Module dependencies

Thinking of each top-level directory as a module, I thought it would be natural to have each module contain its own metadata. That way the module could just declare the classpath and configuration files it contributes, along with a declaration of which other modules it needs.

I gave each module its own manifest file. For example, here is the manifest file for the `StudioClient` module:

```
Required-Components: Common StudioCommon
Class-Path: bin/classes/ lib/StudioClient.jar
Spring-Config: config/beans.xml config/screens.xml config/forms.xml
       config/navigation.xml
Purpose: Selling station. Workflow. User Interface. Load images. Burn DVDs.
```

This format clearly derives from *.jar* file manifests. I found it useful to align the mental function "manifest file" with a familiar format.

Notice that this module uses four separate bean files. Separating the bean definitions by function was an added bonus. It reduced churn and contention on the main configuration files, and it provided a nice separation of concerns.

Our team strongly favored automatic documentation, so we built several reporting steps into the build process. With all the module dependencies explicitly written in the manifest files, it was trivial to add a reporting step to our automated build. Just a bit of text parsing and a quick feed to Graphviz generated the dependency diagram in Figure 4-2.

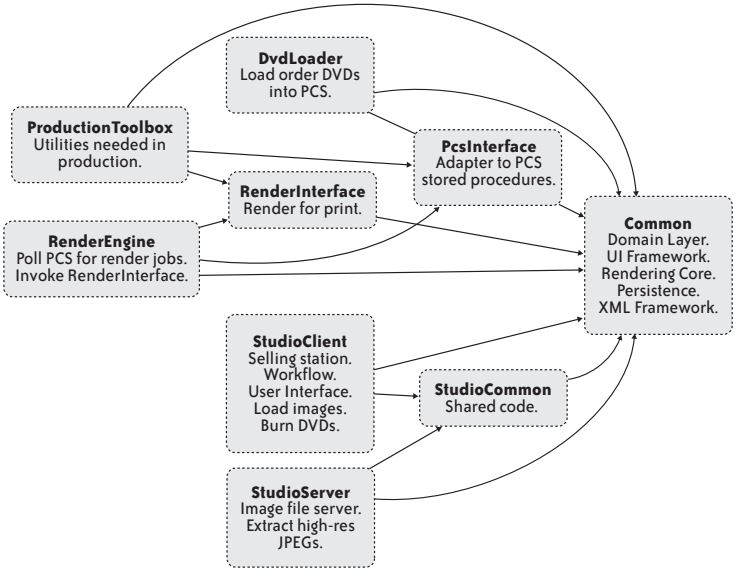

FIGURE 4-2. Modules and dependencies