บทที่ 3

การพัฒนาเชิงชิ้นส่วนและเทคโนโลยีที่เกี่ยวข้อง

3.1 การพัฒนาเชิงชิ้นส่วน (Component-Based Development)

หลักการและแนวคิดที่สำคัญของการพัฒนาเชิงชิ้นส่วนผ่านทางเทคโนโลยีเว็บ อาศัยการทำงานร่วมกันระหว่างชิ้นส่วนที่เป็นอิสระต่อกันนำมาประกอบกันด้วยส่วนการเชื่อมโยงที่เหมาะสม นอกจากนี้ยังต้องคำนึงถึงแนวคิดของการนำกลับมาใช้ใหม่และเทคโนโลยีต่าง ๆ ที่สนับสนุนแนวคิดของการพัฒนาเชิงชิ้นส่วน

3.1.1 แนวคิดของการพัฒนาเชิงชิ้นส่วน

การพัฒนาเชิงชิ้นส่วนเป็นการนำแนวคิดของการทำงานร่วมกันระหว่างชิ้นส่วนผ่านส่วนการเชื่อมโยงที่มีการออกแบบให้สามารถนำชิ้นส่วนมาประกอบเพื่อการทำงานได้ ดังนั้นองค์ประกอบที่สำคัญของการพัฒนาเชิงชิ้นส่วน คือ ชิ้นส่วน (component) และส่วนการเชื่อมโยงระหว่างชิ้นส่วน (interface) โดยขอกล่าวรายละเอียดโดยสรุปดังต่อไปนี้

3.1.1.1 ชิ้นส่วน (Component)

คำนิยามของคำว่าชิ้นส่วนปรากฏอยู่มากมายตามความเข้าใจของผู้พัฒนาแต่ละคน แต่โดยแนวคิดแล้วก็จะมีลักษณะเหมือนหรือคล้ายคลึงกัน ดังนี้

ความหมายของชิ้นส่วนโดยมุมมองทั่วไปของ John Chessman ใน [Chessman, 1998]

1. มุมมองของการบรรจุหีบห่อ (packaging perspective)
   ชิ้นส่วนเป็นหน่วยของการบรรจุหีบห่อ สามารถกระจายหรือส่งมอบได้
2. มุมมองของการให้บริการ (service perspective)
   ชิ้นส่วนเป็นผู้จัดเตรียมการให้บริการในส่วนของการดำเนินการและหน้าที่การทำงานต่าง ๆ
3. มุมมองความคงเส้นคงวาของข้อมูล (integrity perspective)
   ชิ้นส่วนเป็นข้อมูลที่มีความคงเส้นคงวา หรือ เป็นขอบเขตที่มีการห่อหุ้ม (encapsulation boundary)

ตัวอย่างชิ้นส่วนตาม 3 มุมมองข้างต้นแสดงในตารางที่ 3.1

ตารางที่ 3.1 ตารางแสดงตัวอย่างของชิ้นส่วนตามมุมมองของการให้คำนิยาม

Packaging	Service	Integrity
Files, documents, directories	Database services	Databases
Source code files	Operating system services	Operating system
Class libraries	Function libraries	Framework
Template, tables	System utilities	Active X controls
Executables files , dll files	Individual API functions	Some COM classes
	COM classes	Java Applets
		Applications
		Complete APIs

ที่มา : ข้อมูลจากเว็บ Http://www.cbdege.com/cbdweb/technology/whatComp.html.

Brown ได้แบ่งกลุ่มการให้นิยามคำว่าชิ้นส่วนเป็น 4 กลุ่ม [Brown, 1996] ดังนี้

1. **Component**

"A component is a non-trivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interface."

2. **Run-time software component**

" A dynamic bindable package of one or more programs managed as a unit and accessed through documented interfaces that can be discovered at run time."

3. **Software component**

"A unit of composition with contractually specified and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party"

4. **Business component**

"The software implementation of an "autonomous" business concept or business process. It consists of all the software artifacts necessary to express, implement and deploy the concept as a reusable element of a larger business system"

นอกจากนี้ยังมีนิยามของชิ้นส่วน อื่น ๆ อีก ได้แก่

“A component is a coherent package of software that can be independently developed and delivered as a unit, and that offers interfaces by which it can be connected, unchanged, with other components to compose a larger system.”[D’Souza and Will, 1997]

“A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.” [Szyperski, 1998]

3.1.1.2 **ส่วนการเชื่อมโยงหรืออินเทอร์เฟซ** (Interface)

อินเทอร์เฟซ คือ ชุดของการดำเนินการ ซึ่งมีการบริการถูกจัดเป็นกลุ่มในหน่วยของ coherent contractual [Chessman, 1998]

อินเทอร์เฟซเป็นตัวบอกถึงชุดของการดำเนินการและระบุตัวตนของมันรวมทั้งตัวโพรโตคอลที่ใช้ จุดสำคัญของอินเทอร์เฟซ คือ พฤติกรรมของบริการที่ให้กับชิ้นส่วนเพื่อการทำงานร่วมกันกับชิ้นส่วนอื่น ๆ

อินเทอร์เฟซใช้ในการระบุการบริการจะเลือกการให้บริการจากชุดของการดำเนินการซึ่งมีพฤติกรรมที่ระบบสนใจโดยตัวชิ้นส่วน ซึ่งแต่ละชิ้นส่วนอาจใช้อินเทอร์เฟซตัวเดียวกันได้ Kruchten ให้คำนิยามของอินเทอร์เฟซว่า อินเทอร์เฟซเป็นการระบุถึงจุดเข้าถึง (access point) ของชิ้นส่วน ถ้าจะนิยามโดยกระชับ อินเทอร์เฟซก็คือกลุ่มของการดำเนินการซึ่งใช้ในการระบุการบริการของชิ้นส่วน [Kruchten, 1998]

ในตัวอินเทอร์เฟซ แต่ละความหมายของตัวดำเนินการที่ถูกระบุข้อกำหนดนี้จะให้บริการทั้งผู้พัฒนาอินเทอร์เฟซของและผู้ใช้งานอินเทอร์เฟซ ตัวอินเทอร์เฟซมีความสำคัญมากต่อการประกอบกันได้และการปรับปรุงรูปแบบให้เหมาะสมของชิ้นส่วนโดยผู้ใช้ โดยการอนุญาตให้มีการค้นหาชิ้นส่วนที่เหมาะสมและเข้าใจในวัตถุประสงค์การให้บริการ หน้าที่ การใช้งานและข้อกำหนดต่าง ๆ ของชิ้นส่วนนั้น ๆ [Beck, 1994]

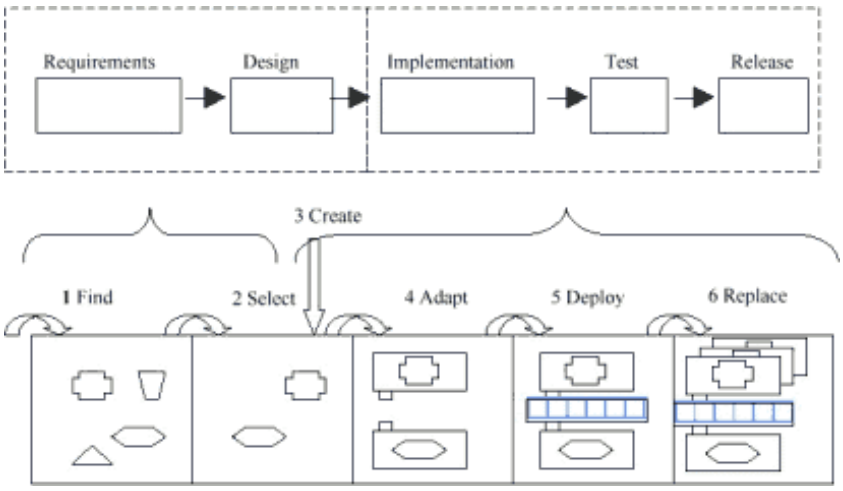
การอธิบายอินเทอร์เฟซของชิ้นส่วนประกอบไปด้วย ส่วนการยืนยันตัวตน (signature part) ซึ่งเป็นส่วนที่ใช้อธิบายการดำเนินการที่จัดเตรียมโดยชิ้นส่วน และ ส่วนพฤติกรรม

(behavior part) ที่ใช้อธิบายพฤติกรรมของชิ้นส่วน ซึ่งเทคนิคส่วนใหญ่ที่ใช้ในการอธิบายส่วนต่าง ๆ เหล่านี้ เช่น IDL (interface definition language) จะเกี่ยวข้องกับส่วนยืนยันตัวตน เพียงอย่างเดียว

ชิ้นส่วนสามารถได้รับบริการโดยอินเทอร์เฟซหนึ่งตัวหรืออาจจะมากกว่าและสามารถใช้อินเทอร์เฟซมาจากชิ้นส่วนอื่น ๆ ได้ ด้วยเหตุนี้ อินเทอร์เฟซที่พัฒนาขึ้นมาต้องเข้ากันได้กับบริการที่ชิ้นส่วนจัดเตรียมไว้ และอินเทอร์เฟซที่ใช้จากชิ้นส่วนอื่นต้องเข้ากันได้กับบริการของอีกชิ้นส่วน

3.1.2 **วัฏจักรชีวิตของการพัฒนาเชิงชิ้นส่วน**

วัฏจักรของการพัฒนาระบบของระบบงานเชิงชิ้นส่วนมีความแตกต่างจากวัฏจักรของการพัฒนาซอฟต์แวร์แบบดั้งเดิม เช่น แบบน้ำตก (waterfall) หรือที่เรียกกันทั่วไปว่าแบบประเพณีนิยม แบบวงเกลียว (spiral) และแบบสร้างต้นแบบ (prototyping) แต่อย่างไรก็ตามวัฏจักรของการพัฒนางานเชิงชิ้นส่วนในแต่ละขั้นตอนก็สามารถนำมาเปรียบเทียบกับการพัฒนาระบบงานในรูปแบบดั้งเดิมเช่น แบบน้ำตกได้ โดยขั้นตอนของการค้นหาชิ้นส่วนที่เหมาะสมและขั้นตอนการเลือกชิ้นส่วนสำหรับระบบงานในวัฏจักรของการพัฒนาระบบงานเชิงชิ้นส่วน ก็เทียบได้กับ ขั้นตอนของการสำรวจความต้องการและขั้นตอนการออกแบบระบบในแนวทางแบบน้ำตก นอกจากนี้ขั้นตอนในการพัฒนา ทดสอบ และบำรุงรักษาระบบของแนวทางแบบน้ำตกก็เทียบได้กับขั้นตอนการสร้างชิ้นส่วน ดัดแปลงชิ้นส่วน ประกอบชิ้นส่วนและการแทนที่ชิ้นส่วนใหม่ที่เหมาะสมของการพัฒนาระบบงานเชิงชิ้นส่วน แสดงดังภาพประกอบ 3.1



ภาพประกอบ 3.1 แสดงการเปรียบเทียบกระบวนการของแนวทางแบบน้ำตก (รูปบน) และแนวทางเชิงชิ้นส่วน(รูปล่าง) ของระบบงาน

[ที่มา : Http://infoeng.ee.ic.ac.uk/~malikz/surprise2001/nr99e/article2/]

กระบวนการในวัฏจักรของการพัฒนาระบบงานเชิงชิ้นส่วน จากภาพประกอบ 3.1 มีรายละเอียดดังนี้

1. **ค้นหาชิ้นส่วนที่เหมาะสมกับระบบงาน** (Find)

การค้นหาชิ้นส่วนเป็นการทำรายการของชิ้นส่วนที่มีความเป็นไปได้ต่อการพัฒนาระบบงาน ซึ่งขึ้นอยู่กับแต่ละระบบงานที่จะพัฒนา ชิ้นส่วนเหล่านี้จะถูกเก็บไว้ในห้องสมุดของชิ้นส่วนหรือคลังชิ้นส่วน ในขั้นแรกของการค้นหาจะใช้ข้อกำหนดของระบบงานเป็นตัวระบุชิ้นส่วน หลังจากนั้นจะพิจารณาถึงการอินเทอร์เฟซของชิ้นส่วนด้วยพฤติกรรมที่ชิ้นส่วนนั้นมีต่อกระบวนการประกอบกันได้ของชิ้นส่วน ซึ่งข้อกำหนดเหล่านี้จะระบุถึงการโต้ตอบของชิ้นส่วนที่อยู่ในรูปของคำนิยามของตัวดำเนินการที่เป็นไปได้ทั้งหมดที่สามารถกระทำกับชิ้นส่วนได้และให้ผลตามที่ต้องการ ประการสุดท้าย ขึ้นอยู่กับความเหมาะสมของเครื่องมือ และสถาปัตยกรรมของแพล็ตฟอร์มที่เลือกใช้ในการพัฒนาระบบงาน คลังชิ้นส่วนที่สามารถนำชิ้นส่วนกลับมาใช้ใหม่ได้นั้นประกอบไปด้วยชิ้นส่วนชนิดต่าง ๆ มากมาย การเลือกชิ้นส่วนจะทำได้ง่ายขึ้นกรณีเป็นชิ้นส่วนที่ถูกกำหนดโดย แพล็ตฟอร์ม ซึ่งเครื่องมือที่ใช้ในการค้นหาชิ้นส่วน ได้แก่ Agora [Seacord, 1998], Jcentral [Jcentral, 2001]

2. **การเลือกชิ้นส่วนที่ต้องการ** (Select)

เมื่อมีการสร้างรายการชิ้นส่วนที่มีความเป็นไปได้ในการใช้งานจากขั้นตอนที่ 1 แล้วนั้นในขั้นตอนต่อไปคือการเลือกชิ้นส่วนที่เหมาะสมกับความต้องการของระบบจากรายการเหล่านั้นกระบวนการของการเลือกชิ้นส่วนนั้นจะรวมทั้งการประเมินแต่ละชิ้นส่วนในส่วนของหน้าที่การทำงานและความสามารถในการดัดแปลงตามความต้องการของระบบ ซึ่งปัญหาของกระบวนการเหล่านี้แก้ไขได้โดยการใช้ “ warpper ” ซึ่งเป็นการเพิ่มรหัสโปรแกรม (code) ห่อหุ้มชิ้นส่วนเดิมเพื่อให้มีการใช้งานหรือไม่ใช้งานหน้าที่ใด ๆ ของชิ้นส่วนได้โดยการกำหนดค่าให้กับอินเทอร์เฟซของชิ้นส่วน ความสามารถในการบำรุงรักษาและความทนทานของชิ้นส่วนเป็นอีกหัวข้อหนึ่งที่นักพัฒนาที่เป็นผู้นำเอาแต่ละชิ้นส่วนมาพัฒนาเป็นแอพพลิเคชั่นก็สามารถที่จะทำการบำรุงรักษาและตรวจสอบข้อผิดพลาดของชิ้นส่วนได้

ในการลดความเสี่ยงของการเลือกชิ้นส่วนผิด และเพื่อรักษาความปลอดภัยในการบำรุงรักษา การพัฒนาของชิ้นส่วน ทำได้โดยการประเมินผู้พัฒนาชิ้นส่วน เพื่อสร้างความแน่ใจในการดำเนินการร่วมกันของผู้พัฒนาระบบงานกับผู้พัฒนาชิ้นส่วนผ่านทางการติดต่อกันอย่างถูกต้องตามกฎหมาย การจำกัดจำนวนผู้ค้าเพื่อลดต้นทุนและความไม่เป็นอิสระของชิ้นส่วน การซื้อชิ้นส่วนที่สามารถนำมาขยายต่อได้ในอนาคตเพื่อให้ง่ายต่อการจัดการระบบงานเพื่อการพัฒนาต่อไปและการได้รหัสโปรแกรมต้นฉบับมาด้วยก็เป็นเทคนิคหนึ่งที่ใช้ในการประเมินในขั้นตอนการเลือกชิ้นส่วนเพื่อเป็นการแน่ใจว่าการพัฒนาระบบต่อไปข้างหน้าจะไม่มีอุปสรรค

3. **การสร้างชิ้นส่วนขึ้นเอง** (Create)

หลังจากได้มาซึ่งชิ้นส่วนที่ต้องการแล้ว แต่มีความต้องการบางอย่างที่ไม่มีชิ้นส่วนที่เหมาะสมหรือไม่สามารถดัดแปลงให้เหมาะสมได้ จึงต้องมีการสร้างชิ้นส่วนบางชิ้นขึ้นมาใหม่ ในขั้นตอนนี้ชิ้นส่วนซึ่งไม่มีอยู่ในตลาดชิ้นส่วนก็อาจจะถูกเขียนขึ้นโดยผู้พัฒนาเอง

4. **การดัดแปลงชิ้นส่วน** (Adapt)

ชิ้นส่วนที่ถูกเลือกนั้นอาจต้องมีการดัดแปลงเพื่อให้มีความเหมาะสมกับต้นแบบของชิ้นส่วนที่มีอยู่แล้วหรือตามความต้องการที่กำหนดไว้ การดัดแปลงนั้นจะต้องมีพื้นฐานอยู่บนกฎที่แน่ใจได้ว่าตัวอุปสรรคในชิ้นส่วนเดิมจะลดลง ซึ่งระดับของการดัดแปลงนั้นก็ขึ้นอยู่กับโครงสร้างภายในของชิ้นส่วนนั้น ๆ ซึ่งอาจมองเป็น 3 ระดับคือ white box, gray box และ black box ในการกรณีของ white box จะสามารถมองเห็นถึงรหัสโปรแกรมต้นฉบับ จึงสามารถทำการดัดแปลง

โปรแกรมต้นฉบับได้ ในกรณีของ gray box จะไม่สามารถเปลี่ยนแปลงรหัสโปรแกรมต้นฉบับได้ แต่สามารถดัดแปลงชิ้นส่วนได้โดยใช้ภาษาที่ใช้ในดัดแปลงซึ่งชิ้นส่วนเป็นตัวจัดเตรียมให้เองหรือที่เรียกว่า API (Application Programming Interface) ส่วนในกรณีของ black box จะมีเพียงแค่รูปแบบของแฟ้มข้อมูลไบนารี (binary file) ของชิ้นส่วนเท่านั้นที่เรามองเห็นซึ่งหมายความว่าจะต้องมีการใช้เทคนิคพิเศษเพื่อทำการดัดแปลงชิ้นส่วนเช่น wrapping, bridging และ mediating เป็นต้น

5. **การประกอบชิ้นส่วน** (Deploy)

ชิ้นส่วนจะถูกประกอบเข้าด้วยกันผ่านทางตัวอินเทอร์เฟซที่เหมาะสม ซึ่งโดยส่วนใหญ่จะมีโครงสร้างสนับสนุน (framework) ที่ช่วยในการพัฒนาระบบเชิงชิ้นส่วน รูปแบบของฐานข้อมูลที่ข้อมูลการดำเนินการทั้งหมดถูกควบคุมจากศูนย์กลางจะสามารถใช้ชิ้นส่วนร่วมกันได้ นอกจากนี้ยังมีรูปแบบฐานข้อมูลแบบกระจายซึ่งต้องอาศัยข้อมูลการเข้าจังหวะในการส่งการติดต่อ หรือ รูปแบบ object require broker (ORB) ซึ่งมีการจัดเตรียมกลไกสำหรับการนิยามตัวอินเทอร์เฟซซึ่งใช้ในการประกอบกันของชิ้นส่วนที่อยู่ต่างที่กัน ตัวเชื่อมต่อ (connector) ใช้ในการโต้ตอบระหว่างชิ้นส่วน [Shaw, 1996] ซึ่งขั้นตอนนี้จะมีเครื่องมือช่วยทำให้โดยอัตโนมัติ หรือที่รู้จักกันโดยทั่วไปว่า สิ่งแวดล้อมพัฒนาการทำงานแบบโต้ตอบ (IDE : interactive development environment)

6. **การแทนที่ชิ้นส่วน** (Replace)

หลังจากที่ระบบงานเชิงชิ้นส่วนพัฒนาขึ้นแล้วและมีการใช้งานไปสักระยะหนึ่ง งานนั้นก็จะกลายเป็นงานเวอร์ชั่นเก่า ซึ่งเมื่อมีการปรับปรุงให้เป็นเวอร์ชั่นใหม่เพื่อตอบสนองต่อความต้องการที่เปลี่ยนไปหรือความต้องการที่เพิ่มขึ้นรวมทั้งการแก้ไขข้อผิดพลาดของระบบเดิม ก็จะต้องมีการแทนที่ชิ้นส่วนเดิมด้วยชิ้นส่วนใหม่โดยการพิจารณาถึงความเป็นไปได้ของชิ้นส่วนใหม่ทั้งในเรื่องของหน้าที่ที่ให้บริการและการเข้ากันได้กับระบบงานเดิมที่มีอยู่แล้วด้วย

กระบวนการของการพัฒนาระบบงานเชิงชิ้นส่วนนี่จะเป็นวัฏจักรตามความต้องการที่เปลี่ยนไปตามกาลเวลา

3.1.3 การนำกลับมาใช้ใหม่

ปัจจัยหลักที่สำคัญในการสนับสนุนการนำชิ้นส่วนกลับมาใช้ใหม่

3.1.3.1 ชิ้นส่วนที่สามารถนำกลับมาใช้ใหม่ได้ (reusable component)

ก. รูปแบบการนำกลับมาใช้ใหม่ (reuse forms)

ความสามารถในการนำกลับมาใช้ใหม่ (reusability) เป็นคุณสมบัติที่

สำคัญของชิ้นส่วนในแนวคิดการพัฒนาเชิงชิ้นส่วน โดย Prieto-Diaz ใน [Pohthong, 2000] ได้ให้มุมมองของการนำกลับมาใช้ใหม่เป็น 6 มุมมอง ดังต่อไปนี้

1. โดยสสาร (substance)

- แนวคิด(ideas, concepts) : เกี่ยวข้องกับการนำแนวคิดกลับมาใช้ใหม่ เช่น แนวคิดในการแก้ปัญหาโดยทั่วไป หรืออัลกอริธึมที่ใช้งานโดยทั่วไป

- สิ่งประดิษฐ์ (artefacts), ชิ้นส่วน (component) : เกี่ยวข้องกับการนำส่วนต่าง ๆ ของซอฟต์แวร์กลับมาใช้ใหม่

- กระบวนการ (procedure), ทักษะ(skill) : เกี่ยวข้องกับการนำขั้นตอนหรือกระบวนการที่ถูกห่อหุ้ม (encapsulate procedure process) กลับมาใช้ใหม่ซึ่งสามารถนำมาใช้ในการพัฒนาโปรแกรมที่สามารถนำกลับมาใช้ใหม่ที่ตรงกับความต้องการเฉพาะรายได้และพัฒนาความรู้ที่ถูกห่อหุ้ม (encapsulate knowledge)

2. โดยขอบเขต (scope)

- แนวตั้ง (vertical) : เกี่ยวข้องกับการนำกลับมาใช้ใหม่ภายในโดเมนเดียวกันหรือแอพพลิเคชั่นเดียวกัน

- แนวนอน (horizontal) : เกี่ยวข้องกับการนำกลับมาใช้ใหม่ของชิ้นส่วนในแอพพลิเคชั่นที่ต่างกัน

3. โดยวิธีการ (mode)

- แบบมีแผนงาน (planning), ความเป็นระบบ (systematic) : เกี่ยวข้องกับกระบวนการที่เป็นการวางแผนการทำงานทีละขั้นตอนในโครงการการนำชิ้นส่วนซอฟต์แวร์กลับมาใช้ใหม่

- แบบไม่มีแบบแผน (ad hoc) : เกี่ยวข้องกับงานที่ไม่ทราบล่วงหน้าหรือการนำกลับมาใช้ใหม่แบบไม่มีรูปแบบที่แน่นอน โดยปกติจะทำในระดับบุคคล ไม่ใช่ในระดับโครงการ

4. โดยเทคนิค (technique)

- การประกอบกัน (compositional) : เกี่ยวข้องกับการใช้ชิ้นส่วนที่มีอยู่แล้วในการสร้างระบบใหม่ในลักษณะส่วนการสร้างงาน (building block)

- การสร้าง (generative) : เกี่ยวข้องกับการนำกลับมาใช้ใหม่ในระดับของข้อกำหนดคุณลักษณะในแง่ของงานประยุกต์หรือการสร้างรหัสโปรแกรมผ่านตัวสร้างรหัส (code generator)

5. โดยความตั้งใจ (intention)

- black-box : เกี่ยวข้องกับการนำชิ้นส่วนซอฟต์แวร์กลับมาใช้ใหม่โดยไม่ต้องมีการปรับปรุงเปลี่ยนแปลงชิ้นส่วนนั้น

- white-box : เกี่ยวข้องกับการนำชิ้นส่วนซอฟต์แวร์กลับมาใช้ใหม่โดยการปรับปรุงหรือดัดแปลงชิ้นส่วนนั้น

6. โดยผลิตภัณฑ์ (product)

- รหัสโปรแกรมต้นฉบับ (source code)
- ผลิตผลการออกแบบ (design)
- ข้อกำหนดคุณลักษณะ (specifications)
- วัตถุ (objects)
- ข้อความ (text)
- สถาปัตยกรรม (architecture)

นอกจากมุมมองเหล่านี้แล้ว ความสามารถในการนำกลับมาใช้ใหม่ที่เสนอโดย Jones สามารถแบ่งได้ออกเป็น 5 กลุ่ม [Jone, 1984] ดังนี้

1. ข้อมูลที่นำกลับมาใช้ใหม่ได้ (Reusable data)
2. สถาปัตยกรรมที่นำกลับมาใช้ใหม่ได้ (Reusable architecture)
3. การออกแบบที่นำกลับมาใช้ใหม่ได้ (Reusable design)
4. โปรแกรมและระบบโดยทั่วไปที่นำกลับมาใช้ใหม่ได้ (Reusable programs and common systems)
5. กลุ่มการทำงานที่นำกลับมาใช้ใหม่ได้ (Reusable modules)

ข. ระดับการนำกลับมาใช้ใหม่ (Degree of Reuse)

ระดับการนำกลับมาใช้ใหม่ของชิ้นส่วน สามารถแบ่งได้เป็น การนำกลับมาใช้ใหม่โดยตรงจากการสร้างเป็นกระบวนการ การนำกลับมาใช้ใหม่หลังจากการกลั่นกรองแล้ว และการนำกลับมาใช้ใหม่หลังจากมีการปรับปรุงเปลี่ยนแปลงแล้ว ซึ่งทั้ง 3 ระดับนี้เสนอขึ้นโดย Standish ใน [Pohthong, 2000] นอกจากนี้ Lewis และคณะได้เสนอระดับการนำกลับมาใช้ใหม่ ใน [Pohthong, 2000] ดังนี้

- การนำกลับมาใช้ใหม่ได้โดยสมบูรณ์ (completely reusable)
- การนำกลับมาใช้ใหม่ได้โดยการเปลี่ยนแปลงเพียงเล็กน้อย (Reusable with slight revision) ปรับปรุงน้อยกว่า 25%
- การนำกลับมาใช้ใหม่ได้โดยการเปลี่ยนแปลงค่อนข้างมาก (Reusable with major revision) ปรับปรุงมากกว่า 25%

- ไม่สามารถนำกลับมาใช้ใหม่ได้ (Not applicable for reuse)

ดังนั้นการนำชิ้นส่วนกลับมาใช้ใหม่ที่ก่อให้เกิดประสิทธิภาพไม่ควรที่จะมีการเปลี่ยนแปลงรายละเอียดในชิ้นส่วนนั้นมากจนคล้ายกับการสร้างชิ้นส่วนใหม่

3.1.3.2 **ห้องสมุดชิ้นส่วน** (Repository)

หลังจากมีการสร้างชิ้นส่วนที่สามารถนำกลับมาใช้ใหม่ได้และนำมาผ่านกระบวนการควบคุมคุณภาพแล้ว จากนั้นก็สามารถส่งชิ้นส่วนไปเก็บไว้ในคลังชิ้นส่วนสำหรับผู้ใช้ ในขั้นตอนนี้ชิ้นส่วนทั้งหมดจะถูกนำมาเข้าสู่กระบวนการของห้องสมุด เช่น การลงทะเบียน การจัดทำหมวดหมู่และดรรชนีและการสร้างส่วนการแนะนำชิ้นส่วนสำหรับผู้ใช้ ซึ่งมีความคล้ายคลึงกับงานของห้องสมุดที่เป็นที่รวบรวมหนังสือและสื่อการเรียนรู้ต่าง ๆ ซึ่งประสิทธิของห้องสมุดชิ้นส่วนนั้นขึ้นอยู่กับการจัดโครงสร้าง ตัวอย่างเช่น ในห้องสมุดนั้นอาจมีข้อมูลจำนวนมากมาย แต่ไม่มีการจัดการในส่วนของการอธิบายสิ่งที่มีอยู่ให้แก่ผู้ใช้เพราะการจัดการในเรื่องนี้อาจนำมาซึ่งค่าใช้จ่ายที่สูงขึ้น ทำให้เกิดปัญหาในการค้นหาสิ่งที่ผู้ใช้ต้องการ โดยทั่วไปกิจกรรมต่าง ๆ ของการจัดการห้องสมุดที่เสนอโดย Gates ใน [Pohthong, 2000] แบ่งได้เป็น 4 กลุ่มดังนี้

1 ระบบการนำหนังสือเข้าสู่ระบบ (the acquisitions system)

2 ระบบการหมุนเวียนหนังสือ หรือการยืม-คืน (the circulation system)

3 ระบบการจัดหมวดหมู่ (the cataloguing system)

4 ระบบการทำอ้างอิง (the reference system)

ซึ่ง Chapman ใน [Pohthong, 2000] ได้เสนอให้เพิ่มกิจกรรมอีก 2 กลุ่ม ได้แก่

1. ระบบการควบคุมแบบอนุกรม (the serials control system)
2. ระบบการบริหารห้องสมุดและระบบการวางแผนงาน (the library administration and planning system)

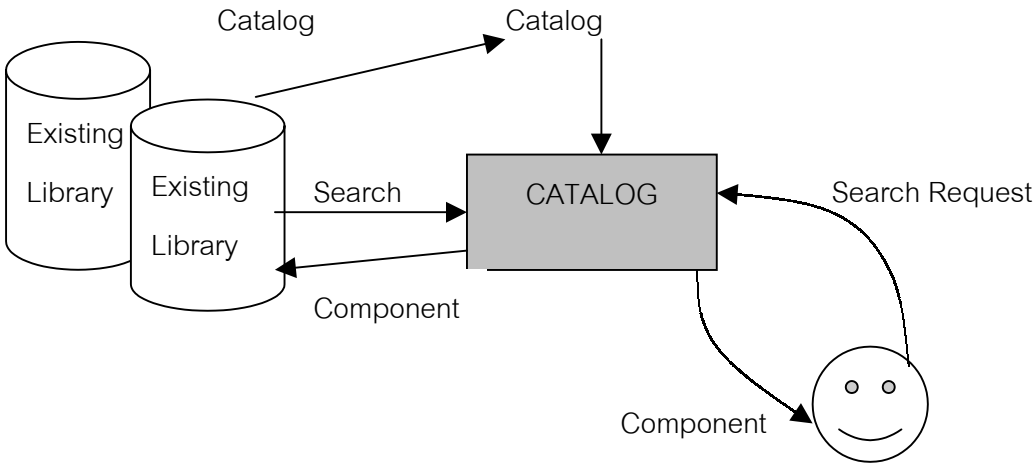
ความสำเร็จในแนวคิดของการจัดการห้องสมุดของหนังสือสร้างต้นแบบให้กับแนวคิดของการทำห้องสมุดชิ้นส่วน ซึ่งบทบาทของห้องสมุดชิ้นส่วนที่สามารถนำกลับมาใช้ใหม่ได้สามารถมองได้ 2 ด้าน คือ ด้านของผู้ใช้ และ ด้านของผู้ให้บริการ ในบางกิจกรรมของทั้ง 2 ด้านสามารถทำด้วยคนหรือทำแบบอัตโนมัติได้ถ้ามีเครื่องมือสนับสนุน ซึ่งหากพิจารณาจากมุมมองของผู้ใช้และผู้ให้บริการ ในแต่ละด้านอาจมีกิจกรรมดังต่อไปนี้

1. มุมมองของผู้ใช้

เมื่อผู้ใช้ที่เป็นนักพัฒนาต้องการค้นหาชิ้นส่วนที่มีอยู่แล้วมาใช้ในการ

พัฒนาซอฟต์แวร์ ผู้ใช้ต้องตอบจากคำถามเหล่านี้ให้ได้

- ต้องการชิ้นส่วนที่สามารถนำกลับมาใช้ใหม่ได้ชนิดใด
- สามารถหาชิ้นส่วนเหล่านั้นได้จากที่ไหน
- สามารถค้นพบชิ้นส่วนจะเข้าใจการทำงานของชิ้นส่วนได้ด้วยวิธีใด
- มีรายละเอียดใดบ้างที่จำเป็นต่อการนำชิ้นส่วนนั้นกลับมาใช้ใหม่
- ถ้ามีชิ้นส่วนที่ตรงกับความต้องการมากกว่า 1 ชิ้น จะมีวิธีใดเลือกชิ้นส่วนที่เหมาะสมมากที่สุด



ภาพประกอบ 3.2 แสดง Reuse Catalog

[ที่มา : Http://www.reuseability.com/serv3.html]

คำตอบของคำถามเหล่านี้ไม่ได้หาได้ง่าย ๆ ประการแรกผู้ใช้ต้องสร้างความกระจ่างความต้องการเพื่อตอบคำถามข้อแรกให้ได้ การทำรูปแบบความต้องการของผู้ใช้สามารถช่วยในกรณีนี้ได้ ถ้ามีห้องสมุดของชิ้นส่วนมากกว่า 1 ที่ ผู้ใช้อาจการส่วนของการแนะนำห้องสมุดชิ้นส่วน หลังจากนั้นผู้ใช้สามารถค้นหาชิ้นส่วนได้ ในขั้นตอนนี้ ผู้ใช้ต้องตัดสินใจเลือกวิธีการค้นหาชิ้นส่วน ซึ่งขึ้นอยู่กับความสามารถของเครื่องมือที่สนับสนุน หลักจากค้นหาชิ้นส่วนที่ต้องการเจอแล้ว จะใช้งานชิ้นส่วนนั้นอย่างไรก็กลายเป็นอีกปัญหาหนึ่งที่สำคัญ ถ้าผู้ใช้ไม่สนใจถึงจุดนี้ อาจทำให้มีการใช้ชิ้นส่วนผิดและมีผลกระทบต่อระบบ ในกรณีนี้ผู้เชี่ยวชาญในการพัฒนาชิ้นส่วนจะต้องสามารถช่วยผู้ใช้หรือมีการแสดงแบบจำลองเครื่องมือที่สามารถใช้ทดสอบประสิทธิภาพของชิ้นส่วนเหล่านั้นให้กับผู้ใช้

2. มุมมองของผู้ให้บริการ

ในด้านของผู้ให้บริการ สามารถแบ่งกิจกรรมออกได้เป็นกลุ่มดังต่อไปนี้

2.1 จัดตั้งห้องสมุดชิ้นส่วน

เป็นการจัดการวางแผนการทั้งหมดของการจัดตั้งห้องสมุดชิ้นส่วน เช่น วัตถุประสงค์ ผู้ดูแล การสร้างอุปกรณ์ เครื่องมือ และส่วนอำนวยความสะดวกต่าง ๆ ที่เกี่ยวข้อง รวมทั้งการให้บริการและกลยุทธ์การวางแผนจัดการ

2.2 จัดโครงสร้างของห้องสมุดชิ้นส่วน

การจัดโครงสร้างอาจประกอบไปด้วยการทำด้วยคนหรือการทำแบบอัตโนมัติ ซึ่งการจัดโครงสร้างห้องสมุดชิ้นส่วนที่ดีจะช่วยอำนวยความสะดวกให้ผู้ใช้ใช้งานง่าย เช่น การนำรหัสต้นฉบับกลับมาใช้ใหม่ ถ้ามีการจัดโครงสร้างดีผู้พัฒนาหาชิ้นส่วนได้รวดเร็วกว่าที่จะมาเขียนเองใหม่ก็ทำให้พัฒนาระบบงานได้เร็วขึ้น

2.3 การจัดเก็บชิ้นส่วนที่นำกลับมาใช้ใหม่

การจัดเก็บชิ้นส่วนและจัดกลุ่มเป้าหมายขึ้นอยู่กับวัตถุประสงค์ของ และการให้บริการ ของห้องสมุด ซึ่งความต้องการนี้จะเป็นความสัมพันธ์ระหว่างผู้ขายหรือผู้พัฒนาชิ้นส่วน ผู้ใช้และผู้บริหารห้องสมุดชิ้นส่วน

2.4 การแบ่งหมวดหมู่ของชิ้นส่วนที่นำกลับมาใช้ใหม่

การแบ่งหมวดหมู่เป็นกระบวนการจัดกลุ่มให้กับชิ้นส่วนที่ขึ้นอยู่กับคุณสมบัติของชิ้นส่วน ซึ่งการจัดกลุ่มนี้เป็นพื้นฐานของการจัดโครงสร้างของชิ้นส่วนในห้องสมุด ดังนั้นผู้ใช้สามารถค้นหาชิ้นส่วนได้ง่ายและรวดเร็ว เช่นเดียวกับการแบ่งหมวดหมู่ของหนังสือในห้องสมุดที่มีประสิทธิภาพ

2.5 การสร้างแคตตาล็อกของชิ้นส่วนที่นำกลับมาใช้ใหม่

แคตตาล็อกคือรายการของชิ้นส่วนที่สามารถนำกลับมาใช้ใหม่ได้ ซึ่งจัดเตรียมในลำดับเชิงตรรกะ เช่น เรียงตามลำดับตัวอักษร หรือตามข้อมูลย่อๆ ของชิ้นส่วน เมื่อมีการแบ่งหมวดหมู่ชิ้นส่วนเป็นส่วนหลัก (major class) และส่วนย่อย (subdivisions) ผู้ใช้สามารถดูรายการในแคตตาล็อกหลัก โดยอาศัยข้อมูลย่อ ๆ ของชิ้นส่วนเช่น ชื่อชิ้นส่วน ชื่อและที่อยู่ของผู้พัฒนาหรือผู้ขาย ราคา และการใช้งาน วันที่สร้าง หรือข้อมูลสำคัญอื่น ๆ ซึ่งข้อมูลต่าง ๆเหล่านี้จะทำให้ผู้ใช้สามารถมองเห็นภาพรวมทั่วไปของชิ้นส่วนก่อนที่จะตัดสินใจศึกษาลงไปในรายละเอียด

2.6 การทำดรรชนีและการนำเสนอชิ้นส่วน

การทำดรรชนีเป็นการเชื่อมโยงแนวคิดของระบบการจัดหมวดหมู่และ

ระบบแคตตาล็อกเพื่อให้สามารถทำการค้นหาชิ้นส่วนได้อย่างมีประสิทธิภาพ โดยอาศัยความหมายและไวยากรณ์หรือการอธิบายของชิ้นส่วน อย่างไรก็ตามในซอฟต์แวร์ที่มีความซับซ้อนมาก ผู้ใช้อาจต้องการรายละเอียดมากกว่าที่ปรากฏในแคตตาล็อกหลัก บางครั้งจึงจำเป็นต้องสร้างการอธิบายชิ้นส่วนในรูปแบบของบทคัดย่อ (abstract) ซึ่งก็คล้ายคลึงกับบทคัดย่อของหนังสือนั่นเอง

2.7 การสืบค้นชิ้นส่วนเพื่อการนำกลับมาใช้ใหม่

การสืบค้นชิ้นส่วนเป็นกระบวนการของการดึงชิ้นส่วนและข้อมูลของมันจากห้องสมุดที่เก็บชิ้นส่วน การใช้เครื่องมือและกลไกการสืบค้นที่มีประสิทธิภาพจะช่วยให้สามารถค้นหาข้อมูลได้ง่ายและตรงตามวัตถุประสงค์ที่ต้องการ

3. การจัดการและบำรุงรักษาห้องสมุดชิ้นส่วน

เมื่อมีการสร้างห้องสมุดชิ้นส่วนขึ้นมา การจัดการและการบำรุงรักษาก็เป็นอีกเรื่องนึงที่ผู้พัฒนาควรให้ความสำคัญ ซึ่งมีการเอาการจัดการส่วนตั้งค่า (configuration management) เช่น การทำ version control มาใช้ การจัดการและบำรุงรักษานี้ควรจัดทำเป็นนโยบายขององค์กรเพื่อความมีประสิทธิผลในการใช้งาน การประเมินและการวัดประสิทธิภาพของห้องสมุดชิ้นส่วน

3.1.3.3. **กระบวนการและวิธีการ** (Process and Methods)

กระบวนการและวิธีการเป็นปัจจัยสนับสนุนที่สำคัญอีกปัจจัยหนึ่งของแนวทางการพัฒนาระบบงานเชิงชิ้นส่วน ซึ่งถูกเสนอไว้หลายแนวทาง เช่น แนวทาง Draco ซึ่งเป็นวิธีการสำหรับการวิศวรรมซอฟต์แวร์ในรูปแบบการนำชิ้นส่วนกลับมาใช้ใหม่ ซึ่งเสนอโดย Neighbors ใน [Pohthong, 2000] ซึ่งแนวทางนี้จะให้ความสำคัญในเรื่องของการวิเคราะห์ขอบเขต (domain analysis) และได้กล่าวถึงประโยชน์ของการวิเคราะห์ขอบเขตไว้ดังนี้

“the use of domain analysis to produce domain language which may be transformed for optimization purpose and implemented by software components, each of which certains multiple refinements which make implementation decisions by restating the problem in other domain languages”

ซึ่งการใช้แนวทางนี้ในการพัฒนาระบบงานซอฟต์แวร์มีขั้นตอนดังต่อไปนี้

1. กำหนด (domain) ที่จะทำการวิเคราะห์
2. ทำการวิเคราะห์ขอบเขต

3. สร้าง domain language และห้องสมุดของแนวคิด (library of concepts)
4. สร้างตัวแปลภาษา (parser) ใน domain language
5. ระบุการเปลี่ยนแปลงรูปแบบซึ่งสร้างภาษาของผลลัพธ์ (output language)
6. ออกแบบข้อกำหนดของระบบคอมพิวเตอร์โดยใช้ domain language แล้วนำมาผ่าน Draco ซึ่งจะเป็นตัวแปลให้อยู่ในรูปที่พร้อมจะทำงาน

อย่างไรก็ตามเราสามารถแบ่งกลุ่มของกระบวนการสำหรับการวิศวกรรมซอฟต์แวร์ที่ใช้แนวคิดของการนำกลับมาใช้ใหม่ได้ ดังนี้

1. Domain modeling process
   เป็นกระบวนการในการมองแบบจำลองขอบเขตของโมเดล
2. Component selection process
   เป็นกระบวนการเลือกชิ้นส่วนซอฟต์แวร์ที่จะนำมาใช้งาน
3. Component engineering process
   เป็นกระบวนการสร้างชิ้นส่วนใหม่
4. Application engineering process
   เป็นการบวนการในการสร้างงานประยุกต์จากชิ้นส่วนใหม่หรือนำชิ้นส่วนเดิมมาประกอบกัน

3.3.1.4 **เครื่องมือช่วยในการประกอบชิ้นส่วน** (Integration Framework)

เครื่องมือช่วยในการประกอบชิ้นส่วนถูกใช้ในการรวมชุดของชิ้นส่วนที่ถูกเลือกเข้ามายังระบบซอฟต์แวร์ [Krueger ใน [Pohthong, 2000]] ในการรวมชิ้นส่วนที่นำกลับมาใช้ใหม่เข้าไปในระบบซอฟต์แวร์ ผู้พัฒนาจะต้องทำความเข้าใจเกี่ยวกับหน้าที่และคุณสมบัติของชิ้นส่วนรวมถึงตัวอินเทอร์เฟซของชิ้นส่วนนั้นให้ชัดเจน และศึกษาถึงความสัมพันธ์ของชิ้นส่วนนี้กับชิ้นส่วนอื่น ๆ ในระบบซอฟต์แวร์ ซึ่งในการนำชิ้นส่วนต่าง ๆ มารวมกันอาจทำให้เกิดปัญหา ซึ่งสาเหตุ 6 ประการที่ทำให้ไม่สามารถรวมชิ้นส่วนเข้าในระบบซอฟต์แวร์ได้ มีดังต่อไปนี้

1. ระบบสิ่งแวดล้อมไม่เข้ากัน (Environment incompatibilities)
2. การใช้รูปแบบที่ไม่ถูกต้อง (Improper form)
3. การไม่เข้ากันของฮาร์ดแวร์ (Hardware incompatibilities)
4. มีความต้องการดัดแปลงชิ้นส่วนมากเกินไป (Too much modification required)

5. ข้อกำหนดคุณลักษณะที่ไม่ใช่หน้าที่การทำงาน (Nonfunctional specification)
6. มีการใช้ซอฟต์แวร์จากภายนอก (Linkage to extraneous software)

3.2 เทคโนโลยีที่ใช้ในการพัฒนาเชิงชิ้นส่วน

3.2.1 เทคโนโลยี XML(eXtensible Markup Language)

3.2.1.1 ความหมายและคุณสมบัติของ XML

XML เป็นมาตรฐานหนึ่งที่ได้รับการยอมรับให้ใช้เป็นมาตรฐานของการรับส่งเอกสารผ่านเครือข่ายอินเทอร์เน็ต โดยทำให้ผู้ใช้สามารถสร้างและดูแลเอกสารที่มีโครงสร้าง (structured documents) ที่บรรจุตัวอักษร (plain text) และทำให้สามารถปรับเปลี่ยนการแสดงผลในรูปแบบที่หลากหลายทำให้เกิดความน่าสนใจ แต่อย่างไรก็ตาม ข้อมูลที่สร้างขึ้นเป็น XML ไม่มีความสามารถ หรือคุณลักษณะพิเศษอะไร ที่ทำให้การแสดงผลออกมาในรูปแบบที่น่าสนใจ เช่น ตัวหนา ตัวเอียง ตัวขีดเส้นใต้ เพียงแต่ว่า XML ทำหน้าที่แยกข้อมูลออกเป็นเนื้อหาเท่านั้น ตามลักษณะเอกสารที่มีโครงสร้าง ส่วนหน้าที่การนำเอาเนื้อหาใน XML มาแสดงในรูปแบบที่น่าสนใจ เป็นหน้าที่การทำงานของส่วนอื่น ดังนั้นจุดประสงค์หลักของ XML คือการแยกส่วนข้อมูลเพื่อประโยชน์ในการแสดงผล ซึ่งเป็นเทคโนโลยีที่เหมาะสมกับการนำมาใช้ในการพัฒนาระบบเชิงชิ้นส่วน ในมุมมองของการสามารถนำส่วนข้อมูลหรือโครงสร้างที่ถูกแยกออกจากกันนี้กลับมาใช้ใหม่ได้ในระบบอื่น ๆ

ก. เอกสารแบบมีโครงสร้าง (Structured Document)

ลองพิจารณาจากตัวอย่างโครงสร้างของ หนังสือ ต่อไปนี้ดู

- หนังสือหนึ่งเล่มประกอบจากเนื้อหาแต่ละบท (chapter)
- ในแต่ละบทประกอบด้วยหัวข้อย่อย (section)
- เช่นเดียวกันในแต่ละหัวข้อย่อย อาจจะถูกอธิบายหรือมีตารางข้อมูล (table) บรรจุอยู่
- ตารางข้อมูลถูกสร้างขึ้นมาจากแถว (row) และคอลัมน์ (column) ดังนั้น จะเห็นว่าหนังสือแต่ละเล่ม มักจะมีรูปแบบ หรือโครงสร้างที่แน่นอน ซึ่งทุกคนที่หยิบหนังสือขึ้นมาอ่านก็จะเข้าใจโครงสร้างของหนังสือ ตามที่ได้อธิบายไว้ข้างต้น

ข. ความหมายของ "plain text"

ตัวอักษร(character) ที่เห็นในจดหมายหรือบนกระดาษทั่วไป เป็นลักษณะตัวอักษรประเภทที่พิมพ์ได้ เช่นที่เราใช้พิมพ์ในเอกสารโปรแกรมพิมพ์รายงานทั่วไป ซึ่งรวมถึงตัวเลขด้วย ถือว่าเป็นตัวหนังสือ

ค. ข้อมูลดิบของเอกสาร XML (Raw-XML)

เอกสาร XML ที่ถือเป็นข้อมูลดิบ คือข้อความที่เกิดจากการรวมกันของตัวอักษร เพื่อทำให้เกิดเป็นเอกสาร ก่อนที่จะนำเอกสารที่เกิดจากข้อมูลดิบมาใช้แสดงผลลัพธ์ตามวัตถุประสงค์ของการใช้งานอีกที ซึ่งเป็นรูปแบบที่น่าสนใจมากกว่าตัวหนังสือ

ง. การปรับเปลี่ยนการแสดงผล (Rendering)

การปรับเปลี่ยนการแสดงผลเป็นศัพท์สำหรับคอมพิวเตอร์สมัยใหม่ ซึ่งหมายถึง รูปแบบของการนำข้อมูลดิบมาปรับเปลี่ยนใหม่ ให้ดูน่าสนใจยิ่งขึ้น สำหรับการมองของมนุษย์ ตัวอย่างของการวาดภาพในระบบคอมพิวเตอร์ เป็นการนำเอาสมการ ทางคณิตศาสตร์ มาใช้ร่วมกับตัวเลขต่างๆ เช่น สมการวาดวงกลม กับค่าตัวเลขหนึ่งให้วงกลมขนาดหนึ่ง กับอีกตัวเลข ให้วงกลมในอีกหนึ่งขนาด ซึ่งตัวเลขและสมการเหล่านี้ เป็นสิ่งที่มนุษย์ หรือผู้ใช้งานไม่อาจได้เห็นเลย แต่จะเห็นเป็นรูปร่างที่เกิดขึ้นแทน ส่วนกระบวนการปรับเปลี่ยนการแสดงผลของเอกสาร มีความหมายว่าเป็นการแสดงข้อมูลในรูปแบบเอกสาร ที่มนุษย์เข้าใจ เช่นหนังสือ หนังสือพิมพ์ หรือเอกสารแบบอื่นๆ ที่มนุษย์เข้าใจ และอ่านได้

จ. การแยกการแสดงผลด้วยเนื้อหา

ข้อมูลที่สร้างขึ้นเป็น XML ไม่มีความสามารถ หรือคุณลักษณะพิเศษอะไร ที่ทำให้การแสดงผลออกมาในรูปแบบที่น่าสนใจ เช่น ตัวหนา ตัวเอียง ตัวขีดเส้นใต้ ดังที่กล่าวมาแล้ว XML ทำหน้าที่แยกข้อมูลออกเป็นเนื้อหาเท่านั้น ตามลักษณะเอกสารที่มีโครงสร้าง ส่วนหน้าที่การนำเอาเนื้อหาใน XML มาแสดงในรูปแบบที่น่าสนใจ เป็นหน้าที่การทำงานของส่วนอื่น ถ้าสังเกตจากหนังสือพิมพ์ เป็นรูปแบบหนึ่งที่นักข่าวได้พยายามหาข้อมูลที่เป็นข่าวมาให้กับ ฝ่ายจัดพิมพ์ ฝ่ายจัดพิมพ์ก็ทำหน้าที่นำข้อมูลเหล่านั้นมาแยกแยะเป็นเนื้อหาต่างๆ หลังจากนั้นก็นำข้อมูลที่แยกตามเนื้อหาไปจัดหน้าตาตามรูปแบบของหนังสือพิมพ์ เช่น หัวข้อพาดหัวข่าว ชื่อคอลัมน์ การใส่กรอบ เพื่อจัดพิมพ์ลงบนกระดาษหนังสือพิมพ์ เหล่านี้คือวิธีการ ปรับเปลี่ยนรูปแบบการแสดงผลของหนังสือพิมพ์แต่กระทำลงบนตัวกระดาษ สำหรับบนระบบคอมพิวเตอร์ การปรับเปลี่ยนการแสดงผลข้อมูล คือการแสดงลงบนหน้าจอคอมพิวเตอร์โดยตรง ซึ่งทำให้มีความหลากหลายในการแสดงผลมาก กว่าการปรับเปลี่ยนการแสดงผลลงบนกระดาษ เช่น ใส่สีสรรต่างๆ รวมถึงในบางครั้งอาจทำให้มีรูปภาพเคลื่อนไหว เกิดขึ้นมาด้วย แน่นอนไม่ว่ารูปแบบการปรับเปลี่ยนรูปแบบการแสดงผลจะเป็นอย่างไรต้องไม่มีผลกระทบกับเนื้อหา ดังนั้นกระบวนการปรับเปลี่ยนข้อมูลการแสดงผลมักจะทำในแบบออนไลน์ เช่น การเปลี่ยนขนาดตัวอักษร สำหรับผู้ใช้ที่มีจอภาพขนาดหนึ่ง ซึ่งไม่เหมือนกับรูปแบบและขนาดตัวอักษรของ ผู้ใช้ที่มีจอภาพอีกแบบหนึ่ง โดยที่ผู้อ่านทั้งสองยังคงได้รับข่าวสารที่เหมือนกัน ต่างกันเพียงรูปแบบข่าวสารในการนำเสนอเท่านั้น

ฉ. ตัวอย่างโครงสร้างเอกสารของหนังสือพิมพ์

หนังสือพิมพ์ เป็นเอกสารที่มีโครงสร้างอย่างหนึ่งด้วย โดยพิจารณาเอกสารที่เกิดจากส่วนประกอบของ หน้ากระดาษ และคอลัมน์ เป็นต้น เมื่อใดก็ตามเนื้อหาของหนังสือพิมพ์ ถูกรวบรวมไว้ในเอกสารที่มีโครงสร้าง ด้วย XML แล้วละก็ เราสามารถนำเอกสารเหล่านั้น ไปพิมพ์ลงบนกระดาษ ด้วยการปรับเปลี่ยนการแสดงผลสำหรับกระดาษเอกสาร และนำเสนอผ่านจอภาพคอมพิวเตอร์ ด้วยการปรับเปลี่ยนการแสดงผลสำหรับจอภาพ ผลลัพธ์จากทั้งสองกรณีจะให้ผู้รับข้อมูล ได้ข่าวสารที่เหมือนกัน (เนื้อหาจาก XML) แต่ต่างสภาพการรับข้อมูล สิ่งนี้นี้เอง ทำทำให้ XML มีข้อพิเศษและดูน่าสนใจ

3.2.1.2 **ประโยชน์ของ** XML

ประโยชน์ของ XML คือ การเป็นแม่แบบหรือต้นแบบในการนิยามข้อมูลเพื่อใช้งานต่าง ๆ โดยสามารถแจกแจงได้หลายประการ ดังนี้

ก. ใช้สำหรับสร้างข้อมูลที่สามารถอธิบายความหมายของตัวมันเองได้ (Self-Describe Data)

จากความสามารถในการสร้างแท็กขึ้นมาอธิบายข้อมูลที่อยู่ภายในแท็กทำให้ ข้อมูลนั้นมีความหมายอยู่ในตัวมันเอง เป็นข้อมูลที่สามารถเขียนโปรแกรมมาดึงข้อมูลไปใช้ได้โดยง่าย และแม้แต่มนุษย์ก็สามารถอ่านได้ด้วย ดังนั้นจึงสามารถกล่าวได้ว่า เอกสาร XML มีคุณลักษณะครบทั้งแบบอ่านเข้าใจโดยมนุษย์ และแบบอ่านเข้าใจโดยเครื่องคอมพิวเตอร์

ข. ใช้สำหรับการแลกเปลี่ยนข้อมูล (Data Exchange)

ด้วยความที่เป็นแฟ้มข้อมูลข้อความธรรมดานี่เอง ทำให้เอกสาร XML เป็นภาษากลางที่ใช้ได้ทุกแพลตฟอร์มไม่ว่าจะเป็นแพลตฟอร์มตระกูล Windows หรือตระกูล Unix หรืออื่น ๆ จึงสามารถแลกเปลี่ยนข้อมูลที่อยู่ในรูปเอกสาร XML ข้ามแพลตฟอร์มได้

ถ้ามีการแลกเปลี่ยนข้อมูลระหว่างองค์กร สมมติว่าข้อมูลนั้นต้องดึงมาจากฐานข้อมูล องค์กรทั้งสองอาจตกลงกันว่าจะใช้เครื่องหมายพิเศษเป็นตัวคั่น (delimiter) ระหว่างฟิลด์ในฐานข้อมูล แล้วส่งข้อมูลเป็นข้อความ (text stream) เครื่องหมายพิเศษที่นิยมใช้เป็นตัวคั่นอย่างเช่น จุลภาค (,) หรือช่องว่าง ข้อมูลประเภทนี้เรียกว่า CVS (comma separate value) ซึ่งปัญหาจะเกิดขึ้นในกรณีเครื่องหมายที่เป็นตัวคั่นนั้น เป็นตัวข้อมูลจริง ๆ อยู่ในบางฟิลด์ด้วย นอกจากนี้ ถึงแม้จะสามารถเขียนโปรแกรมมาอ่านข้อมูลลักษณะนี้ได้ แต่ก็เป็นข้อมูลที่มนุษย์อ่านไม่รู้เรื่อง เพราะไม่มีความหมายในตัวเอง แต่ด้วยคุณสมบัติของ XML จะทำให้ปัญหาข้างต้นหมด

ไป เพราะเราจะใช้แท็ก (tag) ที่กำหนดขึ้นมาและตกลงกันไว้ล่วงหน้าระหว่างองค์กร ในการนิยามข้อมูล

ค. เป็นรูปแบบข้อความในการสื่อสารระหว่างโปรแกรมประยุกต์

มีความนิยมกันมากในแวดวงอินเทอร์เน็ตช่วงหลังปี 2000 เป็นต้นมา คือ แนวคิดของการให้บริการผ่านเว็บ (Web service) ที่เรียกว่า .NET แนวความคิดนี้เป็นการเตรียมซอฟต์แวร์สำหรับให้บริการทำงานบางอย่างที่อยู่ในเซิร์ฟเวอร์เครื่องใดเครื่องหนึ่งภายในเครือข่ายอินเทอร์เน็ต โดยผู้ใช้งานทั่วไปสามารถเรียกใช้บริการนั้นได้

นอกจาก XML แล้ว ยังมีมาตรฐานต่าง ๆ ที่เป็นส่วนสำคัญในการทำงานของ Web Service เช่น SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration) ซึ่งทั้ง SOAP และ UDDI ล้วนมีพื้นฐานมาจาก XML เช่นกัน

ง. ประโยชน์ในเชิงเทคโนโลยีอินเทอร์เน็ตและการพัฒนาเว็บ

ในประเทศไทยยังไม่มีการนำ XML มาใช้อย่างแพร่หลาย เพราะลักษณะงานยังไม่มีความจำเป็นให้นำ XML ไปพัฒนาเท่าใดนัก แต่งานบางอย่างเช่นการพัฒนาเว็บ สามารถนำ XML มาช่วยเพิ่มประสิทธิภาพได้

จ. เป็นรากฐานของภาษาใหม่ ๆ ในการพัฒนาเว็บ

ภาษาใหม่ ๆ ในที่นี้ก็เช่น ภาษา XHTML, MathML, VML, WML และอื่นๆ อีกมากมาย

ฉ. ใช้ในแวดวงธุรกิจแบบ B2B (Business to Business)

กรณีนี้ต้องใช้ภาษาเฉพาะ อย่างเช่น cXML (Commerce XML), xCBL (XML Common Business Language) เป็นต้น โดยมีแท็กที่ใช้สนับสนุนการจัดการเกี่ยวกับแคตตาล็อกสินค้า (catalog) และการพาณิชย์อิเล็กทรอนิกส์ (e-Commerce)

3.2.1.3 ที่มาของเอกสาร XML

ในการใช้งานจริงนั้น เอกสาร XML มักจะเกิดจากการสร้างของโปรแกรมหรือโปรแกรมประยุกต์ เพื่อส่งไปให้โปรแกรมอื่น ๆ หรือ องค์กรอื่น ๆ เอกสาร XML อาจเกิดจากการดึงข้อมูลในฐานข้อมูล แล้วนำมาใส่รูปแบบการแสดงผลด้วยแท็ก ซึ่งกระบวนการนี้อาจใช้กลไกของโปรแกรมฐานข้อมูลบางโปรแกรมที่มีอยู่ หรือภาษาสคริปต์ทางฝั่งเซิร์ฟเวอร์มาช่วยก็ได้ อย่างเช่นโปรแกรม SQL Server 2000 สามารถสอบถาม (query) ข้อมูลในรูปของ XML ได้เลย

3.2.1.4 ตัวแปลภาษา XML

XML Parser หรืออีกชื่อหนึ่งคือ XML Processor ถือว่าเป็นสิ่งที่มีบทบาทสำคัญมาก เพราะเป็นตัวแปลเอกสาร XML หน้าที่พื้นฐานของ XML Parser ก็คือ อ่าน (read), สร้าง (create), แก้ไข (update) และ เข้าถึง (manipulate) เอกสาร XML ในปัจจุบันมีบริษัทซอฟต์แวร์หลายรายผลิตโปรแกรม XML Parser ออกมากันมากมาย โดยสร้างขึ้นจากภาษาต่าง ๆ ที่ได้รับความนิยมมากคือ C, C++, Java, Perl ยกตัวอย่างเช่น

- MSXML ของบริษัทไมโครซอฟต์ เป็นโปรแกรมที่มีมากับ IE 5.0 ขึ้นไป สร้างโดยเทคโนโลยี COM (Component Object Model)
- JAXP (Java API for XML Parsing) เป็นผลิตภัณฑ์ของบริษัทซัน สร้างโดยภาษา Java
- DataChannel XJParser ของเครือบริษัทดาต้าชาเนล สร้างด้วยภาษา Java
- XP เขียนด้วย Java เป็นผลงานของ Mr. James Clark
- XT เขียนด้วย C ของ Mr.James Clark
- expat เขียนด้วยภาษา C ของ Mr.James Clak
- Xerces-J เขียนด้วยภาษา Java ของ Apache XML Project
- Xerces-C++ เขียนด้วยภาษา C++ เป็นผลงานของ Apache XML Project

การจำแนกชนิดของ XML Parser สามารถแบ่งได้หลายวิธี แต่หลักเกณฑ์ที่นิยมมีเพียง 2 วิธี คือ

1. XML Parser แบบ validating และแบบ non-validating
2. XML Parser ที่รองรับ DOM (Document Object Model) และ SAX (Simple API for XML)

เอกสาร XML มีคุณสมบัติเป็นเอกสารแบบ well-formed XML คือ มีการใช้แท็กและระบุแท็กตามรูปแบบที่ถูกต้อง เช่น การเปิด-ปิดแท็ก ไม่มีการระบุแท็กเหลื่อมซ้อนกัน เป็นต้น ซึ่งเป็นคุณสมบัติพื้นฐานที่โปรแกรม XML Parser ทุกโปรแกรมจะต้องตรวจสอบอยู่แล้ว นอกจากนี้คุณสมบัติอีกอย่างหนึ่งของเอกสาร XML คือ valid หมายความว่า เอกสาร XML ต้องมีโครงสร้างตรงตามข้อกำหนดใน DTD หรือ XML Schema (แล้วแต่ว่าเอกสารนั้นกำหนดโครงสร้างด้วย DTD หรือ XML Schema)

สำหรับ XML Parser แบบ validating หมายถึง XML Parser ทีมีการตรวจสอบคุณสมบัติ well-formedness และคุณสมบัติ valid ของเอกสาร ส่วน XML Parser แบบ non-

validating หมายถึง XML Parser ที่ไม่ตรวจสอบคุณสมบัติ valid และสนใจเฉพาะคุณสมบัติ well-formedness

ผู้อ่านอาจคิดว่า XML Parser แบบ validating ดีกว่า ฉะนั้นถ้าเลือกได้ก็ไม่ควรใช้แบบ non-validating แต่แท้จริงแล้ว ในบางกรณีก็ควรใช้ XML Parser แบบ non-validating แทนที่จะใช้แบบ validating ด้วยเหตุผลดังนี้

- ทำงานเร็วกว่า เพราะไม่ต้องคอยตรวจสอบโครงสร้างของเอกสาร
- มีความมั่นใจว่า เอกสารนั้นอยู่ในรูปแบบที่ valid แน่นอน
- ต้องการแค่ค้นหาแท็กบางแท็กในเอกสารเท่านั้น ไม่ต้องตรวจสอบคุณสมบัติ valid

โปรแกรม XML Parser ส่วนใหญ่เป็นแบบ non-validating เพราะสร้างขึ้นมาได้ง่ายกว่านั่นเอง แต่ถ้าจะจำแนกประเภทของ XML Parser ตามวิธีการสำรวจเนื้อหาเอกสาร XML ก็แบ่งได้เป็น XML Parser แบบที่รองรับ DOM และ แบบที่รองรับ SAX

แบบที่รองรับ DOM จะมองเอกสาร XML เป็นโครงสร้างต้นไม้ ดังนั้นการเข้าถึงเอกสาร XML ก็คือการเดิน (traversal) ไปตามโหนดต่าง ๆ ในโครงสร้างต้นไม้ จึงเรียกได้ว่า XML Parser แบบที่รองรับ DOM เป็น Tree-Based Parser

ส่วนแบบที่รองรับ SAX จะใช้วิธีเชื่อมต่ออิลิเมนต์ในเอกสารเข้ากับเหตุการณ์ การสำรวจเนื้อหาเอกสารก็จะขึ้นกับเหตุการณ์ จึงกล่าวได้ว่า XML Parser แบบที่รองรับ SAX เป็น Event-Driven Parser

3.2.2 เทคโนโลยีจาวา

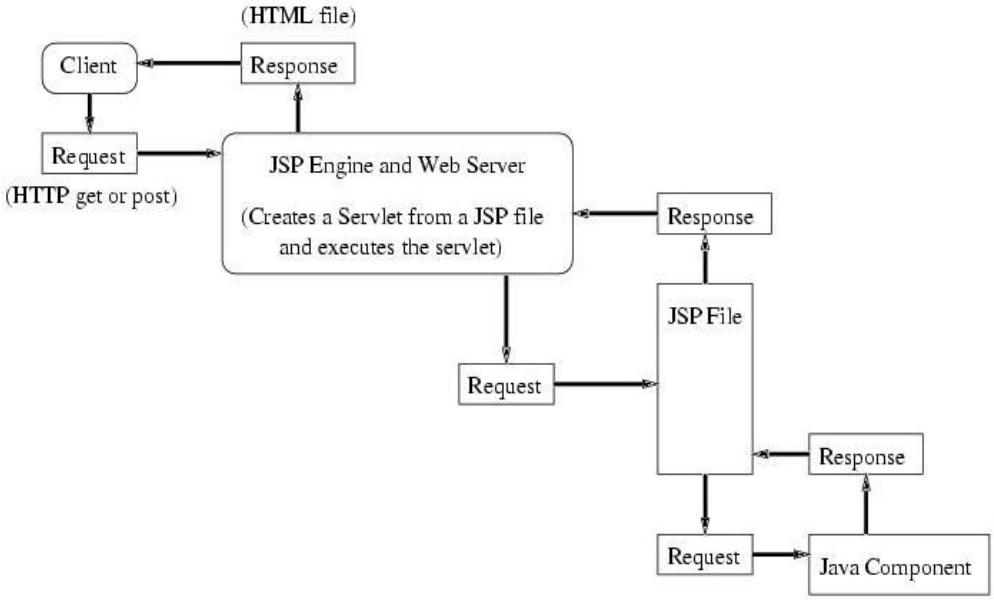
ภาษาจาวาเป็นภาษาโปรแกรมที่ใช้ในการพัฒนาระบบงานโปรแกรมโดยใช้แนวคิดเชิงวัตถุ (object orientation) ซึ่งสามารถลดความซับซ้อนโครงสร้างของโปรแกรมรวมไปถึงอำนวยความสะดวกในการนำกลับมาใช้ใหม่ของส่วนประกอบต่าง ๆ ที่เขียนไว้แล้ว โดยโปรแกรมเหล่านั้นสามารถใช้บนแพ็ตฟอร์มที่แตกต่างกันได้ (platform independent) ซึ่งทำให้นักพัฒนาสามารถพัฒนาระบบโดยใช้สิ่งแวดล้อมอะไรก็ได้ซึ่งโดยทั่วไปจะเป็น Windows โดยจะนำโปรแกรมที่เขียนเสร็จแล้วมาทำงานบน Unix เพื่อเพิ่มความเสถียรภาพของโปรแกรมอีกทีหนึ่ง และนอกจากนี้ภาษาจาวายังมีความเหมาะสมกับการสร้างงานแบบไดนามิกหรืองานโปรแกรมประยุกต์บนเครือข่ายอินเทอร์เน็ต เพราะมีการจัดเตรียมรูปแบบภาษาเชิงวัตถุที่สนับสนุนการทำงานของการพัฒนางานโปรแกรมเชิงชิ้นส่วนและใช้เวลาในการพัฒนางานน้อยกว่าการใช้ภาษา C++ โดยภาษาจาวาได้มีการจัดเตรียมในส่วนของ ต้นแบบเชิงชิ้นส่วน (component model) ได้แก่ JavaBeans ส่วนของสถาปัตยกรรมแบบกระจาย (distributed architecture) ได้แก่ Enterprise JavaBeans และ

CORBA ส่วนของกราฟฟิกไลบรารี ได้แก่ AWT / Swing และส่วนของรูปแบบของเป้าหมายที่แตกต่างกัน ได้แก่ โปรแกรมประยุกต์ applet และ servlet

ดังนั้นภาษาจาวาเป็นภาษาที่เหมาะสมเป็นอย่างยิ่งสำหรับการพัฒนาเชิงชิ้นส่วนตามวัตถุประสงค์ของโครงการวิจัยนี้

3.2.2.1 JSP (Java Server Page)

Java Server Page เป็นเทคโนโลยีเว็บสคริปคล้ายกับ Netscape server-side JavaScript (SSJS) หรือ Microsoft Active Server Pages (ASP) แต่ผิดกันตรงที่หลักสำคัญของ JSP คือ Java ซึ่งเป็นภาษาที่มีแนวคิดหลักแนวคิดเชิงวัตถุ ซึ่งช่วยทำให้ง่ายต่อการพัฒนาในงานใหญ่ ๆ ตลอดจนสามารถนำส่วนประกอบต่างๆ กลับมาใช้ได้อีก ซึ่งเทคโนโลยีนี้เป็นผลพวงมาจากเทคโนโลยีของ Java Servlet ซึ่งเป็นหลักการสร้างระบบการประมวลผลเว็บที่ทำงานโดยอาศัยทรัพยากรของเว็บเซิร์ฟเวอร์ ก่อนจะส่งผลลัพธ์สู่ผู้ใช้ผ่านระบบเน็ตเวิร์คหรืออินเตอร์เน็ตโดยผู้สร้าง JSP สามารถสร้างโดยการแทรกคำสั่ง JSP ไว้ในชุดคำสั่ง HTML ที่ใช้สร้าง เว็บและบันทึกไว้ในไฟล์นามสกุล JSP จากนั้นเก็บไว้ที่เซิร์ฟเวอร์ที่สนับสนุนการทำงานของ JSP และจุดเด่นที่สำคัญของ JSP อีกประการหนึ่งคือ สามารถทำงานได้โดยไม่ขึ้นอยู่กับบริษัทผู้ผลิตซอฟต์แวร์รายใดรายหนึ่งโดยเฉพาะ ซึ่งโดยทั่วไปเทคโนโลยีต่าง ๆ มักจะออกมาในลักษณะของผลิตภัณฑ์แห่งใดแห่งหนึ่ง แต่ JSP ใช้ลักษณะของรูปแบบข้อกำหนดซึ่งกำหนดโดย Sun Microsystems ดังนั้นผู้ผลิตซอฟต์แวร์จึงสามารถอ้างอิงรูปแบบความต้องการที่กำหนดขึ้น ผลิต JSP Container
(ตัวที่ใช้ในการรัน JSP) ขึ้นมาใช้กับแฟลตฟอร์มใดก็ได้



ภาพประกอบ 3.3 แสดง การทำงานของ JSP Engine

ที่มา : [Hanna, 2001]

ข้อแตกต่างของ JSP เมื่อเทียบกับเทคโนโลยีอื่น ๆ

JSP ไม่ได้เป็นเทคโนโลยีเดียว ในปัจจุบันที่สามารถทำให้เว็บ แสดงข้อมูลแบบไดนามิกได้ มีเทคโนโลยีอื่น ๆ ที่ทำงานในลักษณะนี้หลายแบบ เนื้อหาส่วนนี้ เป็นการแนะนำให้ผู้อ่านได้เห็นข้อแตกต่างบางประการ เมื่อเทียบกับ JSP

- **Active Server Page(ASP)** เป็นเทคโนโลยีที่เหมือนกับ JSP แต่เป็นเทคโนโลยีที่ได้รับการพัฒนาโดยบริษัทไมโครซอฟต์ โดยทั่วไปลักษณะ JSP มีรูปแบบที่แตกต่างกันเด่นชัด 2 ประการ คือ ประการแรก JSP สามารถสร้างได้จาก ภาษาจาวาต่างจากภาษา VB(Visual Basic) หรือภาษาใดๆ ซึ่งเป็นข้อกำหนดเฉพาะของไมโครซอฟต์ ประการที่สอง JSP ซึ่งเป็นผลพวงมาจากประการแรกคือ สามารถโยกย้ายการทำงาน ไปใช้งานระบบปฏิบัติการใดๆก็ได้ ทีมีการใช้งานกันอยู่ในปัจจุบัน โดยไม่ต้องแก้ไขหรือเปลี่ยนแปลงใดๆ ในขณะที่ ASP ก็โยกย้ายได้เช่นเดียวกัน แต่ต้องอยู่ในระบบปฏิบัติการที่บริษัทไมโครซอฟต์กำหนดขึ้นเท่านั้น

- **Servlet JSP** สามารถทำงานได้เช่นเดียวกับที่ servlet ทำได้ แต่มีจุดเด่นมากกว่าที่ JSP มีความสะดวกในการสร้าง และเปลี่ยนแปลง มากกว่า เพราะทำโดยตรงที่ไฟล์ HTML มากกว่าการที่ต้องลงมือเขียนคำสั่งภาษาจาวา โดยตรงเหมือนการสร้าง servlet ซึ่งต้องมีการนำไปคอมไพล์ ก่อนการนำไปใช้งาน

- **Server-Side Includes(SSI)** คือรูปแบบการสร้างไดนามิกเว็บแต่รูปแบบการทำงานเป็นการนำเอาข้อมูลที่มีอยู่แล้วบนเซิร์ฟเวอร์นำมาประกอบใส่ ในเว็บเท่านั้น ต่างจาก JSP ซึ่งสามารถนำข้อมูลมาใส่ แต่มีรูปแบบในการทำงาน มีรูปแบบในการประมวลผล หรือการเรียกโปรแกรมภายนอก เช่น servlet ที่ช่วยในการทำให้ข้อมูลมีรูปแบบต่างๆ เช่น ดึงข้อมูลมาจากระบบฐานข้อมูลนอกจากข้อมูลที่มีอยู่บนเซิร์ฟเวอร์

- **JavaScript** เป็นการทำเนื้อหาไดนามิกให้กับเว็บ แต่รูปแบบการทำงานเกิดขึ้นของ JavaScript เกิดจากการประมวลผลและดึงข้อมูล ที่มีอยู่บนเครื่องลูกข่ายเท่านั้น มารวบรวมเนื้อหาในเว็บ ต่างกับ JSP ที่เป็นการทำไดนามิก แต่ข้อมูลถูกสร้าง และดึงมาจากระบบเซิร์ฟเวอร์ ที่มีหลากหลายมากกว่า และทำให้เนื้อข้อมูลเป็นเนื้อเดียวกัน เมื่อผู้ใช้เรียกดู แต่ JavaScript เนื้อข้อมูลเป็นของเครื่องผู้ใช้เอง ซึ่งผู้ใช้ต่างคน(ต่างเครื่อง) ก็จะให้ข้อมูลที่คล้ายกัน คือไม่เหมือนกันทั้งหมด

- **Static HTML** ให้เนื้อข้อมูลที่คงที่ตลอดเวลา ไม่ว่าจะถูกเรียกดูในเวลาใดๆ (ยกเว้นมีคนมาแก้ไขคำสั่ง HTML) ในขณะที่ JSP ทำให้เนื้อข้อมูลมีการเปลี่ยนแปลงตามการใช้งาน หรือเครื่องผู้ใช้โดยไม่ต้องเปลี่ยนแปลงแก้ไขคำสั่ง JSP

ตารางที่ 3.2 ตารางแสดงการเปรียบเทียบเทคโนโลยีเว็บแบบต่าง ๆ

	CGI/Perl	Mod_Perl	ASP	JSP
เว็บเซิร์ฟเวอร์	Any Web server	Apache Web Server	Microsoft IIS or Personal Web Server	Any Web server, including Apache, Netscape, IIS today
โยกย้ายการทำงาน	No	No	No	Yes
ลักษณะนำกลับมาใช้ (Reusable, modular code)	No	No	No	Yes
สคริปต์หรือภาษา	C, Perl	Perl	VBScript, JScript	Java
ปกป้องการใช้งานหน่วยความจำ	Yes	No	No	Yes
สนับสนุนการทำงานร่วมในหลายโปรเซส	No	Yes	Yes	Yes

ที่มา: ข้อมูลจากเว็บ Http://www.javasoft.com/products/jsp/jspguide-wp.html

3.2.2.2 JavaBeans

เทคโนโลยี JavaBeans เป็นโมเดลเชิงชิ้นส่วนสำหรับ Java platform ซึ่งระบุถึง Java software component และความเหมาะสมในการเลือกใช้กับงาน โดยคำจำกัดความของ JavaBeans component ที่กล่าวว่า

“A JavaBeans component is a reusable software component that can be visually manipulate in builder tools” [Johnson, 1997]

แสดงถึงการมองแต่ละ JavaBean เป็นชิ้นส่วนของซอฟต์แวร์ที่สามารถนำกลับมาใช้ใหม่ได้และสามารถทำการปรับปรุงเปลี่ยนแปลงได้ในเครื่องมือที่ใช้สร้าง ตัวอย่างชนิดของ JavaBeans ได้แก่ ปุ่ม, GUI control, Database viewers, stock data feeds, In-house customizable mini-applications, Word processors, spreadsheets และ Invisible server beans

ส่วนชนิดของเครื่องมือที่อาศัยชิ้นส่วนของ JavaBeans เพื่อทำเป็นเครื่องมือที่ใช้สร้างงานอื่น ๆ สามารถแบ่งเป็นชนิดต่าง ๆ ดังนี้

- GUI layout managers
- Web page builders
- Full-scale application environment
- Builders สำหรับ invisible server application

JavaBeans สามารถใช้งานร่วมกับ JSP ได้เป็นอย่างดี โดยใช้เป็นตัวห่อหุ้มส่วนตรรกะของโปรแกรมประยุกต์เพื่อส่งผ่านระหว่างเพจ เพราะในการเขียน JSP สำหรับระบบใหญ่ ๆ นักพัฒนาจะไม่นิยมใส่รหัสโปรแกรมภาษาจาวา ลงไปในแฟ้มข้อมูล JSP มากนัก สิ่งที่อยู่ในแฟ้มข้อมูลมักจะเป็นเพียงรหัส HTML และค่าของตัวแปรต่าง ๆ ที่ได้มาจาก JavaBeans เท่านั้นจะต่อการปรับเปลี่ยนลักษณะของมุมมองการแสดงผลของแฟ้ม JSP ที่เป็นเช่นนี้เพราะ ส่วนที่เป็นข้อมูลกับส่วนที่เป็นการแสดงผลจะอยู่ด้วยกันอย่างหลวม ๆ ดังนั้นเราจึงสามารถเปลี่ยนรูปแบบการแสดงผล เมื่อใดก็ได้ โดยตัว JavaBeans ที่ใช้เก็บค่าของตัวแปรต่าง ๆ ยังคงเหมือนเดิม

JavaBeans อาจถูกมองเป็นลักษณะของกล่อง โดยกล่องนี้มีส่วนที่ใช้ติดต่อกับโลกภายนอกอยู่สองส่วนหลัก ๆ คือ

1. ส่วนที่ใช้สำหรับรับข้อมูลต่าง ๆ โดยข้อมูลเหล่านี้จะถูกใช้เป็นตัวแปรในการควบคุมหรือเปลี่ยนแปลงคุณสมบัติและการทำงานของกล่อง (Setter)

2. ส่วนที่ใช้สำหรับอ่านคุณสมบัติของกล่อง (Getter) ซึ่งโดยส่วนมาก ก็คือค่าของตัวแปรต่าง ๆ ที่บรรจุอยู่ในกล่องในการใช้งานกล่อง (JavaBeans) ดังกล่าวร่วมกับ JSP เราสามารถที่จะสร้างกล่องขึ้นมาโดยใช้ <jsp:useBean …/> ส่งค่าเข้าไปในกล่องโดยใช้ <jsp:setProperty…/> และอ่านค่าต่าง ๆ ที่อยู่ในกล่อง โดยใช้ <jsp:getProperty…/> tag

จากคุณลักษณะของ JavaBeans ที่ถูกมองเป็นชิ้นส่วน ๆ หนึ่งของระบบซอฟต์แวร์ที่สามารถนำกลับมาใช้ใหม่ได้ ทำให้เราสามารถนำมาใช้กับการพัฒนาระบบโดยอาศัยแนวคิดเชิงชิ้นส่วนร่วมกับเทคโนโลยี JSP และ XML ได้เป็นอย่างดี