University of Hertfordshire

7COM1025
Programming for Software Engineers
Lecture 1

Dr. Renato C. de Amorim

THE ONE HUNDRED UNDER FIFTY

Module Leader

Dr. Renato C. de Amorim
Senior Lecturer in Computer Science
LB206
x4345
r.amorim@herts.ac.uk
http://homepages.herts.ac.uk/~comqra/

Classes and Assessment

Activity	Day	Time	Room
Lecture	Tuesday	1600 - 1700	D449
Practical	Thursday	1300 - 1500	LB252
Tutorial	Friday	1400 - 1600	E250

Assignment: 40%

Exam: 60%

APPLICATION PROGRAMMING INTERFACES (API)

- Provides an abstraction for a problem.
- Specifies how clients should interact with software components that implement a solution to that problem.

- Define re-usable building blocks that allow modular pieces of functionality to be incorporated into end-user applications.

- A well-defined interface that provides a specific service to other pieces of software.

- API development provides a logical interface to the functionality of a component while hiding implementation details

APPLICATION PROGRAMMING INTERFACE

Problem: To calculate the power of any valid base and exponent.

```
#include <cmath>
double pow (double base, double exponent);
```

What is a “valid base and exponent”?

Programming by contract

- The general idea is that you don't need to be responsible for every single detail in a (potentially) massive project.

APIS IN C++

An API is a description of how to interact with a component.

With C++ we will normally have the following elements:
1) Headers (.h)
- Files defining the interface and allowing the client code to be compiled against that interface.
- If open source, it will also include .cpp files
2) Libraries
- One or more static (.lib/.a)or dynamic library (.dll/.so) files that provide an implementation for the API
3) Documentation
- Documentation describing how to use the API, often including automatically generated documentation.

API DESIGN

Interfaces are the most important code a developer produces
- Problems with interfaces tend to be more costly to fix than those in implementation code.

Key factors in API design.
- An API is an interface designed for developers, very much like a GUI is an interface designed for end users.
- Multiple applications can share the same API
  (so, errors in the API can affect all applications that depend on the API functionality)
- You must strive for backwards compatibility. Or else, you clients code may fail completely (or worst! It could compile and behave unexpectedly).

Key factors in API design.

- Due to the backward compatibility requirement it is critical to have a change control process in place

- APIs tend to live for a long time.

- There is a strong need for (updated!) documentation, particularly if the API is not open-source.

- There is a strong need to automated testing

WHY SHOULD YOU USE APIS?

There are 2 perspectives:

a) Why should you design and write your own APIs.

b) Why should you use APIs from other providers.

- More Robust code. By hiding the implementation you get to be able to change it without having clients in panic.
- Increases longevity. Systems exposing all code tend to develop into spaghetti code. They become fragile, rigid, immobile.
- Promotes modularization
- Reduces code duplication. You centralise the logical behaviour.
- Removes hardcoded assumptions. APIs can be used to provide access to information without replicating constants.
- Easier to change the implementation
- Easier to optimise

CODE REUSE

Code duplication is one of the cardinal sins in software engineering.

Nowadays companies don't need to produce all the code themselves. There are libraries for common tasks.

Developers don't need to understand every single detail of every software component – leading to faster development!

However, you often have to design a much more general API than you originally intended

PARALLEL DEVELOPMENT

Even if writing code inhouse, your colleagues will need to write code that uses your code.
eg. If you are working on a string encryption algorithm that another develop wants to use, you could:
a) Ask the other developer to wait for you to finish.
b) You both could agree on an appropriate API. Your colleague will be able to work immediately.

In the latter case you could even provide a simple implementation of the function (eg. Returning the original non-encrypted string)
so that your colleague can compile his program.

WHEN SHOULD YOU AVOID APIS

Designing and implementing an API usually requires more work than writing normal application code
(since its a general but robust and stable interface)

License restrictions

Functionality mismatch

Lack of source code

Lack of documentation

API EXAMPLES

APIs are everywhere, chances are you have used at least one.

Operating System (OS) APIs.
    Every OS must provide a set of standard APIs to allow programs to access OS-level services.
Language API.
    Languages normally provide a standard API. Eg. Standard Template Library.
Image APIs
    Normally developers don't write their own image reading and writing API.
3D graphics API
    DirectX and OpenGL
Graphical UserInterface API
    Qt, WxWidgets, etc.

API VS SDK

Software Development Kit is a platform specific package giving access to one of more APIs.

SDKs may include resources such as example source code, supporting tools etc.

Example:
Apple publishes various IPhone APIs (UIKit, WebKit, etc).
They also produce an Iphone SDK which contains the frameworks (headers and libraries) implementing various Iphone APIs

FILE FORMATS AND NETWORK PROTOCOLS

There are several other forms of communication “contracts”.

The file format is a good example

You can save in-memory data of an image using a known layout (say .jpg)

If saved correctly, many programs can be used to load the image.

Network protocols follow a similar strategy. Communication happens if the protocol is followed.

Both of these are conceptually similar to an API as they define the standard interface for data to be exchanged.

EXERCISE

In groups, discuss the characteristics you would expect an API to have.