Software Evolution

Week 13

ดัดแปลงจาก slides ของหนังสือ Software Engineering [1]

หัวข้อที่จะศึกษา

- Evolution processes
- Legacy systems
- Software maintenance

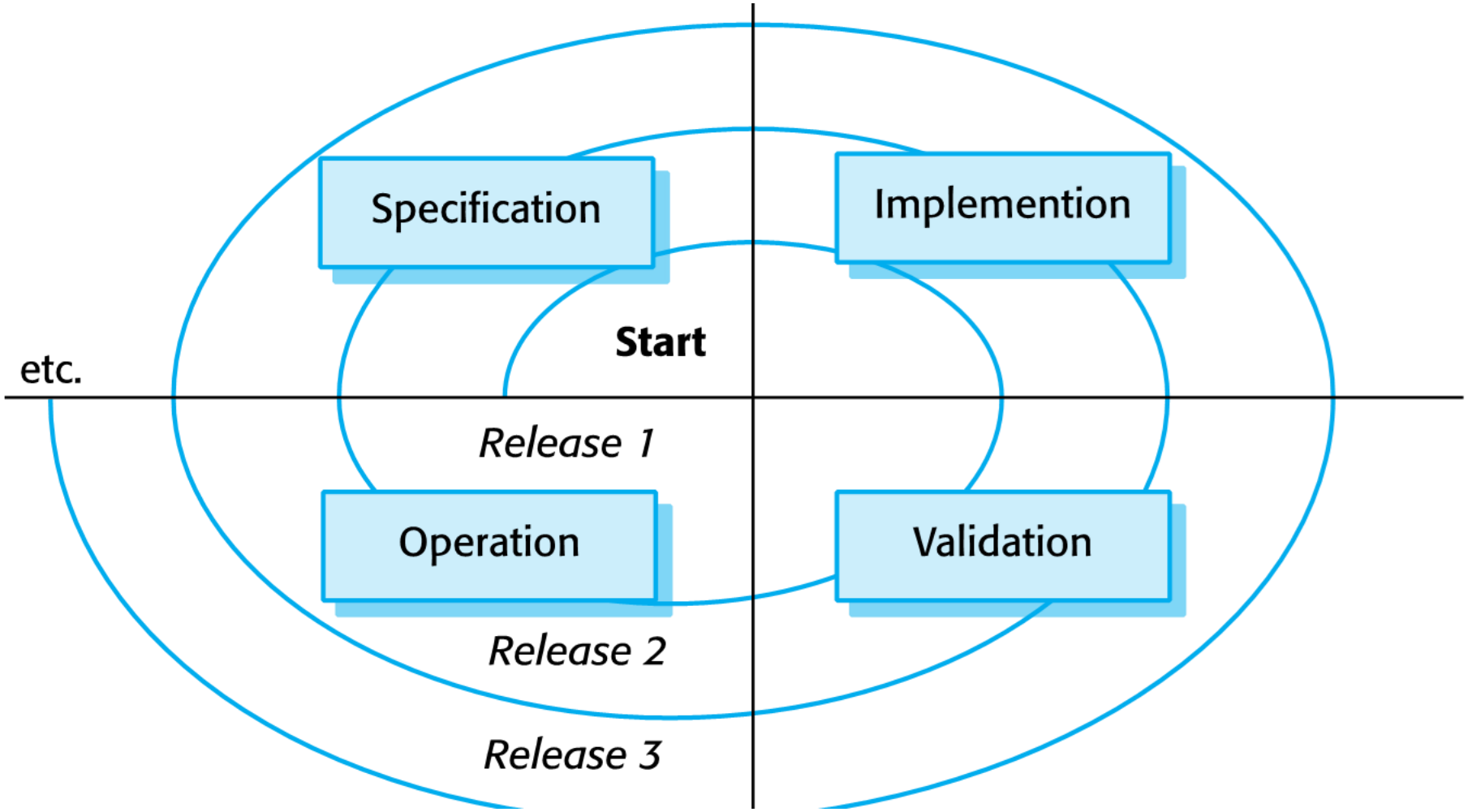
Software change

- การเปลี่ยนแปลงของซอฟต์แวร์เป็นสิ่งที่ไม่สามารถหลีกเลี่ยงได้
  - มี requirements เกิดขึ้นเมื่อใช้ซอฟต์แวร์ไปสักระยะ
  - สภาพแวดล้อมทางธุรกิจเปลี่ยนไปจากเดิม
  - มีข้อผิดพลาดที่ต้องได้รับการซ่อมแซม
  - มีคอมพิวเตอร์และอุปกรณ์ใหม่เพิ่มเข้ามาในระบบ
  - ต้องมีการปรับปรุงประสิทธิภาพหรือความน่าเชื่อถือของระบบ
- ปัญหาสำคัญสำหรับทุกองค์กรคือการพัฒนาและจัดการการเปลี่ยนแปลงของระบบซอฟต์แวร์ที่มีอยู่

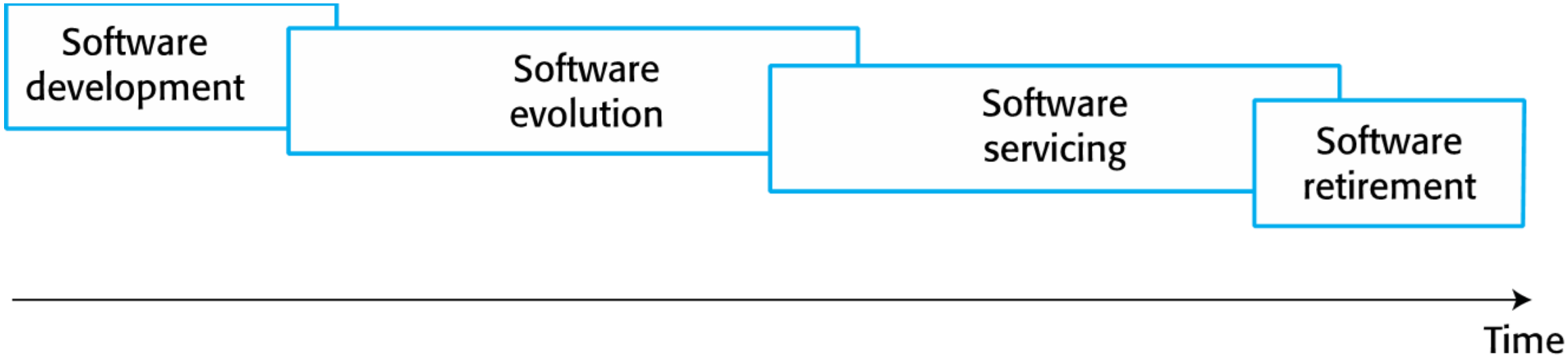
Importance of evolution

- ซอฟต์แวร์ถือเป็นสินทรัพย์ทางธุรกิจที่สำคัญ ซึ่งองค์กรมักจะมีการลงทุนขนาดใหญ่ในเรื่องดังกล่าว
- เพื่อรักษาคุณค่าของสินทรัพย์เหล่านี้ให้กับธุรกิจ ต้องมีการเปลี่ยนแปลงและปรับปรุงซอฟต์แวร์ให้ทันสมัยและสอดคล้องกับความต้องการอยู่เสมอ
- งบประมาณของบริษัทขนาดใหญ่ มีไว้สำหรับการเปลี่ยนและพัฒนาซอฟต์แวร์ที่มีอยู่**มากกว่า**การพัฒนาซอฟต์แวร์ใหม่

A spiral model of development and evolution



Evolution and servicing



Evolution and servicing

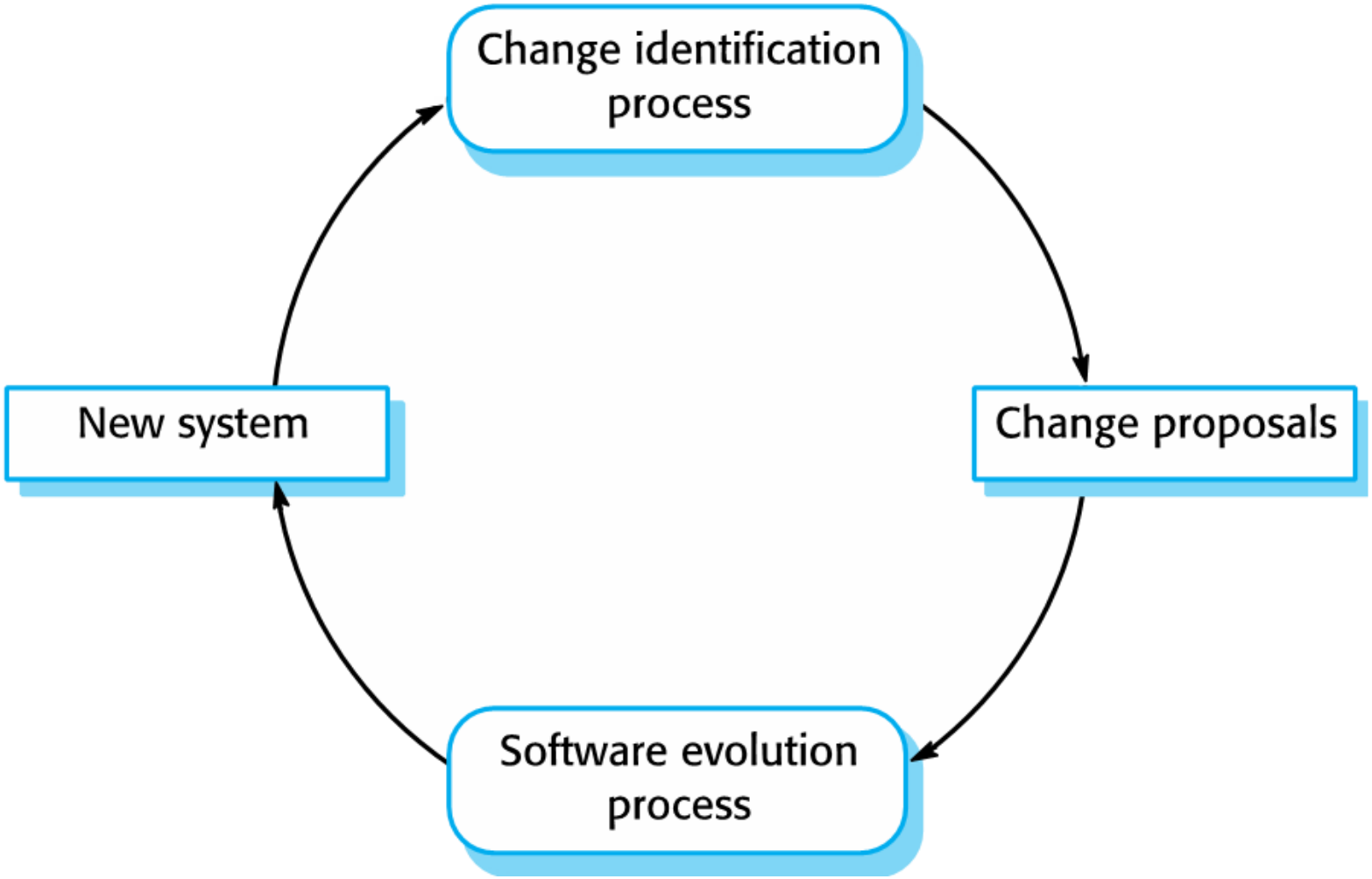
- วิวัฒนาการ (Evolution)
  - ขั้นตอนในวงจรชีวิตของระบบซอฟต์แวร์ที่มีการใช้งานและมีการพัฒนาตามความต้องการใหม่ ๆ
- บริการ (Servicing)
  - ในขั้นตอนนี้ซอฟต์แวร์ยังคงมีประโยชน์ แต่จะต้องมีการเปลี่ยนแปลงบางอย่างเพื่อให้สามารถใช้งานได้ เช่น
    - แก้ไขข้อบกพร่องและการเปลี่ยนแปลง เพื่อให้สอดคล้องกับสภาพแวดล้อมของซอฟต์แวร์
    - ไม่มีการเพิ่มเติม function ใหม่ ๆ
- Phase-out (--> Retirement)
  - ซอฟต์แวร์อาจยังคงใช้อยู่ แต่จะไม่มีการเปลี่ยนแปลงใด ๆ อีกแล้ว

Evolution processes

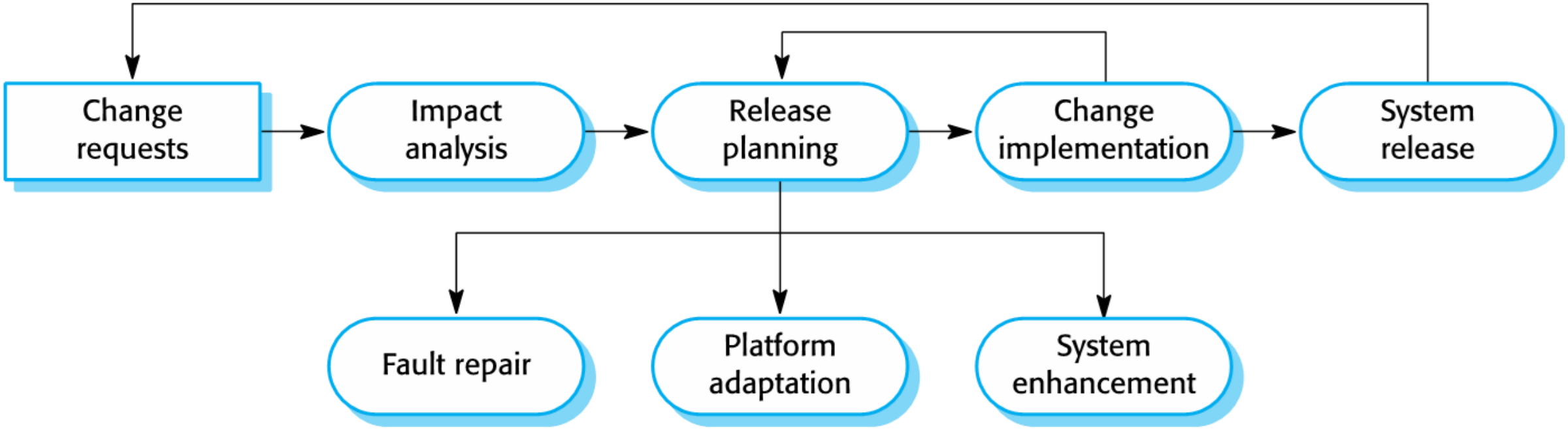
Evolution processes

- กระบวนการวิวัฒนาการของซอฟต์แวร์ขึ้นอยู่กับ
  - ประเภทของซอฟต์แวร์ที่ต้องบำรุงรักษา
  - กระบวนการพัฒนาที่ใช้
  - ทักษะและประสบการณ์ของคนที่เกี่ยวข้อง
- ข้อเสนอ (Proposals) สำหรับการเปลี่ยนแปลง มักจะเป็นตัวขับเคลื่อนสำหรับวิวัฒนาการของระบบ
  - ควรเชื่อมโยงกับ component ที่ได้รับผลกระทบจากการเปลี่ยนแปลง
  - ซึ่งจะทำให้สามารถประมาณการต้นทุนตลอดจนผลกระทบจากการเปลี่ยนแปลง
- การเปลี่ยนแปลงและวิวัฒนาการ จะดำเนินไปตลอดอายุการใช้งานของระบบ

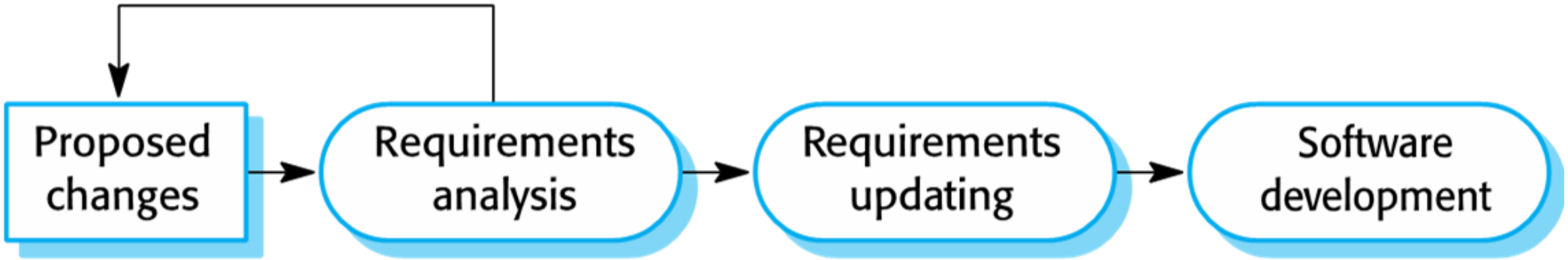
Change identification and evolution processes



The software evolution process



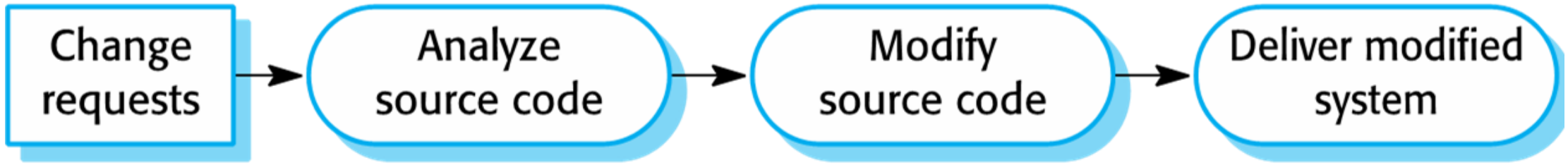
Change implementation



Urgent change requests

- การเปลี่ยนแปลงอย่างเร่งด่วน อาจต้องดำเนินการโดยไม่ต้องผ่านขั้นตอนทั้งหมดของกระบวนการวิศวกรรมซอฟต์แวร์
  - หากมีข้อผิดพลาดร้ายแรงของระบบต้องได้รับการซ่อมแซมเพื่อให้สามารถใช้งานได้ตามปกติ
  - หากการเปลี่ยนแปลงสภาพแวดล้อมของระบบ (เช่น การอัปเกรดระบบปฏิบัติการ) แล้วส่งผลกระทบที่ไม่คาดคิด
  - หากมีการเปลี่ยนแปลงทางธุรกิจที่ต้องการการตอบสนองที่รวดเร็วมาก (เช่นการเปิดตัวผลิตภัณฑ์ที่แข่งขันกัน)

The emergency repair process



Handover problems

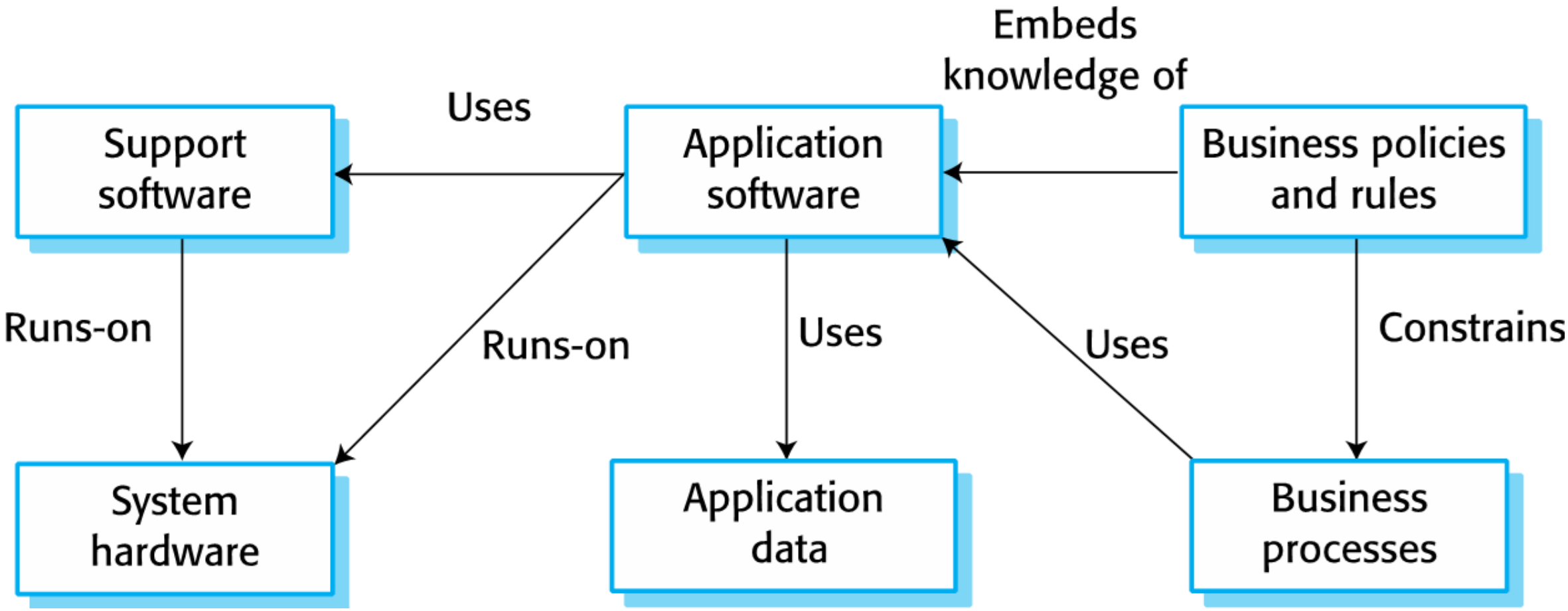
- ในกรณีที่ทีมพัฒนาใช้วิธี agile แต่ทีมวิวัฒนาการไม่คุ้นเคยกับวิธี agile และชอบวิธีการตามแผน (plan-based approach)
  - ทีมวิวัฒนาการคาดหวังเอกสารรายละเอียดเพื่อสนับสนุนวิวัฒนาการ
  - ปกติกระบวนการ agile มักจะไม่มีเอกสารเยอะขนาดที่ต้องการ
- ในกรณีที่มีการใช้วิธีการตามแผน (plan-based approach) เป็นแนวทางในการพัฒนา แต่ทีมวิวัฒนาการต้องการใช้วิธีการแบบ agile

Legacy systems

Legacy systems

- ระบบเดิม (Legacy systems) เป็นระบบเก่าที่อาศัยภาษาและเทคโนโลยีที่ไม่ได้ใช้สำหรับการพัฒนาระบบใหม่อีกต่อไป
- ซอฟต์แวร์ระบบเดิมอาจขึ้นอยู่กับฮาร์ดแวร์ที่เก่ากว่า เช่น คอมพิวเตอร์เมนเฟรม และอาจมีกระบวนการและขั้นตอนเดิมที่เกี่ยวข้อง
- ระบบเดิมไม่ได้เป็นเพียงระบบซอฟต์แวร์ แต่เป็นระบบทางด้านเทคนิคและสภาพแวดล้อมที่กว้างกว่า
  - รวมถึงฮาร์ดแวร์ซอฟต์แวร์ไลบรารี่และซอฟต์แวร์สนับสนุน
  - และกระบวนการทางธุรกิจอื่น ๆ

The elements of a legacy system



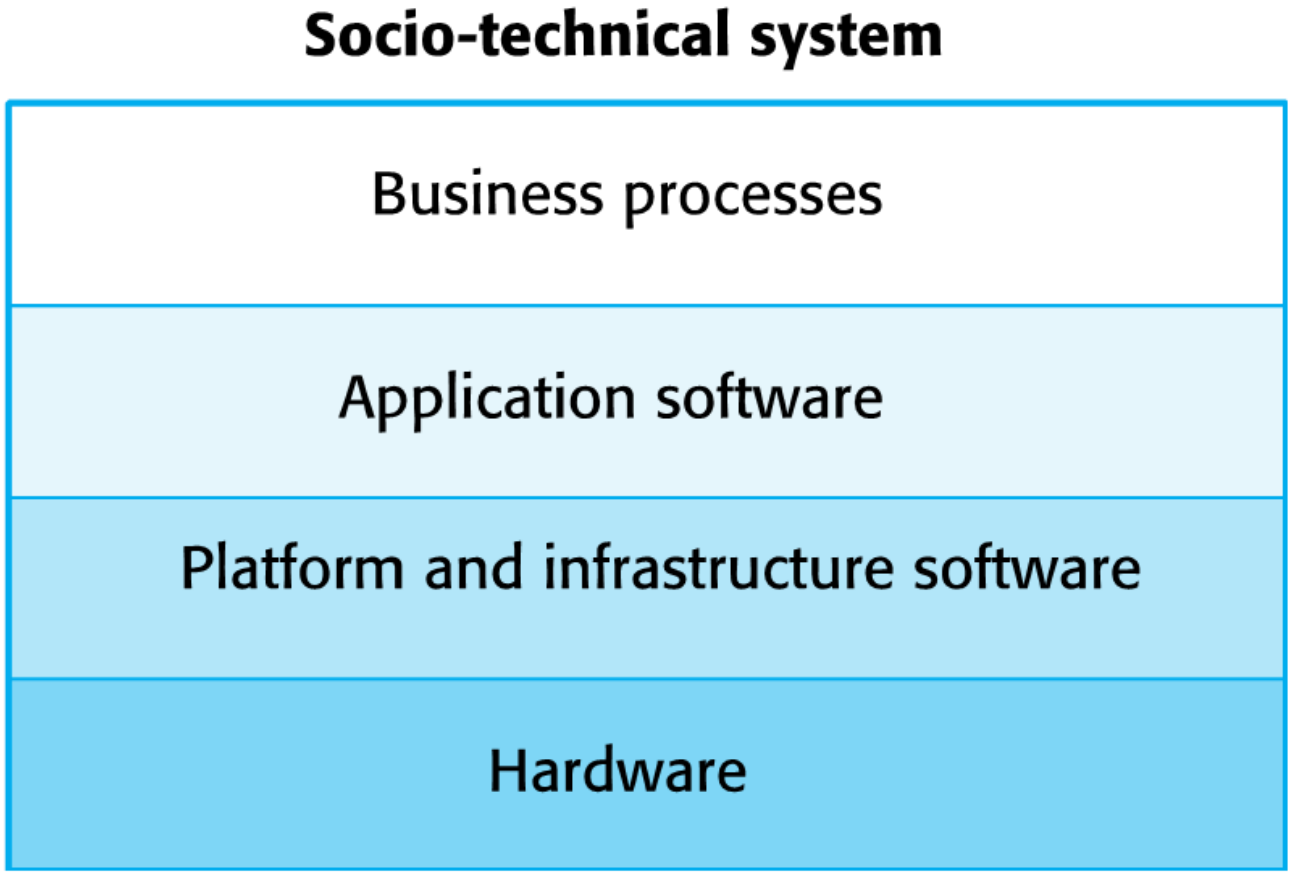
Legacy system components

- ระบบฮาร์ดแวร์
  - Legacy system อาจถูกเขียนขึ้นสำหรับฮาร์ดแวร์ที่ไม่สามารถใช้งานได้อีกต่อไป
- ซอฟต์แวร์ระบบสนับสนุน
  - Legacy system อาจมีซอฟต์แวร์สนับสนุนจำนวนมากที่ล้าสมัยหรือไม่ได้รับการสนับสนุนแล้ว
- ซอฟต์แวร์แอพพลิเคชัน
  - ระบบแอ็พพลิเคชันที่ให้บริการทางธุรกิจมักประกอบด้วยโปรแกรมประยุกต์จำนวนมาก
- ข้อมูลแอ็พพลิเคชัน
  - ข้อมูลเหล่านี้เป็นข้อมูลที่ประมวลผลโดยระบบแอ็พพลิเคชัน อาจไม่อัพเดท ซ้ำซ้อน หรือเก็บไว้ในฐานข้อมูลอื่น ๆ ที่เข้าถึงได้โดยเทคโนโลยีเก่า

Legacy system components

- กระบวนการทางธุรกิจ
  - เป็นกระบวนการที่ใช้ในธุรกิจเพื่อให้บรรลุวัตถุประสงค์ทางธุรกิจบางอย่าง
  - Legacy system อาจได้รับการออกแบบมาให้ครอบคลุมระบบเดิมและมีฟังก์ชันการทำงานที่จำกัด
- นโยบายและกฎเกณฑ์ทางธุรกิจ
  - เป็นคำจำกัดความของการดำเนินธุรกิจและข้อจำกัดในการดำเนินธุรกิจ
  - Legacy system อาจอยู่ภายใต้นโยบายและกฎเหล่านั้น

Legacy system layers



Legacy system replacement

- การแทนที่ Legacy system มีความเสี่ยงสูง แต่ก็มีราคาแพงเมื่อจะให้ธุรกิจยังคงใช้ Legacy system ต่อไป
- การเปลี่ยนระบบมีความเสี่ยงจากหลายสาเหตุ
  - ขาดข้อกำหนดระบบที่สมบูรณ์
  - การผนวกรวมระบบเดิมกับกระบวนการทางธุรกิจอย่างแน่นแฟ้น
  - ไม่มีการบันทึกกฎเกณฑ์ทางธุรกิจไว้ในระบบเดิม
  - การพัฒนาซอฟต์แวร์ใหม่อาจล่าช้าและ/หรือเกินงบประมาณ

Legacy system change

- การแทนที่ Legacy system มีต้นทุนสูงด้วยเหตุผลหลายประการ:
  - รูปแบบการเขียนโปรแกรมที่ไม่สอดคล้องกันโดยนักพัฒนารุ่นเก่า ๆ
  - การใช้ภาษาโปรแกรมที่ล้าสมัย นักพัฒนาจำนวนน้อยที่สามารถใช้ภาษาเหล่านั้นได้
  - เอกสารระบบไม่เพียงพอ
  - ระบบเดิมไม่มีโครงสร้างที่ดี
  - การเพิ่มประสิทธิภาพของโปรแกรม (Program optimizations) ในระบบเดิม อาจยากที่จะทำความเข้าใจ
  - ข้อผิดพลาดของข้อมูล การซ้ำซ้อน และการไม่สอดคล้องกันของข้อมูล

Legacy system management

- องค์กรที่ใช้ Legacy system ต้องเลือกกลยุทธ์สำหรับการพัฒนาระบบ ซึ่งอาจจะเป็นได้ลายวิธี เช่น
  - บำรุงรักษา Legacy system เดิมต่อไป
  - ปรับเปลี่ยนกระบวนการทางธุรกิจ เพื่อไม่จำเป็นต้องใช้ Legacy system อีกต่อไป
  - เปลี่ยนระบบโดย re-engineering เพื่อปรับปรุงการบำรุงรักษา
  - แทนที่ Legacy system ด้วยระบบใหม่ทั้งหมด
- กลยุทธ์ที่เลือกจะขึ้นอยู่กับคุณภาพของระบบและมูลค่าทางธุรกิจ

Legacy system categories

- คุณภาพต่ำและมีมูลค่าทางธุรกิจต่ำ (Low quality, low business value)
  - ระบบเหล่านี้ควรถูกทิ้ง
- คุณภาพต่ำและมีมูลค่าทางธุรกิจสูง (Low-quality, high-business value)
  - ระบบเหล่านี้ มีความสำคัญต่อธุรกิจ แต่มีราคาแพงในการบำรุงรักษา ควรมีการออกแบบใหม่หรือเปลี่ยนใหม่หากมีระบบที่เหมาะสม
- มูลค่าที่มีคุณภาพสูงและมีมูลค่าทางธุรกิจต่ำ (High-quality, low-business value)
  - แทนที่ด้วย COTS, สร้างใหม่ทั้งหมด, หรือบำรุงรักษา
- มูลค่าทางธุรกิจที่สูงและมีคุณภาพสูง (High-quality, high business value)
  - ดำเนินการต่อโดยใช้การบำรุงรักษาระบบตามปกติ

Business value assessment

- การประเมินควรทำจากหลากหลายมุมมอง
  - ผู้ใช้ปลายทางระบบ (System end-users)
  - ลูกค้าธุรกิจ (Business customers)
  - ผู้จัดการสายงาน (Line managers)
  - ผู้จัดการฝ่ายไอที (IT managers)
  - ผู้จัดการอาวุโส (Senior managers)
- สัมภาษณ์ผู้มีส่วนได้เสียที่แตกต่างกันและเปรียบเทียบผลลัพธ์

Issues in business value assessment

- การใช้ระบบ
  - หากระบบถูกใช้งานเป็นครั้งคราวหรือโดยคนจำนวนน้อย ระบบนั้นอาจมีมูลค่าทางธุรกิจต่ำ
- กระบวนการทางธุรกิจที่ได้รับการสนับสนุน
  - ระบบอาจมีมูลค่าทางธุรกิจต่ำถ้าถูกนำไปใช้ในกระบวนการทางธุรกิจที่ไม่มีประสิทธิภาพ
- ความน่าเชื่อถือของระบบ
  - หากระบบไม่น่าเชื่อถือและมีปัญหาที่ส่งผลกระทบโดยตรงต่อลูกค้า ระบบมีมูลค่าทางธุรกิจต่ำ
- ผลลัพธ์ของระบบ
  - หากผลการดำเนินธุรกิจขึ้นอยู่กับผลลัพธ์ของระบบแล้ว ระบบนั้นจะมีมูลค่าทางธุรกิจสูง

System quality assessment

- การประเมินกระบวนการทางธุรกิจ
  - กระบวนการทางธุรกิจสนับสนุนเป้าหมายปัจจุบันของธุรกิจได้ดีเพียงใด?
- การประเมินสิ่งแวดล้อม
  - ระบบมีประสิทธิภาพในสภาพแวดล้อมทางธุรกิจนั้นดีเพียงใด และมีค่าใช้จ่ายมากน้อยเพียงใดในการบำรุงรักษา?
- การประเมินแอพพลิเคชั่น
  - คุณภาพของระบบซอฟต์แวร์แอพพลิเคชันคืออะไร?

Business process assessment

- ใช้แนวทางแสวงหาคำตอบจากผู้มีส่วนได้เสียของระบบ
  - มีรูปแบบกระบวนการที่กำหนดไว้อย่างชัดเจนและดำเนินตามอย่างเคร่งครัด?
  - ส่วนต่าง ๆ ขององค์กรใช้กระบวนการที่แตกต่างกันสำหรับงานเดียวกันหรือไม่?
  - กระบวนการได้รับการปรับใช้ให้หมาะสมกับงานอย่างไร?
  - อะไรคือความสัมพันธ์กับกระบวนการทางธุรกิจอื่น ๆ และกระบวนการนี้มีความจำเป็นหรือไม่?
  - กระบวนการนี้ได้รับการสนับสนุนอย่างมีประสิทธิภาพจากซอฟต์แวร์แอ็พพลิเคชันแบบเดิมหรือไม่?
- ตัวอย่าง – ระบบการจองตั๋วที่สถานีอาจมีผลตอบแทนทางธุรกิจที่ต่ำ เนื่องจากมีการจองตั๋วออนไลน์มากขึ้น, จำเป็นแค่ไหน ที่ต้องพัฒนาระบบจองตั๋วแบบเดิม?

Factors used in environment assessment

Factor	Questions
Supplier stability	Is the supplier still in existence? Is the supplier financially stable and likely to continue in existence? If the supplier is no longer in business, does someone else maintain the systems?
Failure rate	Does the hardware have a high rate of reported failures? Does the support software crash and force system restarts?
Age	How old is the hardware and software? The older the hardware and support software, the more obsolete it will be. It may still function correctly but there could be significant economic and business benefits to moving to a more modern system.
Performance	Is the performance of the system adequate? Do performance problems have a significant effect on system users?

Factors used in environment assessment

Factor	Questions
Support requirements	What local support is required by the hardware and software? If there are high costs associated with this support, it may be worth considering system replacement.
Maintenance costs	What are the costs of hardware maintenance and support software licences? Older hardware may have higher maintenance costs than modern systems. Support software may have high annual licensing costs.
Interoperability	Are there problems interfacing the system to other systems? Can compilers, for example, be used with current versions of the operating system? Is hardware emulation required?

Factors used in application assessment

Factor	Questions
Understandability	How difficult is it to understand the source code of the current system? How complex are the control structures that are used? Do variables have meaningful names that reflect their function?
Documentation	What system documentation is available? Is the documentation complete, consistent, and current?
Data	Is there an explicit data model for the system? To what extent is data duplicated across files? Is the data used by the system up to date and consistent?
Performance	Is the performance of the application adequate? Do performance problems have a significant effect on system users?

Factors used in application assessment

Factor	Questions
Programming language	Are modern compilers available for the programming language used to develop the system? Is the programming language still used for new system development?
Configuration management	Are all versions of all parts of the system managed by a configuration management system? Is there an explicit description of the versions of components that are used in the current system?
Test data	Does test data for the system exist? Is there a record of regression tests carried out when new features have been added to the system?
Personnel skills	Are there people available who have the skills to maintain the application? Are there people available who have experience with the system?

Software maintenance

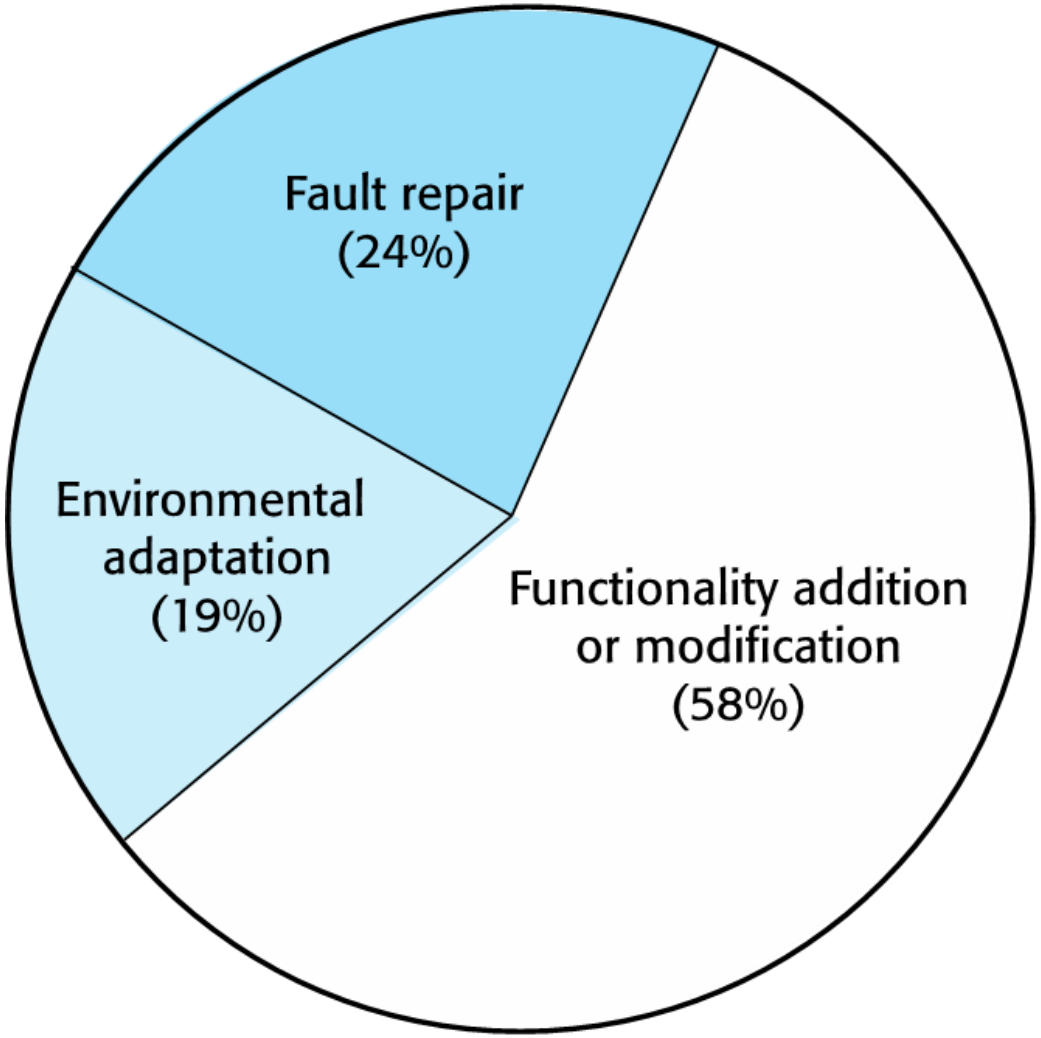
Software maintenance

- คือการแก้ไขโปรแกรมหลังจากการนำไปใช้งาน
- คำนี้ส่วนใหญ่ใช้สำหรับการเปลี่ยนซอฟต์แวร์ที่กำหนดโดยผู้ใช้ (custom software)
  - ส่วนกรณีซอฟต์แวร์ทั่วไป (Generic software) มักใช้เมื่อสร้างเวอร์ชันใหม่ ๆ
- การบำรุงรักษาจะเกี่ยวข้องกับการเปลี่ยนแปลงที่สำคัญในสถาปัตยกรรมของระบบ
- การเปลี่ยนแปลงจะดำเนินการโดยการปรับเปลี่ยนคอมโพเนนต์ที่มีอยู่และเพิ่มส่วนประกอบใหม่ลงในระบบ

Types of maintenance

- ซ่อมแซมข้อบกพร่อง (Fault repairs)
  - การเปลี่ยนระบบเพื่อแก้ไขข้อบกพร่อง / ช่องโหว่และข้อบกพร่องเพื่อให้ถูกต้องตรงกับความต้องการของระบบ
- การปรับตัวให้เข้ากับสภาพแวดล้อม (Environmental adaptation)
  - การบำรุงรักษาเพื่อปรับซอฟต์แวร์ให้เข้ากับสภาพแวดล้อมการทำงานที่แตกต่างกัน
  - การเปลี่ยนระบบเพื่อให้ทำงานในสภาพแวดล้อมที่แตกต่างกัน (คอมพิวเตอร์ OS ฯลฯ)
- การเพิ่มฟังก์ชันและการแก้ไข (Functionality addition and modification)
  - การปรับเปลี่ยนระบบเพื่อตอบสนองความต้องการใหม่ ๆ

Maintenance effort distribution



Maintenance costs

- มักจะสูงกว่าต้นทุนการพัฒนา (2 เท่าถึง 100 เท่า ขึ้นอยู่กับแอพพลิเคชั่น)
- ได้รับผลกระทบทั้งจากปัจจัยด้านเทคนิคและไม่ใช่ทางเทคนิค
- จะเพิ่มขึ้นเรื่อย ๆ ตราบเท่าที่ซอฟต์แวร์ยังคงอยู่
  - การบำรุงรักษาที่กระทบกระเทือนโครงสร้างซอฟต์แวร์ อาจเป็นผลให้การบำรุงรักษาทำได้ยากขึ้น
- ซอฟต์แวร์ที่มีอายุมาก จะมีต้นทุนค่าใช้จ่ายในการสนับสนุนสูง
  - (เช่น ภาษาเก่า, คอมไพเลอร์ เป็นต้น)

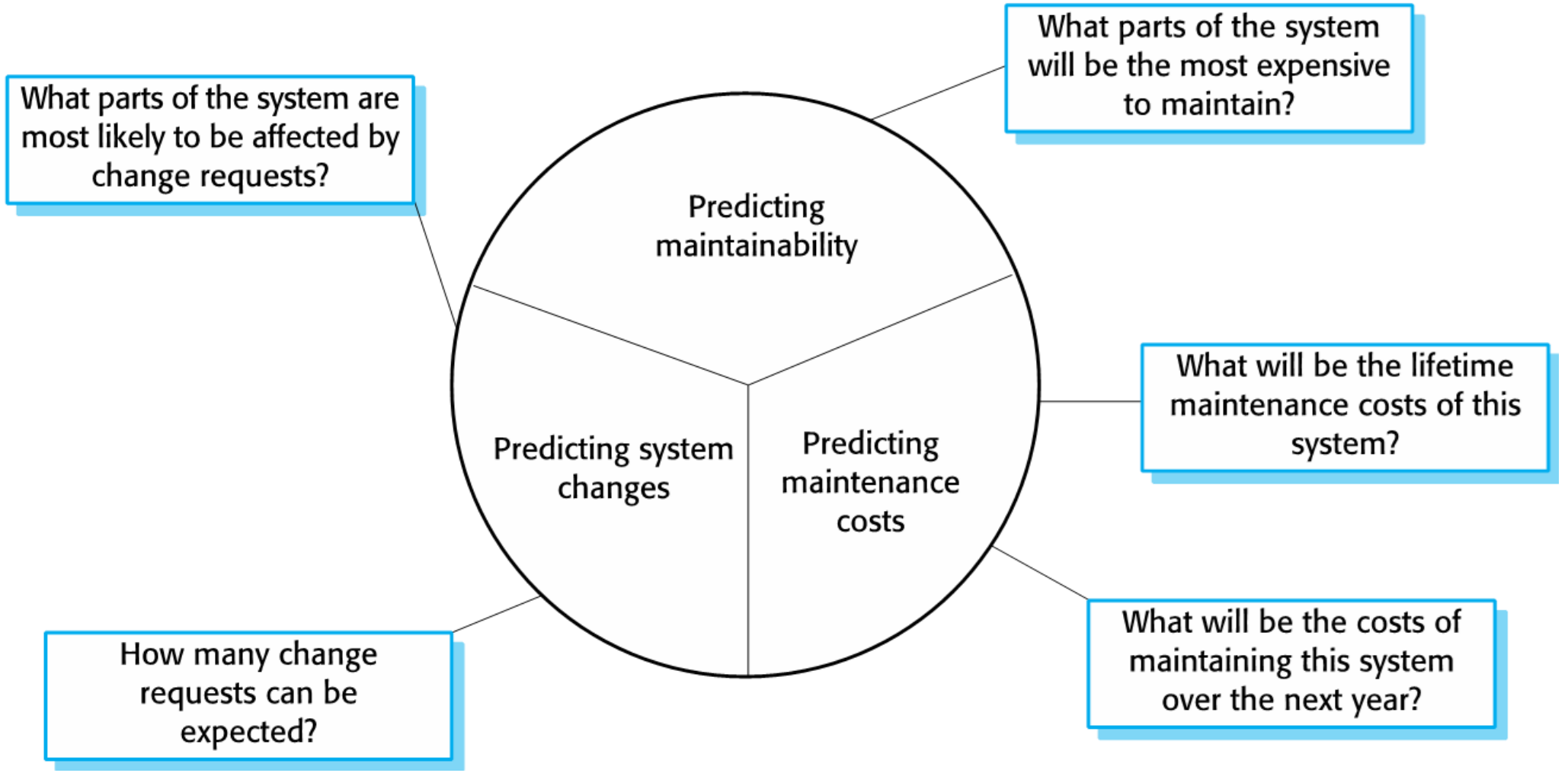
Maintenance costs

- การเพิ่มคุณสมบัติใหม่ ๆ ให้กับระบบในระหว่างการบำรุงรักษา มักจะมีราคาสูงกว่าการเพิ่มคุณสมบัติเดียวกันในระหว่างการพัฒนา
  - ทีมใหม่ต้องศึกษาและทำความเข้าใจโปรแกรมที่จะบำรุงรักษา
  - การแยกการบำรุงรักษาและการพัฒนา อาจทำให้ไม่มีการสร้างแรงจูงใจให้ทีมพัฒนาเขียนซอฟต์แวร์ที่สามารถดูแลรักษาได้
- การบำรุงรักษาโปรแกรมไม่เป็นที่นิยม
  - พนักงานซ่อมบำรุงมักไม่มีประสบการณ์และมีความรู้เกี่ยวกับโดเมนอย่างจำกัด
- เมื่ออายุของโปรแกรมมากขึ้น ผ่านการบำรุงรักษามายาวนาน อาจทำให้ความเป็นโครงสร้างของโปรแกรมแย่ลงและกลายเป็นเรื่องยากที่จะบำรุงรักษา

Maintenance prediction

- การพยากรณ์การบำรุงรักษาเกี่ยวข้องกับการประเมินว่าส่วนใดของระบบอาจทำให้เกิดปัญหาและมีค่าใช้จ่ายในการบำรุงรักษาสูง
  - การยอมรับการเปลี่ยนแปลงขึ้นอยู่กับการบำรุงรักษาส่วนประกอบที่ได้รับผลกระทบจากการเปลี่ยนแปลง
  - การเปลี่ยนแปลงจะทำให้ความเป็นระบบลดลงและลดความสามารถในการบำรุงรักษา
  - ค่าบำรุงรักษาขึ้นอยู่กับจำนวนของการเปลี่ยนแปลงและค่าใช้จ่ายในการเปลี่ยนแปลงขึ้นอยู่กับการบำรุงรักษา

Maintenance prediction



Change prediction

- หมายถึงการคาดการณ์จำนวนการเปลี่ยนแปลงที่ต้องการและการทำความเข้าใจความสัมพันธ์ระหว่างระบบกับสภาพแวดล้อม
- ระบบที่ยึดแน่นกับสภาพแวดล้อม ต้องมีการเปลี่ยนแปลงทุกครั้งที่มีการเปลี่ยนแปลงสภาพแวดล้อม
- ปัจจัยที่มีอิทธิพลต่อความสัมพันธ์ระหว่างระบบกับสภาพแวดล้อม ได้แก่
  - จำนวนและความซับซ้อนของอินเทอร์เฟซระบบ
  - จำนวนความต้องการของระบบที่เปลี่ยนแปลงไปตามธรรมชาติ
  - กระบวนการทางธุรกิจที่ใช้ระบบ

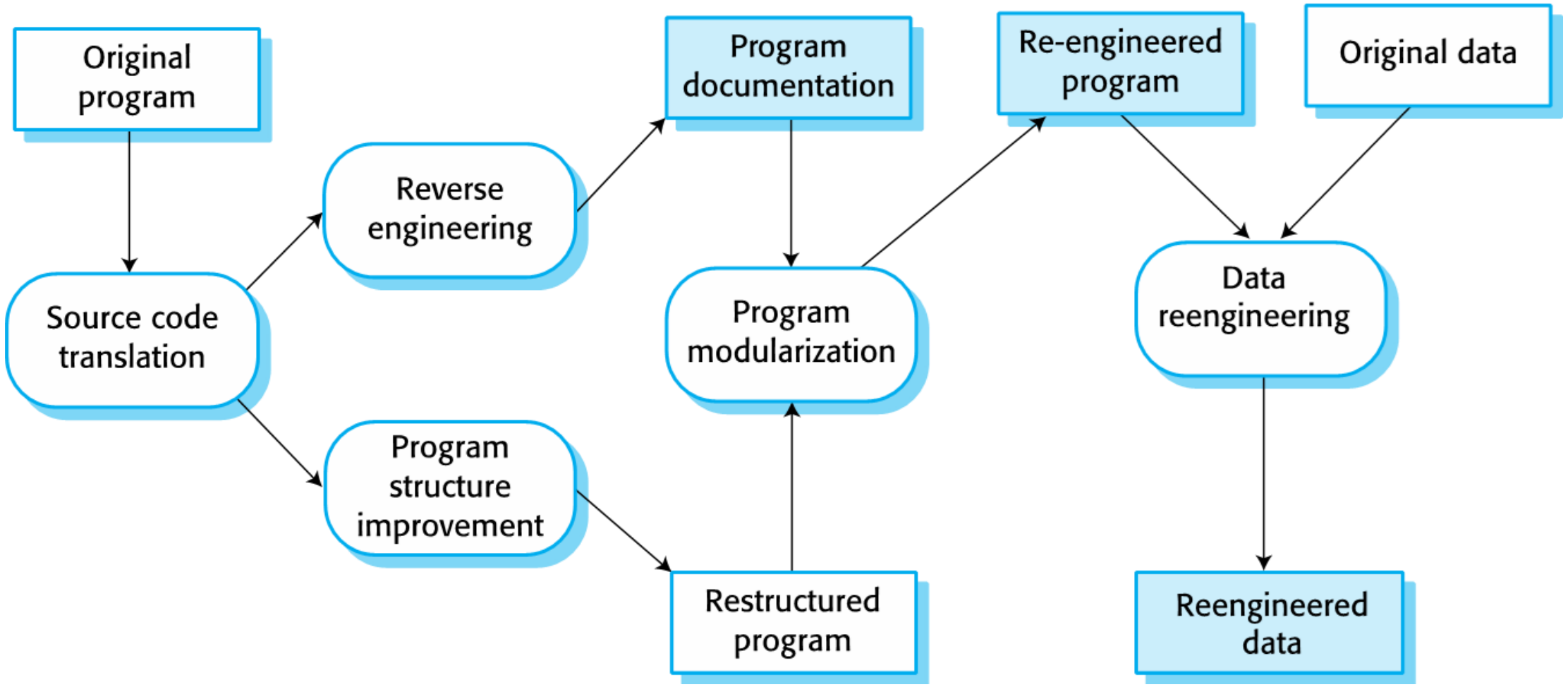
Software reengineering

- ปรับโครงสร้างหรือเขียนใหม่ (ในบางส่วนหรือทั้งหมดของระบบเก่า) โดยไม่ต้องเปลี่ยนฟังก์ชันการทำงาน
- ทำได้เมื่อบางส่วนของระบบขนาดใหญ่ต้องการการบำรุงรักษาบ่อย ๆ
- การรื้อปรับระบบช่วยเพิ่มความสะดวกในการบำรุงรักษา ระบบอาจได้รับการจัดโครงสร้างใหม่และจัดทำเอกสารใหม่

Advantages of reengineering

- ลดความเสี่ยง
  - มีความเสี่ยงสูงในการพัฒนาซอฟต์แวร์ใหม่
  - อาจมีปัญหาในการพัฒนา ปัญหาเกี่ยวกับพนักงาน และปัญหาทางเทคนิค
- ลดค่าใช้จ่าย
  - ค่าใช้จ่ายในการ reengineering มักจะน้อยกว่าค่าใช้จ่ายในการพัฒนาซอฟต์แวร์ใหม่

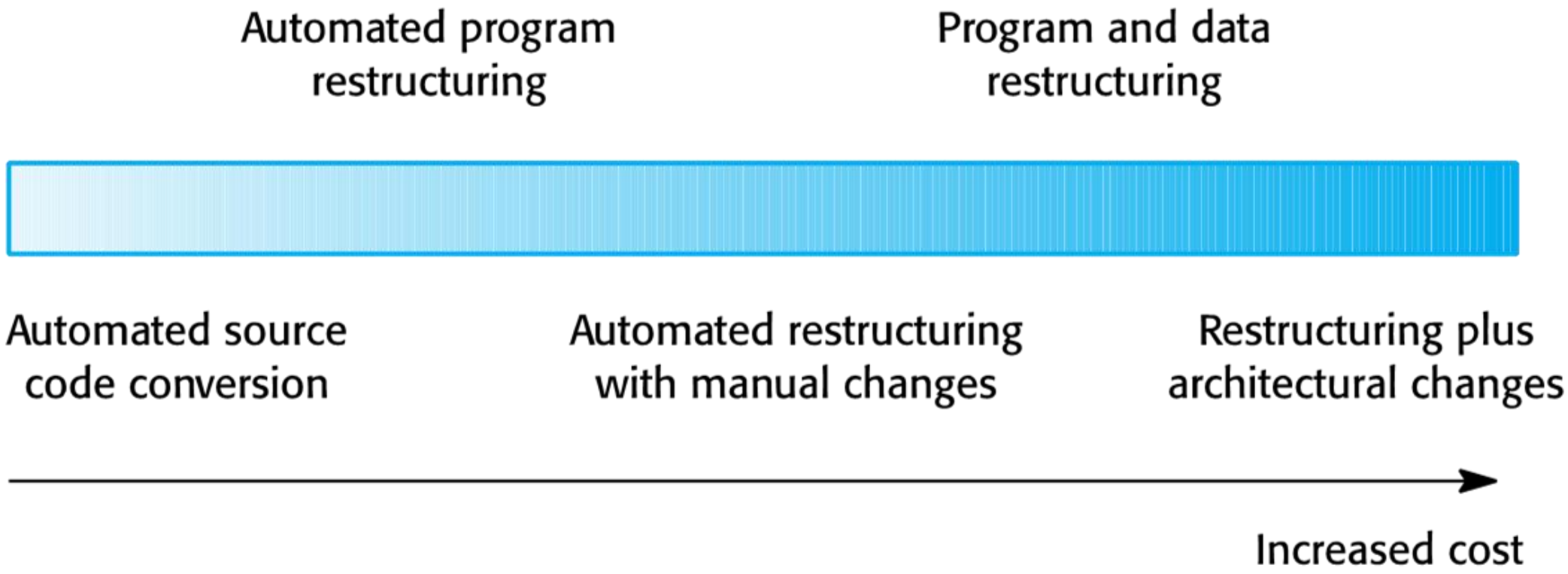
The reengineering process



Reengineering process activities

- การแปลรหัสต้นทาง (Source code translation)
  - แปลงรหัสเป็นภาษาใหม่
- วิศวกรรมย้อนกลับ (Reverse engineering)
  - วิเคราะห์โปรแกรมเพื่อทำความเข้าใจ
- การปรับปรุงโครงสร้างโปรแกรม (Program structure improvement)
  - ปรับโครงสร้างใหม่โดยอัตโนมัติเพื่อให้เข้าใจได้ง่าย
- จัดทำโครงสร้างโปรแกรม (Program modularization)
  - จัดโครงสร้างโปรแกรมใหม่
- การรื้อปรับระบบข้อมูล (Data reengineering)
  - ทำความสะอาดและปรับโครงสร้างข้อมูลระบบ

Reengineering approaches



Reengineering cost factors

- คุณภาพของซอฟต์แวร์ที่จะปรับรื้อ/ปรับระบบใหม่
- การสนับสนุนเครื่องมือสำหรับการปรับรื้อระบบ
- ขอบเขตของการแปลงข้อมูลที่จำเป็น
- ความพร้อมของเจ้าหน้าที่ผู้เชี่ยวชาญในการปรับรื้อระบบ
  - ปัญหานี้อาจเป็นปัญหากับระบบเก่าที่ใช้เทคโนโลยีซึ่งไม่มีการใช้กันอย่างแพร่หลาย

Refactoring

- การทำ Refactoring เป็นกระบวนการในการปรับปรุงโปรแกรมเพื่อชะลอการ degradation เนื่องจากการเปลี่ยนแปลง
- Refactoring เป็น 'การบำรุงรักษาเชิงป้องกัน' ที่ช่วยลดปัญหาของการเปลี่ยนแปลงในอนาคต
- Refactoring เกี่ยวข้องกับการปรับเปลี่ยนโปรแกรมเพื่อปรับปรุงโครงสร้างลดความซับซ้อนหรือทำให้เข้าใจง่ายขึ้น
- เมื่อทำ refactor ไม่ควรเพิ่มฟังก์ชันการทำงาน แต่มุ่งเน้นที่การปรับปรุงโปรแกรม

Refactoring and reengineering

- การออกแบบใหม่ (Re-engineering) เกิดขึ้นหลังจากระบบได้รับการบำรุงรักษาเป็นระยะเวลาหนึ่งและค่าใช้จ่ายในการบำรุงรักษาเพิ่มขึ้น
  - เราใช้เครื่องมืออัตโนมัติในการประมวลผลและ re-engineering ระบบเดิมเพื่อสร้างระบบใหม่ที่สามารถดูแลรักษาได้มากขึ้น
- การทำ Refactoring เป็นกระบวนการต่อเนื่องในการปรับปรุงตลอดกระบวนการพัฒนาและวิวัฒนาการ
  - มีจุดประสงค์เพื่อหลีกเลี่ยงการ degradation ของโครงสร้างโปรแกรม (รวมทั้ง code) ซึ่งจะเพิ่มต้นทุนและความยากลำบากในการบำรุงรักษาระบบ

‘Bad smells’ in program code

- รหัสที่ซ้ำกัน (Duplicate code)
  - รหัสเดียวกันหรือคล้ายกันมากอาจกระจัดกระจายอยู่ในตำแหน่งต่าง ๆ ของโปรแกรม ซึ่งสามารถนำออกและนำมาใช้เป็น method หรือฟังก์ชันเดียวกัน
- เมธอดที่ยาวเกินไป (Long methods)
  - หากเมธอดยาวเกินไปควรแยกออกเป็นเมธอดที่สั้นลง
- คำสั่ง Switch (case)
  - คำสั่ง switch อาจกระจายอยู่ทั่วโปรแกรม ในภาษาเชิงวัตถุมักจะสามารถใช้ polymorphism เพื่อตอบสนองสิ่งเดียวกันนี้

‘Bad smells’ in program code

- ข้อมูลรวมกันเป็นก้อน ๆ (Data clumping)
  - กลุ่มข้อมูลเดียวกันอาจปรากฏครั้งในหลาย ๆ แห่งในโปรแกรม
  - กลุ่มข้อมูลเหล่านี้มักจะถูกแทนที่ด้วยวัตถุที่ encapsulates ข้อมูลทั้งหมด
- การคาดเดา (Speculative generality)
  - กรณีนี้เกิดขึ้นเมื่อนักพัฒนาซอฟต์แวร์เขียนโค้ดเผื่อไว้สำหรับความต้องการในอนาคต กรณีนี้มักจะสามารถลบออกไปได้

Key points

- การพัฒนาซอฟต์แวร์และวิวัฒนาการอาจถือได้ว่าเป็นการรวมกระบวนการทำซ้ำที่สามารถนำมาพัฒนาเป็นแบบ spiral model ได้
- สำหรับระบบที่กำหนดเอง (custom systems) ค่าใช้จ่ายในการบำรุงรักษาซอฟต์แวร์จะสูงกว่าค่าใช้จ่ายในการพัฒนาซอฟต์แวร์
- กระบวนการของวิวัฒนาการของซอฟต์แวร์ถูกขับเคลื่อนโดยการร้องขอ (requests for changes) รวมถึงการวิเคราะห์ผลกระทบการเปลี่ยนแปลง (change impact analysis) การวางแผนปล่อยซอฟต์แวร์ (release planning) และการดำเนินการเปลี่ยนแปลง (change implementation)

Key points

- ระบบเดิม (Legacy systems) เป็นระบบซอฟต์แวร์รุ่นเก่า ที่พัฒนาขึ้นโดยใช้เทคโนโลยีซอฟต์แวร์และซอฟต์แวร์ที่ล้าสมัย แต่ยังคงมีประโยชน์สำหรับธุรกิจ
- การรักษาระบบเดิมมักจะถูกกว่าและมีความเสี่ยงน้อยกว่าที่จะพัฒนาระบบทดแทนโดยใช้เทคโนโลยีที่ทันสมัย
- ควรประเมินมูลค่าธุรกิจของระบบเดิมและคุณภาพของแอปพลิเคชันเพื่อช่วยในการตัดสินใจว่าจะเปลี่ยนระบบ (replace) ดัดแปลง (transform) หรือบำรุงรักษา (maintain) ระบบเดิมต่อไปหรือไม่
- มีการบำรุงรักษาซอฟต์แวร์ 3 ประเภทคือ
  - การแก้ไขข้อบกพร่อง
  - การปรับเปลี่ยนซอฟต์แวร์เพื่อทำงานในสภาพแวดล้อมใหม่
  - การพัฒนาตามข้อกำหนดใหม่หรือความต้องการเปลี่ยนแปลง

Key points

- การ re-engineering ซอฟต์แวร์เป็นเรื่องเกี่ยวกับการจัดโครงสร้างใหม่และการทำเอกสารใหม่
  - เพื่อให้ทำความเข้าใจและดำเนินการเปลี่ยนแปลงได้ง่ายขึ้น
- Refactoring ทำให้การเปลี่ยนแปลงโปรแกรมที่รักษาฟังก์ชันการทำงาน
  - เป็นรูปแบบของการบำรุงรักษาเชิงป้องกัน

คำถาม???