Interfaces and System Design

CITS4401 Software Requirements and Design

Week 9

Recap

1. Introduction to various Software Architectures

2. Using Design Rationale to document the system

Goal of This Week

- **Interfaces**
  - **What is interface design?**
  - **HCI**
  - **Hardware, Software**
  - **API**
  - **System design with interfaces: top down or bottom up**

Interfaces

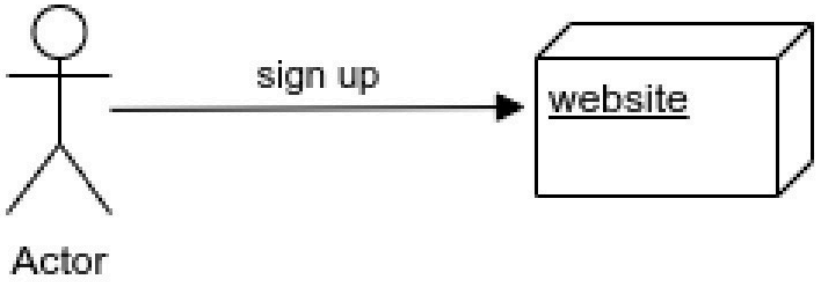
Interfaces

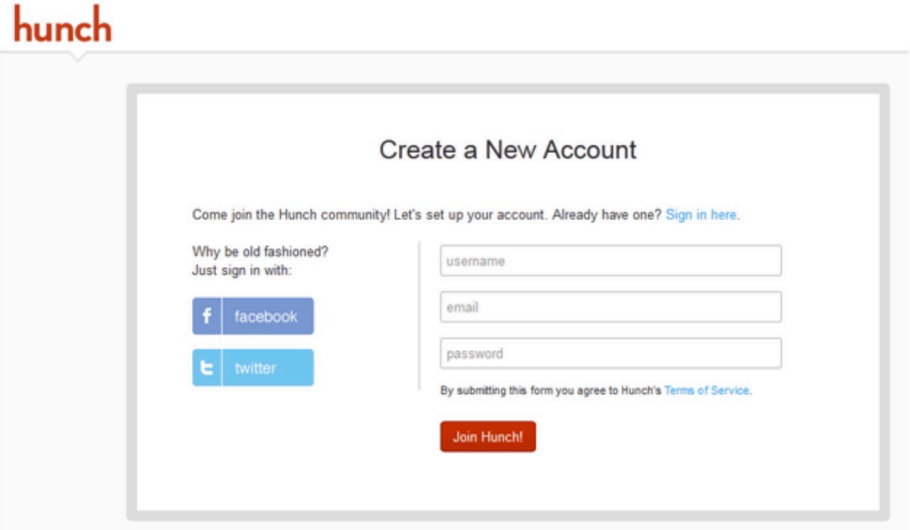
What is an interface?

It just means “the *boundary* where two things meet”

User Interfaces

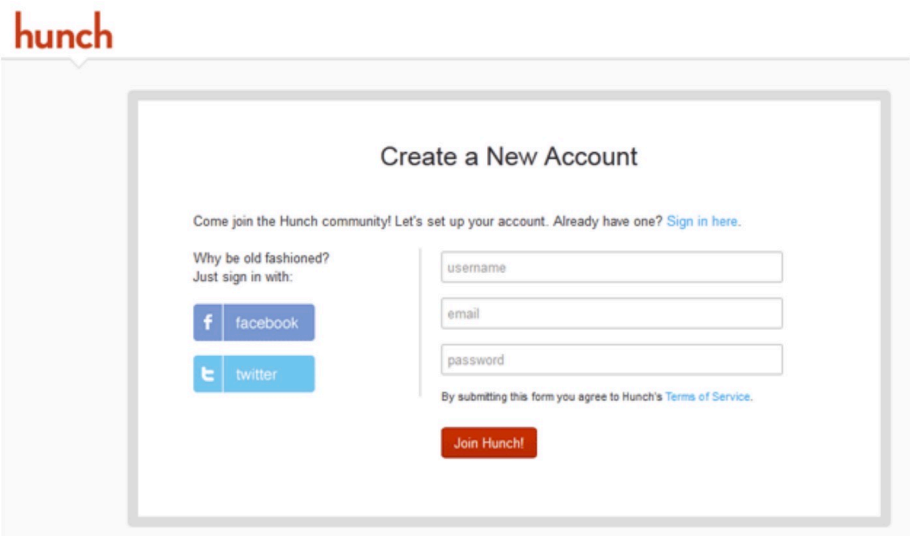
- If the things requesting the service is a person, *a user*, then the interface is a user interface





User Interfaces - Example

Text or button based interface



- Question 3: How is the service delivered?

The "service" just consist in the fact that after having signed up, the user will have an account

- Question 1: How does a user make the request?

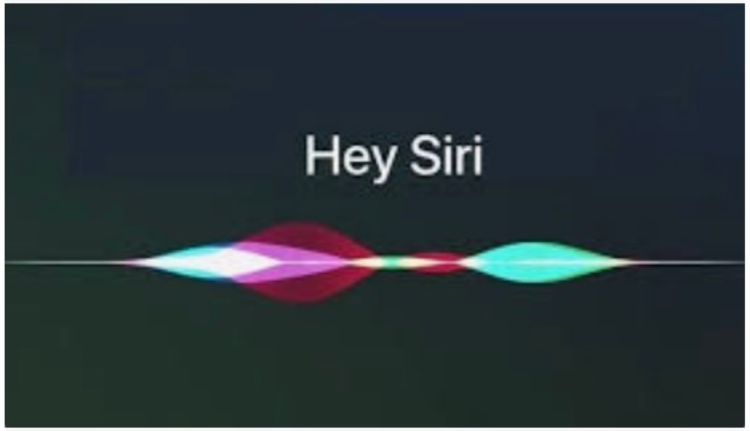
They either click one of the buttons on the left, or fill in the text fields, and click the button underneath

- Question 2: Are there any conditions that must be satisfied in order for the user to request the service?

Often, the email address must be in a valid form. (Additional validation might get performed later – e.g. by sending a confirmation email – but it's not a precondition for using the form.)

User Interfaces - Examples

Voice-based interface



- Users use voice commands to make requests. For instance, saying "Hey, Siri, order me a taxi" to a virtual assistant.

Condition:

- Authentication
- Language and accent recognition
- Interact connection ect.

Delivered:

Query Processing: Siri processes the user's voice command, interprets the intent, and retrieves relevant information or performs the requested action.

Feedback and Responses: Siri provides feedback to the user through voice responses, answering questions, providing information, or executing actions. etc ect!

User Interface - Examples

Gesture-based Interface

Condition for Making the Request:

Users must first activate the touch screen device by unlocking it or waking it from sleep mode.

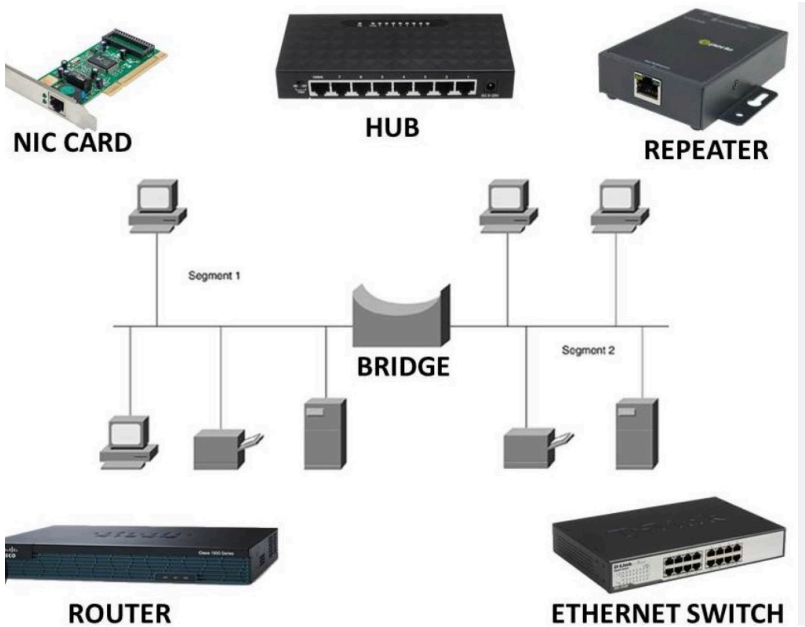
Some applications or features may require permission to access certain gestures or touch inputs for security or privacy reasons.

Delivered:
The touch screen device detects and interprets gestures made by the user, such as tapping, swiping, pinching, or rotating.

Interface navigation, action execution, feedback and response etc ect!

Hardware Interfaces

- If both of the thing involved are items of hardware, then the service will be specified both in terms of physical connection, and perhaps what sort of signals are passed through it and what they mean.



Hardware Interfaces

- To specify the hardware interface of the parallel port in the previous slide, we'll specify
  - the physical characteristics of the port – exactly what dimensions does it have, and what dimensions and configuration must an appropriate plug have to fit into it?
  - often – what signals can be sent across each of the wires the port attaches to, and what they mean.

Interfaces in Software Engineering

- In software engineering, we'll usually be interested in situations where at least one of the things provides some sort of *service* to the other.

- The interface will be defined by
  - how and in what format those services are requested including, what must be *provided* when requesting a service, and any *conditions* that must be satisfied

- how and in what format the services are delivered including anything that will be true *after* they've been delivered

Interfaces to Program Artifacts (APIs)

APIs (Application Programming Interfaces) are commonly used to **call services** from other programs.

APIs define **a set of rules** and protocols that allow different software applications to **communicate and interact** with each other.

Interfaces to Program Artifacts (APIs)

- **API Definition**: The program providing the service exposes an API, which defines **the methods, endpoints, parameters, and data formats** that can be used to interact with the service.

- **API Request**: The calling program sends a request to the API of the target program, specifying the desired action or operation to be performed. This request typically includes details such as the HTTP method (e.g., GET, POST, PUT, DELETE), the endpoint URL, and any required parameters. E.g.,

```
GET /weather?city=New York HTTP/1.1
Host: api.weather.com
```

Interfaces to Program Artifacts (APIs)

- **Service Execution**: The target program processes the API request, performs the requested operation or retrieves the requested data, and generates a response.
- **API Response**: The target program sends a response back to the calling program, containing the result of the requested operation or data retrieval. This response is typically in a structured format such as JSON (JavaScript Object Notation) or XML (eXtensible Markup Language).

```
{ "city": "New York", "temperature": "22°C", "description": "Sunny" }
```

- **Response Handling**: The calling program receives the API response, parses the data if necessary, and handles the response accordingly. This may involve displaying the retrieved data to the user, performing further processing or analysis, or taking action based on the response.

Interfaces to Program Artifacts (APIs)

Software has interfaces to – well-defined ways of requesting some bit of software perform a service. The software program could be:

- A method
- A class
- A set of classes (Subsystems)
- An external service

Interfaces to Program Artifacts (APIs)

A method:

An interface could be defined by the method's signature and functionality. Other parts of the software can interact with this method by invoking it, passing necessary parameters, and receiving a return value (if any).

: e.g., calculateTotal() in ShoppingCart class

Interfaces to Program Artifacts (APIs)

A class:

A class encapsulates data and behavior (methods) related to a specific entity or concept within a software system. The public methods and properties of a class define its interface, allowing other parts of the software to interact with it. Other classes can instantiate objects of this class and use its methods and properties to achieve specific tasks. For example, a Car class could have methods like startEngine() and drive(), forming an interface for interacting with car objects.

Interfaces to Program Artifacts (APIs)

A Set of Classes (Subsystems):
A group of related classes collectively provides a broader interface for a specific functionality or service.

This interface can involve coordination between multiple methods and properties across different classes.

For example, in a banking application, a set of classes related to account management, transaction processing, and user authentication may collectively provide an interface for managing bank accounts.

Interfaces to Program Artifacts (APIs)

```
/**
 * The MathUtils module provides utility methods for common
mathematical operations.  MathUtils presents a Module
 */
public class MathUtils {

    /**
     * Returns the sum of two integers.
     *
     * @param a The first integer.
     * @param b The second integer.
     * @return The sum of the two integers.
     */
    public static int add(int a, int b) {
        return a + b;
    }

    /**
     * Returns the difference between two integers.
     *
     * @param a The first integer.
     * @param b The second integer.
     * @return The difference between the two integers.
     */
    public static int subtract(int a, int b) {
        return a - b;
    }

    /**
     * Returns the product of two integers.
     *
     * @param a The first integer.
     * @param b The second integer.
     * @return The product of the two integers.
     */
    public static int multiply(int a, int b) {
        return a * b;
    }

    /**
     * Returns the result of dividing one integer by another.
     *
     * @param dividend The dividend.
     * @param divisor The divisor (must be non-zero).
     * @return The result of the division.
     * @throws ArithmeticException if the divisor is zero.
     */
    public static double divide(int dividend, int divisor) {
        if (divisor == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return (double) dividend / divisor;
    }
}
```

Interfaces to Program Artifacts (APIs)

An External Service:
External services, such as web APIs, databases, or third-party libraries, provide interfaces for interacting with their functionality.

These services expose APIs that define how other software systems can communicate with them.

Client applications can interact with these external services by making API calls, passing parameters, and receiving responses. For example, a weather service may provide a web API with methods like getWeatherForecast()

Interfaces to Program Artifacts (APIs)

Google maps

OpenAI

Google Maps Platform APIs by Platform

Not sure which API you need? Try the API picker.

Android

Maps SDK for Android. Maps for your native Android app.

Places SDK for Android. Connect your users with information about millions of pla

iOS

Maps SDK for iOS. Maps for your native iOS app.

Places SDK for iOS. Connect your users with information about millions of places.

Web APIs

Maps Embed API. Add a Google Map to your site without code or quota limits.

Maps JavaScript API. Customize maps with your own content and imagery.

OpenAI API

We're releasing an API for accessing new AI models developed by OpenAI.

APIs - Notes

- We can think of the API for a function (or other procedural unit) as constituting a contract between the developer of the function, and the client code using it.

- The “following thing” – the behaviour of the function – will usually be to return some sort of value, or to cause some sort of “side effect”.

- Writing to file? **writeToFile(data, filePath)**
- Send an email?
- Modifying a global variable?
- Updating a database?
- ….more

- ! Always on cautious when using APIs without returning values

APIs – Specification and Implementation

- The API documentation does not normally say how the function is to be implemented – just what its return value and effects are.

- This means that if the library developer decides to reimplement the function in another way (for instance, to improve efficiency), they can, without changing the API.

APIs – Specification and Implementation

- In fact, you can have multiple implementations of the same API by different developers.
- Example:
- Oracle corporation provides an implementation of the Java standard libraries (as well as of the Java compiler, javac, and the Java Virtual Machine or JVM).
- But there are other implementations – for instance, OpenJDK, an open-source version of the standard libraries.
- These adhere to exactly the same specifications as the Oracle versions.

APIs – Specification and Implementation

- The POSIX standard specifies an API for Unix-like systems, and has been implemented multiple times in different ways by different operating systems.

(In fact, even Windows, at various times, has met the POSIX standards.)

Developing Systems with Interfaces Top-down vs Bottom-up

- When developing systems there are basically two ways of doing it:

- **Top-down**, also called “stepwise refinement”: Try and break the desired system down into smaller pieces, until we have something we can handle
- **Bottom-up**: Try and identify individual bits of functionality we can identify, and try to assemble them into a system.

- Historically, top-down was once more common. (Well – in practice, projects used a mix of the two.) But in OO systems, something more like bottom-up is more common.

Summary

- How are interfaces defined? say how services are requested and how are they delivered

- Types of interfaces:
  - human to computer (user interface)
  - hardware
  - software

Application Programmer Interfaces

System design with interfaces: Top down or Bottom up

Recommended Reading

Liskov, Barbara, and John Guttag. *Program development in JAVA: abstraction, specification, and object-oriented design*. Pearson Education, 2000.

David, Jérôme, et al. "The alignment API 4.0." *Semantic web* 2.1 (2011): 3-10.