Software Reuse

Week 10

ดัดแปลงจาก slides ของหนังสือ Software Engineering [1]

หัวข้อที่จะศึกษา

- The reuse landscape
- Application frameworks
- Software product lines
- Application system reuse

Software reuse

- ระบบที่ได้รับการสร้างขึ้นมาใหม่นั้น มักจะทำได้โดยการนำส่วนประกอบที่ใช้ได้ดีอยู่แล้วในระบบอื่น ๆ มาประกอบเข้าด้วยกัน
  - อาจจะนำชิ้นส่วนมาใช้ตรง ๆ หรือนำ abstract ของชิ้นส่วนนั้นมาใช้งาน
- ในสมัยก่อนวิศวกรรมซอฟต์แวร์ ได้มุ่งเน้นการพัฒนาชิ้นส่วนต่าง ๆ ขึ้นเองทั้งหมด
- แต่ปัจจุบัน เป็นที่ยอมรับโดยทั่วกันว่า การนำซอฟต์แวร์มาใช้ซ้ำ จะทำให้ได้ซอฟต์แวร์ที่ดีขึ้น ในเวลาที่รวดเร็วและมีต้นทุนต่ำกว่า
- ในช่วง 10 ปีที่ผ่านมามีการพัฒนาแบบ software reuse เป็นหลัก

Reuse-based software engineering

- ใช้ซ้ำทั้งระบบ
  - Reuse ระบบที่สมบูรณ์ซึ่งอาจรวมถึงโปรแกรมประยุกต์หลายโปรแกรม
- ใช้ซ้ำระดับแอพพลิเคชัน
  - Reuse โดยการนำ application ที่ใช้งานได้แล้วมาผสมผสานเข้าด้วยกัน
- การนำชิ้นส่วนกลับมาใช้ใหม่
  - Reuse ส่วนประกอบของแอ็พพลิเคชันจากระบบย่อยหนึ่งไปใช้ในรูปแบบวัตถุในแอปพลิเคชันอื่น
- นำวัตถุและฟังก์ชันมาใช้ซ้ำ
  - Reuse คอมโพเนนต์ซอฟต์แวร์ขนาดเล็ก ในลักษณะ object หรือ function

ประโยชน์ของการใช้ซอฟต์แวร์ซ้ำ

ประโยชน์	เหตุผล
พัฒนาได้อย่างรวดเร็ว	การนำระบบออกสู่ตลาดให้เร็วที่สุดมักจะมีความสำคัญมากกว่าต้นทุนการพัฒนาโดยรวม การนำซอฟต์แวร์ไปใช้ใหม่สามารถเร่งผลิตระบบได้เนื่องจากลดกระบวนการในการพัฒนาและการตรวจสอบความถูกต้องลงได้เป็นอย่างมาก
ใช้ผู้เชี่ยวชาญอย่างมีประสิทธิภาพ	แทนที่จะพัฒนาชิ้นส่วนต่าง ๆ ซ้ำ เราสามารถใช้ชิ้นส่วนที่ประกอบไปด้วยความรู้และประสบการณ์ของผู้เชี่ยวชาญ
เพิ่มความเชื่อถือได้	ซอฟต์แวร์ที่ใช้ซ้ำ ได้รับการทดสอบและทดลองใช้แล้วในระบบอื่นที่หมือนหรือใกล้เคียงกัน การทำงานจะมีความน่าเชื่อถือได้มากกว่าซอฟต์แวร์ใหม่ ได้รับการแก้ไขข้อบกพร่องในการออกแบบและการใช้งานมาแล้ว
ลดต้นทุนการพัฒนา	ต้นทุนการพัฒนามีสัดส่วนกับขนาดของซอฟต์แวร์ที่พัฒนาขึ้น การนำซอฟต์แวร์กลับมาใช้ใหม่ หมายถึงการเขียนโค้ดที่น้อยลง

ประโยชน์ของการใช้ซอฟต์แวร์ซ้ำ

ประโยชน์	เหตุผล
ลดความเสี่ยงในกระบวนการ	ค่าใช้จ่ายของซอฟต์แวร์ที่พัฒนาเสร็จแล้วสามารถรู้ได้เป็นที่แน่นอน แต่ค่าใช้จ่ายขณะที่กำลังพัฒนามักเป็นเรื่องของการคาดการณ์ (การบริหารโครงการ) การนำชิ้นส่วนมาใช้ซ้ำจะช่วยลดข้อผิดพลาดในการประมาณต้นทุนโครงการ
การปฏิบัติตามมาตรฐาน	มาตรฐานบางอย่างเช่น user interface สามารถนำชิ้นส่วนกลับมาใช้ซ้ำได้ การใช้ user interface มาตรฐานช่วยเพิ่มความน่าเชื่อถือ เนื่องจากผู้ใช้มักจะทำผิดพลาดน้อยลงเมื่อใช้งานส่วนติดต่อที่คุ้นเคย

อุปสรรคของการใช้ซอฟต์แวร์ซ้ำ

ปัญหา	เหตุผล
การสร้าง การบำรุงรักษาและการใช้ไลบรารีของชิ้นส่วน	การสร้าง library ของชิ้นส่วนที่สามารถนำกลับมาใช้ใหม่ เพื่อให้นักพัฒนาซอฟต์แวร์รายอื่นนำไปใช้ได้นั้นอาจมีราคาแพง ต้องมีการพัฒนา process ในการพัฒนา library เพื่อให้แน่ใจว่ามันสามารถใช้จริงได้
การค้นหา การทำความเข้าใจและการปรับชิ้นส่วนที่สามารถนำกลับมาใช้ซ้ำได้	ในบางครั้ง ต้องใช้เวลานานเพื่อที่จะค้นหา library ที่ต้องการ ต้องทำความเข้าใจวิธีนำมาใช้ ต้องปรับให้เหมาะกับการทำงานในสภาพแวดล้อมใหม่ วิศวกรต้องมีความมั่นใจในเหตุผลในการค้นหาชิ้นส่วน library ก่อนที่จะรวมการค้นหาชิ้นส่วนนั้นเข้าเป็นส่วนหนึ่งของกระบวนการพัฒนาตามปกติ

อุปสรรคของการใช้ซอฟต์แวร์ซ้ำ

ปัญหา	เหตุผล
ค่าบำรุงรักษาเพิ่มขึ้น	ถ้าไม่มีการให้ซอร์สโค้ดของชิ้นส่วนที่ใช้ซ้ำได้ ค่าใช้จ่ายในการบำรุงรักษาอาจสูงขึ้น เนื่องจากชิ้นส่วนที่นำกลับมาใช้ใหม่ของระบบอาจไม่สามารถทำงานร่วมกับการเปลี่ยนแปลงระบบในอนาคต
ขาดเครื่องมือสนับสนุน	เครื่องมือผลิตซอฟต์แวร์บางอย่างไม่สนับสนุนการพัฒนาด้วยการนำมาใช้ใหม่ การรวมเครื่องมือเหล่านี้เข้ากับระบบไลบรารีคอมโพเนนต์อาจเป็นเรื่องยากหรือไม่สามารถทำได้
โรค “ฉันไม่ได้ทำเอง” กำเริบ	วิศวกรซอฟต์แวร์บางคนชอบที่จะเขียนคอมโพเนนต์ใหม่เพราะเชื่อว่าสามารถปรับปรุงได้เมื่อต้องการ อาจเกี่ยวกับความไว้วางใจใน source code ของคนอื่นน้อยเกินไป วิศวกรซอฟต์แวร์บางคนเชื่อว่าการเขียนซอฟต์แวร์ต้นฉบับถือเป็นความท้าทาย (และสนุก) มากกว่าการนำซอฟต์แวร์ของคนอื่นมาใช้ใหม่

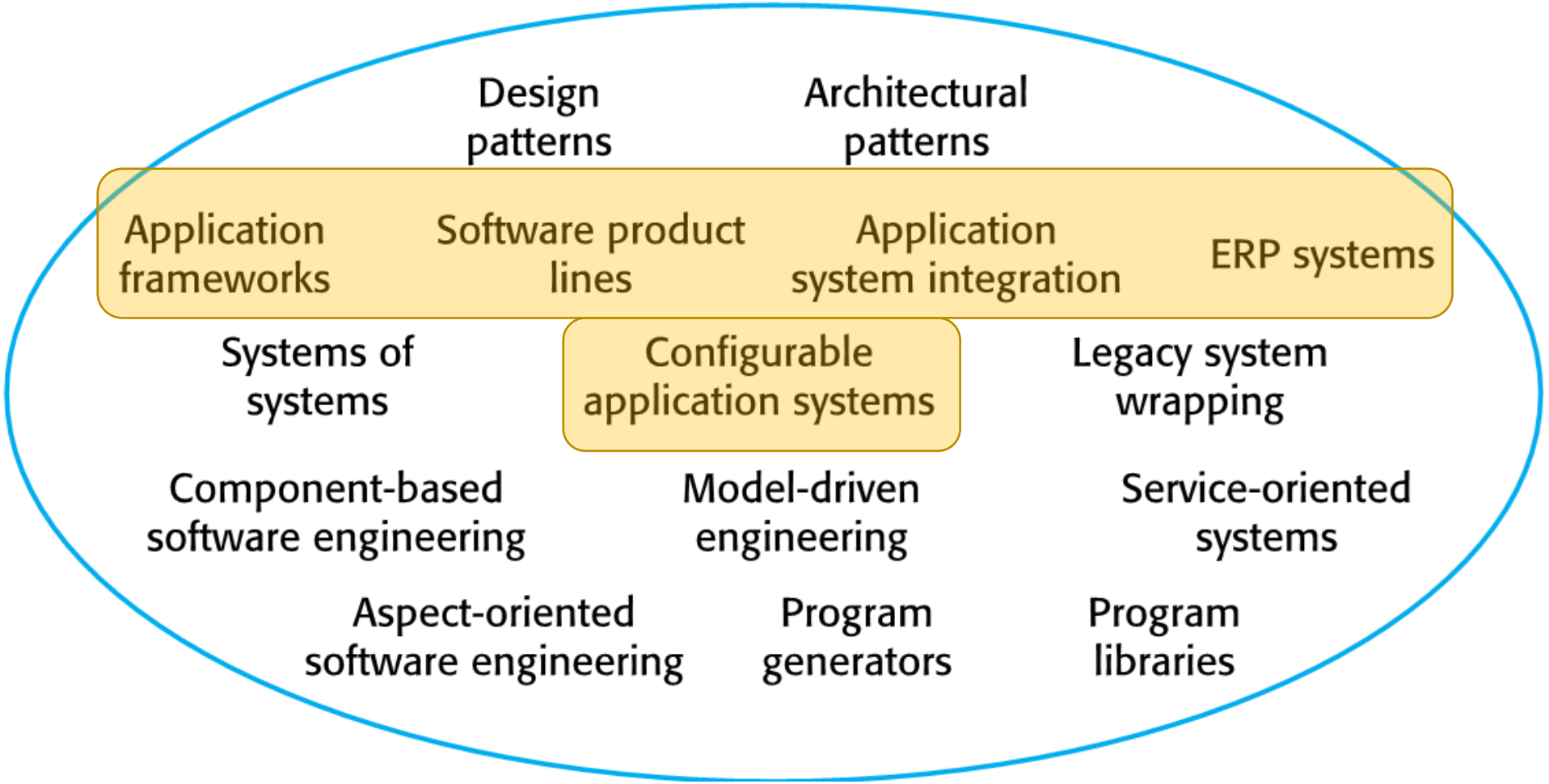
The reuse landscape

แนวทางในการนำมาใช้ใหม่

The reuse landscape

- แม้ว่าการ reuse มักจะหมายถึงการนำส่วนประกอบของระบบมาใช้ใหม่ แต่ก็มีการ reuse ได้หลายวิธีและหลายระดับ
- การ reuse เกิดขึ้นได้ในหลายระดับตั้งแต่ฟังก์ชันง่าย ๆ จนถึงระบบ application ที่สมบูรณ์
- reuse landscape ครอบคลุมทุกช่วงของเทคนิคการ reuse ที่เป็นไปได้

The reuse landscape



Approaches that support software reuse

Approach	Description
Application frameworks	มีการปรับใช้รวมทั้งเพิ่มความสามารถของ abstract และ concrete classes เพื่อสร้าง application ใหม่
Application system integration	มีการรวม application ตั้งแต่ 2 ตัวขึ้นไปเพื่อขยายขีดความสามารถในการทำงาน
Architectural patterns	พัฒนาซอฟต์แวร์โดยนำสถาปัตยกรรมซอฟต์แวร์มาตรฐานมาใช้เป็นพื้นฐาน (design pattern)
Aspect-oriented software development	ชิ้นส่วนต่างๆ จะถูกประสานเข้าด้วยกัน ณ จุดที่ใช้งาน โดยชิ้นส่วนเหล่านั้นถูกสร้างมาจากคนละที่กัน
Component-based software engineering	ระบบที่ถูกสร้างจากการนำชิ้นส่วนมารวมกัน (collections of objects) โดยมีความสอดคล้องกับมาตรฐาน

Approaches that support software reuse

Approach	Description
Configurable application systems	ระบบ Domain-specific ที่ได้รับการออกแบบเพื่อให้สามารถกำหนดค่าให้เหมาะกับความต้องการของลูกค้าระบบเฉพาะ
Design patterns	นิยามสำหรับการออกแบบซอฟต์แวร์ทั่ว ๆ ไป ซึ่ง (ปรากฏอยู่ในทุก application) สามารถนำมาเขียนเป็น นิยามสำหรับ design pattern ที่แสดงถึงการจัดองค์ประกอบและการโต้ตอบกันของวัตถุในระบบ
ERP systems (Enterprise Resource Planning)	ระบบขนาดใหญ่ที่มีการห่อหุ้มฟังก์ชันการทำงานทางธุรกิจทั่วไปและกฎสำหรับองค์กรเข้าไว้ด้วยกัน
Legacy system wrapping	ระบบเดิมถูกห่อด้วยชุดของอินเทอร์เฟซและจัดเตรียมการเข้าถึงระบบเดิมเหล่านี้ผ่านอินเทอร์เฟซเหล่านั้น
Model-driven engineering	ซอฟต์แวร์ที่ถูกออกแบบเป็นโมเดลของโดเมนและใช้โมเดลต่าง ๆ เพื่อสร้างโค้ดให้ทำงานบนระบบต่าง ๆ

Approaches that support software reuse

Approach	Description
Program generators	การสร้างระบบจากชนิดของ application ที่กำหนด ใช้เพื่อสร้างระบบขึ้นจาก system model ที่ผู้ใช้ให้มา
Program libraries	Class และ function libraries ที่สร้างขึ้นจากนิยามมาตรฐาน ที่สามารถนำกลับมาใช้ใหม่ได้
Service-oriented systems	ระบบพัฒนาขึ้นโดยเชื่อมโยงบริการที่ใช้ร่วมกัน ซึ่งอาจได้รับการจัดหาจากภายนอกหรือภายหลัง
Software product lines	Application ที่เขียนขึ้นด้วยสถาปัตยกรรมอย่างกลาง ๆ พร้อมที่จะปรับให้เข้ากับผู้ใช้แต่ละราย
Systems of systems	ระบบสองอย่างหรือมากกว่าที่กระจายอยู่ อาจถูกนำมารวมกันเพื่อสร้างระบบใหม่

ปัจจัยในการ reuse

- กำหนดการในการพัฒนาซอฟต์แวร์
- อายุการใช้งานซอฟต์แวร์ที่คาดไว้
- พื้นฐานทักษะและประสบการณ์ของทีมพัฒนา
- ความสำคัญของซอฟต์แวร์และความต้องการชนิด nonfunctional
- โดเมนของแอ็พพลิเคชัน
- แพลตฟอร์มสำหรับใช้งานซอฟต์แวร์

Application frameworks

Framework definition

- “..an integrated set of software artefacts (such as classes, objects and components) that collaborate to provide a reusable architecture for a family of related applications.”
- “...ชุดของสิ่งประดิษฐ์ซอฟต์แวร์ (เช่น classes, objects และ components) ที่ทำงานร่วมกัน เพื่อให้เกิดเป็นสถาปัตยกรรมที่สามารถนำมาใช้ซ้ำได้สำหรับ application ที่คล้ายกัน"

Application frameworks

- Framework เป็นเอนทิตีขนาดใหญ่ปานกลางที่สามารถนำกลับมาใช้ใหม่ได้
  - พวกมันอยู่ตรงกลางระหว่างการใช้ระบบและองค์ประกอบซ้ำ
- Framework คือระบบย่อย (sub-system) ซึ่งประกอบด้วยคลาสที่เป็นนามธรรมและเป็นรูปธรรมและมีส่วนติดต่อระหว่างกัน
- Sub-system ดังกล่าวจะดำเนินการสร้างแอพพลิเคชั่น
  - โดยการเพิ่มส่วนประกอบที่จำเป็นเพื่อเติมเต็มส่วนต่าง ๆ ของการออกแบบ
  - โดยการสร้าง abstract class ใน framework

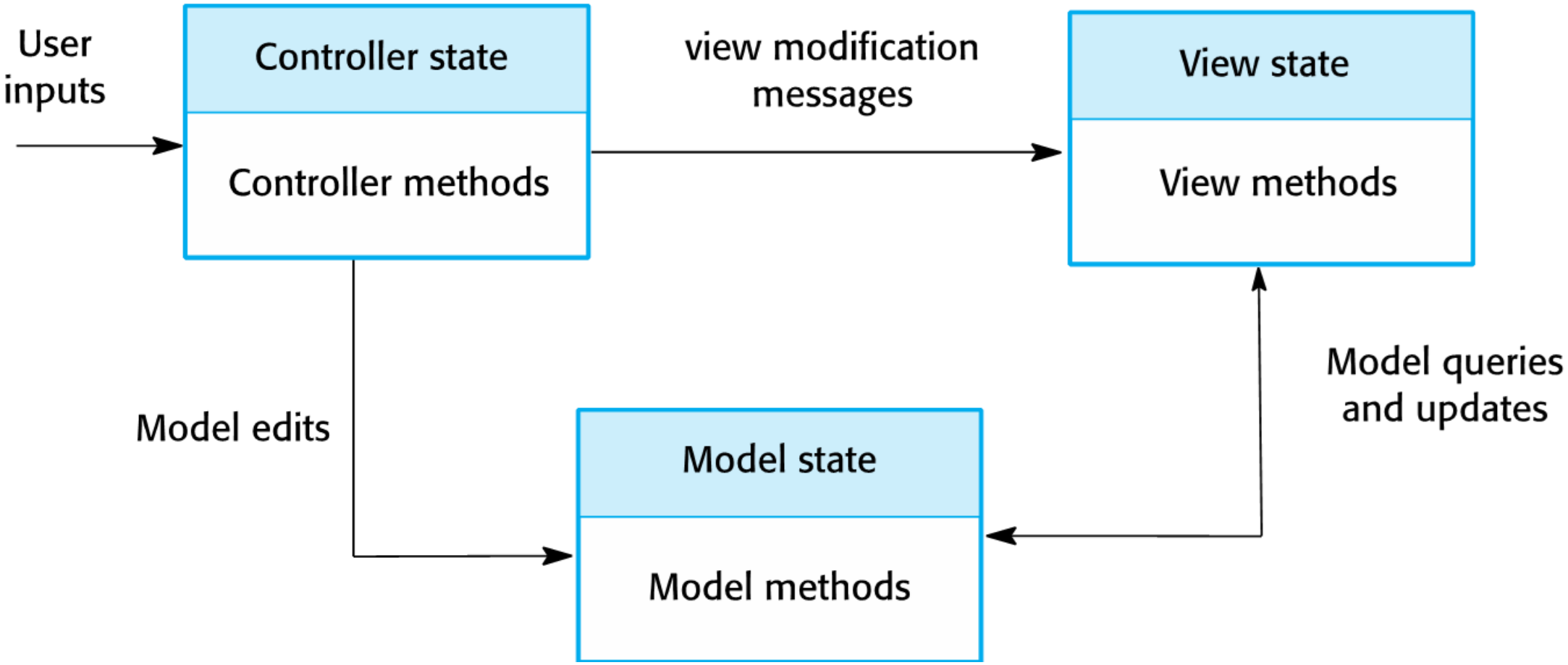
Web application frameworks

- สนับสนุนการสร้างเว็บไซต์แบบไดนามิกเป็นส่วนหน้า (front-end) สำหรับแอปพลิเคชันเว็บ
- WAF มีให้บริการสำหรับภาษาโปรแกรมเว็บที่ใช้บ่อย ๆ เช่น Java, Python, Ruby เป็นต้น
- รูปแบบการโต้ตอบจะขึ้นอยู่กับ design pattern แบบ Model-View-Controller

Model-view controller

- เป็นกรอบโครงสร้างพื้นฐานของระบบสำหรับการออกแบบ GUI
- อนุญาตให้มีการนำเสนอวัตถุได้ในหลาย ๆ ลักษณะ แยกส่วนควบคุม ส่วนปฏิสัมพันธ์และส่วนนำเสนอเหล่าออกจากกัน
- MVC framework เกี่ยวข้องกับ design pattern อื่น ๆ อีกจำนวนมาก (ดูเรื่อง design pattern ประกอบ)

The Model-View-Controller pattern



WAF (Web Application Framework) features

- Security (ความปลอดภัย)
  - WAFs อาจรวมเอา classes เพื่อช่วยในการรับรองความถูกต้องของผู้ใช้ (login) และการเข้าใช้งาน (Authentication)
- Dynamic web pages (หน้าเว็บแบบไดนามิก)
  - Classes ถูกจัดเตรียมไว้เพื่อให้สามารถกำหนดเทมเพลตของเว็บเพจและเติมข้อมูลจากฐานข้อมูลไปยังหน้าเพจแบบไดนามิก
- Database support (สนับสนุนฐานข้อมูล)
  - อาจจัดเตรียมคลาสที่เป็นนามธรรมเพื่อติดต่อกับฐานข้อมูลที่แตกต่างกัน

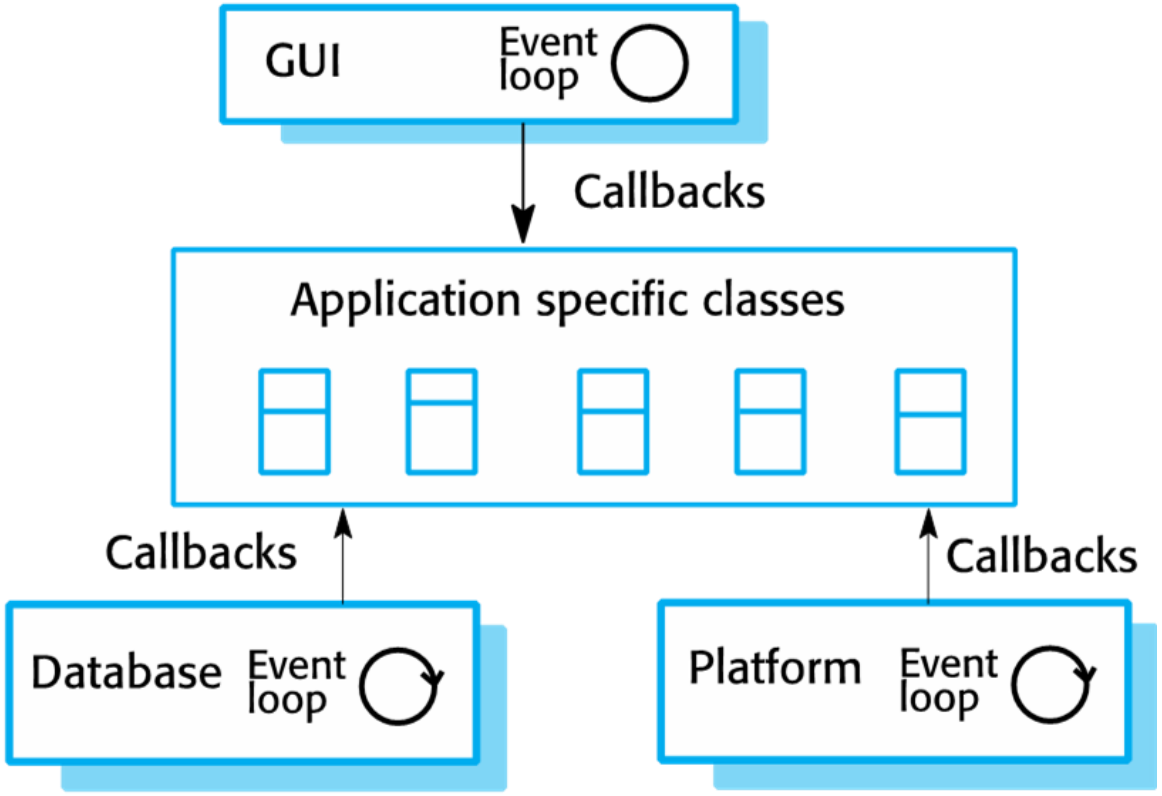
WAF (Web Application Framework) features

- Session management (การจัดการเซสชัน)
  - โดยปกติ WAF จะจัดเตรียมคลาสเพื่อสร้างและบริหาร
- User interaction (การโต้ตอบของผู้ใช้)
  - Web frameworks ส่วนใหญ่มักจะมีการรองรับ AJAX เพื่อการโต้ตอบกับผู้ใช้ที่ดียิ่งขึ้น

Extending frameworks

- Framework มักมีรูปแบบเป็นแบบทั่วไปและสามารถขยายความสามารถเพื่อสร้าง application หรือระบบย่อยเฉพาะเจาะจงมากขึ้น
  - มักถูกใช้เพื่อสร้างโครงสถาปัตยกรรมสำหรับระบบ
- การขยาย framework หมายถึง
  - การเพิ่มคลาสคอนกรีตที่สืบทอดการดำเนินงานจากคลาสนามธรรมใน framework
  - การเพิ่ม method เพื่อตอบสนองต่อเหตุการณ์ที่ framework รู้จัก
- ปัญหาสำคัญเกี่ยวกับ framework คือความซับซ้อน
  - ซึ่งหมายความว่าต้องใช้เวลานานเพื่อศึกษาและใช้งานอย่างมีประสิทธิภาพ

Inversion of control in frameworks



Framework classes

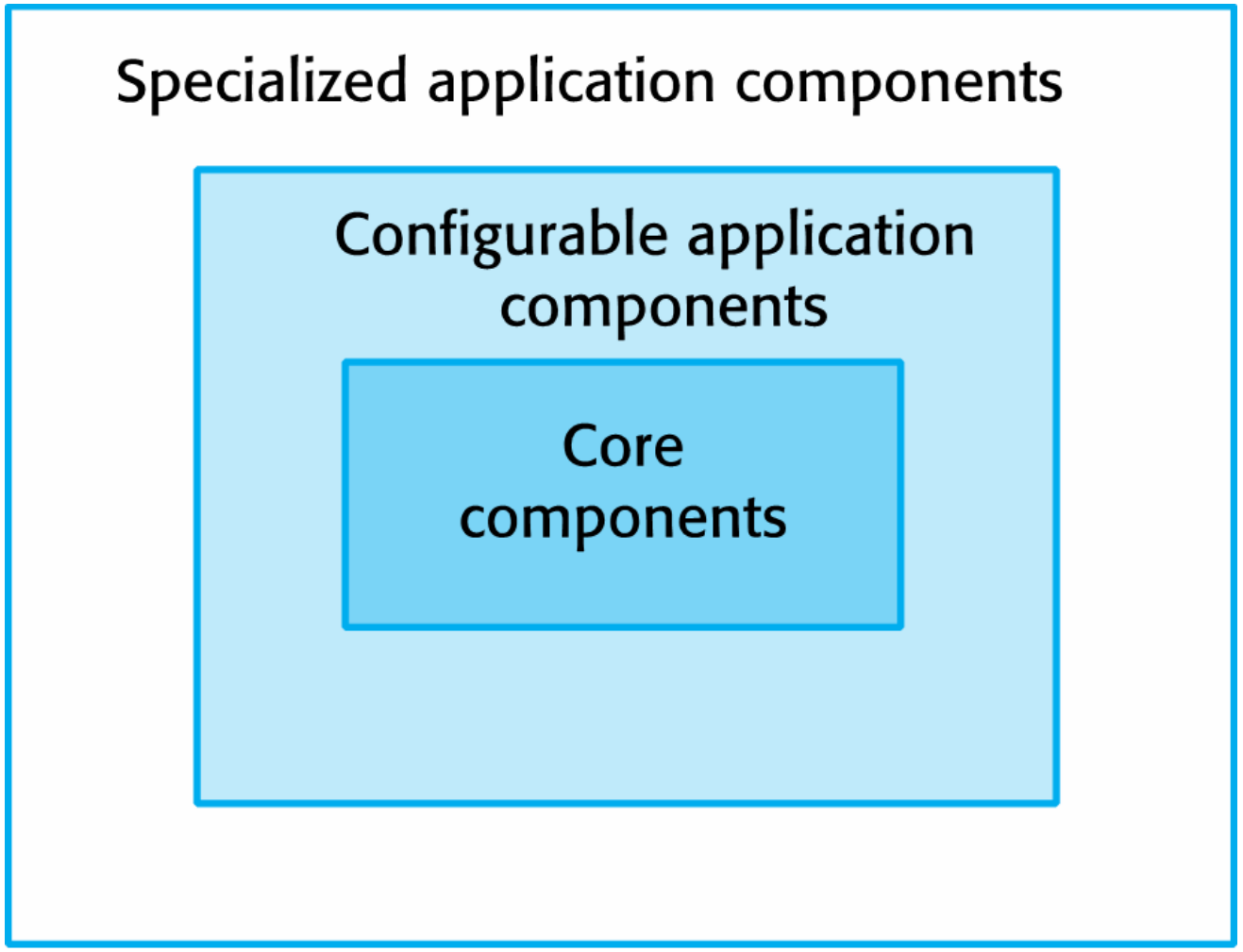
- System infrastructure frameworks
  - สนับสนุนการพัฒนาโครงสร้างพื้นฐานของระบบ เช่น การสื่อสาร อินเทอร์เฟซสำหรับผู้ใช้ และคอมไพเลอร์
- Middleware integration frameworks
  - มาตรฐาน และ class ที่สนับสนุนการสื่อสารและการแลกเปลี่ยนข้อมูล
- Enterprise application frameworks
  - สนับสนุนการพัฒนาโปรแกรมเฉพาะ เช่น ระบบโทรคมนาคมหรือระบบการเงิน

Software product lines

Software product lines

- Software product lines หรือ application families คือ application ที่มีฟังก์ชันการทำงานทั่วไป ซึ่งสามารถปรับและกำหนดค่าเพื่อใช้ในบริบทเฉพาะได้
- Software product lines คือชุดของ application ที่มีสถาปัตยกรรมทั่วไปและชิ้นส่วนที่ใช้ร่วมกัน
  - โดยแต่ละ application มีความสามารถเฉพาะเพื่อตอบสนองต่อความต้องการที่แตกต่างกันของผู้ใช้
- การปรับ application อาจหมายถึง:
  - ส่วนประกอบและการกำหนดค่าระบบ
  - การเพิ่มส่วนประกอบใหม่ลงในระบบ
  - การเลือกจากไลบรารีส่วนประกอบที่มีอยู่
  - แก้ไขส่วนประกอบเพื่อตอบสนองความต้องการใหม่ ๆ

Base systems for a software product line



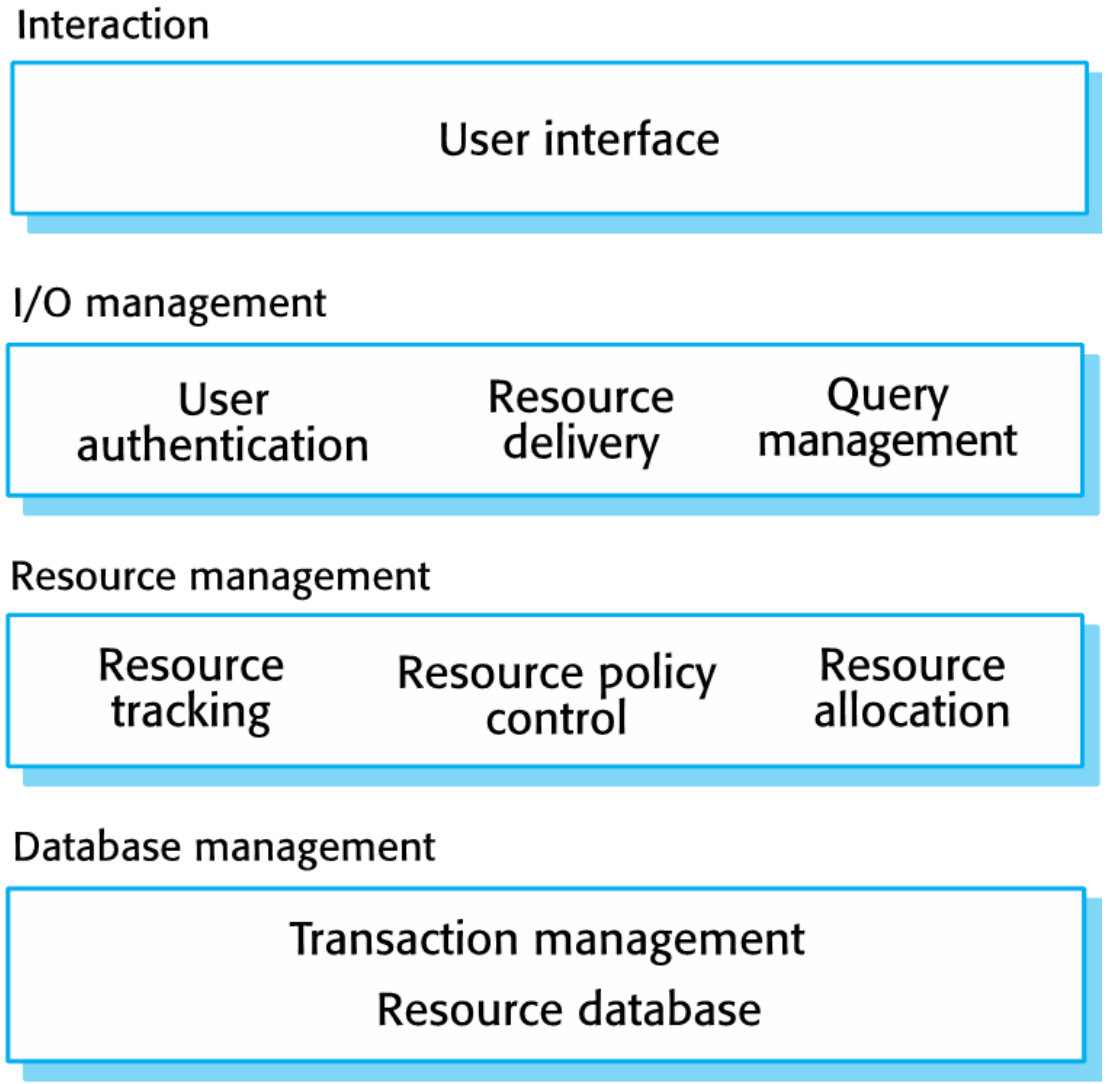
Base applications

- Core components (องค์ประกอบหลัก)
  - ที่ให้การสนับสนุนโครงสร้างพื้นฐาน ส่วนนี้มักจะไม่ต้องแก้ไขเมื่อพัฒนาอินสแตนซ์ใหม่ของสายผลิตภัณฑ์
- Configurable components (ส่วนประกอบที่สามารถกำหนดค่าได้)
  - อาจมีการปรับเปลี่ยนและกำหนดค่าให้จำเพาะเจาะจงกับแอ็พพลิเคชันใหม่
  - บางครั้งสามารถกำหนดค่าคอมโพเนนต์เหล่านี้ใหม่โดยไม่ต้องแก้โค้ดที่เขียนด้วยภาษาหลัก
- Specialized, domain-specific components (คอมโพเนนต์เฉพาะโดเมน)
  - บางส่วนหรือทั้งหมดของส่วนนี้อาจถูกแทนที่เมื่อมีการสร้างอินสแตนซ์ใหม่ของสายผลิตภัณฑ์

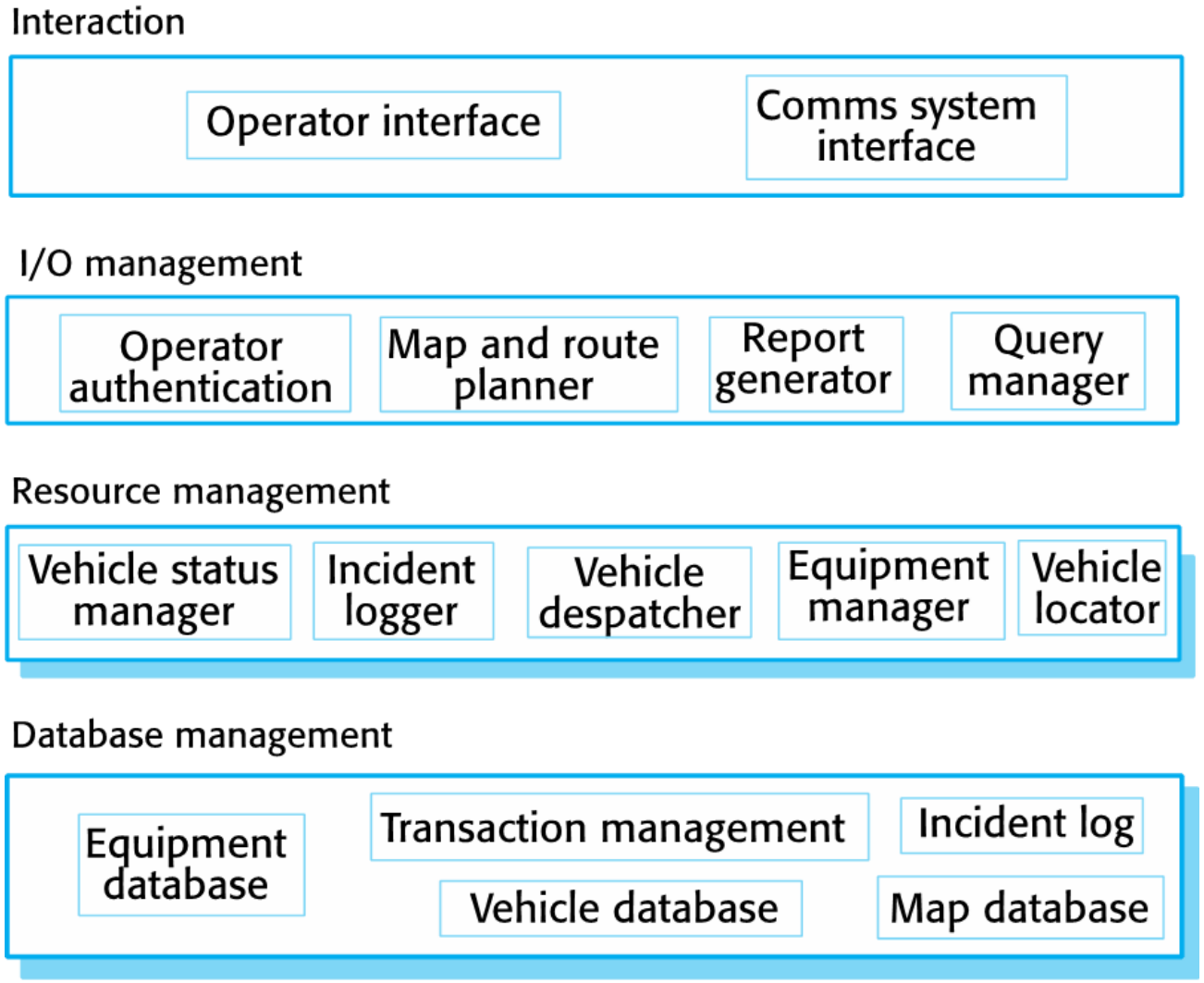
Application frameworks and product lines

- Application frameworks อาศัยคุณสมบัติเชิงวัตถุ เช่น polymorphism เพื่อใช้ส่วนขยาย
  - Product lines ไม่จำเป็นต้องเป็นแบบเชิงวัตถุ (เช่นซอฟต์แวร์ฝังตัวสำหรับโทรศัพท์มือถือ)
- Application frameworks เน้นการให้บริการด้านเทคนิคมากกว่าการสนับสนุนเฉพาะโดเมน
  - Product lines สร้างขึ้นเฉพาะและต้องมีข้อมูลโดเมนและแพลตฟอร์มเสมอ

The architecture of a resource allocation system



The product line architecture of a vehicle dispatcher



Application system reuse

Application system reuse

- Application system product เป็นระบบซอฟต์แวร์ที่สามารถปรับให้เหมาะสำหรับลูกค้ารายต่าง ๆ โดยไม่ต้องเปลี่ยนซอร์สโค้ดของระบบ
- Application systems มีคุณสมบัติทั่วไปและสามารถใช้หรือนำมาใช้ซ้ำได้ในสภาพแวดล้อมที่แตกต่างกัน
- Application system product ได้รับการดัดแปลงโดยใช้กลไกการกำหนดค่าที่มีอยู่แล้วเพื่อให้สามารถทำงานได้ตามความต้องการเฉพาะของลูกค้า
  - ตัวอย่างเช่นในระบบบันทึกผู้ป่วยในโรงพยาบาลอาจมีการกำหนดรูปแบบการป้อนข้อมูลและรายงานการส่งออกแยกต่างหากสำหรับผู้ป่วยประเภทต่าง ๆ โดยไม่ต้องแก้ไข source code

Benefits of application system reuse

- สามารถ deploy ระบบที่เชื่อถือได้ ได้รวดเร็วขึ้น
- สามารถเห็นความสามารถของแอพพลิเคชันที่จะนำมาใช้ใหม่ ดังนั้นจึงง่ายต่อการตัดสินว่าน่าจะเหมาะสมหรือไม่
- หลีกเลี่ยงความเสี่ยงในการพัฒนาโดยใช้ซอฟต์แวร์ที่มีอยู่
  - ธุรกิจสามารถมุ่งเน้นไปที่กิจกรรมหลักของพวกเขาโดยไม่ต้องทุ่มเททรัพยากรจำนวนมากในการพัฒนาระบบไอที
  - การดำเนินงานการปรับปรุงเทคโนโลยีเป็นของผู้ผลิตผลิตภัณฑ์ COTS ซึ่งมีความเชี่ยวชาญเฉพาะด้าน จะส่งผลดีต่อลูกค้ามากกว่า

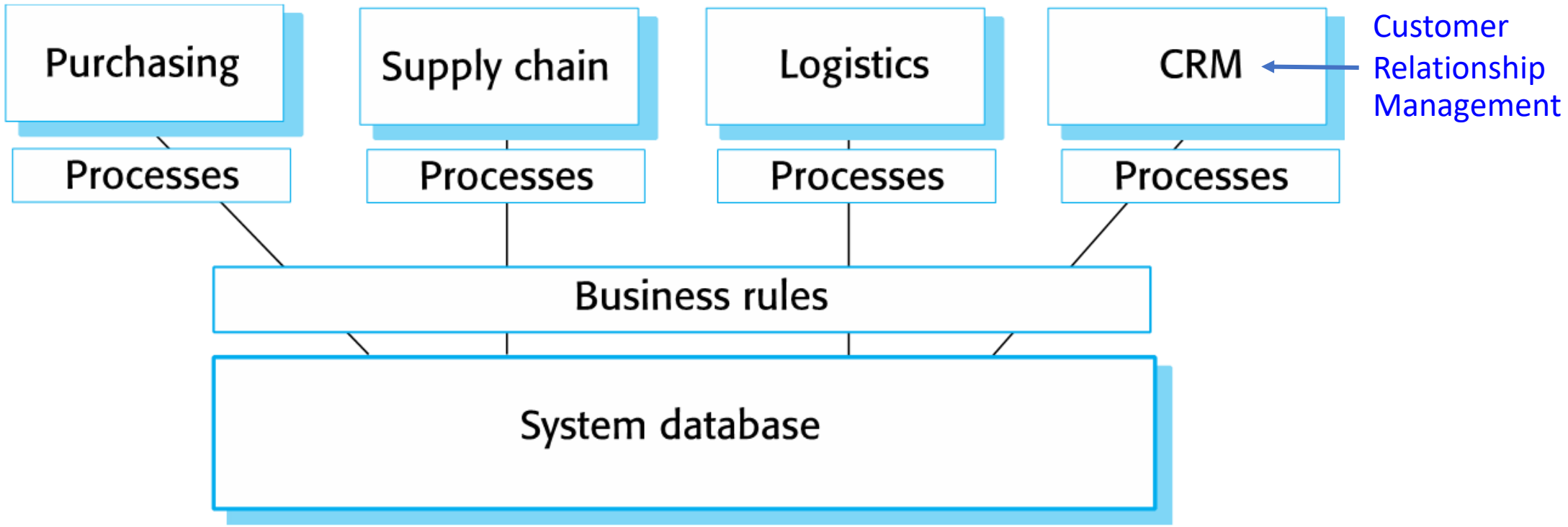
Problems of application system reuse

- อาจจะต้องมีการปรับ requirement เพื่อให้สอดคล้องกับของผลิตภัณฑ์ COTS
- เมื่อใช้ผลิตภัณฑ์ COTS เป็นส่วนประกอบในระบบ ต้องทำใจว่าอาจเป็นไปไม่ได้ที่จะเปลี่ยนแปลงได้ทันความต้องการของลูกค้า
- การเลือกระบบ COTS ที่เหมาะสมสำหรับองค์กรอาจเป็นกระบวนการที่ยากลำบาก โดยเฉพาะอย่างยิ่งผลิตภัณฑ์ COTS จำนวนมากไม่ได้รับการจัดทำเป็นอย่างดี
- อาจมีการขาดความชำนาญจากผู้ผลิต COTS ในการสนับสนุนการพัฒนาระบบ
- ผู้จำหน่ายผลิตภัณฑ์ COTS มักจะกลายเป็นผู้ควบคุมการสนับสนุนและวิวัฒนาการของระบบ

ERP systems

- ระบบการวางแผนทรัพยากรขององค์กร (Enterprise Resource Planning : ERP) เป็นระบบทั่วไปที่สนับสนุนกระบวนการทางธุรกิจ เช่น การสั่งซื้อและการออกใบแจ้งหนี้ การผลิต ฯลฯ
- ระบบเหล่านี้ใช้กันอย่างแพร่หลายในบริษัทขนาดใหญ่ (เป็นรูปแบบที่ใช้กันโดยทั่วไปของซอฟต์แวร์)
- แกนหลักของซอฟต์แวร์ ได้รับการดัดแปลงโดยการรวมโมดูลและรวมเอาความรู้เกี่ยวกับกระบวนการและกฎเกณฑ์ทางธุรกิจเข้าไว้ด้วยกัน

The architecture of an ERP system



ERP architecture

- ประกอบด้วยโมดูลจำนวนหนึ่ง เพื่อรองรับฟังก์ชันต่าง ๆ ทางธุรกิจ
- ชุดกระบวนการทางธุรกิจที่กำหนดไว้ในแต่ละโมดูล เกี่ยวข้องกับกิจกรรมในโมดูลนั้น
- ฐานข้อมูลทั่วไปที่เก็บรักษาข้อมูลเกี่ยวกับฟังก์ชันทางธุรกิจทั้งหมดที่เกี่ยวข้อง
- ชุดของกฎเกณฑ์ทางธุรกิจที่ใช้กับข้อมูลทั้งหมดในฐานข้อมูล

ERP configuration

- เลือกฟังก์ชันที่ต้องการจากระบบ
- สร้างแบบจำลองข้อมูล ที่กำหนดว่าข้อมูลขององค์กรจะมีโครงสร้างในฐานข้อมูลระบบอย่างไร
- กำหนดกฎเกณฑ์ทางธุรกิจที่ใช้กับข้อมูลนั้น
- กำหนดปฏิสัมพันธ์ที่คาดไว้กับระบบภายนอก
- ออกแบบแบบฟอร์มการป้อนข้อมูลและรายงานผลลัพธ์ที่สร้างขึ้นโดยระบบ
- ออกแบบกระบวนการทางธุรกิจที่สอดคล้องกับรูปแบบกระบวนการของระบบ
- ตั้งค่าพารามิเตอร์ที่กำหนดวิธีการนำระบบไปใช้งาน (บนแพลตฟอร์มต้นแบบ)

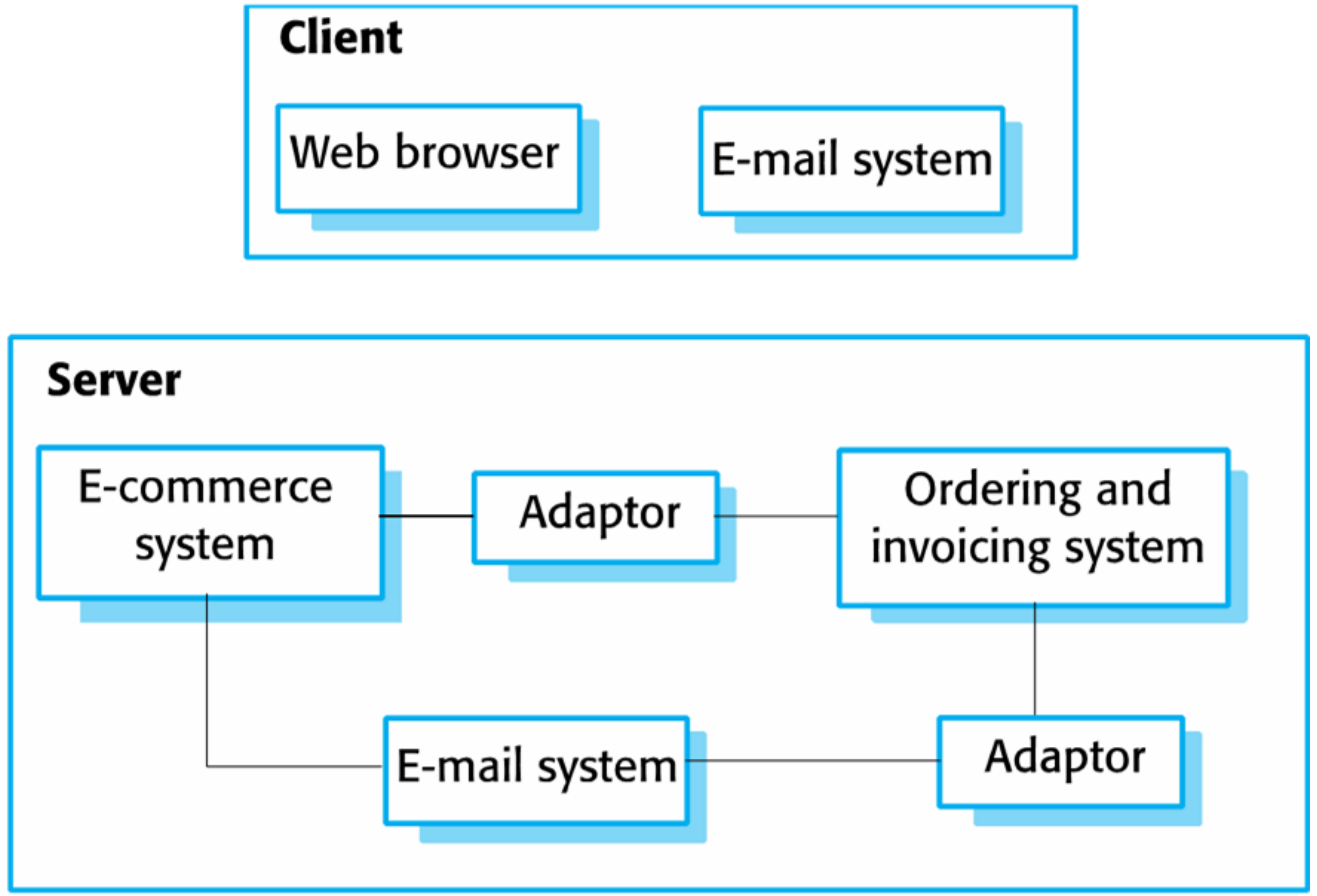
Integrated application systems

- ระบบแอ็พพลิเคชันรวม (Integrated application systems) คือ application ที่มี application systems สองตัวขึ้นไป รวมถึงระบบ application แบบเดิม
- เราอาจใช้วิธีนี้เมื่อไม่มีระบบ application เดียวที่ตรงกับความต้องการทั้งหมด หรือเมื่อต้องการรวมระบบ application ใหม่เข้ากับระบบที่มีใช้อยู่แล้ว

Design choices

- Application systems ใดที่มีฟังก์ชันการทำงานที่เหมาะสมที่สุด
  - โดยปกติจะมีผลิตภัณฑ์ application systems จำนวนหนึ่งพร้อมใช้งาน ซึ่งสามารถนำมาใช้ร่วมกันได้หลายวิธี
- ข้อมูลจะถูกแลกเปลี่ยนอย่างไร?
  - ผลิตภัณฑ์ที่แตกต่างกันมักใช้โครงสร้างและรูปแบบข้อมูลที่ต่างกัน อาจจะต้องเขียนอะแดปเตอร์ที่แปลงจาก application หนึ่งไปยังอีกที่หนึ่ง
- คุณสมบัติของผลิตภัณฑ์ที่จะใช้จริง?
  - ผลิตภัณฑ์หลาย ๆ ตัวที่จะนำมาใช้ร่วมกันนั้นในแต่ละตัวอาจมีฟังก์ชันการทำงานมากกว่าที่ต้องการ
  - ฟังก์ชันการทำงานอาจซ้ำกันในผลิตภัณฑ์ต่างๆ

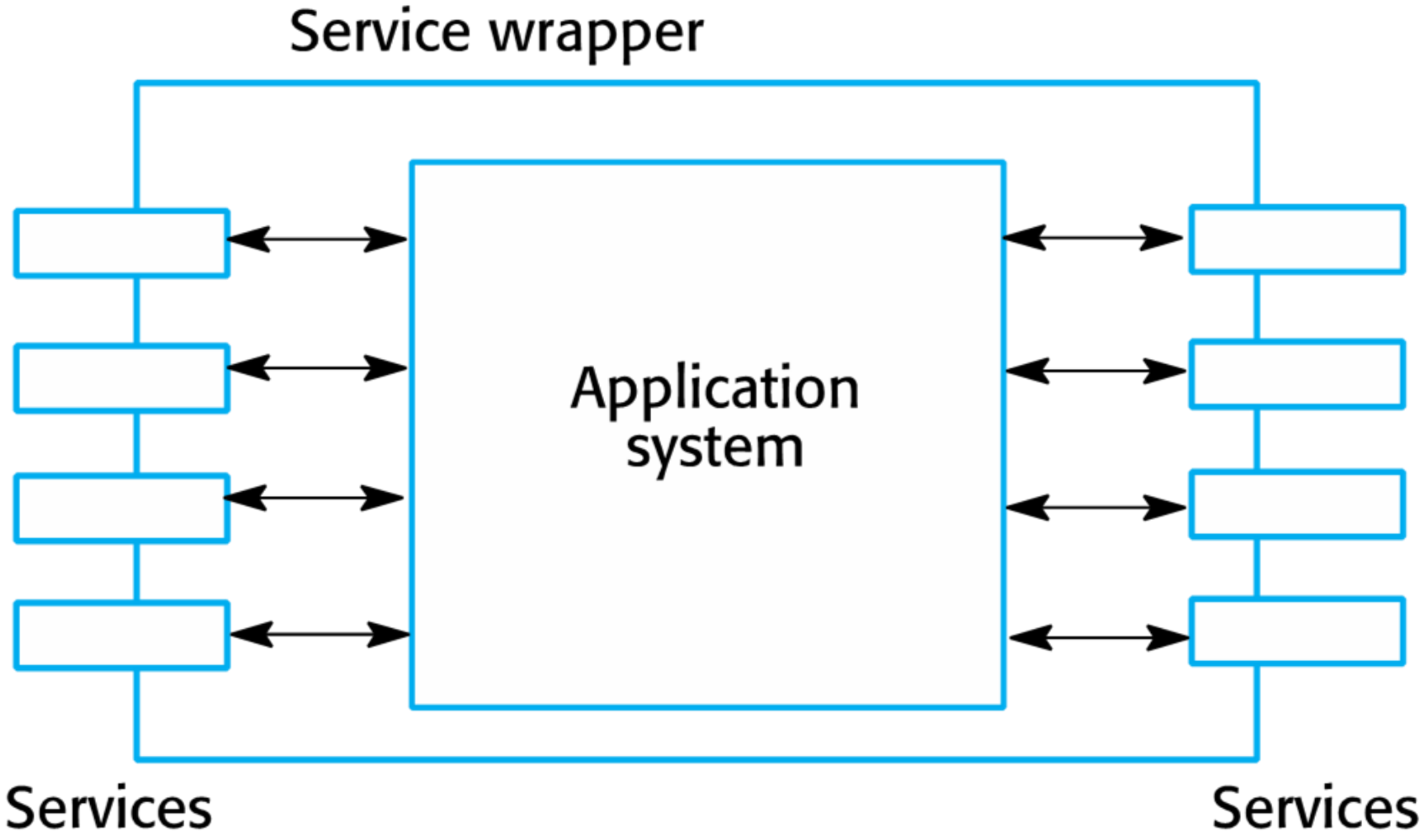
An integrated procurement system



Service-oriented interfaces

- การรวมระบบแอพพลิเคชันสามารถทำได้ง่ายขึ้นถ้าใช้วิธีการ service-oriented
- วิธีการที่มุ่งเน้นบริการหมายถึง
  - การอนุญาตให้เข้าถึงฟังก์ชันการทำงานของ application system โดยใช้ standard service interface
  - มี service สำหรับแต่ละส่วนที่ทำงานแยกกันเด็ดขาด
- บางโปรแกรมอาจมี interface สำหรับ service อยู่แล้ว
  - แต่บางครั้ง service interface นี้จะถูกใช้โดย system integrator
  - อาจต้องสร้างโปรแกรม wrapper ที่ซ่อน application ไว้ และให้บริการที่เรียกใช้จากภายนอก

Application wrapping



Application system integration problems

- ขาดการควบคุมการทำงานและประสิทธิภาพต่ำ
  - Application system อาจมีประสิทธิภาพน้อยกว่าที่ปรากฏ
- ปัญหาเกี่ยวกับการร่วมงานระหว่าง application
  - Application system ที่แตกต่างกันอาจมีส่วนที่ซ้ำซ้อนกันแต่มีกลไกการทำงานที่แตกต่างกันซึ่งหมายความว่าการผสานรวมนั้นทำได้ยาก
- ไม่สามารถควบคุมวิวัฒนาการของระบบ
  - ผู้ขายระบบคือผู้ควบคุมวิวัฒนาการของระบบ เราในฐานะผู้ dev ระบบไม่สามารถควบคุมได้ทั้งหมด
- การสนับสนุนจากผู้ขายระบบ
  - ผู้จำหน่าย application system อาจไม่มีการสนับสนุนตลอดอายุการใช้งานของผลิตภัณฑ์

Key points

- มีหลายวิธีในการนำซอฟต์แวร์มาใช้ใหม่ ตั้งแต่การนำมาใช้ใหม่ของ class และ method ใน library จนถึงการนำมาใช้ใหม่ของ application system ที่สมบูรณ์
- ข้อดีของการนำซอฟต์แวร์กลับมาใช้ใหม่คือ
  - การลดต้นทุนการพัฒนาซอฟต์แวร์
  - ความเสี่ยงที่ต่ำลง
  - ความเชื่อถือได้ของระบบเพิ่มขึ้น
  - สามารถใช้ความเชี่ยวชาญได้อย่างมีประสิทธิภาพมากขึ้น โดยมุ่งเน้นความเชี่ยวชาญในการออกแบบชิ้นส่วนที่นำมาใช้ซ้ำได้

Key points

- Application frameworks คือ collection ของ object แบบ concrete และ abstract
  - ออกแบบมาเพื่อ reuse โดยใช้ความเชี่ยวชาญและการเพิ่มวัตถุใหม่ ๆ
  - มักจะผ่านกระบวนการออกแบบที่ดีโดยใช้ design pattern
- Software product lines เป็น application ที่เกี่ยวข้องกัน ซึ่งพัฒนาขึ้นจาก application พื้นฐานอย่างน้อยหนึ่งรายการ
  - ในระบบทั่วไปอาจมีการปรับแต่งเพื่อตอบสนองความต้องการเฉพาะ
  - อาจจะปรับแต่งฟังก์ชันการทำงานแพลตฟอร์มเป้าหมายหรือปรับแต่งโดยการกำหนดค่าใช้งาน (configuration)

Key points

- การนำ Application system กลับมาใช้ใหม่มักจะทำเมื่อมีการการนำระบบขนาดใหญ่ที่ใช้แล้วมาใช้ซ้ำ
  - ระบบเหล่านี้มีฟังก์ชันมากมายและการนำกลับมาใช้ใหม่สามารถลดต้นทุนและเวลาในการพัฒนาได้อย่างสิ้นเชิง
  - ระบบอาจได้รับการพัฒนาโดยการ configuration ใหม่ หรือโดยการรวม Application system เข้าด้วยกัน
- ปัญหาที่อาจเกิดขึ้นกับการใช้ซ้ำของ Application system ได้แก่
  - การขาดการควบคุมการทำงานและประสิทธิภาพในการทำงาน
  - การขาดการควบคุมวิวัฒนาการของระบบ
  - ในการตอบสนองต่อการเปลี่ยนแปลงคามต้องการของลูกค้า จะต้องได้รับการสนับสนุนจากผู้ขายภายนอก
  - ความยากลำบากในการทำให้ระบบสามารถทำงานร่วมกันได้

คำถาม???