

Methodik der OOSE für Fachinformatiker nach dem Lernfeldansatz unter Einbeziehung der Lehrerfortbildung

Dietmar Johlen

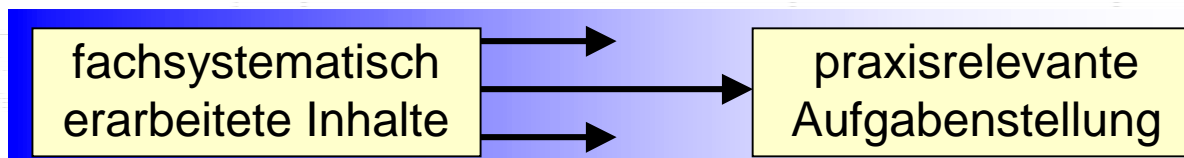
Studienseminar für berufliche Schulen, 34127 Kassel

djohlen@web.de

- Inhalt:
- Lernfeldansatz
 - Ist-Zustand
 - Spannungsfeld
 - Thesen
 - Auf die Situation der Schule angepasstes Vorgehensmodell
 - Das Netzkalendarium
 - Einbeziehung der Lehrerfortbildung
 - Zusammenfassung und Ausblick

Der Lernfeldansatz

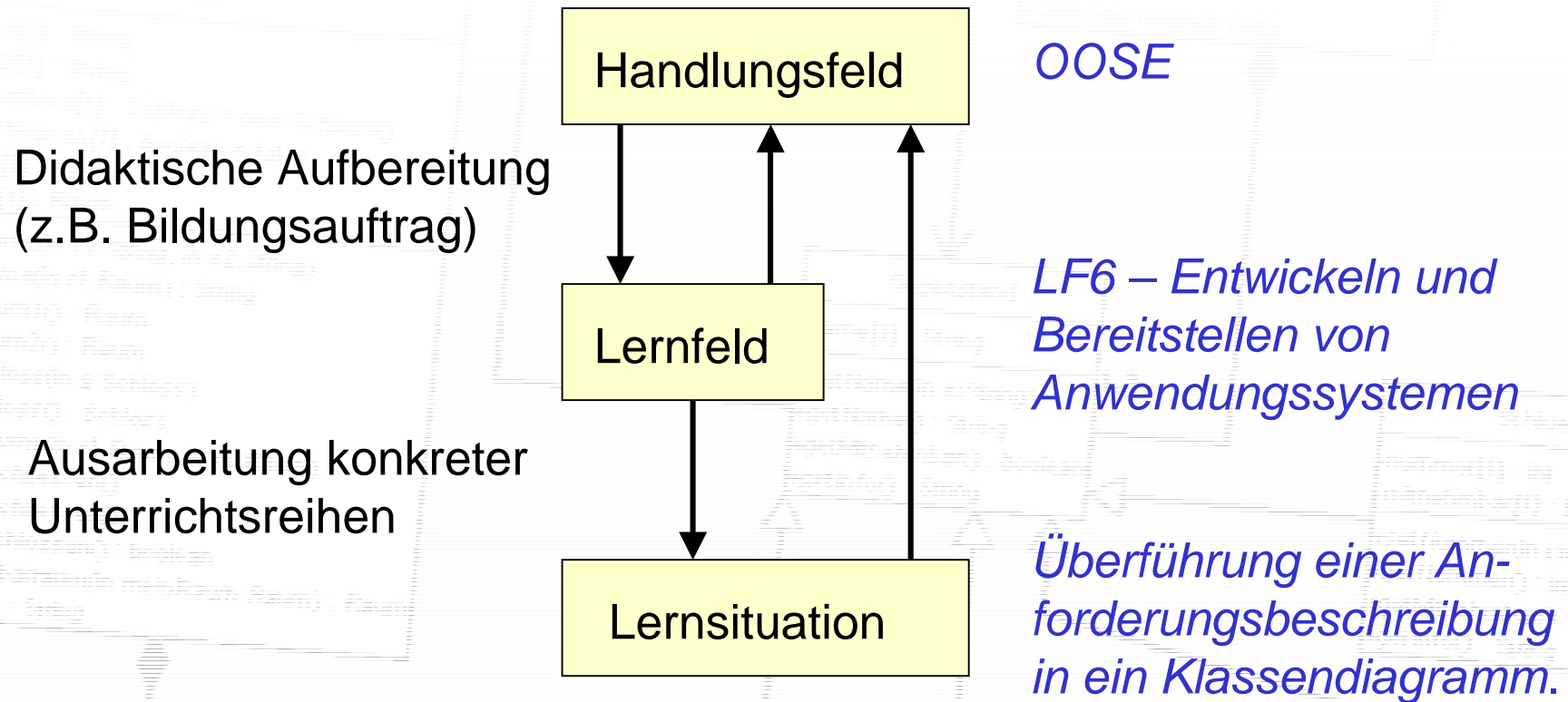
- Mögliche Reformoption für handlungsorientierten Unterricht.
- Curriculare Absicherung von handlungsorientiertem Unterricht.
- Reaktion auf das Transferproblem:



- Rahmenlehrpläne werden zunehmend auf der Grundlage des Lernfeldansatzes formuliert (z.B. für die IT-Berufe).
- Verlagerung von Teilen der curricularen Gestaltung in die Schulen.
- Abkehr von der Fachsystematik. An ihre Stelle tritt die Handlungssystematik (*muss erst gefunden werden*).
- Zentrale Prüfung.



Vom Wissen zum Können



Herausbildung beruflicher Identität als Komponente von sozialer Integration.

Lernfelder für Fachinformatiker

- LF1 Der Betrieb und sein Umfeld
- LF2 Geschäftsprozesse und betriebliche Organisation
- LF3 Informationsquellen und Arbeitsmethoden
- LF4 Einfache IT-Systeme
- LF5 Fachliches Englisch
- LF6 Entwickeln und Bereitstellen von Anwendungssystemen**
- LF7 Vernetzte IT-Systeme
- LF8 Markt und Kundenbeziehungen
- LF9 Öffentliche Netze, Dienste
- LF10 Betreuung von IT-Systemen
- LF11 Rechnungswesen und Controlling

Rahmenlehrplan konkret

Auszug aus dem RLP für Fachinformatiker zum LF6:

Die Schülerinnen und Schüler können komplexe Anwendungssysteme in Projekten analysieren, entwerfen, realisieren und bereitstellen. [...]

Sie wenden für das Entwickeln von Anwendungssystemen eine Programmentwicklungsmethode an und erstellen die (Anwendungs-)Programme auf der Grundlage bekannter Algorithmen und Datenstrukturen unter Nutzung von Softwareentwicklungsumgebungen.

Ist-Zustand

Fokus auf
Programmkontrollstrukturen
(Schleifen, if-Bed.) und
Datentypen.

Wenig Überlapp zwischen
den Lehrinhalten innerhalb
des LF6 und zu
benachbarten Lernfeldern
(z.B. LF7).

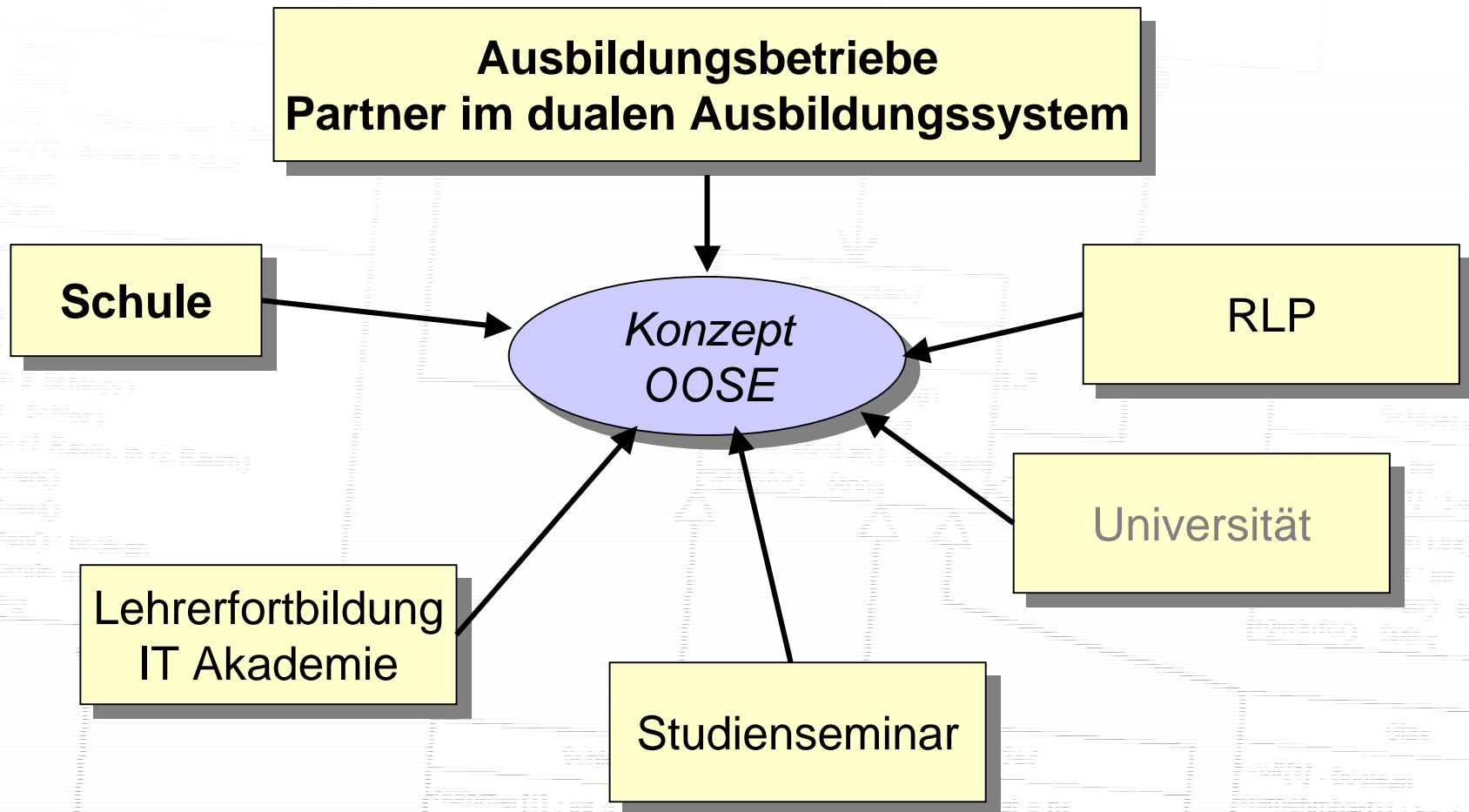
Programmiertechnik stark auf
strukturierte Programmierung
ausgerichtet.

Moderne IDEs
erzeugen Quelltext
automatisch.

Später Umstieg auf
Objektorientierung.

Eindruck, dass die
Objektorientierung nun
einfache Probleme
kompliziert löst.

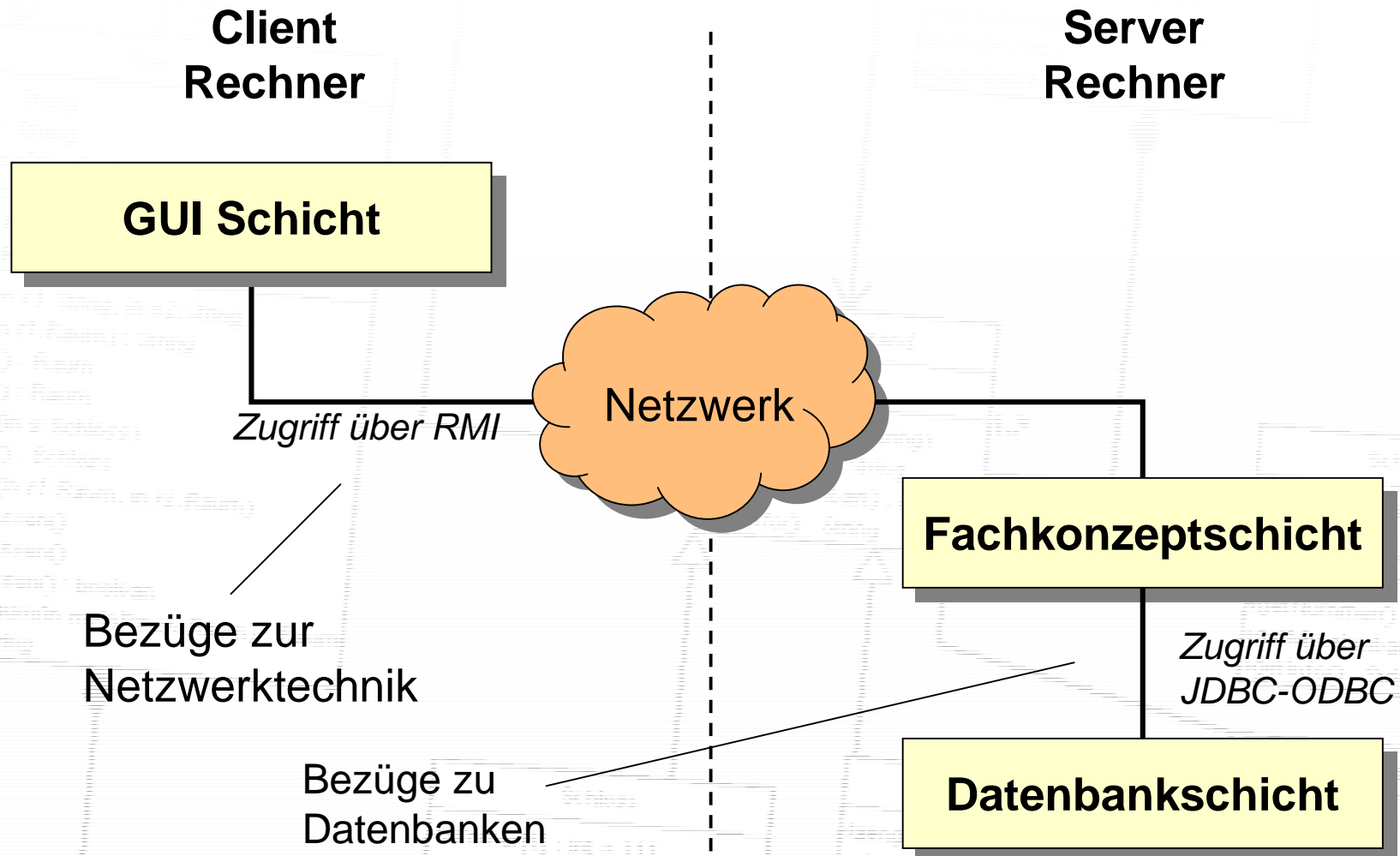
Spannungsfeld



Thesen

- Der Schwerpunkt der OOP liegt auf Objektinteraktionen (Integration von Softwareteillösungen).
- Objektorientierte Modellierung kann ohne die Beherrschung einer objektorientierten Programmiersprache unterrichtet werden.
- Die Verwendung eines Vorgehensmodells hilft den Schülern, eigene Erfahrungen aus den Betrieben einzubringen.
- Gemeinsame Dokumentationsstände zu Beginn jeder Entwicklungsphase fördern das Verständnis des OOSE-Prozesses.
- Die Methoden der OOP können ohne Entwicklungswerkzeuge unterrichtet werden.
- Die OOP fördert vernetztes Denken. Im Gegensatz hierzu verlangt die strukturierte Programmierung linear-analytisches Denken.
- Die OOP wird durch die Metasprache UML (Unified Modeling Language) übersichtlicher.

Verteiltes 3-Schicht-Design



Besondere Situation der Berufsschule

- Blockunterricht (alle 4 Wochen)
 - d.h. 4-5 Unterrichtstermine pro Halbjahr
(Ziel: durchgehende, lernfeldübergreifende Beispiele)
 - Anknüpfung an die zurückliegende Blockwoche
 - Unterrichtsorganisation (z.B. Klausuren)
- Prüfungen IHK
 - Schüler bitten um Vorbereitungsphasen
 - wenig OOSE relevante Aufgaben
- unterschiedliche Betriebe
 - nicht vereinbare Ansprüche an die Schule
 - Absprache für duale Ausbildungsvorhaben
- stark heterogene Klassen (Alter, Leistungsfähigkeit)

Vorgehensmodell

Phasen des
OOSE-Prozesses

Anforderungsbeschreibung

Analyse

Entwurf

Implementierung

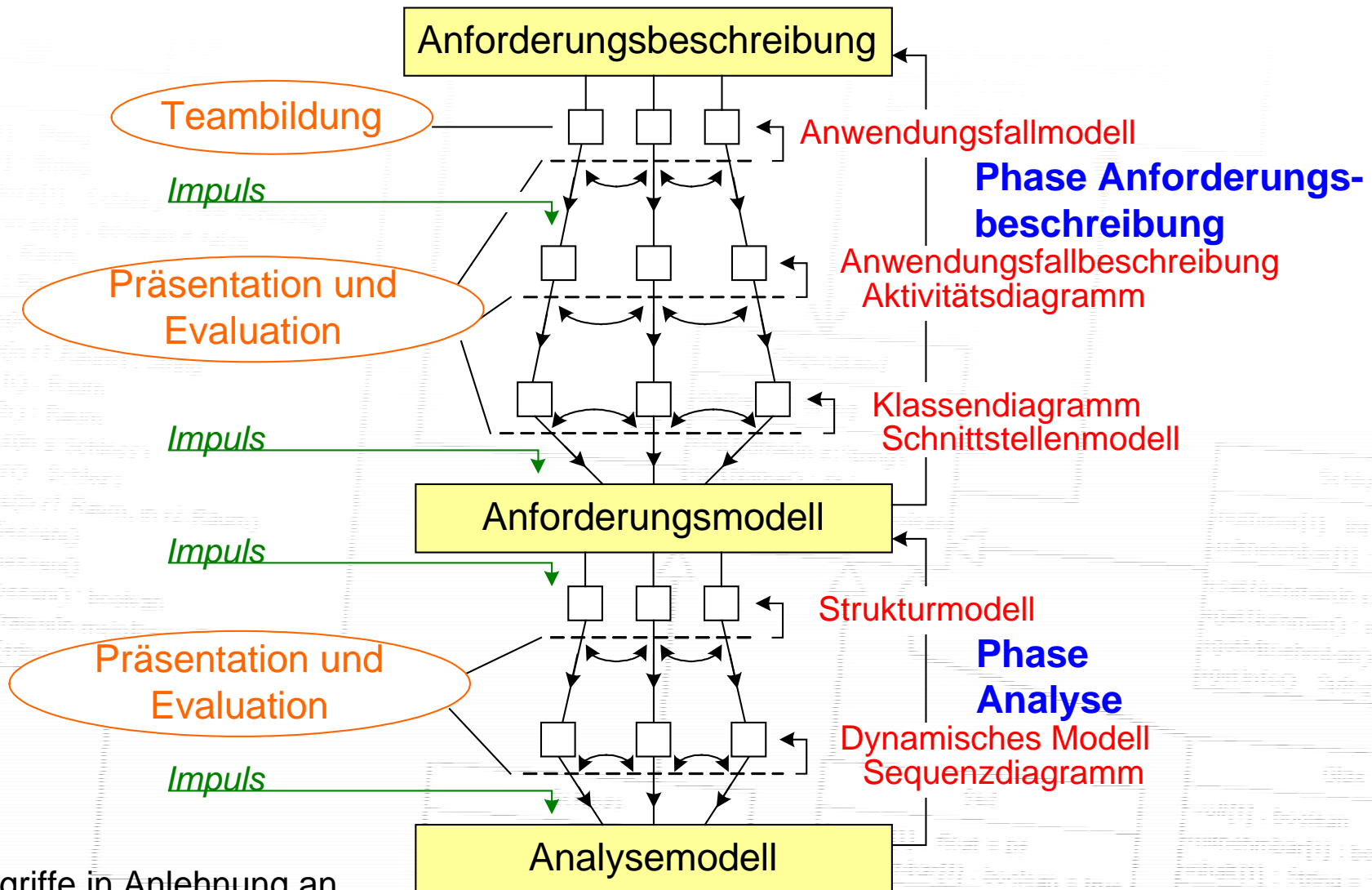
Test

Inbetriebnahme

Einsatz

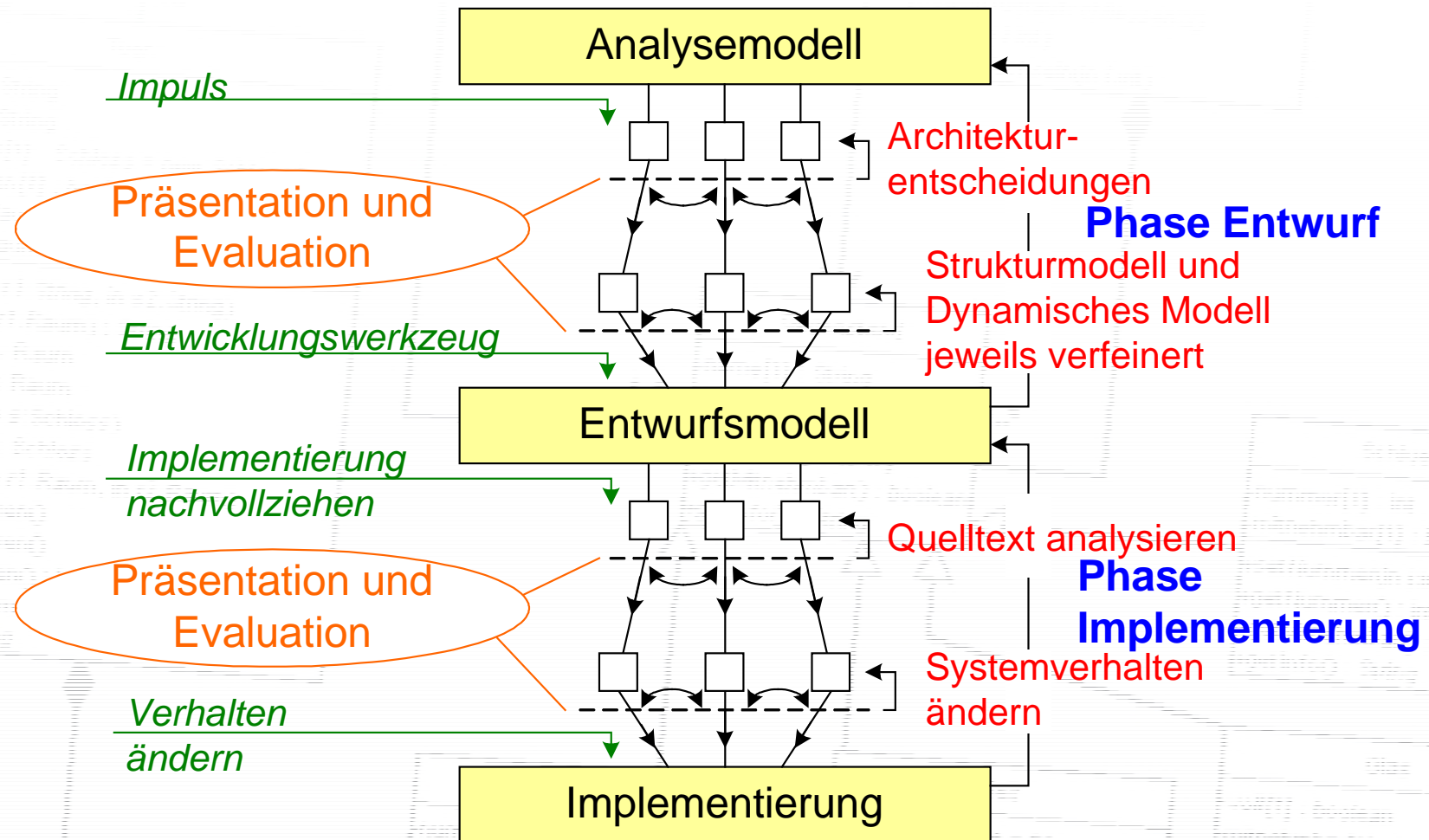
Demissionierung

Vorgehensmodell für die Schule I



(Begriffe in Anlehnung an Hitz, *UML @Work*, dpunkt Verlag)

Vorgehensmodell für die Schule II



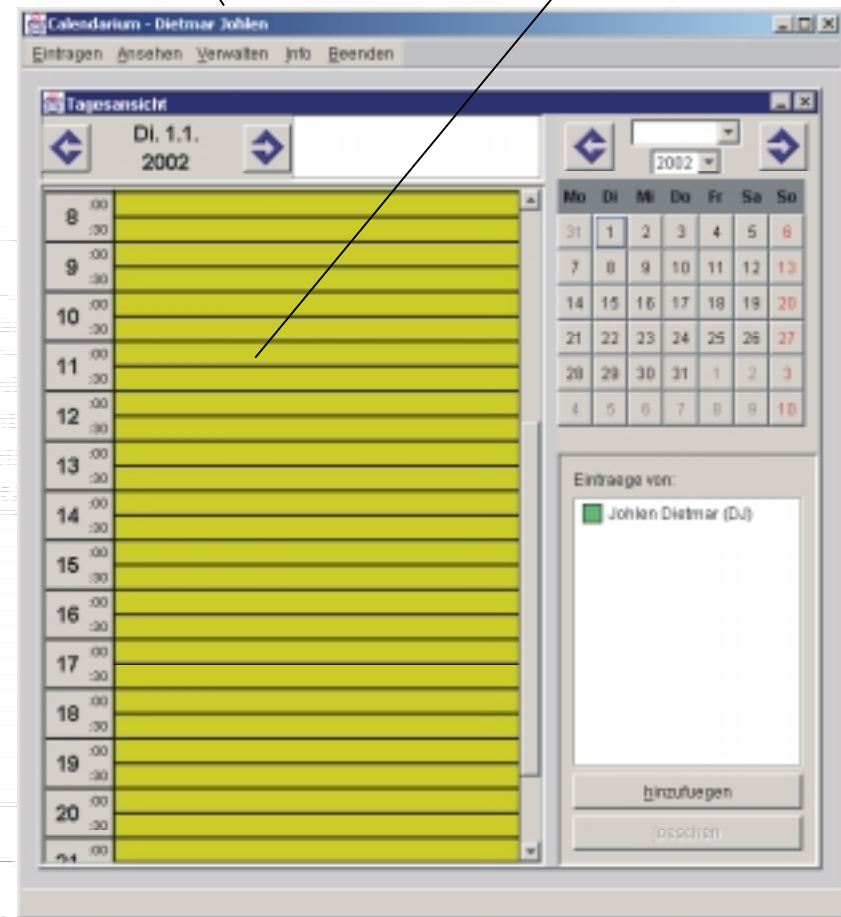
Netzkalendarium

- Terminkalenderprogramm
- mehrbenutzerfähig
- verteilt auf unterschiedliche Hardware Plattformen
- netzwerkfähig
- Verwaltung von To-Do-Listen

Die Hauptaufgabe besteht in der Verwaltung von Terminen und im rechtzeitigen Benachrichtigen aller von einem herannahenden Termin betroffenen Personen.

Client Fenster

Tagesansicht



(In Anlehnung an Hitz, *UML @Work*, dpunkt Verlag)

Anwendungsfall- beschreibung

- Beschreibung für den Anwendungsfall „Termin ändern“

- Als Vorlage dient eine Maske für die Anwendungsfallbeschreibung.

- Die Erarbeitung erfolgt mit Papier und Bleistift.

Anwendungsfall: Termin ändern

Kurzbeschreibung: Ein Termin kann verändert werden, wenn der Benutzer entsprechende Berechtigungen besitzt. Alle betroffenen Teile müssen benachrichtigt werden. Termine müssen in allen geöffneten Kalendern aktualisiert werden.

Vorbedingung: Benutzer ist dem System bekannt. Der Termin wird von keinem anderen Ben. geändert

Nachbedingung: Termin geändert, Teilnehmer verständigt, alle Sichten aktualisiert

Fehler situation: Benutzer ist nicht berechtigt, den Termin zu ändern.

Nachzustand im Fehlerfall: Termin bleibt unverändert

Standardablauf:

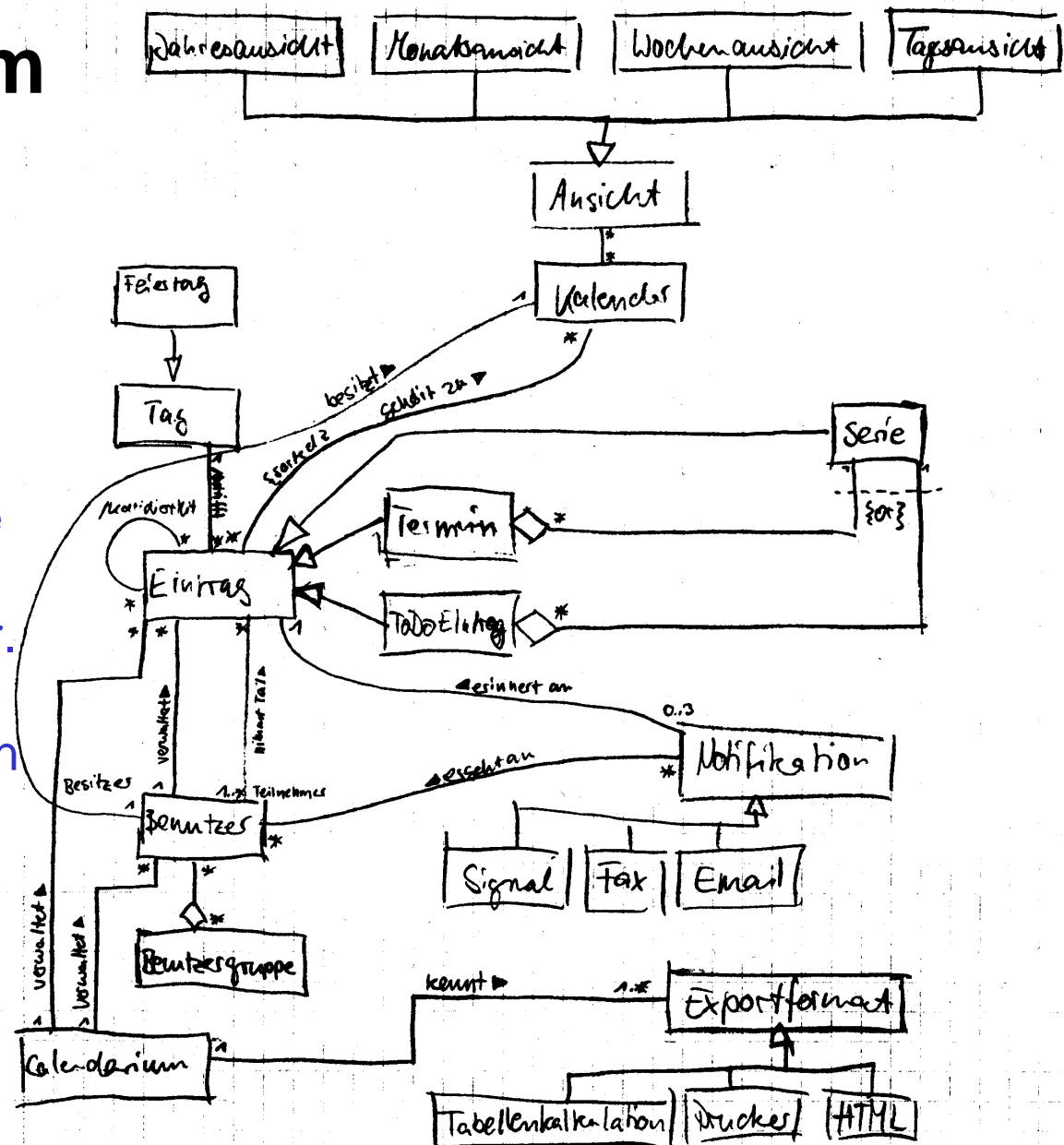
1. Benutzer meldet sich an
2. zu ändernder Termin wird ausgewählt
3. Daten des Termins werden geändert
4. Termin wird in Datenbank geändert
5. Betroffene Teilnehmer werden benachrichtigt
6. Sichten werden aktualisiert

Alternativabläufe:

1. Benutzer hat keine entsprechenden Berechtigungen
2. Termin wird zurzeit geändert
3. behebbares Kollisionsproblem
4. nicht behebbares Kollisionsproblem

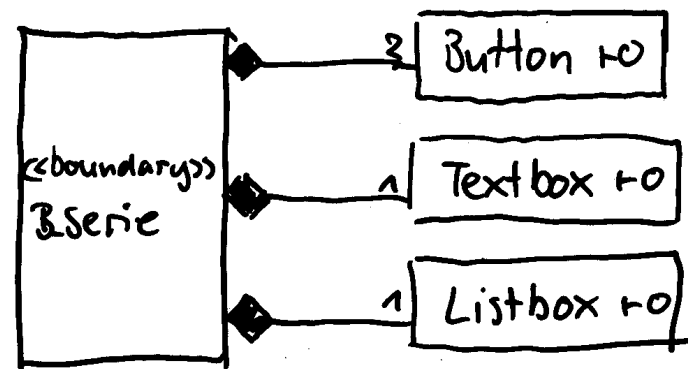
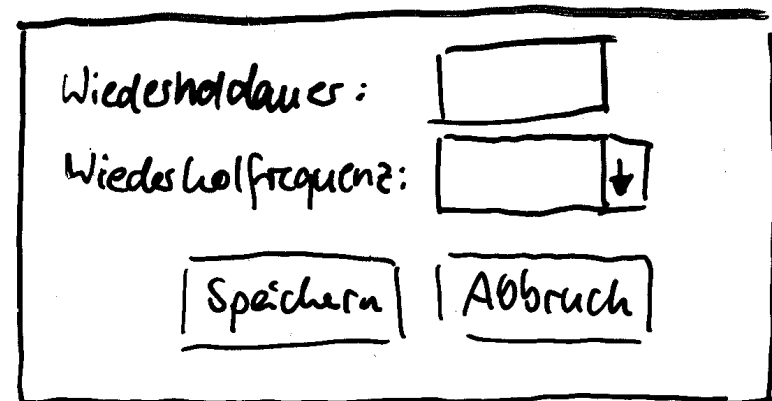
Klassendiagramm

- Klassendiagramm der Phase Anforderungsbeschreibung.
- Aus den Anwendungsfallbeschreibungen abgeleitete Klassen und ihre Beziehungen untereinander.
- Erläuterung der geforderten Eigenschaften der Anforderungsbeschreibung (anwendungsfallgetriebene Entwicklung).



Schnittstellenmodell

- Schnittstellenmodell der Phase Anforderungsbeschreibung.
- Modellierung einer Schnittstelle zum Anwendungsfall „Termin erfassen“ (hier ein Serientermin).



Dynamische Modellierung

Kassel, den 6.3.2002

Firma Objektlabor

Abteilung Implementierung

D. Johlen

Methodenstr. 42

34125 Kassel

Liebes OOA-Team!

Wir haben uns in der Implementierung bereits mit dem neuen Netzkalendarium Projekt beschäftigt.

Mit Ihren aktuell vorliegenden Unterlagen – Anforderungsbeschreibung, Klassendiagramm – haben wir noch Schwierigkeiten. Wir würden uns noch mehr Hilfe zur Erläuterung der Abfolge der Operationen mit den beteiligten Objekten erhoffen.

Vielleicht könnten Sie sich dabei auf den Use Case „Termin löschen“ (s.u.) beziehen.

Gruß,

D. Johlen
Implementierung

Anwendungsfall: Termin löschen

Kurzbeschreibung: Ein bereits eingetragener Termin soll von einem dazu berechtigten Benutzer gelöscht werden. Alle Teilnehmer an dem Termin sollen benachrichtigt werden.

Vorbedingung: Der Benutzer ist dem System bekannt. Termin ist vorhanden.

Nachbedingung: Teilnehmer wurden benachrichtigt.

Termin ist gelöscht.

Kalender sind aktualisiert.

Akteure: Benutzer

Ablauf: 1.) Benutzer meldet sich am Kalendarium an und startet die Maske „Termin löschen“.

2.) Termin wird gelöscht (Berechtigung liegt vor).

3.) Terminnotifikation wird gelöscht.

4.) Teilnehmer (hier z.B. 2) werden über die Löschung benachrichtigt (mit Info Fenster, wenn on-line).

Alternativablauf: 2') Benutzer hat keine Rechte zum Löschen des Termins.

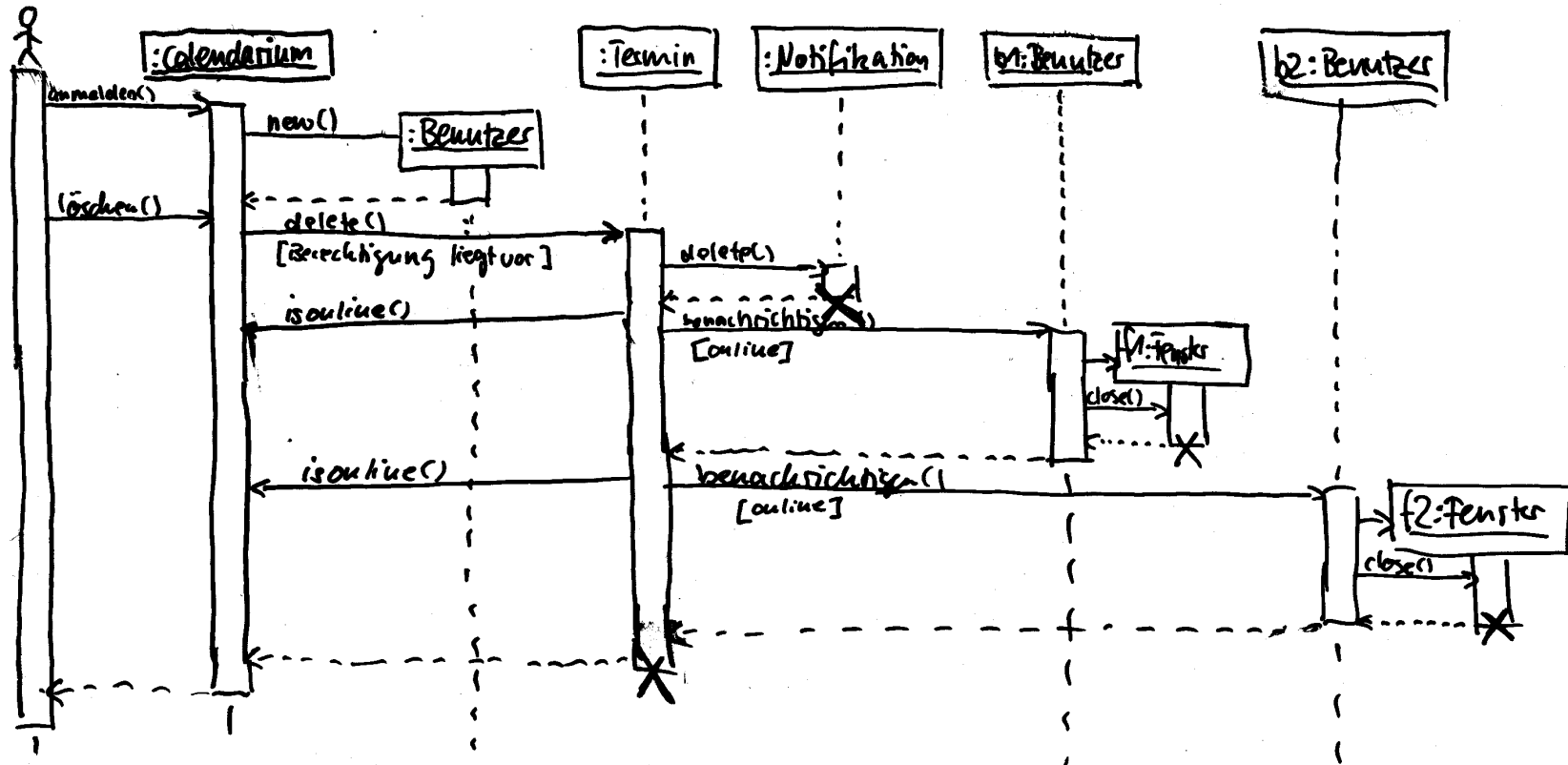
3') Administrator wird benachrichtigt.

Sequenzdiagramm

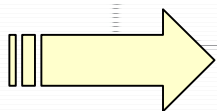
mögliches Vorgehen:

- sinnvolle Objekte bestimmen lassen.
- auf ein gemeinsames, „leeres“ Sequenzdiagramm einigen.
- Anwendungsfall umsetzen. Der Ablauf des Anwendungsfalls wird auf die Kommunikation der beteiligten Objekte übertragen.
- bewusst dynamisches Modell auf der Grundlage eines unvollständigen Klassendiagramms erstellen lassen.

Sequenzdiagramm-Schülerlösung



An dieser Stelle wird den Schülern deutlich, dass Objekte bzw. Klassen nicht nur Information kapseln. Sie übernehmen auch Kontroll- und Schnittstellenaufgaben.



Klassendiagramm um Kontroll- und Schnittstellenklassen erweitern (EBC-Kollaboration).

Weiteres Vorgehen

- Wahl der Programmiersprache und Entwicklungsumgebung.
- Anpassung der Modellierungsergebnisse auf sprachspezifische Eigenschaften.
- Einsatz von Designmustern.
- Exemplarische Umsetzung von UML Elementen in Quelltext (erster Einsatz von CASE-Tools).
- Vergleich mit dem lauffähigen Programm (Orientierung in einem komplexen Programm mit Hilfe der UML Diagramme).
- Änderung des Verhaltens (round trip engineering, Änderungslokalität).

Einbeziehung von Quelltext



The screenshot shows the Together 5.5 IDE interface. On the left is a project browser showing a package structure for 'MV_GUIdbRMI'. The main workspace is divided into three panes:

- Class Diagram:** Shows a class hierarchy. 'PerfectTimeBeispiel' inherits from 'UnicastRemoteObject' and 'Angestellter'. 'Angestellter' inherits from 'AdresseT'. 'AdresseT' has attributes 'strasse: String', 'hausnummer: String', 'PLZ: int', and 'stadt: String'. 'Angestellter' has attributes 'a1: AngestellterI', 'dbName: String', 'dbUrl: String', 'rmiUrl: String', 'user: String', and 'password: String'. 'PerfectTimeBeispiel' has methods '+DisplayPerfectTime', '+Test', '+PerfectTime Skel', '+PerfectTime Stub', and '+PerfectTime'.
- Code Editor:** Contains a Java code snippet:


```

      """ + adresse.getStadt() + """);";
      System.out.println(befehl);
      try{
          System.out.println(dbUrl);
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection c = DriverManager.getConnection(dbUrl, user, password);
          Statement s = c.createStatement();
          s.executeUpdate(befehl);
          s.close(); // Also closes ResultSet
          tabellenAusgabe(dbName, dbUrl, user, password);
      } catch (Exception e) {
      }
      
```
- Console:** Shows two error messages:


```

      ldbRMI\PerfectTimeBeispiel\PerfectTime_Stub.class has wrong or missing package statement.
      ldbRMI\PerfectTimeBeispiel\Test.class has wrong or missing package statement.
      
```

At the bottom, a status bar shows 'Together 5.5 -- MV_G...' and the time '22:15'.

**Einsatz von Entwicklungs-
werkzeugen - Together**

Einbeziehung der Lehrerfortbildung

IT Akademie Hessen: (www.ita.hessen.de)

- Gemeinsame Fortbildung für Lehrer und Ausbilder.
- Basisseminar 2*4 Tage Methodenschulung(OOA/ OOD)
- Aufbauseminar 4Tage Netzkalendarium

Hessisches Kultusministerium

Startseite Impressum Kontakt (Auswahl)

IT Akademie Hessen
Bildung und Wirtschaft

Pilotkurs Grundlagen Softwareentwicklungsprozess (OOA/OOD) mit der Modellierungssprache UML

Kurstyp	Basisseminar - 8 Tage (2 x 4 Tage)
Zielgruppe	Lehrer(innen) und Ausbilder(innen) im technischen und kaufmännischen Berufsfeld sowie der IT-Berufe
Voraussetzungen	Kenntnisse in objektorientierter Programmierung (z.B. Java, C++)
RLP-Bezug	Lernfeld 6: Grundlagen Softwareengineering, objektorientierte Methoden, Entwicklung und Bereitstellung von Anwendungssystemen
Ziele	<ul style="list-style-type: none">• Umsetzung sprachlicher Zusammenhänge in die Modellierungssprache UML• UML Syntax: Use Case Diagramme, Klassen-, Sequenz-, Kollaborations-, Aktivitäts-, Zustandsdiagramme• Objektabhängigkeiten: Assoziation, Aggregation, Komposition und Vererbung• Methoden der objektorientierten Analyse (OOA) und des objektorientierten Designs (OOD)

Zielsetzung für das Kursmodul OOSE

- Vollständig modellierte und kodierte Beispiele.
- Kursteilnehmer erarbeiten Unterrichtsmaterialien.
- Fertige Aufgaben- und Klausurvorschläge mit Lösungen liegen vor.
- Organisation der Fortbildung in Anlehnung an den Unterricht, d.h. viel Gruppenarbeit (keine Vortragsveranstaltung).
- Der Referent tritt als Coach auf.
- Einsatz von Entwicklungswerkzeugen nur am Rande.
- Bereitstellung von CBT und WBT Angeboten (Auswahl der Teilnehmer, Vor- und Nachbearbeitung).
- Vernetzung der Kursmodule untereinander (Java Modul behandelt bereits Elemente des Netzkalendariums mit Unterstützung von UML).
- Kommunikation der Vernetzung. Grundlage für lernfeldübergreifenden Unterricht (Programmiertechnik, Datenbanken).

Zusammenfassung

- Das Vorgehensmodell der OOSE eignet sich als Grundlage für die handlungsorientierte Strukturierung des LF6.
- Die phasenweise Zusammenführung der Modellierungsergebnisse erleichtert die Moderation der Unterrichtsreihe.
- Der Verzicht auf ein Entwicklungswerkzeug bis zur Analysephase verstärkt den Blick auf den Prozess der OOSE.
- Die Metasprache UML ist der Schlüssel zur programmiersprachenunabhängigen Softwareentwicklung.
- Die Komplexität des Netzkalendariums lässt die Vorteile eines definierten Entwicklungsprozesses hervortreten. Der Mehraufwand an Modellierungsarbeit erscheint so gerechtfertigt.

Ausblick

- Übertragung auf andere Schulformen: Berufliches Gymnasium, Fachschule für Technik.
- Lernfeldübergreifende Bearbeitung: Programmierertechnik, Datenbankentwicklung und Netzwerktechnik (Perspektive für das Fach Englisch oder den Bereich Politik Wirtschaft).
- Verteilte Softwareentwicklung durch Partnerschulen.

