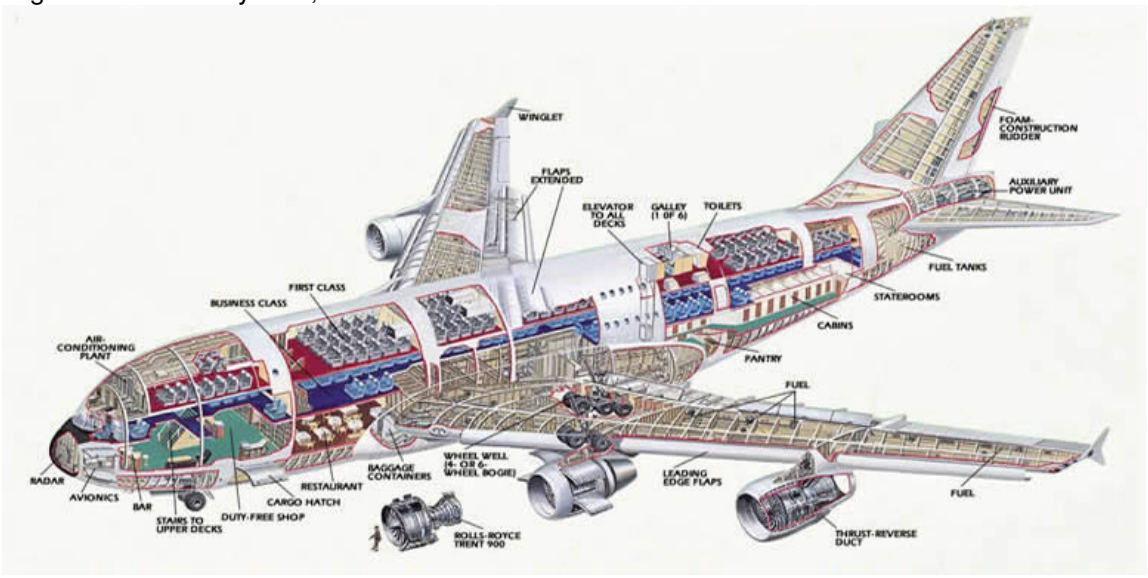
Applikationsentwicklung
Architekturübungen

Aufgabe 1: Systeme und Subsysteme
Gegeben ist das umfangreiche Softwaresystem eines modernen Passagierflugzeuges von der Steuerung und Navigation bis zum Bordunterhaltungssysstem und Kommunikation.

a) Identifizieren Sie mögliche Teilsysteme des Softwaresystems eines Passagierflugzeuges und listen Sie sie auf.
b) Zeichnen Sie die Teilsysteme als ein einfaches Blockdiagramm auf und ergänzen Sie die Beziehungen zwischen den einzelnen Systemen in Form von gerichteten Abhängigkeiten.
c) Ordnen Sie die Teilsysteme nach Hierarchie, Abhängigkeit und Abstraktion ein (Tip: Denken Sie an die Prinzipien der 'Top-Down' bzw. 'Bottom-Up'-Entwicklung)
d) Gewinnen Sie weitere Übung indem Sie die Schritte a.) bis c.) für beliebige andere (Software-) Systeme wiederholen. Beispiele: Schuladministration, Betriebssystem (OS), Fahrgastinformationssystem, etc.

e)



Aufgabe 2

a) Wählen Sie einige Anwendungen Ihres täglichen Gebrauchs aus, deren Funktion und Inhalt Sie somit gut kennen. Beispiele: Thunderbird EMail-Client, iTunes, MSN Chat etc.
b) Identifizieren Sie pro Anwendung die zentralen Teile für jede einzelne der drei logischen Schichten! Achten Sie dabei nicht nur auf Prozesse, sondern auch auf Daten, Technologien etc.
c) Wählen Sie von diesen Anwendungen jede Schicht mindestens einmal aus und versuchen Sie diese in weitere Schichten zu verfeinern.
d) Bewerten Sie für jede Verfeinerung: Macht es Sinn? Welches Potential bietet die Verfeinerung? Hat es Nachteile?

Aufgabe 4 & 5
MVC Example:
http://csis.pace.edu/~bergin/mvc/mvcgui.html

Aufgabe 6
Gegeben sind die folgenden Anforderung an eine Anwendung A:
Es wird eine einfache Notizverwaltung benötigt, welche Notizen erfassen, suchen/anzeigen und löschen kann. Es bestehen keine weiteren Anforderungen in Bezug auf Sicherheit und Rechte, so dass im Wesentlichen eine einzige Datenbanktabelle für die Speicherung der Daten ausreicht. Da aber für das Intranet ein Rich-Client und gleichzeitig für das Extranet ein Thin-Client (Web-Technologie) realisiert werden soll, und in Kürze die ganze DBMS-Plattform ausgetauscht werden wird, ist grundsätzlich eine logische 3-Schicht-Architektur verlangt.

a.) Schlagen Sie für die Anwendung 'A' eine konkrete Architektur vor! Zeigen Sie auf welche Schichten Sie verlangen und auf welche Tiers Sie diese verteilen könnten/würden!
Der Einfachheit halber (nur 1 Tabelle in der Datenbank!) empfiehlt sich hier das Pattern „Table Module“.

b.) Versuchen Sie für die Implementation der Businesslogik die verschiedenen Domain Logic Patterns zu adaptieren. Skizzieren Sie!

Transaction Script Pattern:
Das Transaction Pattern betrachtet die Anwendung von der Funktionsseite, d.h. man überlegt sich, welche Funktionen eine bestimmte Anwendung ausführen können muss.

Entweder jede Funktion (=Transaktion) wird in eine separate Komponente implementiert (Beispiel: Siehe Abbildung) oder aber mehrere Transaktionen werden in einer Komponente gruppiert (Beispiel: Komponente „NoteProcessing“ enthält die Methoden CreateNote, UpdateNote, DeleteNote, usw.).
Dieses Pattern ist nur für kleine Systeme empfehlenswert, wenn beispielsweise absehbar ist, dass nie mehr als eine einzige Datenbanktabelle verwendet wird. Die Kopplung zwischen Datenbank, Business Logik und UI ist sehr gross, was sich bei Änderungen negativ auswirken wird.

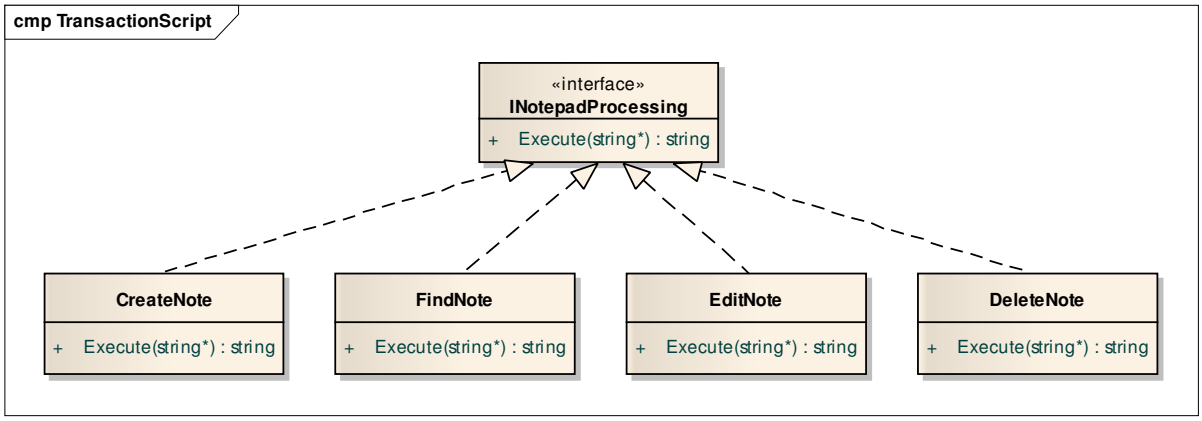
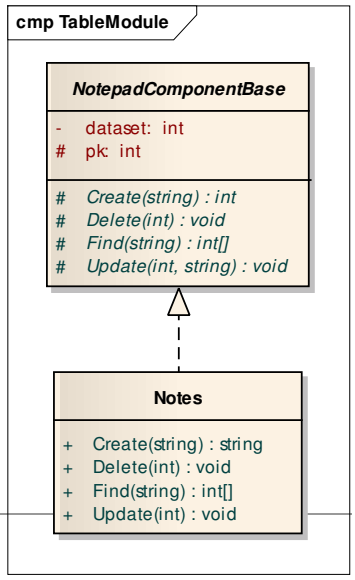


Table Module Pattern
Das Table Module Pattern bezieht sich auf ein Set von Datensätzen einer Datenbank-Tabelle. Pro Tabelle besteht eine (Singleton-) Klasse, welche CRUD-Operationen implementiert. Im Gegensatz zum Domain Model Pattern gibt es nicht für jeden Datensatz ein separates Objekt. Dementsprechend ist es auch nicht nötig, im Objekt einen Primary Key abzuspeichern. Operationen, welche bestimmte Datensätze manipulieren möchten, müssen deshalb den Primary Key bei jeder Ausführung entgegennehmen.

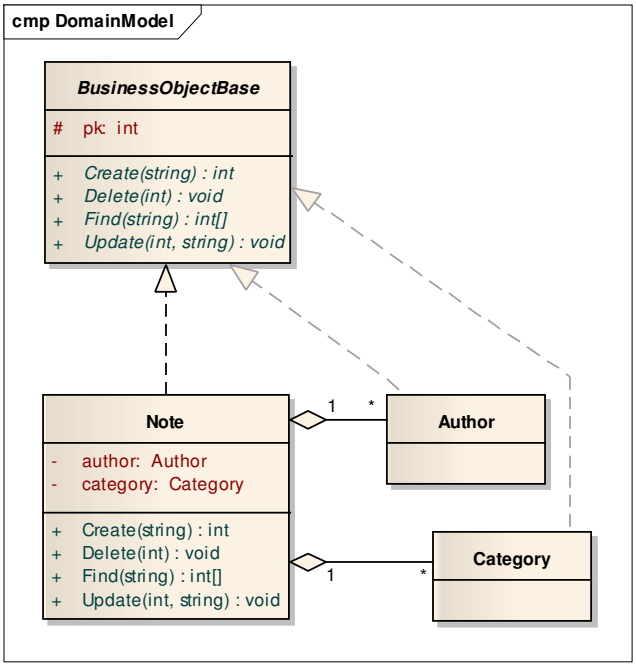
Vorteil dieses Patterns ist vor allem die gute Performance und den geringen Ressourcen-Overhead (CPU/Memory). Es ist ein guter Kompromiss zwischen Transaction Script Pattern und Domain Model Pattern.



Domain Model Pattern

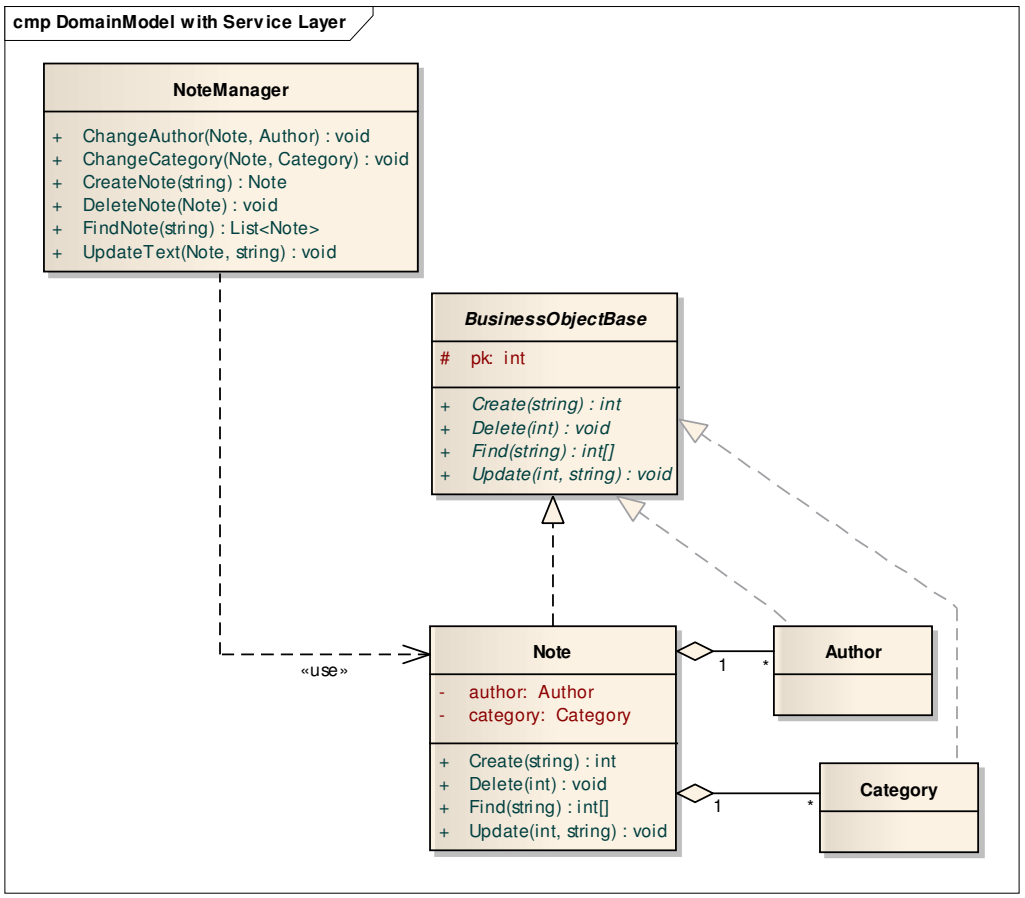
Im Domain Model repräsentiert für jeden Datensatz der Datenbanktabellen ein bestimmtes Domain Object. Diese Objects werden in der Laufzeit der Anwendung erstellt und enthalten definierte Attribute und Daten des jeweiligen Datensatzes. Ein spezieller Data Access Layer (kurz DAL; umgesetzt von Produkten wie JPA, NHibernate oder TopLink) regelt den Datenaustausch zwischen Datensätzen der Datenbank und den Domain Objects.

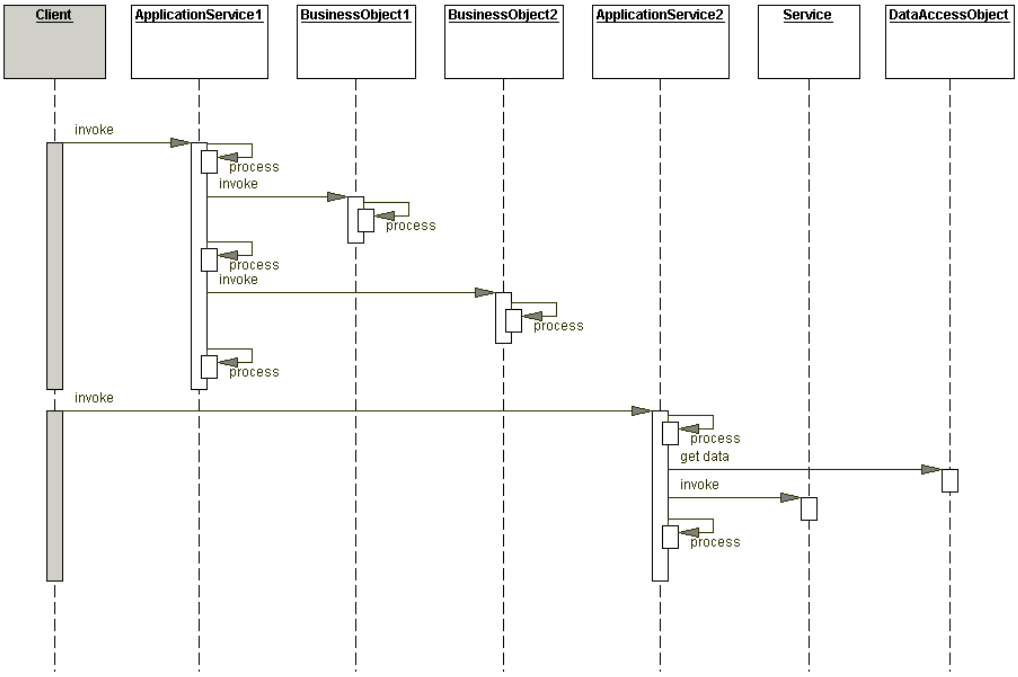
Der grosse Vorteil besteht darin, dass eine komplexe Architektur besser verständlich wird, da jedes reelle Objekt als Domain Object abgebildet wird. Für das Handling mit der Datenschicht ist kein T-SQL Knowhow nötig. Der eingesetzte OR-Mapper dient zudem der physischen Entkopplung der Datenschicht. In grossen Enterprise Architekturen wird die Skalierbarkeit massiv verbessert.



Service Layer Pattern

Sinnvollerweise werden die Objektmethoden (CRUD Funktionen) nicht direkt an die Präsentationsschicht weitergegeben. Zwischen dem sog. Business Object Layer und der Präsentationsschicht wird oftmals ein Business Logic Layer (BLL) geschalten. Im illustrierten Beispiel könnte beispielsweise eine Klasse NoteManager hinzugefügt werden. Diese würde dann beispielsweise auch Methoden wie „ChangeCategory“ anbieten. Der BLL ist zuständig für die Implementierung der Geschäftsprozesse. Innerhalb dieser Geschäftsprozesse werden dann schliesslich die CRUD Operationen der Domain Objects genutzt, um Manipulationen an den Daten vorzunehmen.





Mehr Information zu Domain Logic Patterns:
http://www.techrepublic.com/article/using-common-domain-logic-patterns-in-your-net-applications/5107664

Mehr Informationen zu Enterprise Architecture Patterns:
http://martinfowler.com/eaaCatalog

c.) Welches Domain Logic Pattern erscheint Ihnen am geeignetsten? Aus welchen Gründen?
Bei „Transaction Script“ wäre die Kopplung zwischen GUI Layer und Datenquelle zu gross. Ein Domain Model würde den Rahmen sprengen, insbesondere wenn dann noch ein Service Layer eingesetzt wird. Da wir annehmen, dass die Notizen nicht viel Metainformationen haben werden, könnten wir den Griff zum Table Module Pattern wagen. Siehe auch Ausführungen unter a).

d.) Vergleichen und diskutieren Sie Ihre Lösungen untereinander!
Siehe Ausführungen unter a).

Gegeben sind die folgenden Anforderungen an eine Anwendung B:
Es wird eine Applikation zur Raumverwaltung und -reservation benötigt. Darin können Räume verwaltet und für Veranstaltungen reserviert werden. Es müssen differenzierte Rechte für verschiedene Personen (-gruppen) möglich sein. Die Daten müssen in einem RDBMS gespeichert werden. Bezüglich der Client-Technologie sind keine expliziten Anforderungen vorhanden, es muss aber eine Benutzung auch über das Intranet möglich ein.

e.) Wiederholen Sie die Teilaufgaben a.) bis d.) für die Anwendung 'B'. Zu welchem Ergebnis kommen Sie?
Das Domain Model dürfte hier zwangsläuft die beste Wahl sein. Es erlaubt die höchste Flexibilität, welche in diesem Fall (hohe Komplexität der Datenstruktur) auch nötig ist. Oberstes Ziel bei der Architekturwahl ist, dass bei einem kleinen Change an einem Business Process nicht gleich alle/grosse Teile der darunterliegenden Business Logic angepasst werden müssen.
Auf jeden Fall sollte hier auch ein Service Layer zum Einsatz kommen. Die gesamte Applikation könnte hinter einer „Facade“ verborgen werden.

f.) Vergleichen Sie Ihre Ergebnisse und diskutieren Sie!

Aufgabe 7: DTO

Weiteres Beispiel: http://msdn.microsoft.com/en-us/library/ff649585.aspx