Grundzüge der Wirtschaftsinformatik
Introduction to Business Information Systems

Unit 7

Prof. Dr. Martin Hepp
http://www.heppnetz.de
mhepp@computer.org

http://www.heppnetz.de/teaching/gwi/

Structure of the Lecture

Unit 1: Introduction
Unit 2: Central Processing Units
Unit 3: Storage and Data Structures
Unit 4: Input and Output Devices
Unit 5: Software
Unit 6: Networks, Data Interchange, and the Internet
Unit 7: Design, Development, Deployment, and Operations of Information Systems
Unit 8: Office Applications
Unit 9: Enterprise Applications
Unit 10: Supply Chain Applications and E-Business
Unit 11: Management Support Systems
Unit 12: Exam Review

Assignment from last week

- WI1, pp. 151-322; IBIS, pp. 169-194
- Review the slides

WI1 = Hansen/Neumann: Wirtschaftsinformatik 1; WI2 = Hansen/Neumann: Wirtschaftsinformatik 2; IBIS = Wigand et al: Introduction to Business Information Systems.

Link to the Previous Unit

- Last Unit:
  - How can one computer send data and instructions to another computer?
  - How can data be transmitted over wires, radio communication, or fiber optic cables?
  - How do the Internet and its services work?
- Today:
  - Which security problems exist in networks, and what can we do to mitigate them?
  - Which methods and tools exist for designing and developing software for business problems?
  - Which notations exists for representing data structures and program execution?

Network Security
Threats and Protection

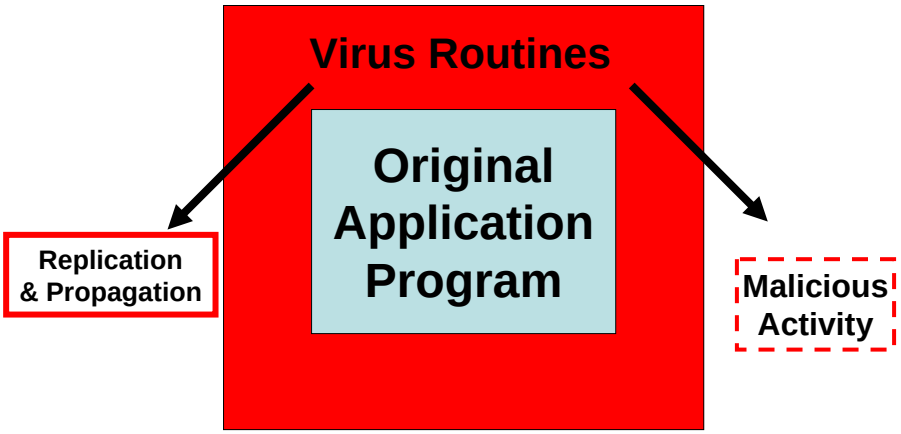
Your Data is at Risk

- Loss
- Manipulation and Corruption
- Unauthorized Access and Usage
- Abuse of your computer for attacking others

Malware (Malicious Software)

- Computer Viruses
- Trojan Horses
  - Password sniffer, Keyboard logger
  - FTP clients
- Worms
- (Advertising) Spyware

Computer Virus

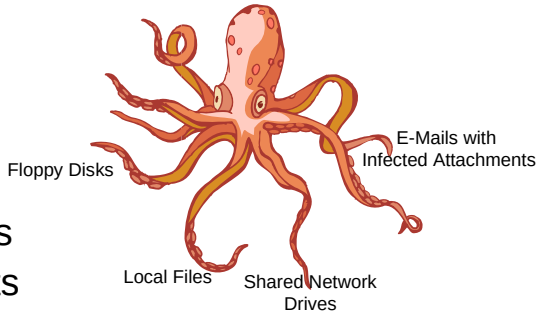


Trojan Horses

Program that appears to be a useful tool but secretly performs malicious activities

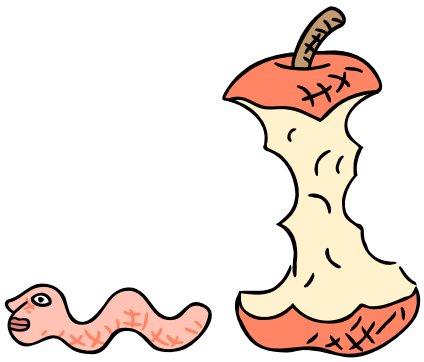
How Malware Propagates

- Media exchange (floppy disk, ZIP, CDR, memory stick,…)
- E-mail attachments
- Infected documents
- Network drives
- Security leaks in the Operating System or Application software



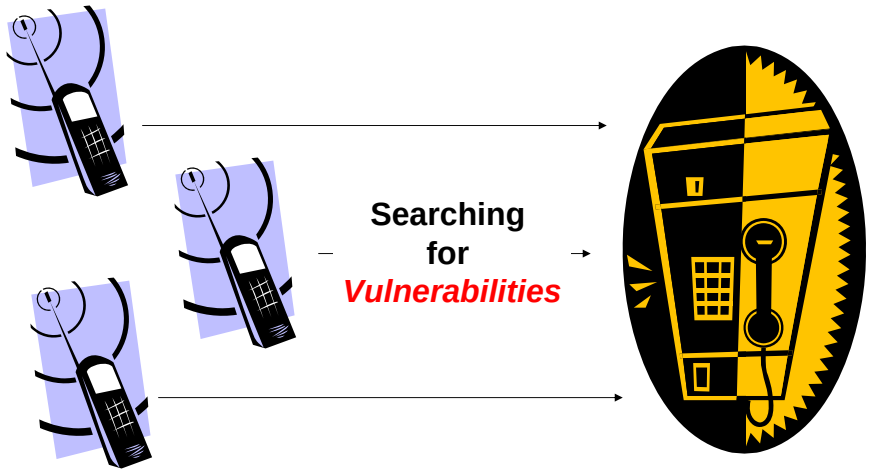
Computer Worm

A self-replicating computer program. Computer viruses attach themselves to, and becomes part of, another executable program; a worm is self-contained and does not need to be part of another program to propagate itself.

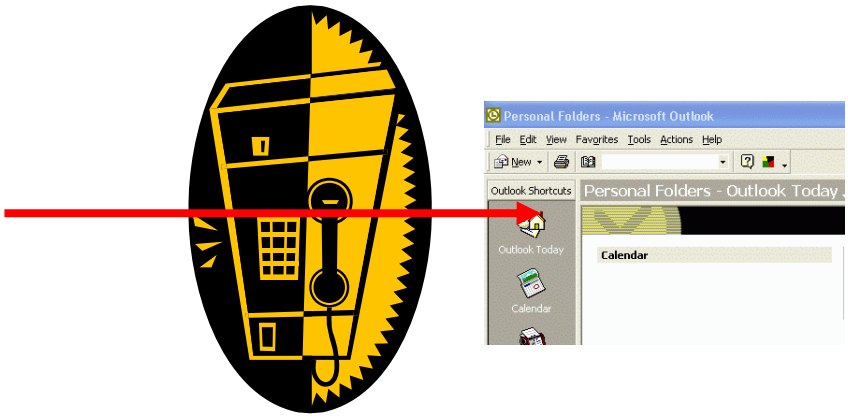


(http://en.wikipedia.org/wiki/Computer_worm)

Malicious Internet Traffic



Security Leaks in Software



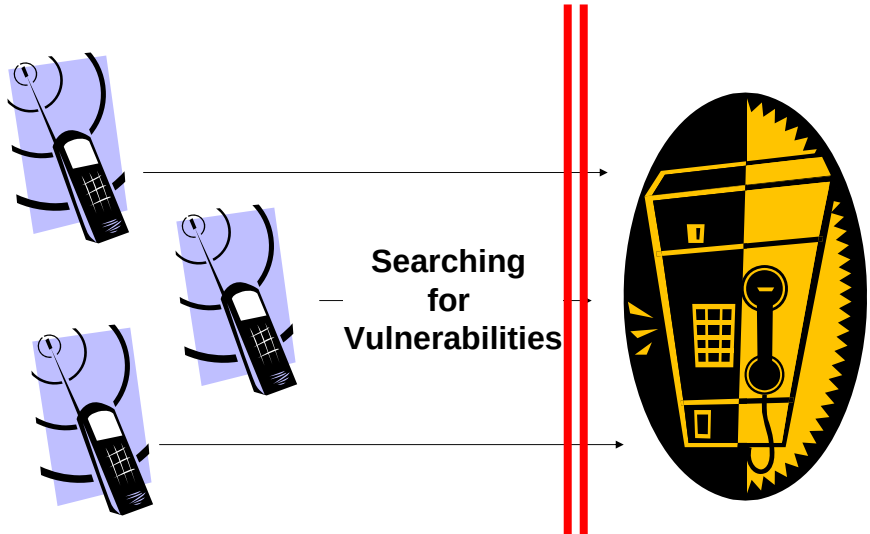
Spyware

- Collect information from your computer and transmit it, without telling you, to a remote computer
- May steal credit card details, passwords, …

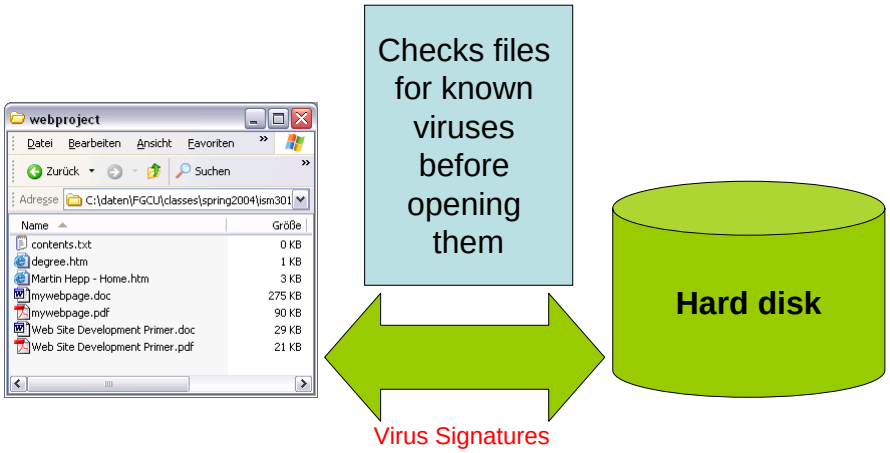
Protect Yourself!

- Backups
- Firewall
- Anti-Virus Software
- Software Updates
- Behavior

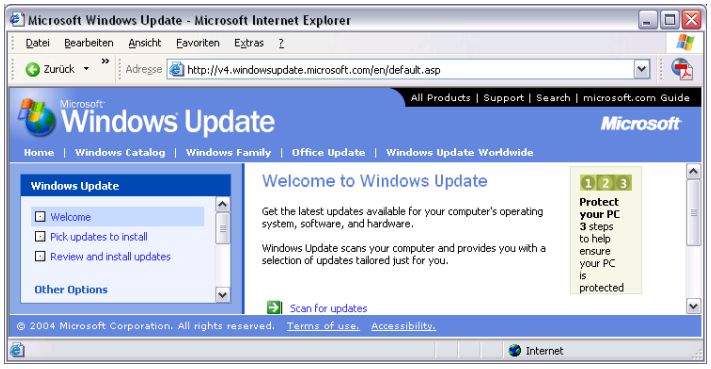
Firewall



Virus Scanner



Updates for Windows and MS Office



http://windowsupdate.microsoft.com

http://office.microsoft.com/productupdates

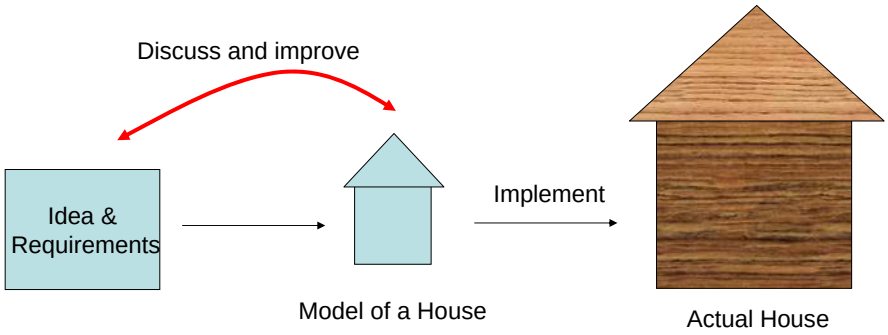
Hoaxes

- Malicious Code (Virus and Trojan) Warnings
- Urban Myths
- Sympathy Letters and Requests to Help Someone
- Traditional Chain Letters
- Threat Chains

Information about hoaxes:
http://hoaxbusters.ciac.org

Design, Development, Deployment, and Operations of Information Systems

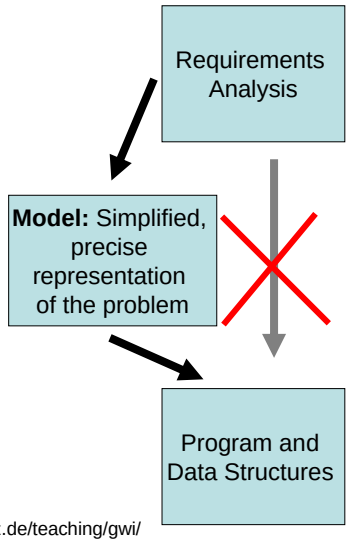
Architects build models first ...



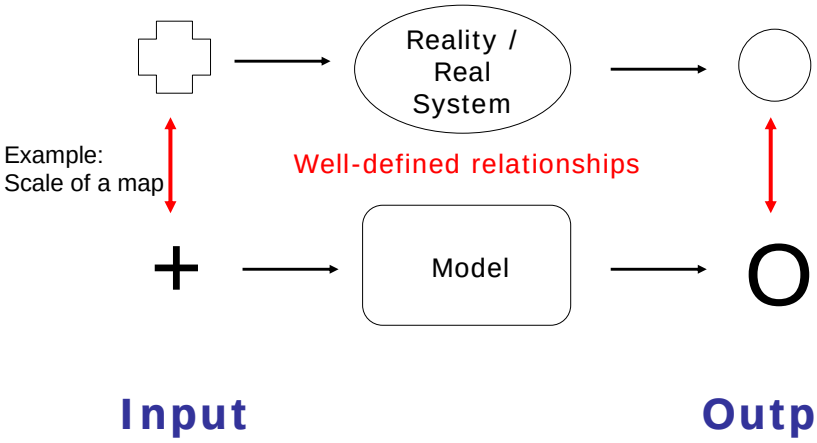
Bottomline: For complex engineering challenges, it makes sense to first create a model of the solution before implementing it.
Studying the model is cheaper and allows for earlier feedback.

Modeling in Information Systems: Foundations

- Same as an architect builds a small-scale model first before actually building a house, we first create models before developing software
- Models can be used to discuss and validate the appropriateness of the solution at lesser costs.
- Absolutely necessary for complex problems.

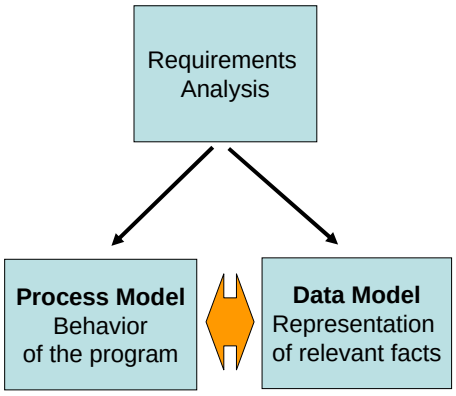


Model



Modeling: Foundations

- **Example: Credit Approval**
  - We want to develop a program that checks whether a bank should approve a credit application.
- Principle:
  - *Check whether all current accounts are within their agreed limits*
  - *Check whether all agreed limits plus the new credit does not exceed the bank's internal limit for that customer.*

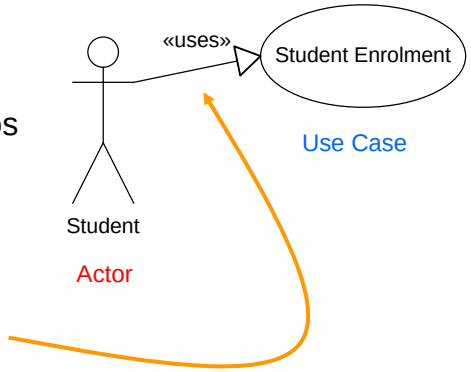


Modeling: Most Important Perspectives

- **Use Case:** How will the system be typically used?
  - Example: *„Bank clerk enters credit application“*
- **Control flow:** What is the proper order of activities in a process?
  - Example: *Before a credit can be approved, the current credits must be checked.*
- **Data:** Which elements and attributes are relevant?
  - Example: *We need to store customers and their addresses in the system.*

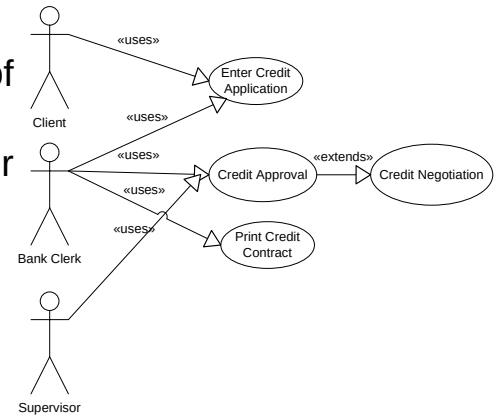
UML Use Case Models

- Diagram type for modeling
  - Use cases (scenarios of usage of the final system)
  - Actors (individuals using the system)
  - Relationships and dependencies between actors and use cases



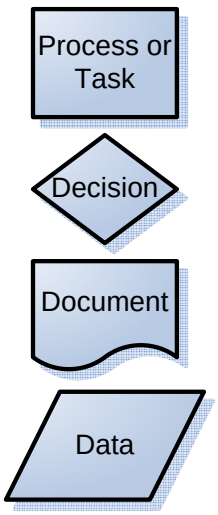
UML Use Case Models

- **Use cases:** Scenarios of usage of the final system
- **Actors:** Individuals or roles using the system
- Relationships and dependencies between actors and use cases



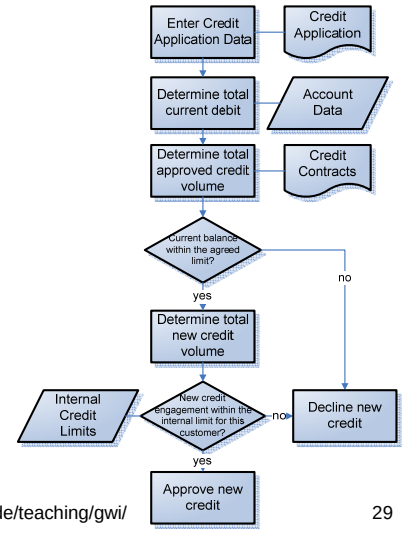
Modeling Control Flows: Basic Flow Chart

- **Process or Task:** Single Activity or Sub-Process
- **Decision:** Decide upon alternative next tasks
- **Document or Data:** Access to this document or type of data is required in a task or decision



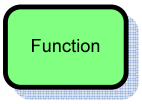
Basic Control Flows: Example

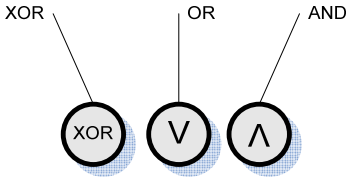
- *Check whether all current accounts are within their agreed limits*
- *Check whether all agreed limits plus the new credit does not exceed the bank's internal limit for that customer.*



Modeling Control Flows: Event-driven Process Chains (EPCs)

- Events: Changes in the state of the world
  - Example: Arrival of a client, update of a dataset
- Functions: Activity
  - Example: Check credit status, compute interest, ...
- Logical Connectors: Determine conditions between events and following functions
  - AND: All must apply
  - OR: One or multiple
  - XOR: Exactly one

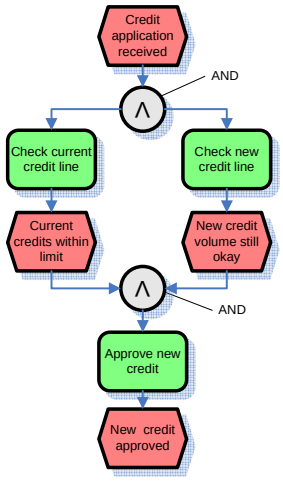




cf. Keller/Scheer/Nüttgens (1992)

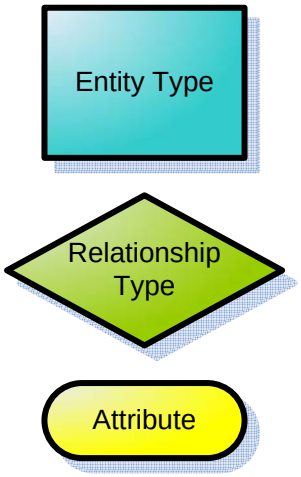
EPC: Example

- *Check whether all current accounts are within their agreed limits*
- *Check whether all agreed limits plus the new credit does not exceed the bank‘s internal limit for that customer.*



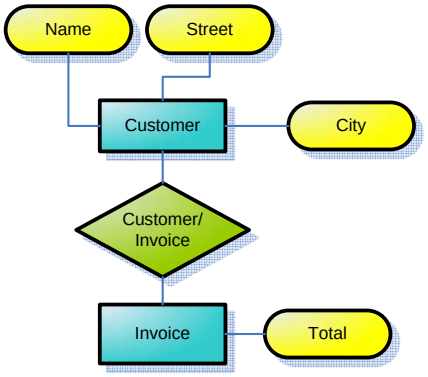
Modeling Data Structures: Entity-Relationship Modeling

- **Entity Type:** A set of objects of the same kind
  - Examples: Customer, Invoice, Student, Contract
- **Relationship Type:** A set of links of the same type that exist between entity types
  - Example: isMarriedTo
- **Attribute:** A relevant property of an entity or relationship type
  - Example: Name of a customer, date of a wedding, ...

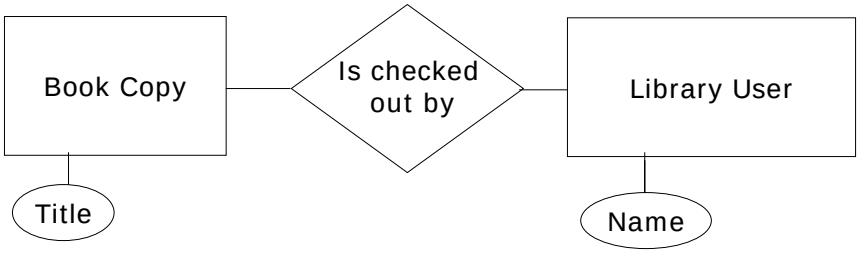


Modeling Data Structures: Entity-Relationship Modeling

- **Entity Type:** A set of objects of the same kind
  - Examples: Customer, Invoice, Student, Contract
- **Relationship Type:** A set of links of the same type that exist between entity types
  - Example: isMarriedTo
- **Attribute:** A relevant property of an entity or relationship type
  - Example: Name of a customer, date of a wedding, ...



Entity-Relationship Model: A very simple library

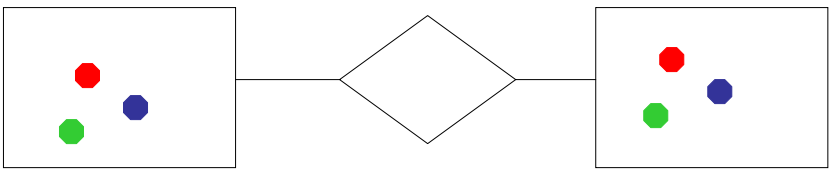


Entity vs. Entity Type

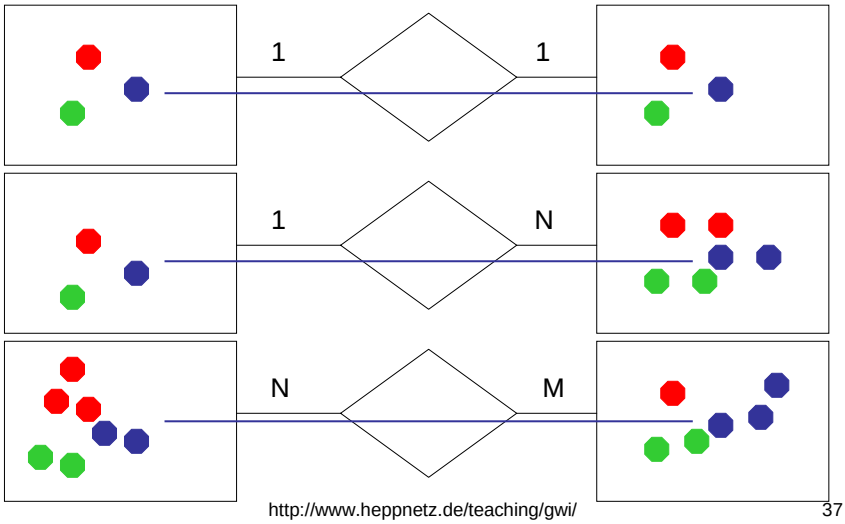
- Entity = an object in reality
  - A specific order, a particular staff member,...
- Entity type: Set of entities of the same type
  - „order“, „staff member“
- Similar for relationship vs. relationship type
- In an entity-relationship model, the symbols represent entity types and relationship types.

Cardinality Constraints

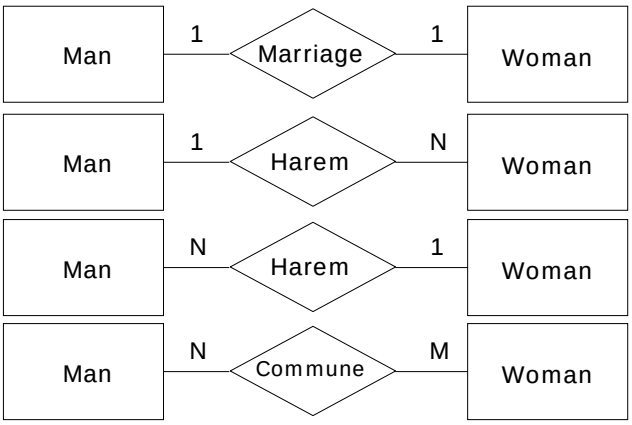
- One can specify quantitative constraints on relationships between entity types.
- Example: *„Each client must have exactly one credit agreement.“*



Cardinality Constraints: 3 Types

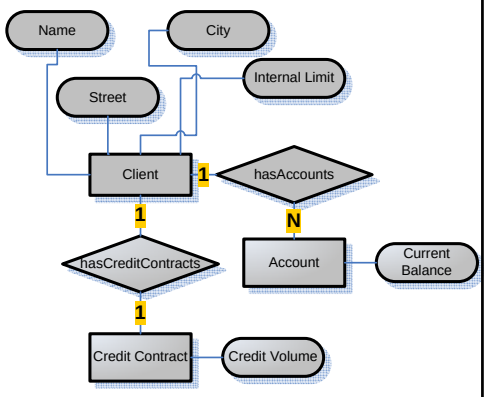


Cardinality Constraints: Example



Entity-Relationship Modeling: Example with Cardinality Constraints

- For a credit approval system, we need to store
  - Clients
  - Accounts
  - Credit Contracts
- It must be represented
  - Which account belongs to which client
  - Which credit contract belongs to which client



Software Engineering Approaches

- Sequential Approach: Waterfall Model
- Iterative Approach: Spiral Model
- Rapid Prototyping
- Extreme Programming

Assignment for Next Week

- WI1, pp. 323-523
- Review the slides

WI1 = Hansen/Neumann: Wirtschaftsinformatik 1; WI2 = Hansen/Neumann: Wirtschaftsinformatik 2; IBIS = Wigand et al: Introduction to Business Information Systems.

Thank you!

The slides and additional materials will be available at
http://www.heppnetz.de/teaching/gwi/