Glamour packages - User study

Presentation

The goal of this study is to assess packages forming a application. You will assess packages through their classes and class references, analyzing what they use and how they are used, both internally and externally to the application.

This study is organized in three sections which require more and more in-depth look at packages and their classes. Each section implies that you play a different role when assessing the application, first as newcomer and potential client who wants to use the application, second as an architect who needs to assess the organization, third as a developer who performs maintenance. There are 11 questions in this study, each question relating to a task.

You will perform the study on Glamour packages. Glamour is an engine for scripting browsers for any kind of models. If you are unfamiliar with Glamour, do not hesitate to test it before the study to get a basic understanding of Glamour capabilities. Information on usage and samples are available on: http://www.moosetechnology.org/tools/glamour

Tool used: Package blueprint browser

Instructions

- Use only the tool indicated for the study. Do not use another browser/tool.
- Browse the documentation before performing the study:
  - **PackageBlueprintsPrinciples.pdf** explains the basics of package blueprints
  - **packageblueprints.mov** shows interactions with the package blueprint browser
  - a draft version of the journal paper describing package blueprints is provided
- Process questions in the given order (do not read questions in advance!)
- Please provide only accurate answers like the name of a class or a package, the association between two classes, the method with references.
- **Time yourself** for each question.
- Do not spend more than **20 minutes** on a question. If you reach this limit, write it down, stop the task, and proceed to the next question.
- You also have a time limit of **1h30** to answer the 11 questions so take care of your time.

A. Application assessment

As a potential client, you are assessing the package dependencies of the application. You want an idea about the size of the application and the kind of dependencies needed, especially if it involves new dependencies to be loaded with the application.

1) How big is the application?
Time taken: 1min
(a) In number of packages: 10

(b) In number of classes (one of the following ranges):
[ ] <100; [ ] 100-200; [ ] 200-300; [x] > 300

2) What are the most important packages?
Time taken: 1”20
(a) In terms of outgoing dependencies
Glamour-Tests
(b) In terms of incoming dependencies
Glamour-Core

(c) Overall, considering both outgoing and incoming dependencies
Glamour-Core

3) Focus on package Glamour-Morphic:
Time taken: 8min
(a) list all package dependencies which are external to Glamour.
Morphic, Polymorph Widgets, Mondrian, Morphic-MorphTreeWidget, Baloon, Shout, Magritte, Graphics, Collections-Strings, Collections-Sequenceable, DeprecatedPreferences, Collections-Arrayed, Freetype, PolymorphToolsDif, Collections-Unordered
(b) in this list, please signal any external package which is not part of Pharo base (i.e., package must be loaded with Glamour).
- I am not familiar enough with pharo

(c) are there other unexpected/unwanted package dependencies?
- no

B. Application architecture assessment
As an architect, you now want to check the organization of your packages. You want your packages to have a good rationale for existence in the application. You want some parts of the application to be modular.

4) Please characterize **each** Glamour package as either:
Time taken:
- a provider package for external clients (package with which external clients interact)
Core, Browsers, Presentations, Helpers, Tools. (All these have external classes that depend on them(

- an internal package (package which should not be accessed by external clients)
all the remaining packages

5) Are some Glamour packages optional/modular (package can be unloaded without impacting application core)?

[argh! I did a search for all the packages that have names starting with Glamour-Core and it tool a loooong time!]

Scripting, and Helpers seem not to depend on Core.

Time taken: 8min

6) What are the important classes (consider incoming, outgoing, inheritance dependencies) in Glamour-Core? If possible, explain their roles.
Time taken: 8min

Using: (GLMTransmission, GLMBrowser, GLMPresentation) are using many other classes from the Core as well as from other packages too.

Used: (GLMPresentation, GLMPane, GLMBrowser) are strongly used by other classes.

7) Are there direct cyclic dependencies from Glamour-Core to another package?
No.
Time taken: 2min

C. Detailed assessment

As a developer, you want a detailed comprehension and assessment of dependencies between classes and packages and optionally to refactor such dependencies, assessing impact of change.

First give a precise answer then provide your explanation.

8) What are the most cohesive packages of the application?
Core, Browsers. *Time taken:*
1min

9) There is a dependency to DeprecatedPreferences in Glamour-Morphic. Can you detect the faulty class? Explain the dependency: do you see an easy way to solve it?
-- The GLMMorphicRenderer is the faulty class. Do not use the Preferences class, but store the preference locally.

Time taken: 1’50”

10) Can you explain the organization of Glamour-Morphic and its relationship with other packages?

Looks like the Glamour-Morphic package provides classes that bridge the Glamour concepts to the Morphic elements (e.g., PanelScroller, TreeModel, TreeMorphNodeModel) and provides a wrapper around various Morphic functionalities (GLMMorphicRenderer) which is the class from Glamour-Morphic that is using the most classes in the Morphic related packages.

Time taken:
7min

11) Multiple packages of Glamour have dependencies to external library Mondrian. List such packages. Could you extract this dependency and make it optional (you can propose a solution)? 5 packages depend on Mondrian, two of which are tests, and one which is in the Examples packages, so we don’t consider them. In the Glamour-Morphic package the only client is GLMMorphicRenderer who only has a single method that uses Mondrian: #mondrianCanvasFor:in:
Removing this dependency is a matter of removind this method.

Another dependency is in Glamour-Presentations, where GLMPresentation creates a MOViewRenderer. If this class would be removed there would be no more dependencies to Mondrian.
To make the dependency optional, one could move the GLMPresentation class together with an extension of the GLMMorphicRenderer to a different package.

Time taken: 8min

D. Personal remarks

You can provide any additional remarks about the study itself, the tasks, the tool used.

E. Personal evaluation of Package blueprints

1	2	3	4	5
Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

Question	1	2	3	4	5
Does package blueprint help you to understand dependencies between packages?					x
Would you use package blueprint when you need to understand packages?				x	
Did the outgoing view help you?					x
Did the incoming view help you?				x	
Did the inheritance view help you?		x			
Was package blueprint useful to get an impression of the most used classes in a package?					x
Was package blueprint useful to get an impression of the most referencing classes in a package?					x
Was package blueprint useful to get an impression of package cohesion?				x	