LINKÖPING UNIVERSITY

729G87 Interaction Programming

Lecture 6 – Project, Prototyping

Philipp Hock, PhD
philipp.hock@liu.se

Agenda

- Feedback/Evaluation
- Assignment 5
- Project
- Wireframes & Diagrams
- Prototyping
- Excursion: Game dev
- Live coding: minigame (if we have time)

Last assignment

https://729g87.gitlab-pages.liu.se/course_page_astro/assignments/assignment5-inputs_from_hell/

- Difficulty
  - Choose a method that suits your skill level
- Submission
  - No presentation on Friday
  - Create a video of your input method
    - ~30s (max 1min)
    - Should show and explain your input method
    - Can have voiceover and/or subtitles
    - Will be available for all in a single-cut video
      - Your video can be anonymized, does not have to be
  - Provide a link in your gitlab page to
    - The input method(s)
    - The video that demonstrates the input method

Recap: Project

- Same groups as during assignments
- Choose your project category
  - Web shop
  - Psychological experiment
  - Game
- Adjust your goals/ideas to your skills
  - Think about basic features that **must** be in the final project
  - Define some features that **should** be in the final project
  - What other things **could** be added in the future?
- Deliverables
  - Wireframe Specification
    - Gui prototype (drawings of your gui)
    - Behavioral aspects (diagram(s))
    - Structural aspects (diagram(s))
  - Implementation
- Point requirements
  https://www.ida.liu.se/~729G87/about/examination/

Level of complexity

- Must Haves
  - This is your basic system
  - This is **your** minimum goal for the assignment
- Should Haves
  - You aim for this to be in the final prototype
- Could Haves
  - Realistically speaking, this won't be in the final prototype, but maybe you achieve it.

You define this for yourself.

Wireframe / Specification

- Deadline: 22 December
- Think about your product before you start coding

- GUI prototype
  - Wireframe
  - No program logic
  - Should visualize the final product
    - Can be more than you implement
- Structural diagrams
  - Model the structure of your system
  - Does not have to model the entire system
- Behavioral diagrams
  - Model behavioral aspects of your system
  - Does not have to model all interactions
- Do not build your final system with the prototyping software

Web shop

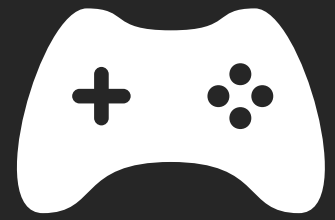
- Browse items and interact with shopping cart
- Functional requirements:
  - browse items
  - items have more then one picture
  - items have a description
  - add/remove/edit items in shopping cart
  - view shopping cart
- Example subcategories: clothing store, food store ..
- Do not clone an existing shop

Psychological experiment

- Get inspired by https://www.psytoolkit.org/experiment-library/
- Classical experiments are
  - Stroop task
  - N-back task
  - Go/No-go task
  - …
- Such experiments are intended to create certain stimuli
  - Increase cognitive load
  - Induce stress
  - Increase memory workload
  - …
- In each experiment, performance is measured and later analyzed
- Experiment must include a performance evaluation (statistics of how participants performed on the task)

- Difficulty to implement differ
  - Get feedback from teaching assistant/me before starting

Game



- Games can be very simple but can become very complex to implement!
  - Initial feedback from teaching assistant/me is mandatory!
- Some games require server-side logic.
  - You have no access to custom server-side logic
  - Do not implment such games
- Probably the most complex choice
  - Rewarded by lots of fun implementing and playing
- Classic games are
  - Pong
  - Snake
  - Blackjack
  - Breakout
  - Tetris (can be very challenging)
  - You can also create your own game
- Add a custom feature that is not in the original game
- https://developer.mozilla.org/en-US/docs/Games/Tutorials/2D_Breakout_game_pure_JavaScript

Libraries & Frameworks

- Web development usually heavily relies on libraries, frameworks, etc
- You learned mostly vanilla web development here
- You are free to use frameworks
- Mind they can have different learning curves
- Only tools/frameworks are allowed that allow development in js/ts
- No game engines (Unity, Unreal, Godot,…)
  - Js game engine libraries are ok!
- We might not be familiar with the framework you use
  - Don’t expect help
- There are many many js frameworks
  - https://blog.logrocket.com/best-javascript-html5-game-engines/
  - https://github.com/KilledByAPixel/LittleJS

Presentation

- Deadline: 12 January
- Present your project
  - Live demo
  - Have a backup video if live demo fails (video)
  - Prepare some slides with insides of the development process
  - Focus on showing the final project not code
  - 180min timeslot, 29 groups
    - 5min presentations
  - Stay in time!

Requirements for game and psy. Exp.

- Should not copy an existing implementation entirely
  - Add your own ideas!
  - Modify the basic concept of the original system
  - Come up with an entirely new idea
  - Be bold
    - Set your goals high, it's ok if you don't achieve them
    - Better not to meet high targets than to meet low targets too early
- Be creative!

Games

- Too simple*
  - Tic Tac Toe
  - Guess the number
- Appropriate complexity**
  - Memory
  - Snake
  - Sudoku
  - Minesweeper
  - Breakout
  - Tower defense
  - Platformer
- Too complex**
  - Multiplayer (nearly impossible without a custom server)
  - Real time strategy game
  - SimCity
  - Online Poker
  - …

* You can add more complexity to a simple game

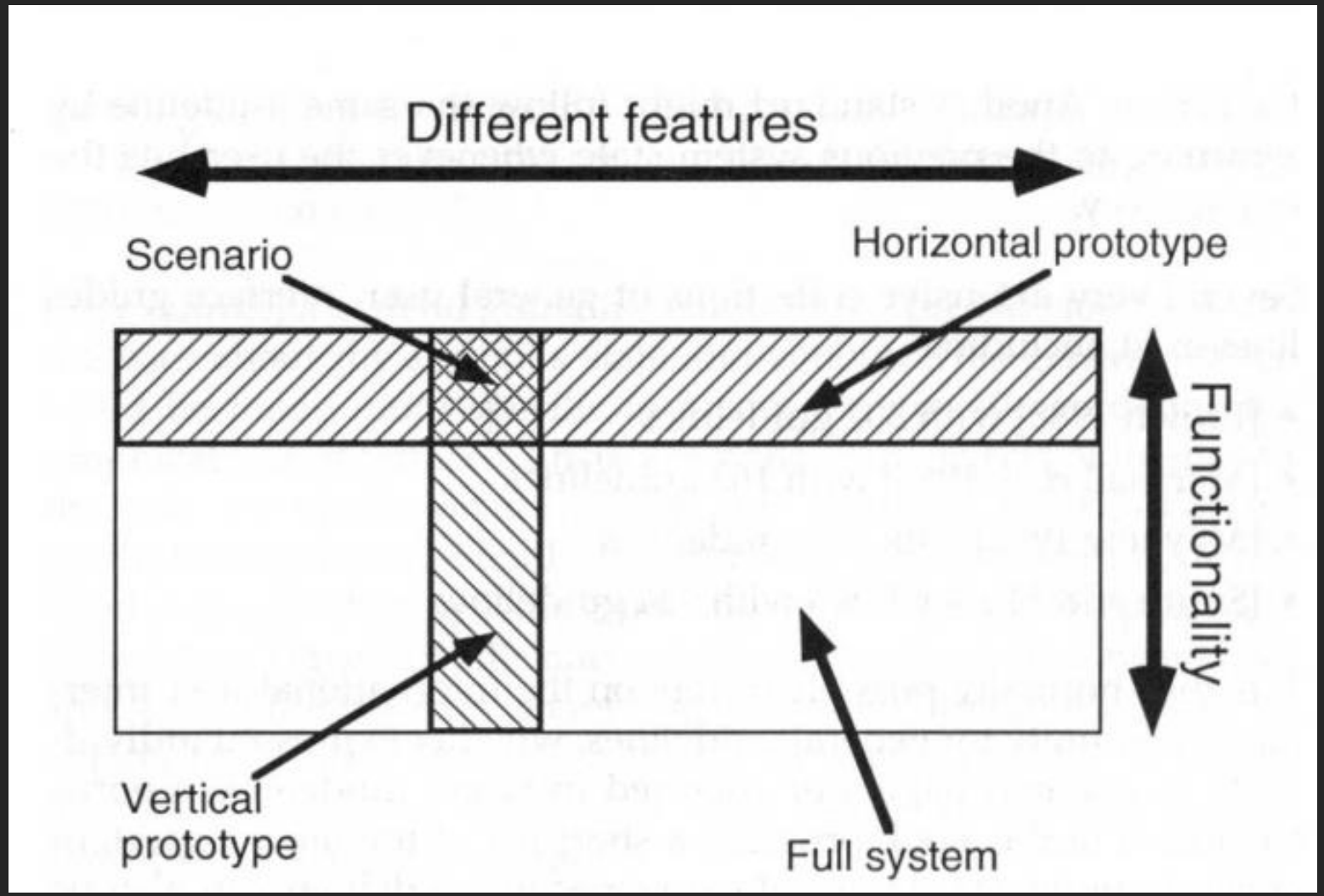
** Varies in complexity

*** You can remove features from a complex game

Psychological experiment

- A lot of experiments are simple
- Add your own ideas to change/improve the experiment
  - Make it customizable
  - Add features
  - Combine experiments
- Stroop-Task
  - add additional difficulty levels
    - Countdown timer
    - Change input modality
- Do not forget to integrate the evaluation of user performance

Vertical & Horizontal prototype



Nielsen, J. (1993). Usability Engineering. Morgan Kaufmann Publishers In. *Inc.–1993*

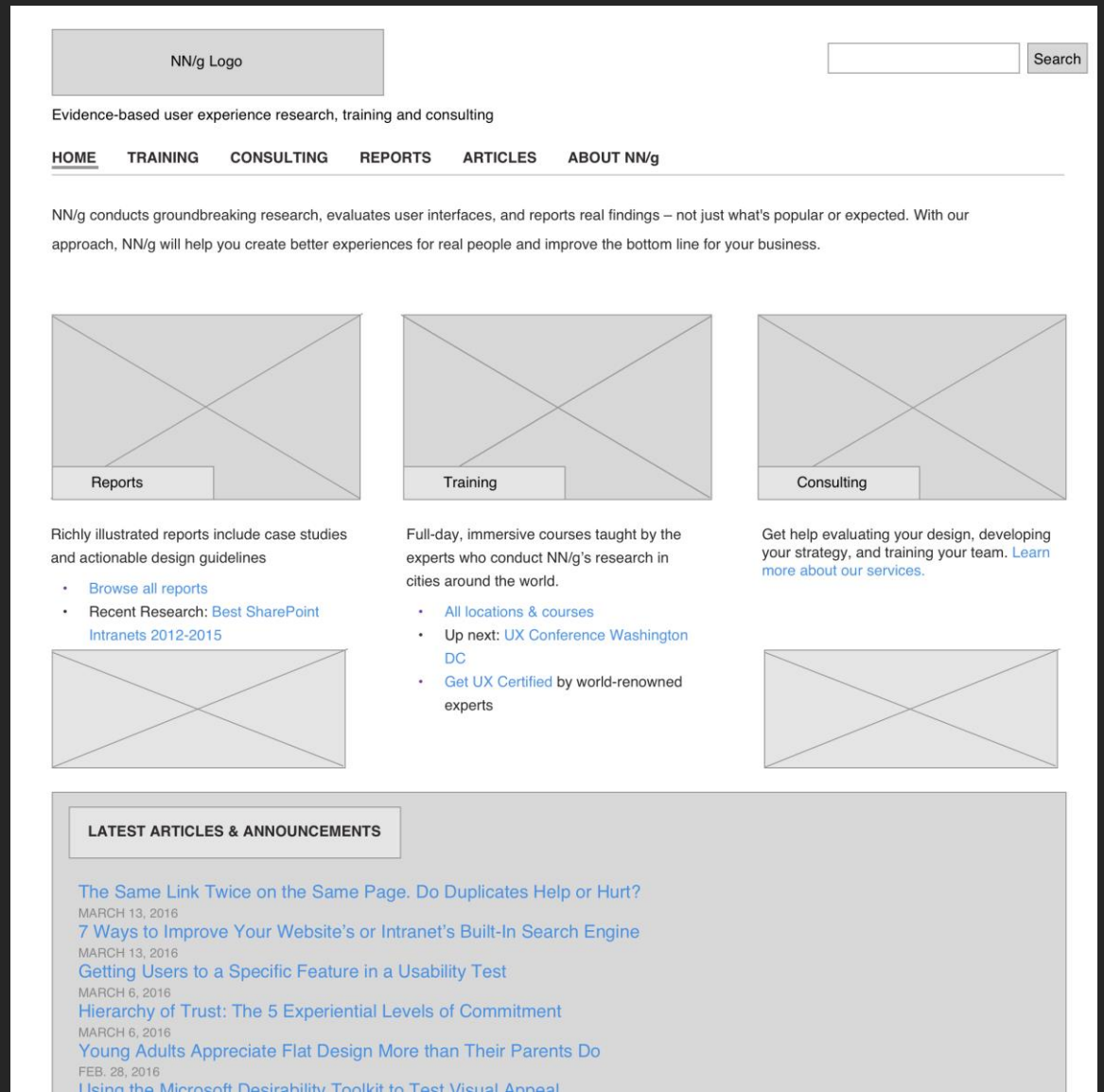
https://www.nngroup.com/articles/wireflows/

Wireframe & Wireflow diagrams

Wireframe diagram

Visualizes the product

Horizontal prototype



https://www.nngroup.com/articles/wireflows/

Wireframe & Wireflow diagrams

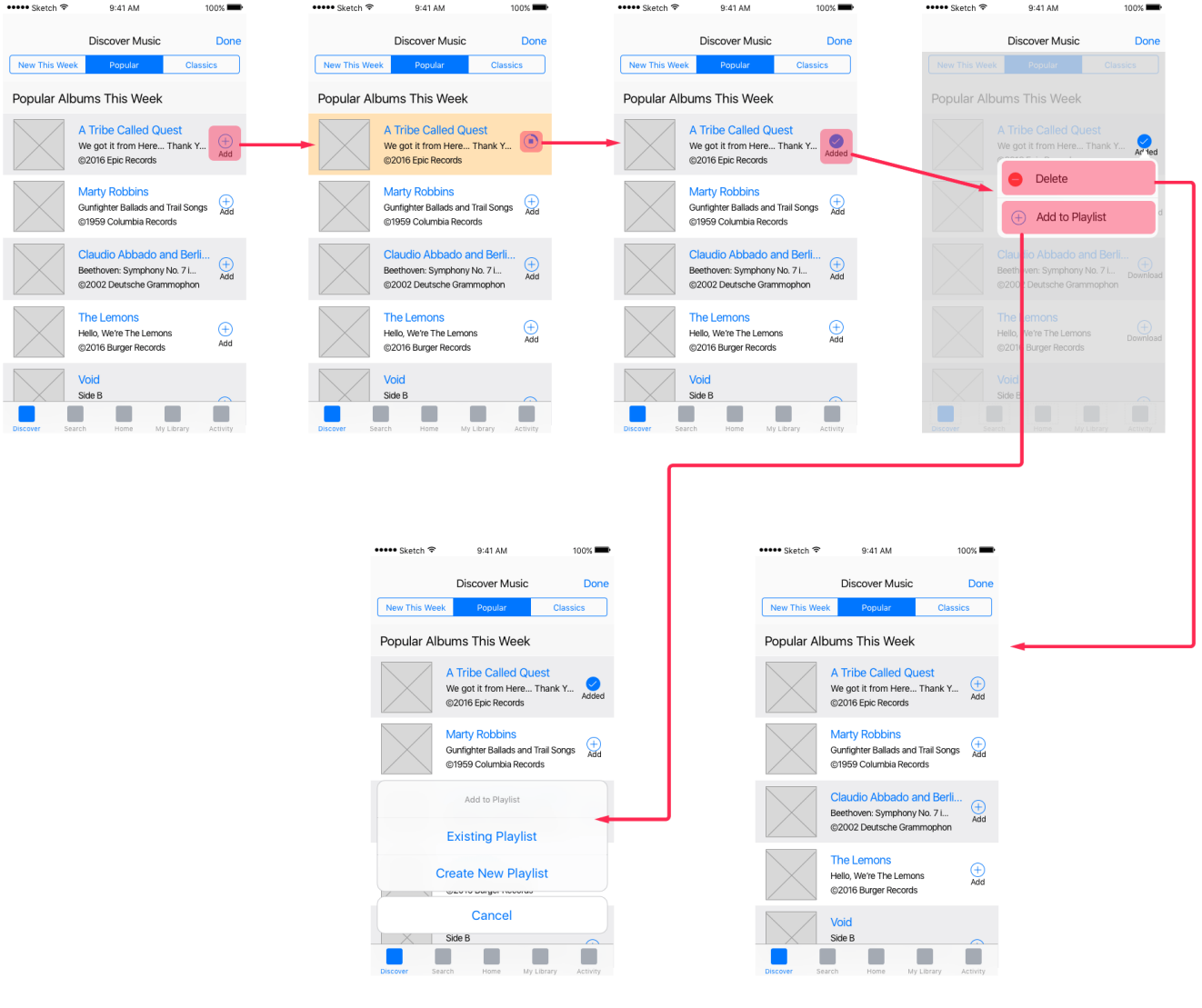
Wireflow diagram

Visualizes the Product experience

From the user's perspective

Interaction flow

Horizontal prototype



Wireframe tools

- Penpot - https://penpot.app/
- Adobe XD - https://www.adobe.com/se/products/xd.html)-
- Axure - installed on Windows computers at IDA
- Figma https://www.figma.com/
- MockFlow - https://www.mockflow.com/
- NinjaMock - https://ninjamock.com/
- Pencil - http://pencil.evolus.vn/
- Wireframe.cc - https://wireframe.cc/

UML Diagrams

- Unified modelling language
- Various different types of diagrams
  - Flow Charts
  - Class Diagram
  - Sequence Diagram
  - Communication Diagram
- Pre-development phase
- Document a process
- Present solution to a problem
- Brainstorm ideas in a meeting
- Design an operation system
- Useful in bigger development teams

UML Diagrams

- Structural diagrams
  - E.g., class diagram, component diagram
- Behavioral diagrams
  - E.g., Activity diagram, flow-charts, state-charts
- Interaction diagrams
  - E.g., component diagram, sequence diagram

From the programmer’s perspective
Models program logic

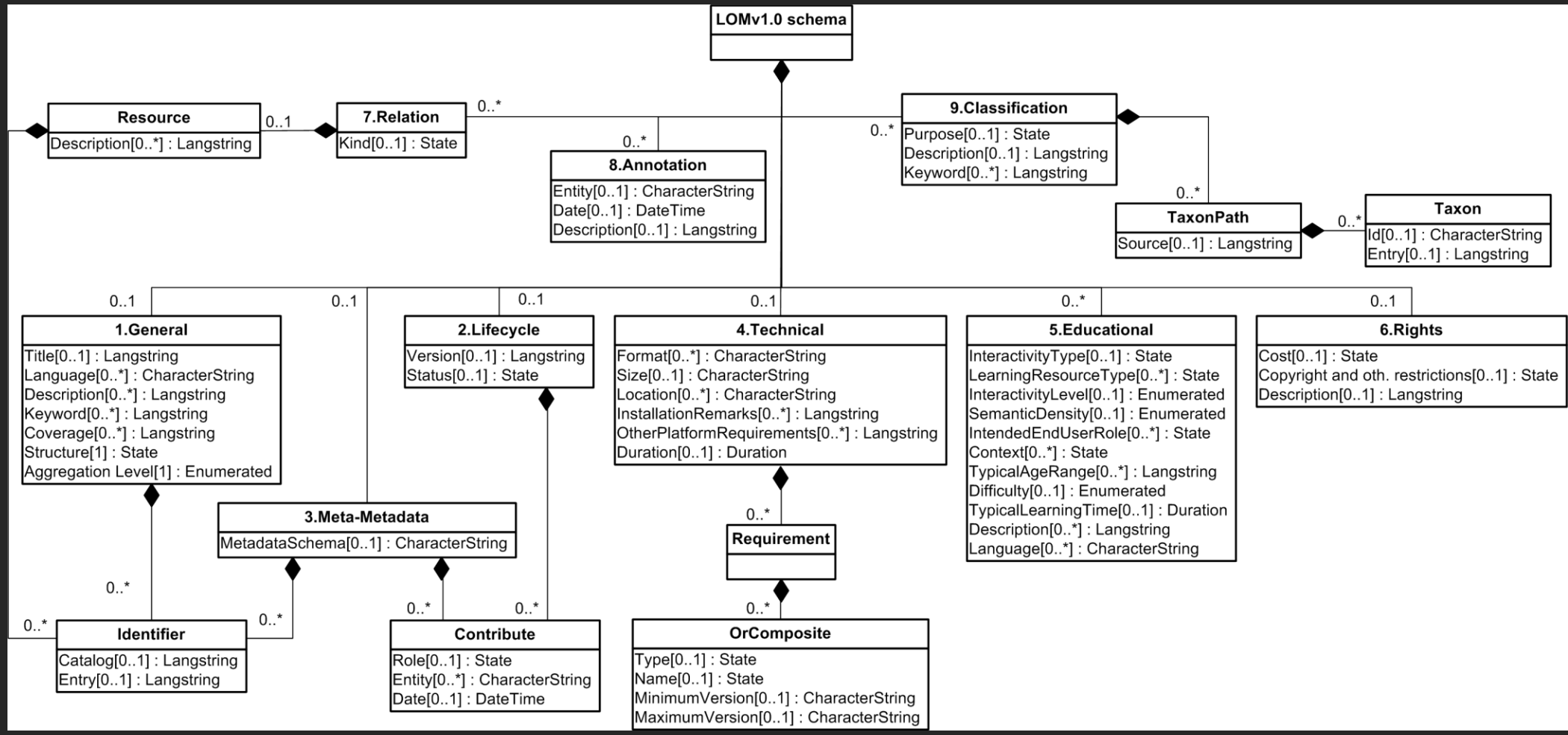
Diagram tools

- https://app.diagrams.net/
- https://inkscape.org/
- https://wireflow.co/
- https://www.yworks.com/products/yed
- MS Visio

Class diagrams (example) [Structural]

+ Public
- Private
Protected
~ Package

Association
Inheritance
Realization / Implementation
Dependency
Aggregation
Composition

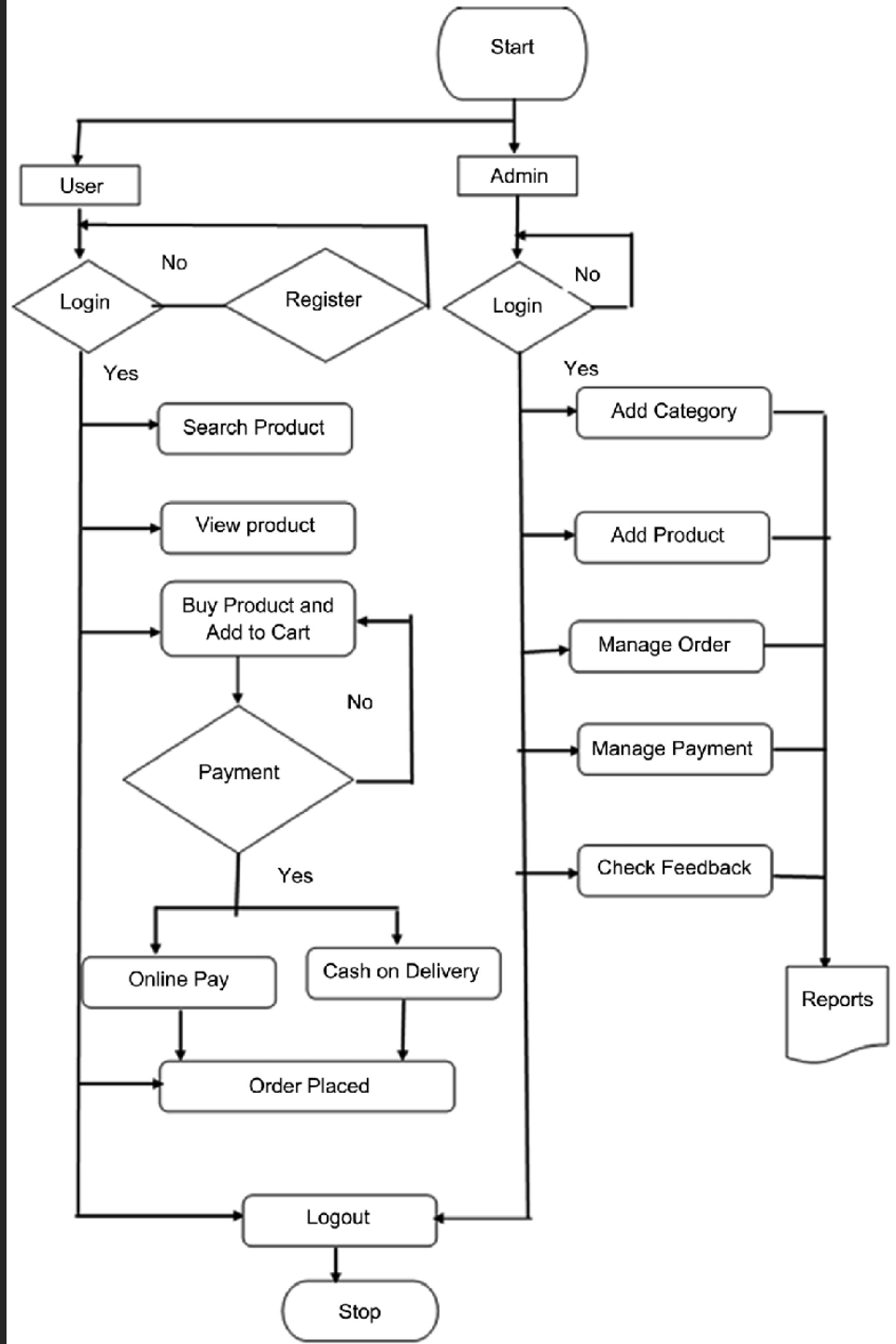


https://commons.wikimedia.org/wiki/File:LOM_base_schema.png

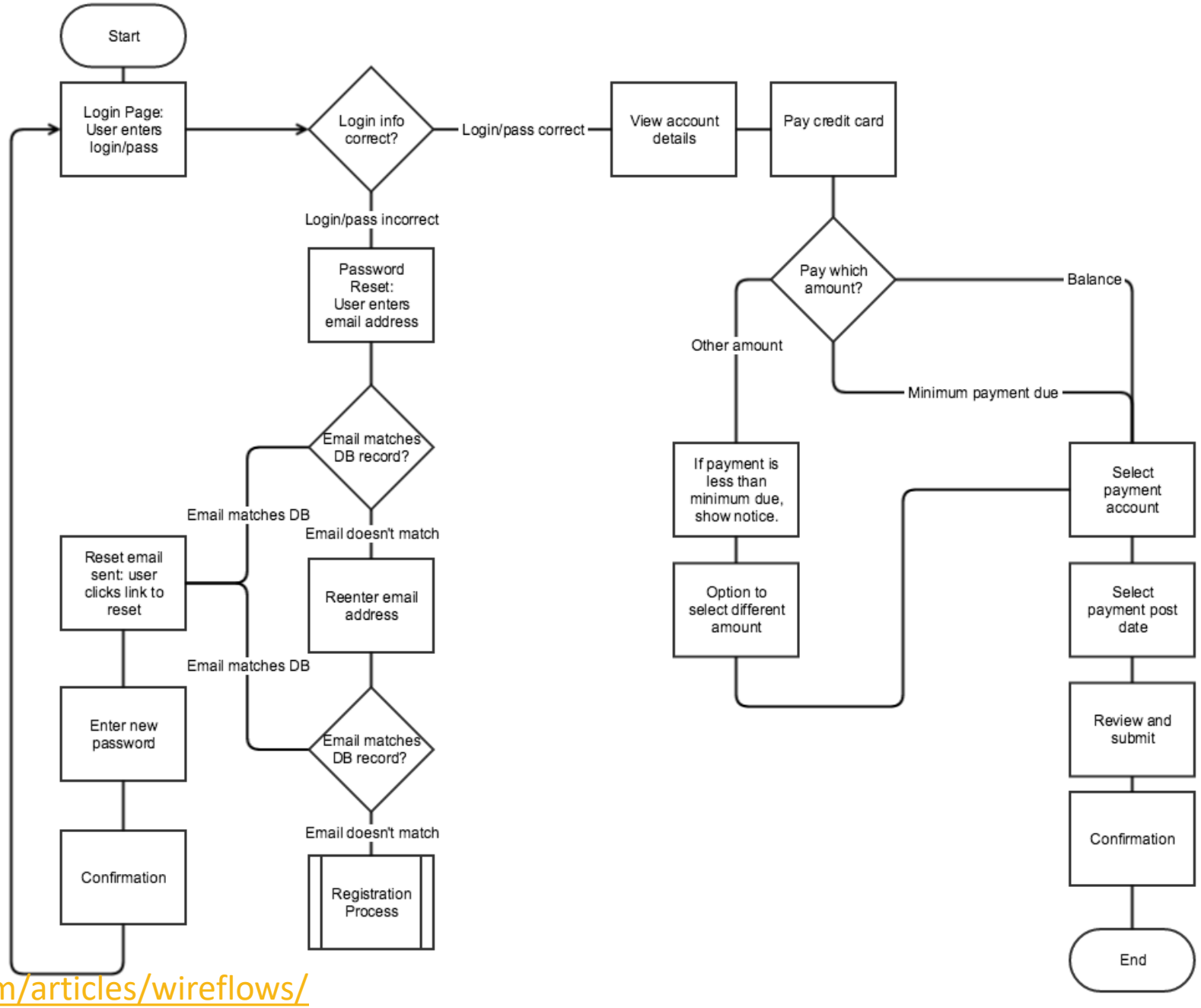
Class diagrams

- Structure Visualization: Depicts the static structure of a system.
- Classes: Represents classes, which are blueprint templates for objects.
- Focus on object-oriented programming
- Attributes: Displays class attributes (variables) that store data.
- Methods: Shows class methods (functions) that define behavior.
- Relationships: Illustrates associations, inheritance, and dependencies between classes.
- System Overview: Provides an overview of data organization and system components.

Flow chart [behavioral]



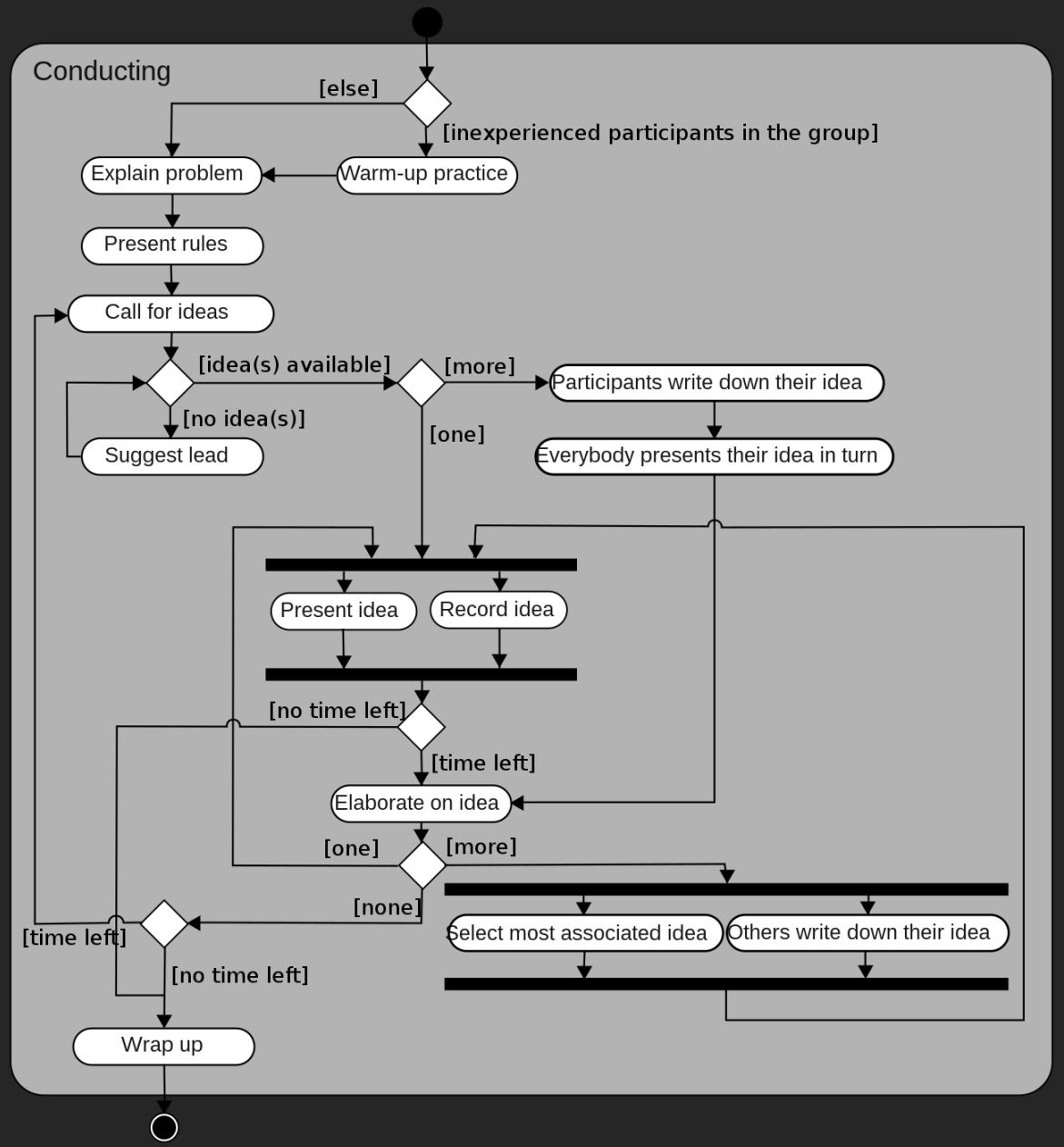
Khan, M. M., Shams-E-Mofiz, M., & Sharmin, Z. A. (2020). Development of E-commerce-based online web application for COVID-19 pandemic. *IBusiness*, *12*(4), 113-126.



https://www.nngroup.com/articles/wireflows/

Activity diagrams [Behavioral]

- **Workflow Representation:**
  - Visualizes workflows and processes.
- **Actions**
  - Depicts actions or tasks within the workflow.
- **Decisions**
  - Includes decision points and branching paths.
- **Concurrent Activities**
  - Shows parallel and synchronized actions.
- **Transitions**
  - Illustrates transitions between activities.
- **Clear Sequence**
  - Highlights step-by-step execution order.



https://en.wikipedia.org/wiki/Activity_diagram#/media/File:Activity_conducting.svg

Activity diagrams [Behavioral]

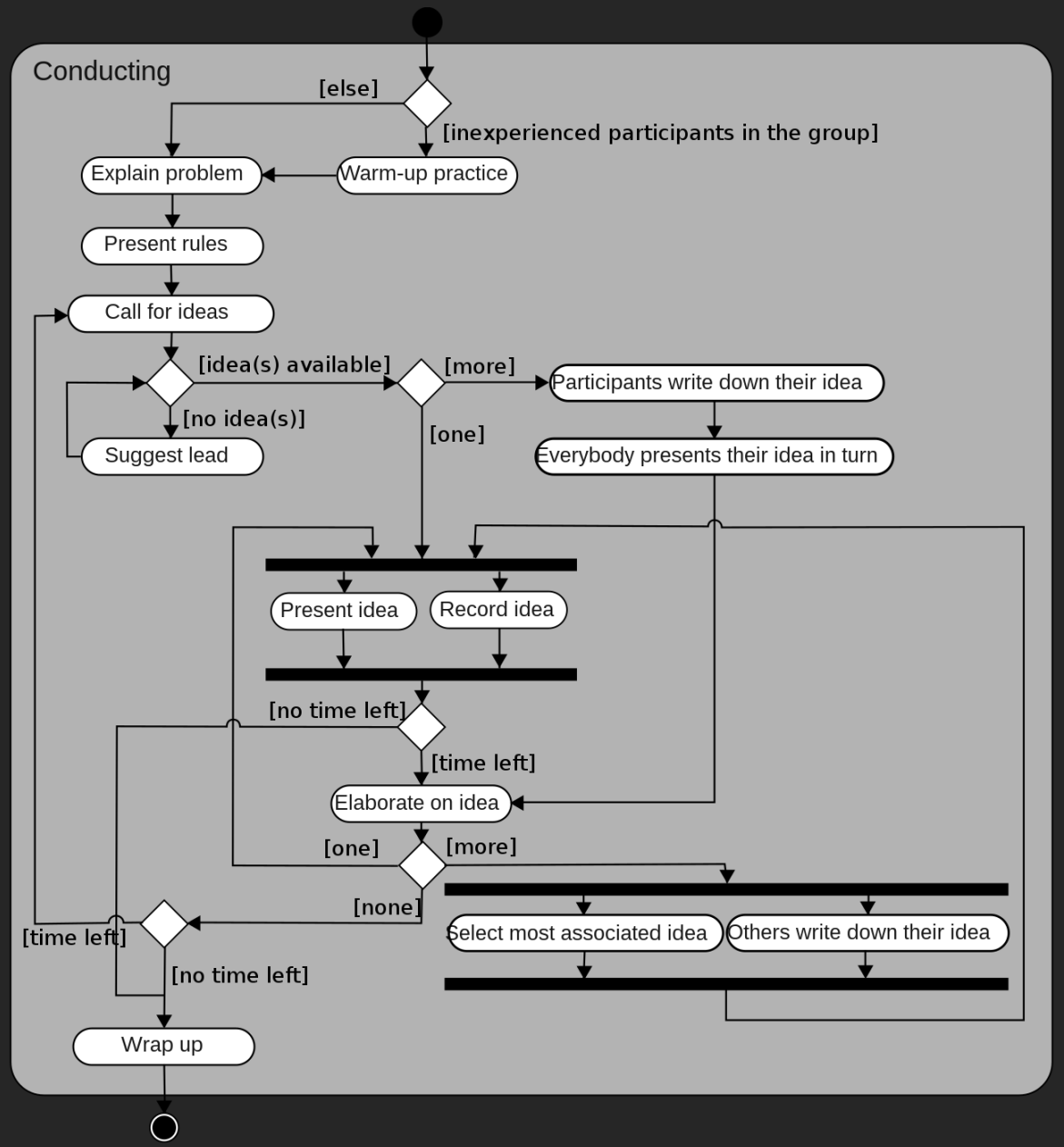
stadia (white boxes) = actions

diamonds represent decisions

bars represent the start (split) or end (join) of concurrent activities;

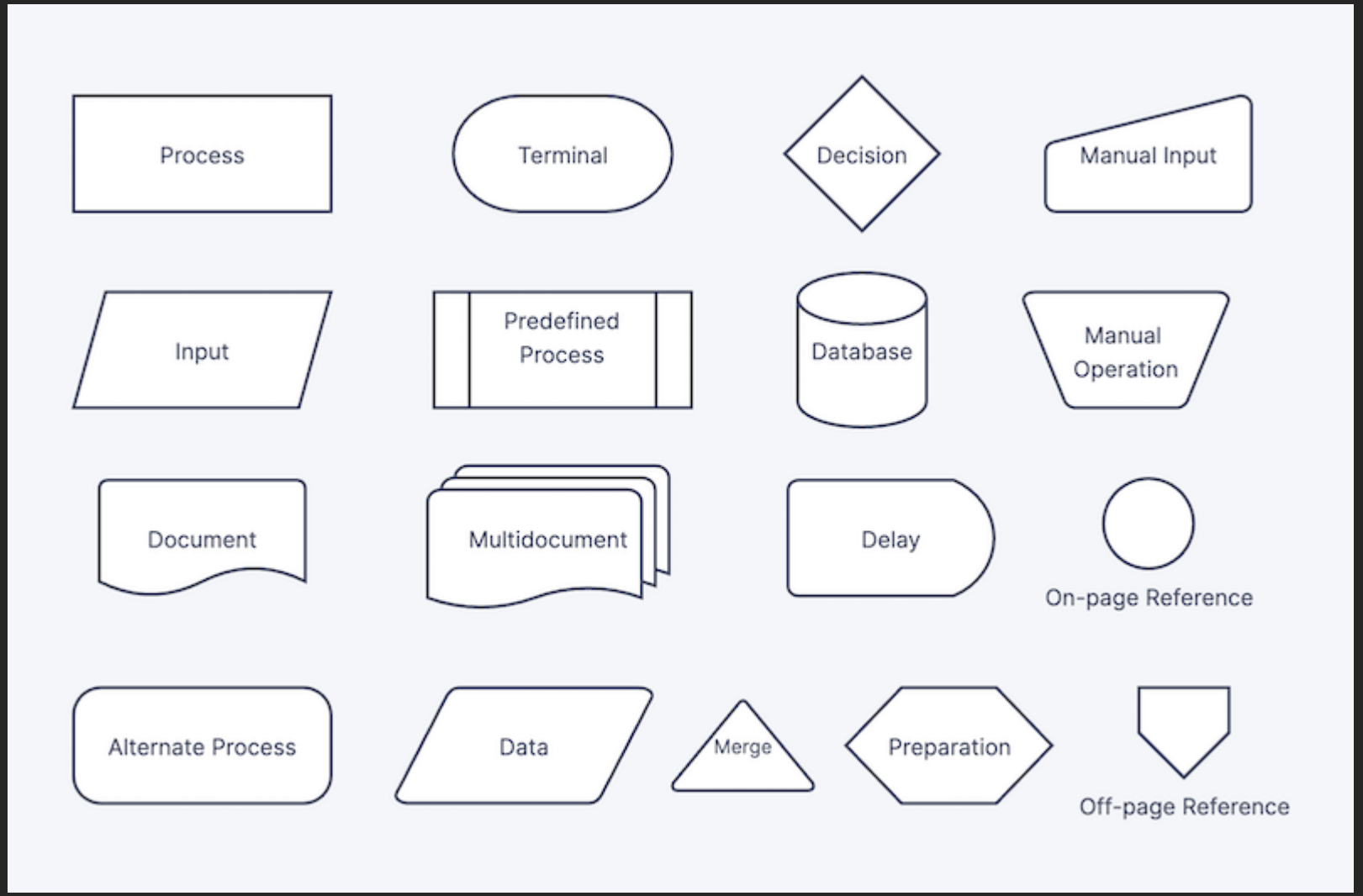
a black circle represents the start (initial node) of the workflow

an encircled black circle represents the end (final node).



https://en.wikipedia.org/wiki/Activity_diagram#/media/File:Activity_conducting.svg

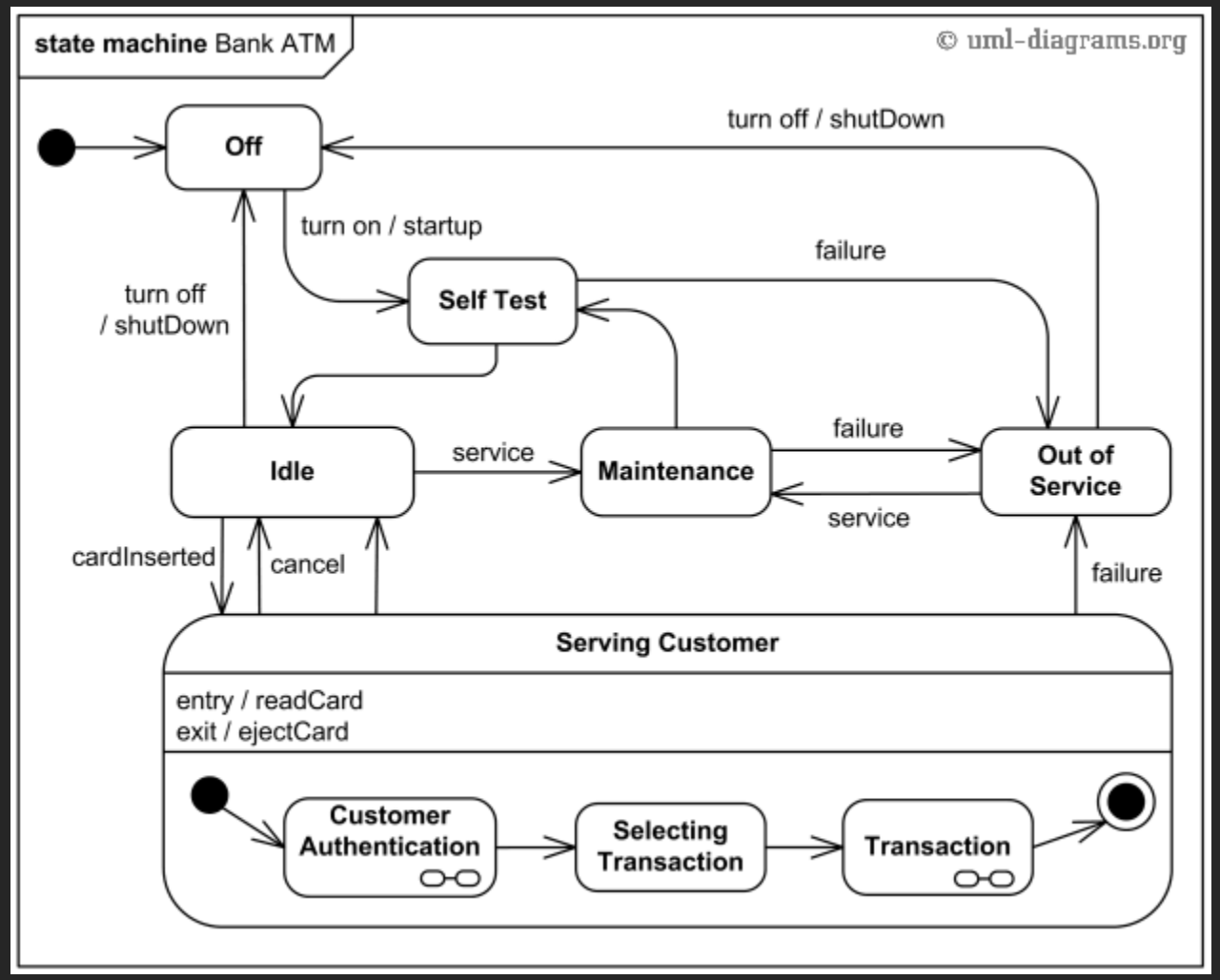
Components



https://www.zenflowchart.com/flowchart

State chart example [Behavioral]

Event / action



LINKÖPING UNIVERSITY

Excursion: Game dev

Simple movement

```
let p_static_y = 0;

const player_static = document.querySelectorAll("div")[1];

function animateStatic() {
  p_static_y += 1;

  player_static.style.left = p_static_y + "px";
  player_static.innerText = parseInt(p_static_y);
  requestAnimationFrame(animateStatic)
}


animateStatic()
```

Problem:
Movement speed dependent on framerate!

https://gitlab.liu.se/729g87/HT2023/ball/
https://ball-729g87-ht2023-a971129c087b15485fd5e41531289e03b2a8d2be7d77.gitlab-pages.liu.se/movement.html

Animate using setTimeout

```
let p_static_y = 0;

const player_static = document.querySelectorAll("div")[1];

function animateStatic() {
  p_static_y += 1;

  player_static.style.left = p_static_y + "px";
  player_static.innerText = parseInt(p_static_y);
  setTimeout(()=>{animateStatic()}, 1000)
}


animateStatic()
```

Problem:
Movement speed dependent on framerate!

Simple movement

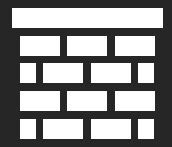
```
let py = 0;
const player =
document.querySelectorAll("div")[0];
const playerSpeed = 60;

let lastT = 0;
function animate() {
    let time = performance.now() / 1000;
    let dt = time - lastT;
    py += playerSpeed * dt;

    player.style.left = py + "px";
    player.innerText = parseInt(py);
    lastT = time;

    requestAnimationFrame(animate);
}
animate();
```

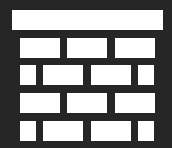
Game Engine



Elements

Attributes:
- Position
- Velocity
- Dimensions
- …

Game Engine



- Position
- Velocity
- Dimensions

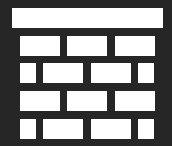
- Position
- Velocity
- Dimensions

- Position
- Velocity
- Dimensions

Elements

Game Engine

Wall-Model



- Position
- Velocity
- Dimensions

Enemy-Model

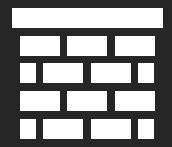
- Position
- Velocity
- Dimensions

Player-Model

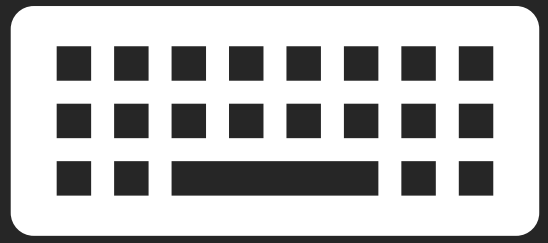
- Position
- Velocity
- Dimensions

Elements

Game Engine



- Position
- Velocity
- Dimensions



- Position
- Velocity
- Dimensions

- Position
- Velocity
- Dimensions

Controller

Elements

Game loop

- read input
- update values
- draw scene

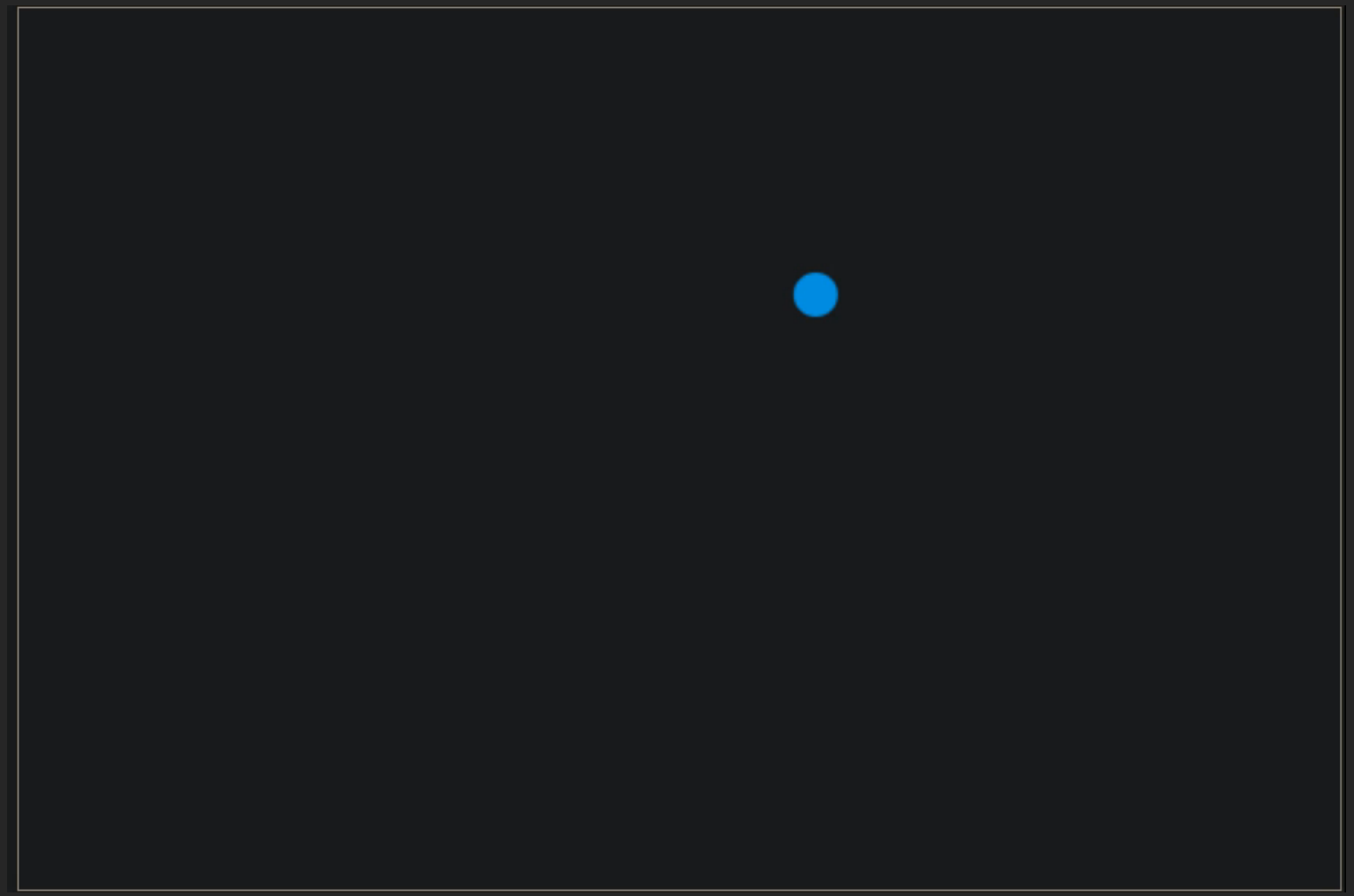
- repeat

Game loop

- read input (user -> controller)
- update values (controller -> model)
- draw scene (model -> renderer)

- repeat (loop/recursion)

The user performs an input (e.g., arrow key is pressed)
The controller registers this and updates the model accordingly (e.g., change position)
The renderer draws the scene based on the model data (draw ball)



https://ball-729g87-ht2023-a971129c087b15485fd5e41531289e03b2a8d2be7d77.gitlab-pages.liu.se/

https://gitlab.liu.se/729g87/HT2023/ball/

Architecture

- game.js
  - Controller
  - Handles player input
- Ball.js
  - Holds the data to render the ball
  - Has position, velocity, radius
  - Has methods to manipulate velocity
- Engine.js
  - Combines rendering and game loop
  - Renders the ball
  - Execures gameloop