได้ผลิตภัณฑ์ที่เบี่ยงเบนไปจากมาตรฐาน

ลักษณะทางคุณภาพดังกล่าวข้างต้น เป็นลักษณะภายนอกของซอฟต์แวร์ที่ผู้ใช้ (End User) สนใจ ในขณะที่นักเขียนโปรแกรม (Programmer) จะสนใจลักษณะภายในของซอฟต์แวร์เท่าๆ กับลักษณะภายนอก ลักษณะเชิงคุณภาพภายใน ได้แก่

- **การดูแลรักษาได้ (Maintainability)** หมายถึง ความง่ายต่อการดัดแปลงแก้ไขระบบซอฟต์แวร์ เพื่อเปลี่ยนแปลงหรือเพิ่มความสามารถ ปรับปรุงสมรรถนะหรือแก้ไขข้อบกพร่อง

- **ความยืดหยุ่น (Flexibility)** หมายถึง ขอบเขตที่สามารถดัดแปลงแก้ไขระบบสำหรับการใช้งานหรือสภาวะแวดล้อมอื่นนอกเหนือจากที่ได้รับการออกแบบมาโดยเฉพาะ

- **การเคลื่อนย้ายได้ (Portability)** หมายถึง ความง่ายในการดัดแปลงแก้ไขระบบเพื่อให้ทำงานในสภาวะแวดล้อมต่างจากที่ได้รับการออกแบบมาโดยเฉพาะ

- **การนำกลับมาใช้ใหม่ (Reusability)** หมายถึง ขอบเขตและความง่ายในการนำบางส่วนของระบบหนึ่งไปใช้ในระบบอื่น

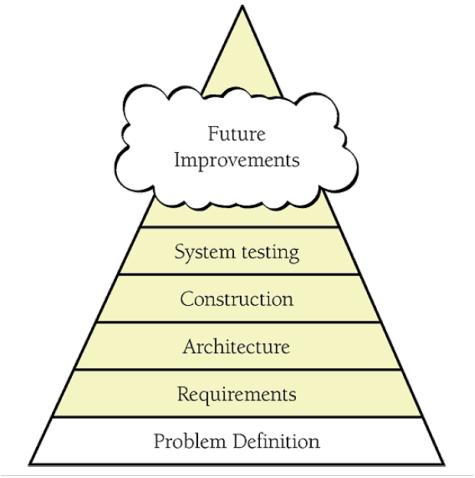
- **การอ่านง่าย (Readability)** หมายถึง ความง่ายในการอ่านและทำความเข้าใจรหัสคำสั่งต้นฉบับของระบบ โดยเฉพาะอย่างยิ่งในระดับของคำสั่งในรายละเอียด

- **การทดสอบได้ (Testability)** หมายถึง ระดับความสามารถในการทดสอบหน่วยย่อย (Unit Testing) และการทดสอบระบบ (System Testing) ระดับของความสามารถในการตรวจสอบว่าระบบมีลักษณะตรงตามความต้องการ

- **การทำความเข้าใจได้ (Understandability)** หมายถึง ความง่ายในการทำความเข้าใจระบบ ทั้งโครงสร้างของระบบ และระดับคำสั่งในรายละเอียด การทำความเข้าใจได้ต้องทำด้วยการเชื่อมโยงของระบบ

โดยปราศจากสถาปัตยกรรมและกระบวนการที่ดี อาจทำให้เกิดการสร้างผลิตภัณฑ์ที่ผิด
หรือสร้างผลิตภัณฑ์ที่ถูกด้วยวิธีที่ผิด เป็นผลให้โครงการไม่ประสบความสำเร็จ

ในการรับรองคุณภาพซอฟต์แวร์ ประเด็นที่ควรให้ความสนใจอีกประการหนึ่งคือกระบวนการ (Process) ถ้าต้องการเน้นคุณภาพในช่วงท้ายของโครงการ ส่วนใหญ่จะให้ความสำคัญกับการทดสอบ (Testing) การทดสอบเป็นสิ่งที่หลายคนนึกถึงเมื่อกล่าวถึงการรับรองคุณภาพซอฟต์แวร์ อย่างไรก็ตาม การทดสอบเป็นเพียงส่วนหนึ่งของยุทธศาสตร์การรับรองคุณภาพซอฟต์แวร์ แต่มิใช่ส่วนที่มีผลกระทบมากที่สุด การทดสอบไม่สามารถตรวจจับข้อบกพร่องอย่างเช่น การสร้างผลิตภัณฑ์ที่ผิด หรือการสร้างผลิตภัณฑ์ที่ถูกด้วยวิธีที่ผิด ข้อบกพร่องดังกล่าวควรจะถูกกำจัดไปแต่เนิ่นๆ ก่อนการทดสอบ หรือก่อนที่การสร้างผลิตภัณฑ์ซอฟต์แวร์จะเริ่มต้น



คุณภาพของซอฟต์แวร์ย่อมเกิดจากกระบวนการผลิตซอฟต์แวร์ หากทุกขั้นตอนของกระบวนการผลิตมีนโยบายด้านคุณภาพเป็นเป้าหมาย จะทำให้ซอฟต์แวร์ที่ผลิตเป็นซอฟต์แวร์ที่มีคุณภาพ

ถ้าต้องการเน้นคุณภาพในช่วงกลางของโครงการ จะให้ความสำคัญกับวิธีปฏิบัติในการสร้าง แต่ถ้าต้องการเน้นคุณภาพในช่วงเริ่มต้นโครงการ จะต้องมีการวางแผนและออกแบบการสร้างผลิตภัณฑ์ตั้งแต่แรก โดยกำหนดปัญหา ระบุวิธีแก้ปัญหา และออกแบบวิธีแก้ปัญหา

ดังนั้นด้วยระบบการประกันคุณภาพซอฟต์แวร์ ซึ่งเริ่มตั้งแต่กระบวนการวางแผนและการออกแบบที่ดีในระยะต้น รวมถึงการใช้กระบวนการที่มีคุณภาพ จะเป็นกุญแจหนึ่งไปสู่การพัฒนาผลิตภัณฑ์ซอฟต์แวร์อย่างมีประสิทธิภาพ และเป็นหัวใจของความสำเร็จของโครงการ

กรุงเทพมหานคร ศูนย์กลางด้านการปกครอง เศรษฐกิจ สังคมและวัฒนธรรม

กรุงเทพมหานครเป็นองค์กรปกครองส่วนท้องถิ่นขนาดใหญ่ กอปรด้วยผู้บริหารระดับสูงมีวิสัยทัศน์ ได้เล็งเห็นถึงความสำคัญในการนำเทคโนโลยีสารสนเทศมาใช้เพื่อเพิ่มประสิทธิภาพในการ

ดำเนินงานอย่างกว้างขวางในทุกหน่วยงาน เพื่อให้การขับเคลื่อนนโยบายพัฒนาองค์กรเป็นไปตามแผนยุทธศาสตร์ที่กำหนด ระบบสารสนเทศของกรุงเทพมหานครประกอบด้วยระบบงานต่างๆ ทั้งในส่วนที่หน่วยงานพัฒนาเอง และการพัฒนาจากภาคเอกชนโดยผ่านกระบวนการด้านจัดซื้อจัดจ้าง การประกันคุณภาพซอฟต์แวร์เป็นปัจจัยหนึ่งที่ช่วยควบคุมคุณภาพของระบบสารสนเทศให้ได้ตามมาตรฐานที่กำหนด และช่วยลดต้นทุนและภาวะการสูญเสียที่ไม่จำเป็น ทั้งในส่วนของเวลา แรงงาน และงบประมาณในการปรับปรุงแก้ไขระบบหรือการพัฒนาต่อยอดในระยะยาว

และนี่เป็นอีกหนึ่งความท้าทายของผู้บริหารเทคโนโลยีสารสนเทศของกรุงเทพมหานคร ในการพิจารณานำระบบการประกันคุณภาพซอฟต์แวร์มาใช้ร่วมกับการพัฒนาระบบสารสนเทศอย่างเป็นรูปธรรม เพื่อให้เกิดประโยชน์สูงสุดต่อองค์กรอย่างยั่งยืน

แหล่งข้อมูลอ้างอิง

1. Code Complete: A Practical Handbook of Software Construction. Steve McConnell, Second Edition, Microsoft Press. 960 pages, 2004.
2. Principles of Software Engineering Management. Tom Gilb. 1988.