System Modeling

Week 06

ดัดแปลงจาก slides ของหนังสือ Software Engineering [1]

หัวข้อที่จะศึกษา

- Context models
- Interaction models
- Structural models
- Behavioral models
- Model-driven engineering

System modeling

- การสร้างแบบจำลองระบบเป็นกระบวนการในการพัฒนานิยาม (abstract) ของระบบ โดยแบบจำลองแต่ละชนิดจะนำเสนอมุมมองที่แตกต่างกันของระบบ
- ในปัจจุบัน การสร้างแบบจำลองของระบบ มักจะหมายถึงการให้นิยามระบบโดยใช้สัญลักษณ์ทางกราฟิก ซึ่งรูปแบบที่นิยมใช้คือ Unified Modeling Language (UML)
- การสร้างแบบจำลองระบบช่วยให้นักวิเคราะห์เข้าใจถึงฟังก์ชันการทำงานของระบบ และสามารถแบบจำลองนี้เพื่อสื่อสารกับลูกค้า

Existing and planned system models

- แบบจำลองของระบบที่มีอยู่แล้ว จะถูกใช้ในช่วง requirements engineering
  - ใช้เพื่ออธิบายสิ่งที่ระบบที่มีอยู่
  - สามารถใช้เป็นพื้นฐานสำหรับการวิเคราะห์จุดแข็งและจุดอ่อน
  - นำไปสู่ความต้องการสำหรับระบบใหม่
- แบบจำลองของระบบใหม่ จะถูกใช้ในช่วง requirements engineering
  - เพื่อช่วยอธิบายความต้องการที่เสนอต่อผู้มีส่วนได้เสียของระบบ
  - ใช้เพื่อแลกเปลี่ยนความคิดในการเขียน proposal
  - ใช้เพื่อจัดทำเอกสารระบบสำหรับการ implementation ระบบ
- ใน model-driven engineering process เราสามารถใช้แบบจำลองเพื่อสร้างบางส่วนหรือทั้งหมดของระบบที่สมบูรณ์ ได้จากแบบจำลอง
  - ถ้าเขียนแบบจำลองได้ดีพอ อาจจะใช้ CASE tools เพื่อลดเวลา coding

System perspectives

- มุมมองจากภายนอก (external perspective)
  - ใช้เพื่อจำลองบริบทหรือสภาพแวดล้อมของระบบ
- มุมมองปฏิสัมพันธ์ (interaction perspective)
  - เป็นแบบจำลองการโต้ตอบระหว่างระบบกับบสภาพแวดล้อม
  - หรือการโต้ตอบระหว่างส่วนประกอบของระบบ
- มุมมองโครงสร้าง (structural perspective)
  - ใช้เพื่ออธิบาย organization ของระบบ
  - ใช้เพื่ออธิบายโครงสร้างข้อมูลที่ประมวลผลโดยระบบ
- มุมมองพฤติกรรม (behavioral perspective)
  - เป็นแบบจำลองพฤติกรรมแบบ dynamic ของระบบ
  - อธิบายวิธีตอบสนองต่อเหตุการณ์

UML diagram types

- แผนภาพกิจกรรม (Activity diagrams)
  - แสดงกิจกรรมที่เกี่ยวข้องในกระบวนการหรือในการประมวลผลข้อมูล
- แผนภาพ use case (Use case diagrams)
  - แสดงปฏิสัมพันธ์ระหว่างระบบและสภาพแวดล้อม
- แผนภาพลำดับงาน (Sequence diagrams)
  - แสดงการโต้ตอบระหว่างผู้ใช้ (actor) กับระบบ
  - แสดงการโต้ตอบระหว่างส่วนประกอบของระบบ
- แผนภาพคลาส (Class diagrams)
  - แสดงคลาสของวัตถุในระบบและความสัมพันธ์ระหว่างคลาส
- แผนภาพสถานะ (State diagrams)
  - แสดงให้เห็นว่าระบบมีการตอบสนองต่อเหตุการณ์ภายในและภายนอกอย่างไร

Use of graphical models

- อำนวยความสะดวกในการสนทนา เกี่ยวกับระบบที่มีอยู่หรือที่นำเสนอ
  - โมเดลที่ยังไม่สมบูรณ์หรือไม่ถูกต้องมากนัก ก็สามารถนำมาใช้ได้ เนื่องจากเราใช้มันเพื่อการ discuss เกี่ยวกับระบบ
  - อย่าเพิ่งเสียเวลามากเกินไป เพื่อที่จะทำให้มันสมบูรณ์แบบตั้งแต่ตอนแรก
- ช่วยจดจำหรือจัดการเอกสารเกี่ยวกับระบบที่มีอยู่
  - แบบจำลองควรเป็นตัวแทนที่ถูกต้องของระบบ แต่ไม่จำเป็นต้องสมบูรณ์
- เป็นรายละเอียดเกี่ยวกับระบบ ที่สามารถใช้ในการสร้างระบบได้จริง
  - แต่สุดท้ายแล้ว แบบจำลองจะต้องถูกต้องและสมบูรณ์

Context models

แบบจำลองบริบท

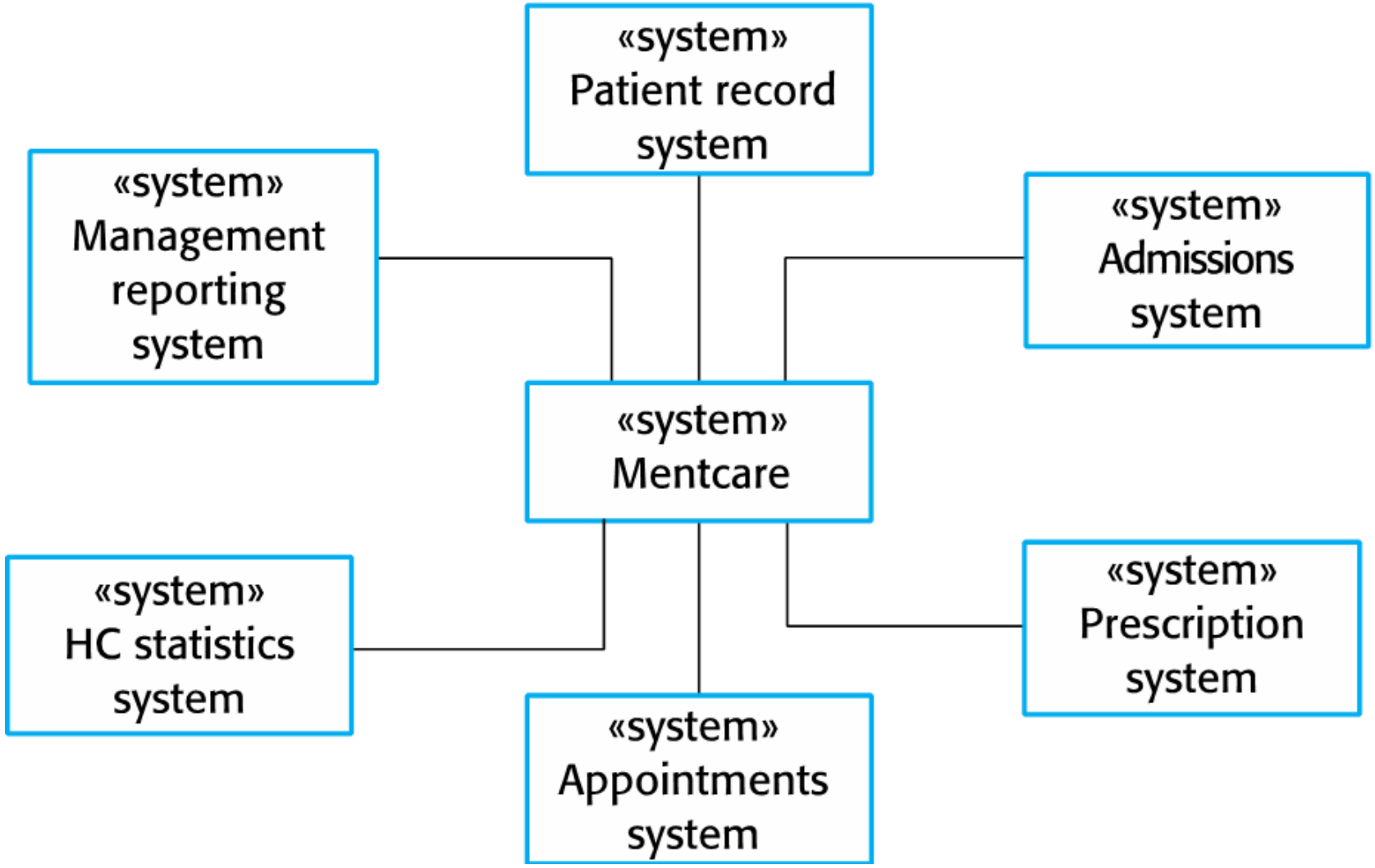
แบบจำลองบริบท (Context models)

- แบบจำลองบริบท ใช้เพื่อแสดงบริบทในการดำเนินงานของระบบ
  - แสดงถึงสิ่งที่อยู่นอกขอบเขตของระบบ
- การคำนึงถึงองค์กรและสังคมภายนอก อาจส่งผลต่อการตัดสินใจว่าควรวางตำแหน่งขอบเขตของระบบไว้ที่ใด
- แบบจำลองที่แสดงถึงระบบและความสัมพันธ์กับระบบอื่น ๆ เรียกว่า แบบจำลองทางสถาปัตยกรรม (Architectural models)

System boundaries

- เราต้องทำการกำหนดขอบเขตของระบบเสมอ
  - เพื่อกำหนดสิ่งที่อยู่ภายในและภายนอกขอบเขตของระบบ
  - เพื่อแสดงถึงระบบอื่น ๆ ที่ใช้ หรือเกี่ยวข้อง หรือขึ้นอยู่กับระบบที่กำลังพัฒนา
- การกำหนดขอบเขตระบบมีผลอย่างมากต่อความต้องการของระบบ
- การกำหนดขอบเขตของระบบคือการตัดสินใจที่อาจส่งผลกระทบต่อหลาย ๆ อย่างหรือได้รับผลจากปัจจัยหลาย ๆ อย่าง
  - อาจจะไปเพิ่มหรือลดภาระงานในส่วนอื่น ๆ ขององค์กร
  - อาจมีแรงกดดันจากภายนอก ที่ส่งผลต่อการกำหนดขอบเขตของระบบ

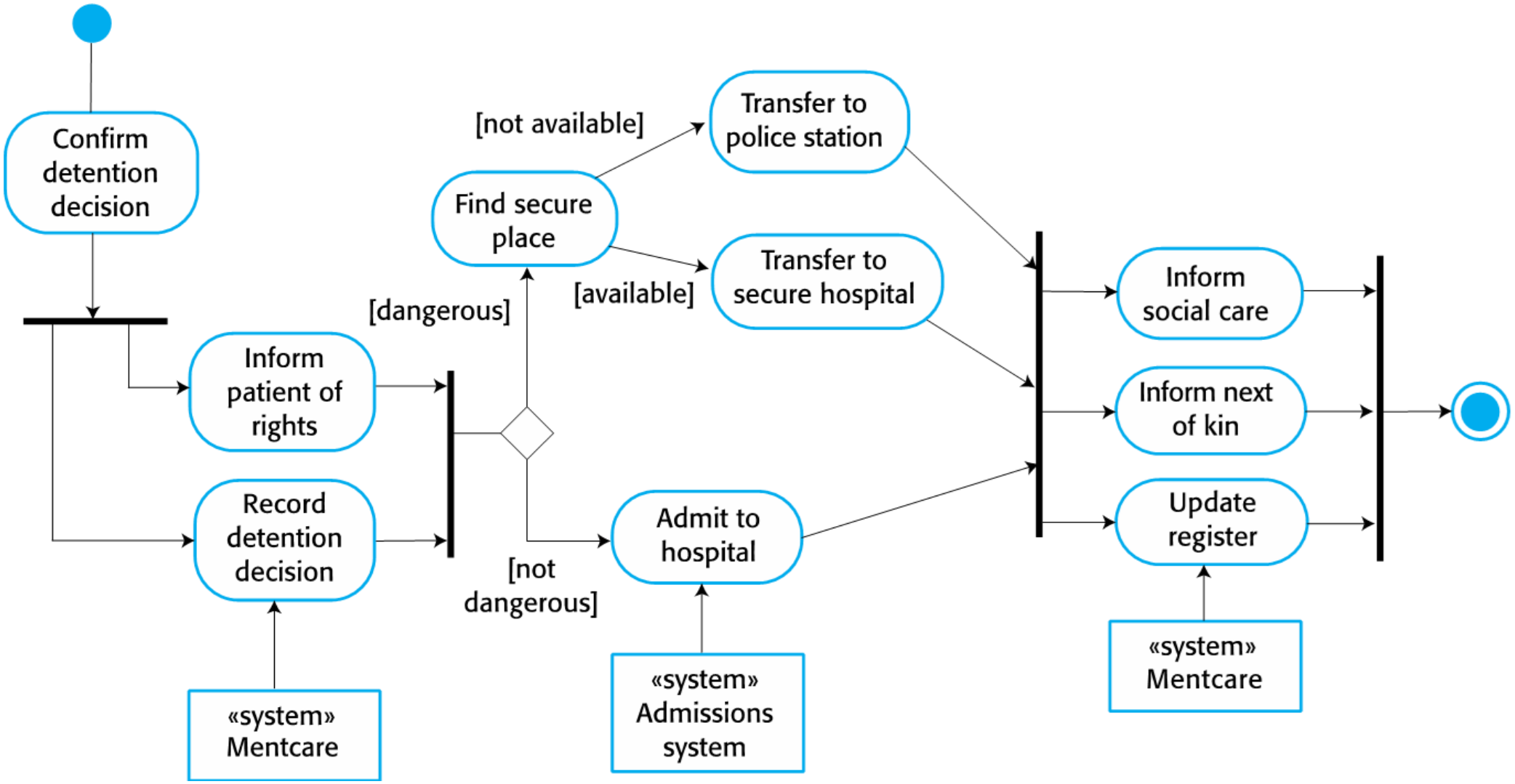
ตัวอย่าง The context of the Mentcare system



Process perspective

- แบบจำลองบริบท (context model) เป็นแบบจำลองที่เผยให้เห็นถึงสภาพแวดล้อมของระบบข้างเคียง
  - ไม่ได้ใช้เพื่อแสดงถึงวิธีการพัฒนาระบบที่ใช้ในสภาพแวดล้อมนั้น
- เป็นการแสดงให้เห็นว่า ระบบที่พัฒนาขึ้นนั้น จะถูกนำไปใช้ในกระบวนการทางธุรกิจที่กว้างขึ้นอย่างไร
- ในการกำหนดโมเดลกระบวนการทางธุรกิจ อาจนำเสนอโดยแผนภาพกิจกรรม UML

ตัวอย่าง Process model of involuntary detention



Interaction models

แบบจำลองโต้ตอบ หรือ แบบจำลองปฏิสัมพันธ์

Interaction models

- การสร้างแบบจำลองการโต้ตอบของผู้ใช้มีความสำคัญ
  - ช่วยในการระบุความต้องการของผู้ใช้
- การโต้ตอบระหว่างระบบกับระบบจำลอง จะเน้นถึงปัญหาการสื่อสารที่อาจเกิดขึ้น
- การจำลองการโต้ตอบ จะช่วยให้เราคาดการณ์ถึงประสิทธิภาพและความเชื่อถือได้ของระบบที่เสนอ
- เราสามารถใช้แผนภาพ use case และแผนภาพ sequence diagram สำหรับการสร้างแบบจำลองการโต้ตอบ

แบบจำลอง Use case

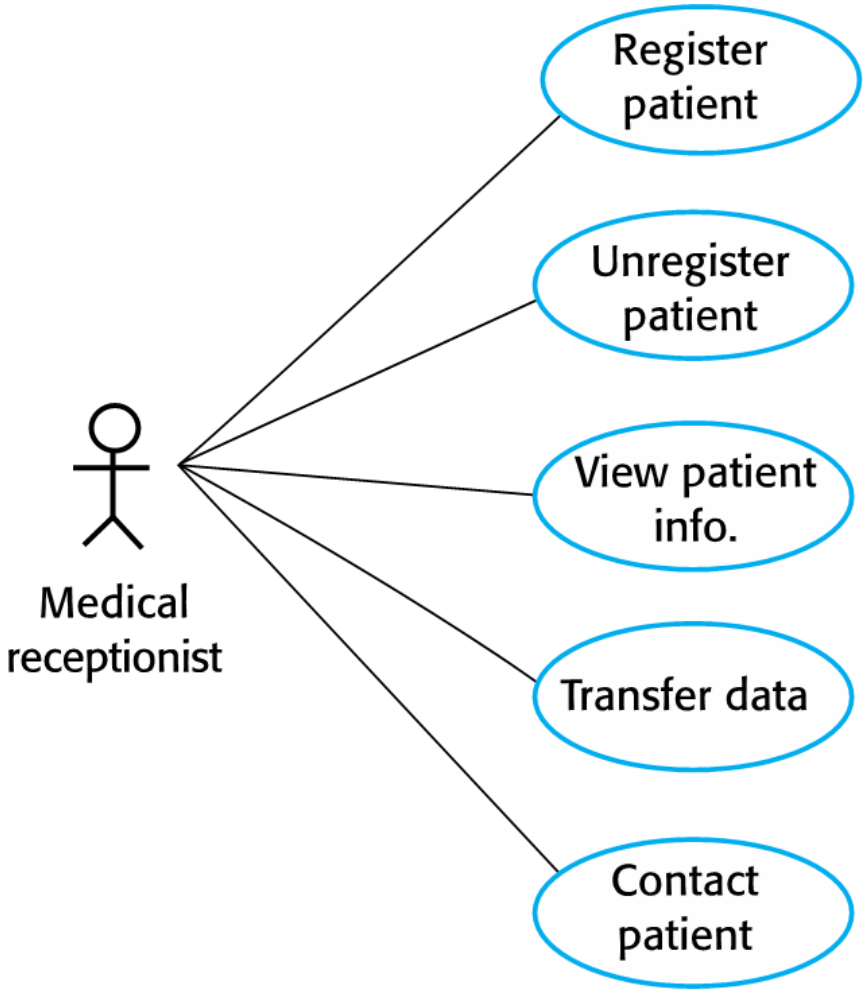
- แบบจำลอง use case ได้รับการพัฒนาขึ้นเพื่อรองรับการกระตุ้นความต้องการ (requirements elicitation) และเป็นส่วนหนึ่งของ UML
- แต่ละ use case แสดงถึงงานที่เป็นอิสระจากกัน และเน้นที่การโต้ตอบกับระบบภายนอก
- Actor ใน use case อาจเป็นบุคคลหรือระบบอื่น ๆ ก็ได้
- การอธิบายด้วย use case ให้ความหมายมากกว่าการอธิบายในรูปแบบบรรยายด้วยตัวอักษร

ตัวอย่าง : Transfer-data use case

‘Transfer data’ use-case ในรูปแบบตาราง

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient’s diagnosis and treatment.
Data	Patient’s personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

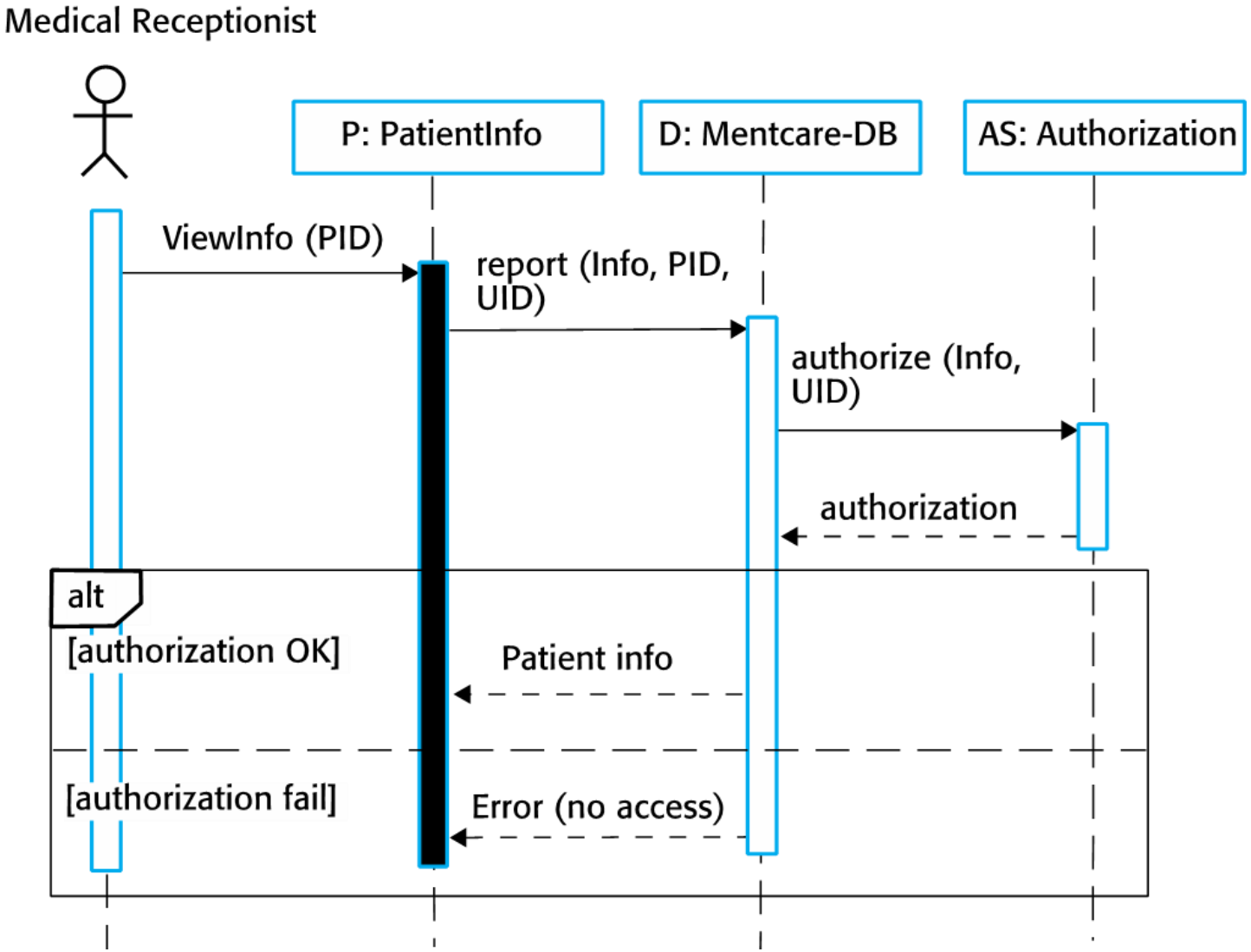
ตัวอย่าง : use case ที่เกี่ยวข้องกับ Medical receptionist

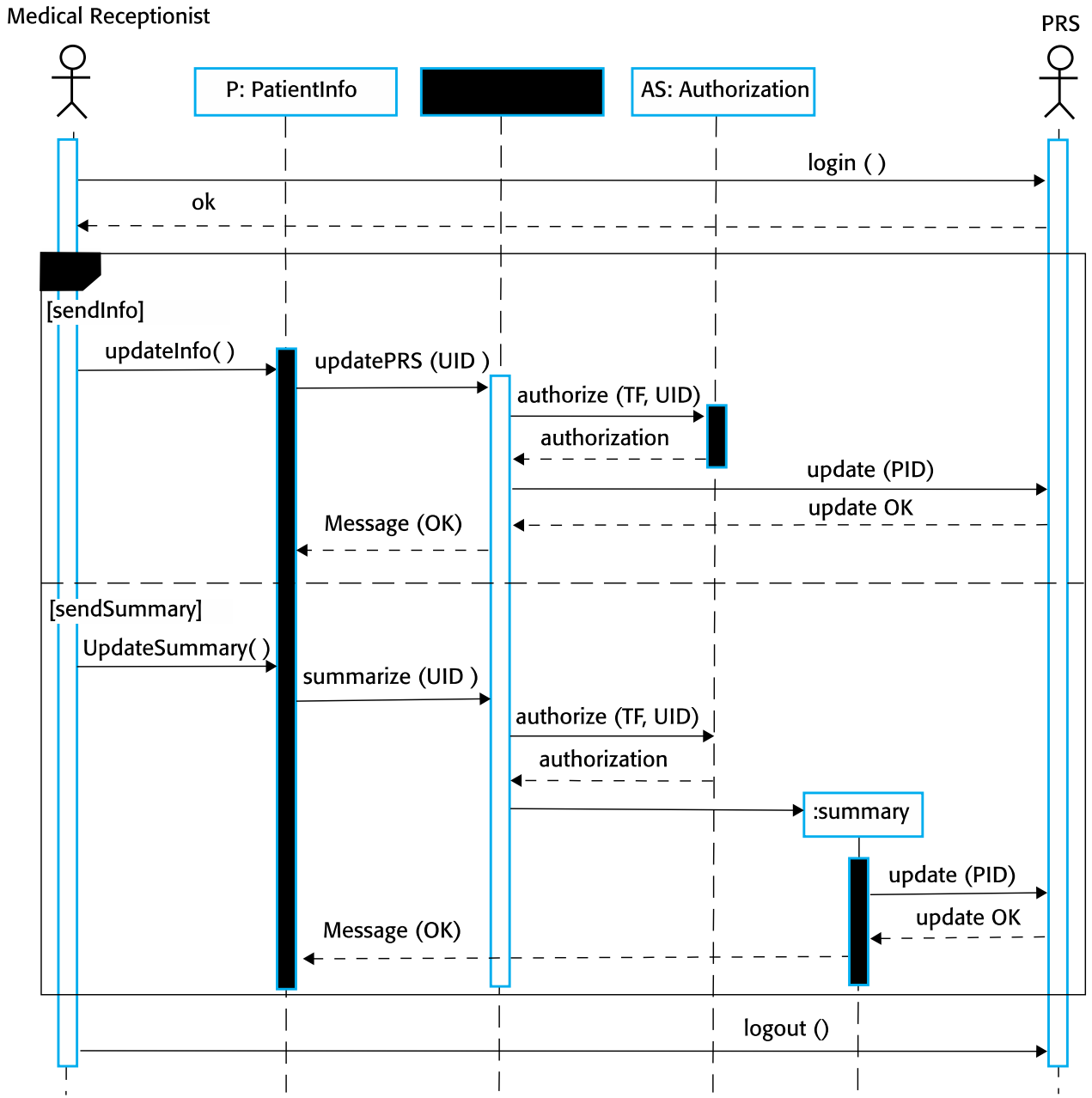


Sequence diagrams

- แผนภาพลำดับ (sequence diagrams) เป็นส่วนหนึ่งของ UML
  - ใช้เพื่อจำลองการโต้ตอบระหว่าง actor และวัตถุภายในระบบ
- แผนภาพลำดับจะแสดงลำดับการโต้ตอบที่เกิดขึ้นระหว่าง use case
- วัตถุและ actor ที่เกี่ยวข้องจะระบุไว้ด้านบนของแผนภาพ ตามด้วยเส้นประที่วาดในแนวตั้ง
  - เส้นประแสดงถึงเหตุการณ์ที่เกิดขึ้นตามเวลา จากบนลงล่าง
- การโต้ตอบระหว่างวัตถุจะแสดงโดยลูกศรที่ใส่คำอธิบายกำกับไว้

Sequence diagram for View patient information





Sequence diagram for Transfer Data

Structural models

แบบจำลองโครงสร้าง

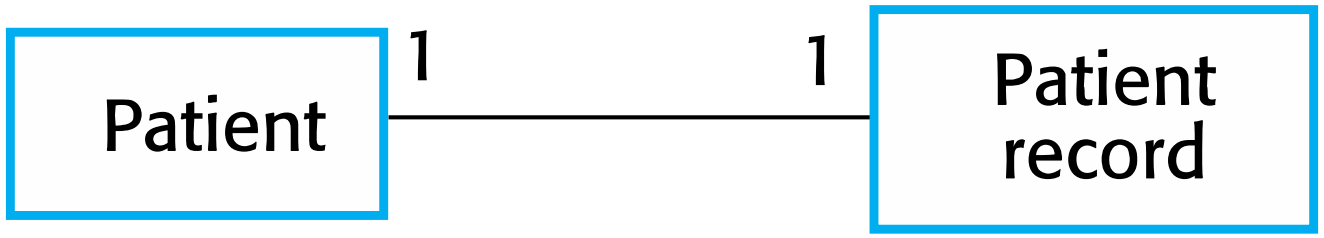
Structural models

- แบบจำลองโครงสร้างของซอฟต์แวร์
  - แสดงถึงการจัดระบบในแง่ขององค์ประกอบต่าง ๆ ที่ประกอบกันเป็นระบบ
  - แสดงถึงความสัมพันธ์ระหว่างองค์ประกอบเหล่านั้น
- แบบจำลองโครงสร้างอาจเป็นได้ทั้งแบบสถิตและไดนามิก
  - แบบสถิตจะแสดงโครงสร้างของการออกแบบระบบ
  - แบบไดนามิกจะแสดงถึงการจัดระบบในขณะปฏิบัติงาน (executing)
- การสร้างแบบจำลองโครงสร้าง จะเกิดขึ้นเมื่อมีการ discuss และออกแบบสถาปัตยกรรมของระบบ (system architecture)

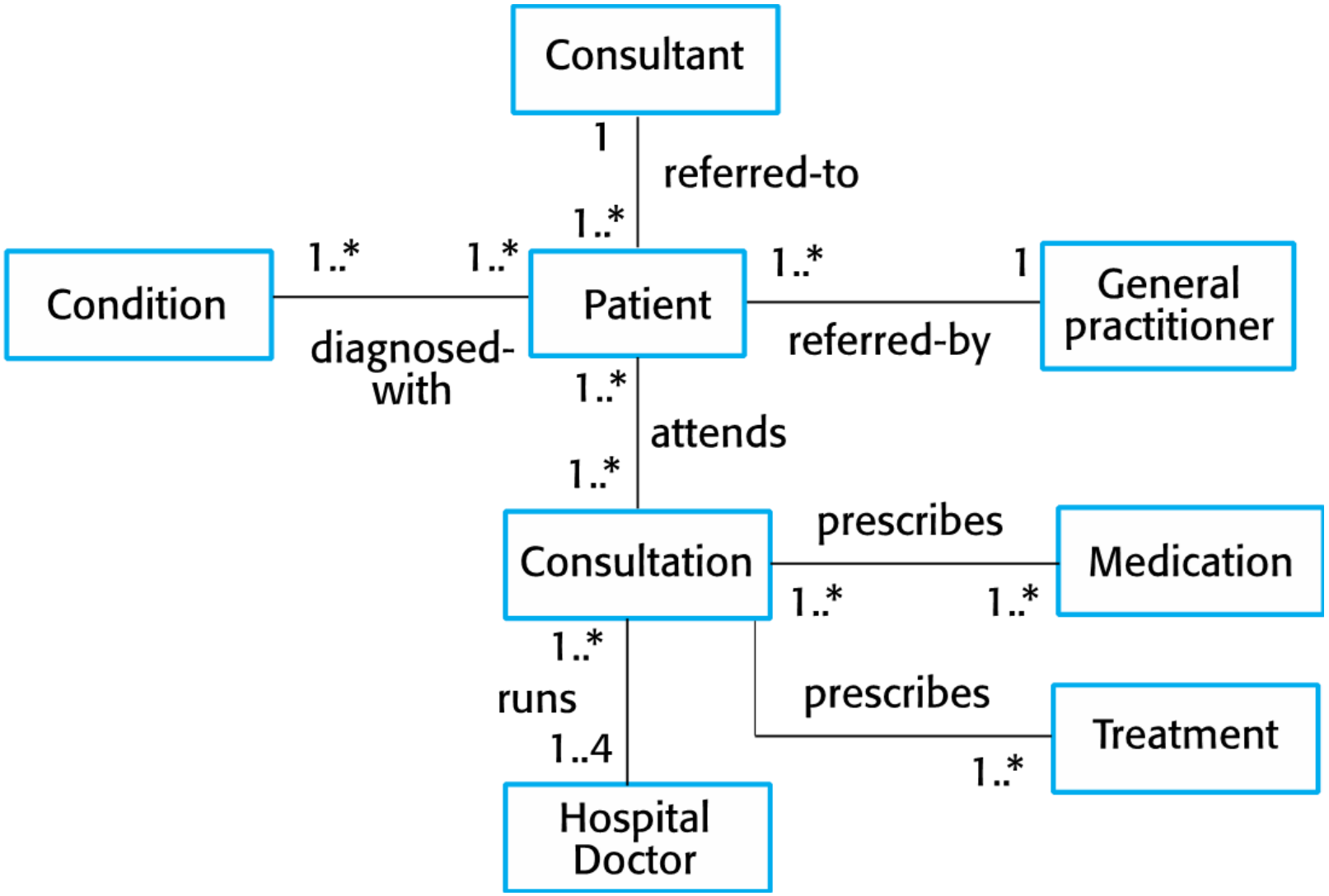
Class diagrams

- เราใช้แผนภาพคลาส (class diagrams) เมื่อสร้างแบบจำลองระบบเชิงวัตถุ
  - เพื่อแสดงคลาสในระบบและความสัมพันธ์ระหว่างคลาสเหล่านั้น
- ความสัมพันธ์ระหว่างคลาสในแผนภาพคลาสมีได้หลายรูปแบบ
  - Aggregation, Generalization, Association
- ในช่วงแรกของการพัฒนาแบบจำลอง วัตถุจะอาจจะเป็นตัวแทนในโลกแห่งความเป็นจริง เช่น ผู้ป่วย แพทย์ ใบสั่งยาเป็นต้น

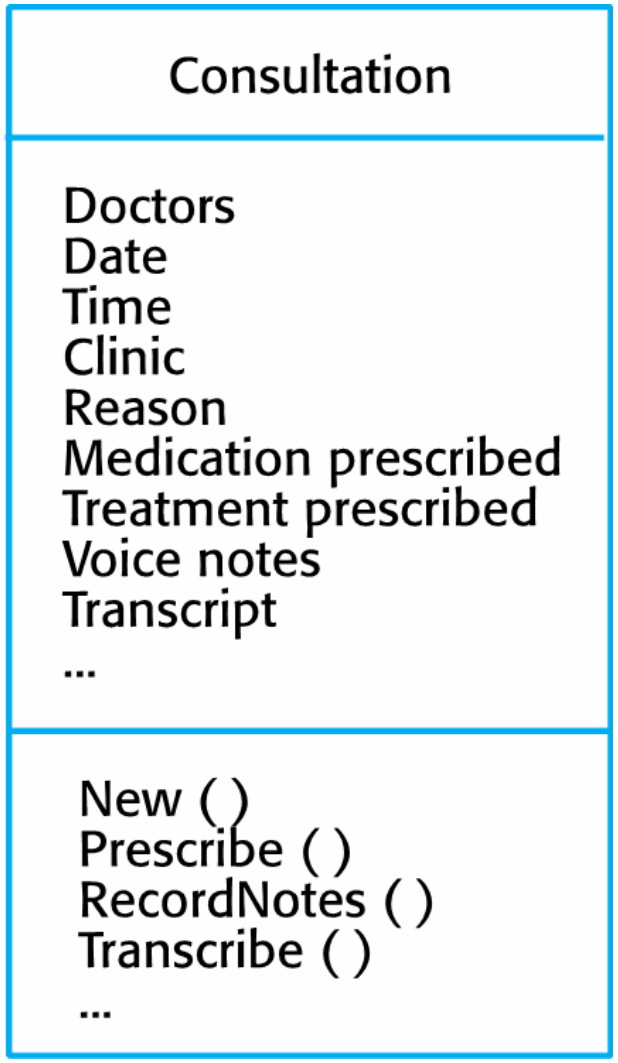
UML classes and association



Classes and associations in the MHC-PMS



The Consultation class



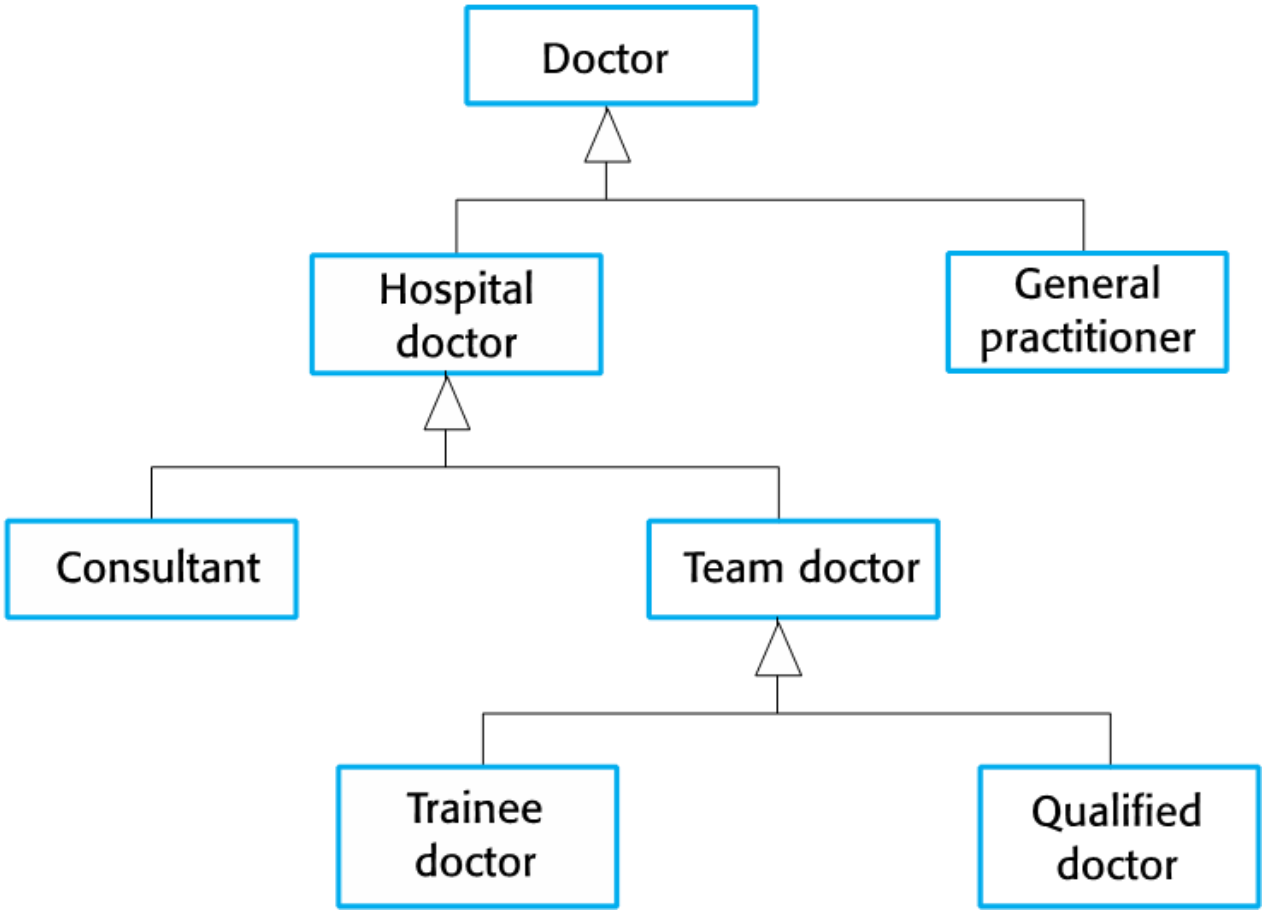
Generalization

- การสรุป (generalization) เป็นเทคนิคประจำวันที่เราใช้ในการจัดการความซับซ้อน
- แทนที่จะเรียนรู้รายละเอียดเกี่ยวกับลักษณะเฉพาะของทุกสิ่งที่เราพบ เราจะพยายามขจัดมันออก และมองทุกอย่างให้ง่ายขึ้น
  - โดยการสรุปเป็นคลาสอย่างง่าย เช่น บ้าน รถ สัตว์ เป็นต้น
- การสรุปจะช่วยให้เราอนุมานได้ว่าสมาชิกอื่นในคลาสเหล่านี้มีลักษณะทั่วไปที่คล้ายกัน เช่น เสือ และ สิงโต เป็นสัตว์ตระกูลแมว

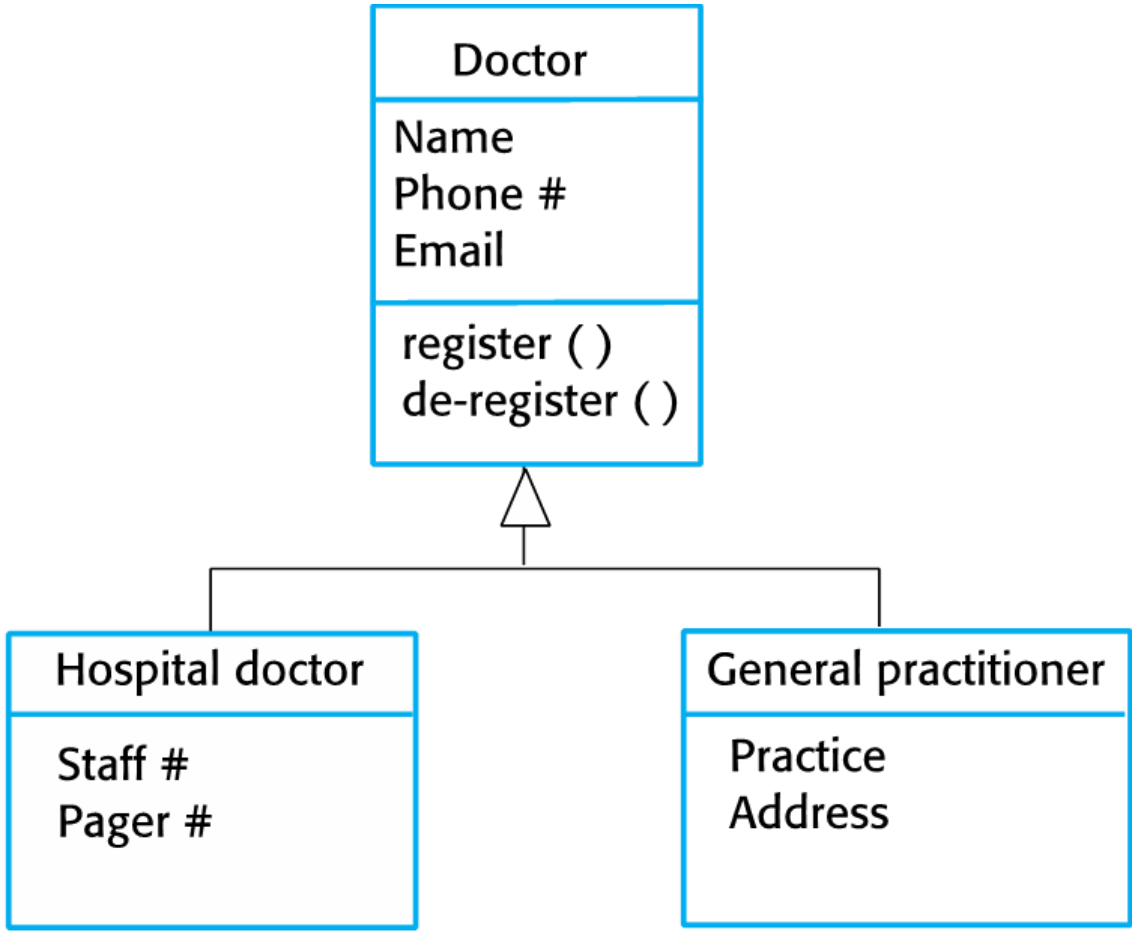
Generalization

- ในการสร้างแบบจำลอง จะเป็นการดี ถ้าเราทำการตรวจสอบคลาสในระบบ เพื่อดูว่าสามารถทำ generalization ได้หรือไม่
  - หากมีการเปลี่ยนแปลงใด ๆ เราไม่ต้องไล่ดูคลาสทั้งหมดในระบบ เพียงแค่ดูคลาสที่เกี่ยวข้องกันเท่านั้นก็พอ
- ในภาษาเชิงวัตถุเช่น Java, C# นั้น generalization จะถูกนำมาใช้โดยใช้กลไกการสืบทอดคลาสซึ่งเป็นองค์ประกอบของภาษา
- ในการทำ generalization ทั้งคุณสมบัติและความสามารถของคลาสระดับบนจะสามารถถ่ายทอดไปยังคาสระดับล่างด้วย
- คลาสระดับล่างจะรับช่วงเอาคุณสมบัติและความสามารถของคลาสระดับบนและสามารถเพิ่มเติมคุณสมบัติและความสามารถของตนเอง

A generalization hierarchy



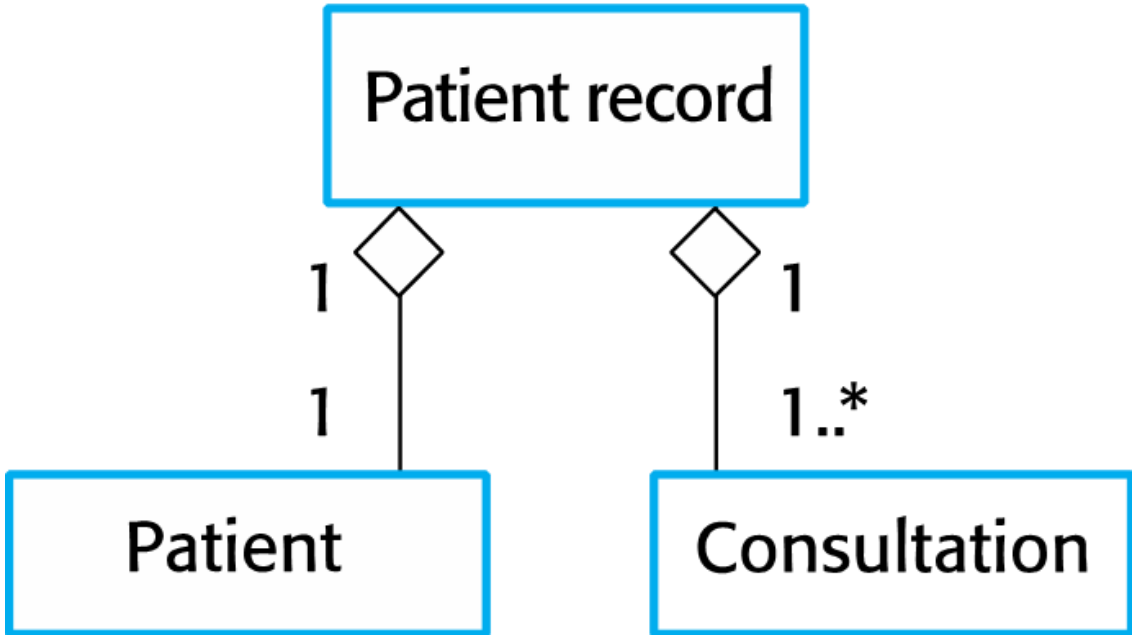
A generalization hierarchy with added detail



Object class aggregation models

- แบบจำลอง aggregation จะอธิบายให้เห็นว่าคลาสนั้นสามารถประกอบกันเป็นคลาสที่ใหญ่กว่าหรือแยกเป็นคลาสย่อย ๆ ได้อย่างไร
- แบบจำลอง aggregation จะสื่อถึงการประกอบกัน is-part-of

The aggregation association



Behavioral models

แบบจำลองพฤติกรรม

Behavioral models

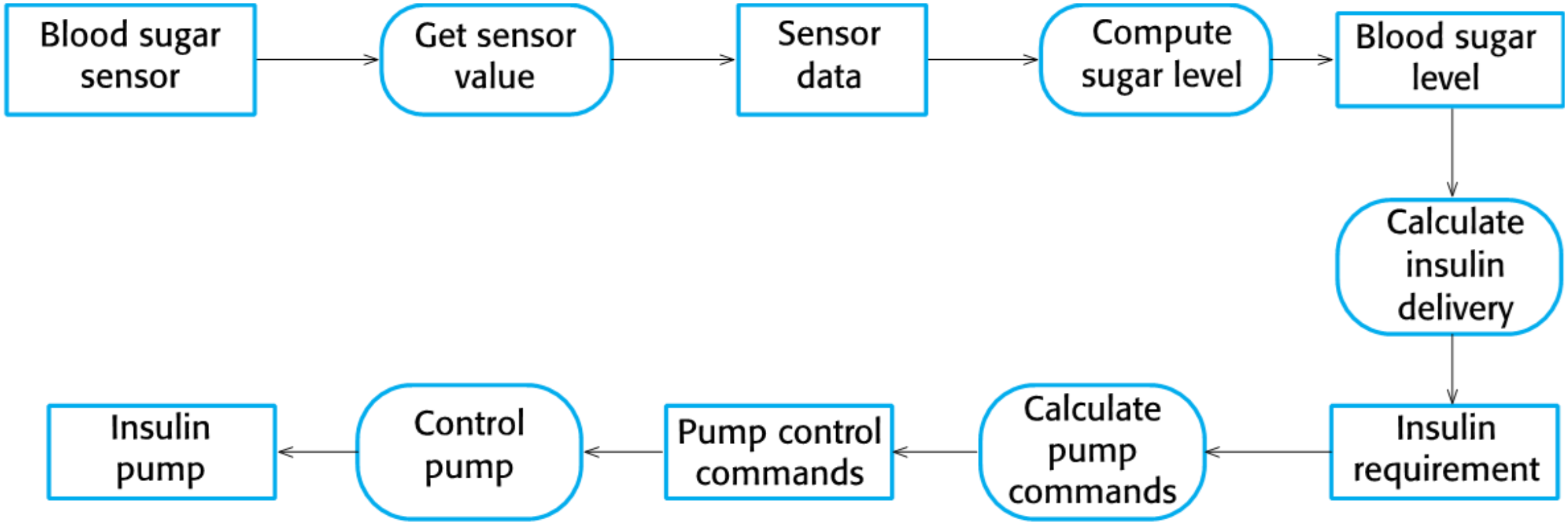
- แบบจำลองพฤติกรรมเป็นแบบจำลองไดนามิกของระบบในขณะที่กำลังดำเนินการ เมื่อระบบตอบสนองต่อสิ่งเร้าจากสภาพแวดล้อม โดยมีหน้าที่
  - แสดงให้เห็นว่าเกิดอะไรขึ้น
  - ระบุสิ่งที่ควรจะเกิดขึ้น
- เราสามารถจำแนกสิ่งเร้าออกเป็นสองประเภท:

1. ข้อมูล เมื่อข้อมูลบางอย่างมาถึง จะต้องได้รับการประมวลผลโดยระบบ
2. เหตุการณ์ เมื่อมีบางเหตุการณ์เกิดขึ้น จะเรียกใช้การประมวลผลของระบบ
   - บางเหตุการณ์อาจมีข้อมูลเข้ามาเกี่ยวข้องด้วย (แต่ก็ไม่จำเป็นเสมอไป)

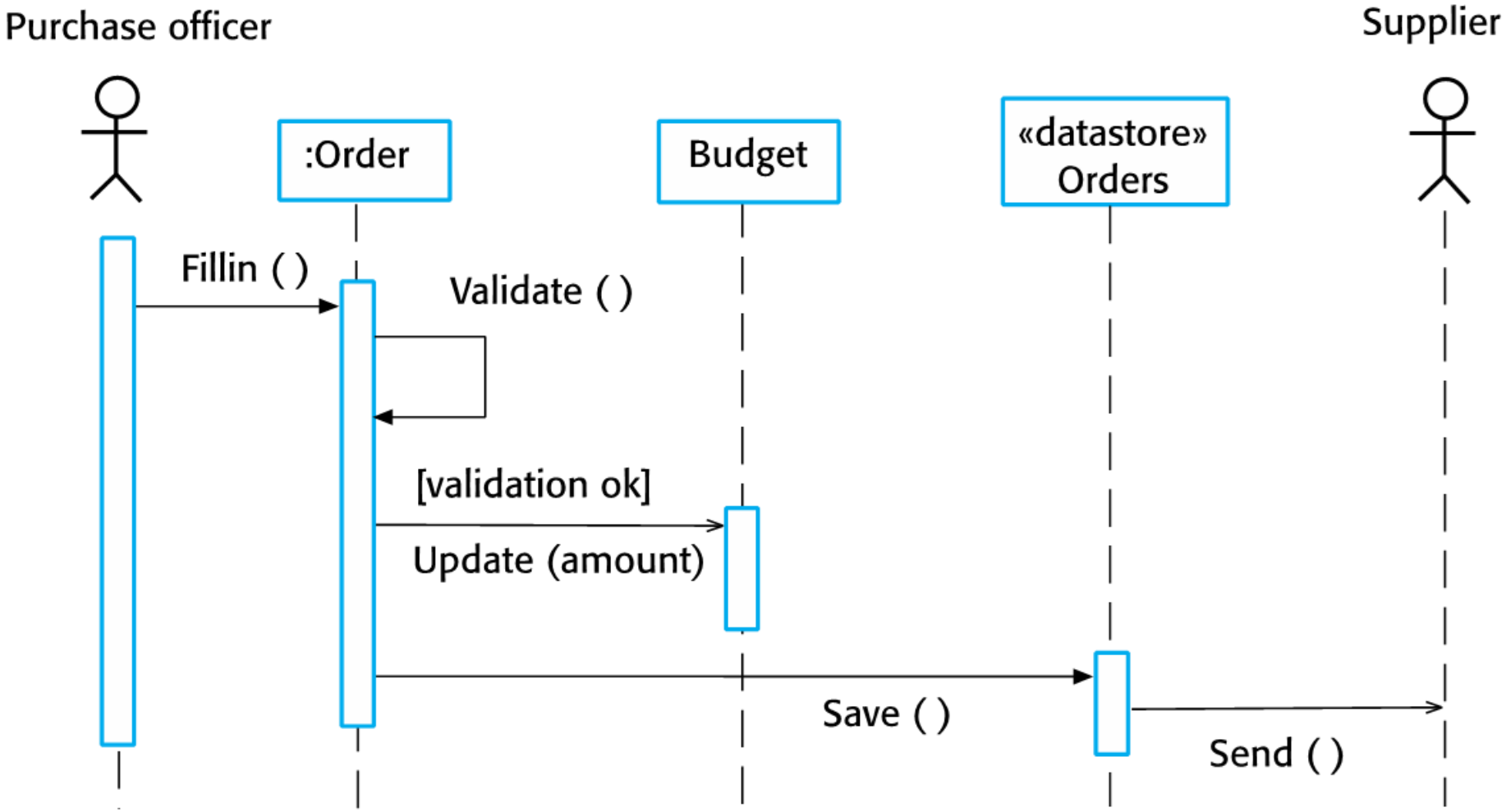
Data-driven modeling

- ในระบบธุรกิจจำนวนมาก จะเป็นระบบประมวลผลที่ขับเคลื่อนโดยข้อมูล (data-driven model)
  - ถูกควบคุมโดยการป้อนข้อมูลลงในระบบ
  - มีการประมวลผลเหตุการณ์ภายนอกค่อนข้างน้อย
- แบบจำลองนี้จะแสดงลำดับของการดำเนินการ ประกอบด้วยการประมวลผลข้อมูลอินพุตและสร้างผลลัพธ์ที่เอาต์พุต
- แบบจำลองมีประโยชน์อย่างยิ่งในการวิเคราะห์ requirement เนื่องจากสามารถใช้เพื่อแสดงการประมวลผลแบบ end-to-end ในระบบ

An activity model of the insulin pump's operation



Order processing



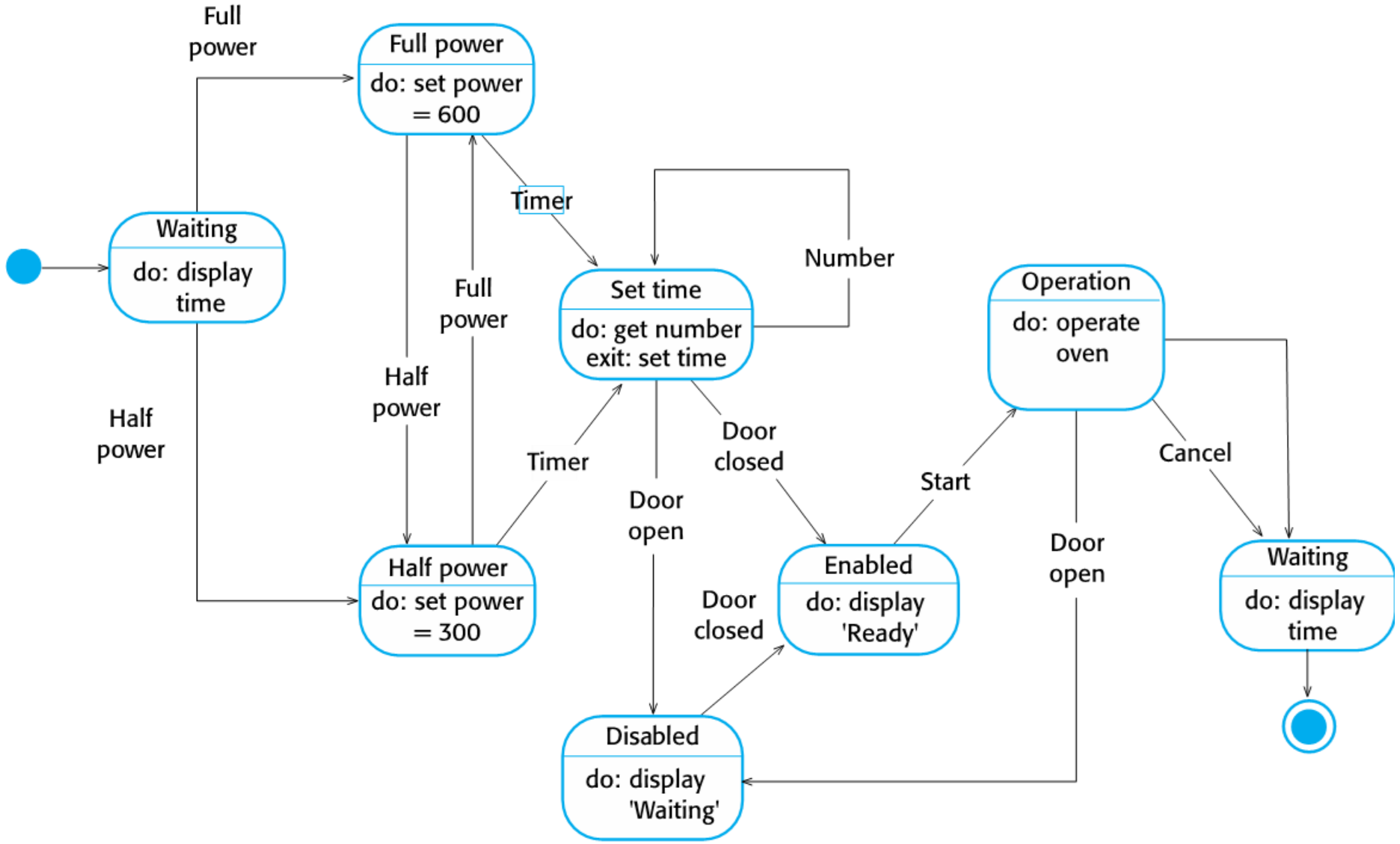
Event-driven modeling

- ตัวอย่างของระบบที่ขับเคลื่อนด้วยเหตุการณ์ (event-driven) ได้แก่ ระบบเรียลไทม์ (real-time systems)
  - เป็นการขับเคลื่อนด้วยเหตุการณ์ ที่มีการประมวลผลข้อมูลน้อยที่สุด
- แบบจำลองขับเคลื่อนด้วยเหตุการณ์ จะแสดงวิธีการที่ระบบตอบสนองต่อเหตุการณ์ ทั้งภายนอกและภายใน
- โดยปกติ ระบบตามแบบจำลองนี้ จะมีสถานการณ์ทำงาน (states) เป็นจำนวนรูปแบบที่จำกัด (finite number of state)
  - เหตุการณ์ที่เกิดขึ้น (stimuli) อาจทำให้เกิดการเปลี่ยนแปลงจาก state หนึ่งไปอีก state หนึ่ง

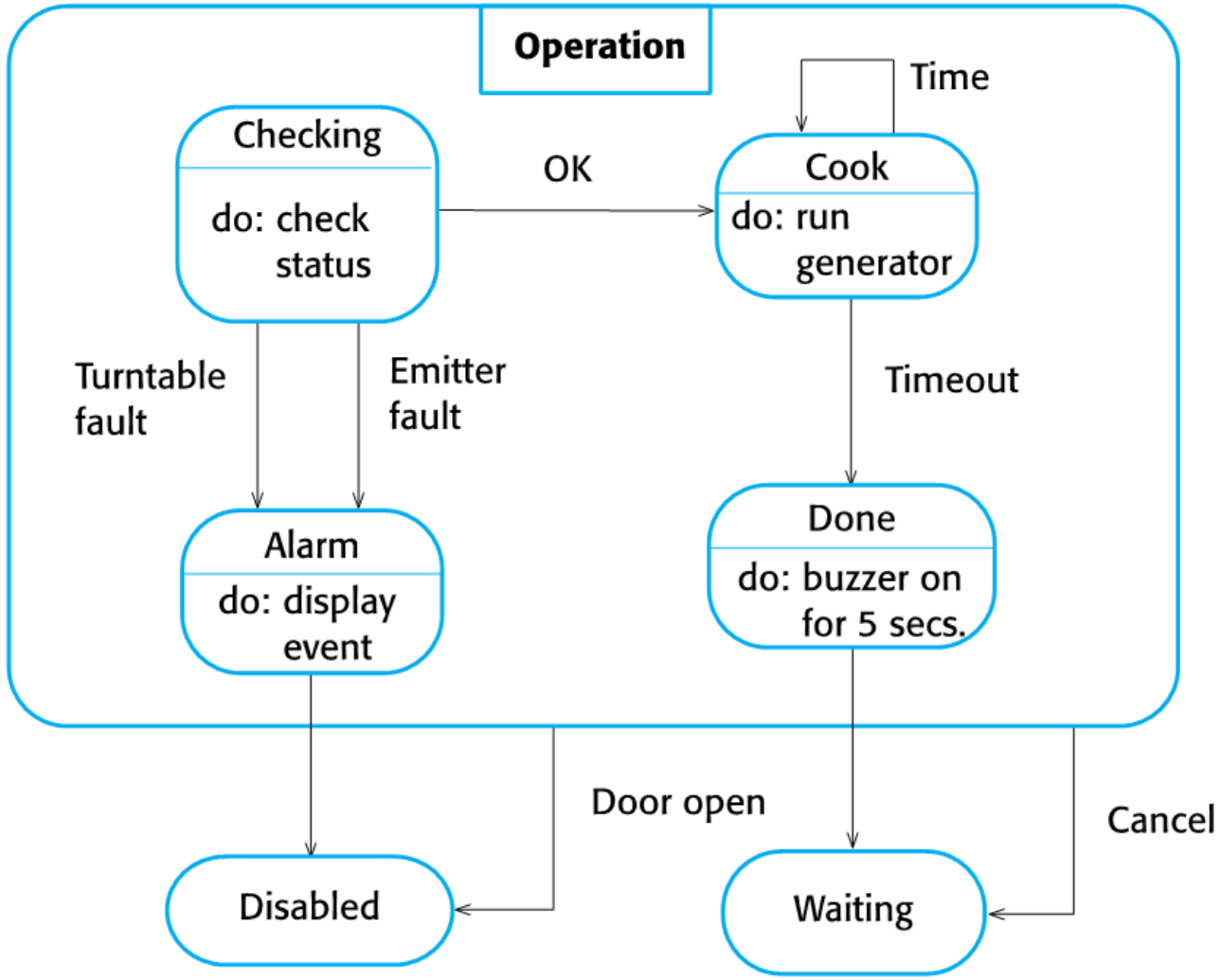
State machine models

- แบบจำลอง state machine จะจำลองพฤติกรรมของระบบในการตอบสนองต่อเหตุการณ์ ทั้งภายนอกและภายใน
  - มักใช้สำหรับการสร้างแบบจำลองระบบเรียลไทม์
- แบบจำลองนี้จะแสดง state ของระบบด้วยสัญลักษณ์วงกลมเป็นโหนดและเหตุการณ์เป็นแส้นโค้งเชื่อมระหว่างโหนด
  - เมื่อเหตุการณ์เกิดขึ้นระบบจะย้ายจาก state หนึ่งไปยังอีก state หนึ่ง
- Statecharts เป็นส่วนสำคัญของ UML และใช้อธิบายแบบจำลอง state machine

State diagram of a microwave oven



Microwave oven operation



States and stimuli for the microwave oven (a)

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows ‘Half power’.
Full power	The oven power is set to 600 watts. The display shows ‘Full power’.
Set time	The cooking time is set to the user’s input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows ‘Not ready’.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows ‘Ready to cook’.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows ‘Cooking complete’ while buzzer is sounding.

States and stimuli for the microwave oven (b)

Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

Model-driven engineering

แบบจำลองพฤติกรรม

Model-driven engineering

- วิศวกรรมแบบจำลอง (Model Driven Engineering : MDE) เป็นแนวทางในการพัฒนาซอฟต์แวร์ซึ่งต้องการเอาต์พุตหลักของกระบวนการพัฒนาที่เป็นโมเดล (ไม่ใช่เป็นโปรแกรม)
- โปรแกรมที่รันบน hardware/software platform จะถูกสร้างขึ้นโดยอัตโนมัติจากโมเดล
- ผู้คิดค้น MDE ยืนยันว่าเรื่องนี้จะเพิ่มระดับความเป็นนิยามหรือนามธรรมในด้านวิศวกรรมซอฟต์แวร์
  - เพื่อให้วิศวกรไม่ต้องกังวลกับรายละเอียดของภาษาโปรแกรมหรือข้อมูลเฉพาะของแพลตฟอร์มที่รันโปรแกรมนั้น ๆ

Usage of model-driven engineering

- วิศวกรรมขับเคลื่อนด้วยแบบจำลองยังคงอยู่ในช่วงเริ่มต้นของการพัฒนาและยังไม่เป็นที่แน่ชัดว่าจะมีผลต่อการปฏิวัติวงการวิศวกรรมซอฟต์แวร์หรือไม่
- ข้อดี
  - ช่วยให้ระบบได้รับการพิจารณาในระดับนิยามที่สูงขึ้น
  - การสร้างโค้ดโดยอัตโนมัติ จะช่วยให้มีต้นทุนที่ถูกกว่าในการ port หรือปรับเปลี่ยนซอฟต์แวร์ เพื่อไปรันบนแพลตฟอร์มใหม่
- จุดด้อย
  - ในปัจจุบันโมเดลที่มีอยู่ เหมาะสำหรับการให้นิยาม และยังไม่สามารถนำไป implement จริงได้
  - การประหยัดจากการเขียนโค้ด แต่อาจจะไปเพิ่มค่าใช้จ่ายในการพัฒนาตัวแปลภาษาสำหรับแพลตฟอร์มใหม่

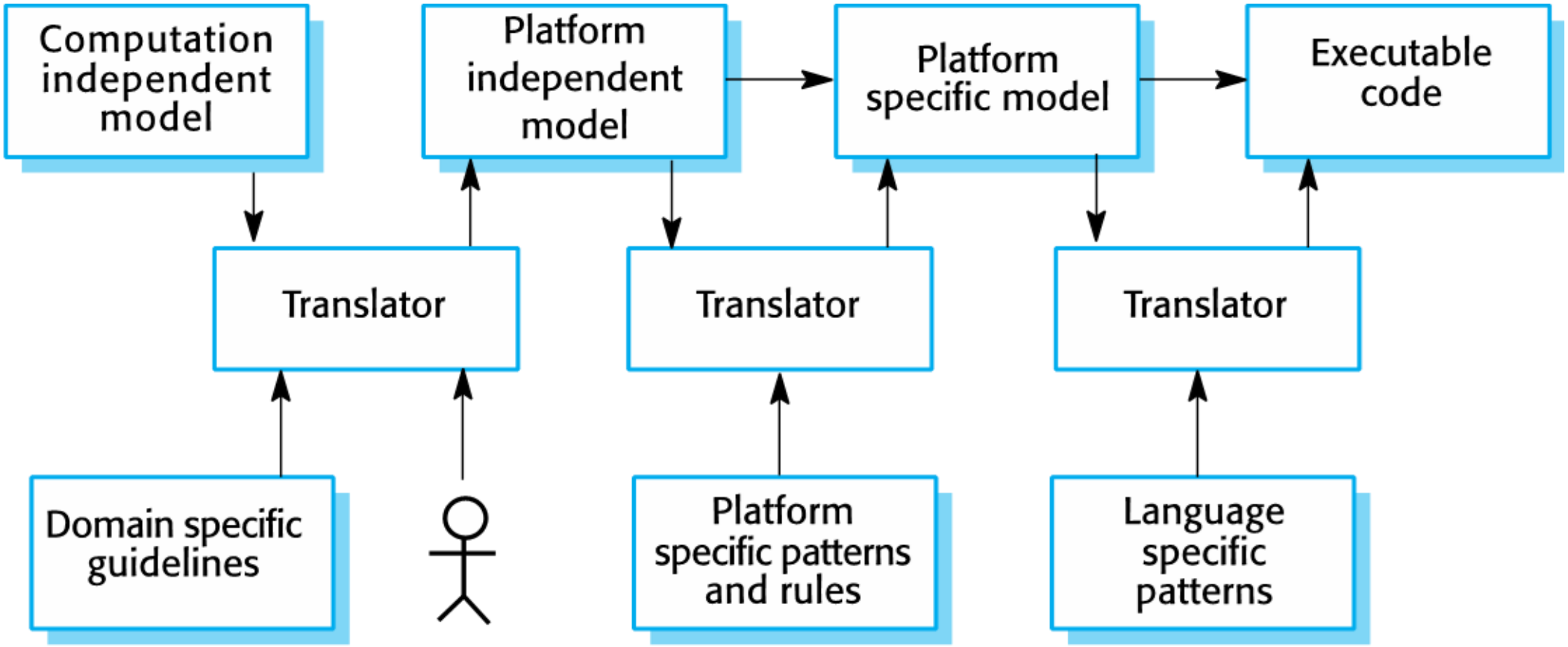
Model driven architecture

- สถาปัตยกรรมที่ขับเคลื่อนด้วยโมเดล (MDA) เป็นจุดกำเนิดของวิศวกรรมแบบจำลองทั่วไป
- MDA เป็นรูปแบบที่เน้นการสร้างแบบจำลองเพื่อการออกแบบและสร้างซอฟต์แวร์
  - อธิบายระบบโดยอาศัยแบบจำลองต่างๆ ที่มีอยู่ใน UML
- ในสถาปัตยกรรมนี้ แบบจำลองซึ่งถูกนิยามในระดับต่าง ๆ จะถูกสร้างขึ้น
  - เริ่มจากการนิยามระดับบนสุด ที่เป็นอิสระจากแพลตฟอร์ม
  - โดยหลักการแล้ว ถ้าออกแบบได้เหมาะสม เครื่องมืออัตโนมัติ จะสร้าง code ที่ทำงานได้ โดยไม่ต้องอาศัยคนเข้าไปเกี่ยวข้อง

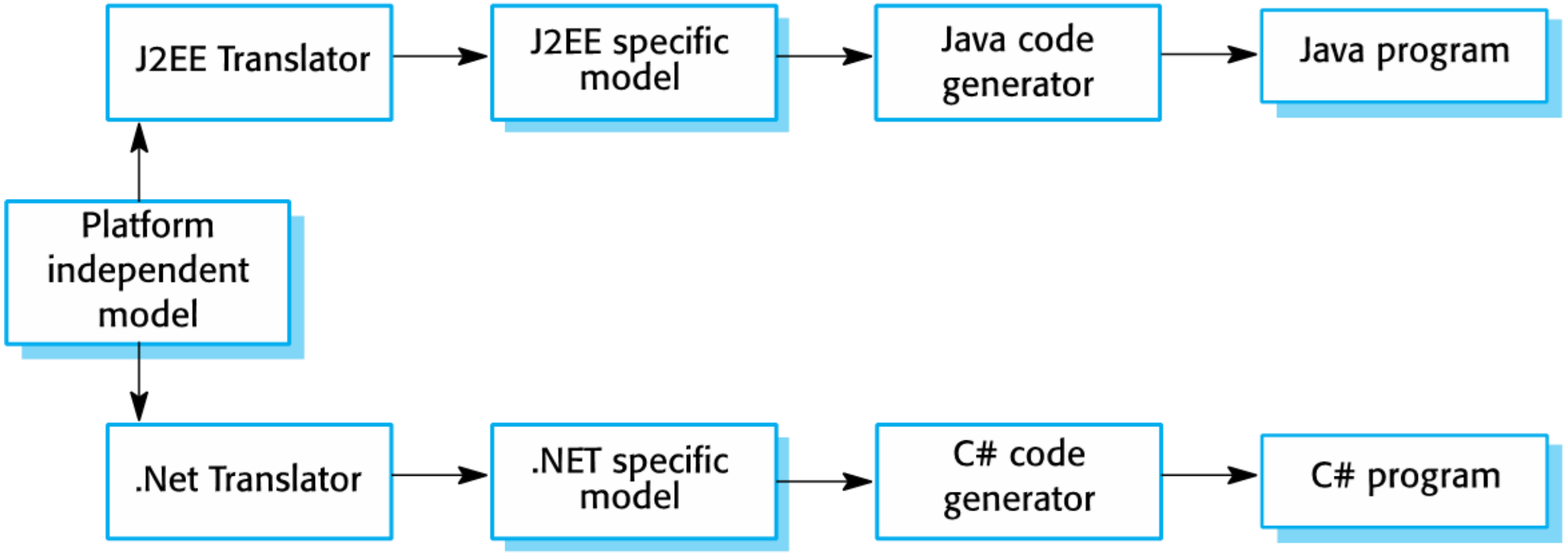
Types of model

- A computation independent model (CIM)
  - โมเดลเหล่านี้ถูกใช้ในการนิยามระบบ บางครั้งเรียกว่าโมเดลโดเมน
- A platform independent model (PIM)
  - ทำการจำลองการดำเนินการของระบบโดยไม่ต้องคำนึงถึงว่ามันจะถูกสร้างได้อย่างไร
  - PIM มักอธิบายโดยใช้โมเดล UML ที่แสดงโครงสร้างสถิต (static) และอธิบายวิธีตอบสนองต่อเหตุการณ์ภายนอกและภายใน
- Platform specific models (PSM)
  - เป็นการดัดแปลงแบบจำลอง PIM โดยมี PSM อิสระสำหรับแต่ละแพลตฟอร์มแอ็พพลิเคชัน
  - ในหลักการอาจมี PSM หลายเลเยอร์ของ โดยแต่ละเลเยอร์จะมีการอธิบายรายละเอียดเฉพาะของแพลตฟอร์มเอาไว้

MDA transformations



Multiple platform-specific models



Agile methods and MDA

- นักพัฒนาซอฟต์แวร์ MDA อ้างว่ามีวัตถุประสงค์เพื่อสนับสนุนแนวทางการพัฒนาแบบ iterative และสามารถใช้งานได้โดยใช้วิธี agile
- แนวความคิดของการสร้างแบบจำลองนี้ ยังขัดแย้งความคิดพื้นฐานในแถลงการณ์ agile และนักพัฒนา agile ส่วนใหญ่ยังรู้สึกเป็นการฝืนใจที่จะใช้ model-driven engineering
- ในอนาคตถ้า PIM สามารถทำการ transformation ได้โดยอัตโนมัติและสมบูรณ์แบบ ก็หมายความว่า MDA อาจใช้ในกระบวนการพัฒนา agile ได้

Adoption of MDA

- มีหลายปัจจัยเป็นข้อจำกัดในการใช้ MDE / MDA
  - ต้องการเครื่องมือเฉพาะทาง เพื่อใช้ในการแปลงโมเดลจากระดับหนึ่งไปอีกระดับหนึ่ง
  - มีความพร้อมใช้งานของเครื่องมือที่จำกัด
  - องค์กรอาจต้องมีการปรับตัวและปรับแต่งเครื่องมือให้เหมาะกับสภาพแวดล้อมของตน
  - บริษัทต่าง ๆ ไม่เต็มใจที่จะพัฒนาเครื่องมือของตน เพื่อใช้สำหรับพัฒนาระบบโดยใช้ MDA
  - มีความกังวลว่าหากนำเครื่องมือ MDA จากบริษัทขนาดเล็ก ๆ มาสร้างระบบที่มีอายุการใช้งานยาวนาน อาจได้รับผลกระทบเมื่อบริษัทเหล่านั้นเลิกกิจการ

Adoption of MDA

- แบบจำลองเป็นเครื่องมือที่ดีในการอภิปรายเกี่ยวกับการออกแบบซอฟต์แวร์
  - อย่างไรก็ตามนิยามที่เป็นประโยชน์สำหรับการอภิปรายอาจไม่ใช่นิยามที่เหมาะสมสำหรับการ implement ซอฟต์แวร์
- สำหรับระบบที่มีความซับซ้อนสูง การ implement ไม่ใช่ปัญหาหลัก
  - ปัญหาสำคัญมักเกิดกับ วิศวกรรมความต้องการ, ความปลอดภัยและเชื่อถือได้, การทำงานร่วมกับระบบที่มีอยู่เดิม จนถึงการทดสอบระบบ

Key points

- แบบจำลองคือมุมมองนามธรรมของระบบที่ละเว้นรายละเอียดของระบบ
  - สามารถเพิ่มเติมแบบจำลองย่อย ๆ เพื่อแสดง บริบท ปฏิสัมพันธ์ โครงสร้างและพฤติกรรม ของระบบ
- แบบจำลองบริบทแสดงให้เห็นว่าระบบที่กำลังสร้าง อยู่ในสภาพแวดล้อมที่มีระบบและกระบวนการอื่น ๆ อย่างไร
- use case diagram ใช้อธิบายปฏิสัมพันธ์ระหว่างระบบกับ actor จากภายนอก
- Sequence diagram เพิ่มรายละเอียดให้กับแบบจำลองปฏิสัมพันธ์ โดยการแสดงปฏิสัมพันธ์ระหว่างวัตถุในระบบ
- แบบจำลองโครงสร้าง แสดงถึงองค์กรและสถาปัตยกรรมของระบบ
- แผนภาพคลาสใช้ในการกำหนดโครงสร้างแบบคงที่ของคลาสในระบบและความสัมพันธ์ของระบบ

Key points

- แบบจำลองพฤติกรรม ใช้เพื่ออธิบายพฤติกรรมไดนามิกในขณะปฏิบัติการของระบบ
  - เราสามารถจำลองจากมุมมองของข้อมูลที่ประมวลผลโดยระบบ (data-driven) หรือโดยเหตุการณ์ที่กระตุ้นการตอบสนองจากระบบ (event driven)
- แผนภาพกิจกรรมสามารถใช้เพื่อจำลองการประมวลผลข้อมูลโดยที่แต่ละกิจกรรมจะแสดงถึงแต่ละขั้นตอนของกระบวนการ
- แผนภาพสถานะใช้เพื่อจำลองพฤติกรรมของระบบเพื่อตอบสนองต่อเหตุการณ์ภายในหรือภายนอก
- วิศวกรรมแบบจำลองเป็นแนวทางใหม่ในการพัฒนาซอฟต์แวร์ ซึ่งระบบจะแสดงด้วยชุดของโมเดลที่สามารถแปลงเป็นรหัสที่ทำงานได้โดยอัตโนมัติ

คำถาม???