Kurs OMSI
im WiSe 2014/15

Objektorientierte Simulation mit ODEMx

Prof. Dr. Joachim Fischer
Dr. Klaus Ahrens
Dr. Markus Scheidgen
Dipl.-Inf. Ingmar Eveslage

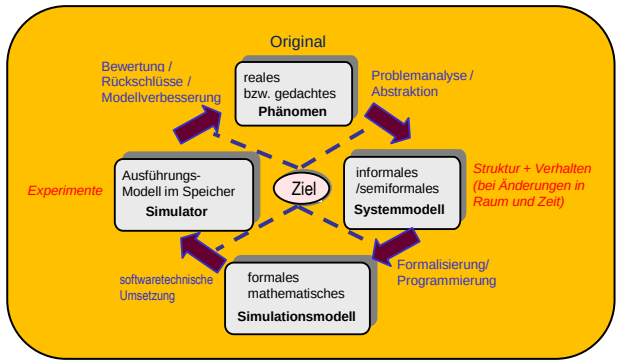
fischer|ahrens|eveslage@informatik.hu-berlin.de

Letzte Vorlesung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Klassifikation dynamischer Systeme
8. M&S eines Niedertemperaturofens

Zusammenfassung

Computersimulation: Charakteristischer ‘Scientific Workflow’



… formuliert als Fragensammlung

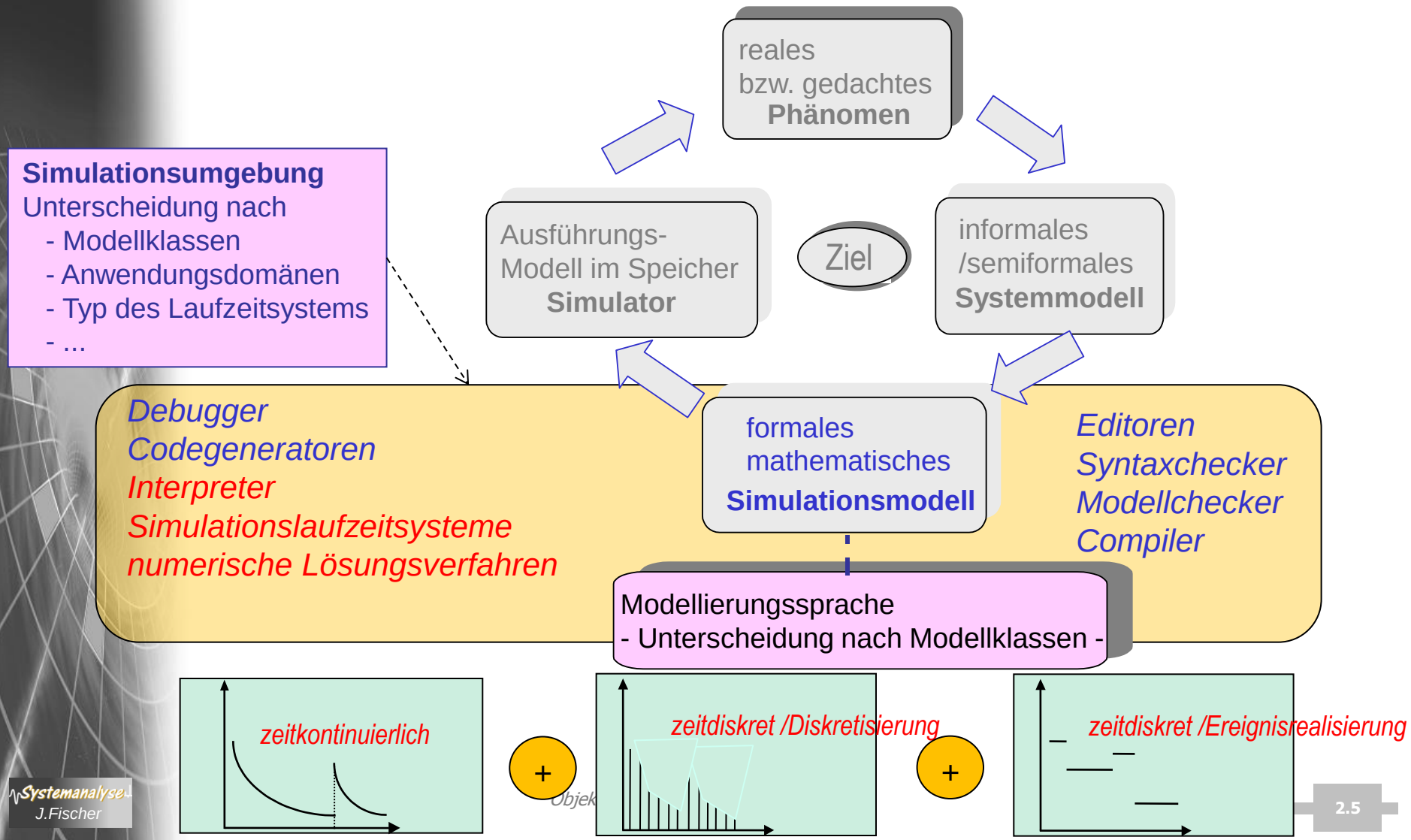
- Charakterisieren Sie Computersimulation als Untersuchungsmethode.
- Was versteht man unter einem Simulator?
- Welche Zeitkonzepte lassen sich bei der Computersimulation unterscheiden?
- Was versteht man unter Echtzeitsimulation?
- Wie kommt man zu ausführbaren Simulationsmodellen?
- Erläutern Sie die Bedeutung des Untersuchungsziels für den Scientific Workflow “Computersimulation”.
- Was versteht man unter einem (dynamischen) System?
- Welche Bedeutung haben Strukturäquivalenzen von Original- und Modellsystem für die Computersimulation?
- Wonach werden Modellsysteme klassifiziert?

1. Einführung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Klassifikation dynamischer Systeme
8. M&S eines Niedertemperaturofens

Modellierungssprachen und Simulationsumgebung

Zustandsänderungen kontinuierlich oder/und diskret in Raum und Zeit

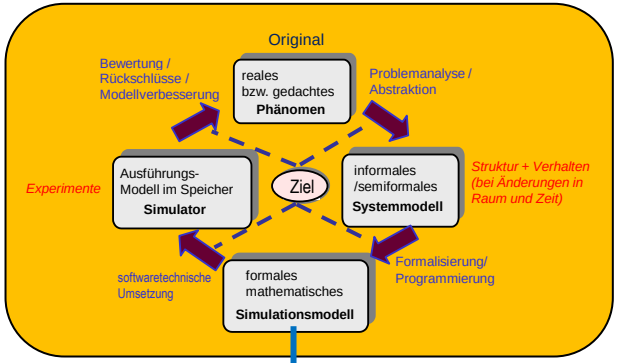


OMSI: C++ als universelle ausführbare Modellierungssprache

~ Simula OO-Sprache mit goto-Anw., Koroutinen
Verwendung als Modellierungssprache: Simulationsbibliothek in Simula „SIMULATION“

Vorteile von C++ als Implementationssprache:

- Portabilität
- Effizienz
- Nutzung weiterer Pakete
- beliebige Konzepterweiterungen sind möglich



UML- Programme

?

ODEMx

alternative Spezialsprache ↔ C++ - Programm

Nachteile von C++ als Modellierungssprache:

- keine kompakte Modellnotation
- aufwendige Anpassung von C++ - *Entwicklungsumgebungen* an Modellierungsdomänen
- für Nicht-Informatiker schwierig zu erlernen (Hardware-Nähe, Pointer-Arithmetik, …)

Klassen, Objekte

passive Klassen, aktive Klassen

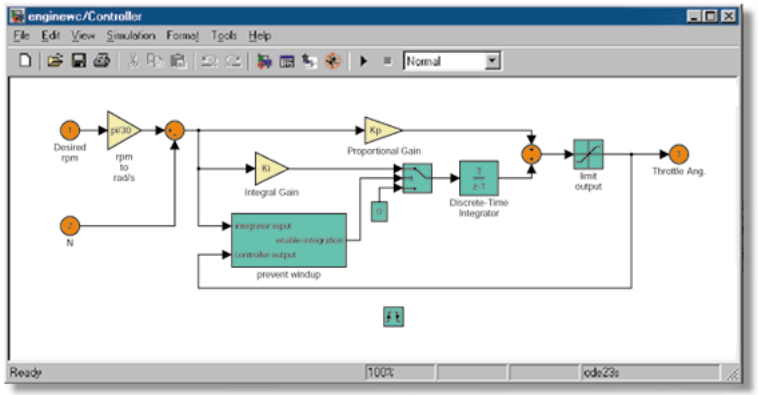
Koroutinen (portable Lösung)

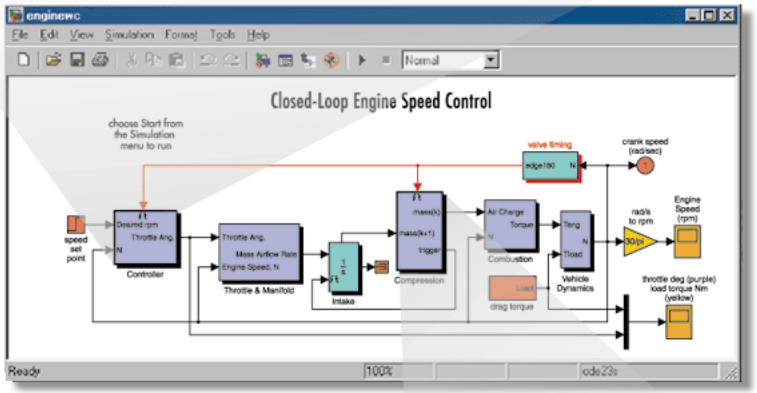
vordefinierte aktive Klassen Process, Continuous

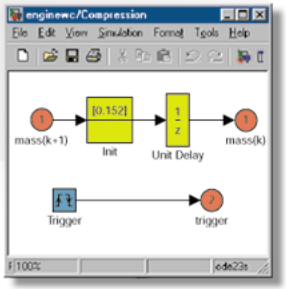
z.B. Bereitstellung eines Debuggers auf einstellbaren Abstraktionslevel (z.B. ODEMx-Level)

unterstützt Kombination von zeitdiskreten und zeitkontinuierlichen Prozessen

Beispiele domänenspezifische Modellierungssprachen





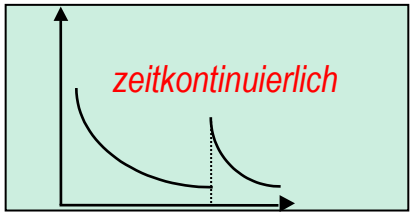


~ hoher Entwicklungsaufwand
(traditioneller Compilerbau
für Spezialsprachen für bestimmte
Anwendungsdomänen)

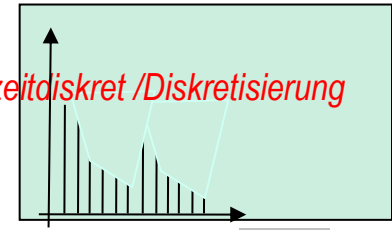
Simulink

- hierarchische graphische Modellierung
- kontinuierliche u. diskreter Schaltblöcke
- S-Functions: eigener Code u. MATLAB

- für einzelne Domänen (wie mechanische, elektrische oder hydraulische Systeme) stehen spezielle Zusätze zur Verfügung, welche die Modellierung von physikalischen Systemen zusätzlich vereinfachen

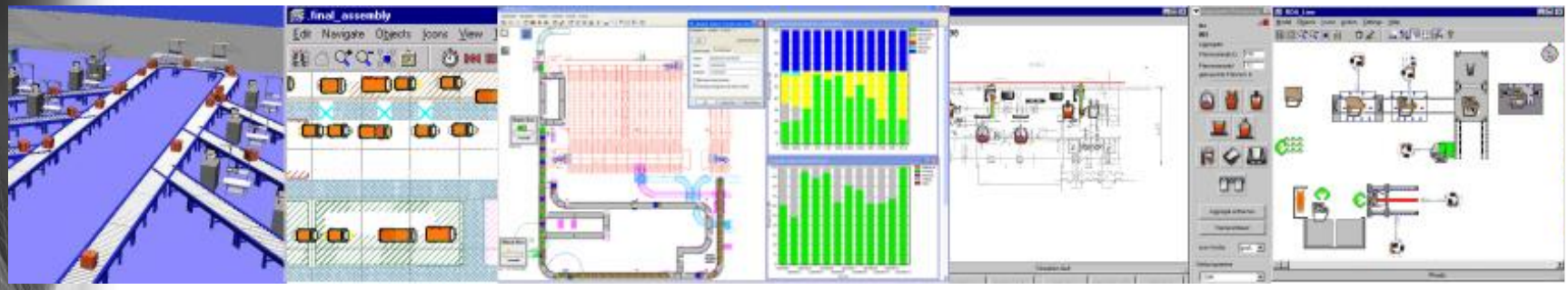


+



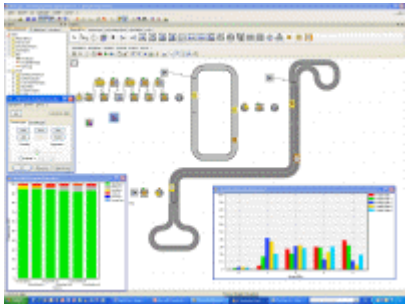
Beispiele domänenspezifische Modellierungssprachen

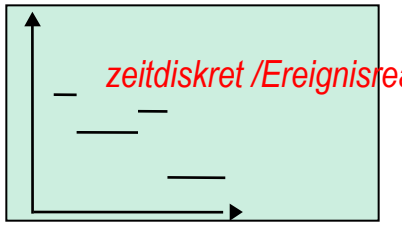
~ sehr hoher Entwicklungsaufwand



Plant Simulation

- graphische Modellierung, Simulation, Visualisierung
- Optimierung von Logistik- und Geschäftsprozessen



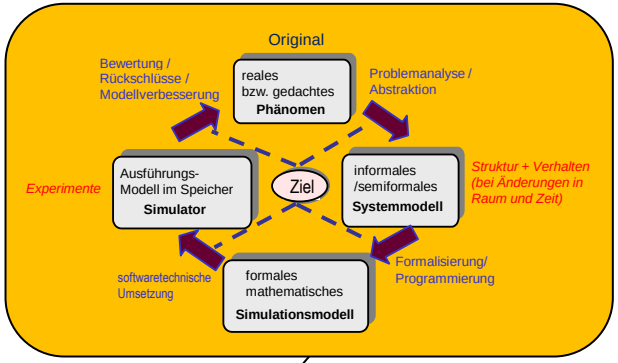


Solution Partner PLM SIEMENS

Zum Vergleich: Modul Workflows

~ Einsatz von SLX einer oo-Sprache als universelle ausführbare Modellierungssprache

Wirtschaftsinformatiker,
WirtschaftsIngenieure,
...



alternative Spezialsprache ↔ C++ - Programm

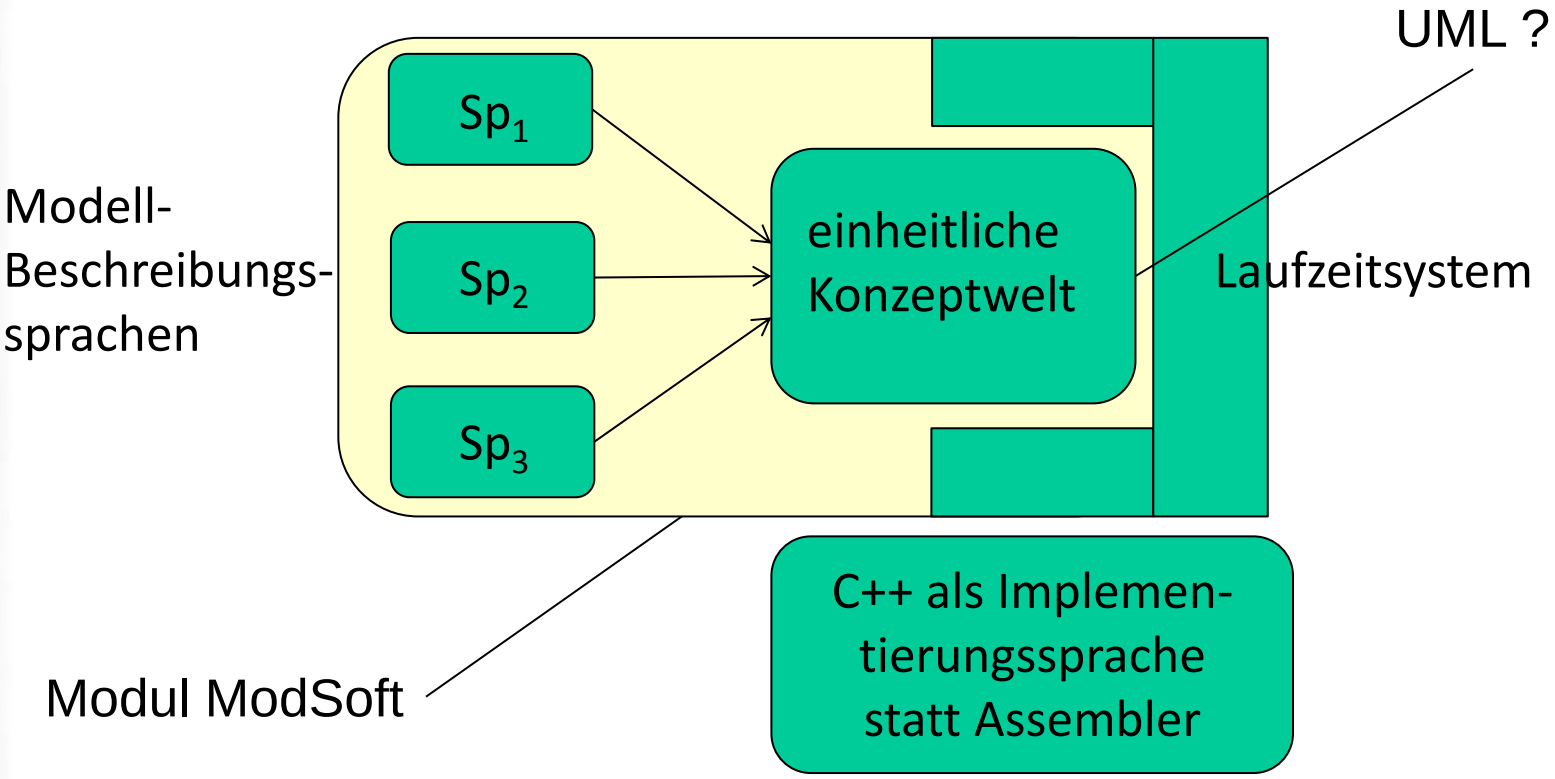
- **Vorteile** von SLX
- Effizienz mit C++ vergleichbar
- wesentlich einfacher als C++
- angepasste Entwicklungsumgebung

Nachteile von SLX :

- eingeschränkte Verfügbarkeit (nur MS Windows)
- nicht so flexibel
- hoher Entwicklungsaufwand (bezogen auf die Nutzermenge) für die Sprache: SLX → Assembler

Vision einer Ideallösung

Trennung von domänenspezifischer
Modellbeschreibungssprache und
effizient ausführbarer Implementierungssprache



1. Einführung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiel(e) aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Klassifikation dynamischer Systeme
8. Scheduler für zeitdiskrete und zeitkontinuierliche Systemmodelle
9. M&S eines Niedertemperaturofens

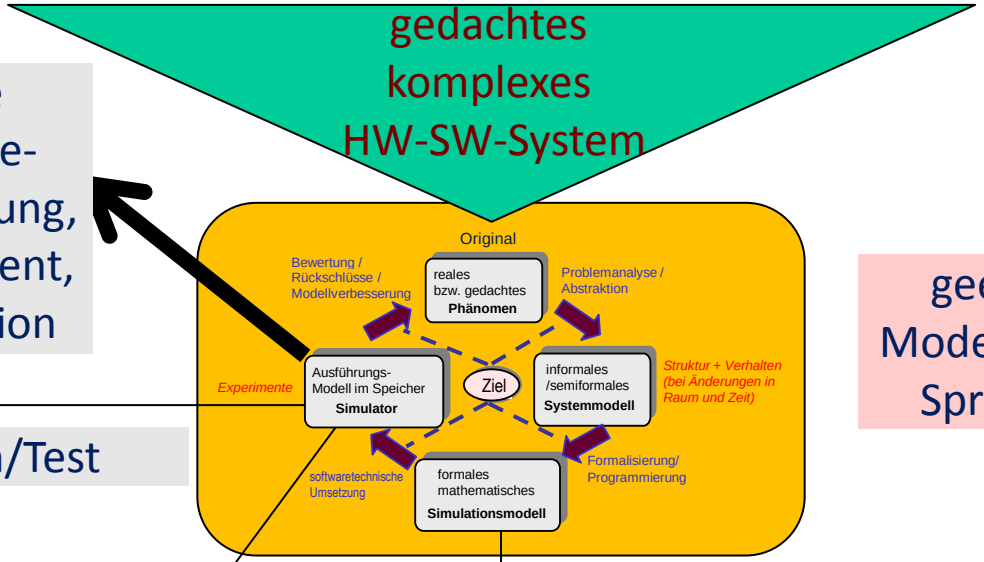
Sonderfall

Traditionelles Gebiet des Software Engineering

Originalsysteme sind Hardware-Software-Systeme
hier: sind Modelle von Softwareprodukten zu erstellen

Was kann die Simulationsmethode hier leisten?

Zielsprachen ?



finale Zielcode-Generierung, Deployment, Installation

Simulation/Test

Model-Checking-Verfahren

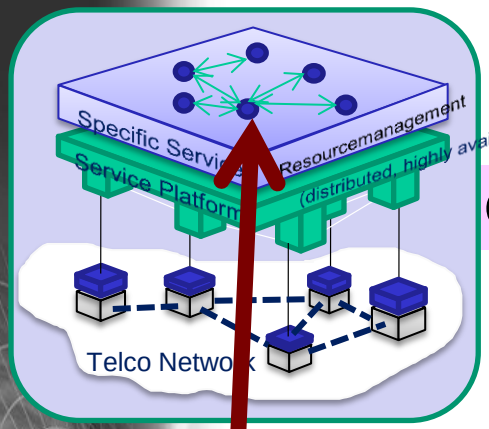
Verfikation

geeignete Modellierungs-Sprachen ?

Ein Beispiel

Entwicklung der Steuerungssoftware von IN-Vermittlungsszentralen der SiemensAG (~ 1994]

Simulation als Funktionstest des Prototypen

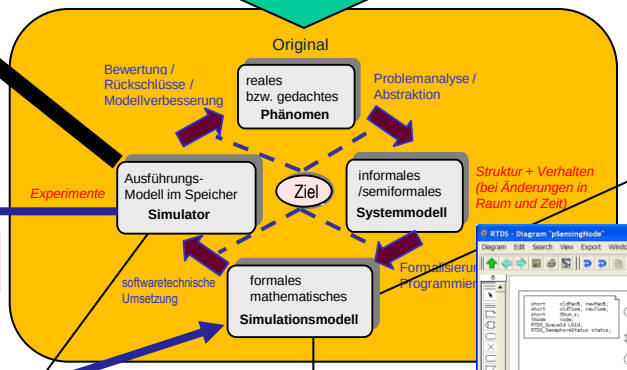


C++/ Siemensplattform

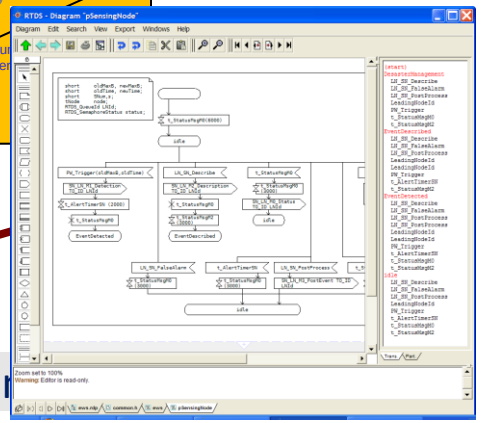
gedachtes komplexes HW-SW-System

finale Zielcode-Generierung, Deployment, Installation

Code-Optimierung



SDL, ASN.1



Simulation/Test

SDL-Laufzeitsystem in ODEMx

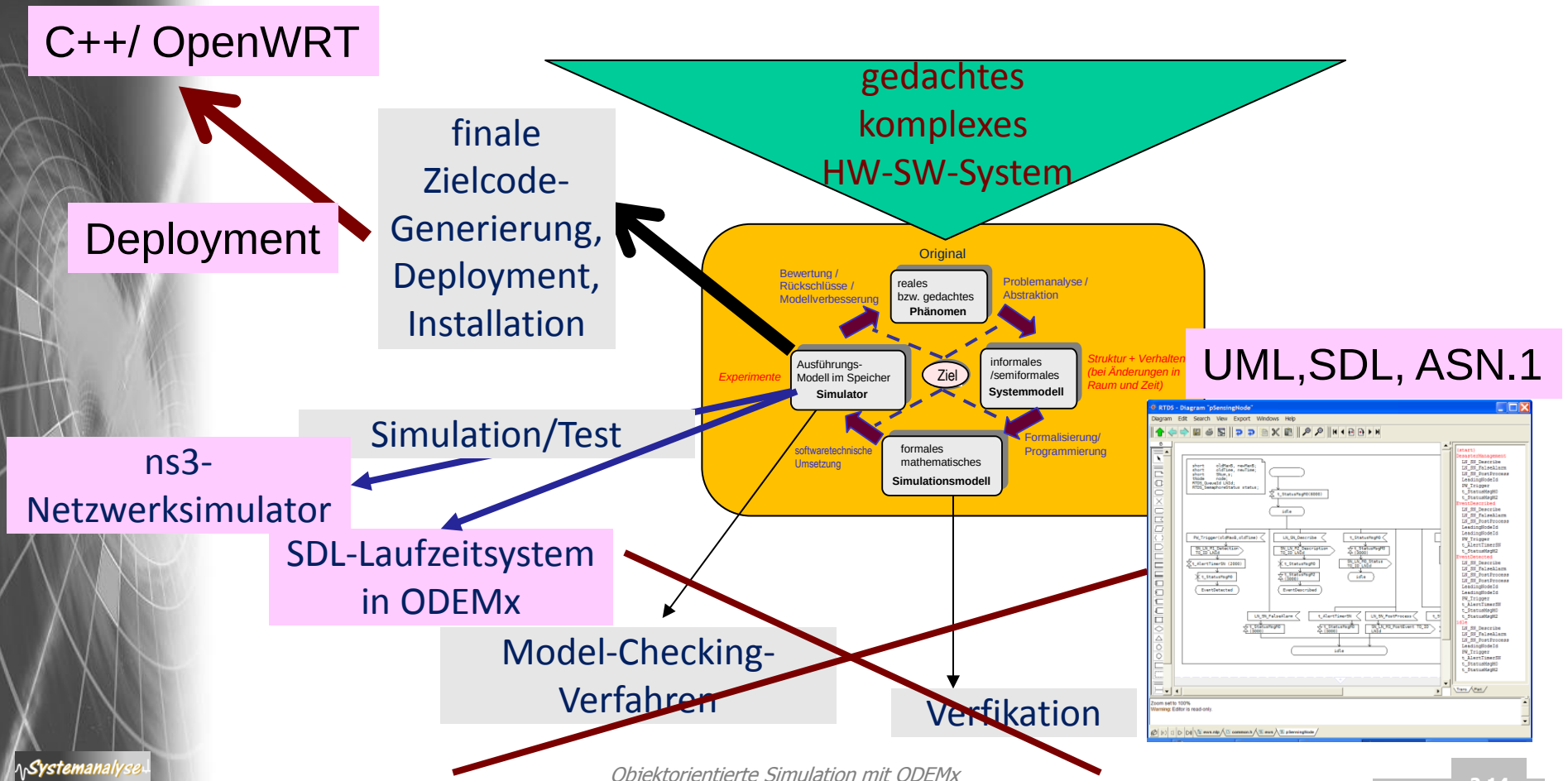
Model-Checking-Verfahren

Verifikation

Modellverbesserung

Ein weiteres Beispiel

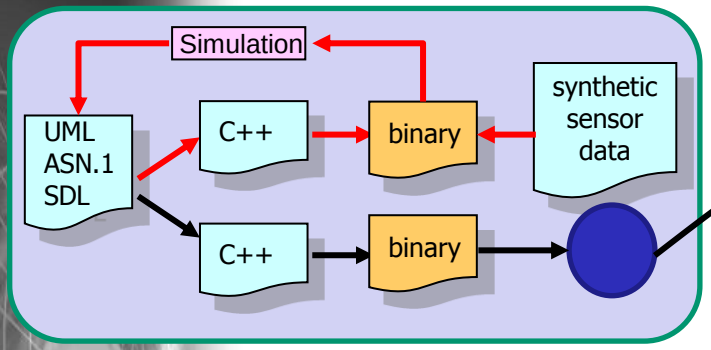
Entwicklung der Steuerungssoftware eines Erdbebenfrühwarnsystems für Istanbul
in Form eines drahtlosen Maschennetzwerkes mit GPS und Sensorik



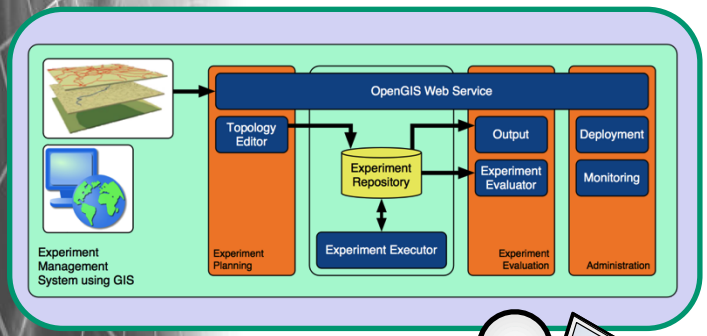
SOSEWIN

Self-organized Seismic Early Warning Information Network

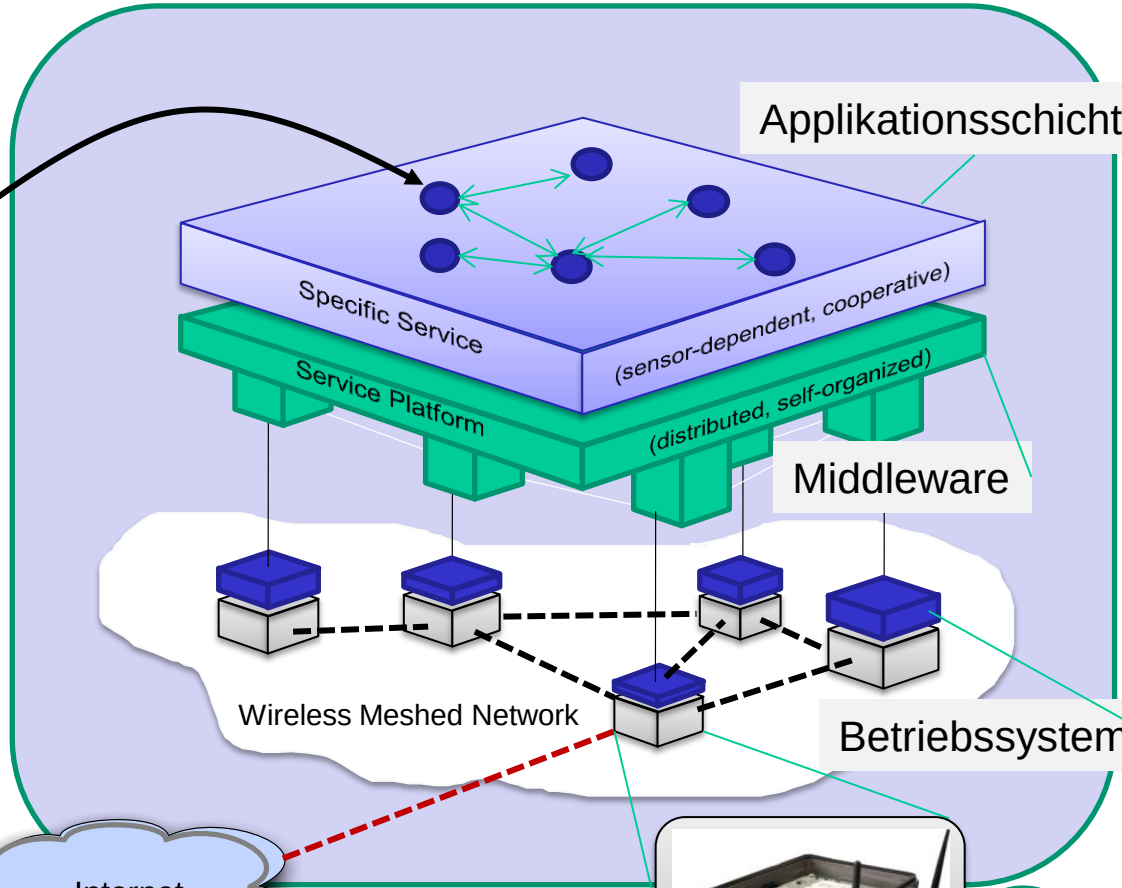
3 SW- Entwicklungstechnologie
(Modelleditor, Simulator, Code -Generator, …)



4 GIS-basierte Netzmanagement- und Experiment-Unterstützung



2 SOSEWIN-HW/SW Architektur



1

Netz-Prototyp

Self-organized Seismic Early Warning Information Network

HU Berlin

GIS

GIS

5

Ataköy Istanbul

Internet

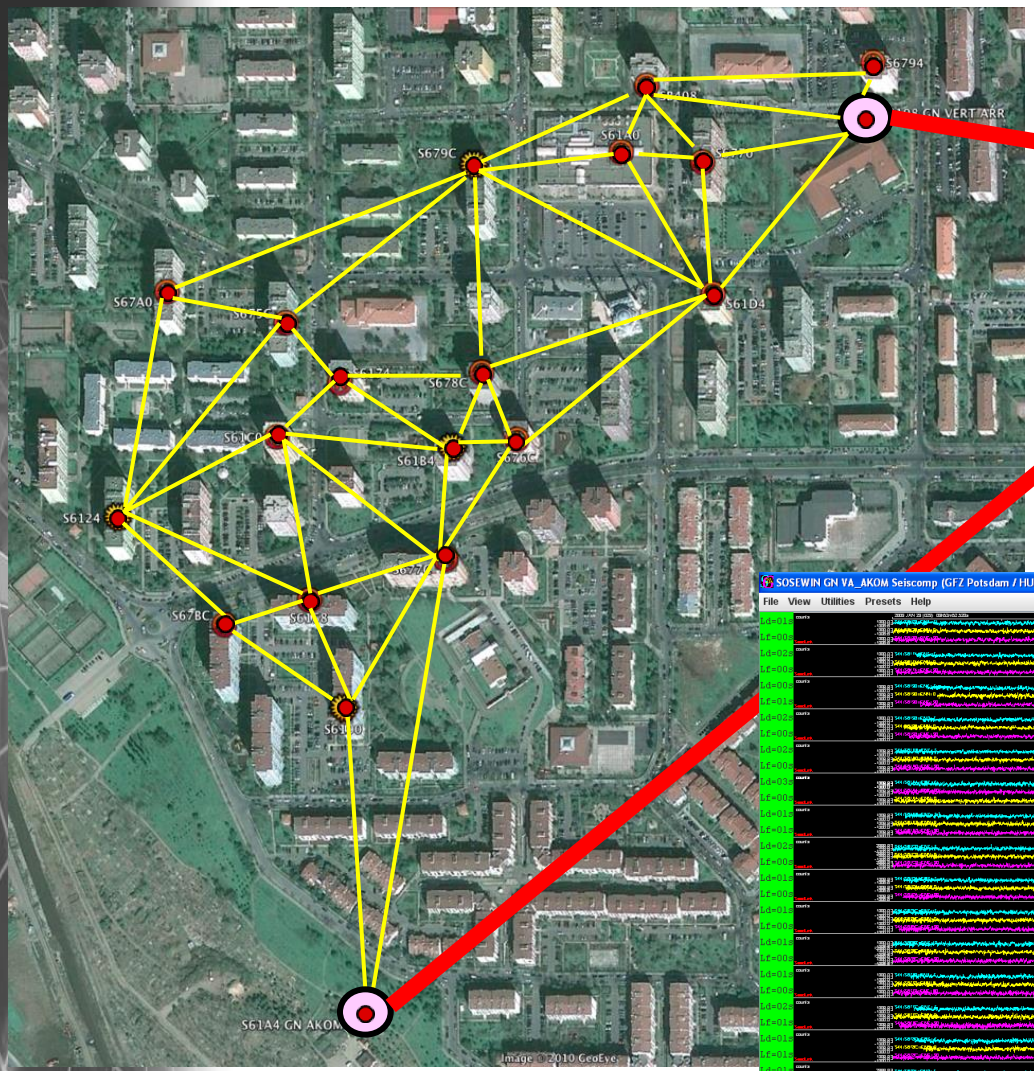
GFZ Potsdam

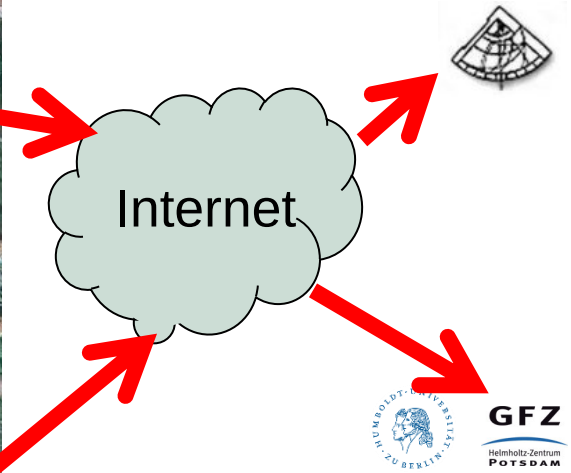
Kandilli Istanbul

GIS

Istanbul-Infrastruktur-daten

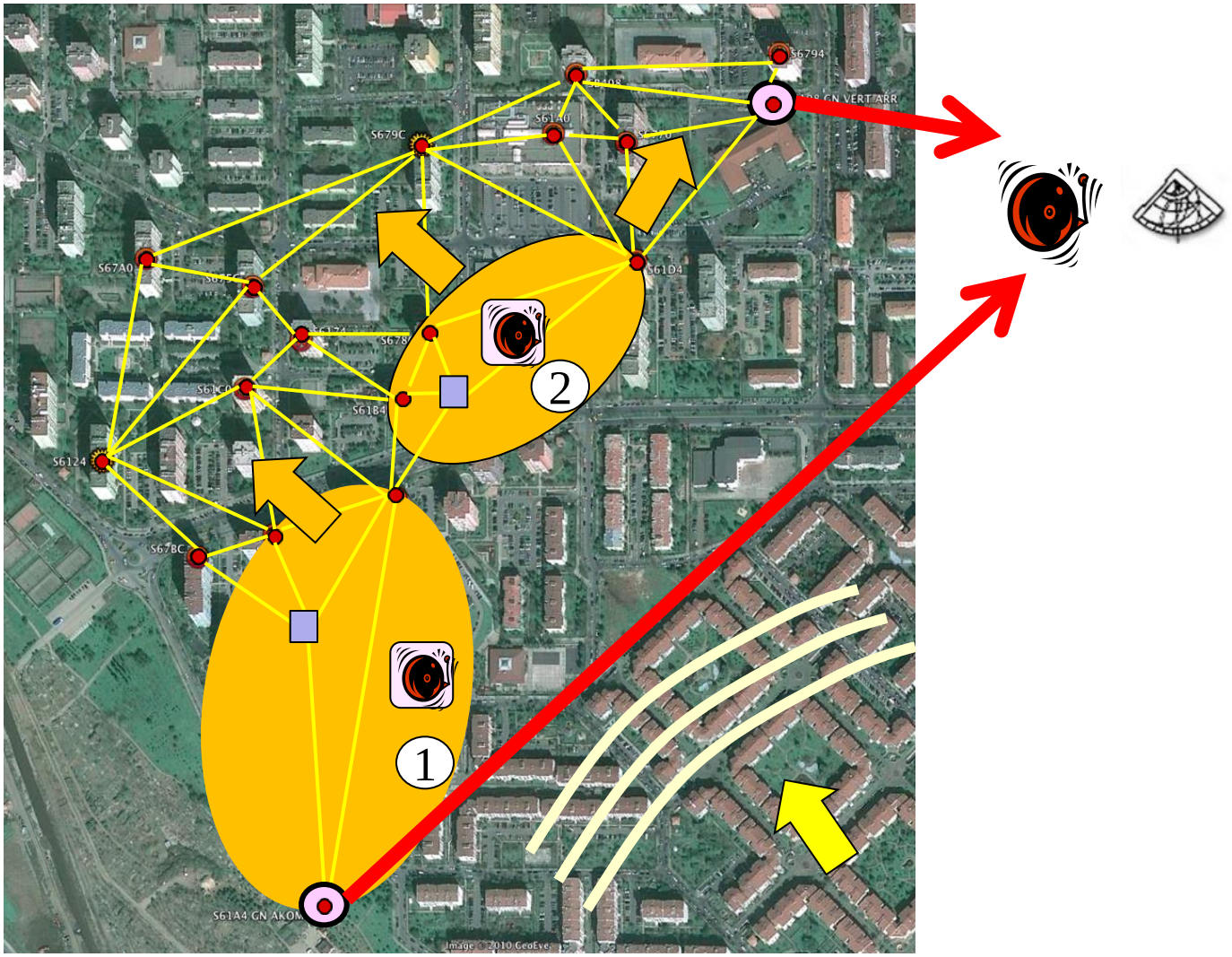
Alle Seismometer arbeiten zusammen





Verzögerung 1..2 s

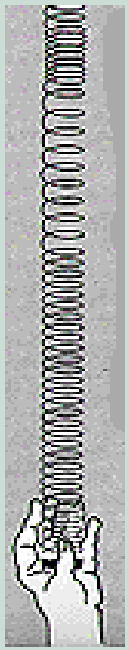
Alarm bei Eintreffen der P-Welle



Wellenarten

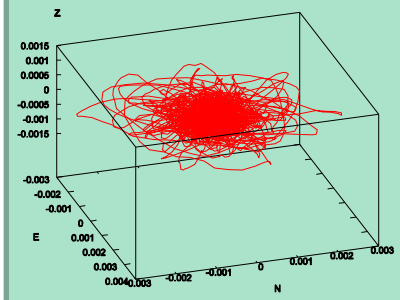
5 km/s Granit

P-Welle

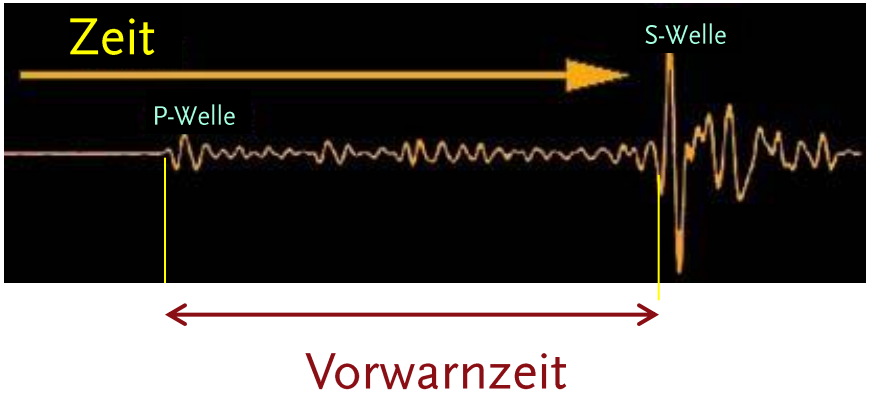


3 km/s Granit

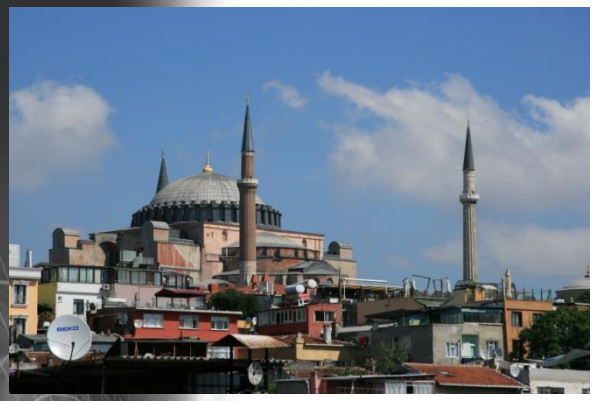
S-Welle

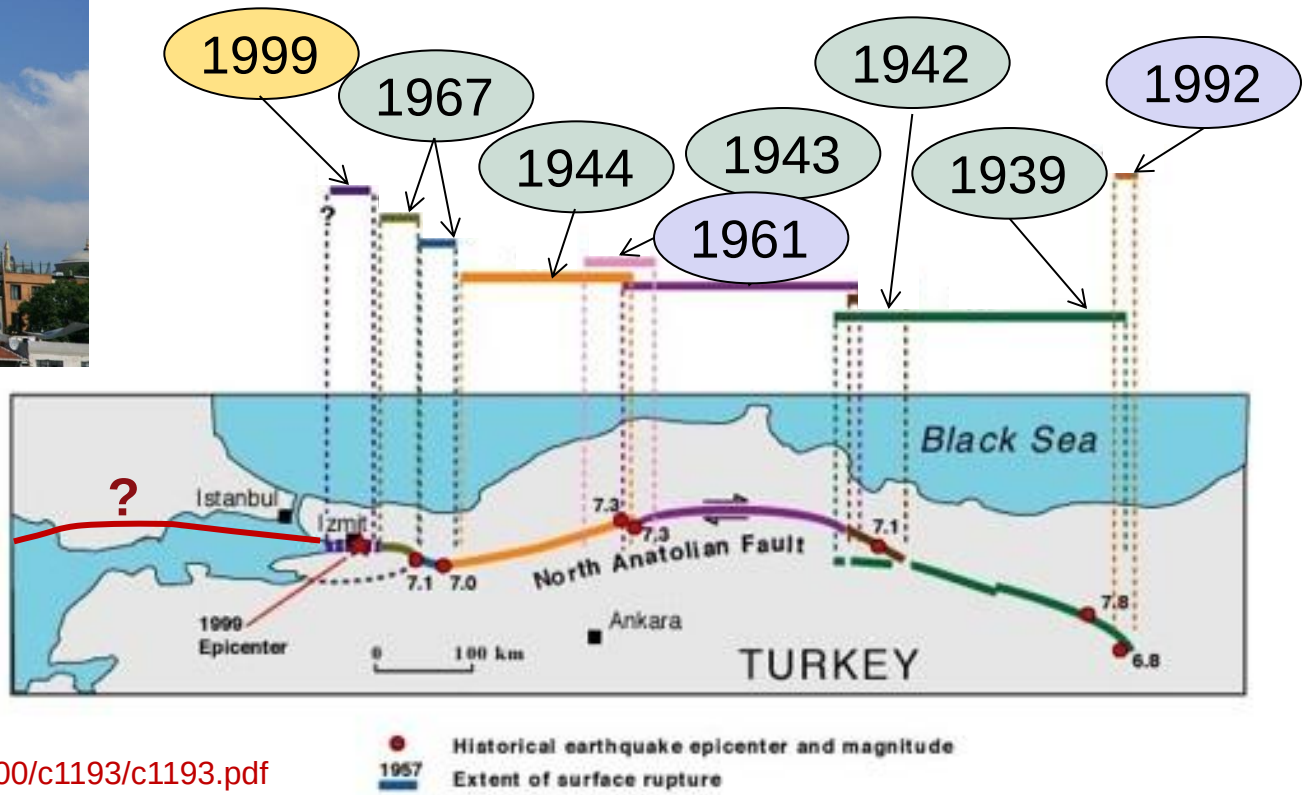


Erschütterung
~ Bewegung
im dreidimensionalen
Raum



Ernste Bedrohung von Istanbul





Quelle:
http://pubs.usgs.gov/circ/2000/c1193/c1193.pdf

Izmet-Beben: M 7,4 ~ 125-fache Energie der Hiroshima-Bombe

- 20.000 zerstörte Häuser
- 40.000 Verletzte
- 24.000 Tote

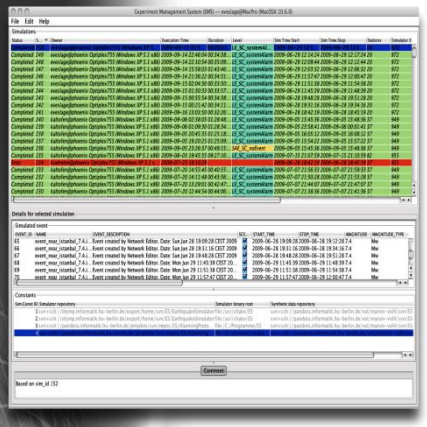
Letzte Warnung 1999: Izmit-Beben

A right-lateral displacement of 2.8 m was observed at the Sapanca segment near Caybasi (40.703° N, 30.451° E)

erte Simulation mit ODEMx

Quelle GFZ Helmholtz-Zentrum Potsdam

Experiment Management Syst



Executables

Earthquake Repository

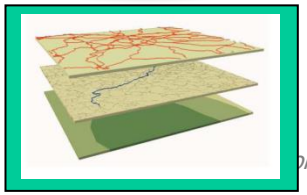
Earthquake Synthesizer

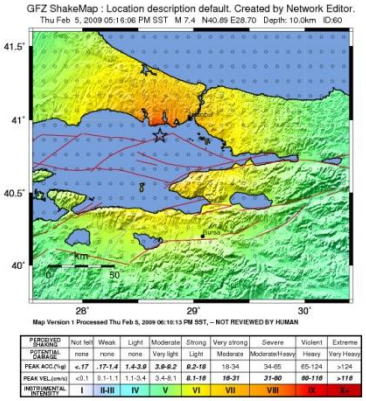
Experiment Repository

Simulation

Virtualisation

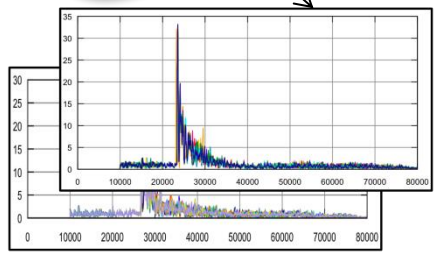
Real World

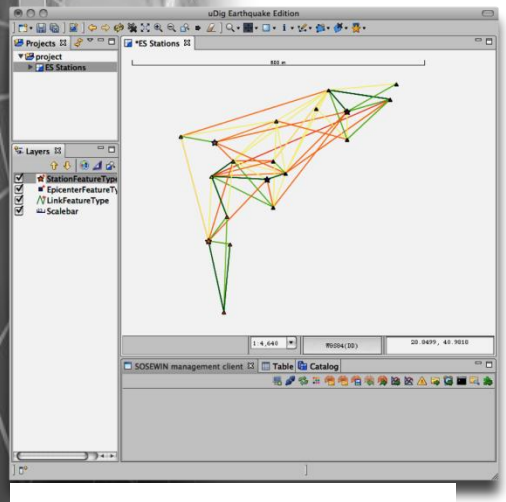




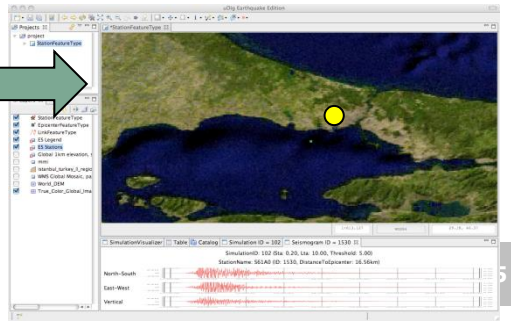
Log File

XML

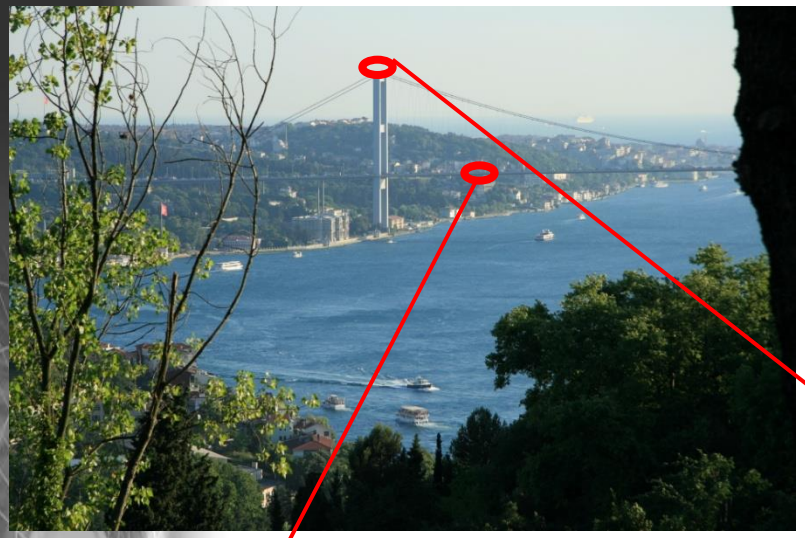




GIS-based editor

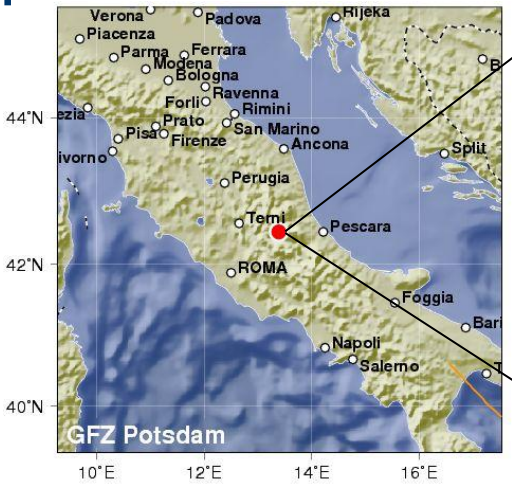


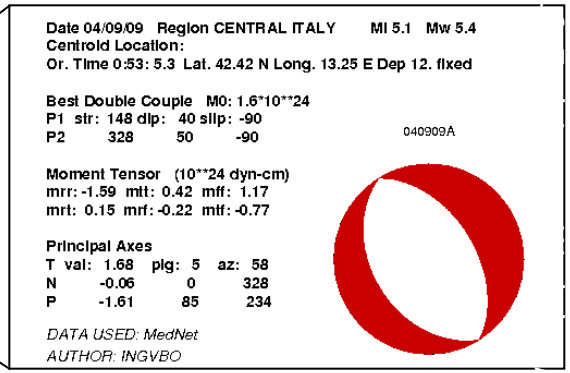
SOSEWIN: temporärer Einsatz (1)



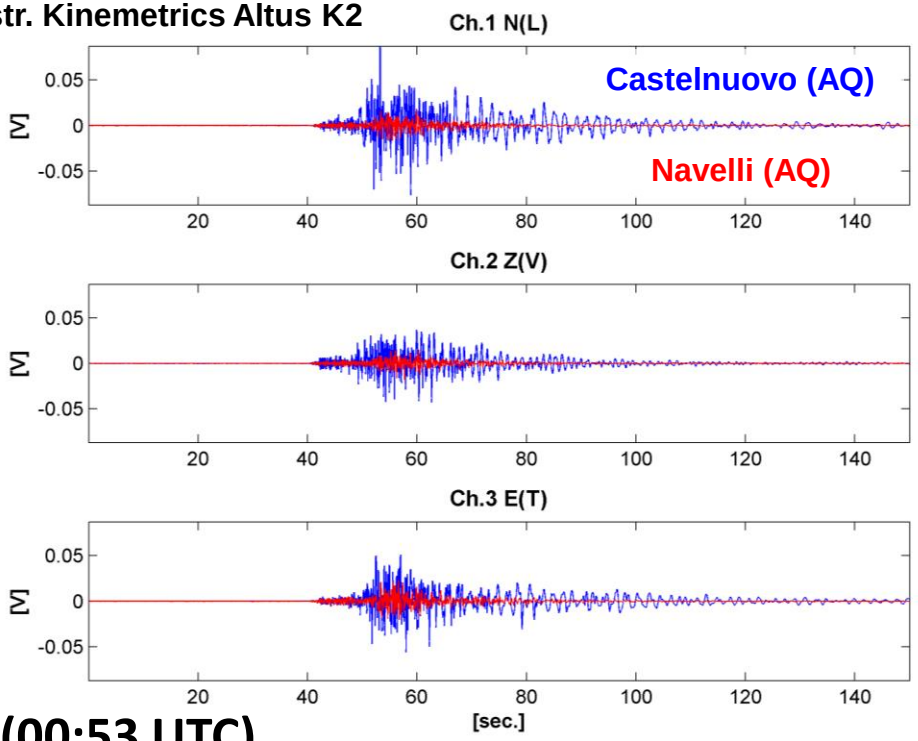
Länge 1.510 m
Breite 39,4 m
Höhe 64 m
Eröffnung 1988

SOSEWIN: temporärer Einsatz (2)



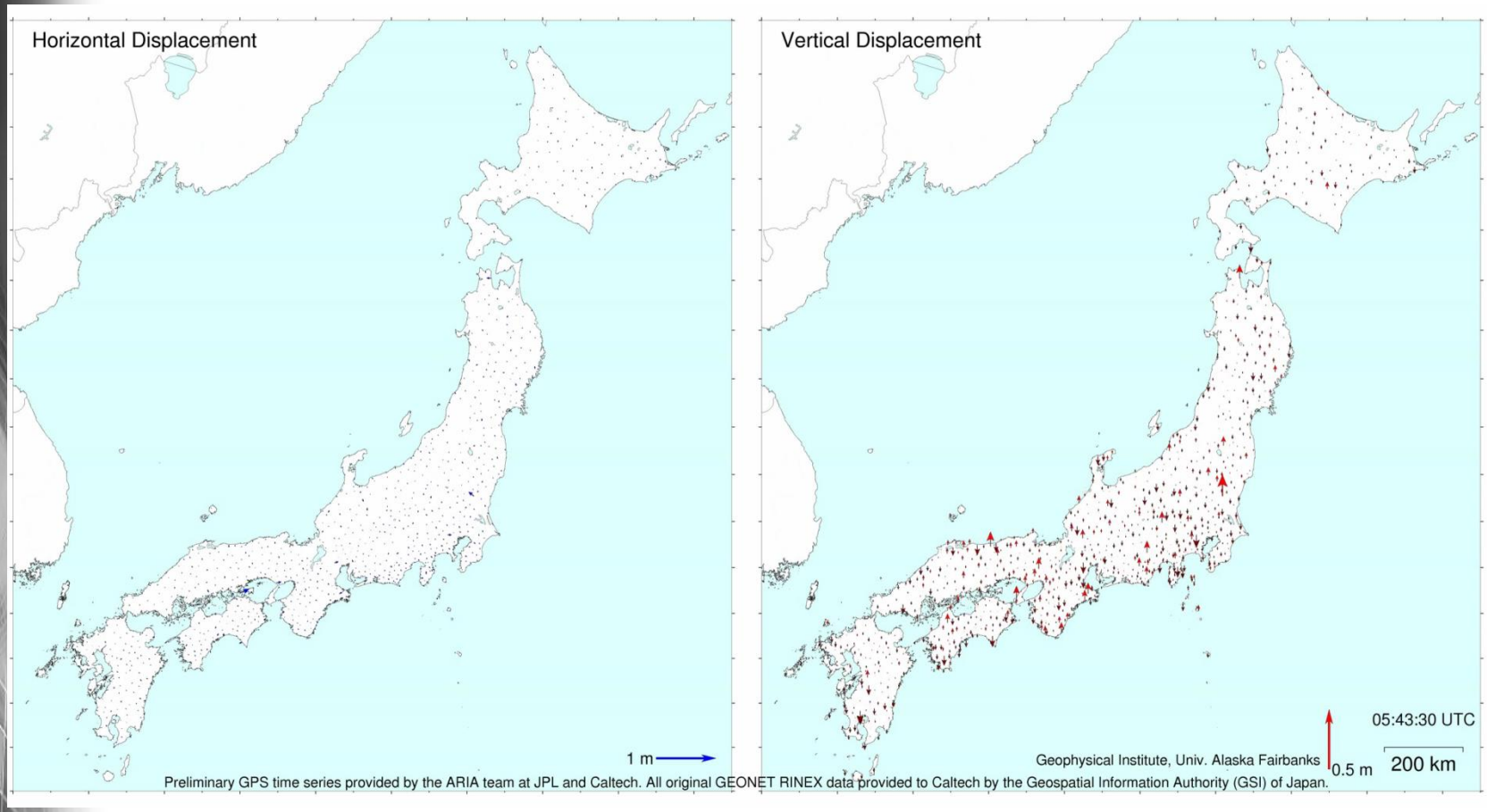


Instr. Kinemetrics Altus K2

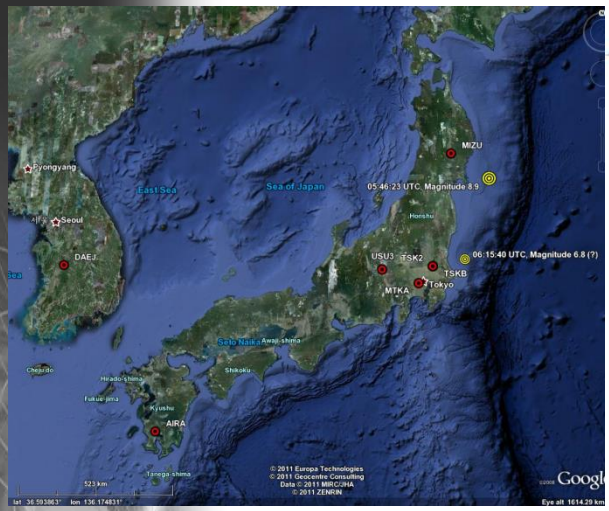


Mw 5.4 aftershock of 9 April 2009 (00:53 UTC)

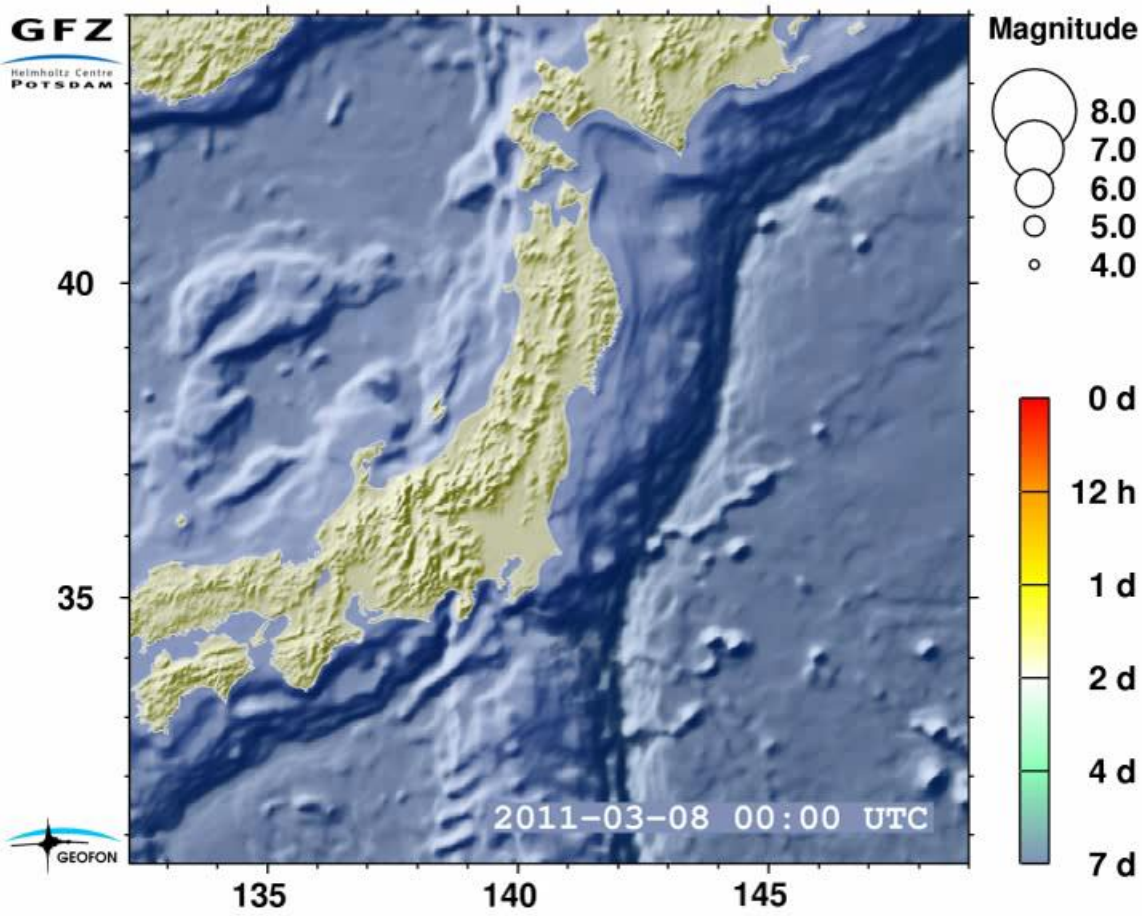
Bedeutung von Animationen: *Visualisierung von Erdbewegungen*

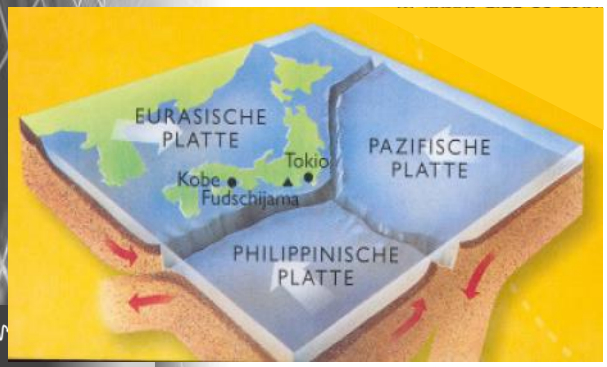


Experimentauswertung: Bedeutung von Visualisierung



hunderte Nachbeben, über einen ganzen Monat verteilt





1. Einführung

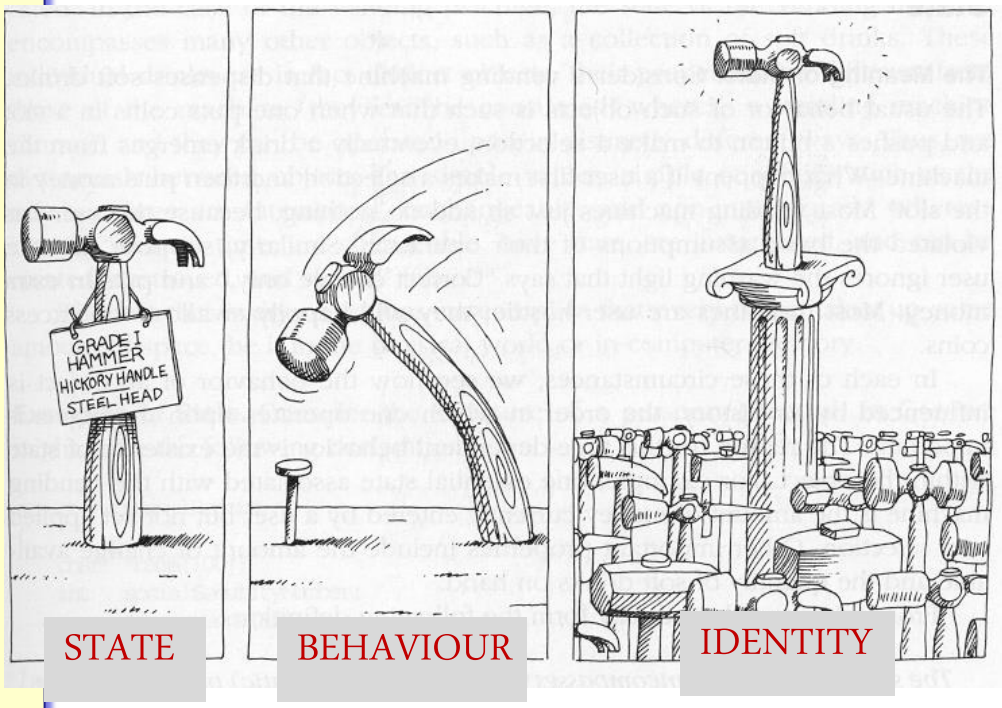
1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Klassifikation dynamischer Systeme
8. Scheduler für zeitdiskrete und zeitkontinuierliche Systemmodelle
9. M&S eines Niedertemperaturofens

Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassen (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen / allg. Classifier
4. Identifikation verschiedenster Beziehungen zwischen Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen

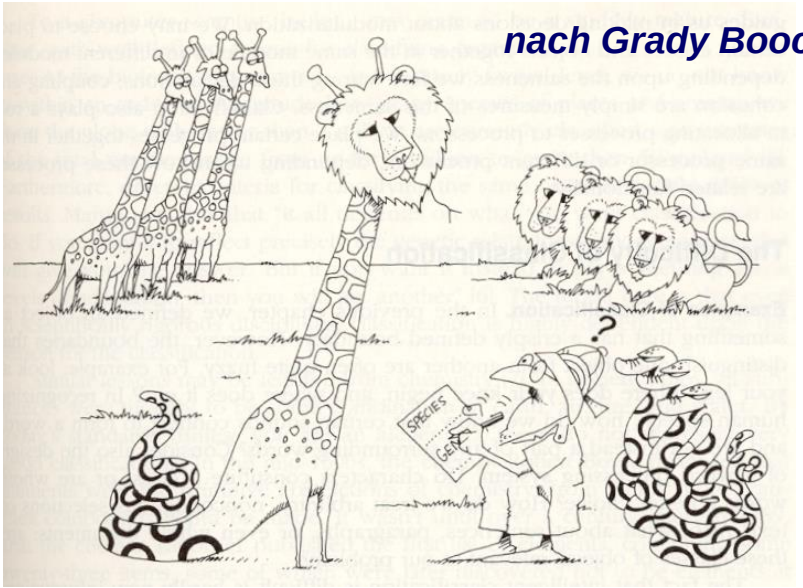
nach Grady Booch



Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen



nach Grady Booch

KLASSIFIKATION VON OBJEKTEN

Definition benötigter Objektklassen
statt
Definition einzelner Objekte

PRINZIP DER DATENKapslung

Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen

OBJEKTE AKTIVER Klassen

~ Vorgänge in RAUM und Zeit

… agieren eigenständig
und in Kooperation miteinander
bei Austausch
von Informationen /Daten
(synchron /asynchron)
statische oder dynamische Existenz

OBJEKTE Passiver Klassen

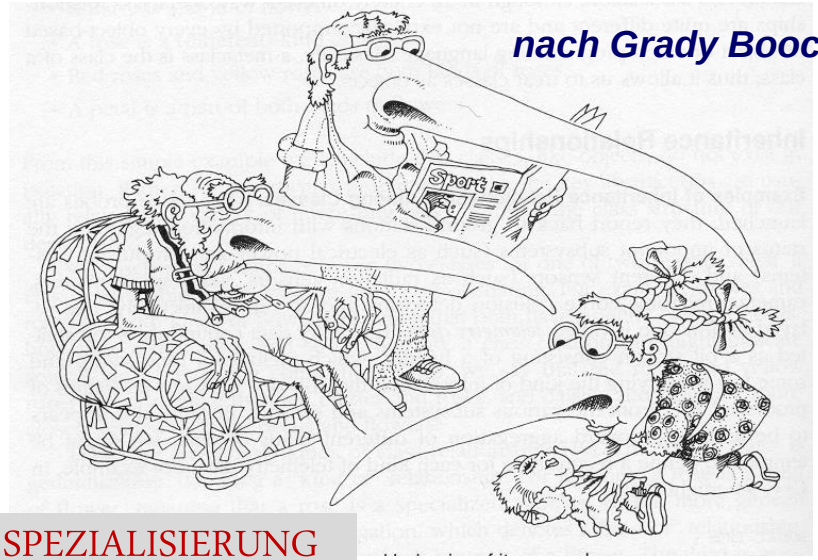
… werden von Objekten
aktiver Klassen benutzt

***Bsp**: ein Thread führt Operationen über Objekte von Java-Klassen aus*

Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - ...
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen



nach Grady Booch

SPEZIALISIERUNG

bedeutet
Wiederverwendung der
gemeinsamen Basiskonzepte
- Attribute
- Verhalten / Methoden

Modellierungsparadigmen

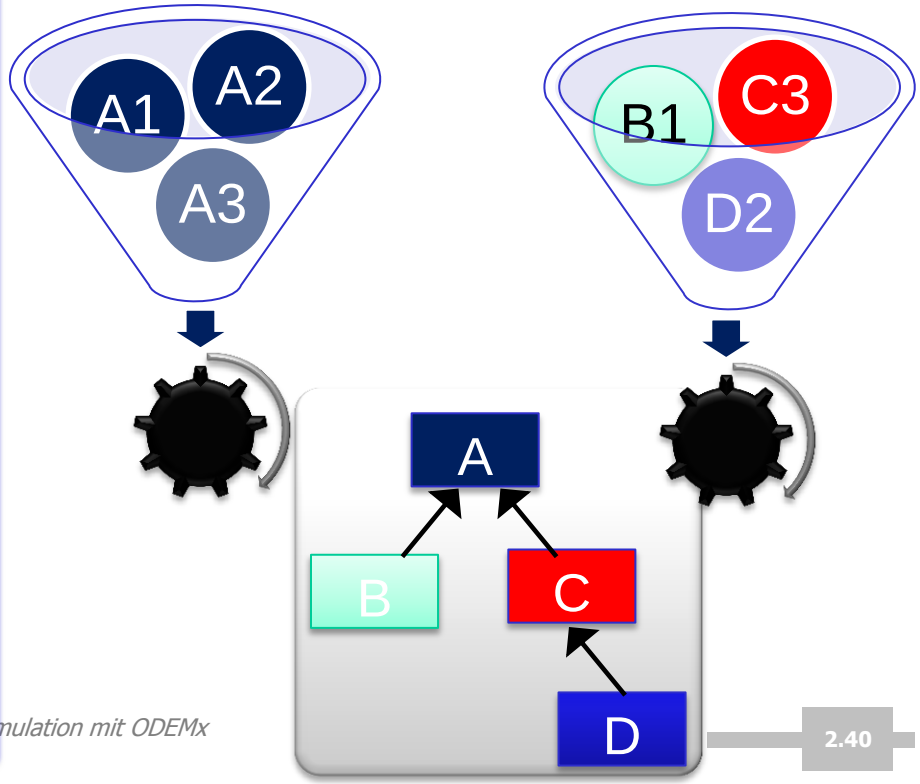
Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen

Vielgestaltigkeit

überall dort, wo die Verarbeitung von Objekten einer (Basis-) Klasse definiert ist,

lässt sich auch eine Verarbeitung von Spezialisierungen organisieren



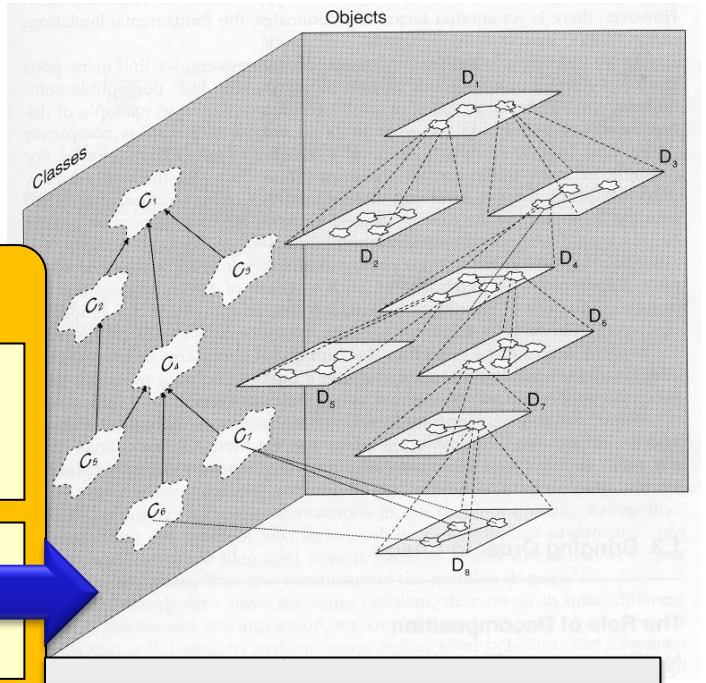
Modellierungsparadigmen

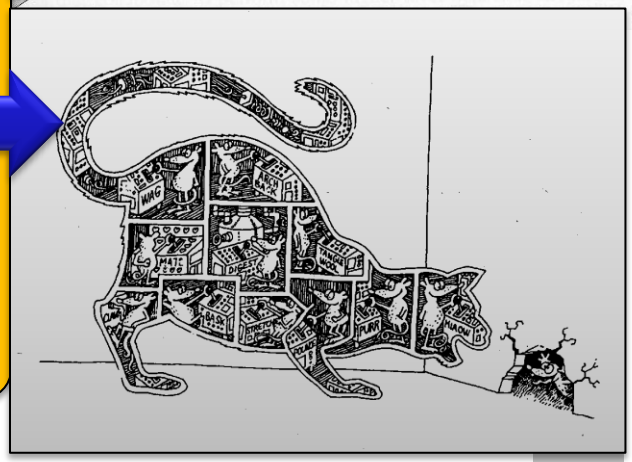
Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekt
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifiziere
6. Polymorphie von (getypten) Referenzen

zusätzlich …

- Gruppierung von Modellelementen
- Komposition/ Dekomposition
- Nebenläufigkeit/ Parallelität/ Synchronisation
- zeitdiskretes/ zeitkontinuierliches Verhalten





Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken

Hoare-Logic

Hoare Milner Jones

CSP **CCS** **VDM**

OO- Konzepte, Modellierung

Programm.

FORTRAN

Algol **Pascal**

C

Management

COBOL

ER-model

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Konzeptuelle Basis von OO:
- Klassen mit Vererbung,
- Objektreferenzen,
- Polymorphie, Virtualität,
- Co-Routinen,
- Garbage Collection
- Standardklassenbibliothek

Simula als erste OO-Sprache
- Implementationssprache
- Modellierungssprache
- Byte-Code

Xerox PARC
SmallTalk (Kay)

Apple
MacInt

OOA(Yourdon)

OMT
(Rumbaugh)

Booch

UML 1.x (Rational/OMG)

Sun
JAVA

EJB

Web services

MSC-2000 (ITU)

ASM (Gurevich)

UML 2.0 (OMG)

Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken

Hoare-Logic

Hoare **CSP** Milner **CCS** Jones **VDM**

OO- Konzepte, Modellierung

Programm.

Management

effiziente Programmausführung
(explizite Speicherverwaltung)
C with Classes

Norwegian Computing C
SIMULA
(Nygaard, Dahl)

Konzeptuelle Basis von OO:
- Klassen mit Vererbung,
- Polymorphie, Virtualität,
- Co-Routinen
- Garbage Collection

Xerox PARC
SmallTalk (Kay)

Bell Labs
C++

OODB

SQL

Apple
MacIntosh

Corba

OOA(Yourdon)

Broy/Stølen
Focus

SDL-92 (ITU)

OMT
(Rumbaugh)

Objectory
(Jacobsson) **Booch**

SDL-2000 (ITU)

ROOM
(Objectime)

MSC-92 (ITU)

UML 1.x (Rational/OMG)

MSC-2000 (ITU)

EJB

Web services

ASM (Gurevich)

UML 2.0 (OMG)