Modeling dynamics of software systems

A crash course in Transition Systems

Mrityunjay Kumar

Jan 23, 2023

Agenda

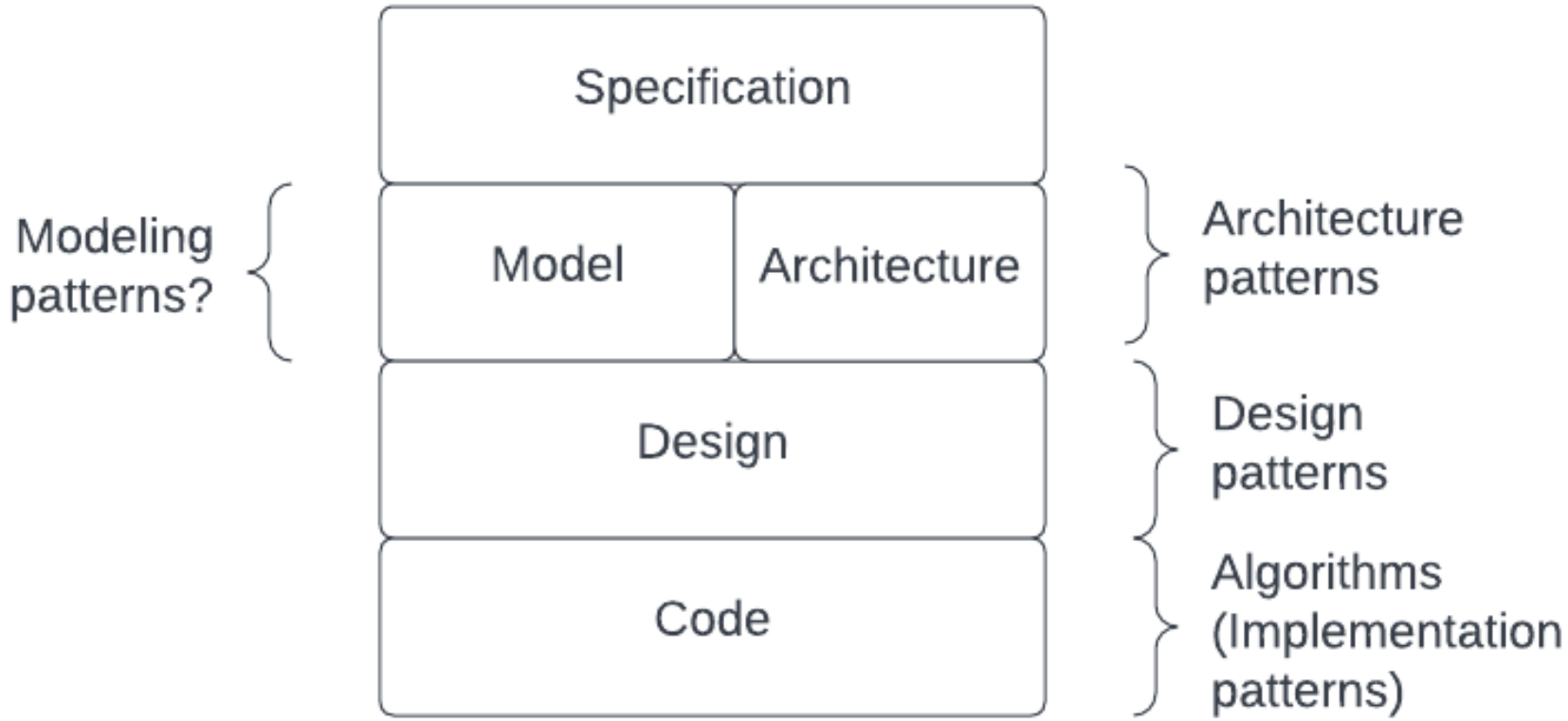
Share some industry perspective on software engineering and product development

Persuade you to think of software as a system and focus on higher-order thinking

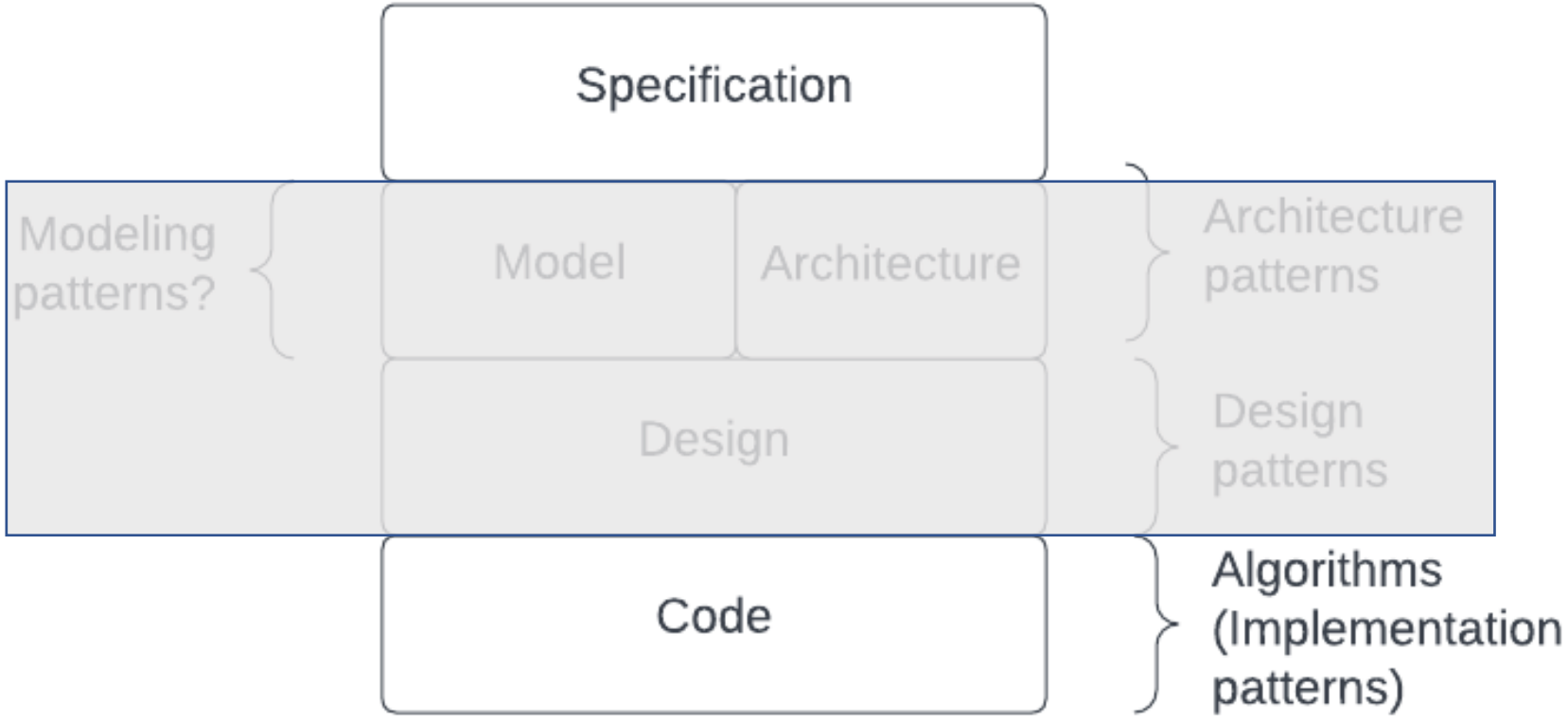
Show a vocabulary and a language that can help you learn about new systems quickly (an important trait of a successful engineer)

Industry view of software development

The industry view of software development



The industry view of software development



Modern Application Software and its development

A large, distributed, interactive system composed of multiple subsystems (services)

Delivered as a 24/7 available service

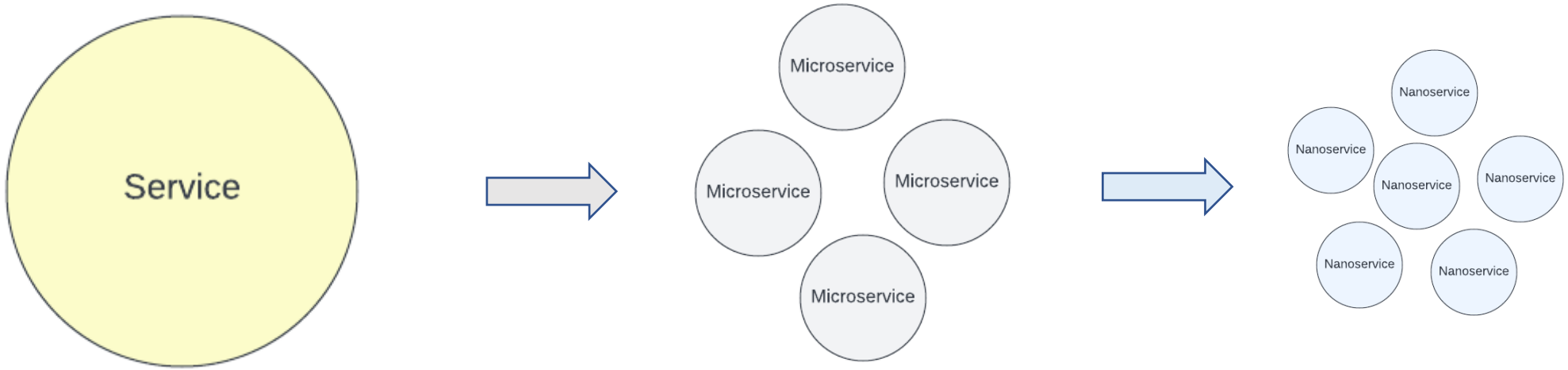
Fast release cycles, often weekly or less (in addition to monthly or quarterly ones)

Quick response cycles from customers (often within hours when things go wrong!)

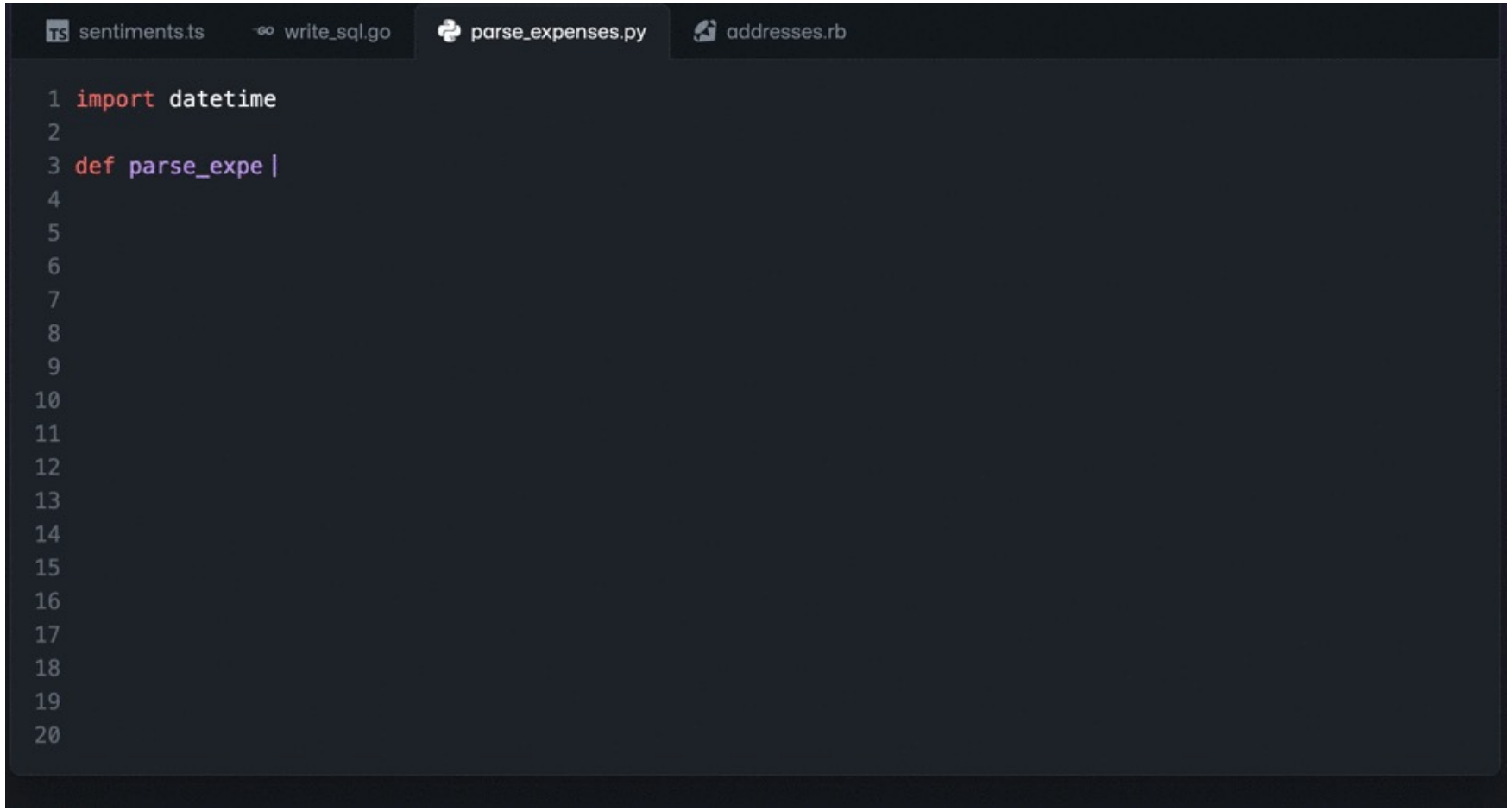
Engineer managing end to end development cycle for their delivery

Infrastructure as code

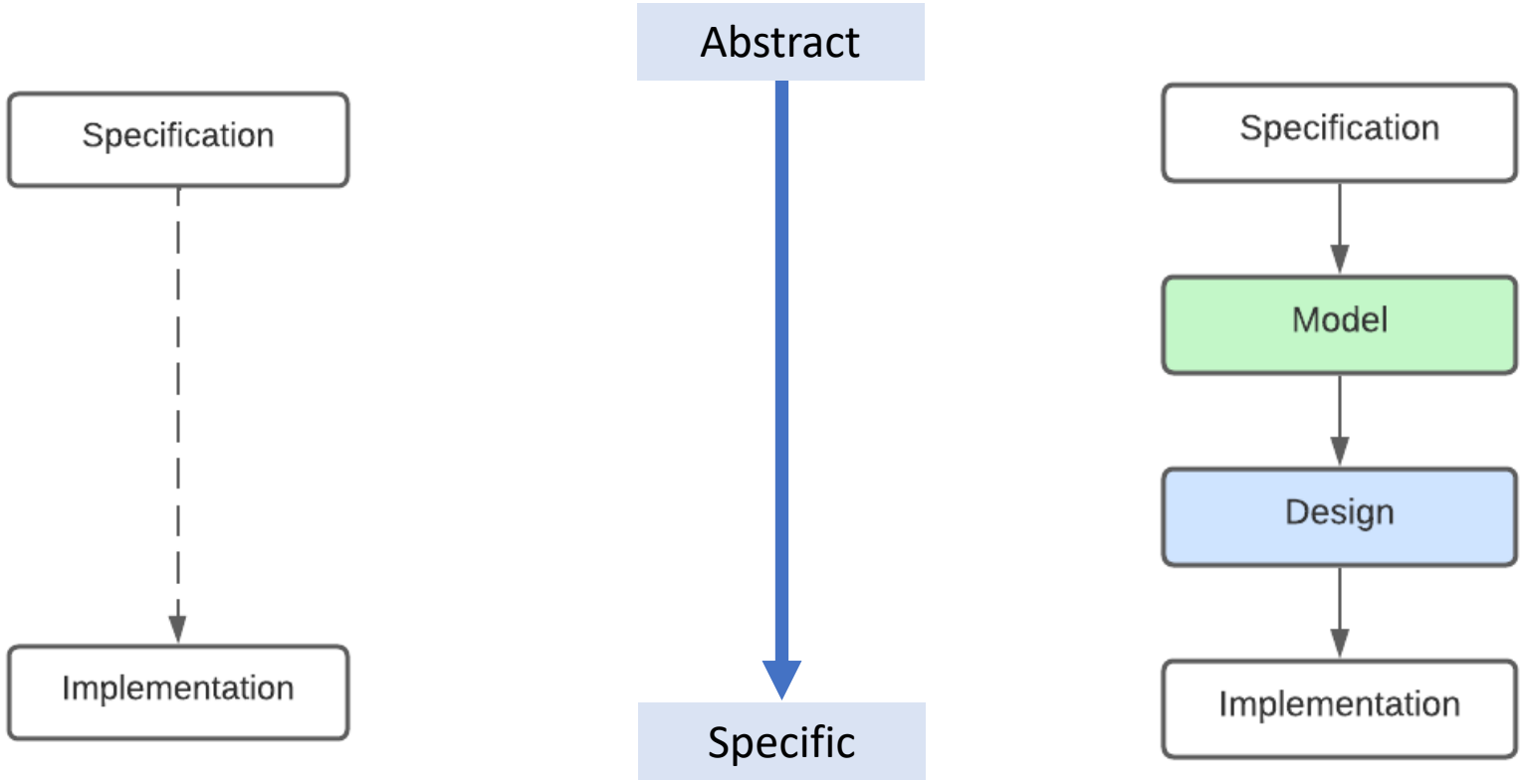
Software Product evolution



AI meets coding



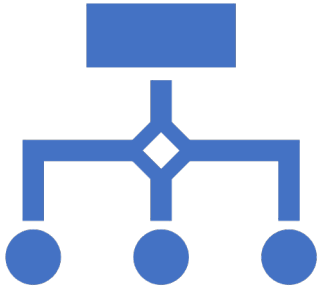
Engineers need to learn to operate at a higher level of abstraction



Two key problem paths you tackle in early parts of your career

Given a system, understand it, use it, or extend it

80%



Given a specification, model, design or implement a system

20%

What is expected of a new engineer?

Comprehend the existing software system and keep the comprehension current

Take end to end responsibility (design, develop, test, deploy, support) for their work

Account for existing system behavior when designing enhancement or defect work.

Onboarding Process = Information Overload

A few technical and product sessions

Access to source code and some partial tech documentation

Demo/test system to learn the system with

Some super-busy SMEs available for asking specific questions

Work on tickets to fix bugs

work on an enhancement

SaaS Unicorn perspective

What does a software engineer do in a software product company?

How do you quickly understand a large, complex software?

My key insight

Industry builds systems (without thinking in systems), students learn programs and algorithms

A **system** is a group of interacting or interrelated elements that act according to a set of rules to form a unified whole.

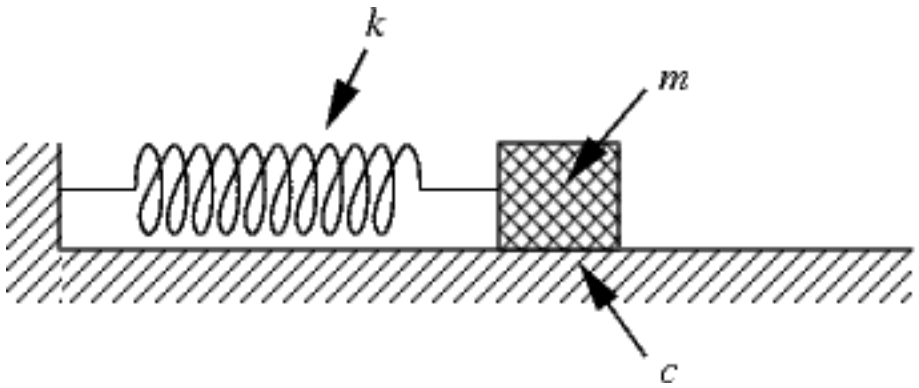
- Wikipedia

Systems Science

- Entity
- Interactions among entities
- Behavior

Spring-Mass System

Entities	Spring, Mass, Floor
Interactions	Forces (gravity, spring tension, friction)
Behaviour	Oscillatory motion



Systems change - Dynamics

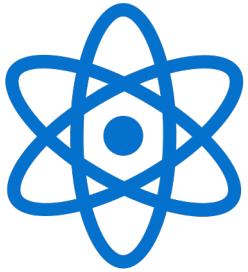
Change over time

Response to stimulus

Dynamics Model

States	Actions	Dynamics
Internal quantities that change	Agents that cause change of state	Rules that specify how actions cause change

Dynamics



Physics

Continuous quantities, continuous time

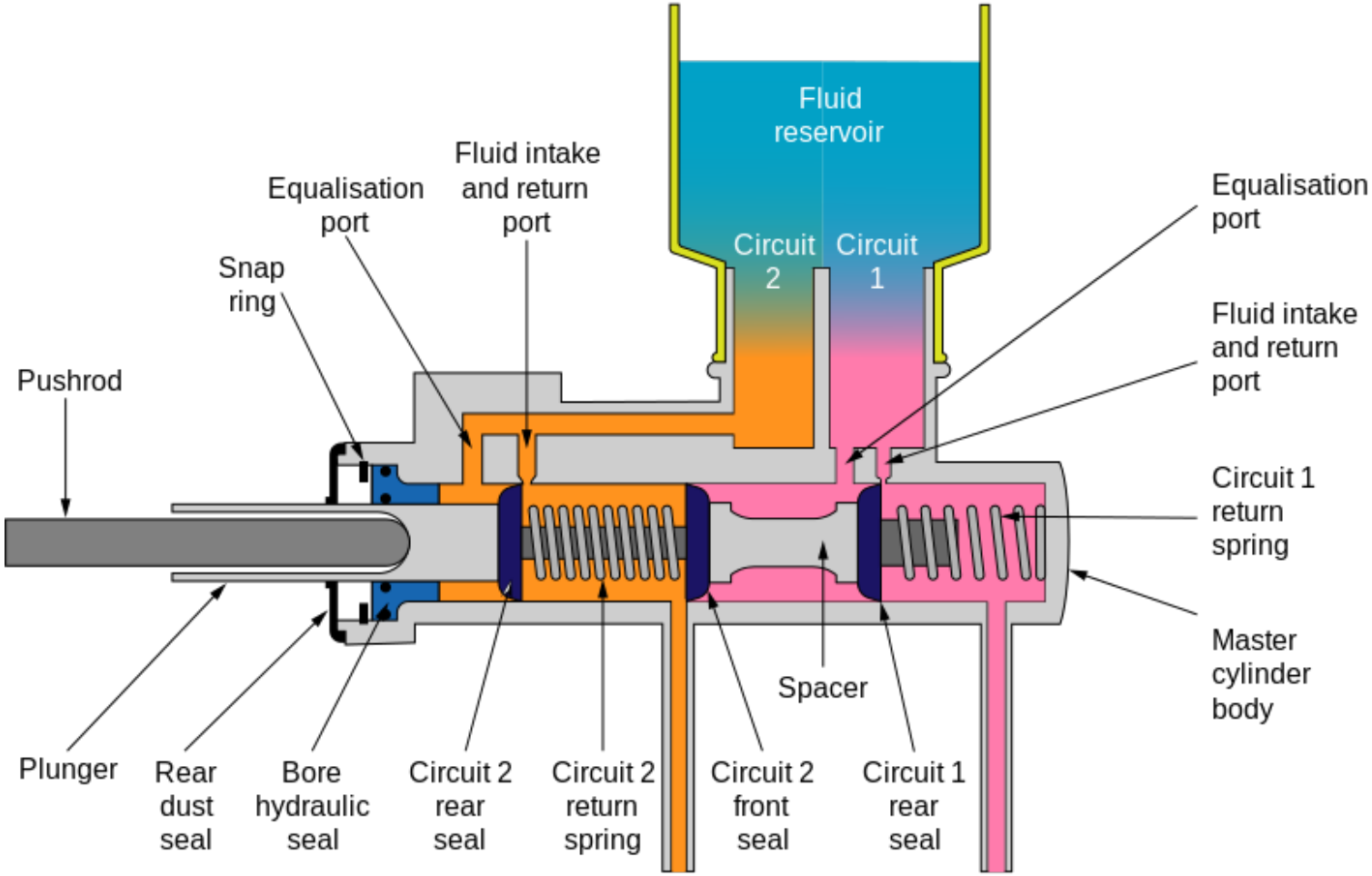
Differential equations

Computing

Discrete quantities, discrete time

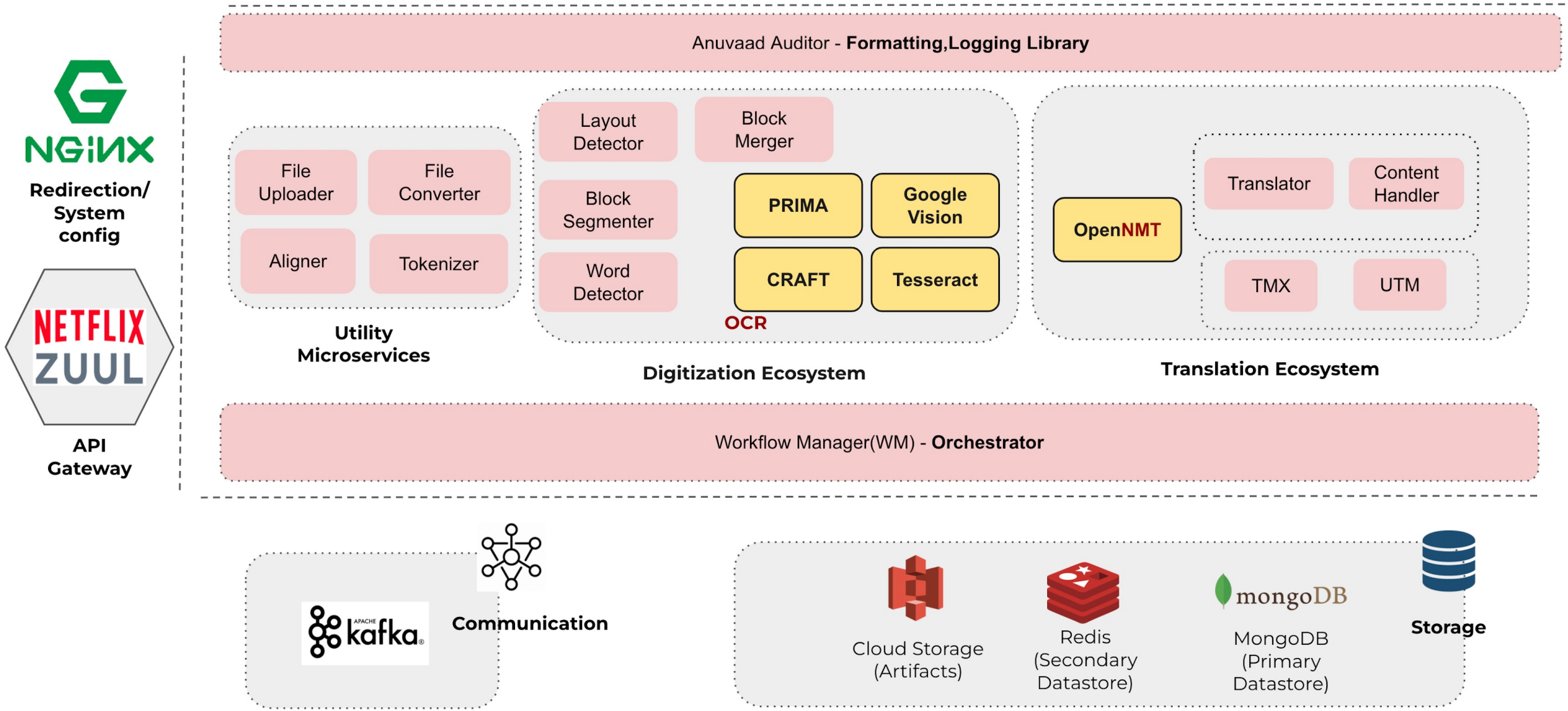
State diagrams

A schematic representation of a basic hydraulic master cylinder



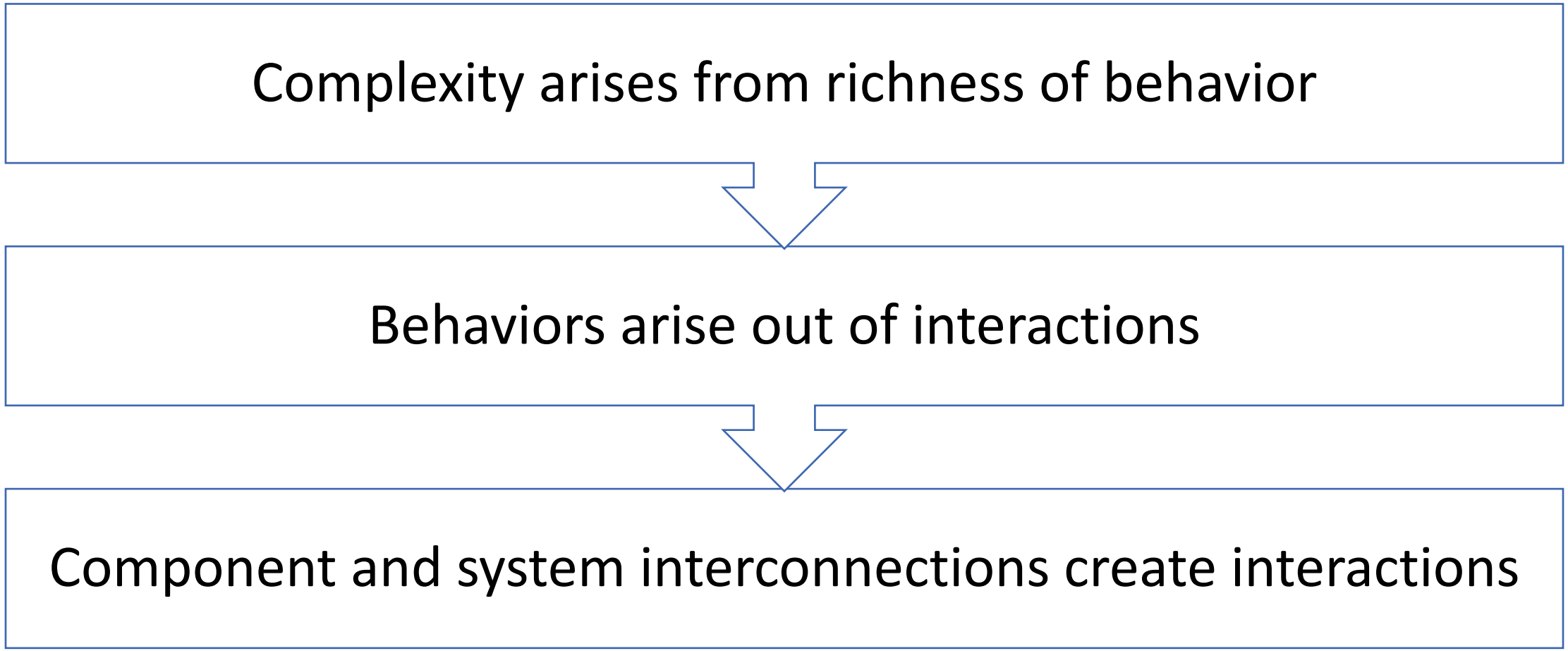
Wikipedia commons

Sunbird Anuvaad Architecture

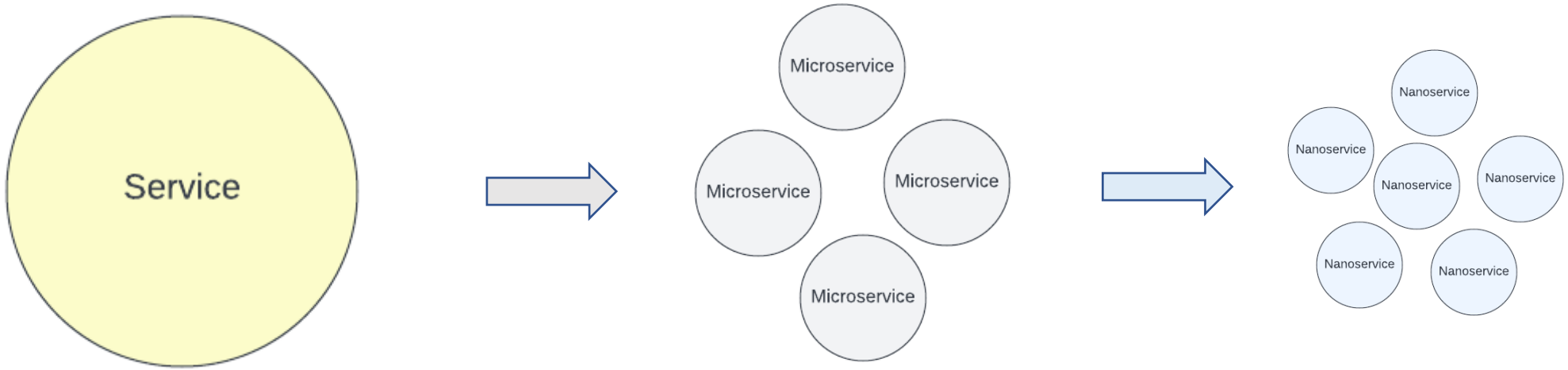


Anuvaad.sunbird.org

System principles we can apply to software



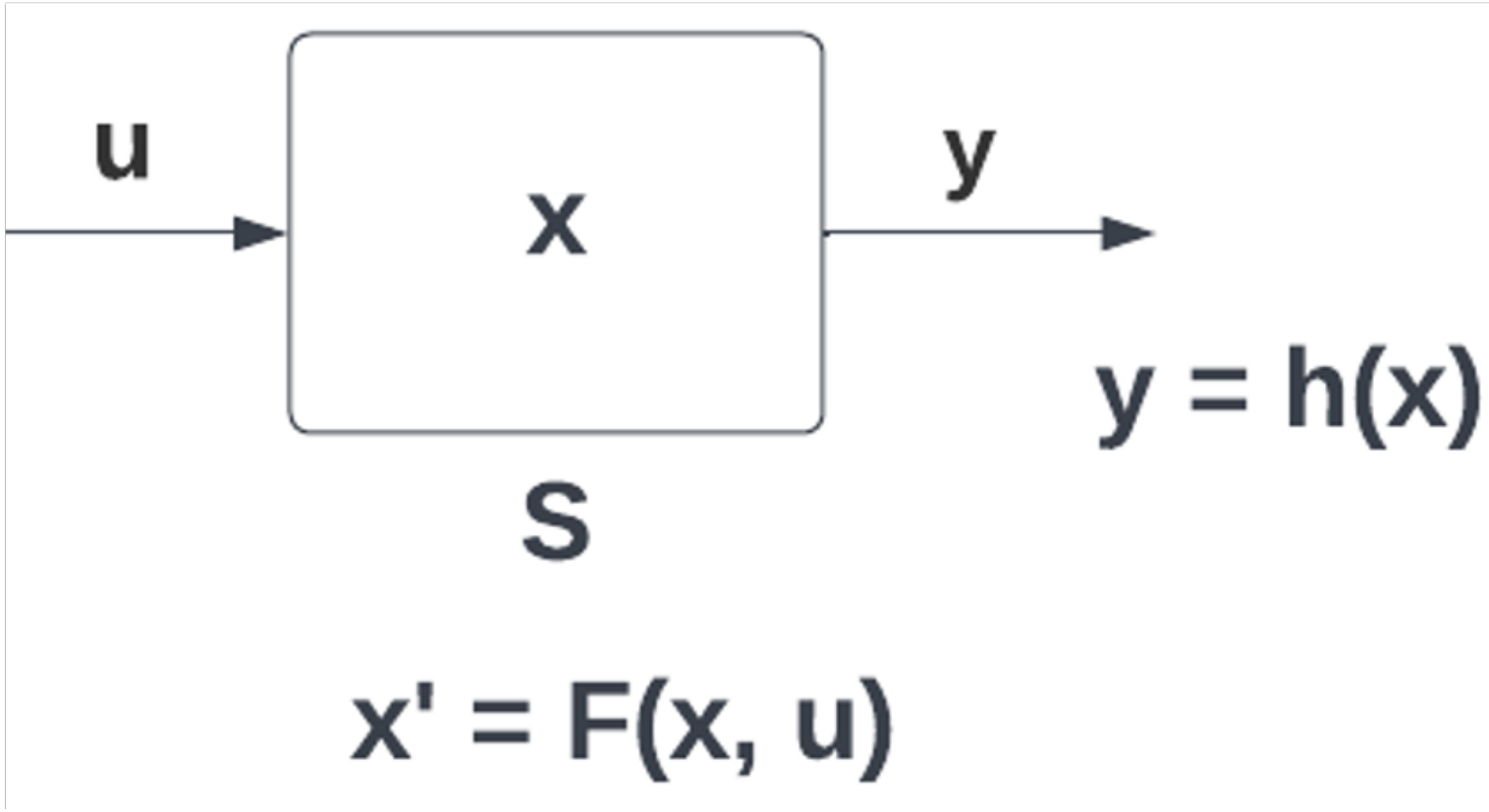
As components increase, behavior complexity increases too



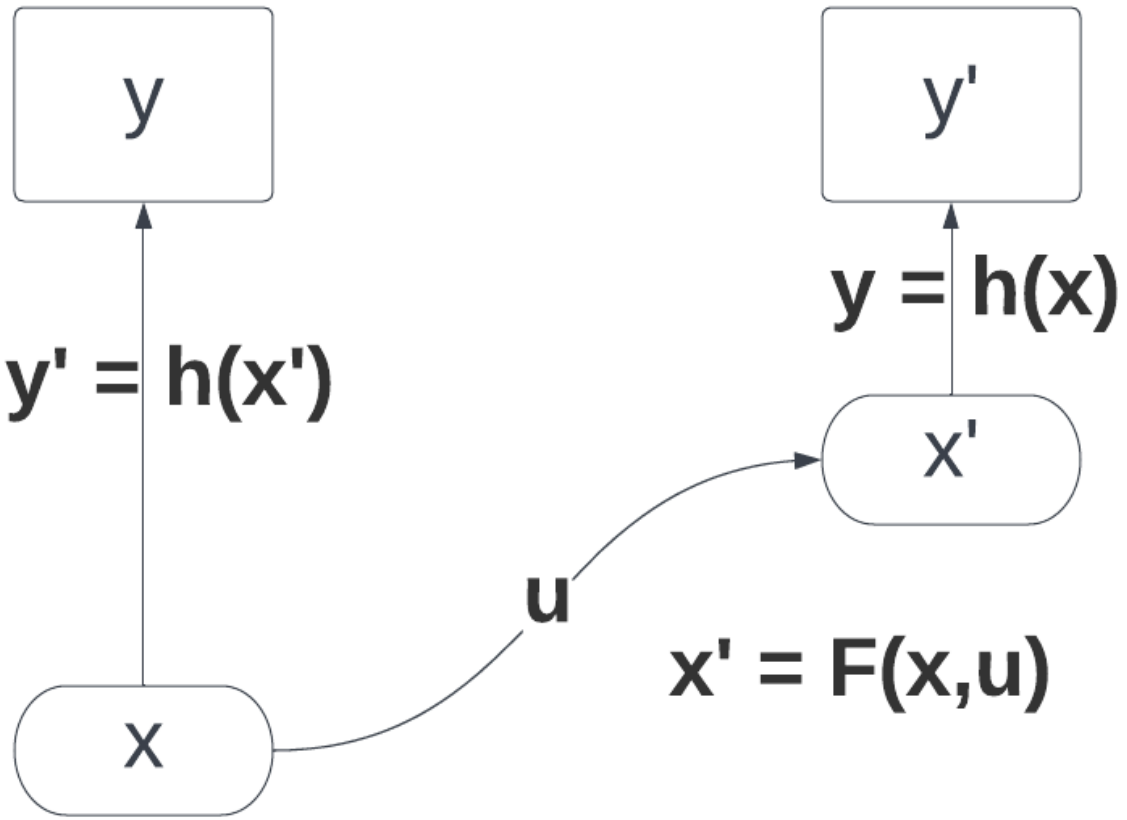
Let’s think of software as systems,
and model them as such!

Transition Systems

Transition Systems illustration



Transition Systems illustration



A transition System is a tuple $(X, X^0, U, \rightarrow, Y, h)$ where

1. X is a **state space**, a set of **states**.
2. X^0 is a subset of X and is the set of **initial states**.
3. U is an **action space**, a set of **actions**.
4. Y is an **observation space**, a set of **observations**.
5. $h : X \rightarrow Y$, called the **display** maps states to observations.
6. $\rightarrow \subseteq X \times U \times X$ is called the **transition relation** or **dynamics**. The transition (x, u, x') is written $x \xrightarrow{u} x'$.

Transition systems six-tuple

$$(X, X^0, U, F, Y, h)$$

A light bulb system

https://algodynamics.io/misc/LightBulbV2.html

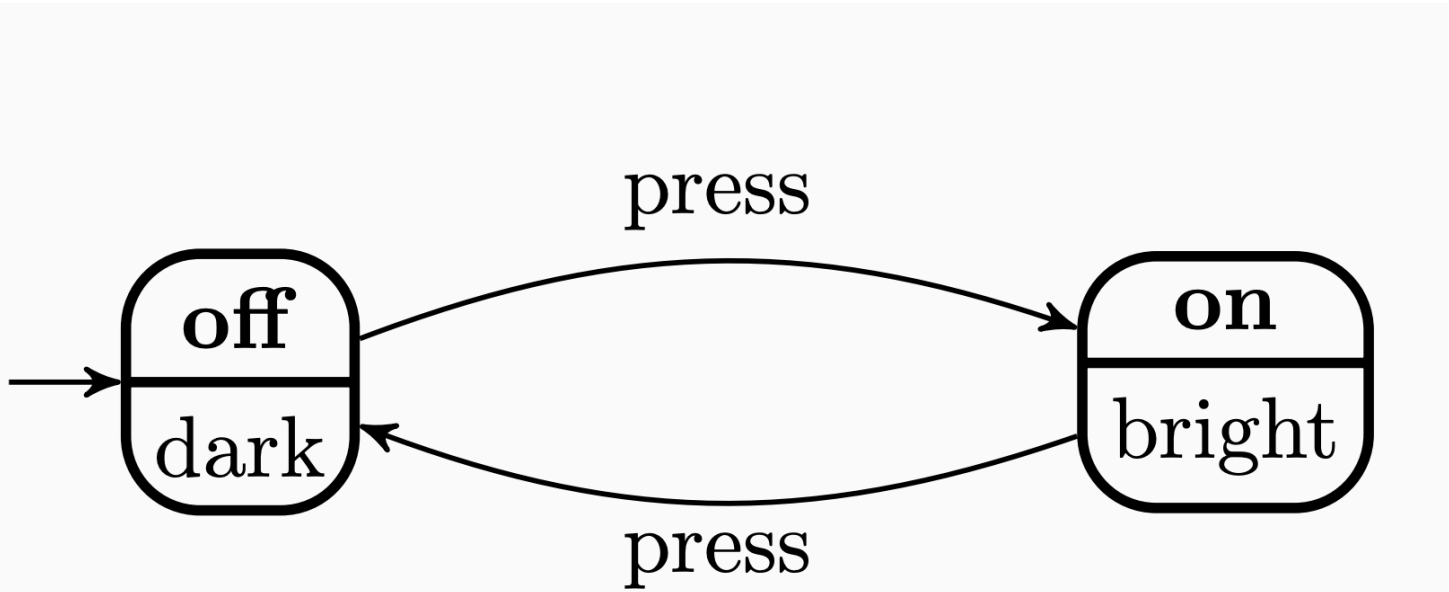
- **States:** $X = \{\text{on}, \text{off}\}$
- **Initial States:** $X^0 = \{\text{off}\}$.
- **Actions:** $U = \{\text{press}\}$
- **Observations:**: $Y = \{\text{bright}, \text{dark}\}$

- **Display:** $h(x) = \begin{cases} \text{bright} & \text{if } x = \text{on} \\ \text{dark} & \text{if } x = \text{off} \end{cases}$

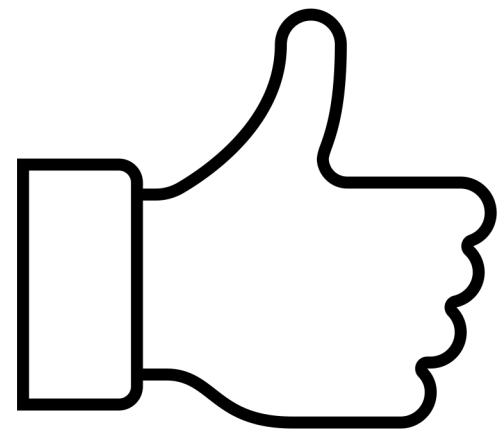
- **Dynamics:** $F : X, U \to X$

$$F(\text{on}, \text{press}) = \text{off}$$

$$F(\text{off}, \text{press}) = \text{on}$$


press
off
dark
on
bright
press

Heart/Like feature



$$X = \{Liked, NotLiked\},$$
$$X^0 = \{NotLiked\},$$
$$U = \{click\},$$
$$Y = \{Redheart, Whiteheart\}$$

$$f(Liked, Click) = NotLiked,\ f(NotLiked, Click) = Liked.$$

$$h(Liked) = Redheart$$

$$h(NotLiked) = Whiteheart$$

Robot walk simulation

https://mrityunjaypalash.github.io/gridwalk/grid.htm

Transition system of Robot walk

A todo list: Sleek

https://github.com/ransome1/sleek

Transition Table snapshot

	SuperState			
Action	ListView	AddView	EditView	ActionView
ClickPlus (null, null)	F1; AddView	X	X	X
AddSave (null, todoinfo)	X	F2; ListView	X	X
Done (todo-id, null)	F3; ListView	X	F4; ListView	X
ClickItem (todo-id, null)	F5; EditView	X	X	X
EditSave (todo-id,todoinfo)	X	X	F6; ListView	X
Delete(todo-id, null)	X	X	X	F7; ListView
RightClick(todo-id, null)	F8; ActionView	X	X	X
UseAsTemplate (null, null)	X	X	X	F9; AddView

For details, look out *for reading notes that will be posted on Wednesday*

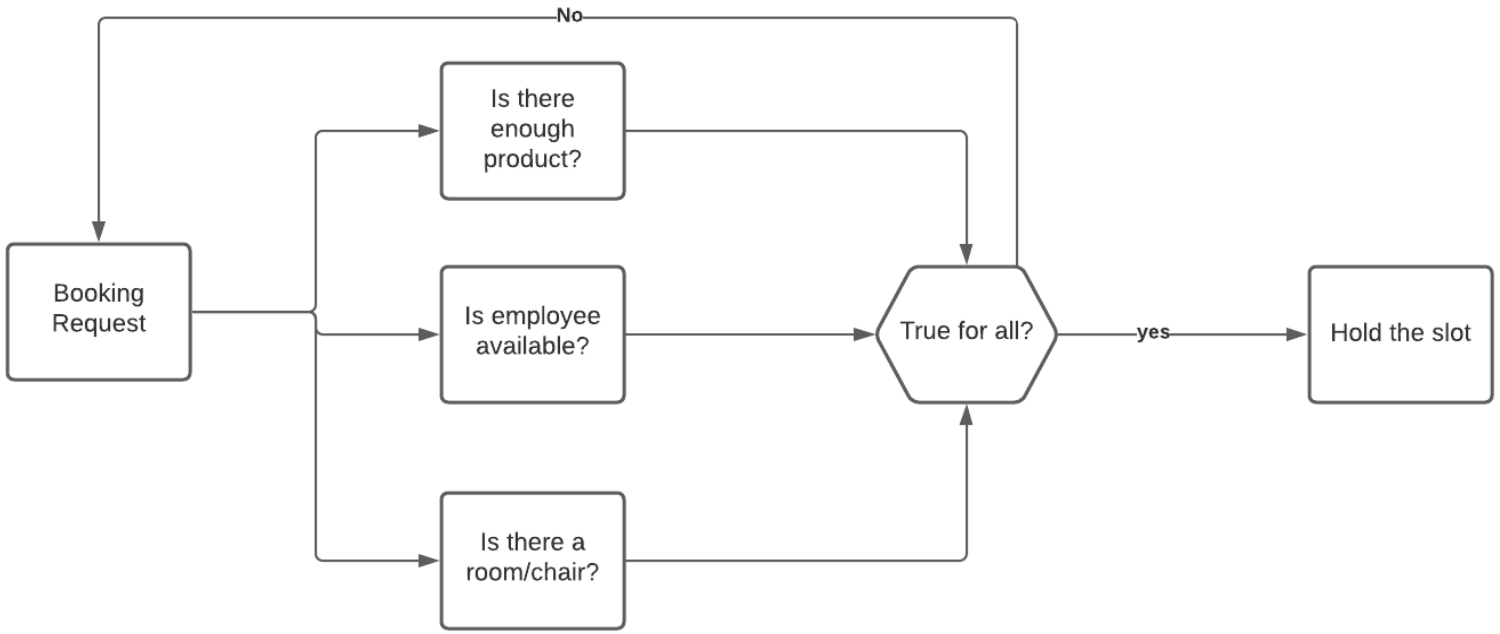
Modeling large systems

Principles to follow for large systems

- Compose large systems from small systems
- Decompose systems when transition function or state definition becomes complex
- Separate ‘stateless’ systems from ‘stateful’ systems
- Think messages (not function call) when connecting systems
- Interconnections are like electrical wirings – draw them out

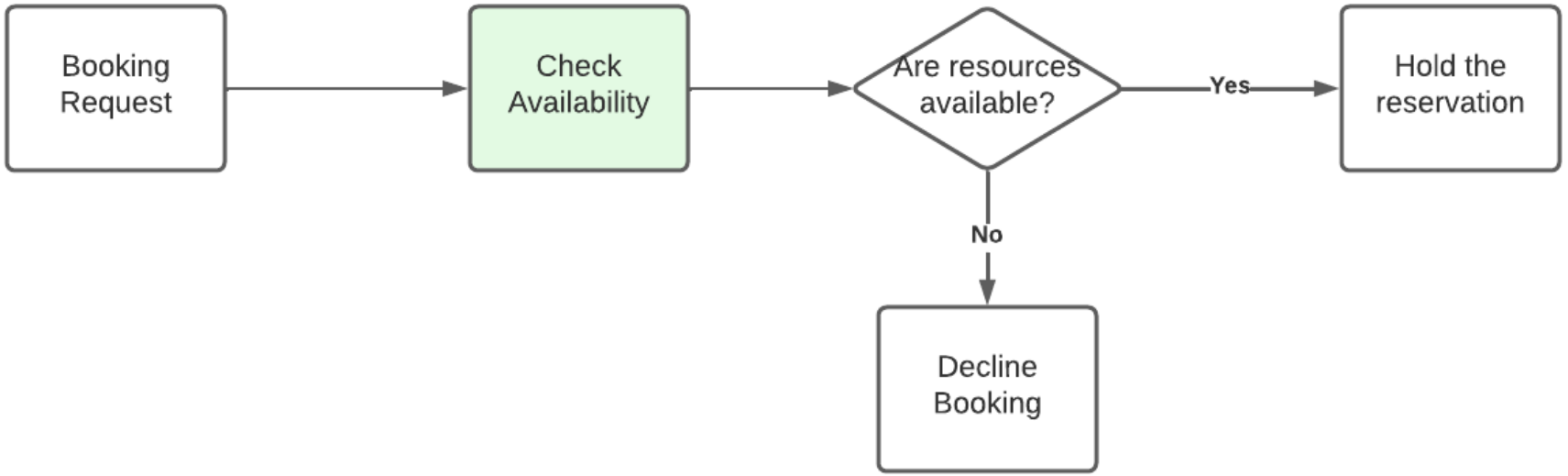
Decomposition by Zooming in

Zoom In: What does the system do internally?



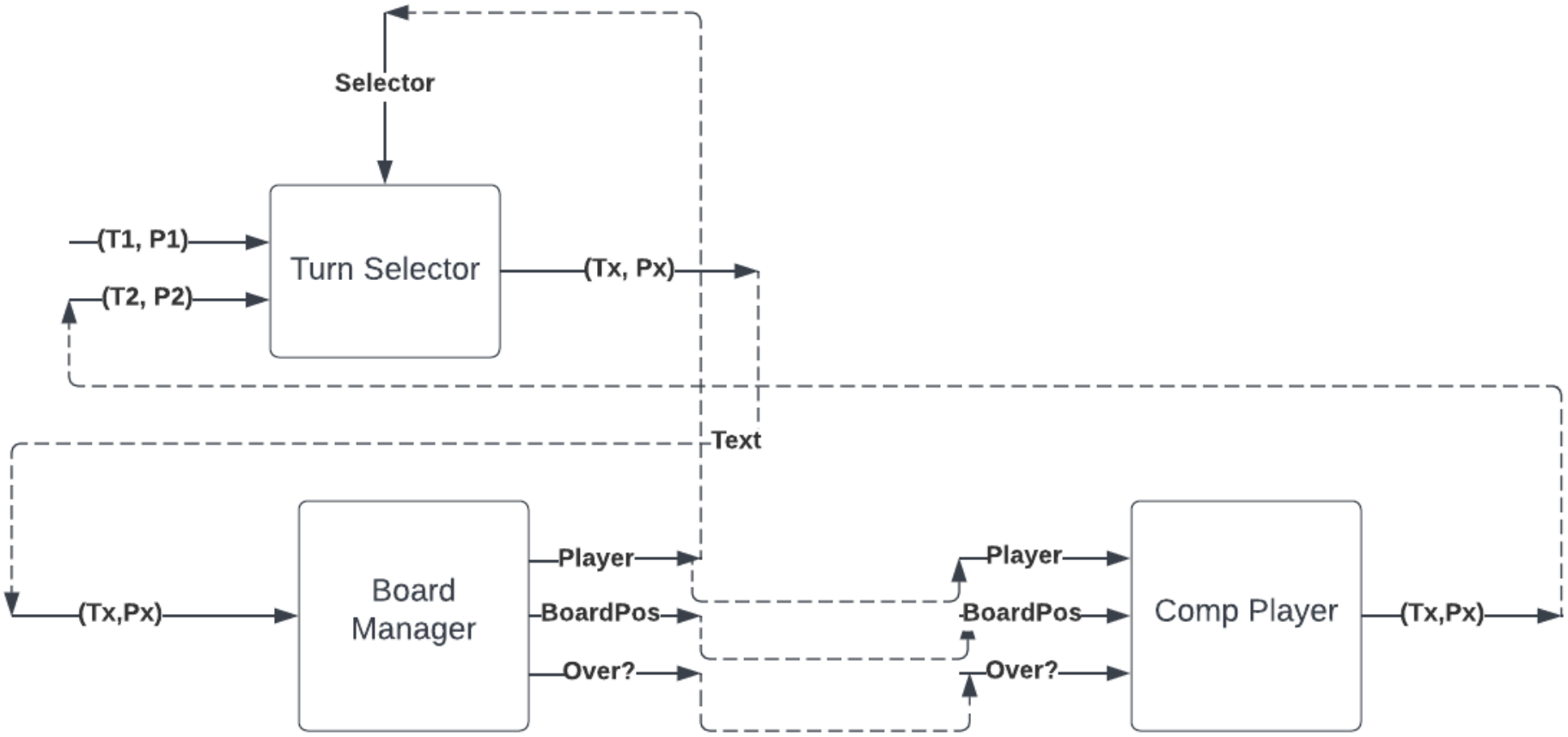
What sub-systems can you identify here?
What interconnections do you see?

Zoom In: Take 2



There are many ways of decomposing a system, choice of Zoom In is with the modeler

System composition and interconnection (Tic-tac-toe)

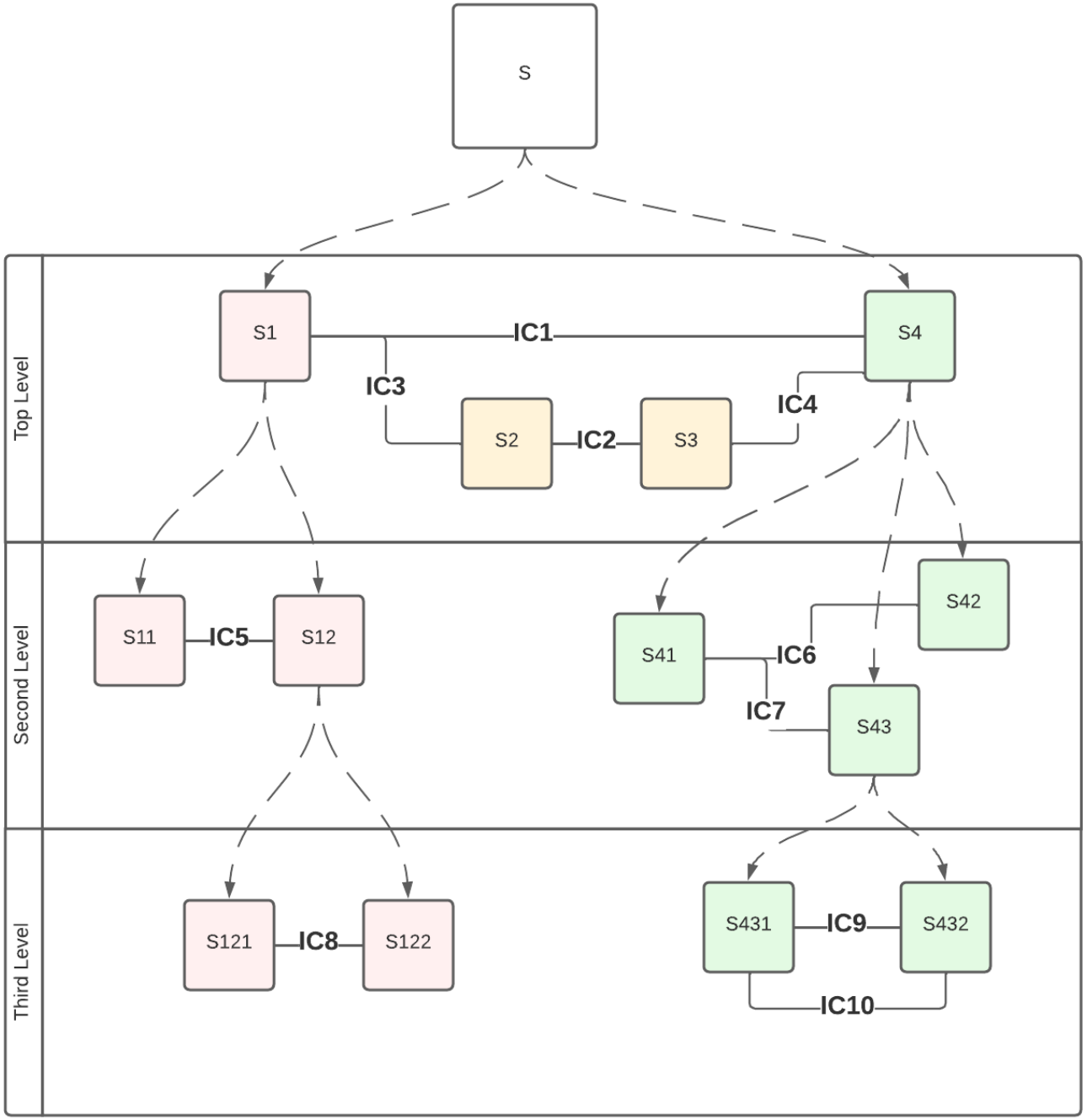


Modeling Approach

EXPAND SYSTEMS AT DIFFERENT ABSTRACTION LEVELS

DEFINE INTERCONNECTIONS BETWEEN SYSTEMS

SPECIFY 6-TUPLE OF EACH SYSTEM



Hierarchical Decomposition

Recommended (iterative) process to model a software system

Define the system at highest level of abstraction (behavioral model).

Identify known subsystems and model them individually

Build the interconnections of subsystems that matches the behavioral model of original system

Analyze the transition function F and state X for a system – what system decompositions will simplify them?

If there is an existing system that matches one of the system's definition, reuse that system by mapping the input (u) and output (y).

Model to Design

A model pattern will map to one or more design patterns

u and y of a system map to interface of the system, which will persist through the zoom-in and zoom-out process.

Representation of X and their performance characteristics will drive design choices

An interconnection will map to one or more design patterns

Communication between systems map to specific messaging architecture choices

Homework - Review software products and try modeling their behavior!

Next steps

- **Reading materials**
  - In a world of systems (10 min video animation)
  - A philosophical look at system dynamics (Donnella Meadows, 53 minutes lecture video)
  - Introduction to Transition Systems (additional notes)
  - Composition of Systems (additional notes)
- Review the assignment (will be posted by Tuesday)
- Read lecture notes to follow-up from class discussion (will be posted on Wednesday)
- Post questions or reach out (mrityunjay.k@research.iiit.ac.in) if you are interested in learning more about Transition Systems modeling

Questions?