Delivering Software Components (SC)
System Development

• Every system $S$ consists of one or more software components $\{c_1, \ldots, c_n\}$

• A Software Component $C$ encompasses:
  • Set of related functions $\{f_1, \ldots, f_n\}$
  • Well defined interface $I$

  ➔ Each Interface $I$ consists of a set of parameters $\{p_1 \ldots p_n\}$
Components Development

- To develop/support a component $C$ we need to satisfy its related functions $\{f_1, \ldots, f_n\}$ and interface $I$, to do so we need to:
  - Understand the business requirements document (Assignment Spec)
  - Deliver Technical specifications document (how the business requirements will be met)
Characteristics of SC

- Encapsulation
- Interface
- Replaceable
- Reusable
Software Component Example
Why SC are good?

- **For component provider**
  - Able to change the implementation of the component as long as the **interface** is still satisfied
  - New requirements can be delivered as new components, without having to change the existing components

- **For application builder**
  - Don’t need to recompile/redeploy anything(with the same **interface** and functionality)
  - No need to understand the inner working, but only the **interface** of the component
Components are like black boxes

- The programmer knows:
  - how the outside looks like
  - what the component can provide

- The programmer does NOT know:
  - how it works internally
Weather Plug-in
Developing Software Components

- DOS Batch file
- Linux/Unix Shell Script
- C# DLL file
- JAR File
- Executable file (.exe)
- Web service (e.g. REST)
Java component

- Plug-in in Java application
- Back-end service in web app
- Part of workflow

Java Component (.jar)
How to Generate .jar File

- Export from IDE (e.g. Eclipse)
- Use command line:
  - `jar cf jar-file input-file(s)`
- Use popular build tools:
  - Maven
  - Ant
  - Buildr
  - ... ...
C# and .NET

- Component technology for Windows systems
- DLL = basic component that can be executed by a Windows application
- Many utilities for creating and managing components
DLL File Example

- Create C# Classes
- Generate DLL file
- Generate EXE file
- Run the EXE file
REST APIs
Web service
WSDL/REST
Representational State Transfer (REST) is an “architectural style” defined by Roy Fielding.

- The concepts of REST are independent of the Web, but the Web is well suited for the REST.

**REST includes:**
- Resources (things) with
- Unique ids (URLs) that can come in many representations (text, html, json, xml)
- Verbs (GET, PUT, POST, DELETE)
Most common operators

GET
  Retrieve a representational of resource (without changing it)

PUT
  Create or replace a resource by supplying representational to it

DELETE
  Ensure that a given resource is no longer exist

POST
  Augment a resource with additional representational
Representational

As an external user you cannot manipulate a resource directly. Instead you manipulate representation of that resource

- Many people can “get” representation of single resource
- Same resource can be manipulated in different ways
Unique IDs for resources (URIs)
Verbs (HTTP operators)
Multiple representations (Media Types)
The Web is an example of a REST system!

All of those Web services that you have been using all these many years - book ordering services, search services, online dictionary services, etc - are REST-based Web services.
Restlet is a Java framework for implementing REST architecture.

- Operators, Resources, Representations are all class entities in Restlet
- Highly pluggable implementation to support extensibility and interfaces to other web technologies

Atom, GWT, JSON, XML, SSL, Jetty, etc.
Calling several APIs

- Can be done programmatically
- Can use Business Process Management Framework
  - BPEL/BPMN
- Can use a workflow language
  - TAVERNA
Handling **input files** and **output files** REST-fully

Current Problem
Solution #01
Enable File upload via REST commands

Input:

- **Upload** entire files to the web service.
  - E.G. upload pictures to Facebook, or files to Dropbox
  - Granted there are UIs to facilitate this, and for this first deliverable there is no user interface.

- Achieved through standard **HTTP request verbs**
  - E.G. **POST**, **PUT**
  - Make clear API(s) using HTTP for file uploads.
Solution #01
Enable File upload via REST commands

Output:

- Teams have more flexibility in module output.

**OPTION #01**
- Return output as [JSON](#) response.
- This is a very common return format for API calls in the real world.

**OPTION #02**
- Return [download links](#) to output files.
- Links would be returned as part of a JSON response (as opposed to all the information being contained in a JSON response as with the first option).

**SUGGESTION** Examine the responses from API calls from available services like Twitter
Solution #02
Multipart / form-data

- Sending multipart / form-data message
- Very complicated!
Solution for SENG Workshops

- Both solutions are applicable to all three projects
  - Creative, alternative and effective software designs are always impressive (to us).

- For deliverable 2 there will be points allocated to the adoption rate of your modules.
  - Practical indicator of design quality => How many people use it!

- Document Well
  - Your solution can’t be used if no one knows how to use it properly!

- ASK QUESTIONS!!!
  - Filling in gaps in your knowledge and information provided: That’s part of the real process out there
  - Asking effective questions early is paramount
Common Mistakes

- Component run accurately, but Log file incomplete or doesn’t exist.
- No clear instructions on how to execute the component.
- The group said the version on their website is the wrong version, they will upload the correct version as soon as possible.
- Clear execution instructions, but lack of unit testing, errors generated when running the component.
- Output doesn’t change when changing input parameters (i.e. hardcoded the parameters).
- Who is doing what in the group, clarify from the beginning don’t leave it to late.