

# ECP Implementation Workflow Walkthrough

## version 1.1

### Introduction

The aim of this document is to walk you through part of the implementation workflow for the ECP example from the point of view of the OO analyst/designer. This example should be read in conjunction with the book “UML and the Unified Process” [Arlow].

### References

[Arlow] - UML and the Unified Process, Jim Arlow and Ila Neustadt, Addison Wesley, 2002, ISBN0201770601

[Alur] - Core J2EE Patterns, Deepak Alur et al, Sun Microsystems Press, 2001, ISBN0130648841

[ECPRequirements] - ECP Requirements Walkthrough, Clear View Training, 2002

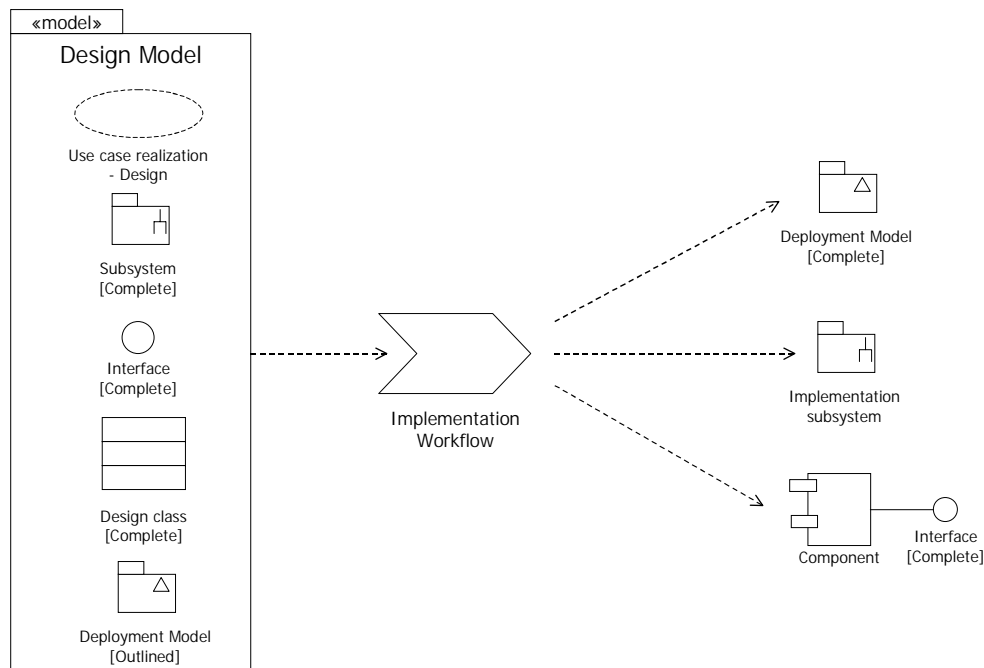
[ECPAnalysis] - ECP Analysis Walkthrough, Clear View Training, 2002

[ECPDesign] - ECP Design Walkthrough, Clear View Training, 2002

### Inputs and outputs

Here are the inputs and outputs to the UP Design Workflow for the ECP:

Figure 1



There is no explicit implementation model - all UML implementation artefacts are considered to be part of the design model.

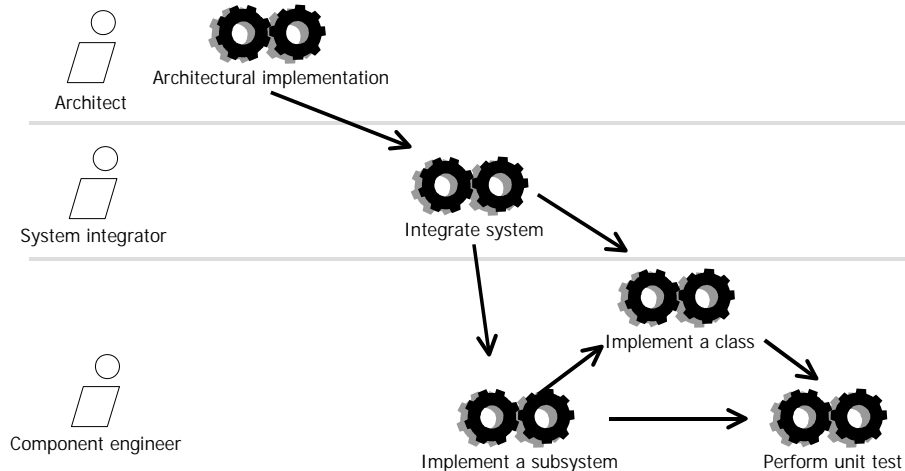
### *Scope*

We will be implementing a vertical slice through the ECP from the user interface down to the database. This has already been described in [ECPDesign].

### *The Implementation Workflow*

The Unified Process (UP) Implementation Workflow is shown below:

Figure 2



This workflow is made up of a number of activities:

**Table 1**

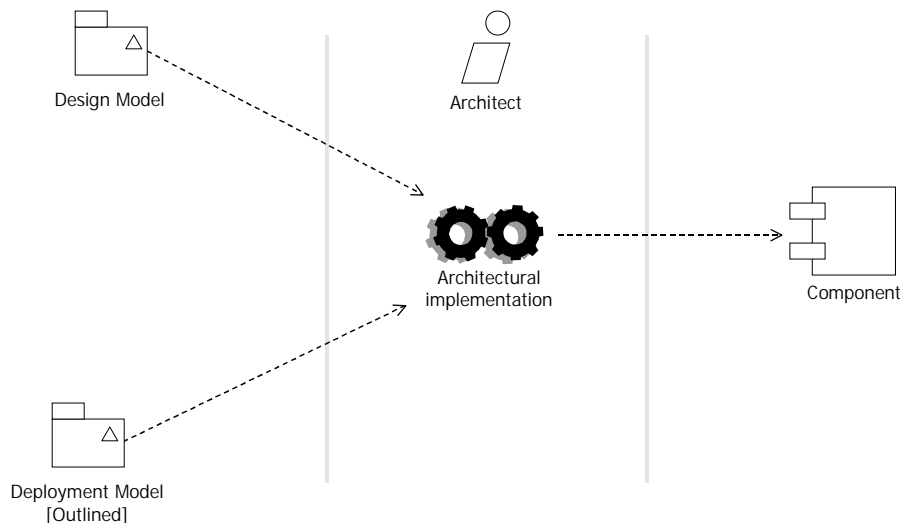
Activity	Purpose
Architectural implementation	Identifying architecturally significant components and mapping components onto nodes.
Integrate system	Create an integration build plan describing the builds in each iteration. Integrate each build prior to performing integration tests.
Implement a class	Specify a file (or other) component for each class and then write or generate and write the code for the class.
Implement a subsystem	Ensure that the subsystem fulfils its role in each build as defined in the integration build plan. Ensure that the subsystem fulfils all its requirements.
Perform unit test	Perform specification or “black-box” testing to verify the external behaviour of components. Perform structure or “white-box” testing to verify the implementation of a component.

Only two of these activities involve the analyst/designer role: Architectural implementation and Implement a class. However, for completeness, we will discuss all of the activities except Perform unit test, which is a specialized subject in its own right.

### *Activity: Architectural Implementation*

In architectural implementation, we take the design classes we have created and map these onto components. These components may be deployed onto nodes. Here are the inputs and outputs for the activity:

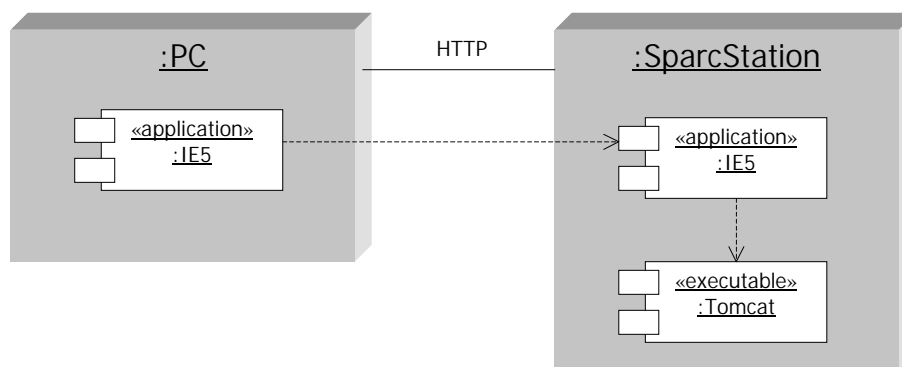
Figure 3



A component is essentially just the physical packaging mechanism of a design class. Two examples of components in Java technology are Java class files (.class files), and Java ARchives (.jar files).

Here is a deployment diagram for the system that we created in design. This shows the htdocs directory on the server that contains all our HTML, JSP, servlet and tag library components for the web application.

Figure 4



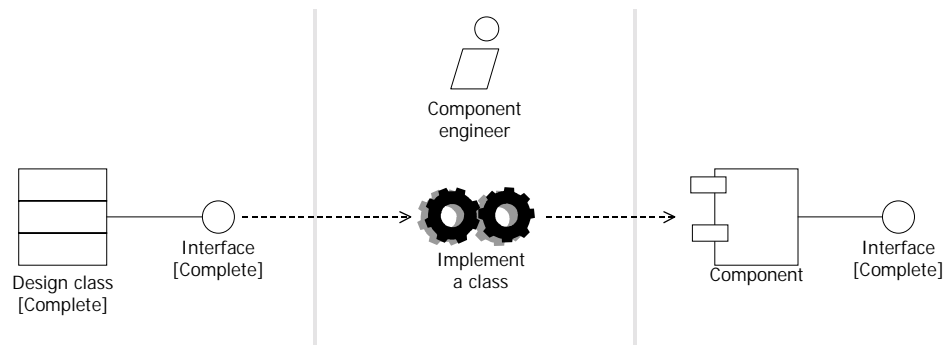
*Activity: Integrate system*

This activity is about planning and integrating builds of the system. It is out of scope for our walkthrough, as we are concentrating on the OO analysis and design aspects of the UP.

*Activity: Implement a class*

In this activity we have to assign our design classes to physical components that hold them. Here are the inputs and outputs for the activity:

Figure 5



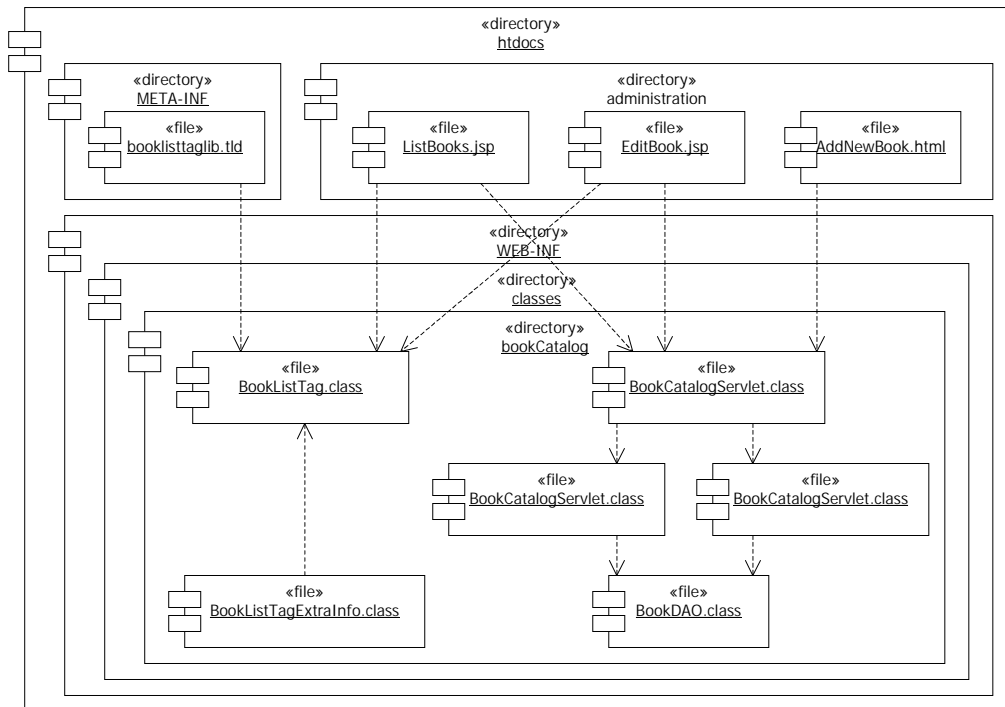
In Java, each class should normally reside in a file with the same name as the class. Java source code is kept in files with the extension `.java`, and when this is compiled to Java bytecode, the extension becomes `.class`.

With web applications, the situation may be more complex than this, as several Java class (and other) files may be packaged into a Web Application aRchive (WAR) file. This is a compressed file (ideal for downloading via the Internet) that is a special kind of Java ARchive (JAR) file.

Because of the way our Tomcat server has been set up, we are *not* using WAR files, but are deploying the web application by copying the various files into the appropriate server directories. This means that the activity Implement a class is more or less trivial - all Java classes go into `.java` files, which are then compiled to `.class` files.

The component diagram below shows how our web application components must be deployed on the server:

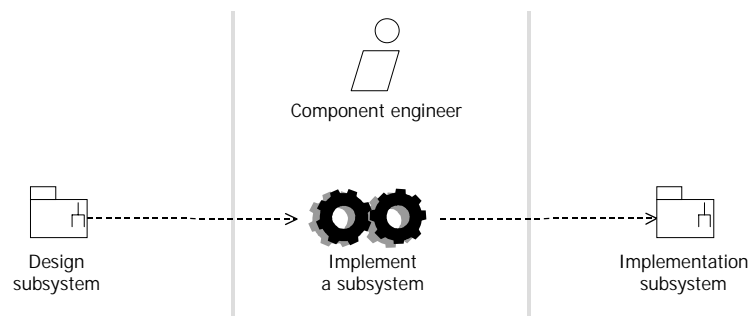
Figure 6



### Activity: Implement a subsystem

For the ECP, the inputs and outputs for this UP activity are:

Figure 7



In our ECP example, all we have to do here is check that the components in our component model implement all of the classes in the design model. According to the UML specification, there is always one-to-one mapping between design and implementation subsystems, so we don't need to produce a diagram for the implementation subsystems.

### Summary

This concludes our brief walkthrough of the implementation process. The complete design model may be found on our website at [www.clearviewtraining.com/example1/ecpexample.htm](http://www.clearviewtraining.com/example1/ecpexample.htm).