บทความวิชาการ

การนำซอฟต์แวร์กลับมาใช้ใหม่ (Software Reuse)

เดือนเพ็ญ กชกรจารุพงศ์

Duenpen Kochakornjarupong Ph.D. (Artificial Intelligence in Education)

อาจารย์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยทักษิณ

Lecturer, Department of Mathematics, Faculty of Science, Thaksin University

1. บทนำ

การนำซอฟต์แวร์กลับมาใช้ใหม่นับเป็นกุญแจหลักของวิธีการดำเนินการด้านวิศวกรรมซอฟต์แวร์ รวมทั้งกลุ่มคนที่ทำงานเกี่ยวข้องกับซอฟต์แวร์ เนื่องจากเป็นเรื่องที่เกี่ยวข้องกับการเพิ่มคุณภาพของซอฟต์แวร์ให้มีวิธีการทำงานที่รวดเร็วยิ่งขึ้น โดยเฉพาะอย่างยิ่งในปัจจุบันอุตสาหกรรมซอฟต์แวร์มีการแข่งขันกันสูง เริ่มมีความต้องการซอฟต์แวร์มากขึ้นเรื่อยๆและทั่วโลกจะมีนักพัฒนาซอฟต์แวร์ไม่เพียงพอต่อความต้องการซอฟต์แวร์ เมื่อมีการพัฒนาเกี่ยวกับการนำซอฟต์แวร์กลับมาใช้ใหม่เพิ่มขึ้นเรื่อยๆ จึงทำให้เกิดงานวิจัยทางด้านนี้เพิ่มขึ้นมากมายในปัจจุบัน (เช่น Guo, 2003; Ali, and Du 2004; Aggarwal et al., 2005; Almeida et al., 2005; Frakes, and Kang, 2005; Mascena et al., 2005; Poulin 2006; Sherif et al., 2006; Burégio et al., 2007) ดังนั้นการนำซอฟต์แวร์กลับมาใช้ใหม่หรือการใช้ซอฟต์แวร์ที่มีอยู่แล้วอย่างมีประสิทธิภาพจะช่วยให้ประหยัดงบประมาณพร้อมทั้งยังลดค่าใช้จ่าย ช่วยเพิ่มประสิทธิภาพในการพัฒนาซอฟต์แวร์และเพิ่มผลกำไรให้กับผู้พัฒนา อีกทั้งยังลดเวลาในการผลิตซอฟต์แวร์เช่นกัน บทความเรื่องการนำซอฟต์แวร์กลับมาใช้ใหม่นี้จะกล่าวถึงความหมายของการนำซอฟต์แวร์กลับมาใช้ใหม่ ประเภทของการนำกลับมาใช้ใหม่ เหตุผลที่กล่าวว่าทำไมการนำซอฟต์แวร์กลับมาใช้ใหม่ในอดีตจึงไม่ได้รับความนิยม และการนำโปรแกรมประยุกต์บนเครือข่ายกลับมาใช้ใหม่

2. การนำซอฟต์แวร์กลับมาใช้ใหม่คืออะไร

การนำซอฟต์แวร์กลับมาใช้ใหม่คือกระบวนการสร้างและปรับปรุงระบบของซอฟต์แวร์โดยใช้องค์ประกอบเดิมที่เกิดจากการพัฒนาซอฟต์แวร์ที่มีอยู่แล้วได้แก่ ส่วนของโปรแกรมและเอกสารที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ ตัวอย่างส่วนของโปรแกรม ได้แก่ รหัสต้นทาง (source code) คลังโปรแกรม (library) คอมโพเน้นท์ (component) เป็นต้น ตัวอย่างองค์ประกอบเดิมของซอฟต์แวร์ ได้แก่ เอกสารความต้องการของระบบ (requirement documents) ข้อเสนอโครงการ (proposal) เอกสารประกอบการออกแบบ (design documents) แบบแผนการออกแบบ (design pattern) และสถาปัตยกรรมซอฟต์แวร์ (software architecture) เอกสารการพัฒนาและทดสอบโปรแกรม รวมถึงคู่มือการใช้โปรแกรมและชุดทดสอบโปรแกรม นอกจากนี้ยังรวมถึง เว็บเซอร์วิส (web services) ซีแมนติกเว็บเซอร์วิส (semantic web services) (Tjoa et al., 2005) โดยที่ซีแมนติกเว็บจะทำงานร่วมกับออนโทโลยี (ontology) เพื่อสนับสนุนการใช้

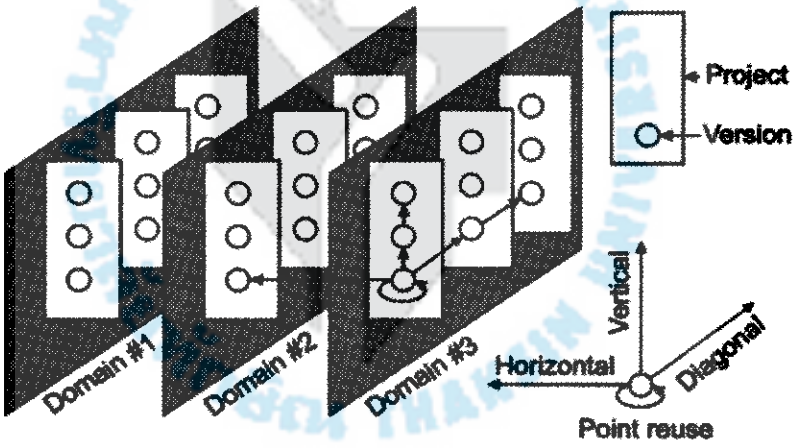
งานเว็บเซอร์วิส จากที่กล่าวมาข้างต้นล้วนเป็นองค์ประกอบที่เกิดจากความพยายามในการพัฒนาซอฟต์แวร์ให้สามารถนำกลับมาใช้ใหม่ได้อย่างมีศักยภาพนั่นเอง

3. ประเภทของการนำซอฟต์แวร์กลับมาใช้ใหม่

จอร์แดน (Jodan, 1997) ได้แบ่งประเภทของการนำซอฟต์แวร์กลับมาใช้ใหม่เป็น 2 ประเภท ตามทิศทางการใช้งาน คือ แบบแนวนอน (Horizontal reuse) และแบบแนวตั้ง (Vertical Reuse) ต่อมาโคแวค (Kovàcs, 1999) และคณะได้แบ่งประเภทของการนำซอฟต์แวร์กลับมาใช้ใหม่เพิ่มขึ้นอีก 2 ประเภท คือ แบบแนวทแยงมุม (Diagonal) และการนำจุดสำคัญกลับมาใช้ใหม่ (Point reuse) ดังรูปที่ 1

การนำซอฟต์แวร์กลับมาใช้ใหม่แบบแนวนอน (Horizontal reuse) เป็นการนำคอมโพเน้นท์ของซอฟต์แวร์กลับมาใช้ใหม่ในโดเมนที่แตกต่างกัน นั่นคือมีการนำคอมโพเน้นท์ของซอฟต์แวร์กลับมาใช้ใหม่ในโปรแกรมประยุกต์ทั่วๆ ไปที่แตกต่างกัน เช่น คลาสของลิงค์ลิสท์ ชุดคำสั่งในการจัดการสตริงหรือฟังก์ชันที่เกี่ยวกับส่วนประสานกับผู้ใช้ (GUI: Graphic User Interface)

การนำซอฟต์แวร์กลับมาใช้ใหม่แบบแนวดิ่ง คือ การนำโดเมนหรือฟังก์ชันของระบบกลับมาใช้ใหม่โดยใช้กับตระกูลของระบบที่มีหน้าที่การทำงานใกล้เคียงกัน แนวคิดดังกล่าวทำให้มีวิศวกรรมขอบเขต (Domain Engineering) เกิดขึ้น วิศวกรรมขอบเขตคือกระบวนการที่เป็นวงชีวิตที่มีการทำซ้ำได้ ที่เข้าใจได้ที่องค์กรนำมาใช้เพื่อวัตถุประสงค์ทางธุรกิจอย่างมีชั้นเชิง กระบวนการดังกล่าวสามารถเพิ่มผลิตภัณฑ์ของโครงงานด้านวิศวกรรม โปรแกรมประยุกต์ (Application Engineering) ผ่านมาตรฐานของผลิตภัณฑ์ที่เป็นประเภทเดียวกัน วิศวกรรมขอบเขตทำให้เกิดวิศวกรรมโปรแกรมประยุกต์ ซึ่งวิศวกรรมโปรแกรมประยุกต์ หมายถึงโครงงานที่สร้างผลิตภัณฑ์ เพื่อให้ตรงกับความต้องการของผู้ใช้ ฟอร์มและโครงสร้างของกิจกรรมของวิศวกรรมโปรแกรมประยุกต์จะสร้างจากขอบเขต ดังนั้นวิศวกรรมขอบเขตจะเน้นในเรื่องการสร้างและการบำรุงรักษาเพื่อนำชุดฟังก์ชันกลับมาใช้ใหม่ ส่วนวิศวกรรมโปรแกรมประยุกต์ เป็นการใช้งานที่เกี่ยวกับชุดฟังก์ชัน เพื่อสร้างผลิตภัณฑ์ใหม่



รูปที่ 1 ทิศทางการนำซอฟต์แวร์กลับมาใช้ใหม่ (Reuse directions)

การนำซอฟต์แวร์กลับมาใช้ใหม่แบบแนวทแยงมุม (Diagonal reuse) ตามแนวตั้ง (Vertical Reuse) และตามจุดสำคัญ (Point reuse) จะถูกจำกัดพื้นที่การนำมาใช้ภายในโดเมนหนึ่งที่เฉพาะเจาะจง แต่การนำซอฟต์แวร์กลับมาใช้ใหม่แบบแนวทแยงมุม เป็นการนำคอมโพเน้นท์ของซอฟต์แวร์กลับมาใช้ใหม่ ซึ่งมีขอบเขตอยู่ภายในโดเมนเดียวกัน แต่อยู่คนละโครงงาน ในขณะที่การนำซอฟต์แวร์กลับมาใช้ใหม่แบบแนวตั้ง เป็นการนำส่วนประกอบ (element) ของซอฟต์แวร์กลับมาใช้ใหม่ภายในโครงงานเดียวกัน แต่มาจากซอฟต์แวร์คนละเวอร์ชัน สำหรับการนำจุดสำคัญกลับมาใช้ใหม่ เป็นการนำคอมโพเน้นท์ของซอฟต์แวร์กลับมาใช้ใหม่ภายในซอฟต์แวร์เวอร์ชันเดียวกัน

ในปัจจุบันมีการจำแนกประเภทของการนำซอฟต์แวร์กลับมาใช้ใหม่ได้ 3 ประเภท ตามขนาดของงาน (กิตติ, 2550) นั่นคือ

1) การนำระบบของโปรแกรมประยุกต์กลับมาใช้ใหม่ (Application System Reuse)

การนำระบบของโปรแกรมประยุกต์กลับมาใช้ใหม่ เป็นการนำระบบเก่าที่พัฒนาไปแล้วกลับมาใช้ใหม่ทั้งระบบ โดยไม่มีการเปลี่ยนแปลงตัวระบบเลย แต่เปลี่ยนแปลงผู้ใช้ระบบเป็นกลุ่มใหม่ ที่มีความต้องการใช้ระบบคล้ายกับระบบเก่า

2) การนำคอมโพเน้นท์กลับมาใช้ใหม่ (Component Reuse)

การนำคอมโพเน้นท์กลับมาใช้ใหม่ เป็นการนำคอมโพเน้นท์ จากระบบเก่ามาใช้ในระบบใหม่ ระบบเก่ากับระบบใหม่อาจคล้ายกัน โดยมีการพัฒนาร่วมกันกับระบบใหม่ เรียกการพัฒนาลักษณะนี้ว่า วิศวกรรมเชิงชิ้นส่วน (Component-Based Software Engineering หรือ CBSE) ทั้งนี้การนำคอมโพเน้นท์จากระบบเก่ากลับมาใช้ใหม่นั้น จะต้องประกอบด้วยสภาพแวดล้อมดังต่อไปนี้ (Pressman, 2005)

- ความสามารถของฐานข้อมูลสำหรับการจัดเก็บคอมโพเน้นท์ของซอฟต์แวร์ และการจัดหมวดหมู่ของข้อมูลที่จำเป็นสำหรับการเรียกใช้งานคอมโพเน้นท์ของซอฟต์แวร์อีกครั้ง
- ระบบจัดการไลบรารี่สามารถเข้าถึงฐานข้อมูลดังกล่าวได้
- ระบบเรียกใช้งานคอมโพเน้นท์ของซอฟต์แวร์ที่สามารถเป็นแอพลิเคชั่นฝั่งไคลเอนท์ เพื่อให้เรียกใช้งานคอมโพเน้นท์และรับการบริการจากเซิร์ฟเวอร์ที่ให้บริการไลบรารี่ได้
- เครื่องมือ CBSE ที่สนับสนุนการรวมของคอมโพเน้นท์เพื่อการนำกลับมาใช้ได้ใหม่เพื่อนำไปสู่การออกแบบและพัฒนาระบบต่อไป

3) การนำอ็อบเจกต์และฟังก์ชันกลับมาใช้ใหม่ (Object and Function Reuse)

การนำอ็อบเจกต์และฟังก์ชันกลับมาใช้ใหม่สามารถทำได้โดยการนำอ็อบเจกต์และฟังก์ชัน จากระบบเก่ามาใช้พัฒนาในระบบใหม่ หรือเชื่อมโยงกับระบบอื่นที่ใช้อ็อบเจกต์และฟังก์ชันเหมือนที่ระบบใหม่ต้องการ

4. ทำไมการนำซอฟต์แวร์กลับมาใช้ใหม่ในอดีตจึงไม่ได้รับความนิยม

การนำซอฟต์แวร์กลับมาใช้ใหม่เป็นเรื่องที่มีการถกเถียงกันมากกว่าสามสิบปี (Schmidt, 2006) ในกลุ่มของผู้ที่ทำงานเกี่ยวข้องกับซอฟต์แวร์ ผู้พัฒนาระบบส่วนใหญ่จะใช้หลักการนำกลับมาใช้ใหม่ในบางโอกาส เช่นการตัดและแปะโค้ดจากโปรแกรมที่มีอยู่แล้วไปยังโปรแกรมใหม่ วิธีการนี้อาจเหมาะกับการพัฒนาซอฟต์แวร์ขนาดเล็ก ซึ่งอาจจะมีข้อจำกัดกับโปรแกรมเมอร์แต่ละคน หรือกลุ่มผู้พัฒนาโปรแกรมกลุ่มเล็กๆ แต่วิธีการนี้จะไม่เหมาะสมกับการพัฒนาซอฟต์แวร์ของหน่วยงานทางธุรกิจ

เพื่อจัดหาซอฟต์แวร์ที่นำกลับมาใช้ใหม่ได้อย่างมีระบบ สามารถลดเวลาและค่าใช้จ่ายในการพัฒนาคุณภาพของซอฟต์แวร์ สามารถทำได้โดยการสร้างและใช้ชุดการ

ใช้งานที่หลากหลาย เช่น สถาปัตยกรรม แบบแผน (pattern) คอมโพเน้นท์ และเฟรมเวิร์ค เช่น เฟรมเวิร์คที่ซับซ้อนของคอมโพเน้นท์ที่นำกลับมาใช้ได้ใหม่ ซึ่งจะอยู่ในรูปของภาษาเชิงวัตถุที่สามารถทำงานได้กับระบบปฏิบัติการต่างๆ เฟรมเวิร์คเหล่านี้จะเน้นการทำงานร่วมกับกลุ่มโดเมนที่สัมพันธ์กันเช่น ตัวประสานการทำงานระหว่างผู้ใช้ในรูปแบบกราฟฟิก หรือไลบรารี่ของ C++ (the Standard Template Library: STL) การนำซอฟต์แวร์กลับมาใช้ใหม่เป็นจุดเด่นที่สำคัญของการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) เนื่องจากการเขียนโปรแกรมเชิงวัตถุจะเน้นการเขียนโปรแกรมในรูปของวัตถุ ซึ่งจัดเป็นแนวทางหนึ่งของการจัดระบบโปรแกรม โดยจะเน้นที่แนวทางการออกแบบโปรแกรมโดยไม่ได้เน้นหนักเกี่ยวกับรายละเอียดของตัวดำเนินการแต่ละตัวนอกจากนี้แล้วการนำซอฟต์แวร์กลับมาใช้ใหม่นั้น ยังมีข้อจำกัดในการใช้งานของไลบรารี่และเครื่องมือที่ผู้พัฒนาไม่ได้สร้างขึ้นเอง ข้อจำกัดดังกล่าวอาจจัดการได้ยากกว่าการนำบางส่วนของซอฟต์แวร์ที่มีอยู่แล้วนำมาเป็นส่วนหนึ่งของการพัฒนาซอฟต์แวร์ให้กับองค์กรหนึ่งๆ

ดังที่ได้กล่าวมาแล้วนี้ จัดเป็นปัจจัยเชิงเทคนิคที่เป็นอุปสรรคต่อการนำซอฟต์แวร์กลับมาใช้ใหม่ ส่วนปัจจัยที่ไม่เกี่ยวข้องกับทางด้านเทคนิคในการพัฒนาซอฟต์แวร์กลับมาใช้ใหม่ (Schmidt, 2006) ได้แก่

- อุปสรรคทางด้านการจัดองค์กร ซึ่งเกี่ยวข้องกับเรื่องของความต้องการทางด้านธุรกิจ
- อุปสรรคทางด้านเศรษฐกิจ เช่นการลงทุนกับกลุ่มที่นำซอฟต์แวร์กลับมาใช้ใหม่
- อุปสรรคทางด้านการบริหารจัดการ เนื่องจากมีความยุ่งยากในการจัดหมวดหมู่ จัดเอกสารและค้นหา ชิ้นส่วนของซอฟต์แวร์ เพื่อนำกลับมาใช้ใหม่
- อุปสรรคทางด้านการเมือง เช่น กลุ่มผู้พัฒนาซอฟต์แวร์ให้นำกลับมาใช้ได้ใหม่นี้จะถูกจับตามองจากกลุ่มผู้พัฒนาโปรแกรมประยุกต์ในแง่ของความปลอดภัยที่อาจจะละเมิดชิ้นงานจากกลุ่มผู้พัฒนาโปรแกรมประยุกต์ไป
- อุปสรรคทางด้านจิตวิทยา เช่น กลุ่มผู้พัฒนาโปรแกรมประยุกต์อาจจะมีความพยายามในการนำซอฟต์แวร์กลับมาใช้ได้ใหม่ แต่ถ้าหากการจัดการในเรื่องนี้ยังคงขาดความเชื่อมั่น อาจจะขาดความมั่นใจในเรื่องของความสามารถทางด้านเทคนิค

อุปสรรคต่างๆ เหล่านี้อาจจะส่งผลให้ผู้พัฒนาขาดความรู้ ความชำนาญในเรื่องของแบบแผนการออกแบบขั้นพื้นฐาน (fundamental design patterns) เกี่ยวกับโดเมน ซึ่งทำให้ยากต่อความเข้าใจว่า ทำอย่างไรจึงจะสร้างเฟรมเวิร์คและคอมโพเน้นท์สำหรับนำกลับมาใช้ได้ใหม่ให้มีประสิทธิภาพได้

โดยสรุปแล้วสาเหตุที่การนำซอฟต์แวร์กลับมาใช้ใหม่ในอดีตไม่ประสบความสำเร็จ เนื่องจากการนำซอฟต์แวร์กลับมาใช้ใหม่นั้นต้องอาศัยหลักการวิธีการและความชำนาญในการพัฒนา แต่ในปัจจุบันเมื่อคำนึงถึงในทางปฏิบัติแล้ว การพัฒนาคอมโพเน้นท์และเฟรมเวิร์คของซอฟต์แวร์เพื่อนำกลับมาใช้ได้ใหม่ให้มีคุณภาพนั้นทำได้ยาก เพราะจะต้องอาศัยความชำนาญในการพัฒนาและการใช้งานเพื่อสนับสนุนการพัฒนาผู้เชี่ยวชาญในด้านการพัฒนาซอฟต์แวร์ให้นำกลับมาใช้ได้ใหม่ อย่างไรก็ตามการพัฒนาโปรแกรมประยุกต์ขนาดใหญ่ เมื่อถึงเวลาที่จะนำซอฟต์แวร์ขนาดใหญ่กลับมาใช้ใหม่นั้น จะต้องอาศัยทั้งเรื่องของเทคนิคและที่ไม่เกี่ยวกับเทคนิค ในกรณีที่ไม่เกี่ยวกับเรื่องของเทคนิคนั้นจะเกี่ยวข้องกับเรื่องของแรงขับเคลื่อนทางสังคมและเศรษฐกิจ เพราะจะส่งผลต่อการนำเทคโนโลยีมาใช้เช่นกัน (Schmidt, 2006)

5. การนำโปรแกรมประยุกต์บนเครือข่ายกลับมาใช้ใหม่

แม้ว่าการพัฒนาระบบคอมพิวเตอร์ได้เพิ่มขึ้นอย่างรวดเร็วในยุคปัจจุบัน พร้อมกับความเร็วในการรับส่งข้อมูลผ่านระบบเครือข่ายในอัตราความเร็วที่สูงขึ้นเรื่อยๆ แต่การออกแบบและพัฒนาโปรแกรมประยุกต์บนเครือข่ายยังต้องใช้ค่าใช้จ่ายที่สูง อีกทั้งยังมีข้อผิดพลาดในการใช้งานอยู่บ้าง ซอฟต์แวร์บางอย่างใช้งานกับระบบปฏิบัติการบางตัวไม่ได้ หรือใช้งานได้กับสถาปัตยกรรมที่แตกต่างกัน อย่างไรก็ตามการพัฒนาโปรแกรมประยุกต์บนเครือข่าย เพื่อให้นำกลับมาใช้ได้ใหม่สามารถทำได้โดยการแทรกซอฟต์แวร์ตัวกลาง (middle software) ลงไประหว่างโปรแกรมประยุกต์และระบบปฏิบัติการนั้น ทั้งนี้จะต้องอาศัยชั้นของโปรโตคอลของเครือข่ายและฮาร์ดแวร์ร่วมด้วย ซึ่งสิ่งเหล่านี้จะต้องใช้ค่าใช้จ่ายจำนวนมาก ซอฟต์แวร์ตัวกลางจะเข้าไปเชื่อมระหว่างโปรแกรมประยุกต์และฮาร์ดแวร์ระดับล่าง ร่วมกับโครงสร้างของซอฟต์แวร์ เพื่อให้โปรแกรมประยุกต์นั้นเชื่อมกับสิ่งเหล่านี้ได้ และเพื่อให้ง่ายต่อการรวมคอมโพเน้นท์ต่างๆ ที่ถูกพัฒนาจากผู้ผลิตทางด้านเทคโนโลยีที่หลากหลาย

ดังนั้นซอฟต์แวร์หรือโปรแกรมประยุกต์บนเครือข่ายอาจจะมีการนำกลับมาใช้ใหม่ได้อย่างมีระบบ จะต้องอาศัยปัจจัยต่อไปนี้

- ควรให้มีการพัฒนาคอมโพเน้นท์และเฟรมเวิร์คภายใต้เทคโนโลยีซอฟต์แวร์ตัวกลาง (middleware) เช่น CORBA, J2EE, และ .NET
- ควรให้มีการเพิ่มจำนวนผู้พัฒนาโครงงานจากสิบปีที่ผ่านมา โดยให้มาใช้เทคนิคของการออกแบบเชิงวัตถุ เช่น UML และ แบบแผน (pattern) และสนับสนุนให้มีการเขียนโปรแกรมโดยใช้ภาษาเชิงวัตถุ เช่น C++ , Java, และ C#

แนวโน้มเหล่านี้เหมาะกับงานด้านธุรกิจ เช่น การค้าอิเล็กทรอนิกส์ (electronic commerce) และเครือข่ายข้อมูล (data networking) ที่สามารถลดวงจรเวลาในการพัฒนาระบบที่อาจจะกระทบต่อความสำเร็จทางด้านธุรกิจ

นอกจากนี้การพัฒนาเว็บแอพพลิเคชั่น (web application) นับเป็นการพัฒนาโปรแกรมประยุกต์บนเครือข่ายอินเตอร์เน็ตที่ได้รับความนิยมมากในปัจจุบัน การพัฒนาเว็บแอพพลิเคชั่นในปัจจุบันผู้พัฒนามักจะนิยมใช้ เว็บเซอร์วิส (web services) ซึ่งเป็นชิ้นส่วนของโปรแกรมที่บริการประมวลผลข้อมูลตามที่มีการร้องขอจากโปรแกรมประยุกต์ (application) เนื่องจากเว็บแอพพลิเคชั่นในบางหน่วยงานอาจมีการประมวลผลที่เหมือนกัน ทำให้เกิดความซ้ำซ้อนและสิ้นเปลืองเวลาในการทำงาน จึงเกิดแนวความคิดในการพัฒนาเว็บเซอร์วิส เพื่อให้มีการนำชิ้นส่วนของโปรแกรมที่ต้องการมาใช้ซ้ำอีก เพื่อให้ลดเวลาในการพัฒนาโปรแกรม เช่น จากผลงานวิจัยของสิรยาและสุวิมล (2550) ได้สรุปว่าเว็บเซอร์วิสสำหรับบริการข้อมูลด้านบุคลากรของมหาวิทยาลัยทักษิณมีประสิทธิภาพอยู่ในระดับดีมาก ทำให้กระบวนการพัฒนาเว็บแอพลิเคชั่นด้านงานบุคลากรพัฒนาได้รวดเร็วยิ่งขึ้น เนื่องจากมีการเรียกใช้เว็บเซอร์วิสมาประกอบเป็นระบบงานที่สมบูรณ์ รวมทั้งประหยัดเวลาลดจำนวนบรรทัดในการเขียนโปรแกรม และลดการจัดการกับฐานข้อมูลด้วยเช่นกัน

อย่างไรก็ตามการสร้างชิ้นส่วนของซอฟต์แวร์ให้นำกลับมาใช้ได้ใหม่นั้นต้องอาศัยองค์กรที่มีความพร้อมทางด้านผู้พัฒนาและนักออกแบบ ที่สามารถแยกแยะแหล่งข้อมูล ซึ่งจะเป็นกุญแจสู่ความหลากหลาย (variability) ของโดเมนของโปรแกรมประยุกต์นั้นๆ การกำหนดและแยกแยะโปรแกรมประยุกต์บนเครือข่ายที่มีความซับซ้อนจะต้องอาศัยกระบวน การพัฒนาซ้ำแล้วซ้ำอีกเนื่องจากการออกแบบชิ้นส่วนของซอฟต์แวร์ให้นำกลับมาใช้ได้อีกให้มีความถูกต้องในครั้งแรกนั้นเป็นเรื่องที่ทำได้ยาก ดังนั้นนักออกแบบและพัฒนา

ส่วนใหญ่จึงนิยมใช้โมเดลวงจรชีวิตในการพัฒนาซอฟต์แวร์แบบน้ำตก (waterfall) แบบบนลงล่าง (top-down)

7. บทสรุป

จะเห็นได้ว่าการนำซอฟต์แวร์กลับมาใช้ใหม่จะช่วยเพิ่มความน่าเชื่อถือให้กับระบบใหม่เพราะระบบเก่าได้ผ่านการใช้งาน พบปัญหา แก้ไข และทดสอบมาแล้ว แต่จะต้องคำนึงถึงค่าใช้จ่ายในการบำรุงรักษาเพิ่มเติม หากไม่สามารถนำซอฟต์แวร์นั้นกลับมาใช้ใหม่ได้ แม้ว่าการนำซอฟต์แวร์กลับมาใช้ใหม่จะใช้เวลาในการพัฒนาซอฟต์แวร์น้อยลง เพราะใช้เวลาในการพัฒนาและตรวจสอบน้อยลง แต่ผู้พัฒนายังขาดเครื่องมือสนับสนุน เพราะเครื่องมือในการพัฒนาซอฟต์แวร์บางชนิดอาจไม่สนับสนุนการนำกลับมาใช้ใหม่ นอกจากนี้แล้วผู้พัฒนาซอฟต์แวร์จะต้องเข้าใจและใช้งานคอมโพเน้นท์ในระบบเก่าได้ดี หากไม่เข้าใจแล้วจะทำให้ขาดความท้าทายในการพัฒนาซอฟต์แวร์ เพราะนักพัฒนาซอฟต์แวร์นิยมสร้างหรือพัฒนาซอฟต์แวร์ใหม่มากกว่าการนำกลับมาใช้ใหม่ อย่างไรก็ตามการนำซอฟต์แวร์กลับมาใช้ใหม่จะช่วยลดความเสี่ยงในการพัฒนาซอฟต์แวร์ อีกทั้งยังช่วยเพิ่มทักษะให้กับผู้พัฒนาซอฟต์แวร์และช่วยให้ผู้ใช้ได้ระบบที่อาจคล้ายกับระบบเก่า เพราะคุ้นเคยกับระบบเดิม เช่น ส่วนต่อประสานกับผู้ใช้ (user interface) คล้ายกับระบบเดิมเพื่อลดข้อผิดพลาดในการใช้ระบบและทำให้ผู้ใช้ทำงานได้เร็วขึ้น

เอกสารอ้างอิง

กิตติ ภักดีวัฒนะกุล และพนิดา พานิชกุล (2550). วิศวกรรมซอฟต์แวร์ (Software Engineering). กรุงเทพฯ: บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด.

สิรยา สิทธิสาร และสุวิมล จุงจิตร์ (2550). การพัฒนาเว็บเซอร์วิสสำหรับบริการข้อมูลด้านบุคลากรของมหาวิทยาลัยทักษิณ. การประชุมวิชาการและเสนอผลงานวิจัย (การวิจัยเพื่อพัฒนาคุณภาพชีวิตอย่างยั่งยืน) มหาวิทยาลัยทักษิณ ครั้งที่ 17 ประจำปี 2550.

Aggarwal, K.K., Singh, Y., Kaur, A., and Malhotra, R. (2005). Software reuse metrics for object-oriented systems. The 3rd ACIS International Conference on Software Engineering Research, Management and Applications, IEEE/CS Press, MI, USA (2005), pp. 48–54.

Ali F. M., and Du W. (2004).Toward reuse of object-oriented software design models. Information and Software Technology, 46(8), pp. 499-517

Almeida, E.S.d., Alvaro, A., Lucrédio, D. Garcia, V.C., and Meira, S.R.d.L. (2005). A survey on software reuse processes. IEEE International Conference on Information Reuse and Integration (IRI 2005), IEEE/CS Press, Las Vegas, USA (2005).

Burégio, V.A.d.A., Almeida, E.S.d., Lucrédio, D., and Meira, S.R.d.L. (2007). Specification, design and implementation of a reuse repository. The 31st IEEE Annual International Computer Software and Applications (COMPSAC) Conference – Short paper, Beijing, China.

Douglas C. Schmidt. (2006). Why Software Reuse has Failed and How to Make It Work for You. Retrieved from http://www.cs.wustl.edu/~schmidt/reuse-lessons.html [Accessed on 28 November 2007] (an earlier version of this article appeared in the C++ Report magazine, January 1999, Last modified September 2006).

Frakes, W.B., and Kang, K. (2005). Software reuse research: status and future. IEEE Transactions on Software Engineering, 31(7), pp. 529–536.

Jiang Guo. (2003). Software reuse through re-engineering the legacy systems. Information and Software Technology, 45(9), pp. 597-609.

Kimberly Jordan. (1997). Software reuse. Retrieved from http://baz.com/kjordan/swse625/htm/tp-kj.htm [Accessed on 28 November 2007].

Kovács, G. L., Kopácsi, S., Nacsa, J., Haidegger, G. and Groumpos, P. (1999). Application of software reuse and object-oriented methodologies for the modelling and control of manufacturing systems. Computers in Industry, 39(3), pp. 177-189.

Krueger, C. (1992). Software reuse. ACM Computing Surveys, 24 (2), pp. 131–183.

Mascena, J.C.C.P., Almeida, E.S.d.,and Meira, S.R.d.L.(2005). A comparative study on software reuse metrics and economic models from a traceability perspective. IEEE International Conference on Information Reuse and Integration (IRI), IEEE/CS Press, Las Vegas, NV, USA (2005).

Poulin, J. (2006). The business case for software reuse: reuse metrics, economic models, organizational issues, and case studies. The 9th International Conference on Software Reuse – Tutorial Notes, Torino, Italy.

Roger S. Pressman. (2005). Software Engineering: A Practitioner's Approach. Singapore: McGraw-Hill.

Sherif, K., Appan, R. and Lin, Z. (2006). Resources and incentives for the adoption of systematic software reuse. International Journal of Information Management 26 (1), pp. 70–80.

Tjoa. A. M., Andjomshoaa A., Shayeganfar F., and Wagner R. (2005). Semantic Web Challenges and New Requirements. Proceedings of the 16th International Workshop on Database and Expert Systems Application (DEXA'05). IEEE Computer Society.