

Übungsblatt 2

Ausgabe: 29.10.2018

Abgabe: 08.11.2018 12:00 Uhr

Organisatorisches

Bitte beachten Sie, dass Abgaben, auf denen Informationen wie Gruppennummer, Gruppe für Abholung bei Mehrfachabgaben oder Matrikelnummern fehlen, mit 0 Punkten bewertet werden.

Ansonsten gelten die Hinweise von Übungsblatt 1.

Hausübung

Aufgabe 2.1 (Umsetzung von Architekturmustern)

6 P.

- (a) Nennen Sie Beispiele aus der realen Welt, in denen folgende aus der Vorlesung bekannten Architekturmuster 4 P.

- (i) Client-Server Web-Anwendungen
- (ii) Peer-to-Peer Bittorrent
- (iii) Pipe-and-Filter Unix-Filter (VL)
- (iv) Sharding ?

umgesetzt werden. Erläutern Sie, warum das Beispiel eine Instanz des Musters darstellt und welche Elemente des Beispiels den Komponenten des Entwurfsmusters entsprechen.

- (b) Ihre Firma wird beauftragt, ein Softwaresystem zum elektronischen Bezahlen in der Mensa einer großen deutschen Hochschule – ähnlich dem Zahlen per UniCard an der TU Dortmund – zu entwickeln. Bevor die Arbeit beginnen kann, muss jedoch eine Architektur für das zu entwerfende System feststehen. 2 P.

Nennen Sie zwei Architekturentwurfsmuster, die Sie für ein solches System für *ungeeignet* erachten. Begründen Sie mit stichhaltigen Argumenten, warum das Muster für diesen Anwendungsfall nicht verwendet werden sollte oder nicht benötigt wird.

Peer-to-Peer, Pipe-and-Filter

Aufgabe 2.2 (Model, View, Whatever)

8 P.

In der Vorlesung wurde das „Model-View-Controller“-Architekturmuster vorgestellt. Dieses Muster ist jedoch nicht der einzige Ansatz, um Anwendungen mit grafischen Oberflächen zu strukturieren.

Für das Framework *Windows Presentation Foundation* wurde von Microsoft das „Model-View-ViewModel“-Muster¹ erfunden; inzwischen existieren Implementierungen für weitere Umgebungen. Bereits in den Neunzigern stellte Mike Potel in einem Whitepaper das „Model-View-Presenter“-Muster² vor, welches von der Firma *Taligent* genutzt wurde.

Lesen Sie sich in die Materie ein. Beschreiben Sie in eigenen Worten alle drei Ansätze. Beantworten Sie in Ihren Antworten insbesondere folgende Fragen:

- Wie grenzt sich die Muster untereinander ab?
- Wie wird jeweils der Fluss von Daten zwischen den Komponenten realisiert?
- Wie reagiert das System jeweils auf Benutzereingaben?

Aufgabe 2.3 (Konfigurationsmanagement mit GIT)

6 P.

Für diese Aufgabe sollten Sie sich mit dem de-factor-Standard-Versionskontrollwerkzeug Git vertraut machen. Ein guter Einstiegspunkt ist die auf der Git-Webseite verfügbare Dokumentation³, insbesondere das Buch „Pro Git“ von Chacon und Straub.

(a) Führen Sie zunächst folgende Schritte aus:

2 P.

- S1 Erstellen Sie ein leeres Git-Repository und fügen Sie einen Commit hinzu, der die leere Datei `data.txt` hinzufügt.
- S2 Legen Sie einen Branch `fork` an, der von diesem Commit aus verzweigt.
- S3 Nun erstellen Sie je einen Commit im Branch `fork` und im Hauptentwicklungszweig `master`, in denen Sie den Inhalt der Datei durch `f` bzw. `m` ersetzen.
- S4 Benutzen Sie einen Merge, um den Branch `fork` in den Hauptentwicklungszweig `master` zu reintegrieren. Wie Sie den Mergekonflikt auflösen, ist Ihnen überlassen.

Illustrieren Sie Ihr Vorgehen:

¹<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>

²<http://www.wildcrest.com/Potel/Portfolio/mvp.pdf>

³<https://git-scm.com/doc>

- (i) Geben Sie in chronologischer Reihenfolge alle Git-Kommandozeilenbefehle an, die genutzt werden mussten.
 - (ii) Zeichnen Sie den Commit-Graphen⁴ nach allen Operationen. Eine digitale Reproduktion des Graphen ist auch akzeptabel.
- (b) Wiederholen Sie die Schritte S1–S3 aus Aufgabenteil (a). Dann: 2 P.
- S4 Benutzen Sie eine Rebase-Operation, um den Branch **fork** auf den Stand des Hauptentwicklungszweiges **master** zu bringen. Wie Sie den Mergekonflikt auflösen, ist Ihnen überlassen.
- S5 Benutzen Sie einen Merge, um den Branch **fork** in den Hauptentwicklungszweig **master** zu reintegrieren. Stellen Sie sicher, dass *kein* Merge-Commit angelegt wird.
- Illustrieren Sie Ihr Vorgehen:
- (i) Geben Sie in chronologischer Reihenfolge alle Git-Kommandozeilenbefehle an, die genutzt werden mussten.
 - (ii) Zeichnen Sie den Commit-Graphen⁴ nach allen Operationen. Eine digitale Reproduktion des Graphen ist auch akzeptabel.
- (c) Beide vorgestellten Varianten zur Reintegration von Branches werden in der Praxis oft genutzt; viele Projekte legen Wert, dass nur einer von den zwei Ansätzen genutzt wird. Welche Vorteile haben die Verfahren gegenüber dem jeweils anderen? 2 P.

⁴z. B. so: `git log --graph --oneline --decorate --all`