TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Institut für Managementwissenschaften

Integration von Business & MGT mit IT

Enterprise Information Systems

Sommersemester 2019

09. Mai 2019

Walter S.A. Schwaiger

Christian Fischer-Pauzenberger

Fachbereich - Finanzwirtschaft und Controlling

Institut für Managementwissenschaften,

Fakultät für Maschinenwesen & Betriebswissenschaften

TU Wien

http://www.imw.tuwien.ac.at

Tutorial: 3-Schichten-Architektur in R/Shiny

Agenda zum gemeinsamen Vormittag

- Motivation: Enterprise Information System in R/Shiny. Warum?
- Workshop: R/Shiny-Konzepte (I, II, III)
- Workshop: REA-Duality (IV, V, VI)

+ „5 Minuten“ Gruppenarbeiten

3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

Tutorial: 3-Schichten-Architektur in R/Shiny

Intended Learning Outcomes

I) Ableiten einer physischen SQLite Datenbank aufgrund eines konzeptionellen Datenmodells

II) Durchführen einer R/DBI Datenbankanbindung zu einer SQLite Datenbank

III) Entwickeln eines R/Shiny User Interfaces mithilfe Shiny Reactivity

IV) Entwerfen einer R6-Ojektorientierten Implementierung

V) Durchführen eines Git-Checkout in Rstudio von github

VI) Analysieren eines vorgegebenen R/Shiny Dashboards

+ VI) Entwickeln eines eignen R/Shiny Dashboards

Motivation: Enterprise Information System in R/Shiny. Warum?

Enterprise Information Systems: Dunn et al. (2006)

… a well-designed REA-based Accounting Information System is the Enterprise Information System.

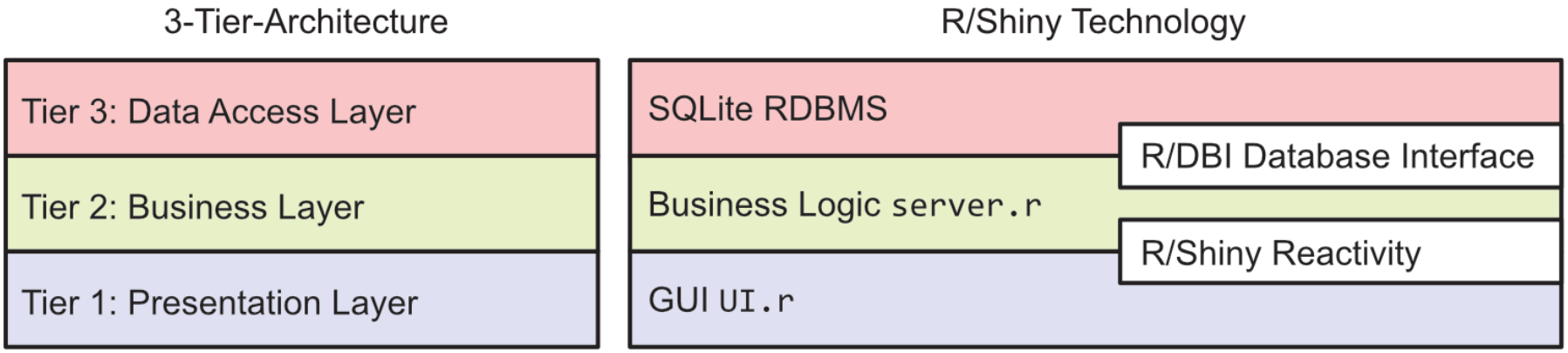
- Bereits bekannte Konzepte
  - *REA-based*
  - *Accounting Information System*
  - *Enterprise Information System*
  - *Well-designed?*

    3-Schichten-Architektur!

Tutorial: 3-Schichten-Architektur in R/Shiny

Motivation: Enterprise Information System in R/Shiny. Warum?

- Datenhaltungsschicht
- Logikschicht
- Präsentationsschicht



Fischer-Pauzenberger, Christian, and Walter SA Schwaiger. "OntoREA© Accounting and Finance Model: Hedge Portfolio Representation of Derivatives." *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, Cham, 2018. *Auf TISS zum Download.*

Tutorial: 3-Schichten-Architektur in R/Shiny

Motivation: Enterprise Information System in R/Shiny. Warum?

- Demonstration: R/Shiny Enterprise Information System als MVP im Unternehmenseinsatz
- Problem: **Lücke in der Varianzanalyse der Expense-Accounts durch ein neues ERP**
- Methode: **REA-EIS als R/Shiny MVP**
  - REA-based?
  - Accounting Information System?
  - Enterprise Information System?
  - Well-designed?
- **Resultat: Integration von Business & MGT mit IT**

„5 Minuten“ Gruppenarbeit:
Erläutern Sie

- **Analyse einer beliebigen Software anhand der 3-Schicht-Architektur**
- **Begriff MVP im Software Engineering**

Tutorial: 3-Schichten-Architektur in R/Shiny

I) REA-Datenmodell: Coding in SQLite

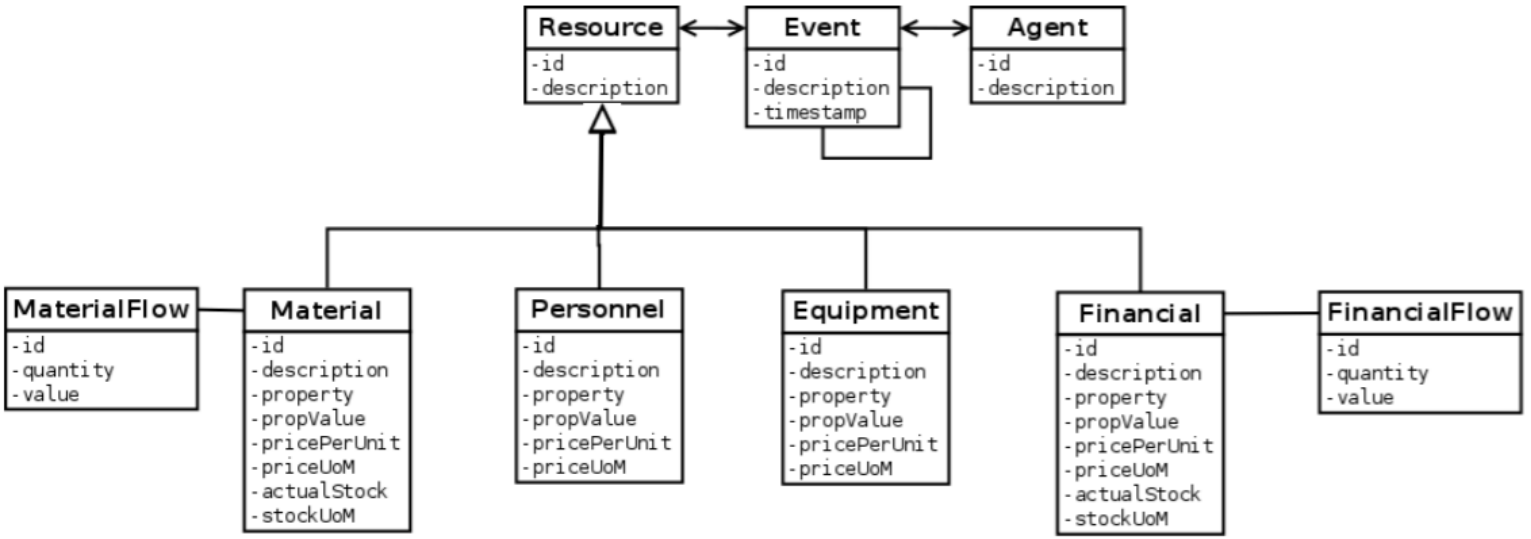
a) REA-Datenmodell: Modellierung

b) REA-Datenmodell: Coding in SQLite

3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

I) REA-Datenmodell: Coding in SQLite

a) REA-Datenmodell: Modellierung



REA-Modell mit Ressourcen-Modellen
ERPControl1905.pdf *Auf TISS zum Download.*

I) REA-Datenmodell: Coding in SQLite

b) REA-Datenmodell: Coding in SQLite

- SQLite als dateibasiertes RDBMS

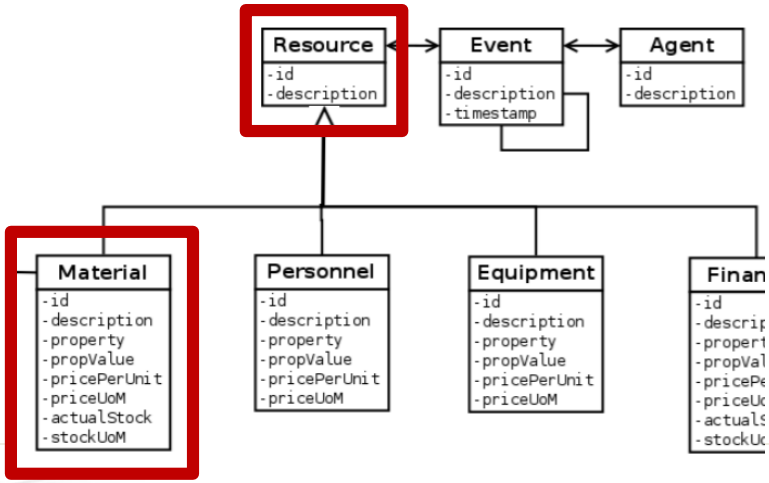
  SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.

- Download und installieren von SQLite Studio
  - Open-source, GPLv3
  - Cross-platform
  - https://sqlitestudio.pl/

I) REA-Datenmodell: Coding in SQLite

b) REA-Datenmodell: Coding in SQLite (EIS.sqlite)

- Anlegen Tabelle Resource mit Attributen
- Anlegen Tabelle Material mit Attributen
- Anlegen Foreign Key *Material.id=Resource.id*



„5 Minuten“ Gruppenarbeit:
Erläutern Sie

- **Objektorientierte Generalisierung in relationalem Datenmodell**

I) REA-Datenmodell: Coding in SQLite

b) REA-Datenmodell: Coding in SQLite

- Einfügen der Stammdaten des Repetierfaktors Material
- Reihenfolge durch Foreign Key Constraint beachten
  1. Resource (id, description)
  2. Material (id, description, …) Die Redundanz (*description*) dient der Veranschaulichung.

Material							
id	**description**	**property**	**propValue**	**pricePerUnit**	**priceUoM**	**actualStock**	**stockUoM**
1	Paraffin	firmness	medium	1,2000	EUR/kg	512,40	kg
2	Wick	thickness	thick	33,2600	EUR/kg	6,81	kg
3	Color	color	red	24,1500	EUR/kg	13,76	kg
4	Container	size	large	0,5177	EUR/kg	189,58	kg
5	Pull Candle	size	regular	1,8260	EUR/kg	7.579,25	kg

Tutorial: 3-Schichten-Architektur in R/Shiny

II) server.R: DBI Database interface

a) R DBI Datenbank Anbindung

b) SQL Abfrage SQLite

c) Speichern als server.R

3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

II) server.R: DBI Database interface

a) R DBI Datenbank Anbindung

- Download und installieren von R
  - Open-source, GPLv3
  - Cross-platform
  - https://cran.r-project.org/
- Download und installieren von RStudio IDE
  - Open-source, GPLv3
  - Cross-platform
  - https://www.rstudio.com/products/rstudio/#Desktop

II) server.R: DBI Database interface

a) R DBI Datenbank Anbindung

- RStudio
  - New Project
  - Install Package RSQlite, *install.packages("RSQLite")*

```
library(RSQLite)
con <- DBI::dbConnect(dbDriver("SQLite"), "EIS.sqlite")
rs <- DBI::dbSendQuery(con, "select * from Material")
df_Material <- DBI::dbFetch(rs)
DBI::dbClearResult(rs)
DBI::dbDisconnect(con)
```

II) server.R: DBI Database interface

b) SQL Abfrage SQLite

```
library(RSQLite)
con <- DBI::dbConnect(dbDriver("SQLite"), "EIS.sqlite")
rs <- DBI::dbSendQuery(con, "select * from Material")
df_Material <- DBI::dbFetch(rs)
DBI::dbClearResult(rs)
DBI::dbDisconnect(con)

View(df_Material)
print.data.frame(df_Material)
```

c) Speichern als server.R

Tutorial: 3-Schichten-Architektur in R/Shiny

III) ui.R: R/Shiny user interface reactivity

a) ui.R Datei anlegen
b) *actionButton* anlegen
c) verbatimTextOutput anlegen
d) Hallo REA ausgeben
e) SQL Abfrage ausführen
f) Tabelle ausgeben

3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

Tutorial: 3-Schichten-Architektur in R/Shiny

III) ui.R: R/Shiny user interface reactivity

a) ui.R Datei anlegen

Grundgerüst ui.R und server.R (vorhandenen Code auskommentieren!)

https://shiny.rstudio.com/images/shiny-cheatsheet.pdf

b) *actionButton* anlegen (ui.R) – **Try and Error approach**

actionButton("actionButton_SagHallo", "Sag Hallo!")

c) verbatimTextOutput anlegen (ui.R)

verbatimTextOutput("verbatimTextOutput_Textfeld")

„5 Minuten“ Gruppenarbeit:
Erläutern Sie

- **R/Shiny**

III) ui.R: R/Shiny user interface reactivity

d) Hallo REA ausgeben (server.R)

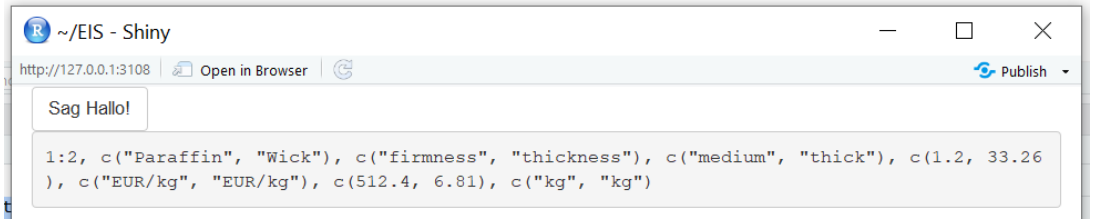
```
observeEvent(input$actionButton_SagHallo,
        {
          output$verbatimTextOutput_Textfeld <- renderText({
            "Hello REA"
          })
        })
```

Tutorial: 3-Schichten-Architektur in R/Shiny

III) ui.R: R/Shiny user interface reactivity

e) SQL Abfrage ausführen (server.R)

```
observeEvent(input$actionButton_SagHallo,
        {
          con <- DBI::dbConnect(dbDriver("SQLite"), "EIS.sqlite")
          rs <- DBI::dbSendQuery(con, "select * from Material")
          df_Material <- DBI::dbFetch(rs)
          DBI::dbClearResult(rs)
          DBI::dbDisconnect(con)
            #View(df_Material)
          #toString(print.data.frame(df_Material))
         output$verbatimTextOutput_Textfeld <- renderText({
           toString(print.data.frame(df_Material))
         })
        })
```



Tutorial: 3-Schichten-Architektur in R/Shiny

III) ui.R: R/Shiny user interface reactivity

f) Tabelle ausgeben (ui.R)

```
tableOutput("tableOutput_Tabelle")
```

f) Tabelle ausgeben (server.R)

```
output$tableOutput_Tabelle <- renderTable({
        df_Material
    })
```

Tutorial: 3-Schichten-Architektur in R/Shiny

IV) Objektorientierung in R

a) Persistence Manager für SQLite

b) OOP: Tabelle ausgeben

3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

IV) Objektorientierung in R

a) Persistence Manager

```
library(R6)
Persistence_Manager <- R6Class("Persistence_Manager", list(
  SQLite_filename = NULL,
  table = NULL,
  table_content = NA,
  initialize = function(SQLite_filename, table) {
    self$SQLite_filename <- SQLite_filename
    self$table <- table
    con <- DBI::dbConnect(dbDriver("SQLite"), SQLite_filename)
    df_Material <- DBI::dbReadTable(con, table)
    DBI::dbDisconnect(con)
    self$table_content <- toString(df_Material)
  }
))
```

Tutorial: 3-Schichten-Architektur in R/Shiny

IV) Objektorientierung in R

b) OOP: Tabelle ausgeben

```
REA_PM <- Persistence_Manager$new("EIS.sqlite", "Material")
        output$verbatimTextOutput_Textfeld <- renderText({
          REA_PM$table_content
        })
```

id	description	property	propValue	pricePerUnit	priceUoM	actualStock	stockUoM
1	Paraffin	firmness	medium	1.20	EUR/kg	512.40	kg
2	Wick	thickness	thick	33.26	EUR/kg	6.81	kg

Tutorial: 3-Schichten-Architektur in R/Shiny

V) Git Versionsverwaltung

a) Git installieren

b) Auschecken eines Git-repositories von github.com in RStudio

„5 Minuten“ Gruppenarbeit:
Erläutern Sie

- **Git**

3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

V) Git Versionsverwaltung

a) Git installieren

- Download und installieren von Git
  - Open-source, GNUv2
  - Cross-platform (MacOS und Linux sollten mit Git ausgestattet sein)
  - https://git-scm.com/download/win

b) Auschecken eines Git-repositories von github.com in Rstudio

- RStudio
  - New Project
  - Git
  - https://github.com/TU-Wien-IMW-FC/Shiny-ERP

Tutorial: 3-Schichten-Architektur in R/Shiny

VI) Shiny Dashboards

a) App.R, server.R und ui.R analysieren

b) Ein neues Modul analog zu Modul 1000 mit beliebiger Funktionalität erstellen

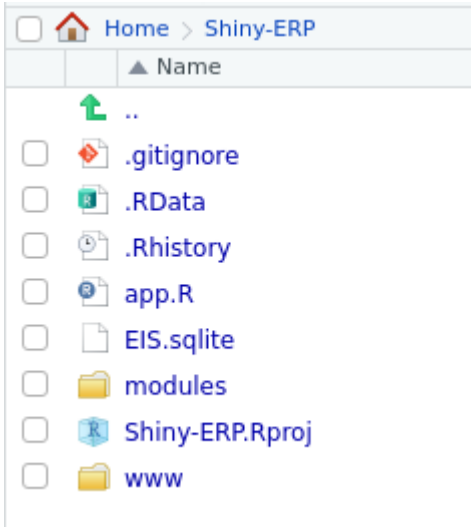
3-Schichten-Architektur	1. Presentation Layer (User Interface)	2. Business Logic	3. Data Access Layer
REA-Duality	**V)** Git Versions-verwaltung **VI)** Shiny Dashboards	**IV)** Objekt-orientierung in R	
R/Shiny-Konzepte	**III)** ui.R: R/Shiny user interface reactivity	**II)** server.R: DBI Database interface	**I)** REA-Datenmodell: Coding in SQLite

VI) Shiny Dashboards

a) App.R, server.R und ui.R analysieren

b) Ein neues Modul analog zu Modul 1000 mit beliebiger Funktionalität erstellen

- SQLite
- Tabellen
- Pivotierung
- Excel-Export



„120 Minuten“ Gruppen-Hausarbeit:

- **VI)b)**

Tutorial: 3-Schichten-Architektur in R/Shiny

Recap: Intended Learning Outcomes

I) Ableiten einer physischen SQLite Datenbank aufgrund eines konzeptionellen Datenmodells

II) Durchführen einer R/DBI Datenbankanbindung zu einer SQLite Datenbank

III) Entwickeln eines R/Shiny User Interfaces mithilfe Shiny Reactivity

IV) Entwerfen einer R6-Ojektorientierten Implementierung

V) Durchführen eines Git-Checkout in Rstudio von github

VI) Analysieren eines vorgegebenen R/Shiny Dashboards

+ VI) Entwickeln eines eignen R/Shiny Dashboards

Literatur

- Dunn, Cheryl L., et al. Enterprise information systems: A pattern-based approach. Vol. 3. Boston: McGraw-Hill/Irwin, 2005.
- Fischer-Pauzenberger, Christian, and Walter SA Schwaiger. "OntoREA© Accounting and Finance Model: Hedge Portfolio Representation of Derivatives." IFIP Working Conference on The Practice of Enterprise Modeling. Springer, Cham, 2018.
  Auf TISS zum Download.
- RStudio. “R/Shiny Cheat Sheet”, 2015.
  https://shiny.rstudio.com/images/shiny-cheatsheet.pdf