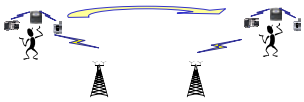
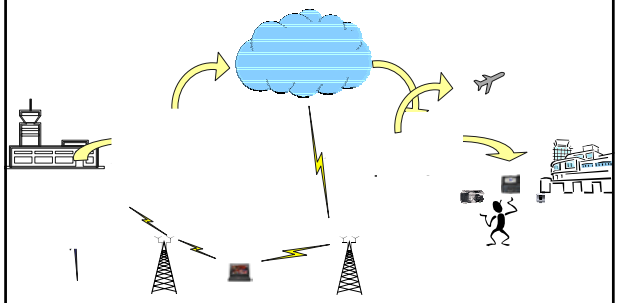


UML for Global Computing

Martin Wirsing
LMU München
in cooperation with
Hubert Baumeister, Piotr Kosiuczenko, Nora Koch
Stephan Merz, Julia Zappe

Travelling Reporter



Global Computing

„massive networked infrastructure composed of highly diversed interconnected objects"

EU Initiative „Global Computing"

⇒ Systems which support
dynamic configuration
complex topologies
autonomous and mobile objects

Mobile Computing (Mobile Hardware)
Mobile Computation (Mobile Code)

Current ...

- Software Engineering Standards (UML)
  - support only particular features of GC such as concurrency and real time
- Programming Languages (C++, Java, ML, Haskell, ...)
  - support only particular features of GC such as concurrency and distribution
- Middleware (CORBA, Jini, COM, SOAP, .Net, ...)
  - improves on client server model

  but
  - are based on simple topology assumptions (e.g. TCP)
  - support only particular aspects of GC
  - have no (good) support for validation&verification

Global Computing needs ...

- New support for software development
  - New features for modeling languages
  - Adequate development processes
- New features for programming languages and middleware

and

- Coupling of pragmatic and formal techniques for analysis, validation, verification of GC systems

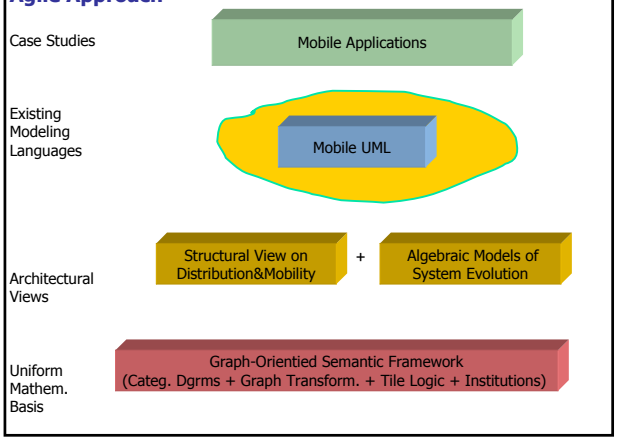
The AGILE Project

- **Goal:**

  Architectural approach to mobile systems development over a uniform mathematical framework supporting
  - sound methodological principles,
  - formal analysis, and
  - refinement across levels of development
- 2002-2005, sponsored by EC Initiative on "Global Computing"
- **Partner:**

  LMU München, ATX Lissabon, U. Pisa, U. Lissabon, U. Florenz, ISTI Pisa, U. Warschau, U. Leicester

Agile Approach



Contents

- **Lecture 1: Introducing UML for Mobility**
  - **Use Case Driven design**
  - **Class Diagrams**
  - **Sequence Diagrams**
  - **Activity Diagrams**
- Lecture 2: Refining Mobility Designs
- Lecture 3: Property-Driven Development of Mobile Systems

Unified Modeling Language

- Graphical modelling language for OO systems
- Means of communication
  - between developers
  - developers and users (?)
- "Unification" of several predecessor methods
- First standardization attempt Sep. 1997 by OMG
- Current version 2.0 (Notation of Lecture 1.5)
- Developed by Booch, Rumbaugh (OMT), Jacobson (OOSE)...
- Consist of:
  - A number of modelling concepts
  - A concrete notation
- Supports "Unified Process" development method

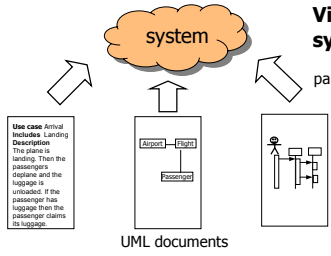
Goals of the UML

- Description of essential program characteristics like a construction plan
- Structuring the problem as well as the solution
- Abstracting from implementation details
- Definition of different views

UML Description Techniques

- Structure:
  - Class diagram
- Interaction, Behaviour:
  - Object diagram
  - Sequence diagram
  - Collaboration
  - Statechart
- Requirements, Functionality:
  - Use Case Diagram
  - Activity Diagram
- Implementation:
  - Component Diagram
  - Deployment Diagram
- Object Constraint Language (OCL)

(UML) Software Engineering Principles



View-oriented system modelling

partial views

UML documents

A system is described by several documents from several viewpoints

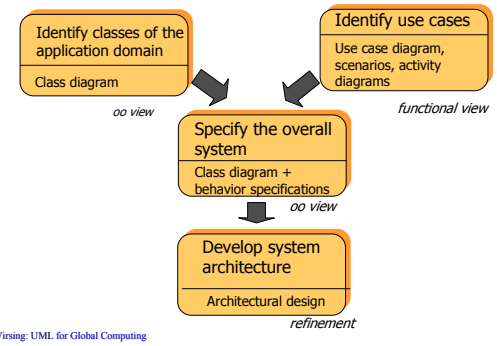
(UML) SE Principles (cont'd)

- Principle of underspecification
- Data encapsulation and
  separation of interfaces and implementations
- Incremental development by refinement steps
- Hierarchical composition and decomposition

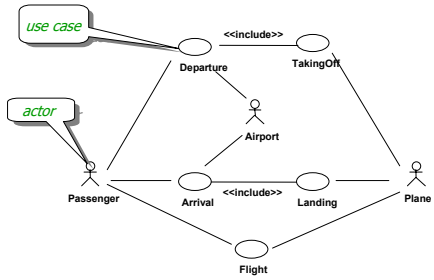
Use Case Driven Design

- **Use Case**
  an interaction between user and system which serves to fulfill a task
  where
  "system" = software to be developed
  "user" = person or external system called "actor"
- Use cases serve as **functional requirement description**
  and provide test cases

Major Development Steps



Use Case Diagram: Example Airport



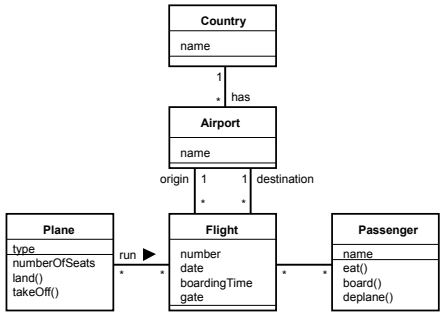
Use Case Description: Example

Use case Arrival
Includes Landing
Actors plane, passenger
Description The plane is landing. Then the passengers deplane and the luggage is unloaded. If the passenger has luggage then the passenger claims its luggage.

Class Diagram

- A **class diagram** defines the static structure of the system
  consisting of
  classes and interfaces
  connected by static relationships such as
  association, aggregation and inheritance

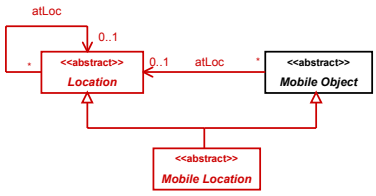
Example: Class Diagram for Airport



Classes for Mobile Objects

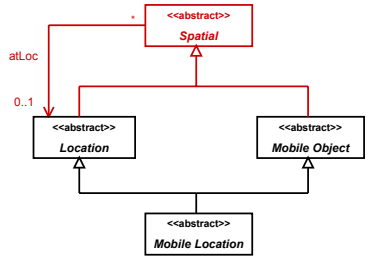
- Goal of the extension
  - Explicit notations for mobile objects and locations
- Extensions
  - UML profile for mobility with stereotypes for locations, mobile objects, and mobile locations

Abstract Classes for Mobility: 1st Approach

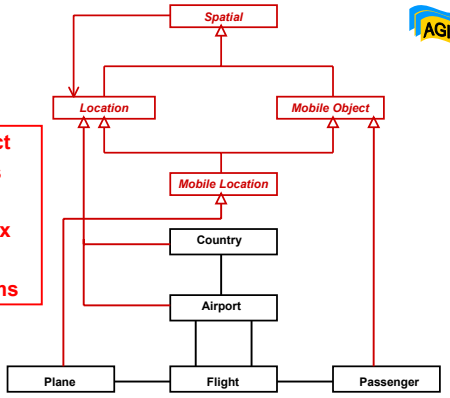


Problem: ***Mobile Location*** inherits „atLoc“ twice:
from ***Location*** and from **Mobile Object**

Abstract Classes for Mobility: 2nd Approach

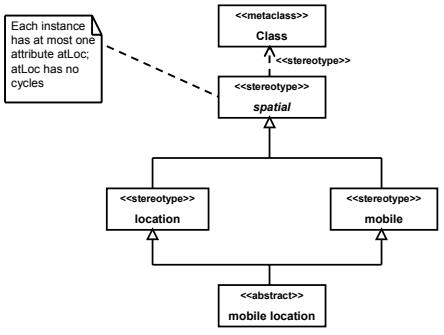


Improvement: Due to the introduction of the abstract class „Spatial“, Mobile Location inherits „atLoc“ only once, **but** ...

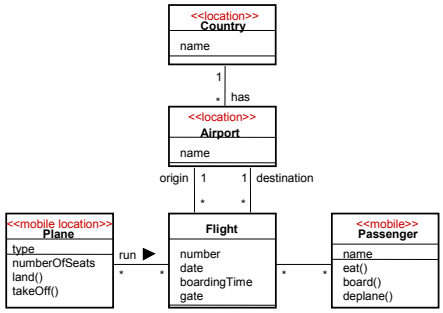


Abstract classes lead to complex class diagrams

Solution: UML Profile for Locations



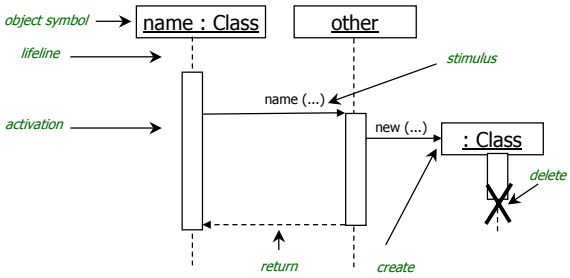
Example: Class Diagram for Airport



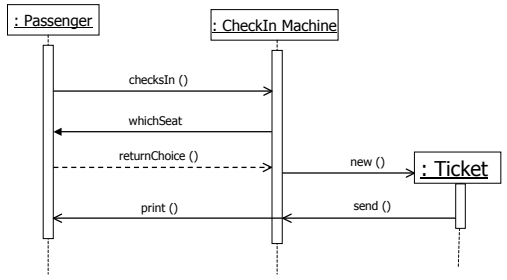
Sequence Diagram

- A sequence diagram describes a **scenario**, i.e. an exemplary message flow where
  - primary scenario: the typical case
  - secondary scenario: variant, exceptional case
- Scenarios can describe **different levels of abstraction**
  - e.g. for use case, components, operations
- Other description technique for scenarios:
  - collaboration diagrams

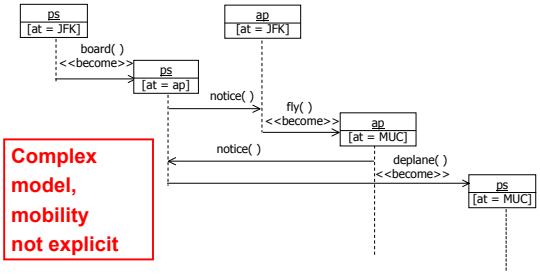
Sequence Diagram



Example: Check In



Mobile runs with SD: A first approach

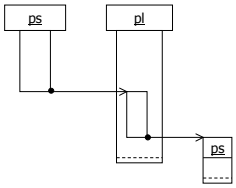


Complex model, mobility not explicit

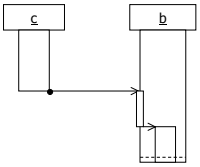
Sequence Diagrams for Mobility

- SDM are an extension of SD for modelling
  - the interaction between objects
  - topology - the containment relation
  - the migration of objects
- **Idea**
  - blow up lifelines and message arrows for modelling topology
  - generalize the concept of lifeline to contain object flow for modelling mobility

Migration of an Object

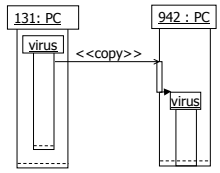


A person ps migrates into the plane pl and leaves it .



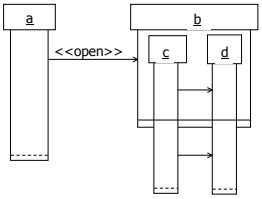
c migrates through a firewall into b and terminates at the same time as b.

Copying Objects

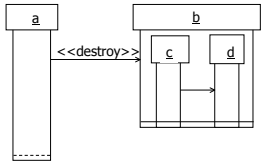


The mobile Object virus in PC 131 is copied;
the copy lifes in PC 942

Opening and Destroying an Object



An „open“ message terminates b; the inner objects c and d continue to life

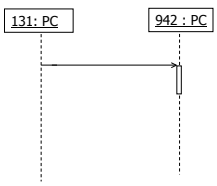


A „destroy“message terminates b, c und d.

Advanced Features

- Zoom-in/zoom-out view
- Matching action boxes
- Lifeline of a migrating object

Zoom-Out View

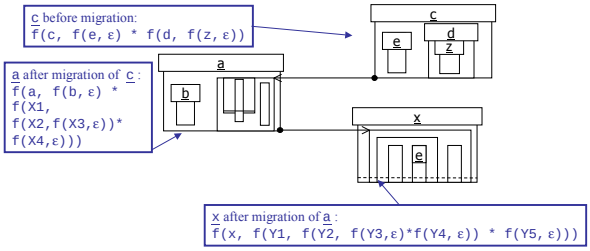


Using Zoom-Out one may hide details of inner objects; only the communication with outer objects remains visible.

Pattern Matching [cf. Maude]

- Representation of mobile objects and object configurations by terms
  - Object constructor: `f : Names x Objects ---> Objects`
  - Configuration constructor `* : Objects x Objects ---> Objects`
  - Empty configuration `ε : ----> Objects`
  - Variables `X: Names`
- Unification modulo commutativity and associativity for computing the names of migrated objects

Pattern Matching Beispiel



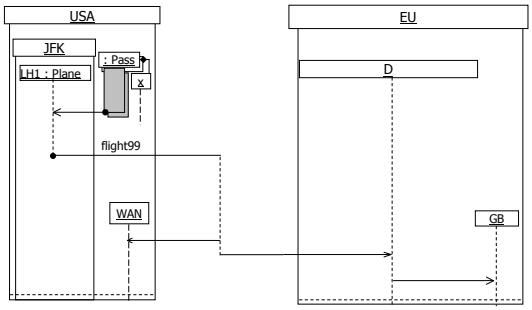
Pattern matching of the 3 terms gives:

{X1 –> c, X2 –> d, X3 –> z, X4 –> e, Y1 –> a, Y2 –> c, Y3 –> z, Y4 –> e, Y5 –> b}

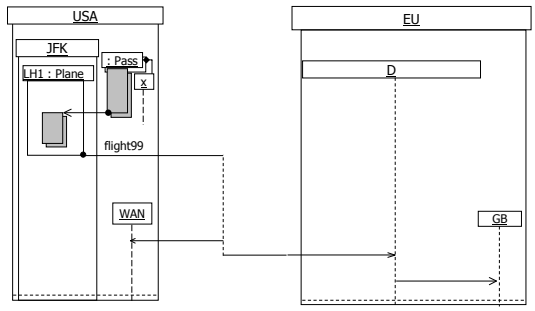
The Traveling Reporter Example

The traveling reporter flies from New York to Munich

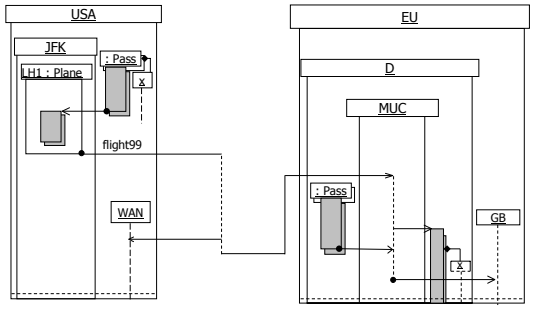
Flight from New York to Munich



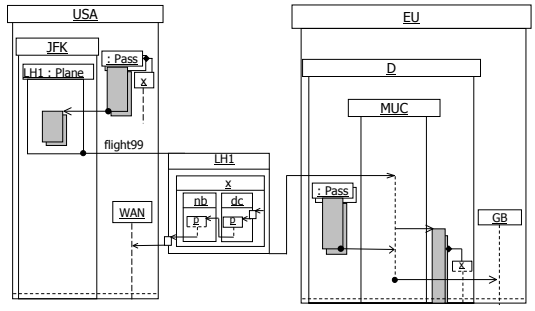
Flight from New York to Munich



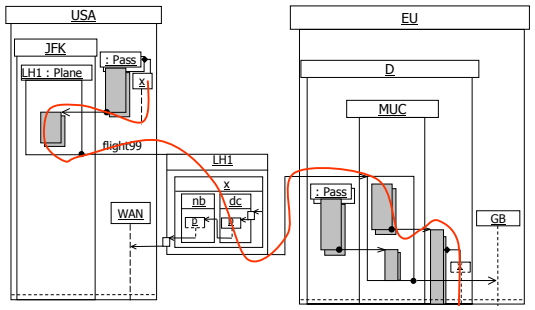
Flight from New York to Munich



Flight from New York to Munich



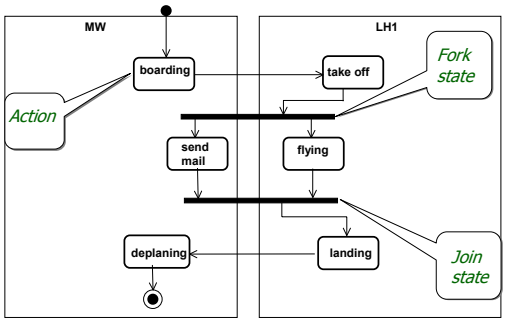
Flight from New York to Munich



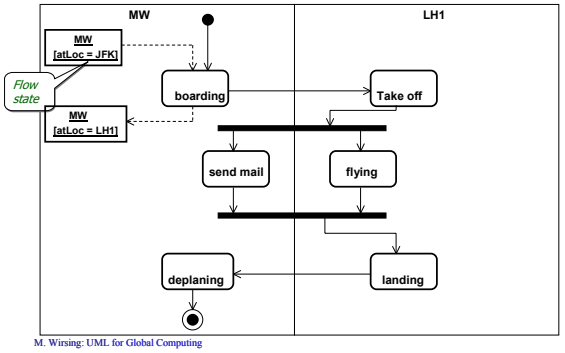
Activity Diagrams

- Activity diagrams are intended for applications that need control flow or object/data flow models
- An activity graph describes a flow of actions which are connected by transitions
- Concurrency is expressed by fork and join states
- Swimlanes represent responsibilities of parts of the activity diagram

Activity Diagram Example: Arrival



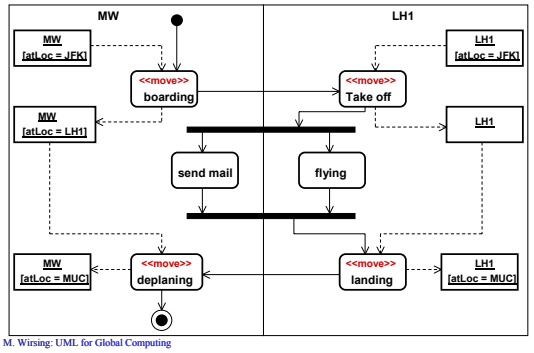
Activity Diagram with Flow States



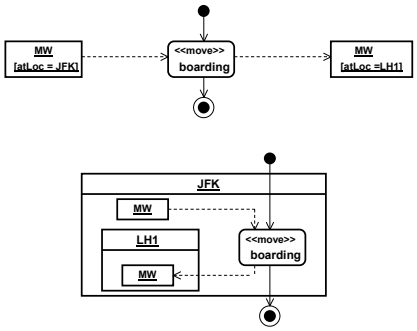
Activity diagrams for Mobility

- Goal of the extension
  - To model how actions change the location of mobile objects
- Extensions
  - Stereotypes <<move>> / <<clone>> for actions
  - Two Notations
    - Responsibility Centred
    - Location Centred

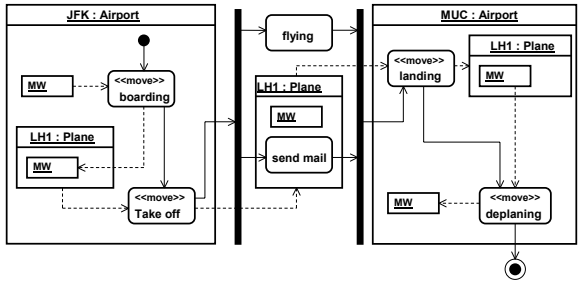
Responsibility Centered View



Move Actions



Modeling the Flight



Summary and Current Work

- Summary
  - First approach for explicit modeling of mobility in UML
  - Simple solution
  - Used already in industry and for teaching (DEGAS-Project, SWE-Praktikum)
- Current Work
  - Semantic Foundation (Translation to GTS and Tiles with Corradini/Montanari, Coordinated Categories with Fiadeiro, Mobile TLA with Merz)