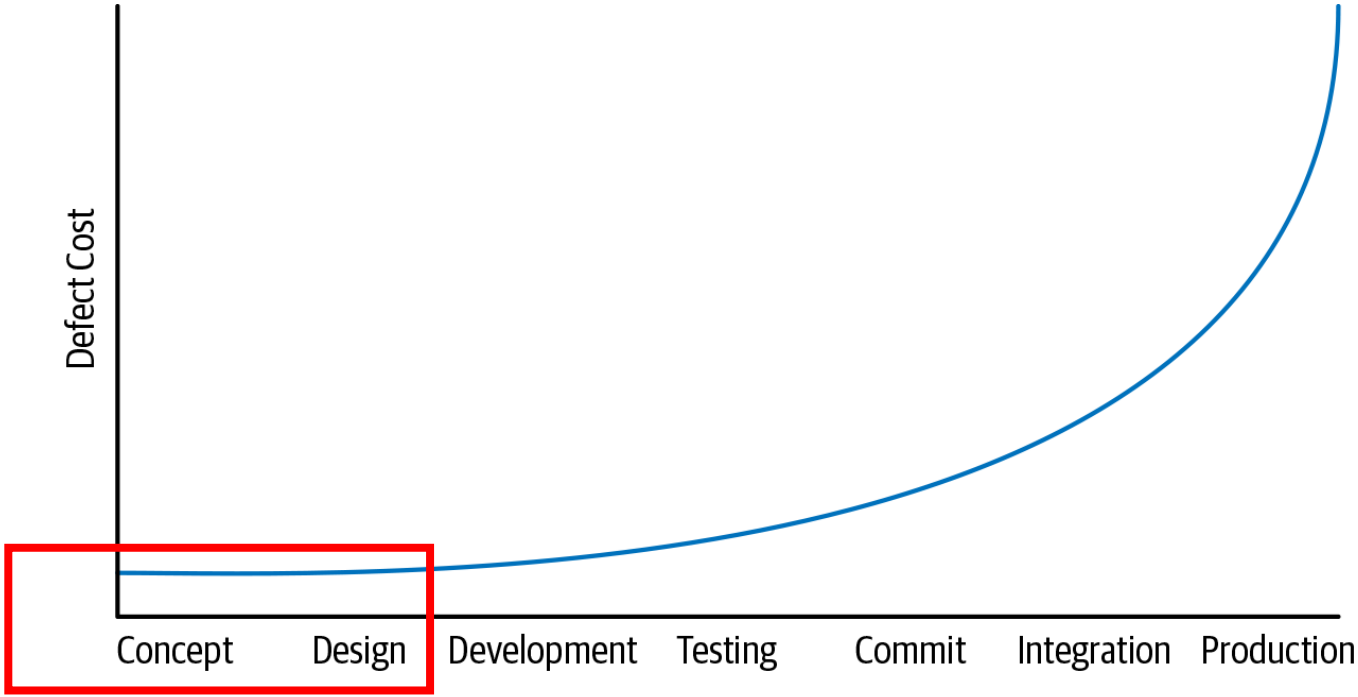
CS3213: Foundations of Software Engineering

Systems Modeling

Systems Modeling



https://abseil.io/resources/swe-book/html/ch01.html

Laksa Stall at the Deck

Laksa Stall: Structural Perspective

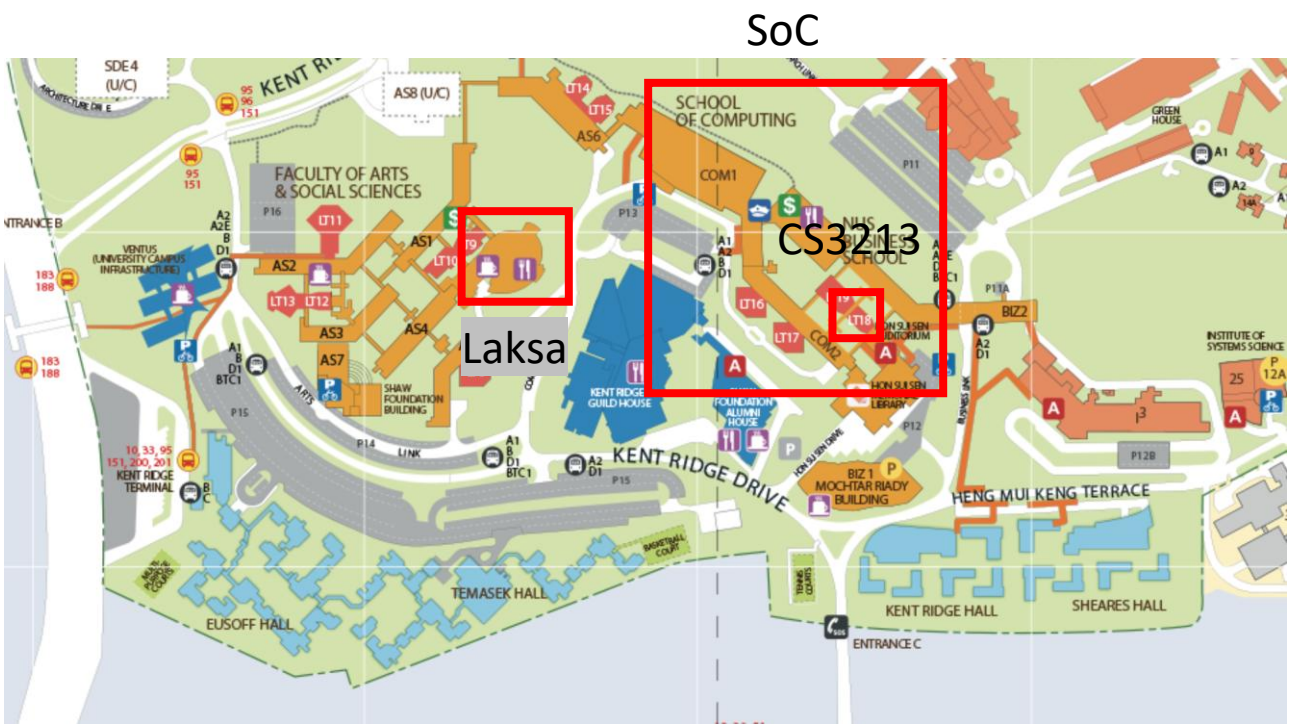
Various toppings

Noodles (optional)

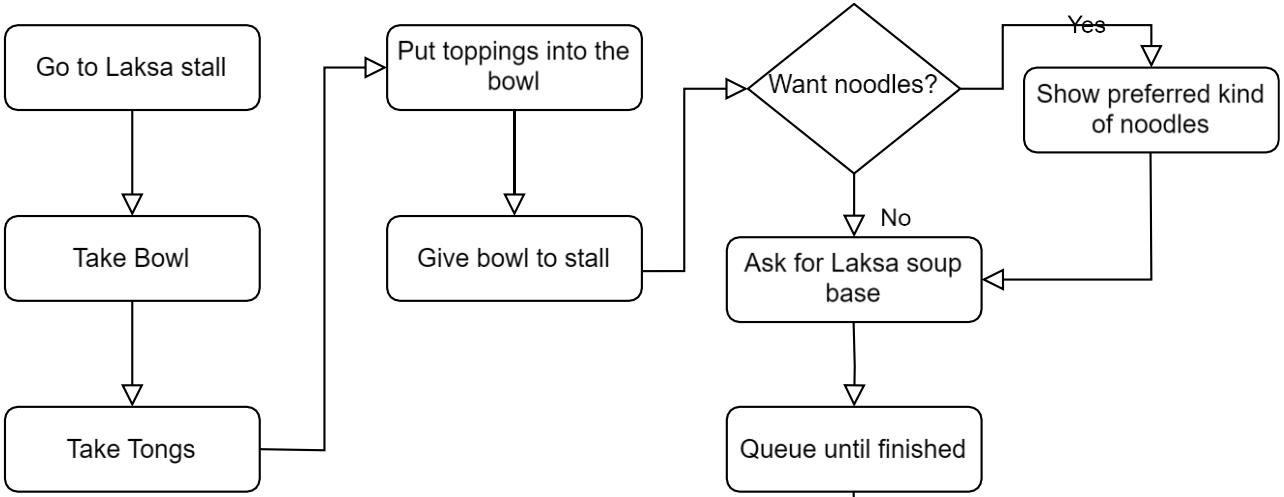
Spicy, thick, curry-like, coconut-y soup base

Model: does not account for other types of Laksa (e.g., Penang Laksa)

Laksa Stall: External Perspective

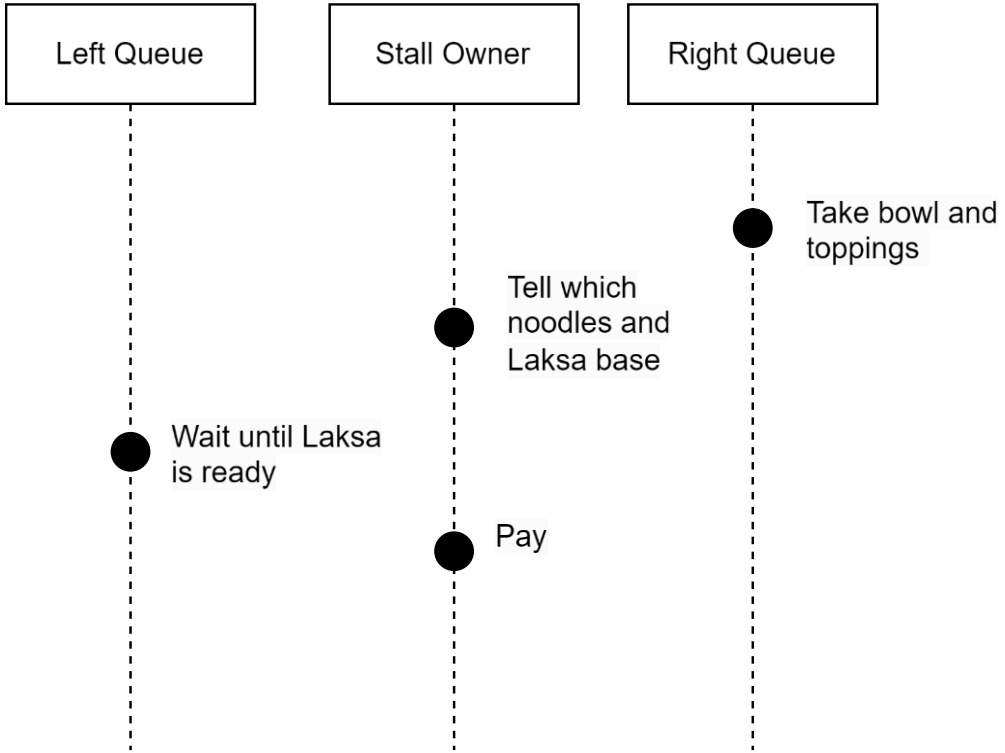


Laksa Stall: Behavioral Perspective



Flowchart (similar to activity diagrams in UML)

Laksa Stall: Interaction Perspective



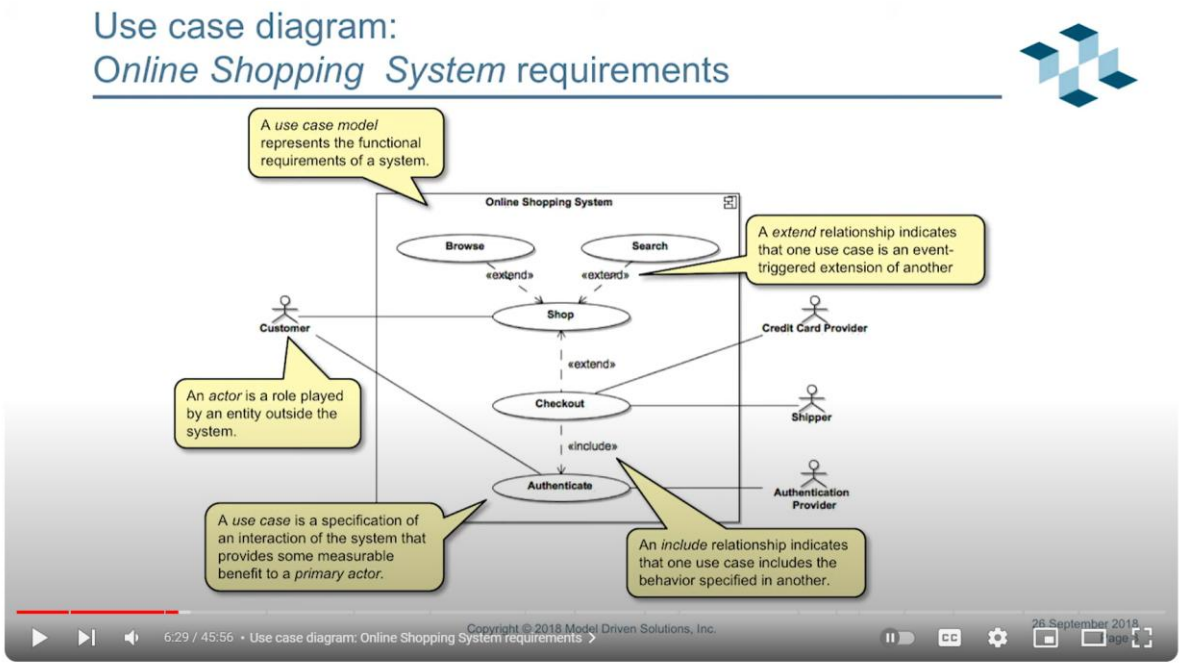
Perspectives

- **Structural perspective**: model the organization of a system or the structure of the data that is processed by the system
- **External perspective**: context or environment of the system
- **Behavioral perspective**: model the dynamic behavior of the system and how it responds to events
- **Interaction perspective**: interactions between a system and its environment, or between the components of a system

Modeling Languages Examples

- Unified Modelling Language (UML)
- C4
- Business Process Model and Notation (BPMN)
- Systems modeling language (SysML)
- Entity Relationship Diagram
- …

OMG, UML, and SysML



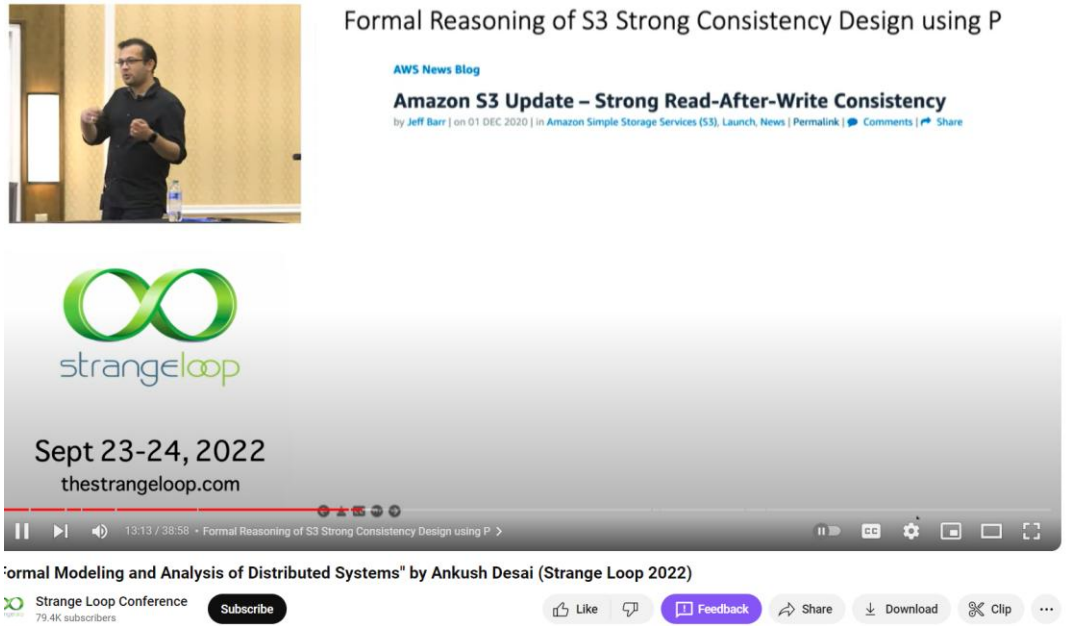
Overview: UML® (Unified Modeling Language™) and SysML® (Systems Modeling Language™)

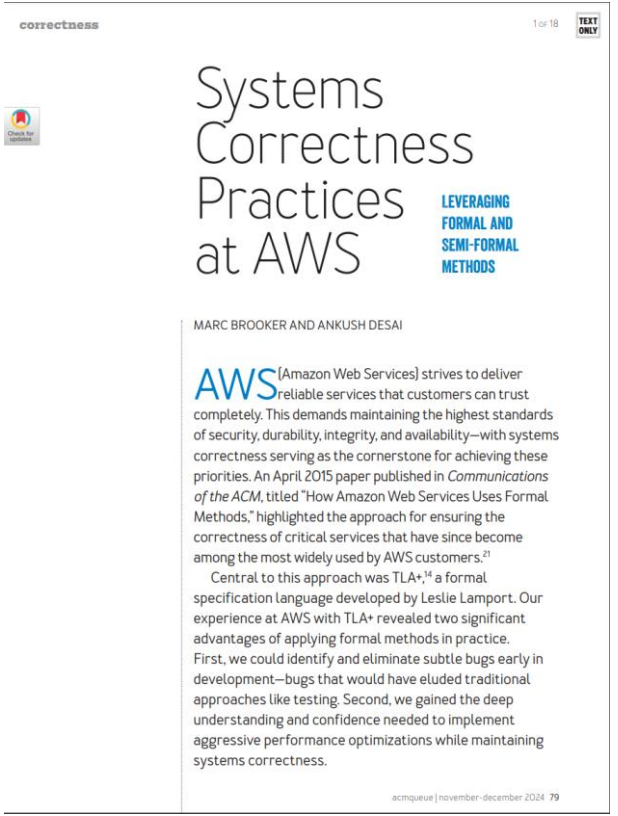
https://www.youtube.com/watch?v=vAHHdnIV8rU

System Modeling

- Goal: develop **abstract models** of a system
- Typically means a graphical notation based on diagram types in the **Unified Modeling Language (UML)**
- Emphasized in plan-driven development, but also used in agile development in a lightweight manner

Formal Models: P





https://www.youtube.com/watch?v=5YjsSDDWFDY

System Models and SE Activities

- System models are developed as part of **requirements engineering** and **system design** processes
  - RE: to help derive the **detailed requirements** for a system
  - System design: to **describe the system** to engineers implementing the system
- After implementation to **document** the system's structure and operation

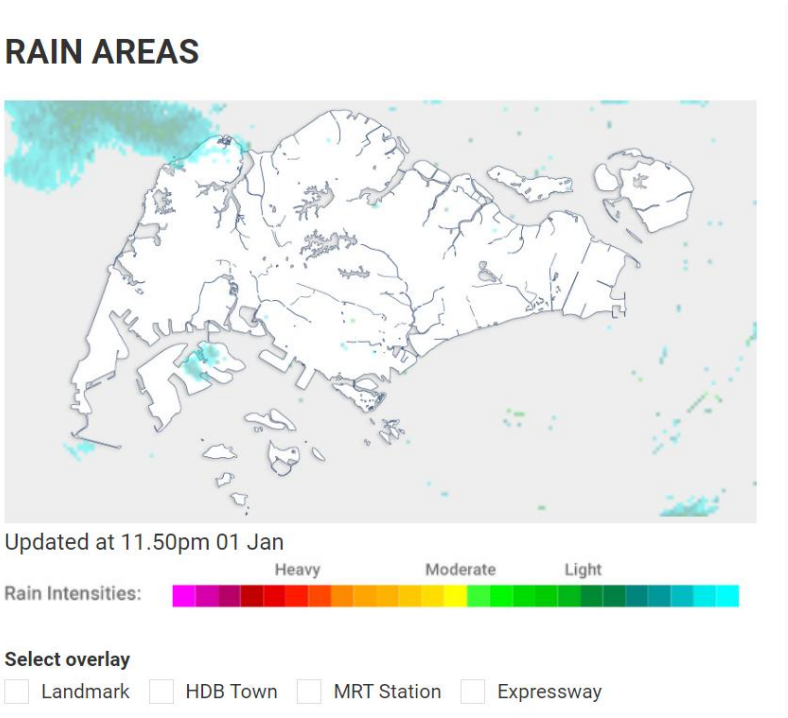
Abstraction vs Representation

- **Modeling** is about abstractions
- **Abstraction**: leaves out details to make the system easier to understand
- **Representation**: Maintains all information about the system being represented

Singapore: MRT Map

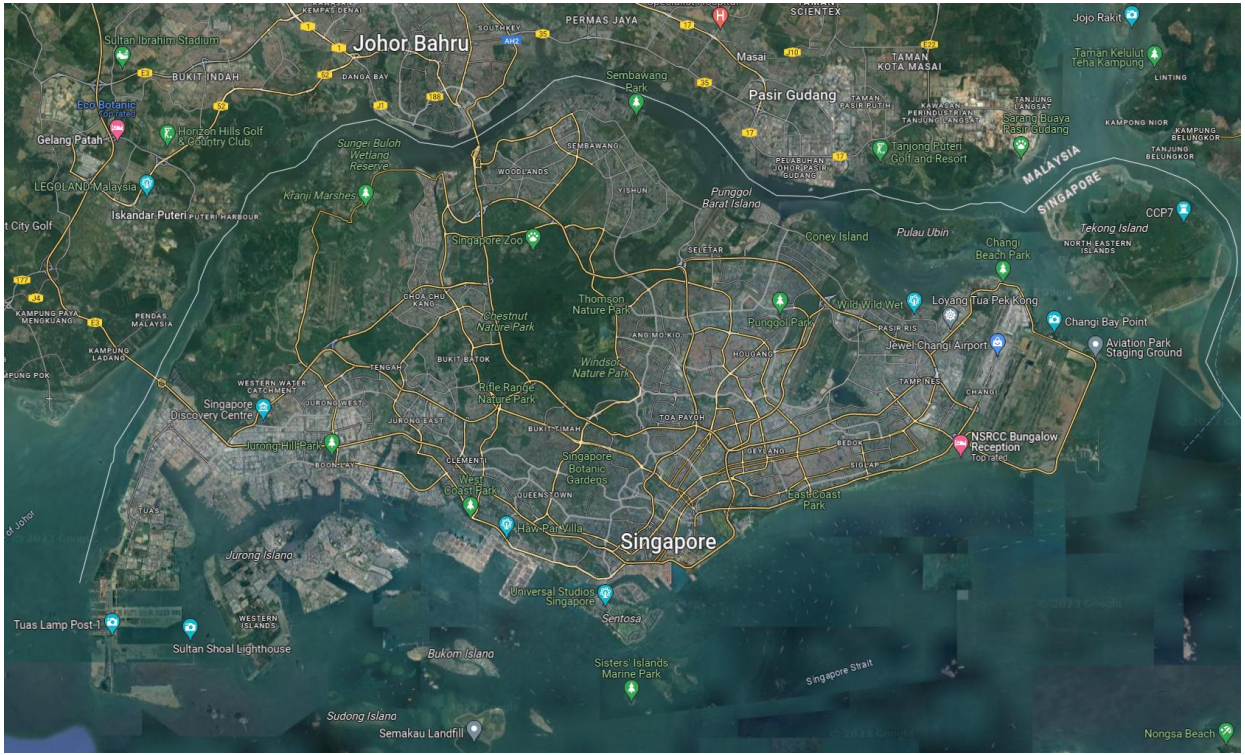
https://www.lta.gov.sg/content/ltagov/en/getting_around/public_transport/rail_network.html

Singapore: Rain Area View



https://www.nea.gov.sg/corporate-functions/weather

Singapore: Satellite View

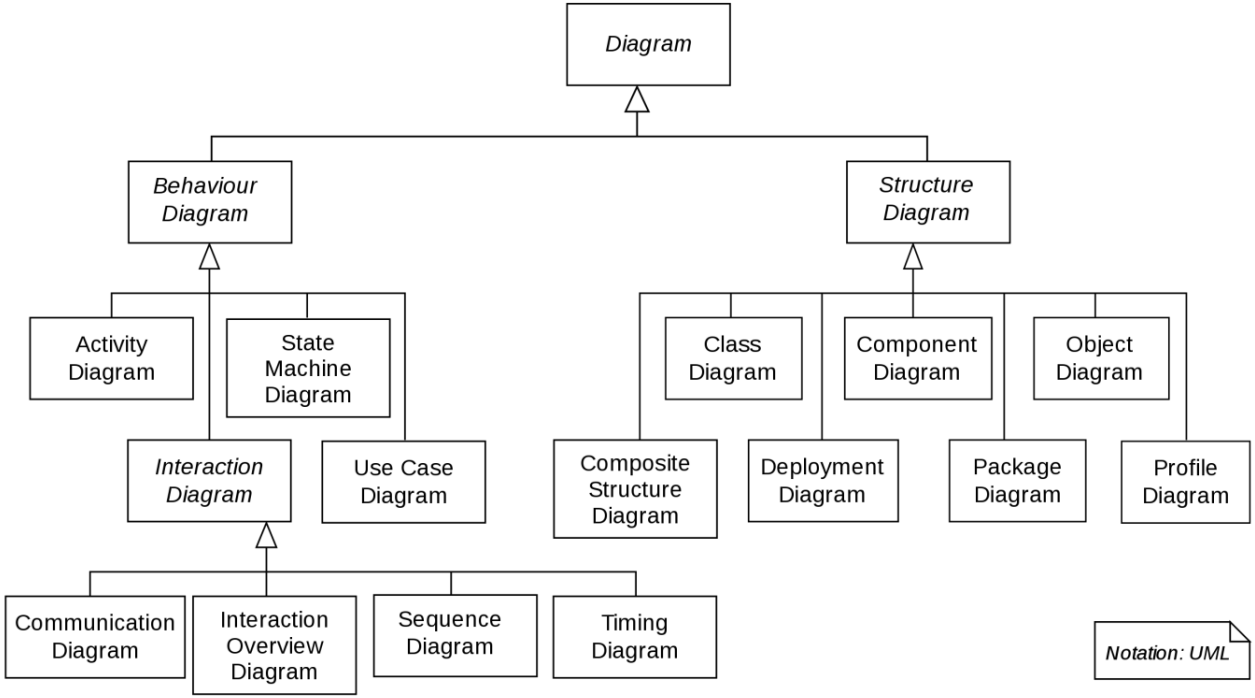


https://www.google.com/maps/

Modeling Languages Examples

- Unified Modelling Language (UML)
- C4
- Business Process Model and Notation (BPMN)
- Systems modeling language (SysML)
- …

Why UML?



https://en.wikipedia.org/wiki/Unified_Modeling_Language

Study on the Use of UML

- Interview with 50 practitioners
- 11 use it selectively
- 4 use it for automatic code generation or retrofit it (e.g., due to customer requirements)

UML in Practice

Marian Petre
Centre for Research in Computing
The Open University
Milton Keynes, UK
m.petre@open.ac.uk

***Abstract*—UML has been described by some as "the lingua franca of software engineering". Evidence from industry does not necessarily support such endorsements. How exactly is UML being used in industry – if it is? This paper presents a corpus of interviews with 50 professional software engineers in 50 companies and identifies 5 patterns of UML use.**

***Index Terms*—UML, software development, software design, notation, empirical studies.**

I. Introduction: 'Where's the UML'?

The Unified Modeling Language (UML) has been heralded by many as "the lingua franca" (e.g., [23], [24]) or the "de facto standard" (e.g., Sjøberg, interviewed in [26], [6]) of software engineering. And yet there are others who argue that it is not fulfilling this role, because of issues such as size, complexity, semantics, consistency, and model transformation (e.g., [16], [8], [25]). Budgen et al. [6], in their systematic literature review of empirical evidence about UML, conclude that: "There is little to give confidence that the UML has really been evaluated as an artefact in its own right" and "There are few studies of adoption and use in the field" (p. 387). How exactly is UML being used in industry – is it, in practice, the universal notation that it is intended to be? This paper presents a corpus of interviews with professional software engineers about their UML "with rigor" (as he later expressed to the informant). In contrast, the informant concluded that probably 45 of the 47 were like him: "selective borrowers" … "who use some of the principles sometimes". The IBM speaker and the informant had very different models of what 'using UML' means in practice, with different implications.

Budgen et al. [6] point out that UML development has been guided more by expert opinion than by empirical evidence or cognitive theory. They call for "more and deeper studies of [UML's] longer term use in the field" (p. 387). The work reported here is based on the notion that understanding the nature of actual UML use is important to the discipline, and that understanding how software professionals 'use' UML can inform the development of software design notations and tools.

The study reported in this paper has its origins in a discrepancy of experience. After conducting empirical studies of software design in industry for decades, the author found recently that some of her papers on design representation were challenged by academic referees who asked: "Where's the UML?" Discussions at conferences such as ICSE and ESEC/FSE reinforced the discrepancy, with delegates surprised or even distrustful that the reported professional software design practice did not include use of UML. The response was predictable: to seek new evidence.

Why Not UML?

The majority of informants (35/50) do not use UML. One stated categorically that his global corporation “doesn’t use UML”. There is a consistent pattern to the responses concerning why practitioners do not use UML: **they have considered it, often having adopted it for a project or two, and found that it did not offer them advantages over their current, evolved practices and representations**. As one informant phrased it, and many others expressed: “What was good about UML was not new, and what was new about UML was not good.” Another identified where elements of UML notation originated pre-UML, arguing that use of those prior notations alone does not constitute UML use. Some felt that UML **came with too much ‘philosophy’ or ‘ideology’,** that it required them to make changes to their culture or ethos that were not warranted by adding benefits: “Why would I [use UML]? Doesn’t add anything except religion.”

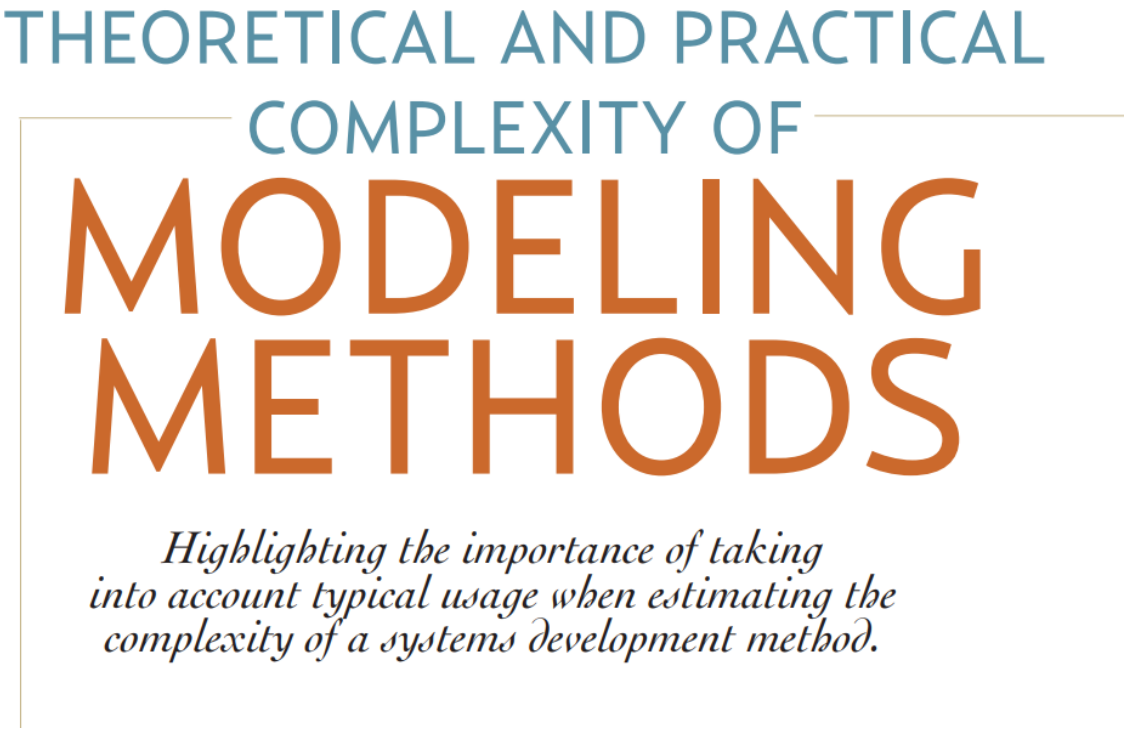
Selective Use of UML (11/50)

- UML as a thought tool
- For communication with stakeholders
  - “UML – mainly sequence diagrams – is useful for requirements elicitation with key stakeholders”
  - Also activity diagrams
- Collaborative dialogues: in teams and collaborative setups
  - “UML helps a lot when talking to [international] speakers – it provides a language bridge” allowing participants to “…resolve the ambiguity using lower language skills”
- Adapted version: “I adapt UML to the task.”

UML: Which Parts?

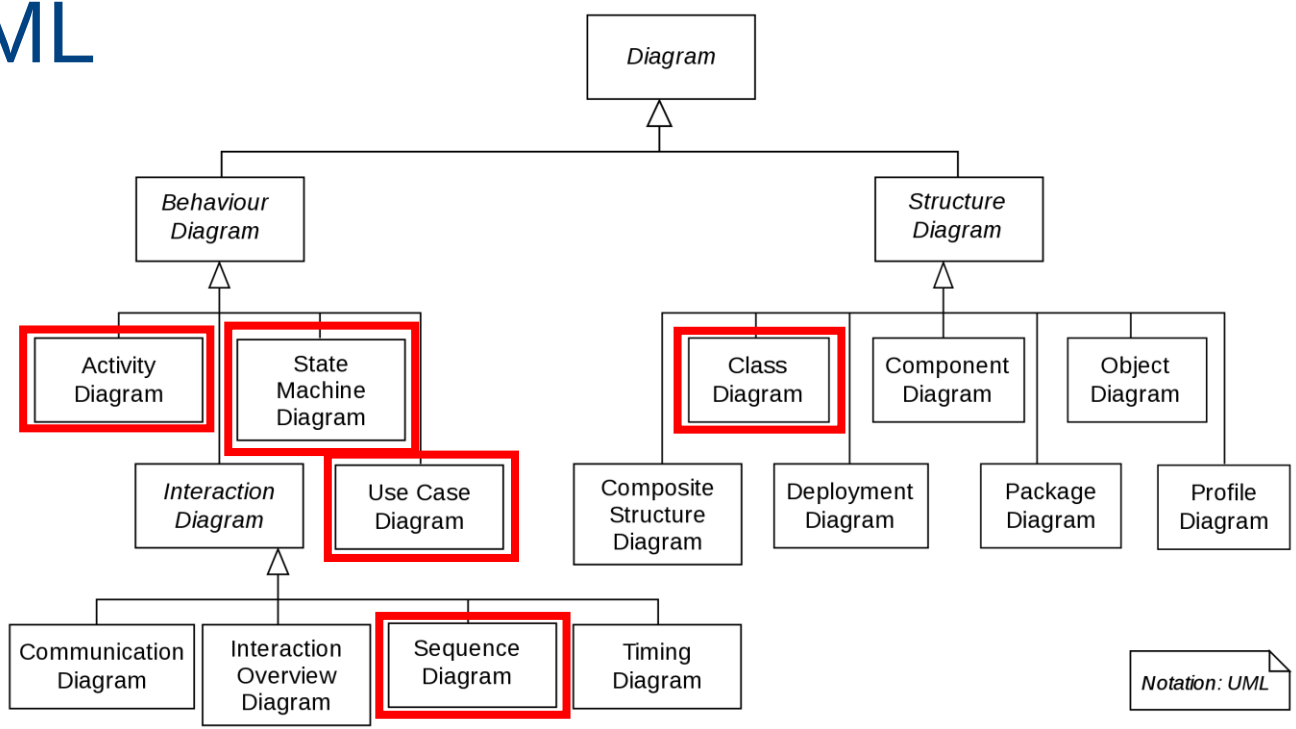
UML diagrams	Number of users	Reported to be used for…
Class diagrams	7	structure, conceptual models, concept analysis of domain, architecture, interfaces
Sequence diagrams	6	requirements elicitation, eliciting behaviors, instantiation history
Activity diagrams	6	modeling concurrency, eliciting useful behaviors, ordering processes
State machine diagrams	3	
Use case diagrams	1	represent requirements

UML: Which Parts?



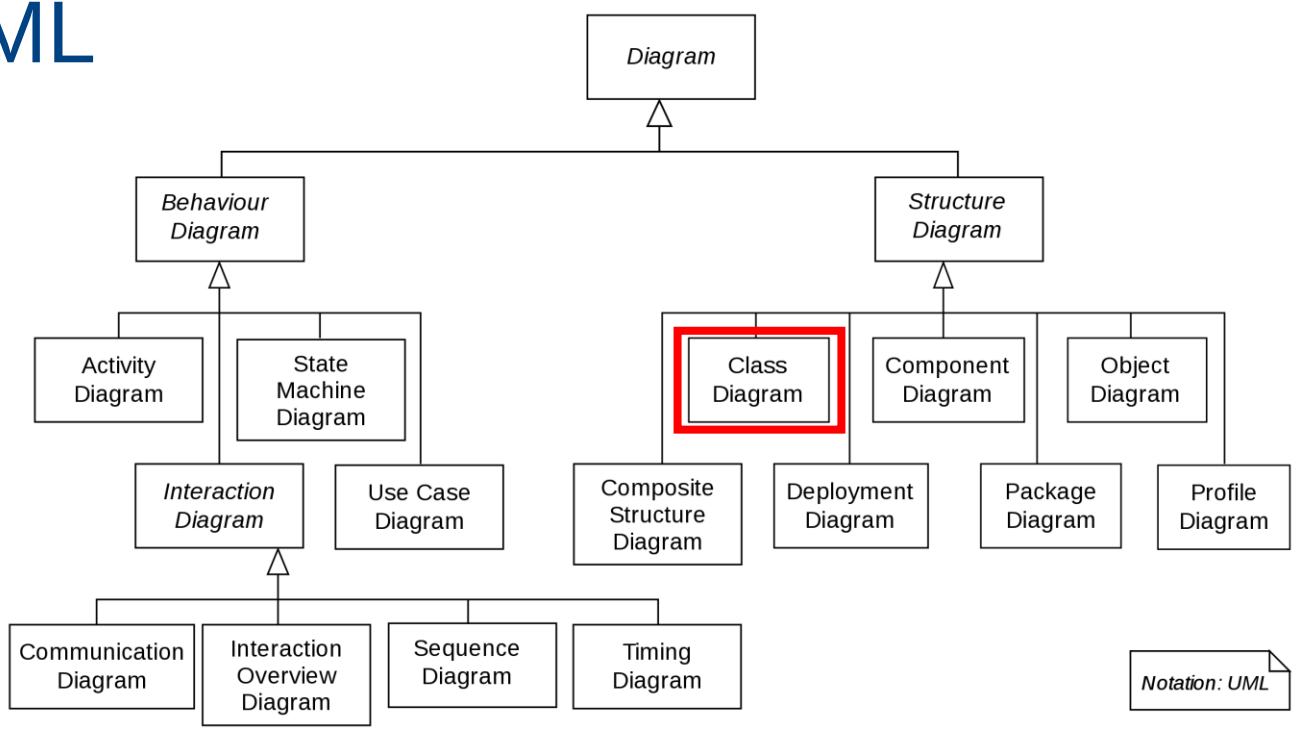
https://dl.acm.org/doi/pdf/10.1145/1278201.1278205

UML



https://en.wikipedia.org/wiki/Unified_Modeling_Language

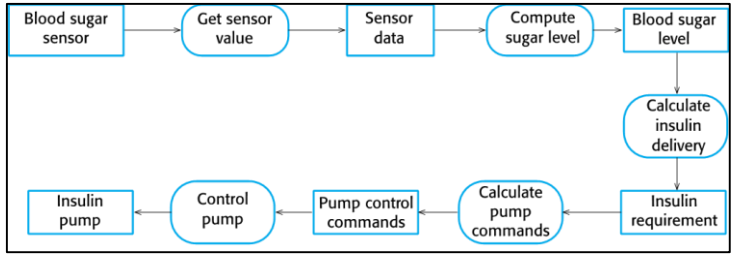
UML

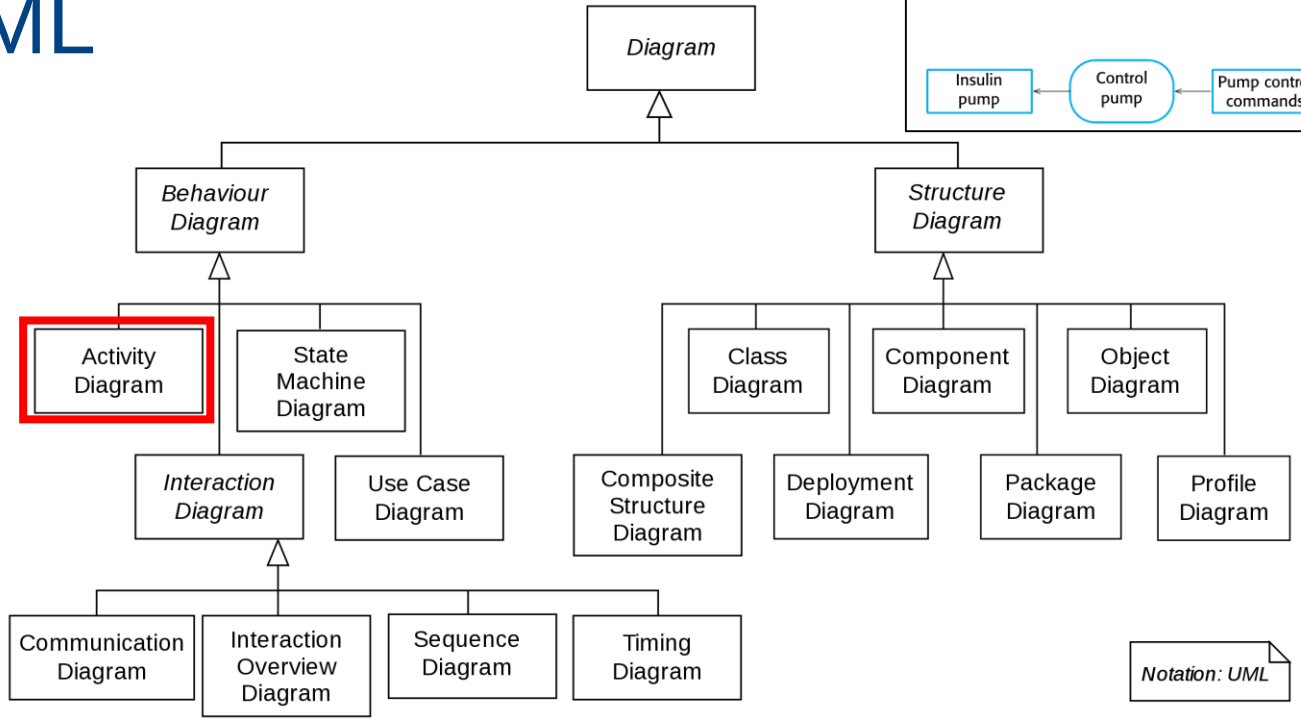


Consultation
Doctors Date Time Clinic Reason Medication prescribed Treatment prescribed Voice notes Transcript ...
New ( ) Prescribe ( ) RecordNotes ( ) Transcribe ( ) ...

https://en.wikipedia.org/wiki/Unified_Modeling_Language

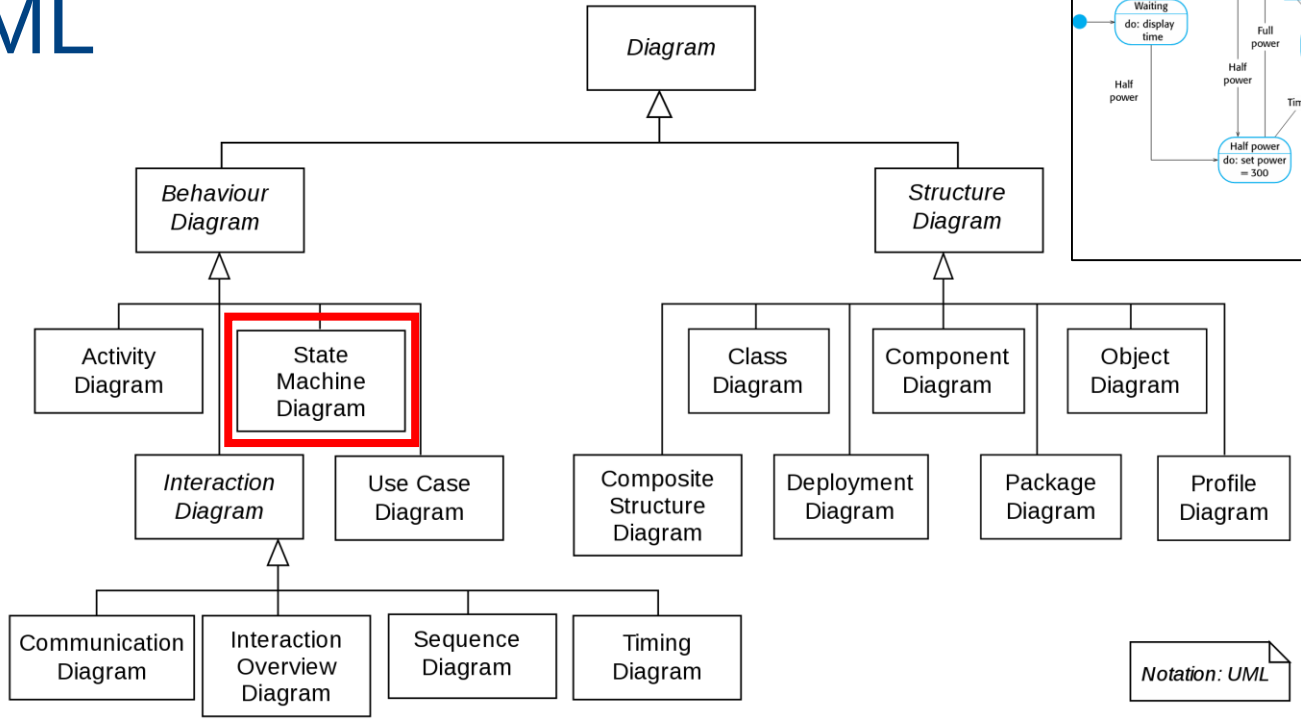
UML

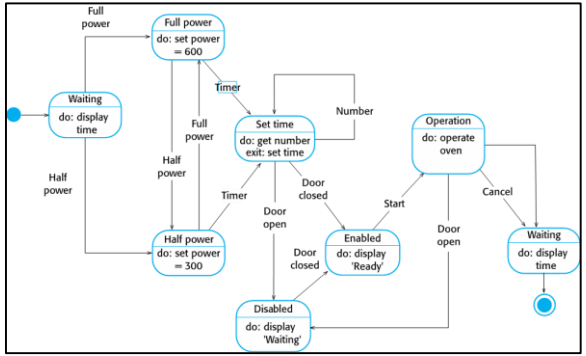




https://en.wikipedia.org/wiki/Unified_Modeling_Language

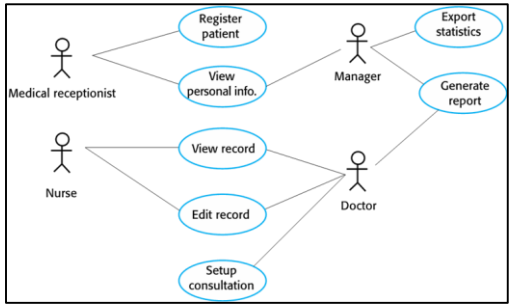
UML

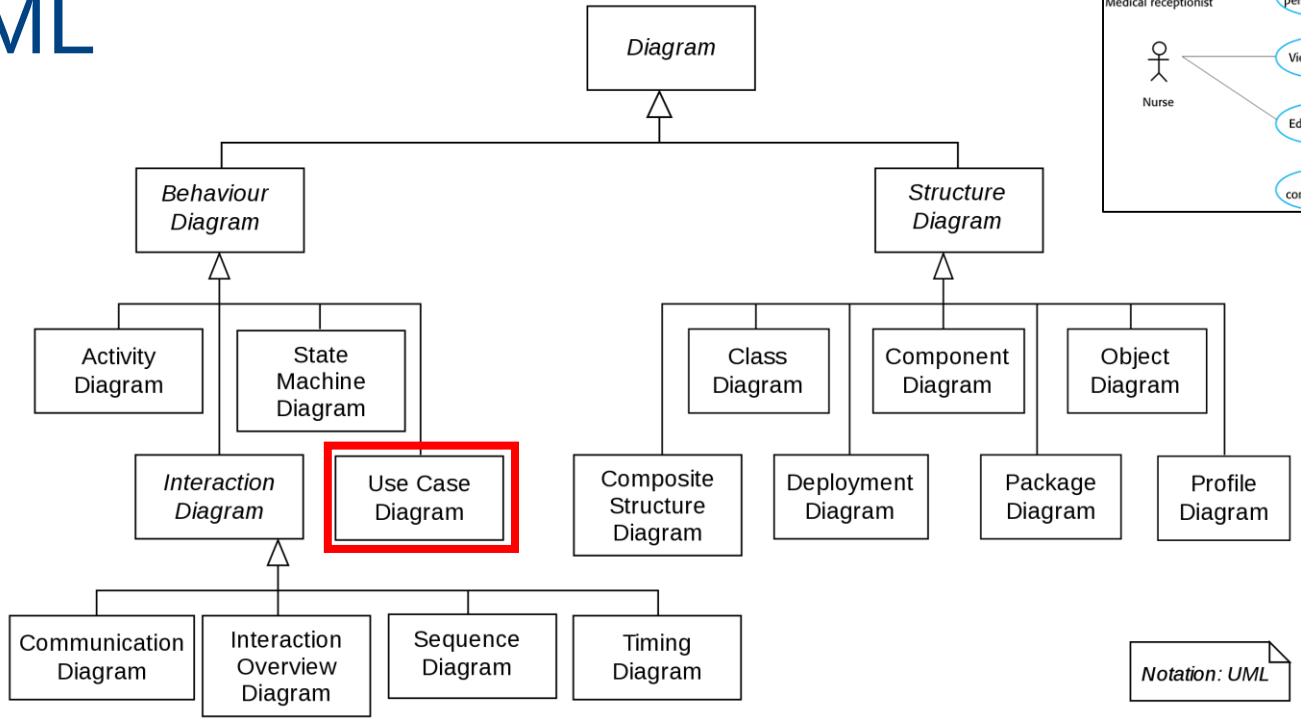




https://en.wikipedia.org/wiki/Unified_Modeling_Language

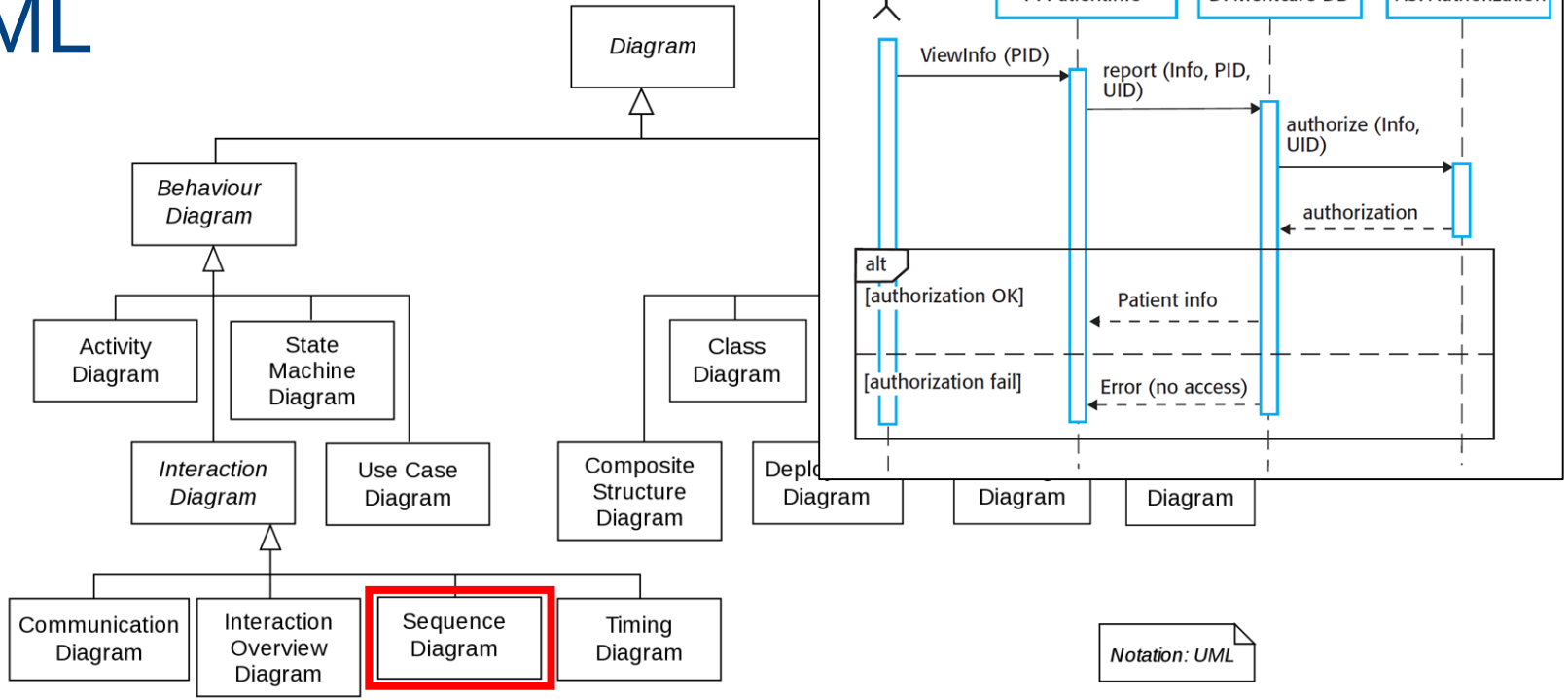
UML





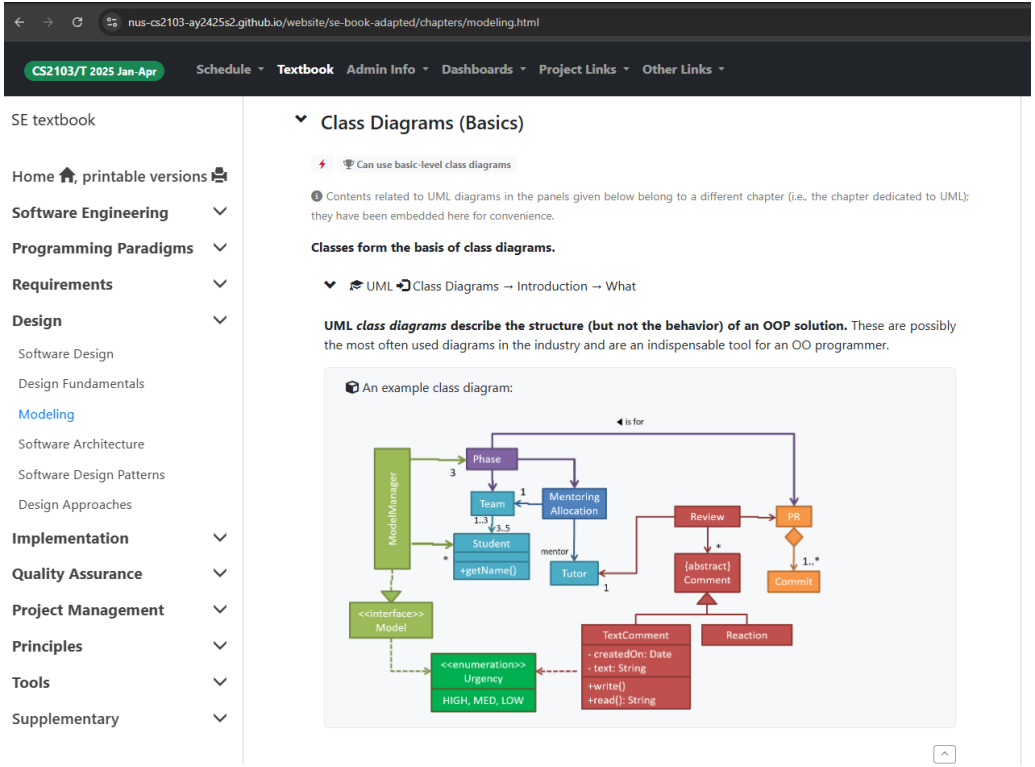
https://en.wikipedia.org/wiki/Unified_Modeling_Language

UML



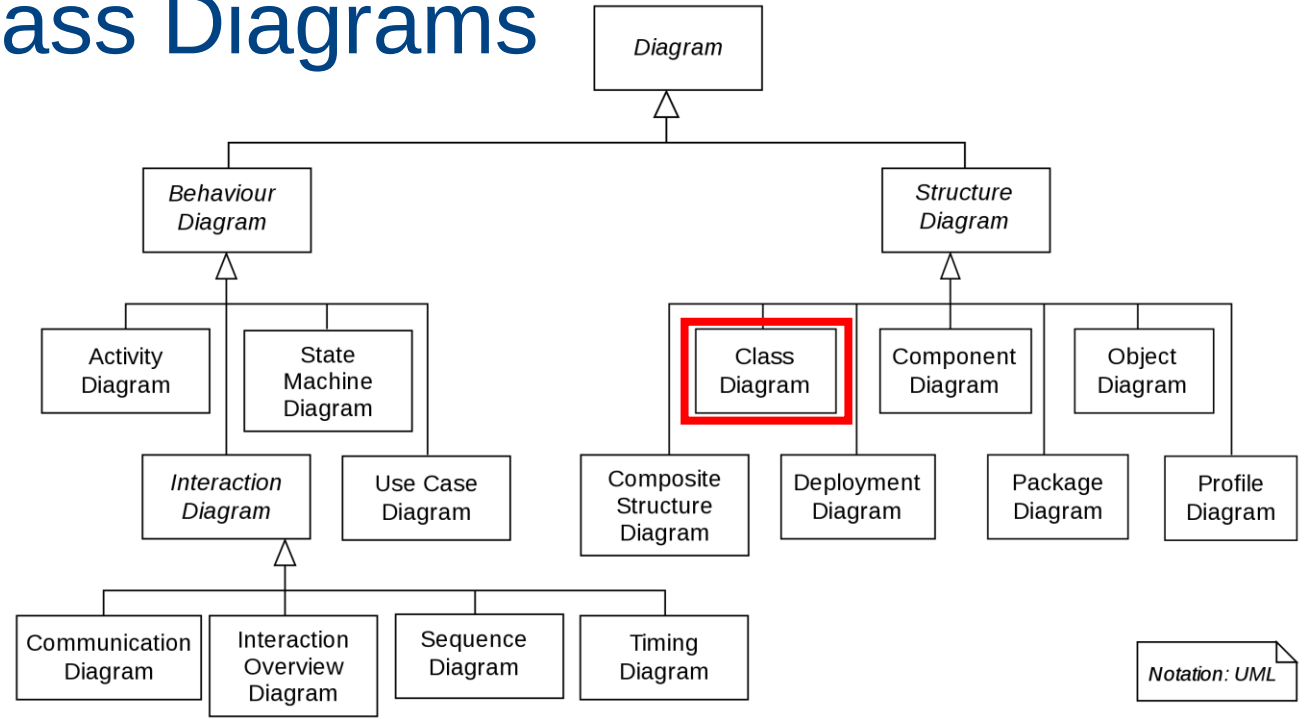
https://en.wikipedia.org/wiki/Unified_Modeling_Language

Key Diagrams



https://nus-cs2103-ay2425s2.github.io/website/se-book-adapted/chapters/modeling.html

Class Diagrams



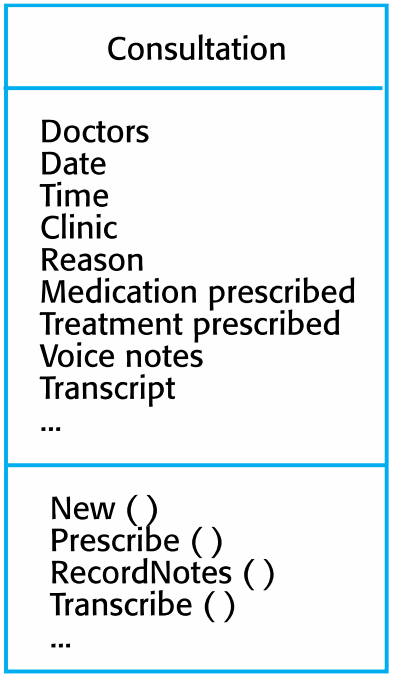
Consultation
Doctors Date Time Clinic Reason Medication prescribed Treatment prescribed Voice notes Transcript ...
New ( ) Prescribe ( ) RecordNotes ( ) Transcribe ( ) ...

https://en.wikipedia.org/wiki/Unified_Modeling_Language

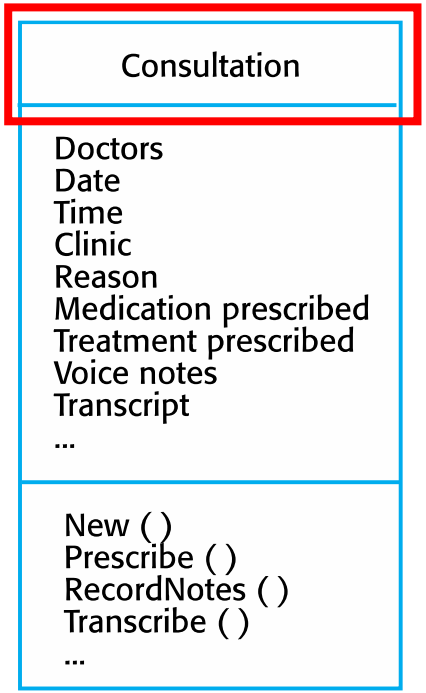
Important Concepts

- Classes
- Attributes
- Associations
- Generalization
- Constraints

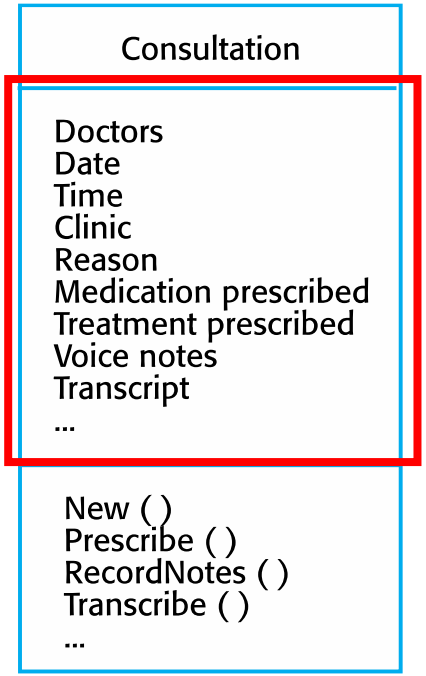
Consultation Class



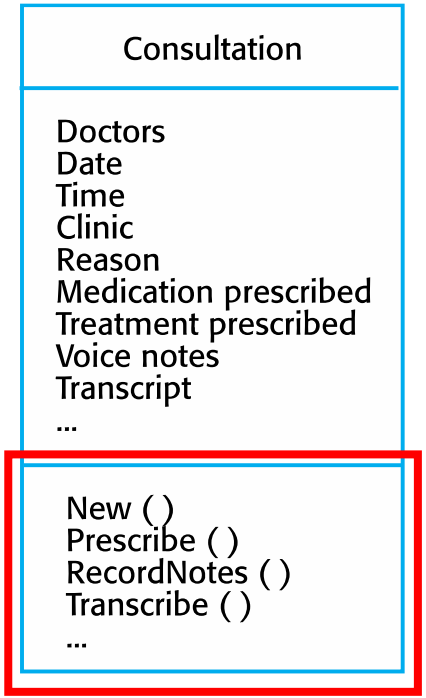
Consultation Class



Consultation Class



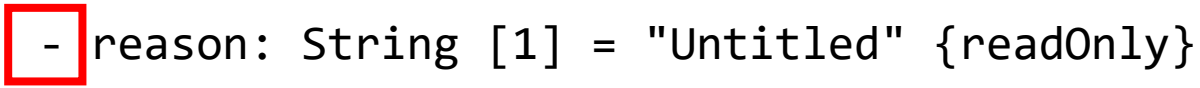
Consultation Class



Attributes

```
- reason: String [1] = "Untitled" {readOnly}
```

Attributes



Visibility: public (+) or private (-)

Attributes

```
- reason: String [1] = "Untitled" {readOnly}
```

Attribute name

Attributes

```
- reason: String [1] = "Untitled" {readOnly}
```

Attribute type

Attributes

```
- reason: String [1] = "Untitled" {readOnly}
```

Multiplicity

Attributes

```
- reason: String [1] = "Untitled" {readOnly}
```

Attribute type

Attributes

```
- reason: String [1] = "Untitled" {readOnly}
```

Attribute type

Operations

Prescribe (med: Medication) : boolean

Operations

Prescribe (med: Medication) : boolean

Operation name

Operations

Prescribe (med: Medication) : boolean

Operation parameters

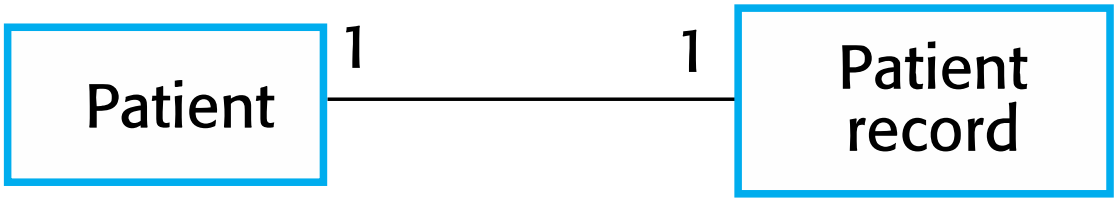
Operations

Prescribe (med: Medication) : boolean

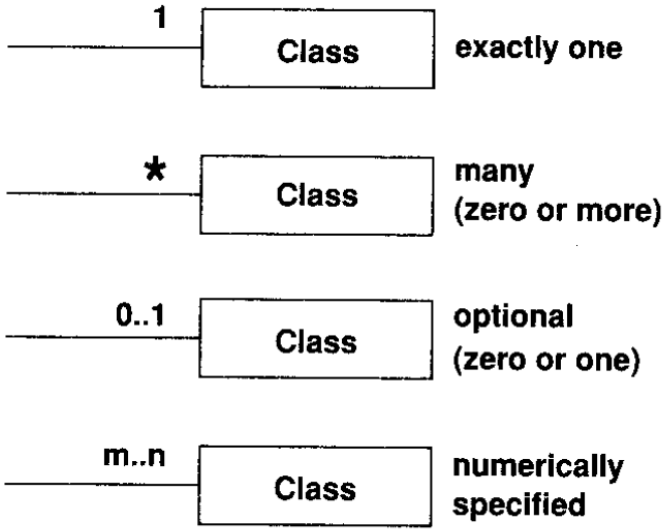
Return type

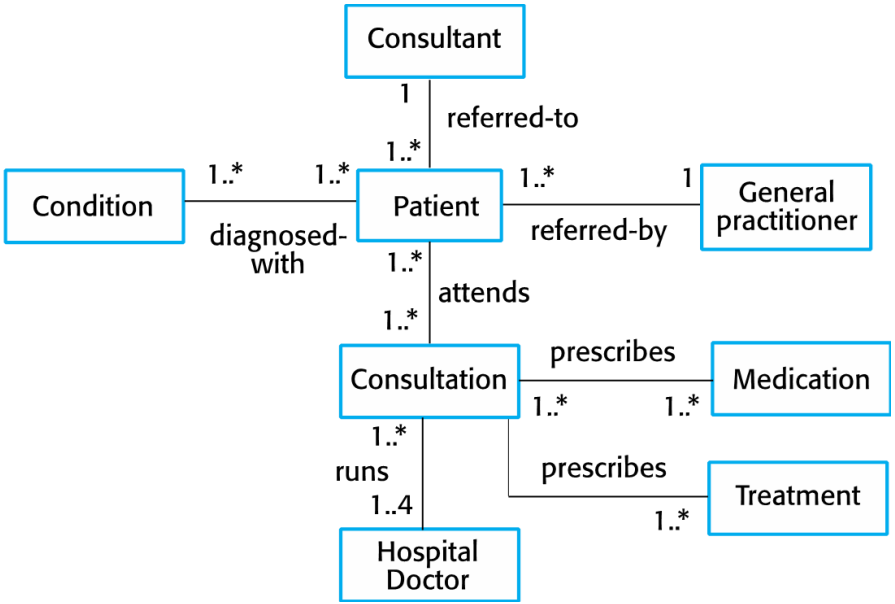
Classes and Association

- **Associations**: solid line between two classes
  - Multiplicities: how many objects may fill the property

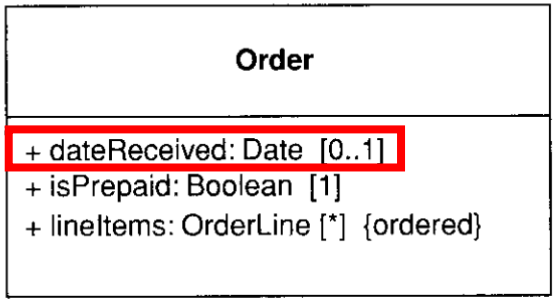


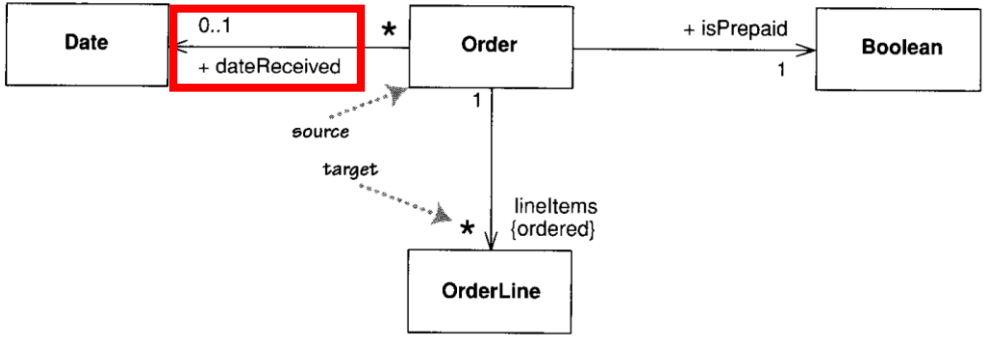
Associations and Multiplicities





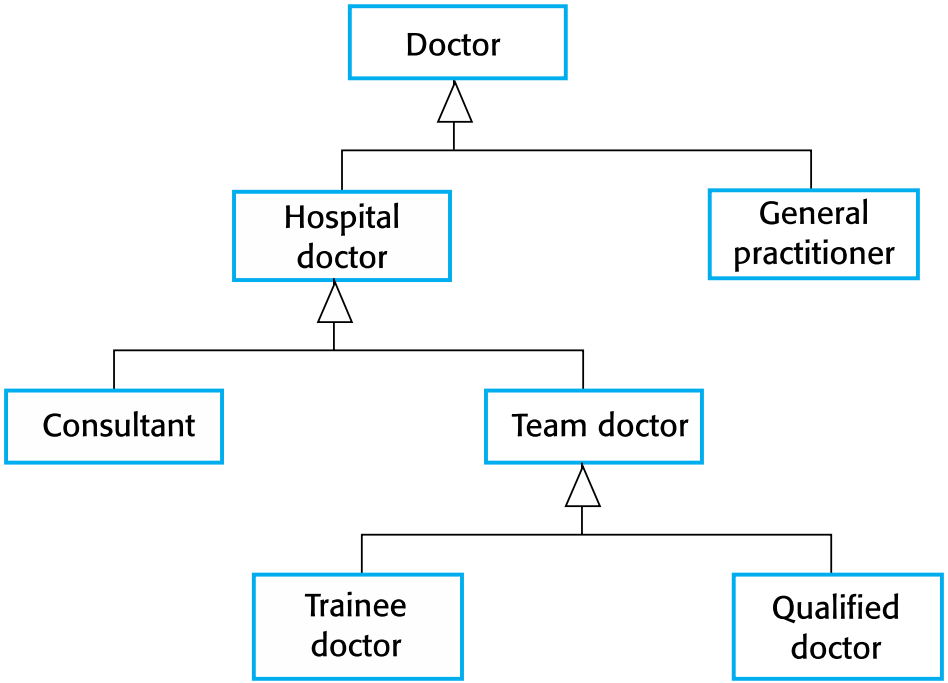
Properties: Attributes vs. Associations



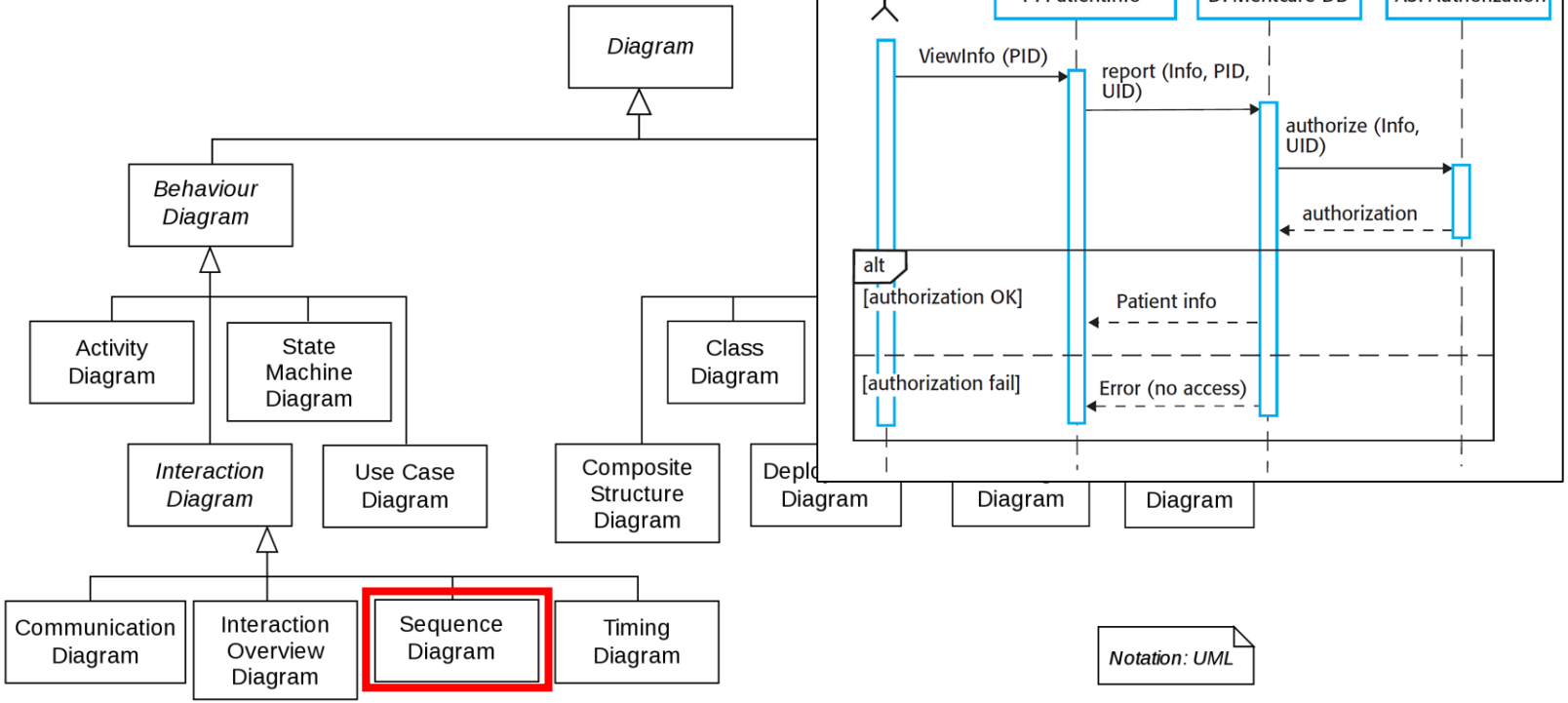


- Properties can often be shown both as attributes or associations
- Use attributes for things less important for the respective diagram

Generalization

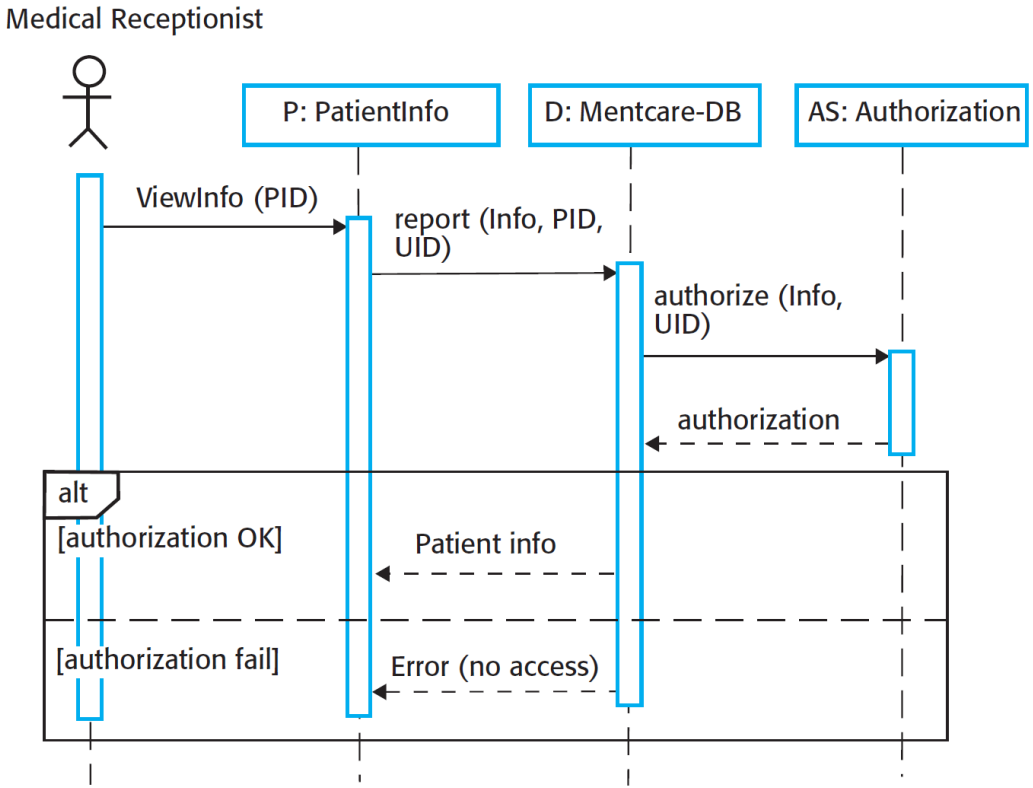


Sequence Diagrams



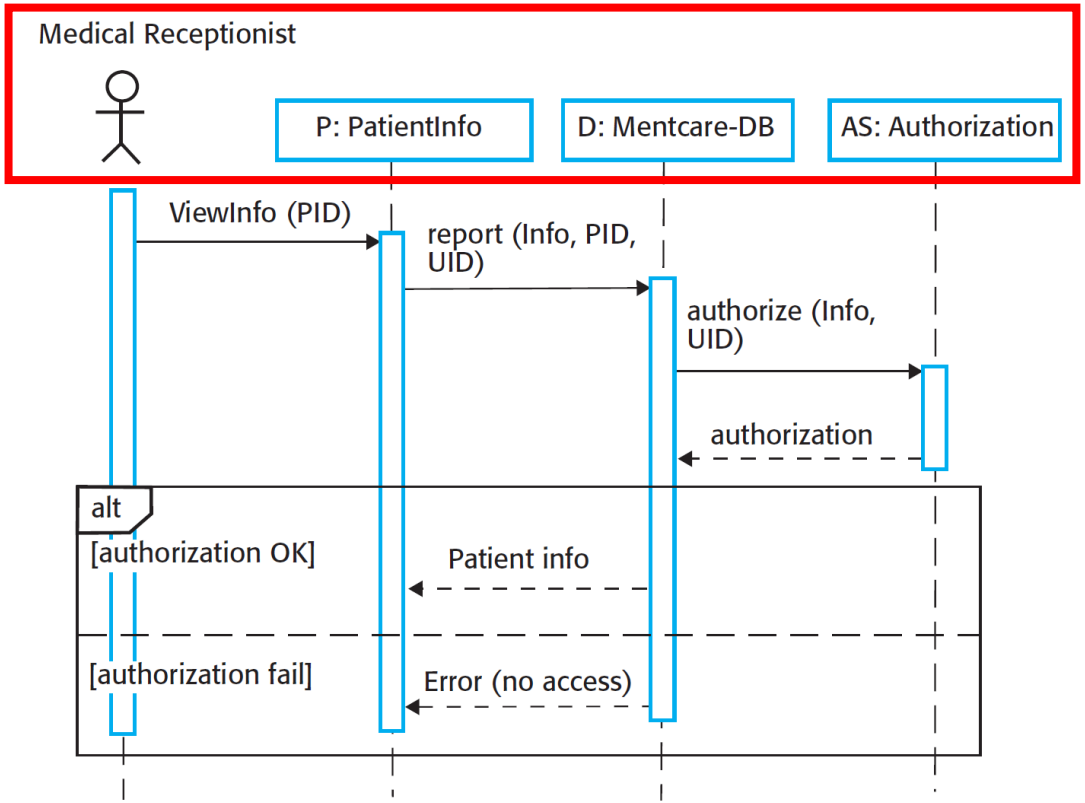
https://en.wikipedia.org/wiki/Unified_Modeling_Language

Sequence Diagrams



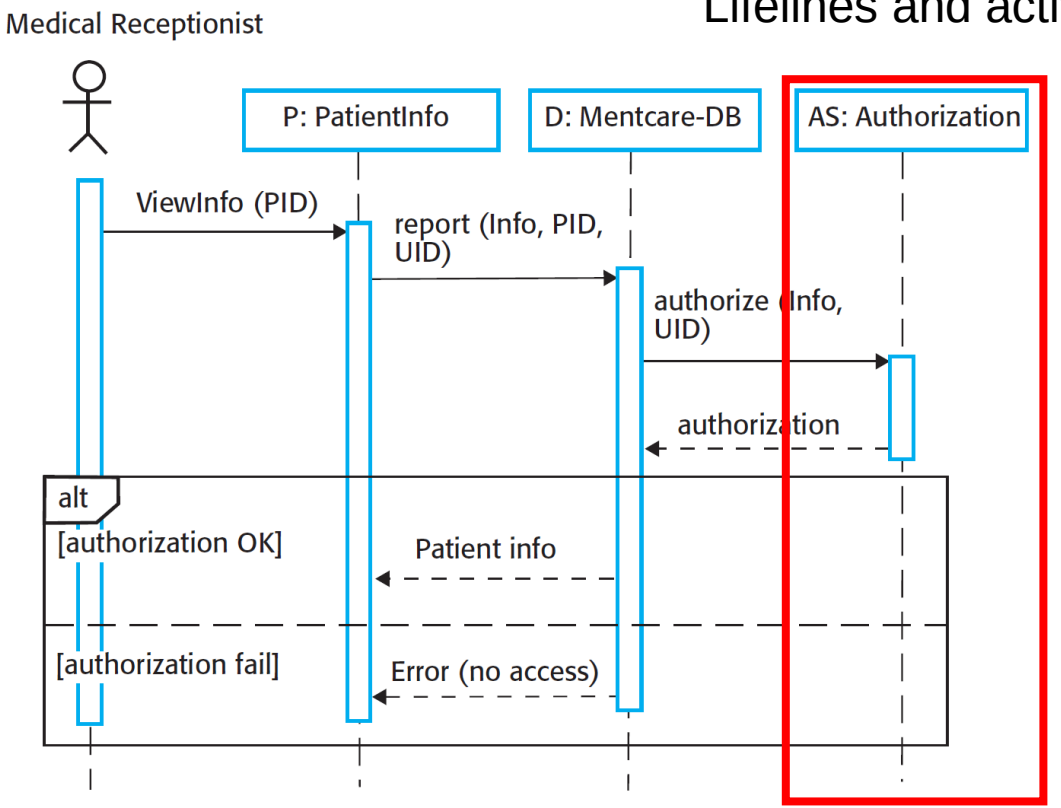
Sequence Diagrams

Objects or actors

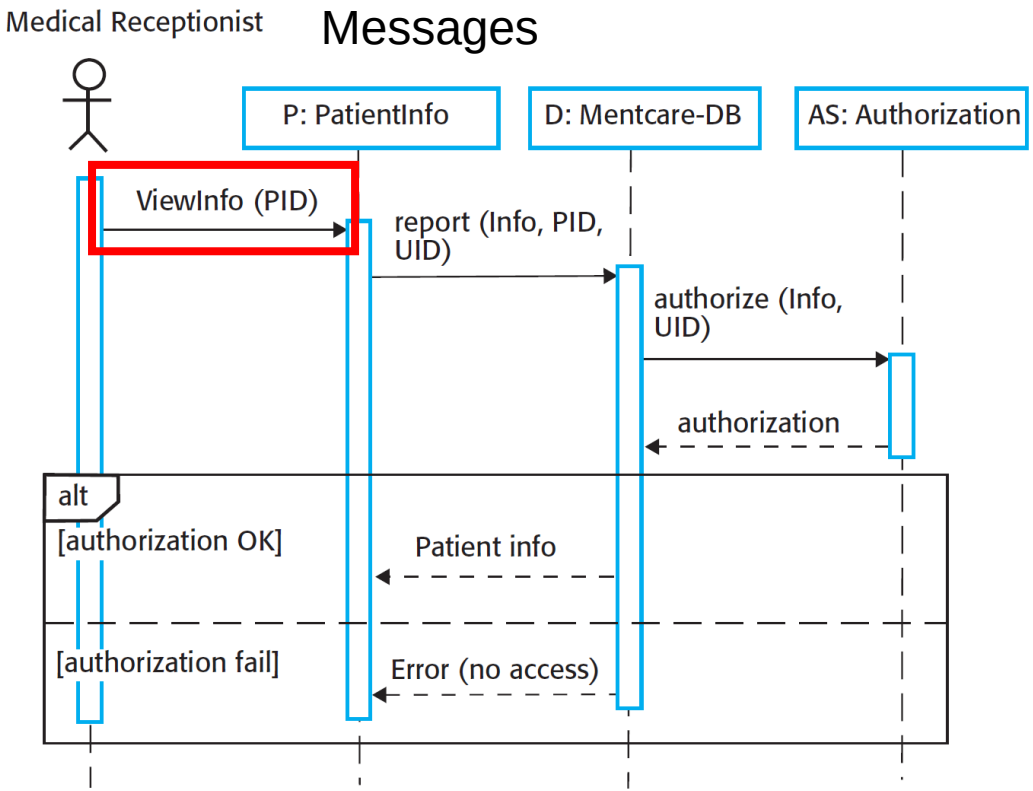


Sequence Diagrams

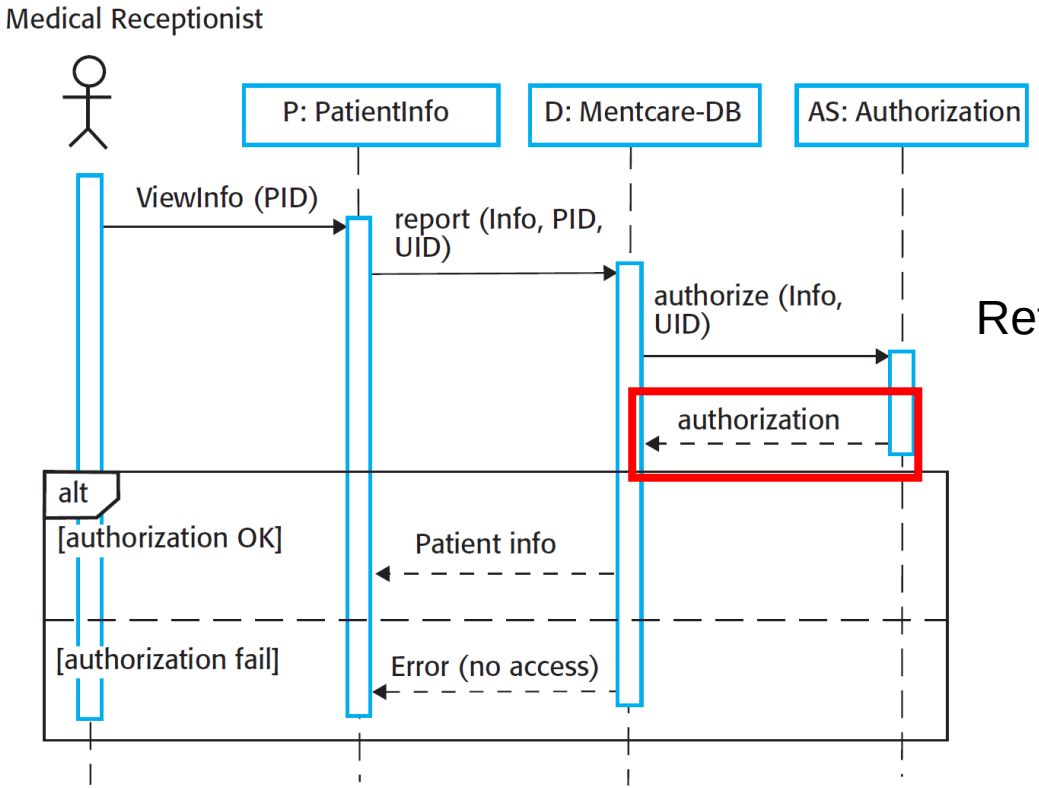
Lifelines and activation bars



Sequence Diagrams



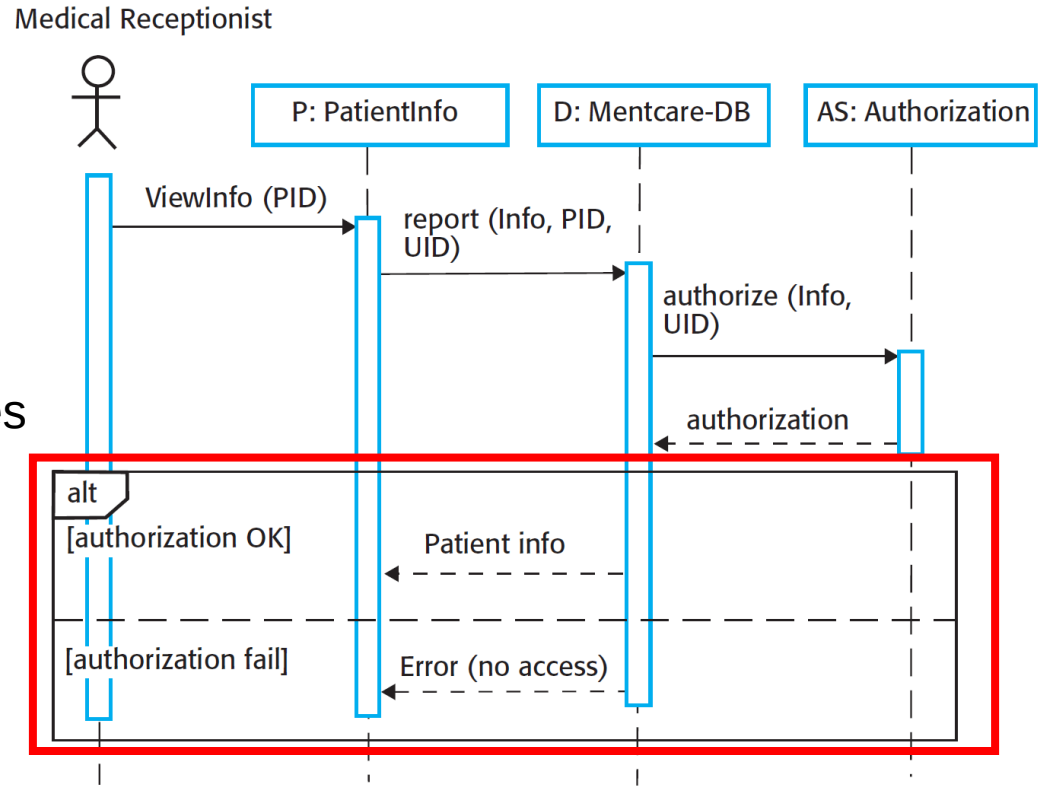
Sequence Diagrams



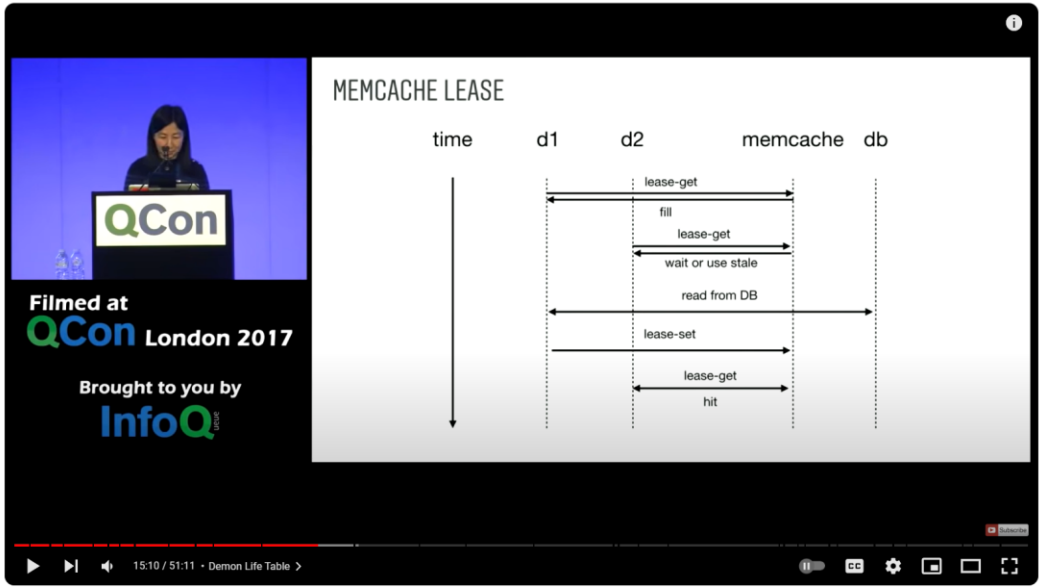
Return message

Sequence Diagrams

Alternatives

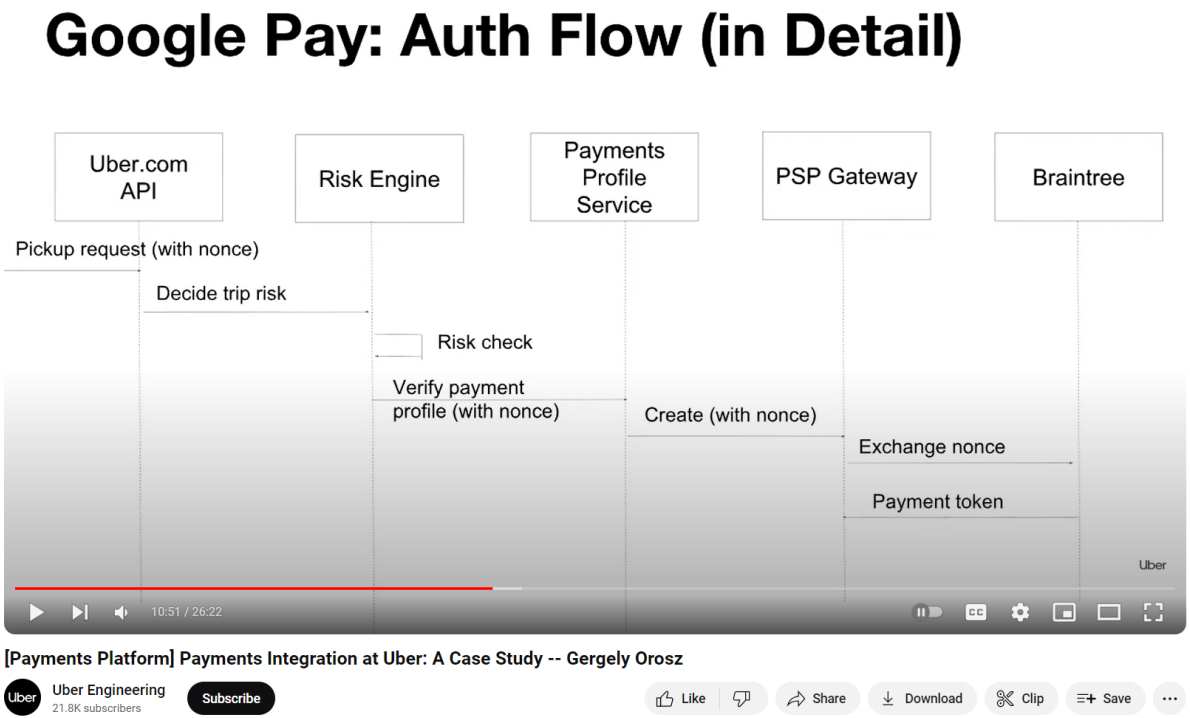


Sequence Diagrams in Practice



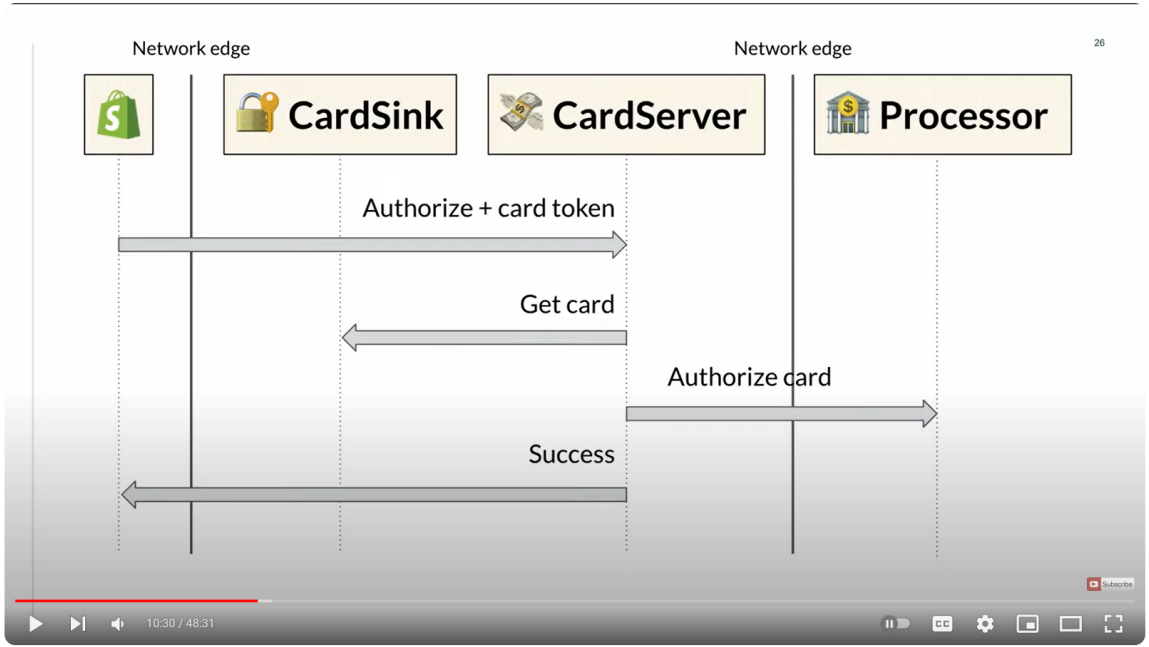
https://www.youtube.com/watch?v=hnpzNAPiC0E

Sequence Diagrams in Practice



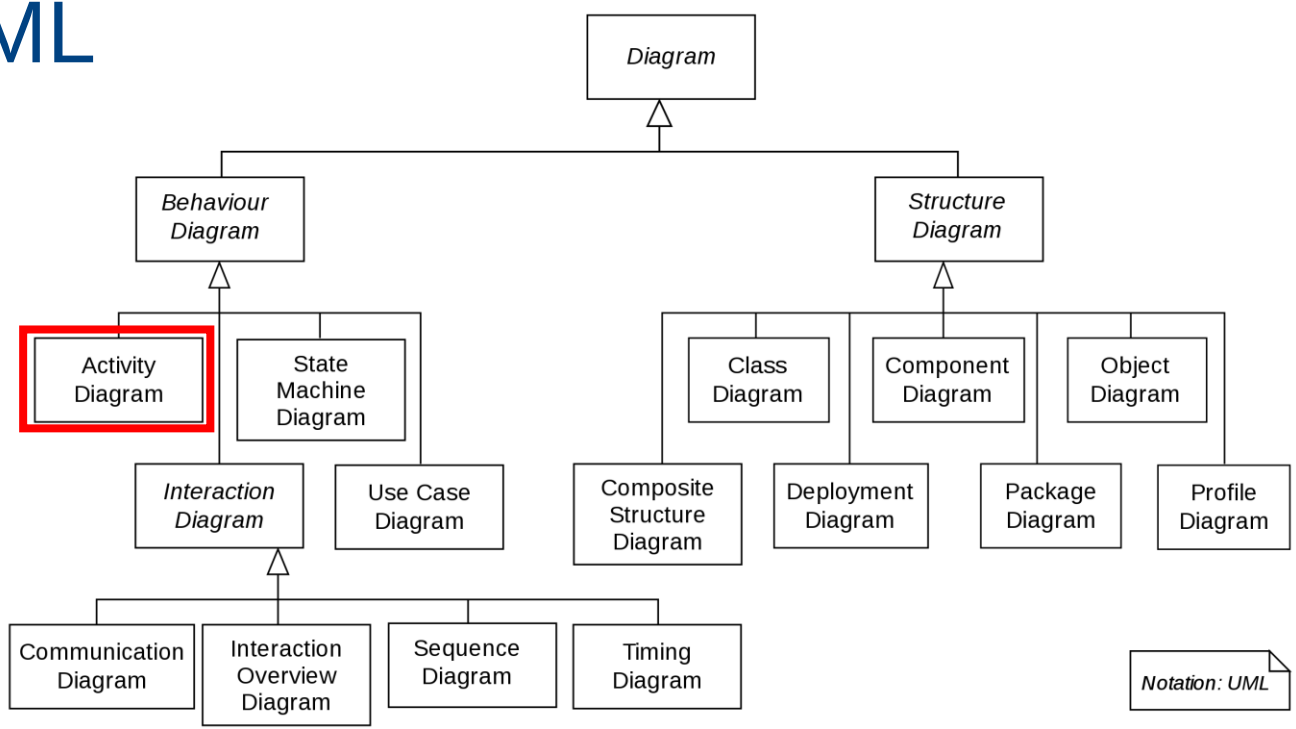
https://www.youtube.com/watch?v=yooCE5B0SRA

Sequence Diagrams in Practice



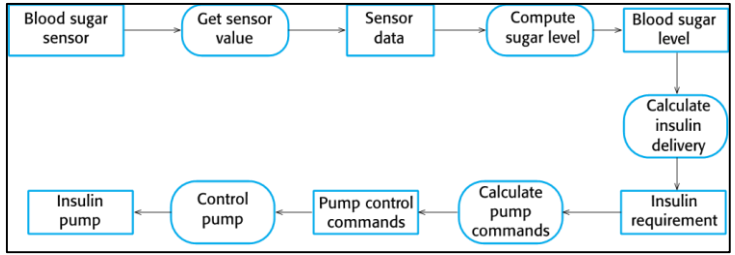
https://www.youtube.com/watch?v=MV5Kdwzwcag

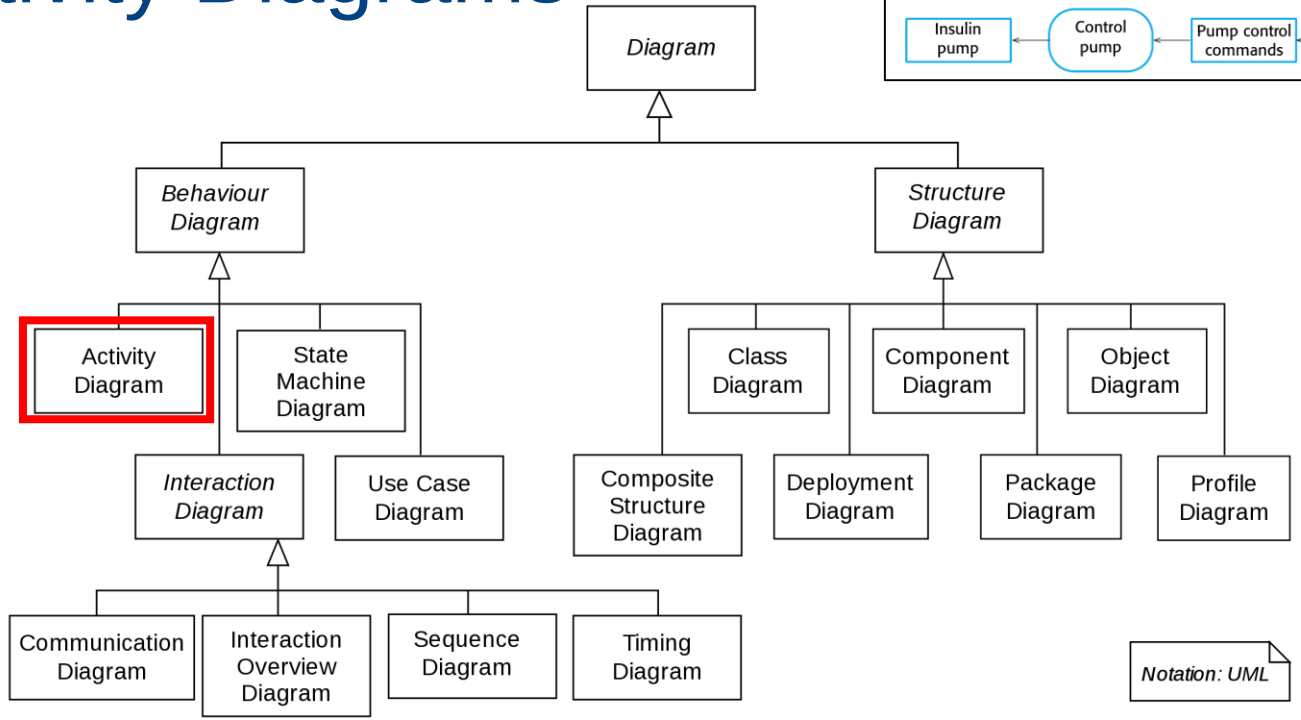
UML



https://en.wikipedia.org/wiki/Unified_Modeling_Language

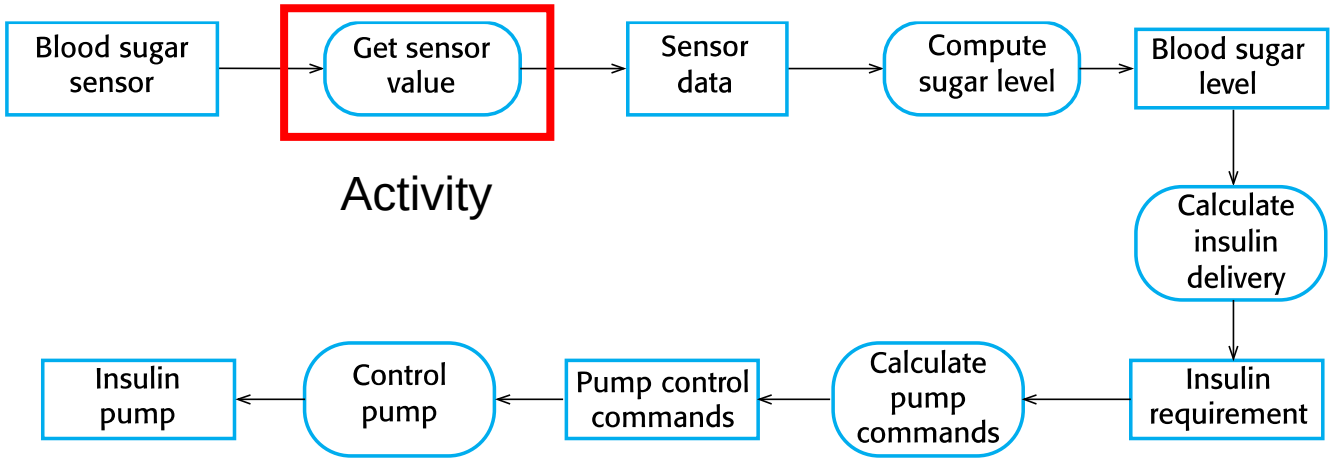
Activity Diagrams



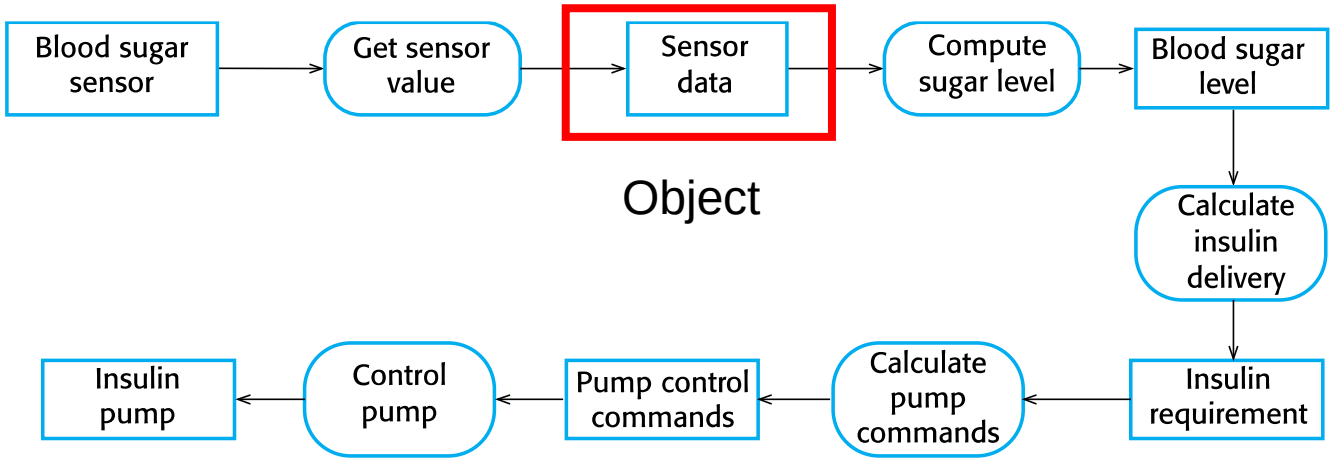


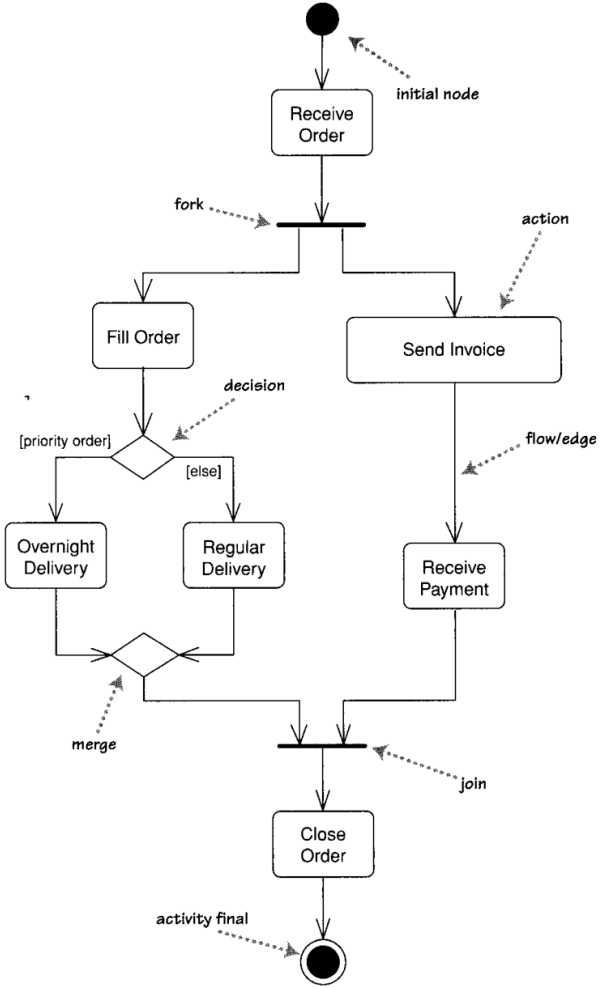
https://en.wikipedia.org/wiki/Unified_Modeling_Language

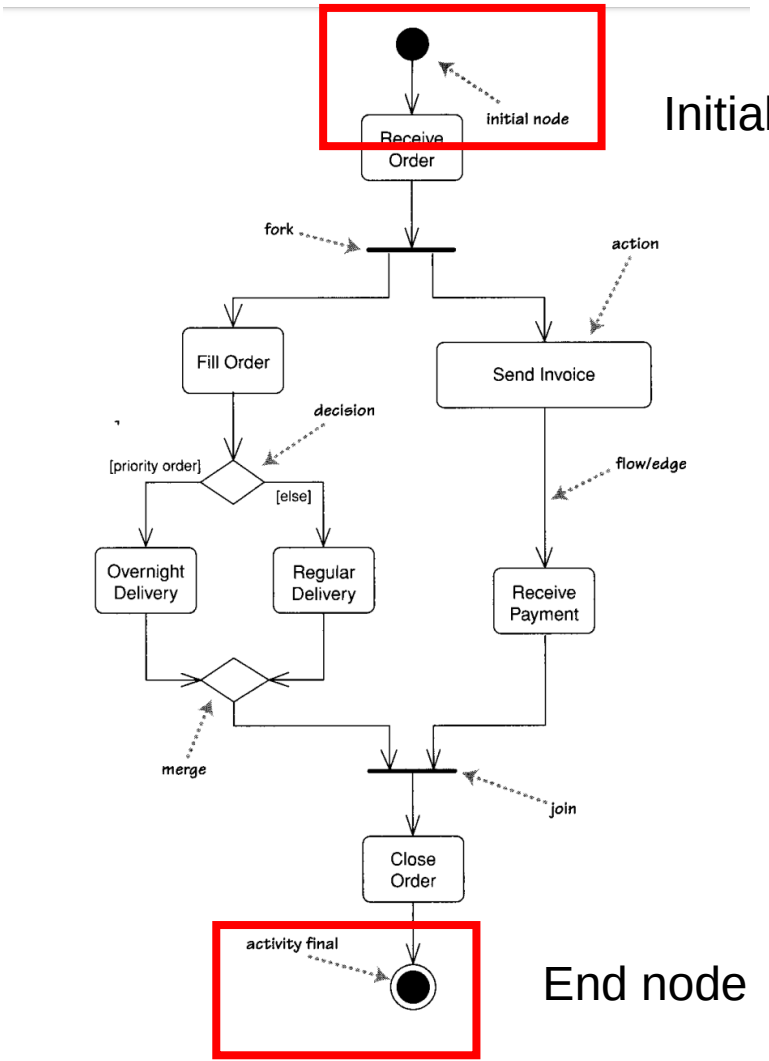
Activity Diagram



Activity Diagram

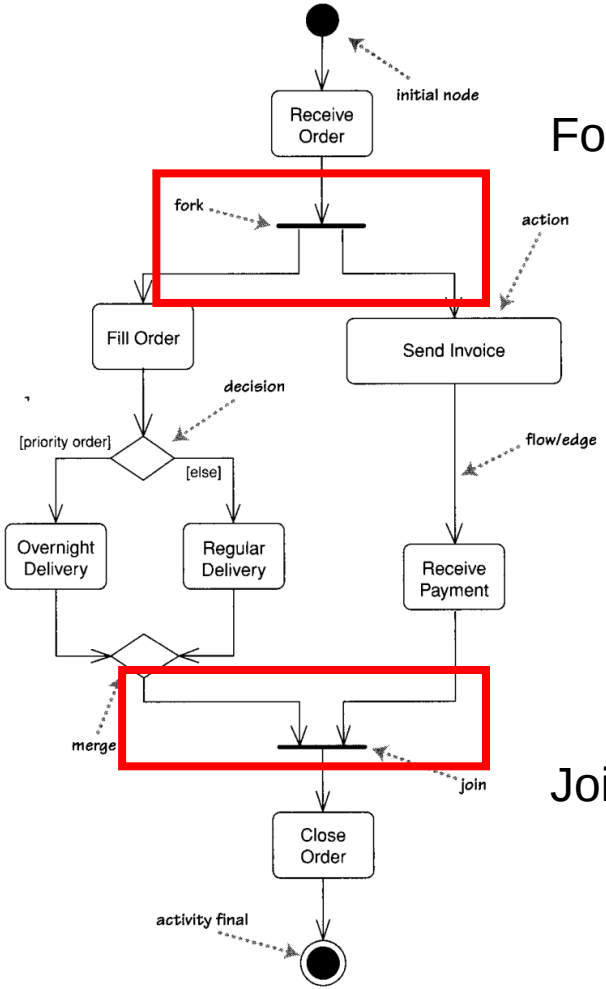



initial node
Receive Order
fork
action
Fill Order
Send Invoice
decision
[priority order]
[else]
flow/edge
Overnight Delivery
Regular Delivery
Receive Payment
merge
join
Close Order
activity final



Initial node

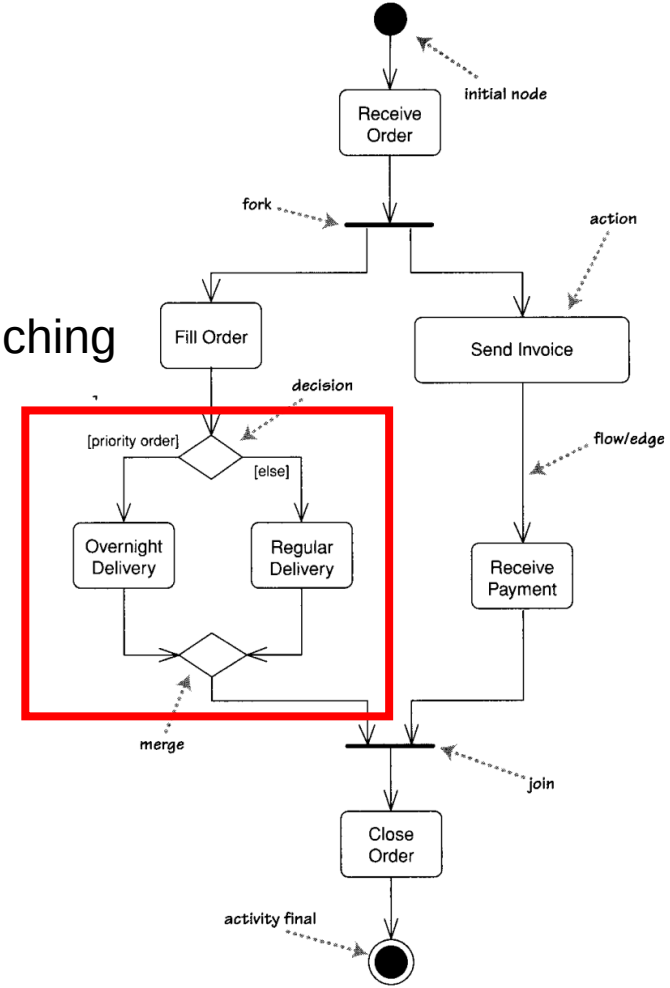
End node


initial node
Receive Order
fork
action
Fill Order
Send Invoice
decision
[priority order]
[else]
flow/edge
Overnight Delivery
Regular Delivery
Receive Payment
merge
join
Close Order
activity final

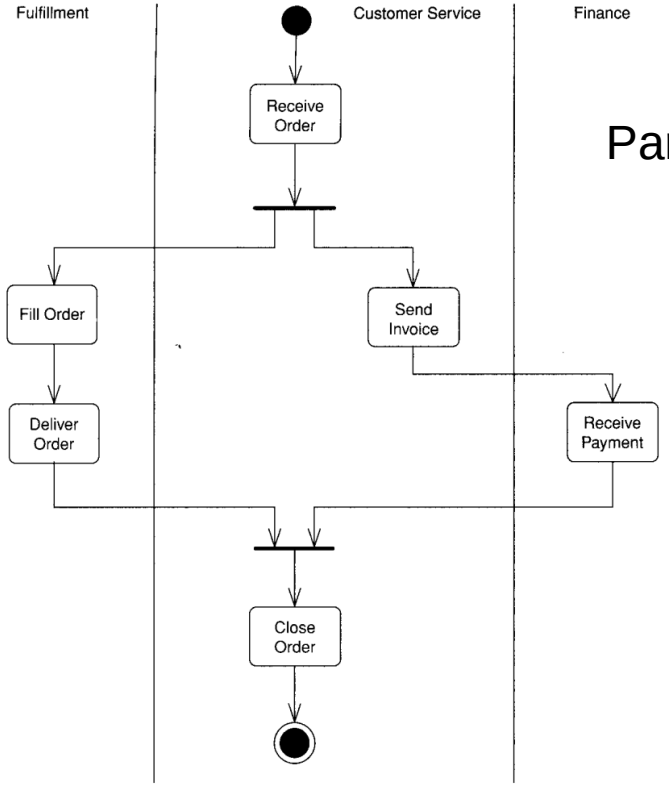
Fork

Join

Branching

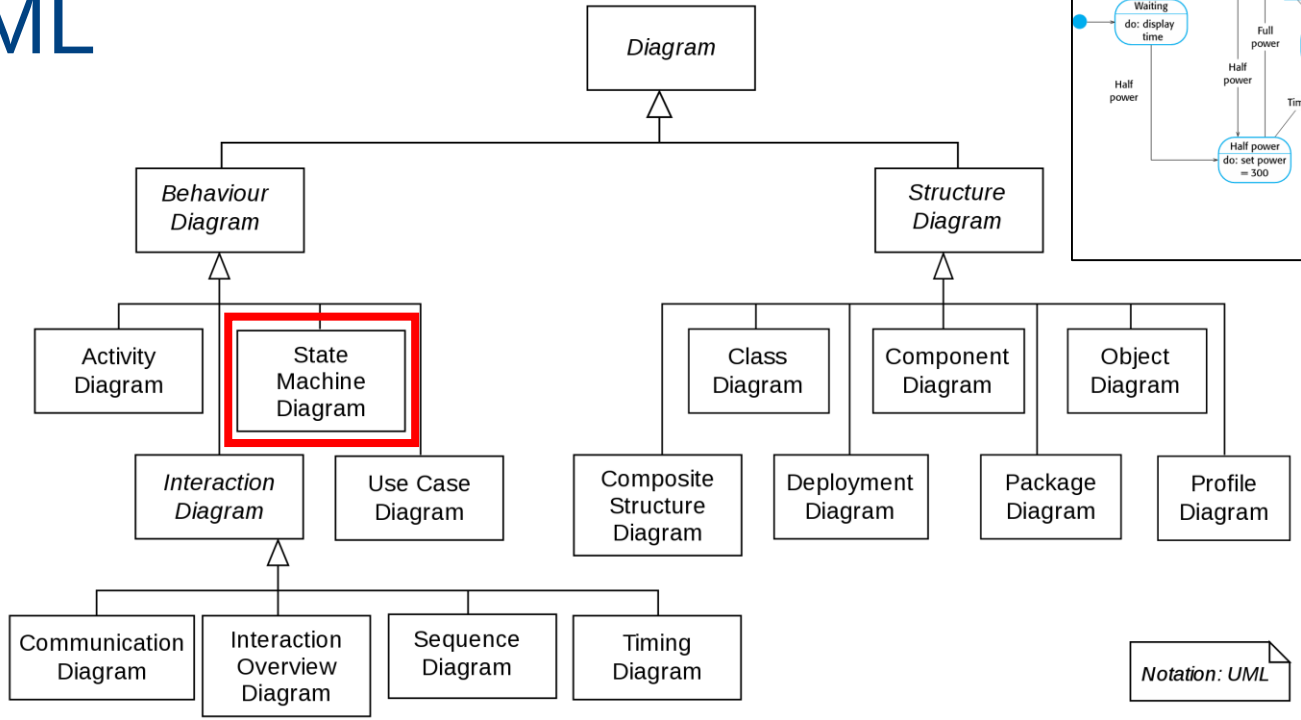

initial node
Receive Order
fork
action
Fill Order
Send Invoice
decision
[priority order]
[else]
flow/edge
Overnight Delivery
Regular Delivery
Receive Payment
merge
join
Close Order
activity final

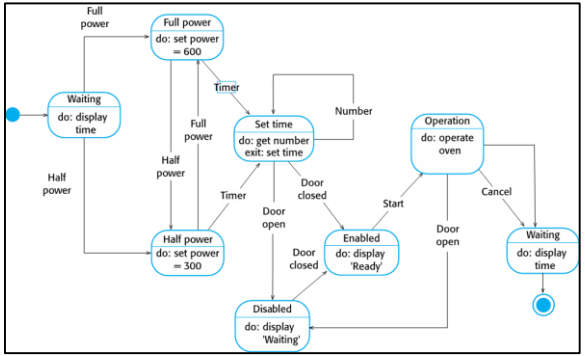
Activity Diagrams: Partitioning



Partitioning/swim lanes

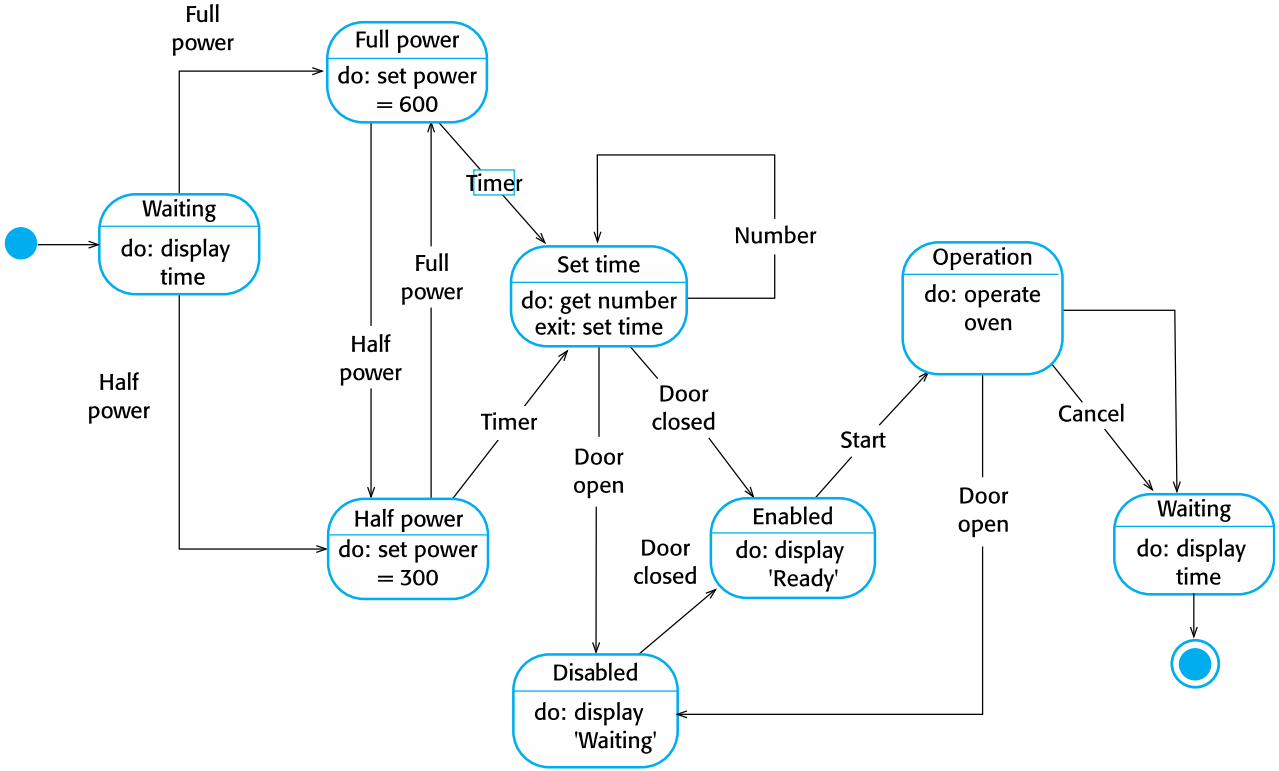
UML



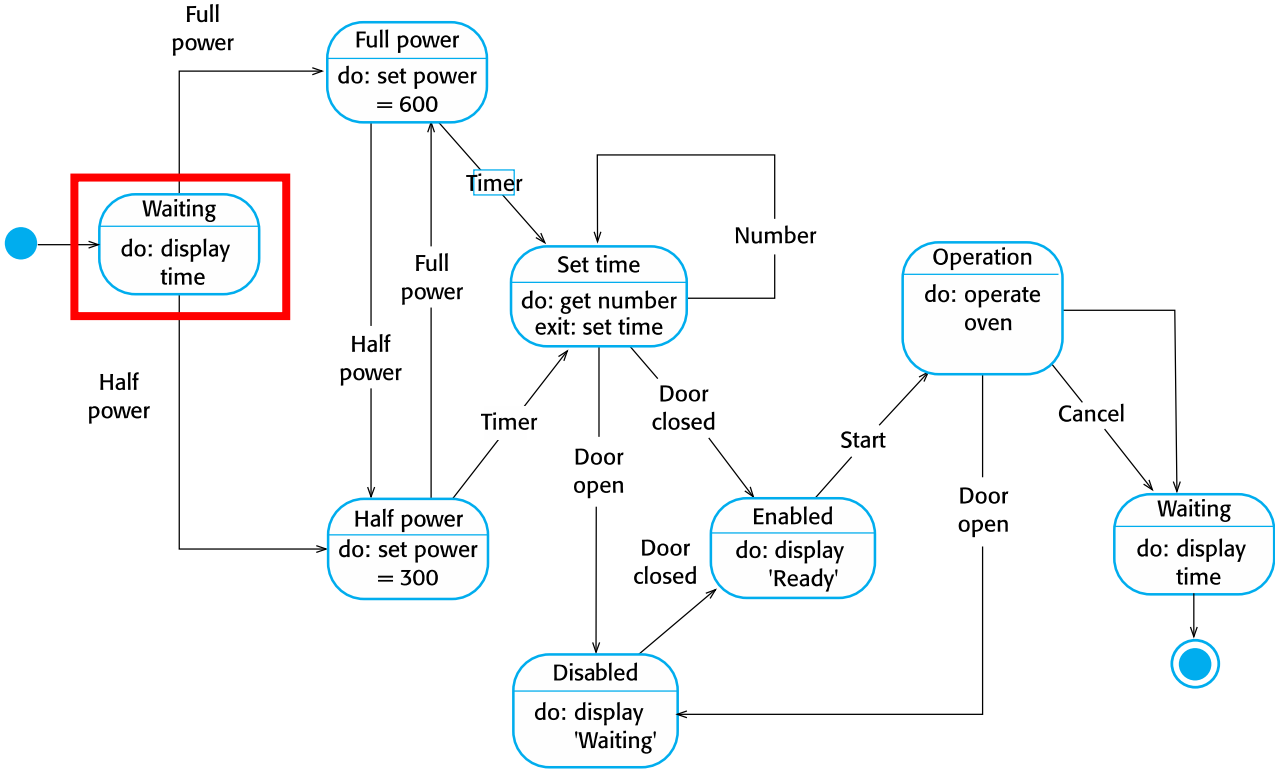


https://en.wikipedia.org/wiki/Unified_Modeling_Language

State Machine Diagram



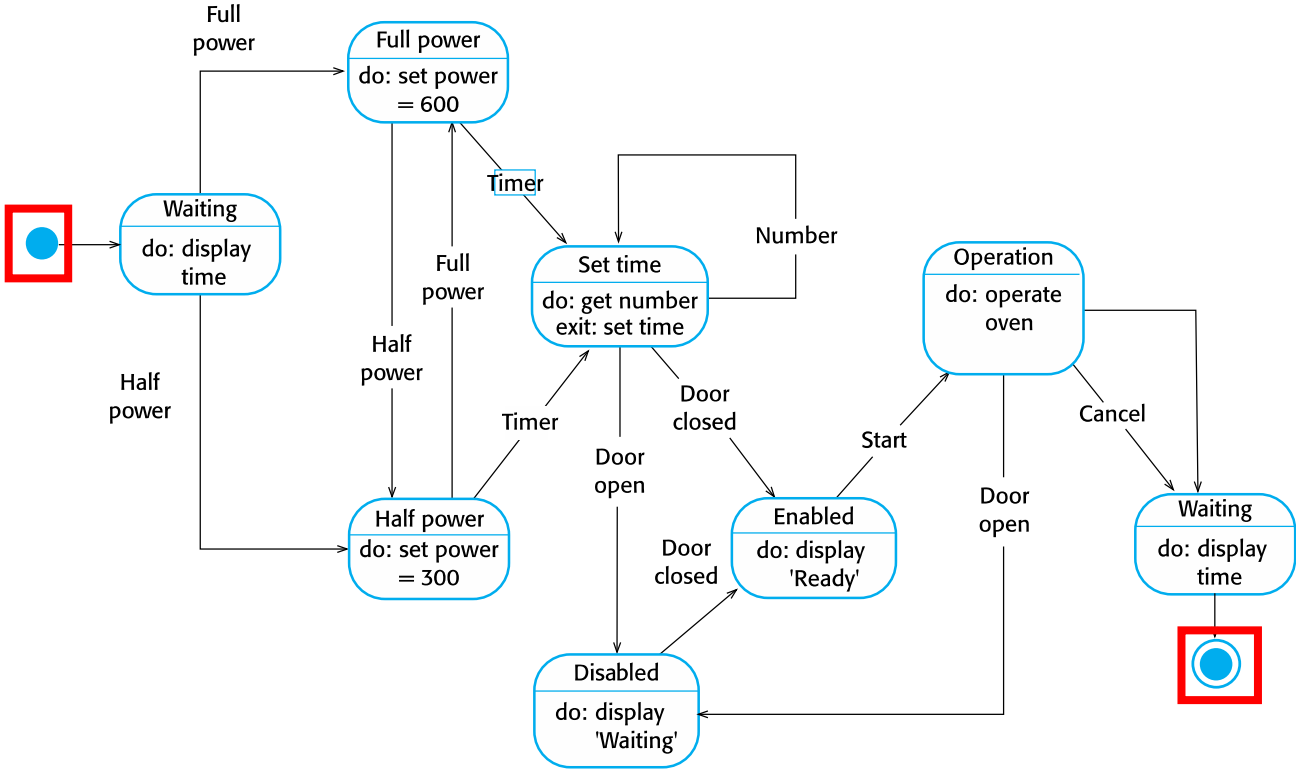
State Machine Diagram



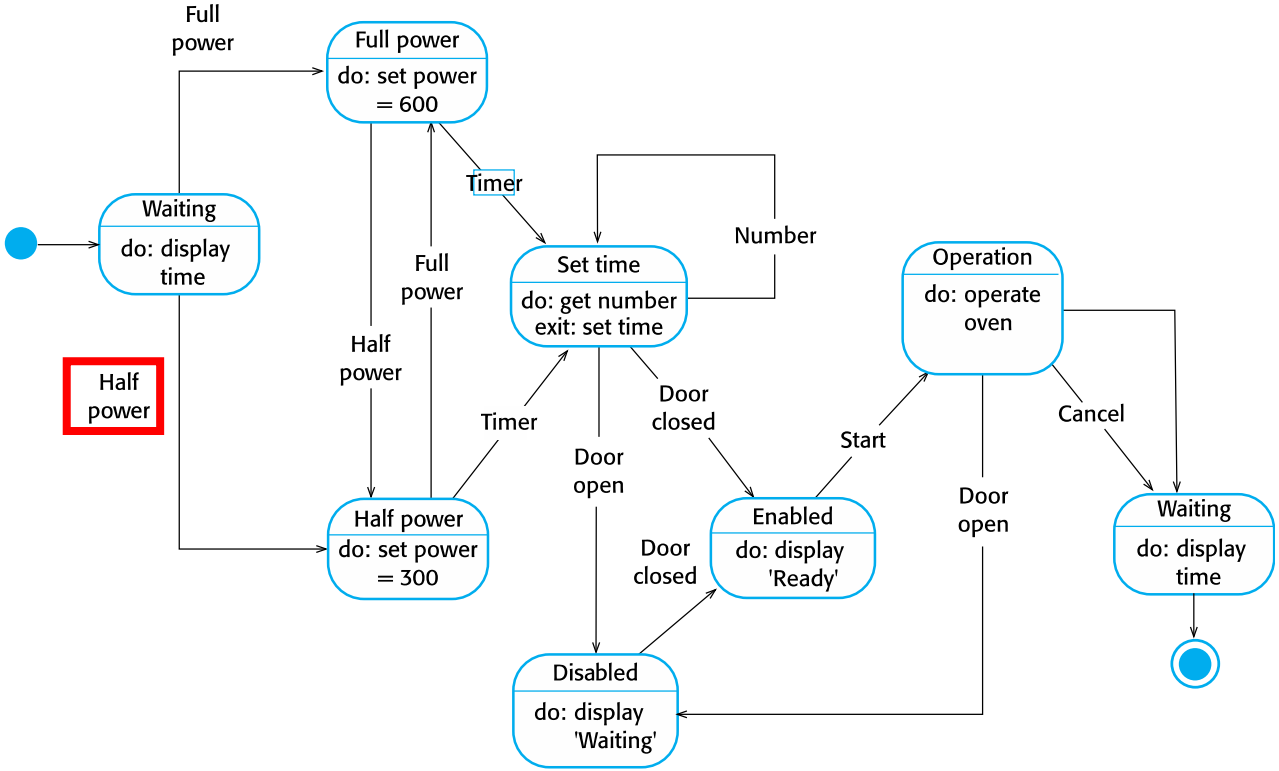
States

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.

State Machine Diagram



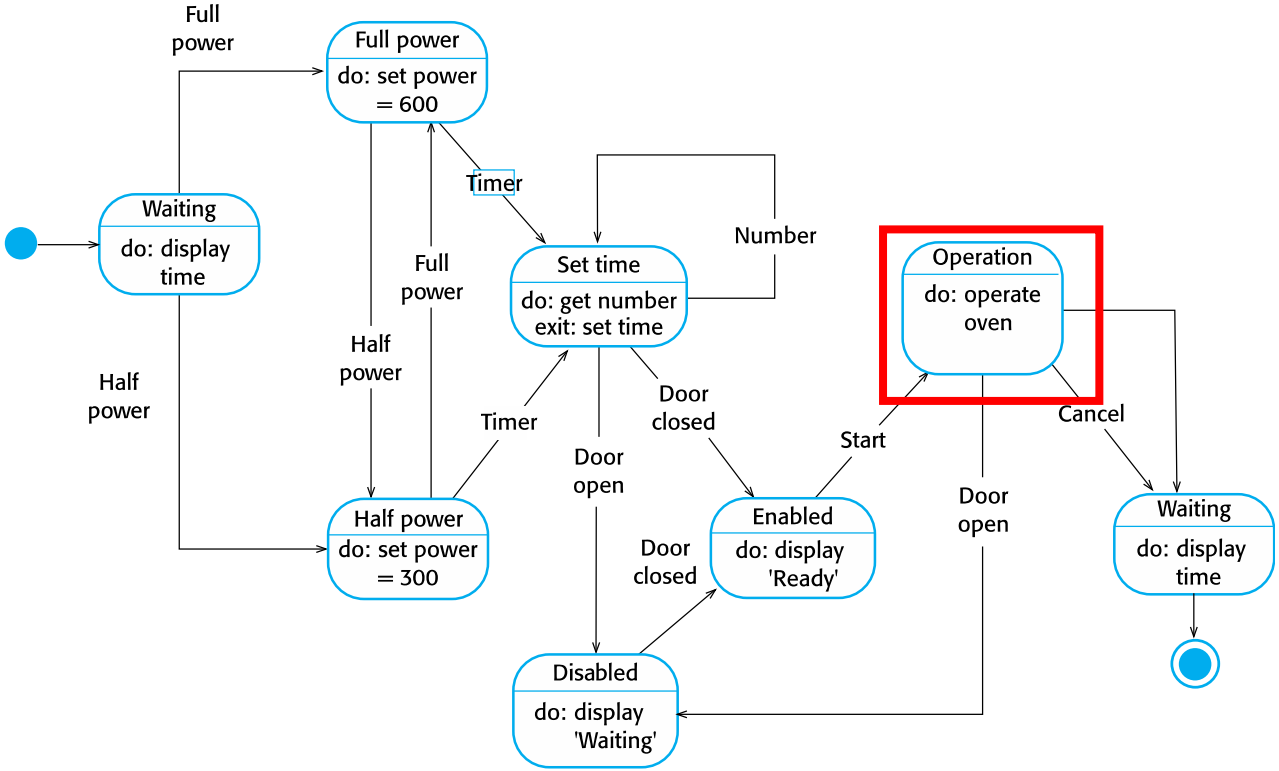
State Machine Diagram



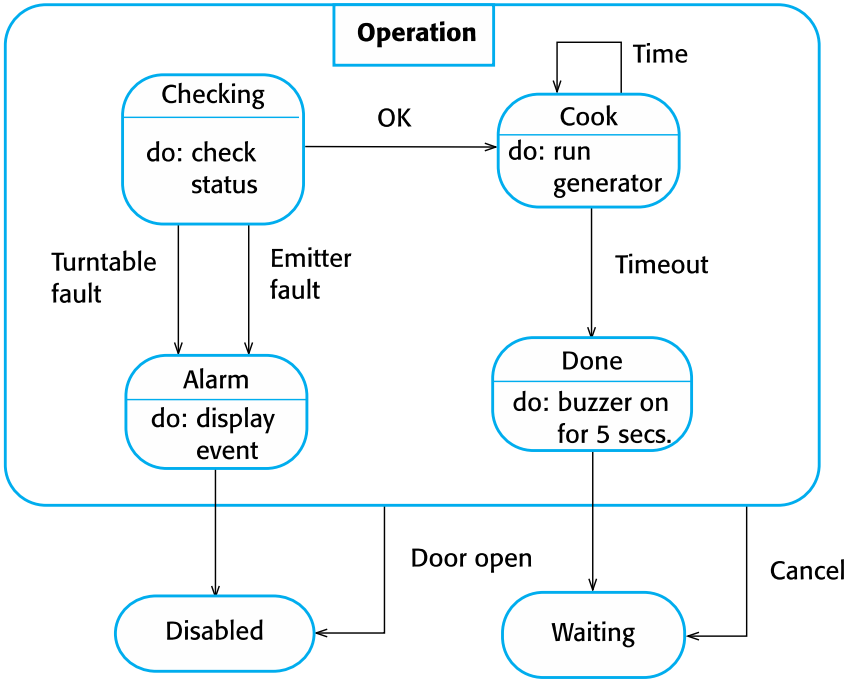
Stimuli

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.

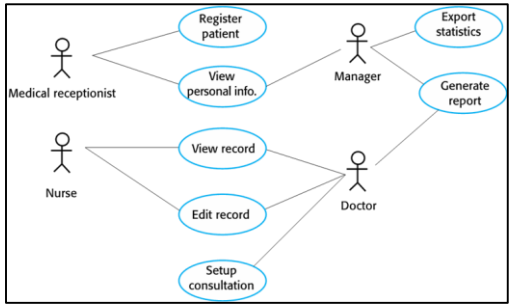
State Machine Diagram

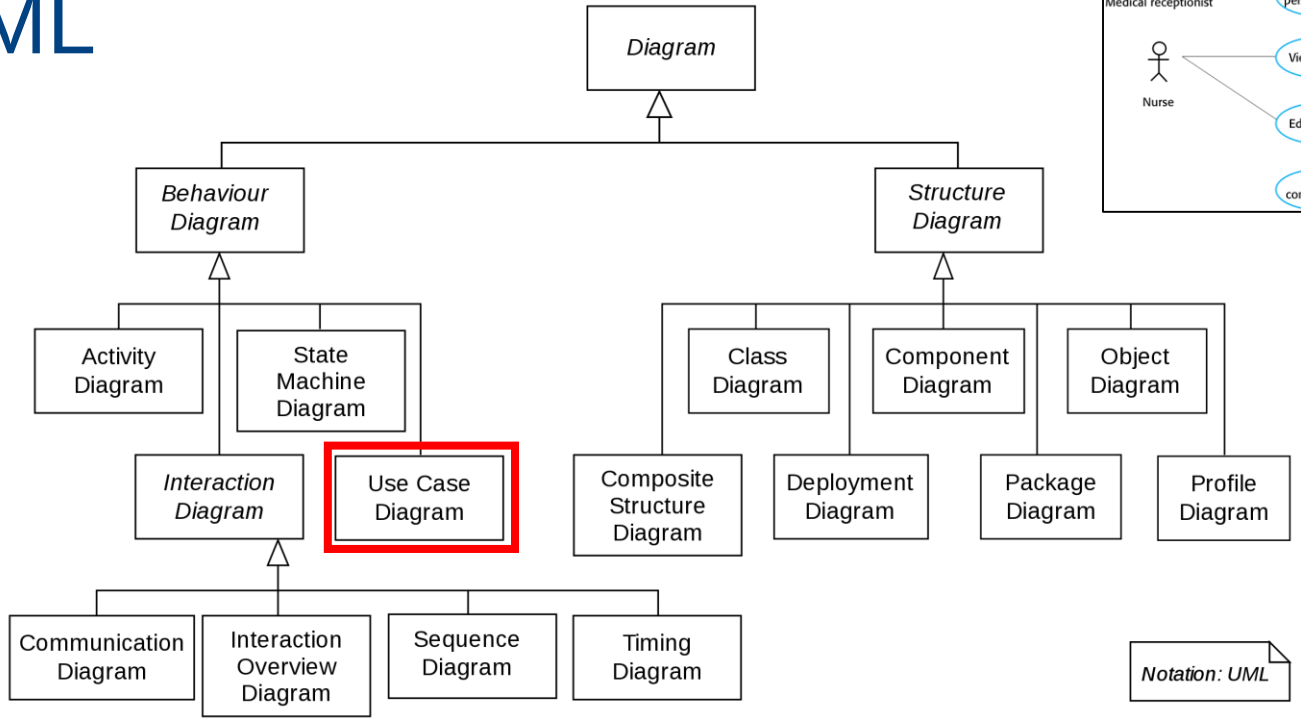


State Machine Diagram: Microwave



UML





https://en.wikipedia.org/wiki/Unified_Modeling_Language

UML: Which Parts?

UML diagrams	Number of users	Reported to be used for…
Class diagrams	7	structure, conceptual models, concept analysis of domain, architecture, interfaces
Sequence diagrams	6	requirements elicitation, eliciting behaviors, instantiation history
Activity diagrams	6	modeling concurrency, eliciting useful behaviors, ordering processes
State machine diagrams	3	
Use case diagrams	1	represent requirements

Modeling Languages Examples

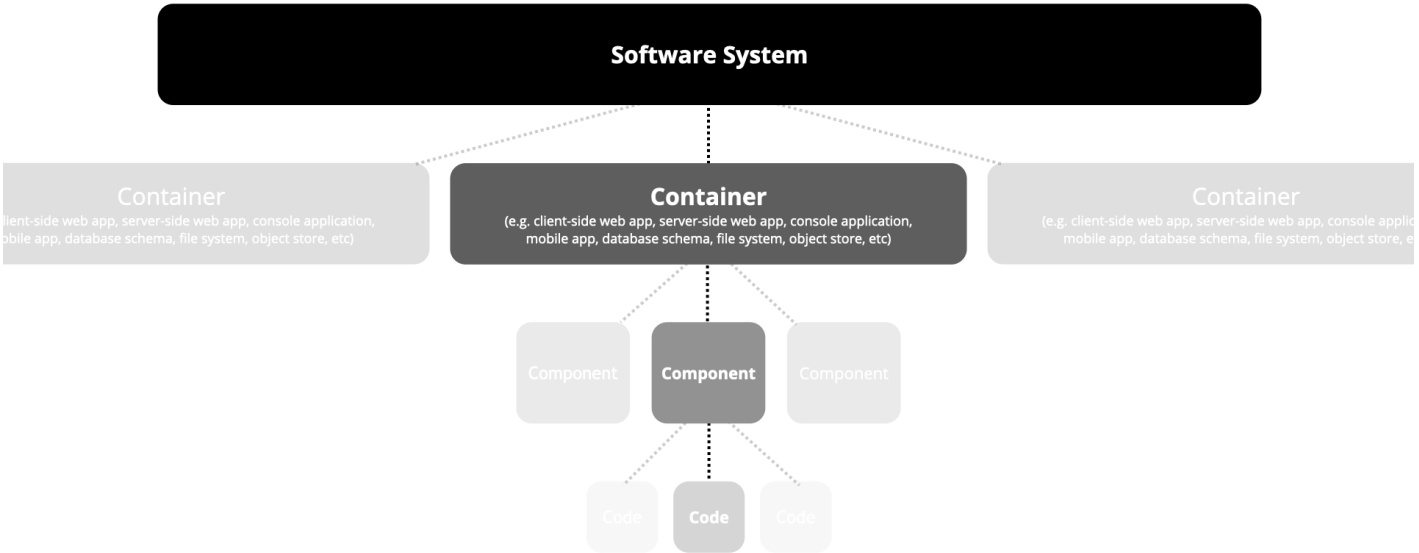
- Unified Modelling Language (UML)
- C4
- Business Process Model and Notation (BPMN)
- Systems modeling language (SysML)
- …

C4: Agile Architectural Modeling

“In the age of agile, too many software designers are afraid to over-design their applications upfront. As a result, ***many software teams have abandoned architectural thinking****, up front design, documentation, diagramming, and modelling. ‘In many cases this is a knee-jerk reaction to the heavy bloated processes of times past, and in others it's a misinterpretation and misapplication of the Agile Manifesto.’ While ‘big design’ upfront goes too far in excluding engagement and input down the line,* ***the goal is ‘to do enough upfront design, enough upfront thinking, to put a starting point in place, and set a general direction,’*** *he says. ‘Maybe not the perfect direction, maybe not the final direction, but at least some vague notion of direction that we as a team can then follow.” -- Simon Brown*

https://www.zdnet.com/article/what-weve-lost-in-the-push-to-agile-software-development/

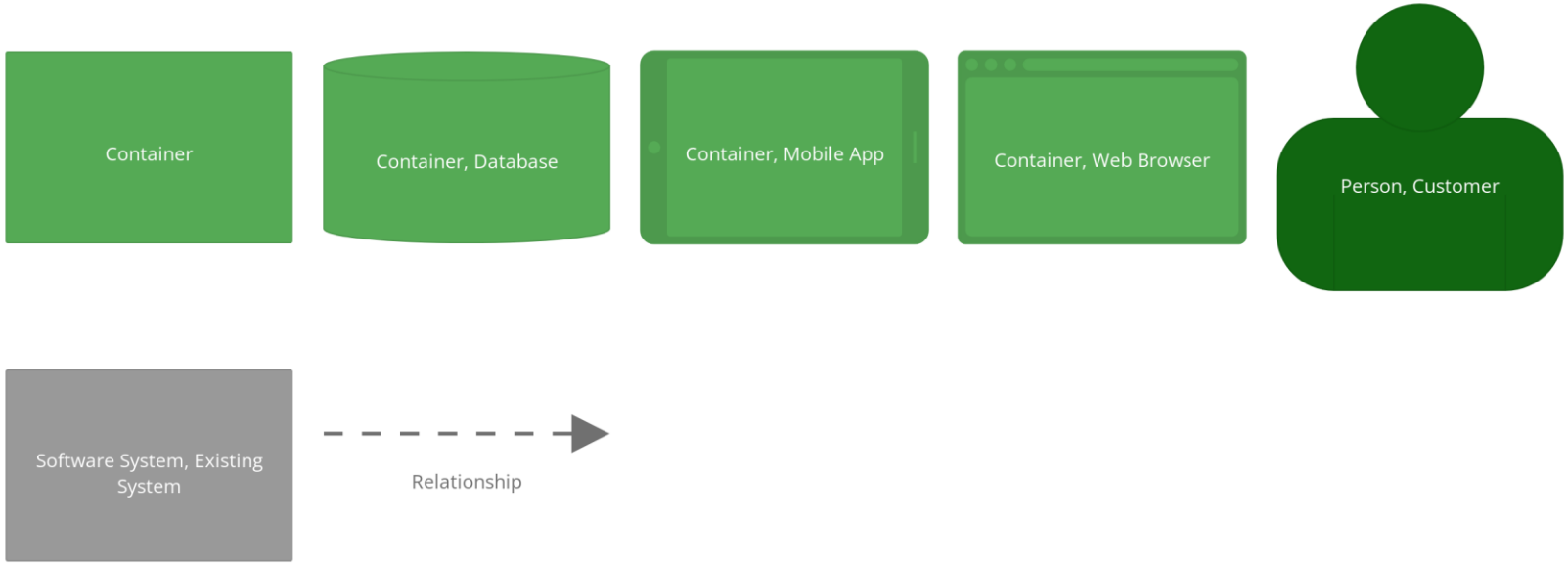
C4



A **software system** is made up of one or more **containers** (applications and data stores), each of which contains one or more **components**, which in turn are implemented by one or more **code** elements (classes, interfaces, objects, functions, etc).

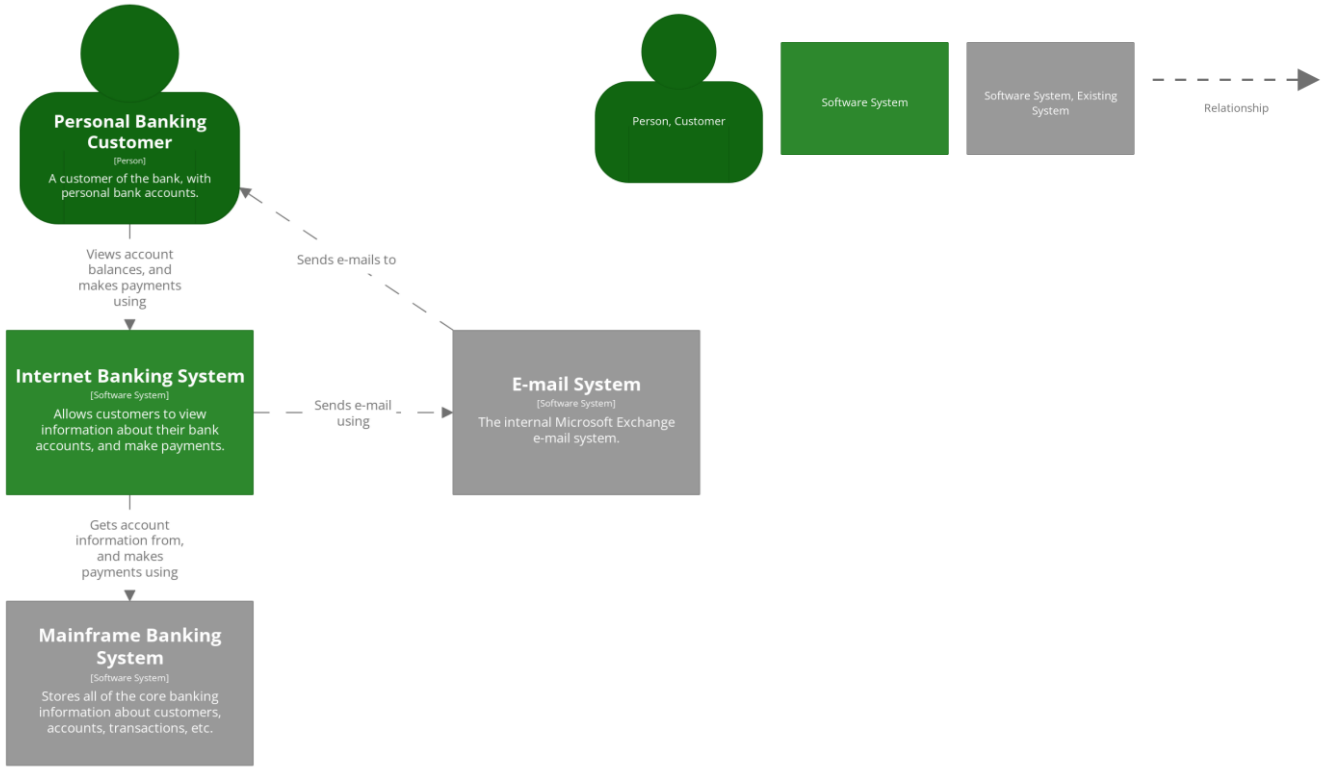
https://c4model.com/

C4



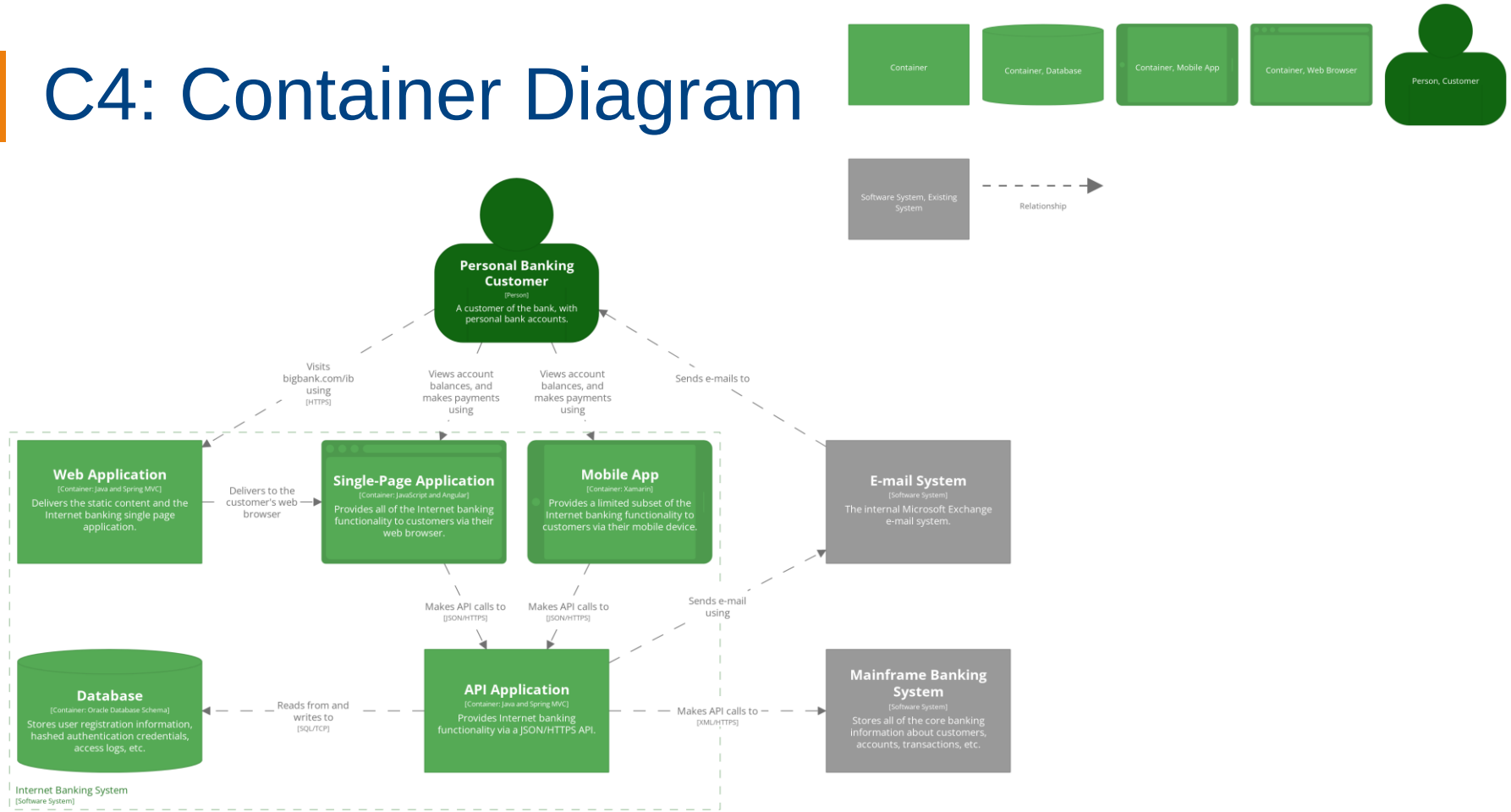
https://c4model.com/diagrams/notation

C4: System Context Diagram



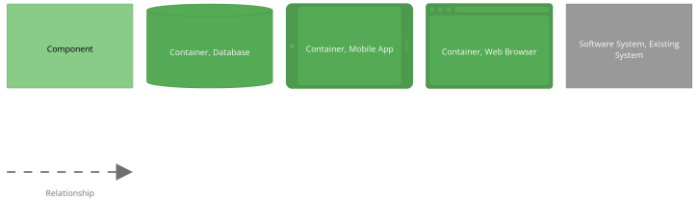
https://c4model.com/diagrams/system-context

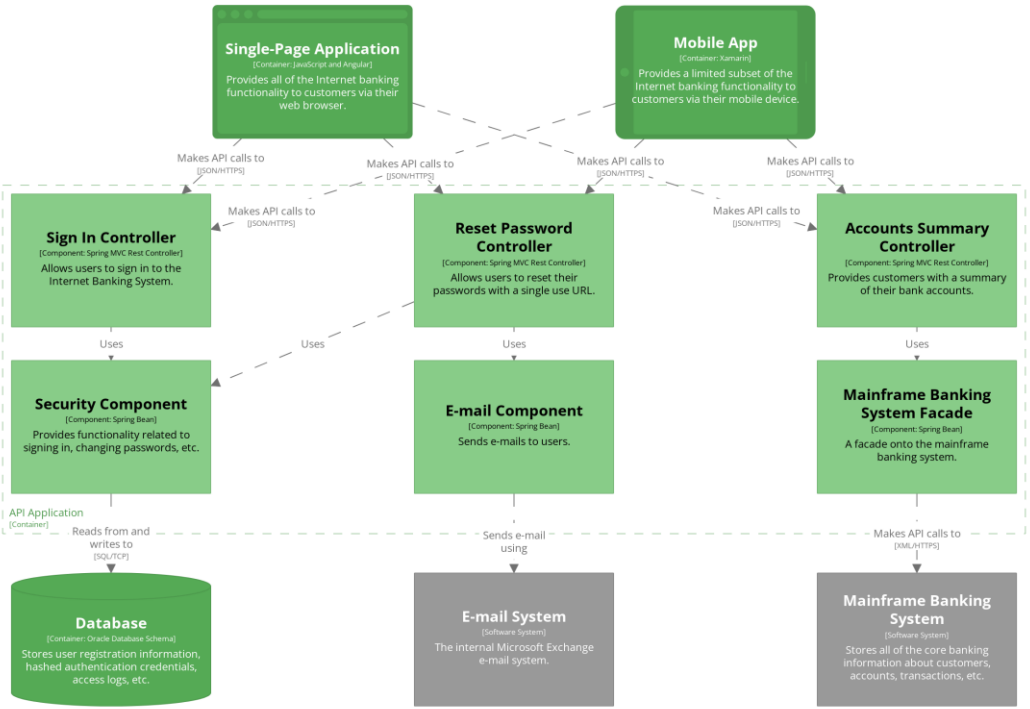
C4: Container Diagram



https://c4model.com/diagrams/container

C4: Component Diagram



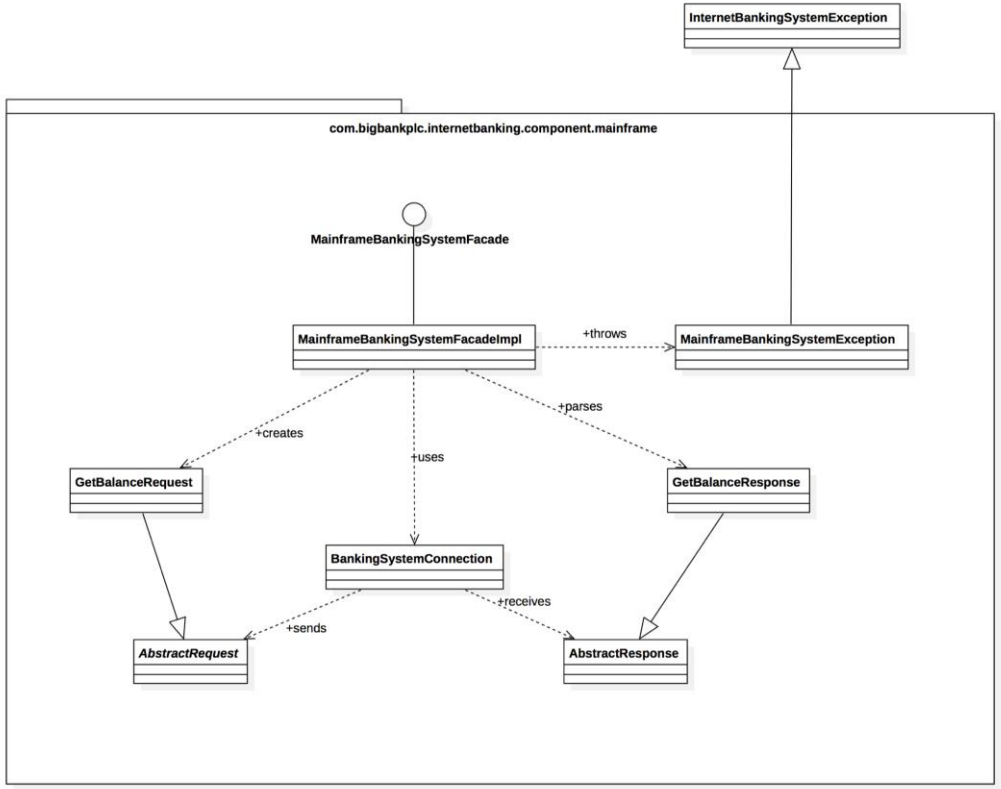


[Component] Internet Banking System - API Application

The component diagram for the API Application - diagram created with Structurizr.
Wednesday, March 22, 2023 at 8:16 AM Coordinated Universal Time

https://c4model.com/diagrams/component

C4: Code

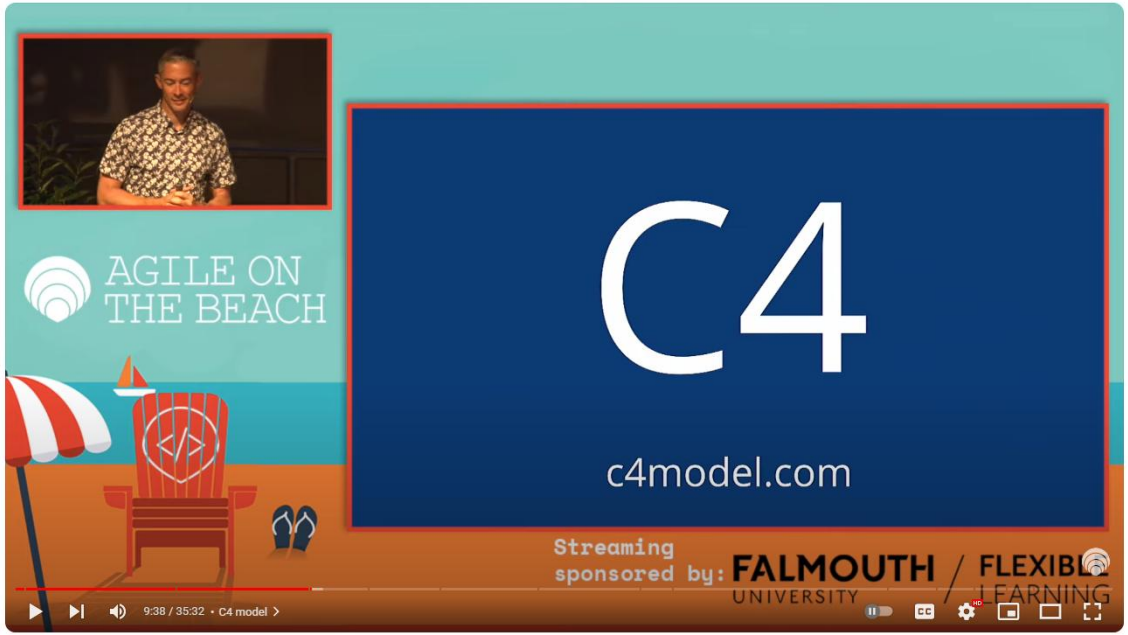


https://c4model.com/diagrams/code

C4 and UML

C4	UML
System Context Diagram	Use case diagrams, component diagrams
Container Diagram	Component diagrams (high-level), deployment diagrams
Component Diagram	Component diagrams (detailed), class diagrams
Code Diagram	Class diagram

C4



https://www.youtube.com/watch?v=x2-rSnhpw0g

Summary and Key Points

- Modeling is about creating **abstractions** using different **perspectives**
- Useful for **requirements engineering** as well as when **designing** the software
- **UML**: comprehensive modeling standard with many diagram types
- **C4**: architectural modeling language designed for an agile context

Modeling: Sources and References