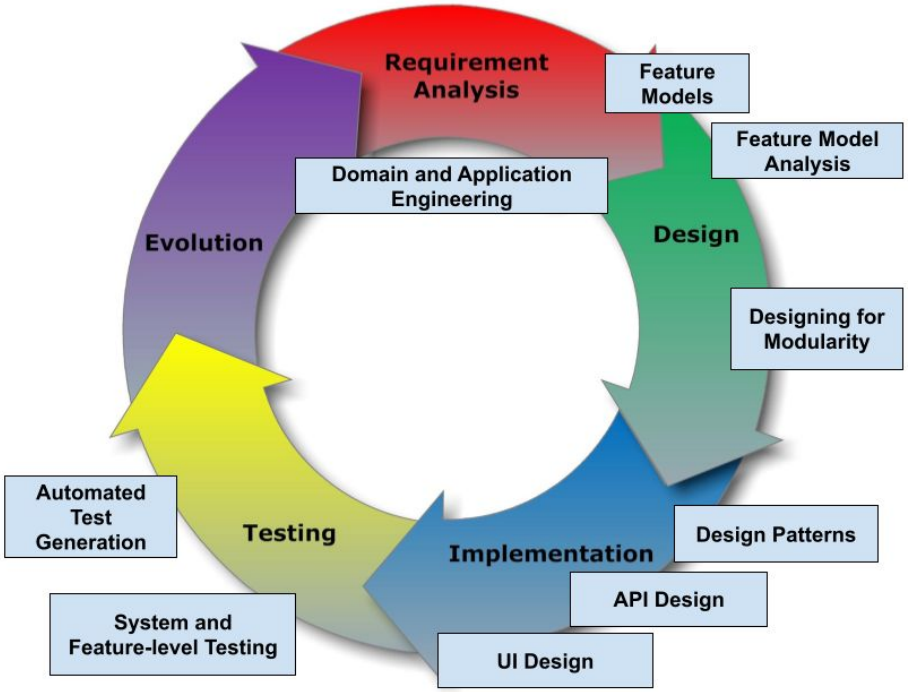
Lecture 14: Course Summary

Gregory Gay
TDA 594/DIT 593 - December 16, 2021

SE Principles for Complex Systems



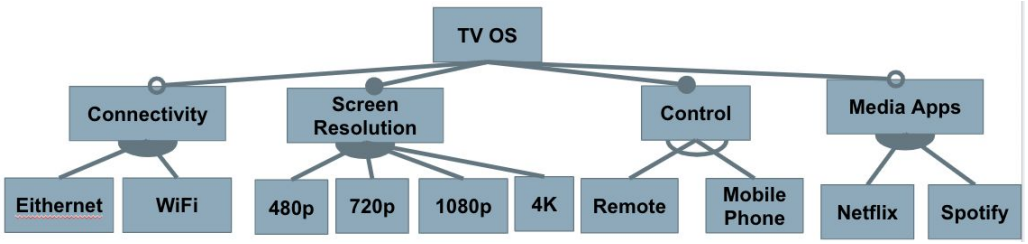
Individual Assignment - Jan 10-14

- Take home exam, 10ish open-ended questions.
  - Will require reading one research paper.
- Will be made available on Canvas at 9:00 on January 10th and due January 14, 23:59.
- Based on topics covered in lectures, more theory-based than group assignments.

Practice Assignment:
https://bit.ly/3oxH3Yc

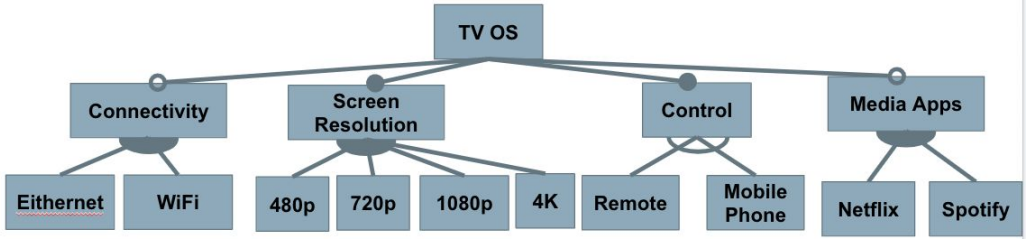
Question 1 - Domain/Application

Your company has developed SPL for smart TVs, according to the following feature diagram:



For each request, decide if you will extend the platform, add the feature to a single application, or decline the request.

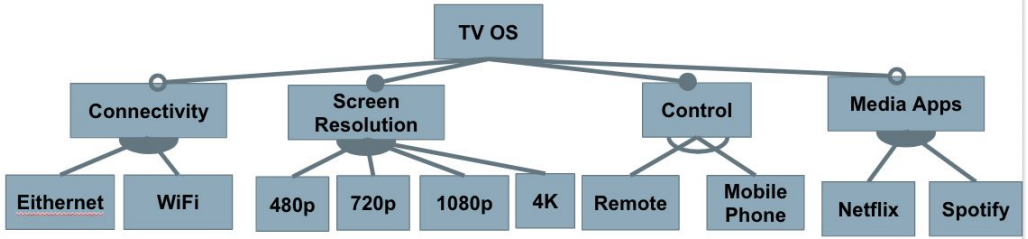
Question 1 - Domain/Application



2K resolution for video games (offering good balance between image sharpness and game speed).

- **Extend the Platform**: Easy to support. Few hardware incompatibilities. Gamers would appreciate the option.
- **One Product:** Limited customer base. Potential hardware incompatibility. Performance will change in the future.

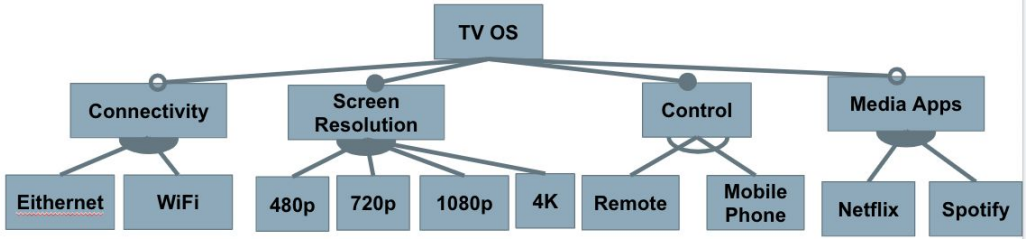
Question 1 - Domain/Application



App Store where developers can publish their own media apps, and customers can select the ones they want.

- **Extend the Platform or Ignore**: Would require extensive effort to create and manage. Would need to work on all hardware. Could earn a lot of additional revenue - so may be worth the effort, but can argue either way.

Question 1 - Domain/Application



TVs for a restaurant chain that display a special app, and have a remote with simplified button layout.

- **Single Product**: Relatively little development work, and may sell a lot of TVs to that restaurant.
- Remote could be added to platform and sold with future products (or as an add-on).

Question 2 - Feature Modelling

You are developing a **word processor** as a software product line, so that you can easily provide different feature sets for different types of customers.

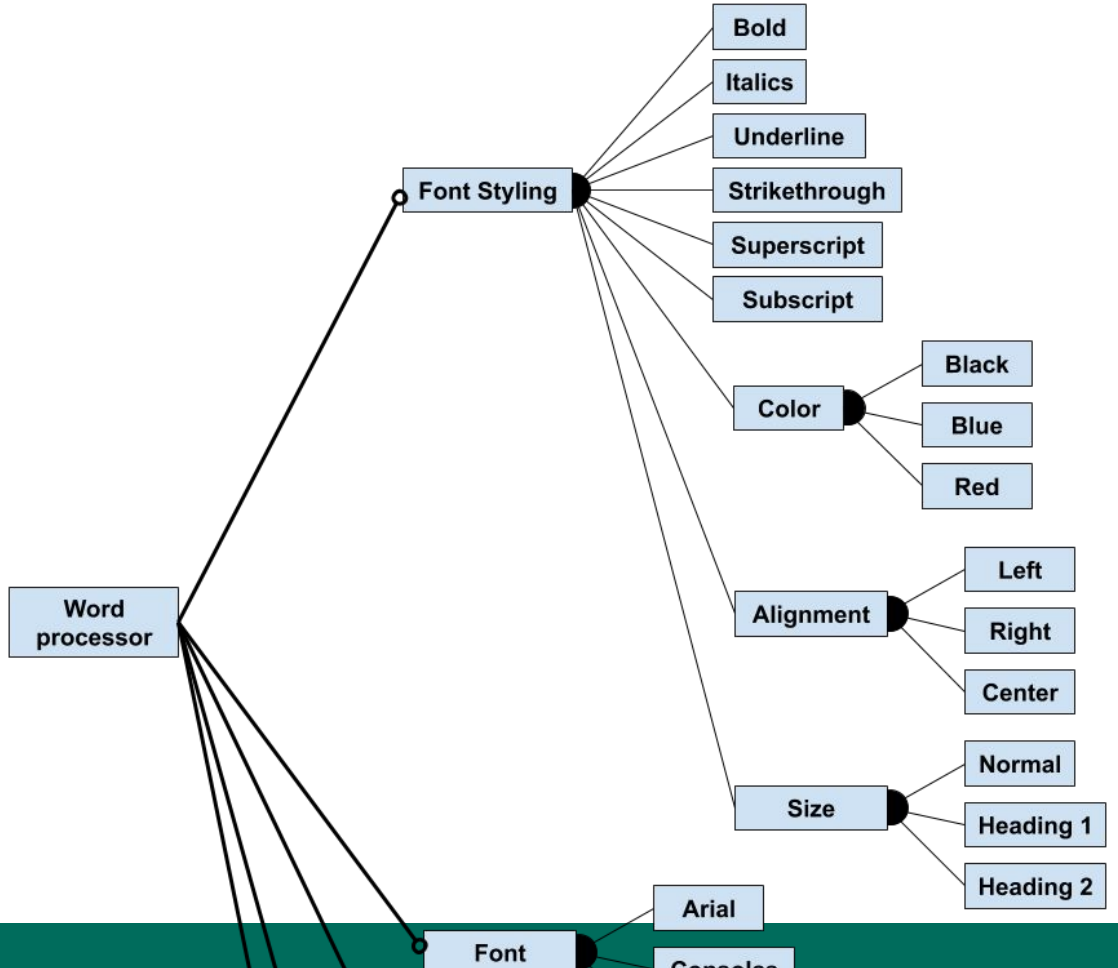
1. Analyze the domain and identify a set of features.
2. Model the domain with a feature diagram.

Question 2 - Feature Modelling

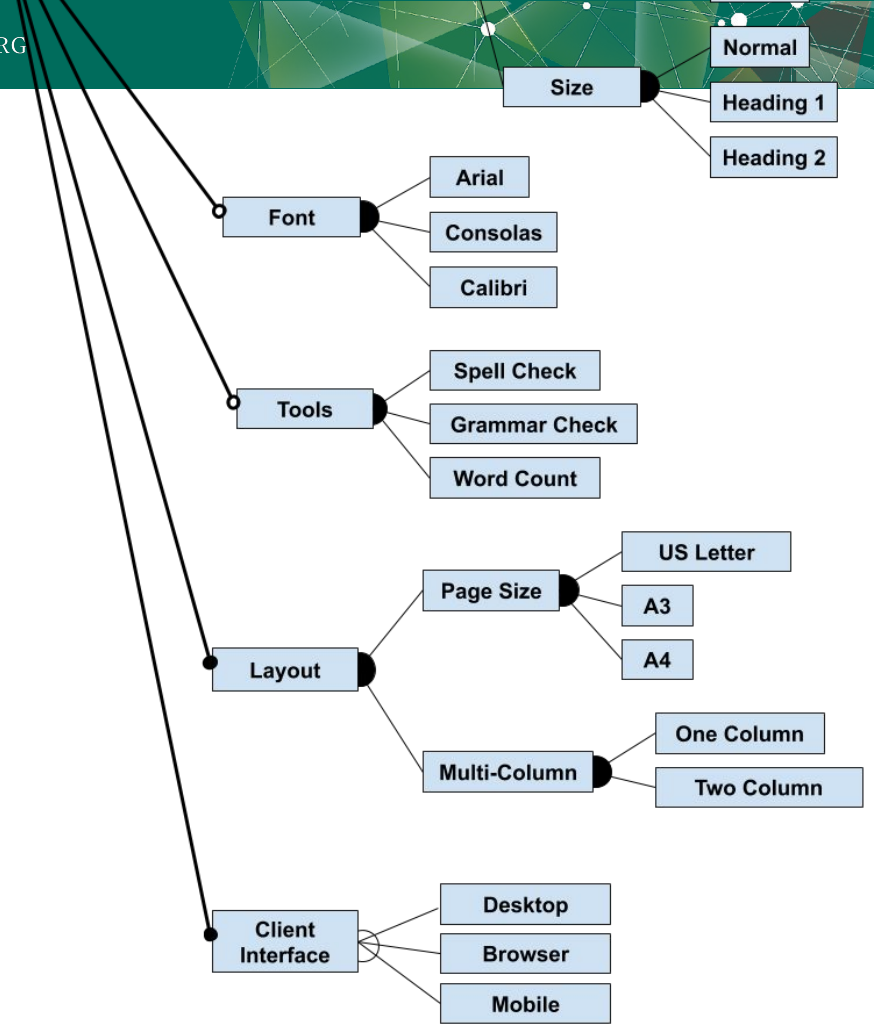
Some items to consider:

- Which features will be requested by many customers?
- Which features will be requested only by a few customers?
- Which features could distinguish your products from others?
- Pay attention to feature dependencies and make sure you capture cross-tree constraints and model structures.

Question 2

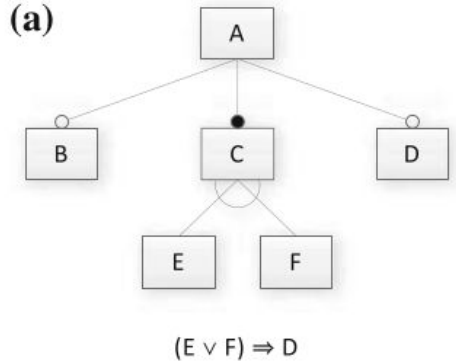

Word processor
Font Styling
Bold
Italics
Underline
Strikethrough
Superscript
Subscript
Color
Black
Blue
Red
Alignment
Left
Right
Center
Size
Normal
Heading 1
Heading 2
Font
Arial

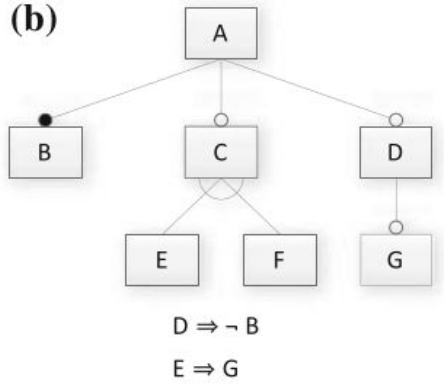
Question 2

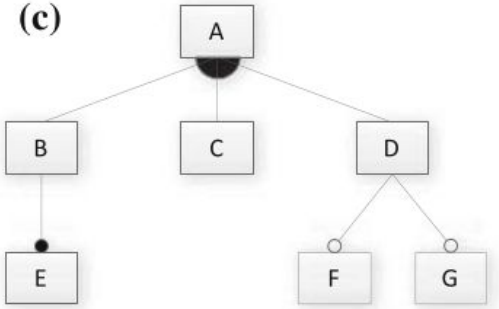

Size
Normal
Heading 1
Heading 2
Font
Arial
Consolas
Calibri
Tools
Spell Check
Grammar Check
Word Count
Layout
Page Size
US Letter
A3
A4
Multi-Column
One Column
Two Column
Client Interface
Desktop
Browser
Mobile

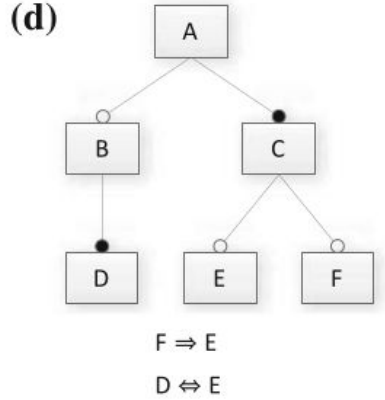
Question 3

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?



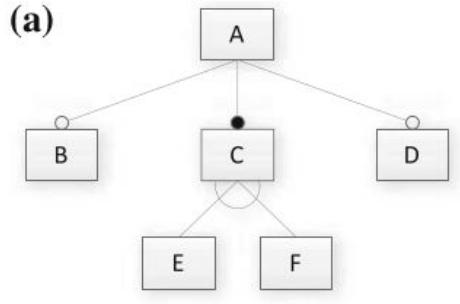






Question 3 (A)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?

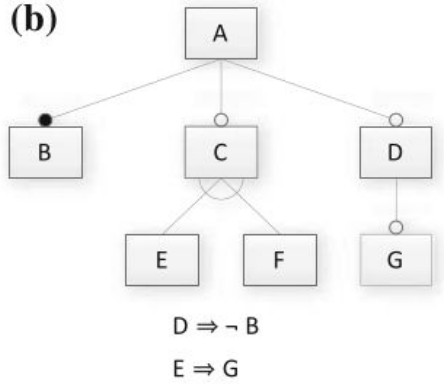


$A \land (B \Rightarrow A) \land (C \Leftrightarrow A) \land (D \Rightarrow A) \land$
$((C \Leftrightarrow (E \lor F)) \land \neg(E \land F)) \land ((E \lor F) \Rightarrow D))$

- Valid: A, B, C, D, F ; A, C, D, E
- Invalid: A, B, C, D, E, F ; A, B, C, E
- Is it consistent: Yes

Question 3 (B)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?



A ∧ (B ⇔ A) ∧ (C ⇒ A) ∧ (D ⇒ A) ∧
((C ⇔ (E ∨ F)) ∧ ¬(E ∧ F)) ∧ (G ⇒ D) ∧ (D ⇒ ¬B) ∧
(E ⇒ G)

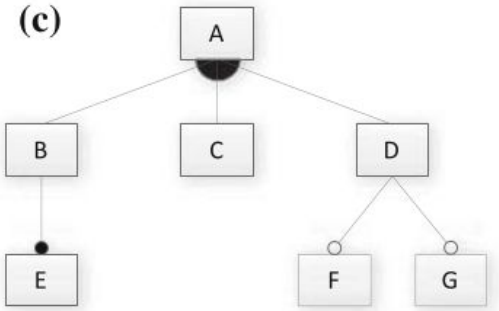
- Valid: A, B ; A, B, C, F
- Invalid: A, B, D, G ; A, B, C, E
- It is consistent: Yes, but D, E, and G are dead features (because B is mandatory).

Question 3 (C)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?

A ∧ ((B ∨ C ∨ D) ⇔ A) ∧ (E ⇔ B) ∧ (F ⇒ D) ∧ (G ⇒ D)

- Valid: A, C ; A, B, C, D, E, F, G
- Invalid: A, B, C; A, C, E
- It is consistent: Yes (just remember that B and E need to come as a pair)

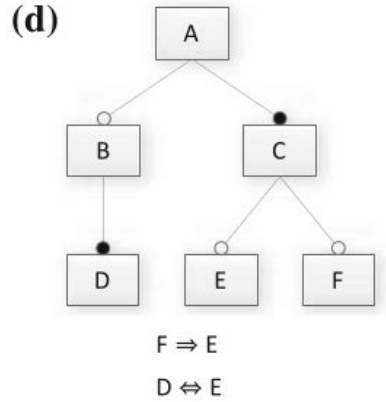


Question 3 (D)

- Translate model into propositional logic formula.
- Provide two valid and two invalid features.
- Is it consistent? If not, why not?

A ∧ (B ⇒ A) ∧ (C ⇔ A) ∧ (D ⇔ B) ∧ (E ⇒ C) ∧ (F ⇒ C) ∧
(F ⇒ E) ∧ (D ⇔ E)

- Valid: A, C ; A, B, C, D, E
- Invalid: A, B, C, D ; A, C, F
- It is consistent: Yes, but remember that if you have F, you need E, D, and B as well.



Question 4 - Implementation

Consider compile-time and load-time binding of variability decisions.

1. Define each and note how they differ.
2. Explain potential advantages and disadvantages.

Question 4 - Implementation

Compile-time Binding:

- Features selected when code is compiled.
  - Preprocessors.
- Unselected features removed from code.
  - Faster code, lower system requirements, more secure.

Question 4 - Implementation

Load-time Binding:

- Features selected when program executed.
  - Parameters, design patterns, frameworks, components can do this (or run-time).
  - Configuration file, command line
- Flexible, user can change settings.
- Slower, insecure.

Question 4 - Implementation

Discuss best binding times are suitable for:

- Multiple alternative localization features for the GUI of a satellite navigation system.
  - (language selection, metric versus imperial units, etc.)

Run-time:

User may want to change preferences without a reboot.

Multiple users may share same device, could customize options for each profile.

Question 4 - Implementation

Discuss best binding times are suitable for:

- Two alternative sorting features in a database:
  - Very fast or power- efficient sorting algorithm.

Load or Run-time:

Switch based on current needs.

If run-time, can switch based on battery state (but could be issues if switched at runtime)

Question 4 - Implementation

Discuss best binding times are suitable for:

- Two alternative features in an operating system:
  - Single-processor support and multi-processor support.

Compile or Load-time:

Hardware can’t change without reboot.

Could change hardware and reboot (load-time), but this is often compile-time (company sells different products).

Question 4 - Implementation

Discuss best binding times are suitable for:

- Two alternative features for edge representations in a library of graph algorithms:
  - directed and undirected edges.

Compile or Load-time:

Other features depend on edge type, so issues at run-time.

Compile-time could be efficient if changes infrequent.

Question 4 - Implementation

Discuss best binding times are suitable for:

- Multiple optional features for supported file formats in printer firmware.

Compile-time:

The file types are unlikely to change often.

Embedded system benefits from simple/fast executable.

Question 5 - Design Patterns

Choose one of: strategy, decorator, factory, facade, adapter, and template method pattern.

1. Describe the goal of the pattern and how it is applied to a system.
2. Describe how the pattern would help enable controlled variability.
3. Give an example of a system that would benefit from the application of this pattern.

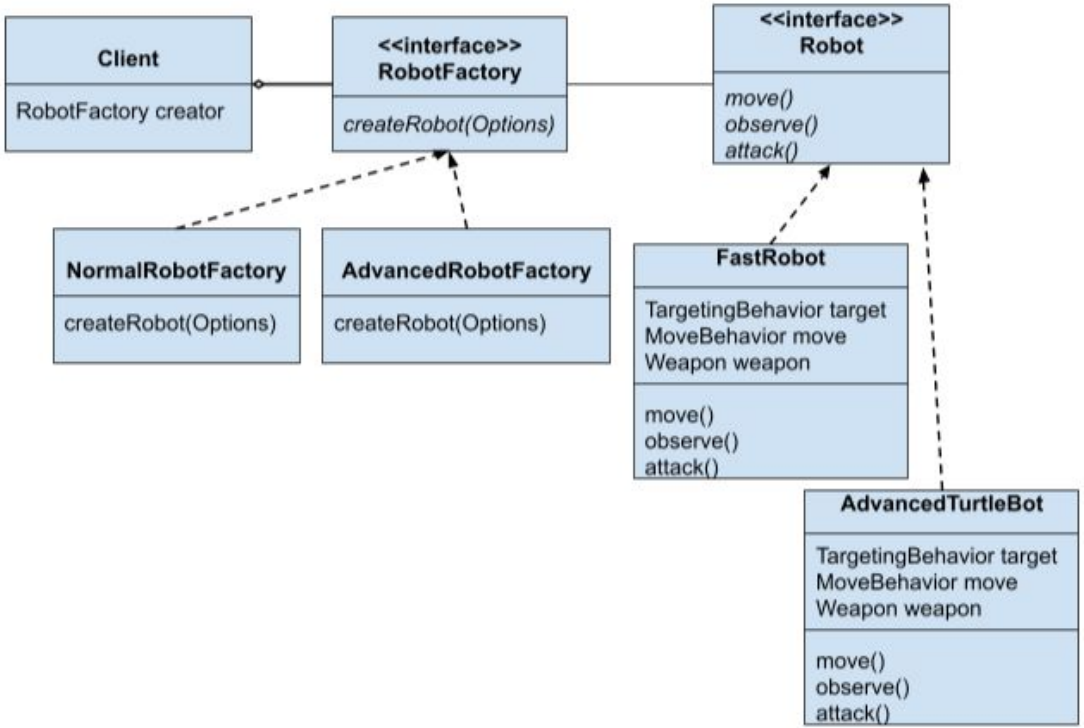
Let’s Take a Break

Question 5 - Design Patterns

Factory Pattern

- Performs object creation based on selected options
- Returns an object (Product) on request.
  - All Products implement a common interface.
  - New Products can be added, or existing ones changed.
  - Client does not need to know details of object creation or which object it is interacting with.

Question 5 - Design Patterns



Question 6 - Modularity/APIs

“Let’s Make a Deal” is a game where contestants are presented with three doors.

- One leads to a prize, the others lead to nothing.
- Users select one door.
- Host opens one of the other doors.
- Users can then choose to open their door or the remaining unopened door.

Question 6 - Modularity/APIs

Implement Let's Make a Deal as a web service:

- Creation of games.
- User selection of a door.
- The game will open one of the other doors.
- User opening of a door.
- Querying of the current state of the game and outcome (if complete) by user.
- Deletion of a game.

Question 6 - Modularity/APIs

1. Create a REST API for this game.
   a. Determine the appropriate resources and verbs, and explain your API.
2. Extend your API into a generic, reusable API that could be used as the interface for additional games.
   a. Redesign your interface as a generic “game” interface.
   b. Explain why your new design could be reused for a different game.

Question 6 - Modularity/APIs

Resource	Verb
/games	get – status of games server post – create new game
/games/{gid}	get – status of game (in_play, won, lost) delete – delete the game resource
/games/{gid}/doors	get – status of all doors
/games/{gid}/doors/{1..3}	get – door status {closed, selected, opened} put – update door status

Once game is over, only GET is allowed for {gid}

Question 6 - Modularity/APIs

Resource	Verb
/games	get – status of games server post – create new game
/games/{gid}	get – status of game **{results based on game}** delete – delete the game resource
/games/{gid}/items	**get – status of any in-game item** **post - create a new item (requires authentication)**
/games/{gid}/items/{iid}	**get – item status {results based on game}** **put – update item status** **delete - delete the item resource (requires authentication)**

Question 7 - Testing

`find(pattern,file)`

- Finds instances of a pattern in a file
  - **`find(“john”,myFile)`**
    - Finds all instances of `john` in the file
  - **`find(“john smith”,myFile)`**
    - Finds all instances of `john smith` in the file
  - **`find(““john” smith”,myFile)`**
    - Finds all instances of `“john” smith` in the file

Question 7 - System Testing

- Parameters: pattern, file
- What can we vary for each?
  - What can we control about the pattern? Or the file?
- What values can we choose for each choice?
  - **File name:**
    - File exists with that name
    - File does not exist with that name
- What constraints can we apply between choice values? (if, single, error)

Question 7 - System Testing

$(2^2*3^3*4^1) = 108$ test specifications

- Pattern size:
  - Empty
  - single character
  - many characters
  - longer than any line in the file
- Quoting:
  - pattern has no quotes
  - pattern has proper quotes
  - pattern has improper quotes (only one “)
- Embedded spaces:
  - No spaces
  - One space
  - Several spaces

- File name:
  - Existing file name
  - no file with this name
- Number of occurrence of pattern in file:
  - None
  - exactly one
  - more than one
- Pattern occurrences on any single line line:
  - One
  - more than one

ERROR and SINGLE Constraints

4 (error) + 2 (single) + (1^2*2^3*3^1) = 30

- Pattern size:
  - [error] Empty
  - single character
  - many character
  - [error] longer than any line in the file
- Quoting:
  - pattern has no quotes
  - pattern has proper quotes
  - [error] pattern has improper quotes (only one “)
- Embedded spaces:
  - No spaces
  - One space
  - Several spaces

- File name:
  - Existing file name
  - no file with this name [error]
- Number of occurrence of pattern in file:
  - None
  - exactly one [single]
  - more than one
- Pattern occurrences on target line:
  - One
  - more than one [single]

37

IF Constraints

4 (error) + 2 (single) + (1^3*2^3) (quoted = true) + (1^4*2^2) (quoted = false) = 18

- Pattern size:
  - [error] Empty
  - single character
  - many character
  - [error] longer than any line in the file
- Quoting:
  - pattern has no quotes
  - [property quoted] pattern has proper quotes
  - [error] pattern has improper quotes (only one “)
- Embedded spaces:
  - No spaces
  - [if quoted] One space
  - [if quoted] Several spaces

- File name:
  - Existing file name
  - no file with this name [error]
- Number of occurrence of pattern in file:
  - None
  - exactly one [single]
  - more than one
- Pattern occurrences on target line:
  - One
  - more than one [single]

38

Question 8 - Combinatorial Testing

Allow Content to Load	Notify About Pop-Ups	Allow Cookies	Warn About Add-Ons	Warn About Attack Sites	Warn About Forgeries
Allow	Yes	Allow	Yes	Yes	Yes
Restrict	No	Restrict	No	No	No
Block		Block			

- Full set of test specifications = 144
- Create set covering all pairwise value combinations.
  - Hint: Start with two variables with most values. Add one variable at a time.

Question 8 - Combinatorial Testing

Allow Content	Allow Cookies	Pop-Ups	Add-Ons	Attacks	Forgeries
Allow	Allow	Yes	Yes	Yes	Yes
Allow	Restrict	No	No	Yes	No
Allow	Block	No	No	No	Yes
Restrict	Allow	Yes	No	No	No
Restrict	Restrict	Yes	-	-	Yes
Restrict	Block	No	Yes	Yes	No
Block	Allow	No	-	-	Yes
Block	Restrict	-	Yes	No	-
Block	Block	Yes	No	Yes	No

Question 9 - Automation

Metaheuristic search techniques can be divided into local and global search techniques.

1. Define “local” search and “global” search.

- **Local search:** formulate a solution, and improve by making small changes.
- **Global search:** more than one solution at a time, and freely change those solutions.

Question 9 - Automation

Metaheuristic search techniques can be divided into local and global search techniques.

2. Contrast the two approaches. What are the strengths and weaknesses of each?

Local search: Fast, easy to understand. Depend on initial guess to not get stuck.

Global search: Do not get stuck easily, but are slower.

Question 9 - Automation

3. Choose and explain an algorithm.

Simulated Annealing:

- Generate random initial solution.
- Each round, pick a neighbor.
  - If better, make it new solution.
  - If worse, make it solution based on result of a probability.
- Probability of accepting worse solution decreases over time - allows rapid early exploration.

Question 10 - Research in SPLs

Read the following research paper: “Variability Management With The Virtual Platform”

1. What problem are the authors attempting to address?
2. Why is this problem important to address?
3. What did the authors do to address this problem?
4. What conclusions did they come to?
5. What is one thing you think could be done to extend this work in the future?

Question 10 - Research in SPLs

What problem are the authors addressing?

- Clone & Own: New variant created by cloning existing code and making changes.
  - Flexible, inexpensive to start, but hard to maintain and does not scale to large number of variants.
- Product line development: Asset platform, compose new products from platform
  - Expensive upfront effort, but scales well.
- Authors want to ease transition.

Question 10 - Research in SPLs

Why is this problem important to address?

- Very expensive to start with a platform.
- Many developers *cannot* start with SPL.
- Transition is difficult and failure has high consequences (bad or cancelled products).
- Simple, incremental transition methods can reduce that risk.

Question 10 - Research in SPLs

What did the authors do to address this problem?

- Virtual platform collects metadata (features, feature locations, clone traces).
- Metadata used in operators available to developers
  - Clone and feature creation/evolution/management.
- Operators allow management of clones, transition to platform.

Question 10 - Research in SPLs

What conclusions did they come to?

- Costs - maintaining features, dealing with omissions during feature maintenance.
- Benefits - reduced manual feature location, clone detection and maintenance.
- Benefits far outweigh costs.
- Reduced effort from pure clone & own.

Question 10 - Research in SPLs

What is one thing you think could be done to extend this work in the future?

- Use metadata to train a prediction model.
  - Could predict feature-to-asset mappings.
  - Could suggest operators to the developer that they may want to apply.
    - (You are moving code - perhaps you should use the “move asset” operator?)
- You do NOT need to be an expert - just think a little and try to be creative!

Wrap-Up

- ***Thank you for making this a great course!***
- Any remaining questions?

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY