การพัฒนาระบบประมวลผลคะแนนโดยใช้เทคนิคการออกแบบซอฟต์แวร์แบบระดับชั้น และ Inversion of Control

ธนกร หวังพิพัฒน์วงศ์ และ ธนิต แซ่ตั้ง

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยกรุงเทพ
คณะวิทยาศาสตร์ มหาวิทยาลัยกรุงเทพ
Email: thanakorn.w@bu.ac.th, 7510700276@bu.ac.th

บทคัดย่อ

การศึกษาวิจัยนี้ เป็นการทดสอบแนวความคิดของการออกแบบซอฟต์แวร์แบบระดับชั้น ร่วมกับการใช้เทคนิค Inversion of Control ว่าสามารถเพิ่มความยืดหยุ่นของซอฟต์แวร์ ทำให้สามารถปรับเปลี่ยนหรือแก้ไขส่วนต่างๆ ได้ง่ายขึ้น ทำให้รองรับต่อการเปลี่ยนแปลงความต้องการ หรือกลยุทธ์ใหม่ๆ ขององค์กรได้หรือไม่ โดยดำเนินการทดสอบด้วยการนำมาใช้ในการออกแบบระบบประมวลผลคะแนน (Grading System) ซึ่งผลลัพธ์ที่ได้หลังจากการออกแบบและพัฒนาซอฟต์แวร์เสร็จเรียบร้อยแล้ว พบว่าการออกแบบซอฟต์แวร์แบบระดับชั้นร่วมกับเทคนิค Inversion of Control ช่วยทำให้การปรับปรุงหรือเปลี่ยนแปลงเงื่อนไขภายในซอฟต์แวร์ สามารถทำได้สะดวก และง่ายขึ้นจริง

คำสำคัญ– การออกแบบซอฟต์แวร์*, Inversion of Control, *IoC

1. บทนำ

การพัฒนาซอฟต์แวร์ในองค์กรที่มีทีมพัฒนาภายในของตนเองนั้น มักจะเกิดปัญหาในเรื่องการปรับเปลี่ยนความต้องการของซอฟต์แวร์ที่ได้จัดทำเสร็จเรียบร้อยแล้ว เพื่อให้รองรับกลยุทธ์ใหม่ๆ ขององค์กร ทำให้การพัฒนาหรือแก้ไขโปรแกรมทำได้ค่อนข้างช้า โดยเฉพาะอย่างยิ่งองค์กรที่มีการพัฒนาซอฟต์แวร์ใช้มาเป็นระยะเวลานาน และมีซอฟต์แวร์เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งในสถาบันการศึกษา

สาเหตุที่เป็นเช่นนั้น เพราะซอฟต์แวร์แต่ละตัวที่ถูกสร้างขึ้นนั้น “ผูกติด” กับความต้องการทางธุรกิจ และยิ่งไปกว่านั้น ซอฟต์แวร์แต่ละตัวที่ถูกสร้างขึ้นยังมีความสัมพันธ์เชื่อมโยงกันเองอย่างแนบแน่น ทำให้การเปลี่ยนแปลงเงื่อนไข หรือความต้องการทางธุรกิจภายในซอฟต์แวร์หนึ่งๆ จะมีผลกระทบกับซอฟต์แวร์อื่นๆ ที่เกี่ยวข้องกัน และเมื่อจำนวนซอฟต์แวร์มีมากขึ้น ผนวกกับความต้องการเปลี่ยนไปตลอดเวลา ทำให้ผู้พัฒนาต้องคอยตามแก้ซอฟต์แวร์ที่เกี่ยวข้องทั้งหมดให้ตรงตามเงื่อนไขใหม่ ส่งผลให้สิ้นเปลืองทั้งเวลา และทรัพยากรบุคคล

วิธีการหนึ่งที่สามารถนำมาช่วยแก้ปัญหาได้คือการออกแบบซอฟต์แวร์ให้มีลักษณะเป็นระดับชั้น (ดังรูปที่ 1) เพื่อลดความซับซ้อน และซอฟต์แวร์ที่จัดทำขึ้นจะมีลักษณะเหมือนกับการแบ่งโปรแกรมเป็นส่วนๆ และนำเอาส่วนของโปรแกรมที่ทำเสร็จแล้วนั้นมาประกอบกันเป็นแอพพลิเคชัน โดยมีเป้าหมายเพื่อให้การพัฒนาแอพพลิเคชั่นขนาดใหญ่ สามารถทำได้ง่ายขึ้น และส่วนประกอบต่างๆ ของซอฟต์แวร์ที่สร้างเสร็จแล้ว สามารถนำมาใช้ใหม่ได้ และเป้าหมายที่สำคัญที่สุดคือ การปรับเปลี่ยนแก้ไขในอนาคต สามารถทำได้ง่ายและรวดเร็ว [1]

รูปที่ 1. การออกแบ่งซอฟต์แวร์เป็นระดับชั้น

ปัจจุบัน มหาวิทยาลัยกรุงเทพใช้ระบบประมวลผลคะแนนนักศึกษาที่สร้างขึ้นด้วยแนวทางการเขียนซอฟต์แวร์แบบดั้งเดิม ทำให้เกิดปัญหาเมื่อผู้ใช้ต้องการปรับเปลี่ยนความต้องการบางประการ การเปลี่ยนแปลงและแก้ไขซอฟต์แวร์ทำได้ยากและมีความซับซ้อน อีกทั้งเมื่อมีความต้องการจากผู้ใช้เพิ่มเติม การเพิ่มเติมเงื่อนไขลงในระบบมีความยุ่งยาก และในบางกรณีไม่สามารถดำเนินการได้

การศึกษานี้จึงเป็นการนำแนวความคิดของการออกแบบและพัฒนาซอฟต์แวร์แบบระดับชั้น ร่วมกับเทคนิค Inversion of Control มาพัฒนาระบบประมวลผลคะแนนสำหรับนักศึกษา (Grading System) โดยมีสมมติฐานที่ว่าการออกแบบซอฟต์แวร์ที่เป็นระดับชั้นและพัฒนาระบบด้วย Inversion of Control จะมีความยืดหยุ่น ช่วยให้โปรแกรมนั้นสามารถรองรับการเพิ่มเติมความต้องการหรือปรับเปลี่ยนรูปแบบการทำงานในอนาคตได้ง่ายกว่าการพัฒนาโดยใช้ระบบเดิม

2. Inversion of Control

Inversion of Control เป็นแนวคิดในการพัฒนาระบบที่ต่างจากแนวความคิดในการพัฒนาระบบแบบเก่า ที่การพัฒนาระบบจะทำการเขียนข้อมูลส่วนที่เป็นการกำหนดการทำงานรวมเข้าไว้ในส่วนของโค้ดที่อยู่ภายในระบบ และโค้ดที่สร้างขึ้นไม่ได้มีการแบ่งลักษณะการทำงานอย่างชัดเจน ซึ่งทำให้ยากต่อการแก้ไข ปรับปรุง หรือเพิ่มเติมความสามารถอื่นๆ เข้าไปภายหลัง และยิ่งเป็นไปได้ยาก หากผู้พัฒนาต้องการนำโค้ดบางส่วนของระบบที่ได้พัฒนาเสร็จแล้ว ไปใช้กับระบบอื่นๆ ในอนาคต

ด้วยแนวคิดของการทำ Inversion of Control นั้น ผู้ออกแบบระบบจะแยกเอาส่วนของข้อมูลที่เป็นการกำหนดการทำงานต่างๆ ไปเขียนไว้ในส่วนที่เรียกว่า Container ซึ่งทุกครั้งเมื่อระบบเริ่มทำงาน ก็จะทำการอ่านค่าที่เป็นการกำหนดการทำงานจาก Container ก่อนเสมอ หลังจากนั้นจึงจะไปเรียกใช้งานโมดูลต่างๆ ตามแต่ที่ถูกกำหนดไว้ตามเงื่อนไขที่กำหนด ด้วยวิธีนี้ ทำให้ผู้พัฒนาสามารถนำโค้ดหรือโมดูลต่างๆ ที่ได้รับการพัฒนาและใช้งานได้ดี นำไปใช้กับระบบอื่นๆที่ต้องการในภายหลังได้ โดยไม่จำเป็นต้องเข้าไปทำการแก้ไขภายในโค้ดหรือโมดูลนั้นๆ ในบางกรณีอาจจะมีการแก้ไขปรับปรุงเพียงเล็กน้อยก็สามารถทำได้เช่นเดียวกัน [2]

นอกจากนี้ การเปลี่ยนแปลงที่เกิดขึ้นในภายใน Container ยังไม่มีผลกับโค้ดที่ได้ Deploy ไปแล้วเนื่องจากการอ่านค่าจาก Container นี้จะเป็นการอ่านในช่วงของการเปิดโปรแกรมใหม่ทุกครั้ง ทำให้เมื่อมีการเปลี่ยนแปลง หรือแก้ไขส่วนของ Container ผู้ใช้เพียงแค่ปิดและเปิดโปรแกรมใหม่เท่านั้น ทำให้มีข้อดีคือผู้พัฒนาไม่จำเป็นต้องเสียเวลา Deploy ระบบใหม่ทุกครั้ง

3. สถาปัตยกรรมแบบลำดับชั้น

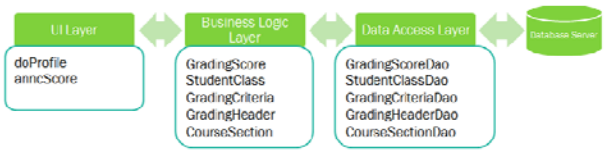
การออกแบบซอฟต์แวร์โดยใช้สถาปัตยกรรมแบบลำดับชั้น หมายถึง การออกแบบซอฟต์แวร์ที่เน้นการจัดกลุ่มของงานที่มีความสัมพันธ์กันและงานประเภทเดียวกันไว้ที่ชั้นเดียวกัน โดยแบ่งแต่ละลำดับชั้นให้ชัดเจนตามหน้าที่ของงานนั้นๆ โดยหน้าที่ในแต่ละชั้นจะถูกกำหนดด้วยกฎที่เหมือนกันหรือมีความสัมพันธ์กันภายในชั้น และแต่ละชั้นจะมีการติดต่อสื่อสารกัน เพื่อให้แต่ละชั้นทำหน้าที่ชัดเจนและมีหน้าที่ที่ต้องรับผิดชอบเฉพาะอย่าง เพื่อรองรับความยืดหยุ่นและการดูแลรักษาระบบ ที่เกิดขึ้นในอนาคต[3]

4. การออกแบบระบบประมวลผลคะแนนนักศึกษา

เครื่องมือที่ใช้พัฒนาระบบระบบประมวลผลคะแนนนักศึกษานั้น ประกอบด้วยภาษา C# ทำงานในรูปแบบ ASP.NET ที่จะประมวลผลในฝั่งเครื่องแม่ข่าย ร่วมกับระบบฐานข้อมูล SQL Server โดยตัวระบบประมวลผลคะแนนนักศึกษานี้ถูกออกแบบในลักษณะที่เป็นระดับชั้น โดยแบ่งเป็นสามชั้น คือ

1. ชั้นการเชื่อมต่อกับผู้ใช้ (User Interface Layer) คือชั้นที่เกี่ยวข้องกับการแสดงผล ตัวอย่างคลาสที่เกี่ยวข้องกับเลเยอร์นี้ เช่น
   - doProfile ใช้สำหรับการจัดการข้อมูลเบื้องต้นของรายวิชาและเกณฑ์การวัดผล
   - anncScore ใช้สำหรับการประกาศผลคะแนน
2. ชั้นเงื่อนไขทางธุรกิจ (Business Logic Layer) ในชั้นนี้จะเกี่ยวข้องกับกระบวนการและเงื่อนไขทางธุรกิจ ตัวอย่างเช่น
   - GradingCriteria ใช้สำหรับกำหนดเกณฑ์เกี่ยวกับการประมวลผลคะแนน
   - CourseSection ใช้สำหรับกำหนดเงื่อนไขในการเลือกวิชาและกลุ่ม
3. ชั้นการเข้าถึงข้อมูล (Data Access Layer) เป็นชั้นที่เกี่ยวข้องกับการติดต่อกับฐานข้อมูล ตัวอย่างเช่น
   - CourseSectionDao สำหรับการติดต่อกับระบบฐานข้อมูลเพื่อจัดการข้อมูลเกี่ยวกับวิชาและกลุ่ม
   - GradingScoreDao สำหรับการติดต่อกับระบบฐานข้อมูลเพื่อจัดการเกี่ยวกับข้อมูลคะแนน

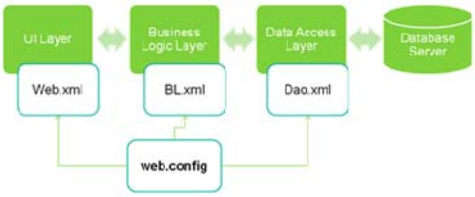
รูปที่ 2 แสดงตัวอย่างของคลาสในแต่ละเลเยอร์ของระบบประมวลผลคะแนน



รูปที่ 2. ตัวอย่างคลาสในแต่ละระดับชั้น

ในการทำ Inversion of Control ส่วนของไฟล์ที่ทำหน้าที่ควบคุมระบบจะต้องถูกแยกออกจากส่วนของโปรแกรมดังรูปที่ 3 โดยในระบบนี้จะประกอบด้วย 4 ไฟล์หลักที่ทำหน้าที่ควบคุมการทำงานของแต่ละระดับชั้นของโปรแกรม คือ

1. web.config ทำหน้าที่เป็นไฟล์หลักในการตั้งค่าระบบ
2. Dao.xml ทำหน้าที่กำหนดค่าเกี่ยวกับการใช้งานข้อมูลจากระบบฐานข้อมูล
3. BL.xml ทำหน้าที่กำหนดค่าเกี่ยวกับการทำงานในกระบวนการและเงื่อนไขทางธุรกิจ
4. Web.xml ทำหน้าที่กำหนดค่าเกี่ยวกับหน้าเวบแต่ละหน้า

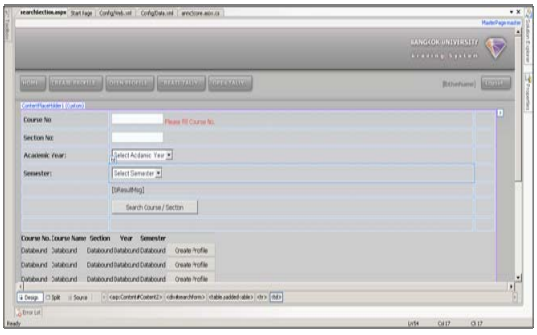


รูปที่ 3. ไฟล์ที่ทำหน้าที่ควบคุมระบบ

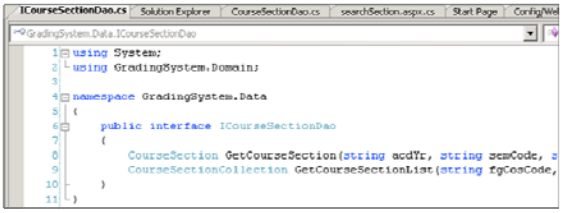
โดยไฟล์ควบคุมหลักนี้จะทำหน้าที่รวบรวมหรือประกอบส่วนต่างๆของโปรแกรมที่สร้างขึ้นเข้าด้วยกันเป็นโปรแกรมของแต่ละระดับชั้น โดยรูปที่ 4 แสดงตัวอย่างของไฟล์ Web.xml ที่แสดงให้เห็นถึงการทำ Inversion of Control โดยการประกอบ (Injection) คลาสที่ต้องการเข้าไปใน doProfile และรูปที่ 5 แสดงถึงหน้าจอ UI ในส่วนของการสร้าง Profile รายวิชา

รูปที่ 4. Web.xml

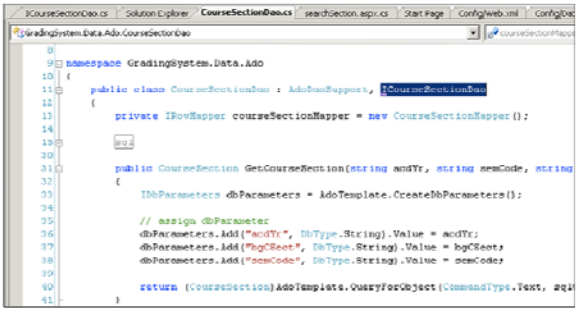
ในการสร้างระบบให้มีความยืดหยุ่น มีอิสระต่อกัน และมีลักษณะการเชื่อมต่อในแต่ละส่วนของการทำงานแบบหลวม (Loosely Couple) ซึ่งทำให้ผู้พัฒนาสามารถปรับเปลี่ยนเงื่อนไขหรือความต้องการได้โดยไม่กระทบกับโครงสร้างของโค้ดในระดับชั้นอื่น และสามารถนำมาประกอบกันเป็นส่วนของระบบได้นั้น สามารถดำเนินการได้โดยการสร้างอินเทอร์เฟซขึ้นมาเชื่อมต่อ ดังรูปที่ 6 เป็นตัวอย่างของการสร้าง Interface ICourseSectionDao และรูปที่ 7 แสดงการ Implement โค้ดของอินเทอร์เฟซ ICourseSectionDao



รูปที่ 5. หน้าจอแสดงผล Profile



รูปที่ 6. ตัวอย่าง Interface ICourseSectionDao

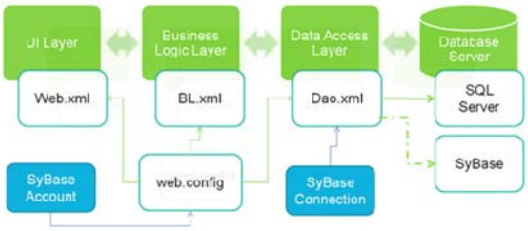


รูปที่ 7. ตัวอย่างการ Implement อินเทอร์เฟซ ICourseSectionDao

สังเกตว่าในกรณีที่มีการเปลี่ยนแปลงโค้ดภายใน CourseSectionDao จะไม่กระทบกับโค้ดส่วนอื่นๆ ที่มีการอ้างอิงถึง IcourseSectionDao ซึ่งหมายถึง การเปลี่ยนแปลงเงื่อนไขของผู้ใช้สามารถทำได้ โดยการแก้ไขโค้ดเฉพาะที่เท่านั้น โดยไม่ต้องไปตามแก้ไขในส่วนอื่นๆ ทั้งหมด ทำให้ระบบมีความยืดหยุ่น รองรับต่อการเปลี่ยนแปลงแก้ไขในอนาคต

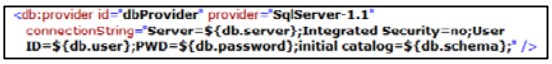
ในระหว่างการพัฒนานั้น ผู้พัฒนาสามารถสร้างเฉพาะส่วนของอินเทอร์เฟซขึ้นโดยยังไม่จำเป็นต้องมา Implement จริง และนำส่วนของอินเทอร์เฟซนั้นมาประกอบเพื่อสร้างเป็นแอพพลิเคชั่นเพื่อทดสอบการทำงานโดยรวมของระบบก่อนได้ และเมื่อระบบทำงานได้อย่างถูกต้อง ค่อยมา Implement โค้ดในภายหลัง

ในกรณีที่ต้องการปรับเปลี่ยนกระบวนการทำงานบางอย่างภายในระบบ ผู้พัฒนาสามารถปรับเปลี่ยนจากภายในไฟล์ควบคุมได้โดยง่าย เช่นการเปลี่ยนระบบฐานข้อมูลจาก SQL Server ไปเป็นฐานข้อมูล Sybase (รูปที่ 8) สามารถดำเนินการได้โดยการเปลี่ยนข้อมูลในไฟล์ Web.config เพื่อกำหนดข้อมูลการเชื่อมต่อกับฐานข้อมูล (รูปที่ 9) และเปลี่ยนแปลงข้อมูลในไฟล์ Dao.xml เพื่อเชื่อมต่อกับฐานข้อมูล Sybase (รูปที่ 10)



รูปที่ 8. แสดงการเปลี่ยนการเชื่อมต่อระบบฐานข้อมูล

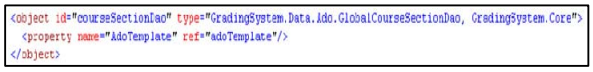
รูปที่ 9. การเปลี่ยนข้อมูลการ Logon เข้าฐานข้อมูล



รูปที่ 10. การเชื่อมต่อกับระบบฐานข้อมูล

นอกเหนือจากกรณีของการเปลี่ยนฐานข้อมูลแล้ว การเปลี่ยนคลาสต่างๆ ที่ใช้ในระบบก็สามารถทำได้ด้วยการเปลี่ยนแปลงข้อมูลในไฟล์ Container ที่ทำหน้าที่ควบคุมในการทำงานของระดับชั้นนั้นๆ เช่นในกรณีที่ผู้พัฒนาระบบต้องการนำคลาสที่ถูกพัฒนาขึ้นมาใหม่เพื่อไปแทนที่ Object ที่มี id=courseSectionDao ที่ถูกใช้อยู่เดิม ก็สามารถทำได้โดยการเปลี่ยนแปลงข้อมูลในไฟล์ Dao.xml จากเดิมที่กำหนด Type เป็นGradingSystem.Data.Ado.**OldClass** (รูปที่ 11) ก็เปลี่ยนให้เป็น GradingSystem.Data.Ado.**NewClass** (รูปที่ 12)

รูปที่ 11. Dao.xml แสดงการตั้งค่า courseSectionDao เดิม



รูปที่ 12. Dao.xml แสดงการตั้งค่า courseSectionDao ใหม่

จะเห็นว่าในการเปลี่ยน Class ดังกล่าวไม่กระทบในชั้นการทำงานอื่น เพราะชั้นการทำงานอื่นนั้นใช้การอ้างอิงถึง Class ที่ทำงานโดยใช้ id ซึ่งเป็นค่าที่ไม่มีการเปลี่ยนแปลง

5. บทสรุปและการอภิปรายผล

การศึกษาครั้งนี้เป็นการทดสอบแนวความคิดในการพัฒนาซอฟต์แวร์แบบระดับชั้นร่วมกับการใช้เทคนิค Inversion of Control โดยเริ่มจากการออกแบบคลาสที่เกี่ยวข้องกับในแต่ละระดับชั้น การกำหนดไฟล์ที่ทำหน้าที่ควบคุม และสุดท้ายพัฒนาระบบตัวอย่างเพื่อใช้งานจริง

แอพพลิเคชันที่ถูกเลือกขึ้นมาทดสอบในการศึกษานี้คือระบบประมวลผลคะแนนนักศึกษา เนื่องจากระบบเดิม ถูกพัฒนาด้วยวิธีการแบบเก่า ซึ่งทำให้การปรับปรุงแก้ไขตามความต้องการใหม่ๆ ทำได้ยาก

จากการทดลองออกแบบและพัฒนาพบว่า การออกแบบซอฟต์แวร์แบบระดับชั้นทำให้โปรแกรมที่เขียนในแต่ละส่วนมีความชัดเจนมากขึ้น ผู้ออกแบบมีความเข้าใจว่าส่วนใดคือระดับชั้นในส่วนการติดต่อกับผู้ใช้ (User Interface Layer) ส่วนใดคือส่วนของเงื่อนไขและกระบวนการทางธุรกิจ (Business Layer) และส่วนใดคือการติดต่อกับระบบฐานข้อมูล (Data Access Layer) นอกจากนี้การใช้อินเทอร์เฟซทำให้ส่วนของคลาสที่สร้างขึ้นมีความเป็นอิสระต่อกัน ซึ่งส่งผลดีต่อการนำกลับมาใช้ใหม่ นอกจากนี้ยังพบว่าระบบมีความยืดหยุ่น การปรับปรุงหรือเปลี่ยนแปลงเงื่อนไขใหม่ๆ ทำได้ง่ายขึ้น ทำให้ระบบมีความสามารถรองรับความต้องการใหม่ๆ ที่เพิ่มขึ้นในอนาคต

อย่างไรก็ตามข้อจำกัดของการออกแบบและพัฒนาระบบโดยใช้เทคนิคที่กล่าวมานั้น ผู้ออกแบบจะต้องมีความเชี่ยวชาญในความต้องการของซอฟต์แวร์ที่ต้องจัดทำขึ้นเพื่อให้ตอบสนองต่อกระบวนการทำงานต่างๆ ของธุรกิจนั้นๆ เป็นอย่างดี เพื่อที่จะได้นำมาออกแบบการทำงานในแต่ละระดับชั้นได้อย่างถูกต้อง สามารถแยกส่วนประกอบต่างๆ ของซอฟต์แวร์ได้อย่างชัดเจน ซึ่งอาจจะมีปัญหากับการพัฒนาซอฟต์แวร์ขนาดใหญ่ได้

เอกสารอ้างอิง

[1] J.W. Kim and K.J. Lim, “An Approach to Service-Oriented Architecture using Web Service and BPM in the Telecom-OSS Domain”, **Internet Research**. 17(1). 2007, pp. 99-107.

[2] M. Fowler, “Inversion of control containers and the dependency injection pattern” Retrieve by 10 July 2009 from http://martinfowler.com/articles/injection.html. 2004.

[3] M. P. Team, Microsoft Application Architecture Guide (Patterns & Practices) 2nd ed. Washington: Microsoft Press. 2009.