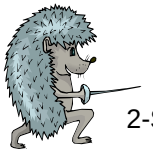
UNIVERSITY
OF OSLO

INF-5150 2005
by Øystein Haugen and Ketil Stølen
plus assistants Atle Refsdal, Gøran Olsen

Version 050902

INF 5150



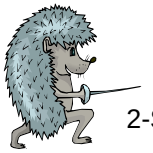
UNIVERSITY OF OSLO

Øystein Haugen <oysteinh@ifi.uio.no>

- 80-81: UiO, Research assistant for Kristen Nygård
  - 81 : IN 105 together with Bjørn Kirkerud
- 81-84: Norwegian Computing Center, Simula-machine
- 84-88: SimTech, typographical applications
- 88-90: ABB Technology, SDL, prototype SDL tool, ATC
- 89-97: SISU project, methodology, V&V, ITU
  - 93: Engineering Real Time Systems
  - 96: Integrated Methodology -> TIMe
- 96-00: Rapporteur ITU for MSC
- 97: Practitioners' verification of SDL systems (dr. scient.)
- 97- 03: Ericsson, NorARC
- 98- 03: Ifi, UiO as Part time Associate Professor
  - IN-TIME (98) IN-RTIMe (99) IN-RTIMe (2000) INFUIT (2001 og 2002)
- 99- : Participates in OMG wrt. UML 2.0
  - 2003: Responsible for UML 2.0 chapter on Interactions
- 04 - : Associate Professor at Ifi

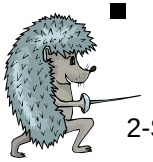
Ketil Stølen <ketil.stolen@sintef.no>

- Leader of Group for Quality and Security Technology at SINTEF
- Professor II at IFI
- Background from University of Manchester (4 years); Technical University Munich (5 years); Institute for Energy Technology (3 years); Norwegian Defence Research Establishment (1 year); SINTEF (5 years)
- PhD in formal methods
- Leading role in the development of the Focus method - a formal method providing the basic foundation for the refinement part of this course
- Lead the development of the CORAS methodology for model-based security analysis providing the basic foundation for the security part of this course
- Is currently managing research projects with a total budget of 35 million NOK



Atle Refsdal <atler@ifi.uio.no>

- Education:
  - Electrical engineer, Gjøvik Ingeniørhøyskole
  - Master thesis in Language, Logic and Information (Språk, Logikk og Informasjon – SLI) at the Faculty of Arts, University of Oslo
- Previous employees:
  - Beijer Electronics, Drammen. Industrial automation/Programmable Logical Controllers
  - Computas AS, Oslo/Lysaker. Knowledge Systems "Saksbehandlingssystemer".
- Currently employed at the Department of Informatics as a PhD student in the SARDAS project
  - where Ketil Stølen, Øystein Haugen, Birger Møller-Pedersen and Rolv Bræk are supervisors
  - we describe/analyze availability using UML or UML-like models
- Took INF-5150 in 2003
- Was assistant to INF-5150 in 2004

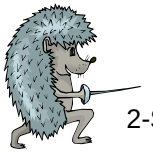


Gøran K. Olsen <gorano@ifi.uio.no>

- Education:
  - Ongoing Master, Object-orientation, modelling and languages.
    - Supervisor Øystein Haugen. Finishes A2005
  - Bachelor IT/Economics Bodø University College

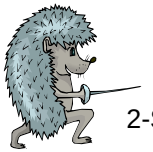
- Previous employees:
  - Ifi, Student assistant:
    - INF-2120
    - INF-1000
  - Bodø University College
    - Algorithms and Data Structures (BUC)
    - Database1 and 2 (BUC)
  - Bjølsen Skole

- Took INF-5150 in 2004



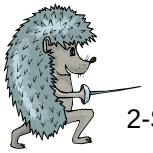
Books and Curriculum

- We will produce and refer to written material to support the lectures
  - UML 2.0 support literature: The UML Reference Manual 2nd edition
  - and chapter from UML for Real
  - Semantics of Interactions: Report on STAIRS
- The slides of the lectures will be made available on the web in Acrobat format
- The lectures are part of the curriculum
- INF 5150 will use the required planning pages which are at: http://www.ifi.uio.no/INF5150/



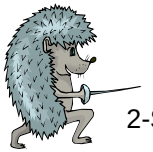
Goal: Unassailable IT-Systems

- The course INF-UIT aims at teaching the students
  - how software is made unassailable meaning that
    - the software is easily analyzed with respect to reliability and dependability
    - the software is easily maintained
- The overall goal is to explain
  - how practical software development can benefit from theories about
    - state machines
    - refinement
    - formal reasoning
    - modularity
    - security and related matters



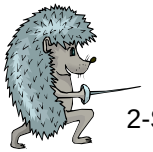
Practical details

- When?
  - Friday 9.15 - 12.00
- Where?
  - Lille Auditorium, Informatikkbygget
    - except for 16. September when it is in Vilhelm Bjerknes Aud. 4
- Language: English
- Exam
  - Credits: 10 studiepoeng
  - Form: written
  - Grades: A - F
- Obligatory Exercises
  - There will be one obligatory exercise done in groups of 5-6
  - The students may be asked to explain details in their solution
  - The obligatory exercise will have two drops



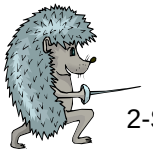
Unassailable IT-Systems

- Unassailable?
- IT?
- Systems?



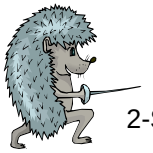
Unassailable

- Not assailable : not liable to doubt, attack, or question
- Where is this important?
  - for all software?
    - to some extent, but possibly less than one would like to think
  - for some critical software
    - telecom
    - surveillance (of patients, of production processes)
    - within computers themselves
- This course is not concerned with attacks that come from hackers towards data bases with sensitive content
  - we are concerned with helping software to perform desirably even in unexpected situations



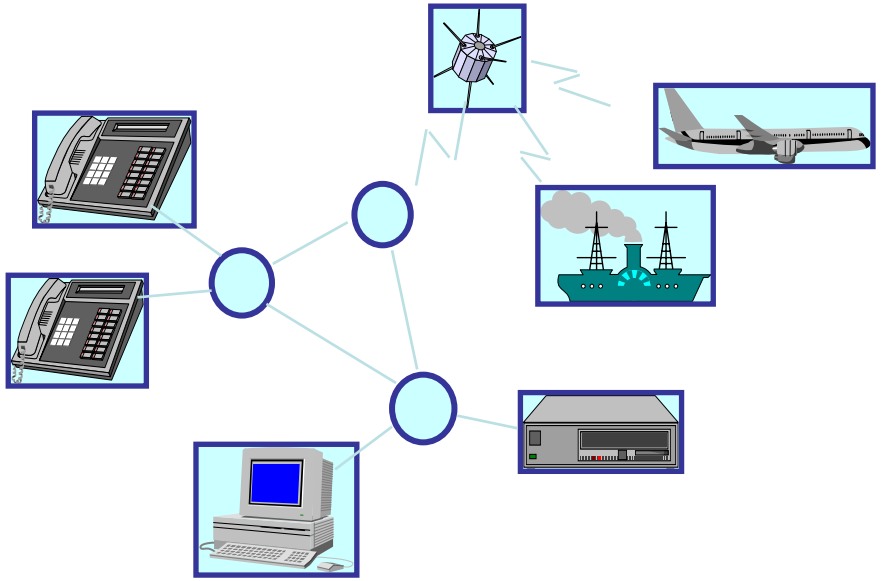
IT?

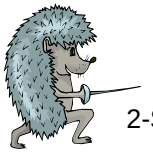
- Information Technology
  - using computers
  - with emphasis on practical systems
  - with emphasis on behavior
- Engineering
  - Well acknowledged and asserted techniques
  - Creativity only when and where needed
  - Replication of earlier efforts
  - Pragmatics as well as theory



Systems?

- distributed
- concurrent
- real-time
  - In synchrony with real life
  - often small amounts of time for each service e.g. Automatic Train Control
  - the actual durations may or may not be significant
- reactive
- heterogeneous
- complex

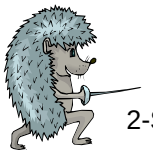




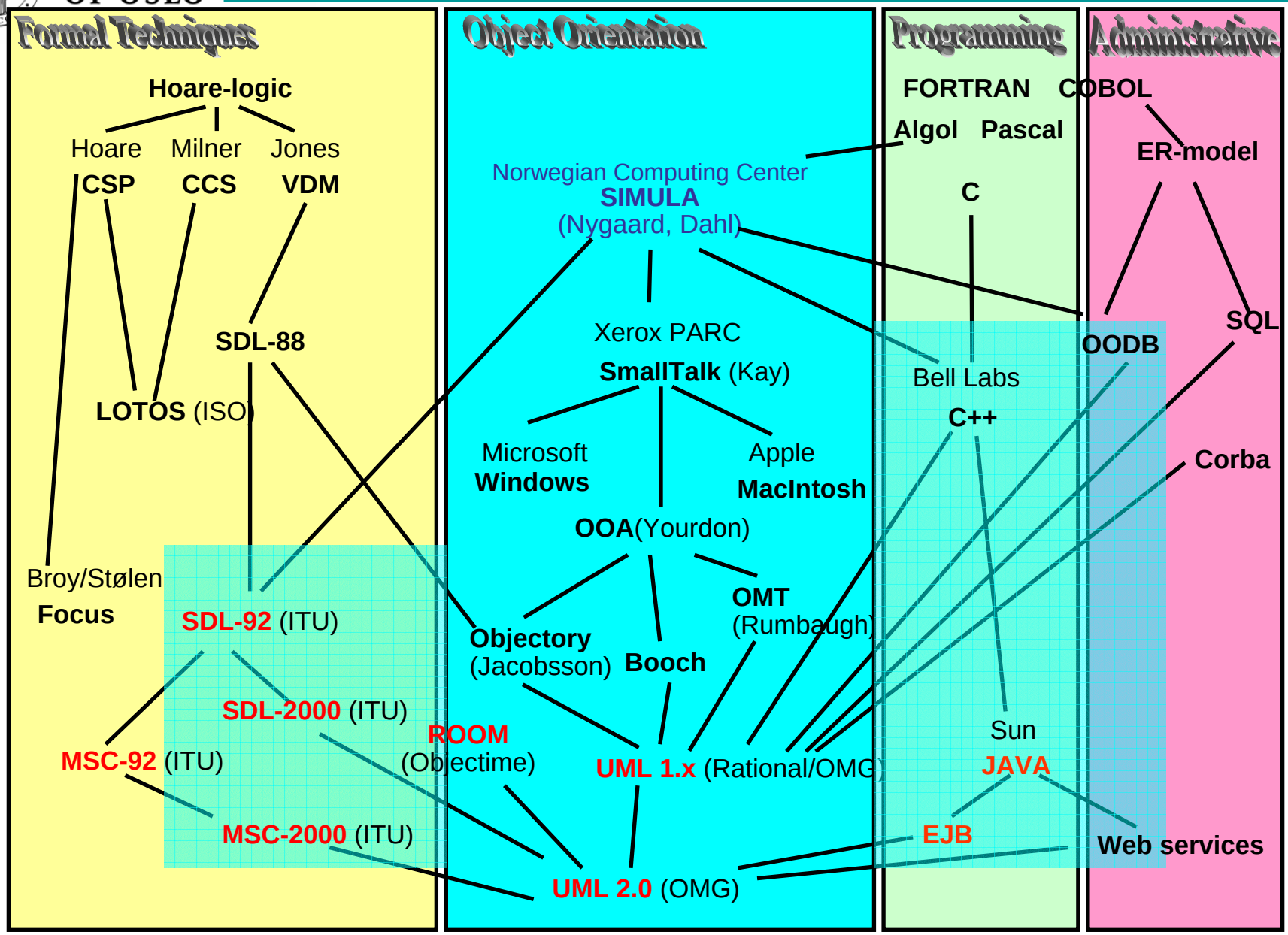
Lecture plan - INF-5150 Autumn 2005

Undervisningsplan (INF5150 - Høst 2005)

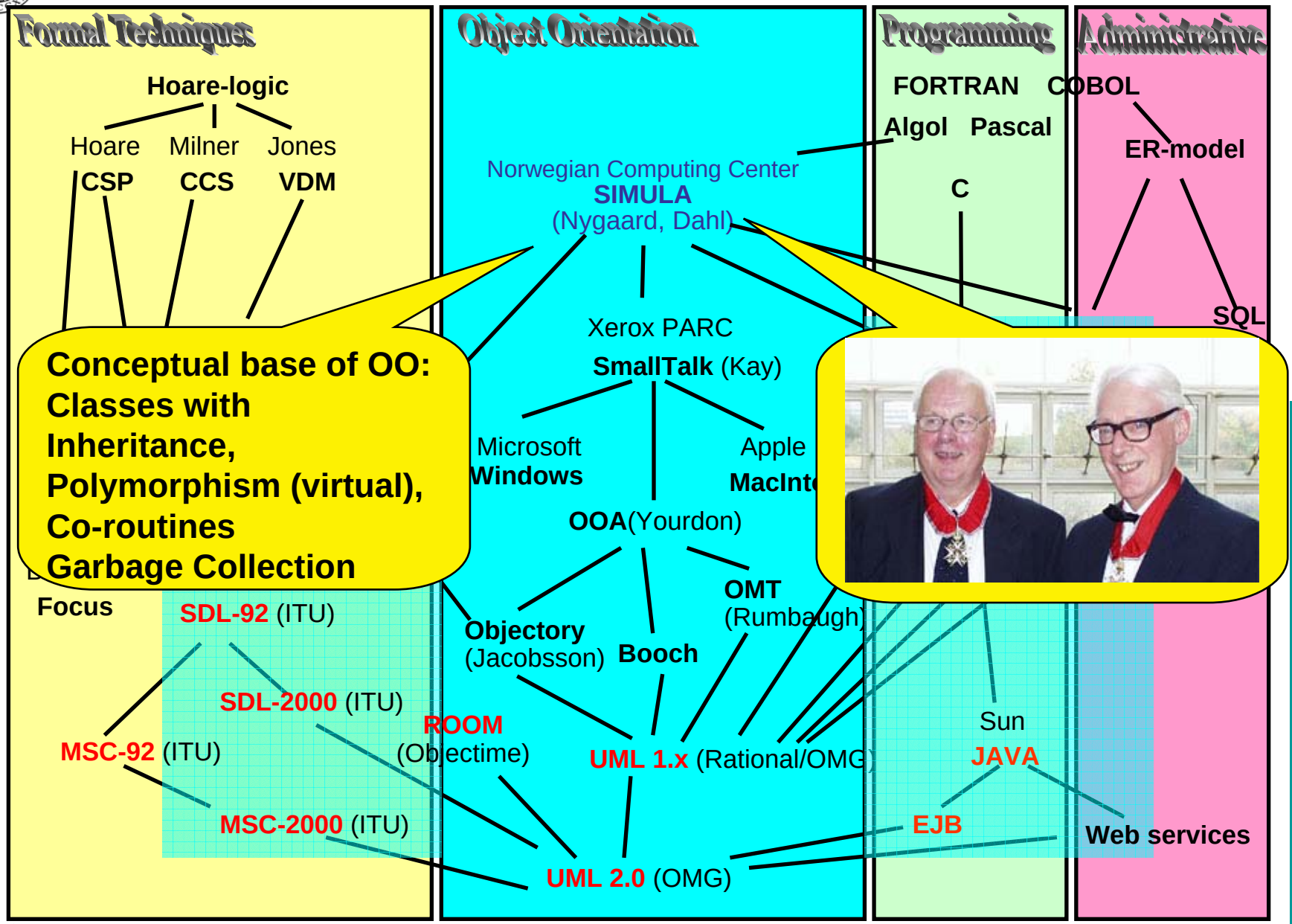
Dato	Undervises av	Sted	Tema	Kommentarer / ressurser
02.09.2005	Haugen	Lille Aud	Introduction	Everyone must attend in order to follow the course. THIS PLAN IS NOT FIXED. There will probably be changes during the semester.
09.09.2005	Haugen	Lille Aud	UML Sequence Diagrams 1	
16.09.2005	Haugen	some other place than Lille Aud	UML Sequence Diagrams 2	
23.09.2005	Haugen	Lille Aud	UML State Machines	
30.09.2005	Stølen	Lille Aud	Refinement 1	
07.10.2005	no lecture		Obligatory exercise Drop 1	Deadline 23.59 sharp!
14.10.2005	Haugen	Lille aud.	Walk-through of Drop 1	The groups will criticize each other. The teacher assistants will also assess the reports, and the lecturers will give a tentative grade publicly. The grade has no formal impact on the final grade.
21.10.2005	Stølen	Lille Aud	Refinement 2	
28.10.2005	Stølen	Lille Aud.	Security analysis 1	
04.11.2005	Stølen	Lille Aud.	Security Analysis 2	
11.11.2005	Haugen	Lille Aud.	Development Methodology	
18.11.2005	Stølen	Lille Aud.	Security Analysis 3 and Obligatory Exercise Drop 2	Deadline 23.59 sharp!
25.11.2005	Stølen / Haugen	Lille Aud.	Walk-through of Drop 2	Same procedure as with Drop 1, but now we also want some of the groups to demonstrate running systems



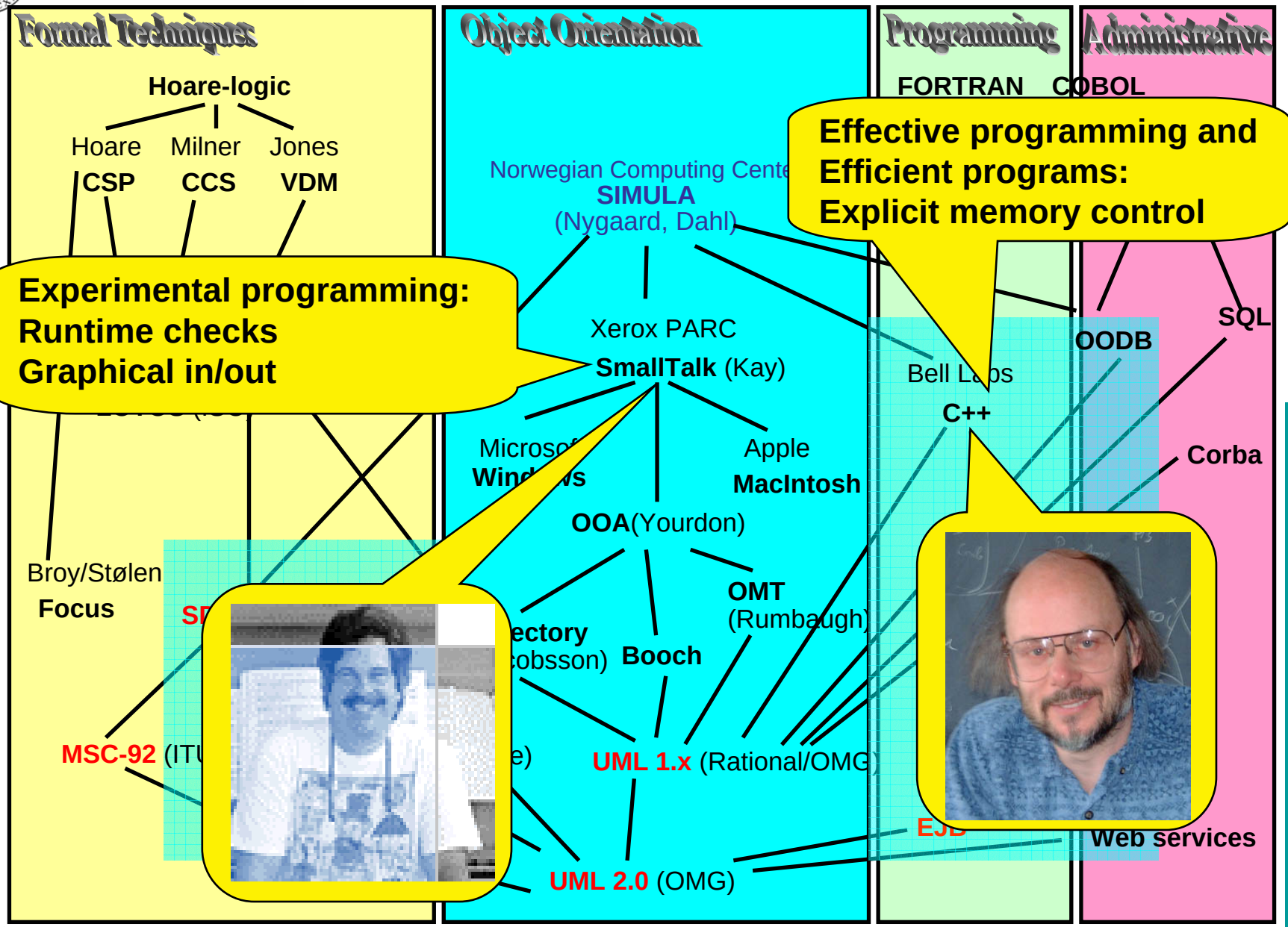
UML 2.0 in the history of languages



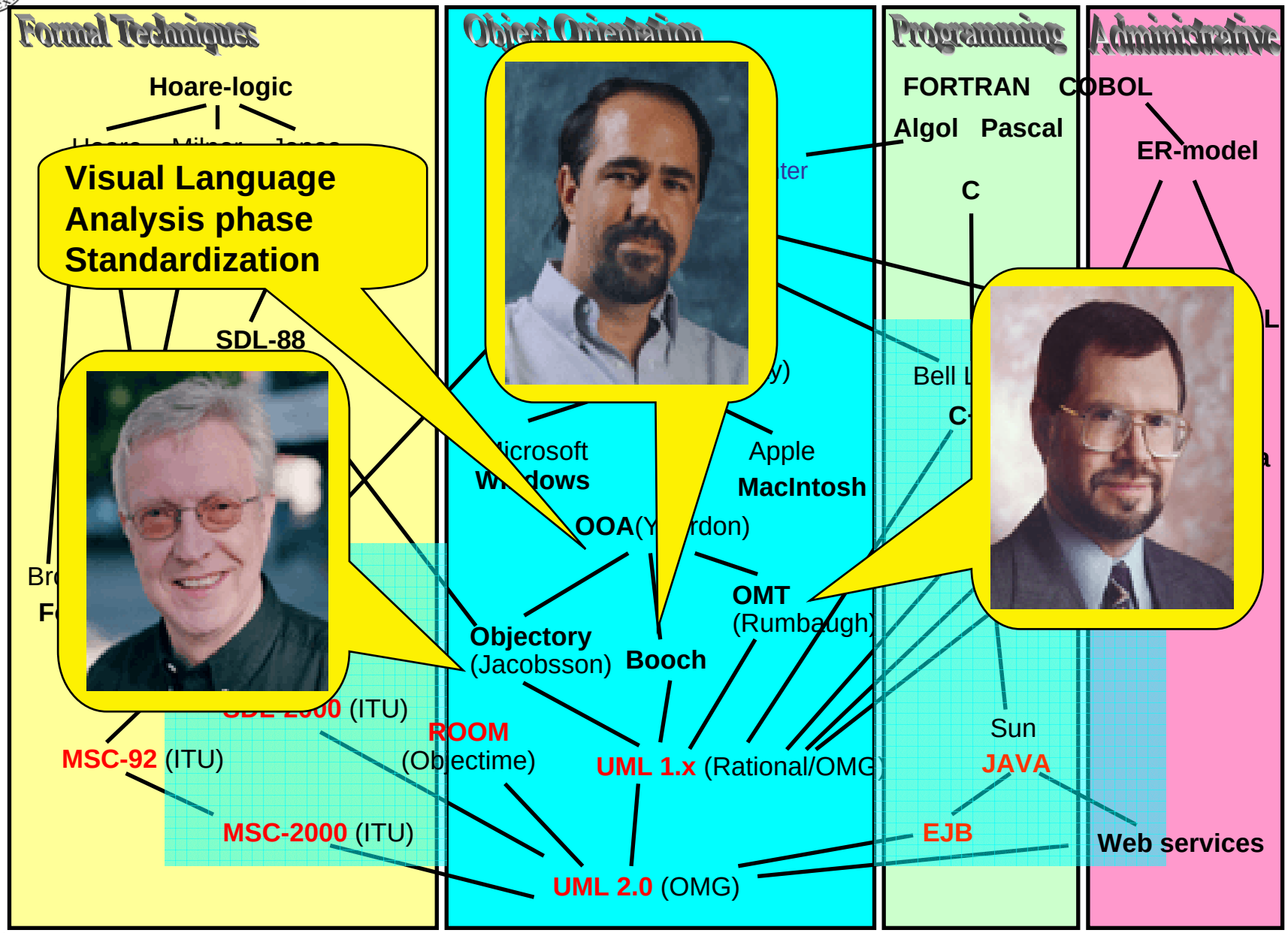
The founding fathers



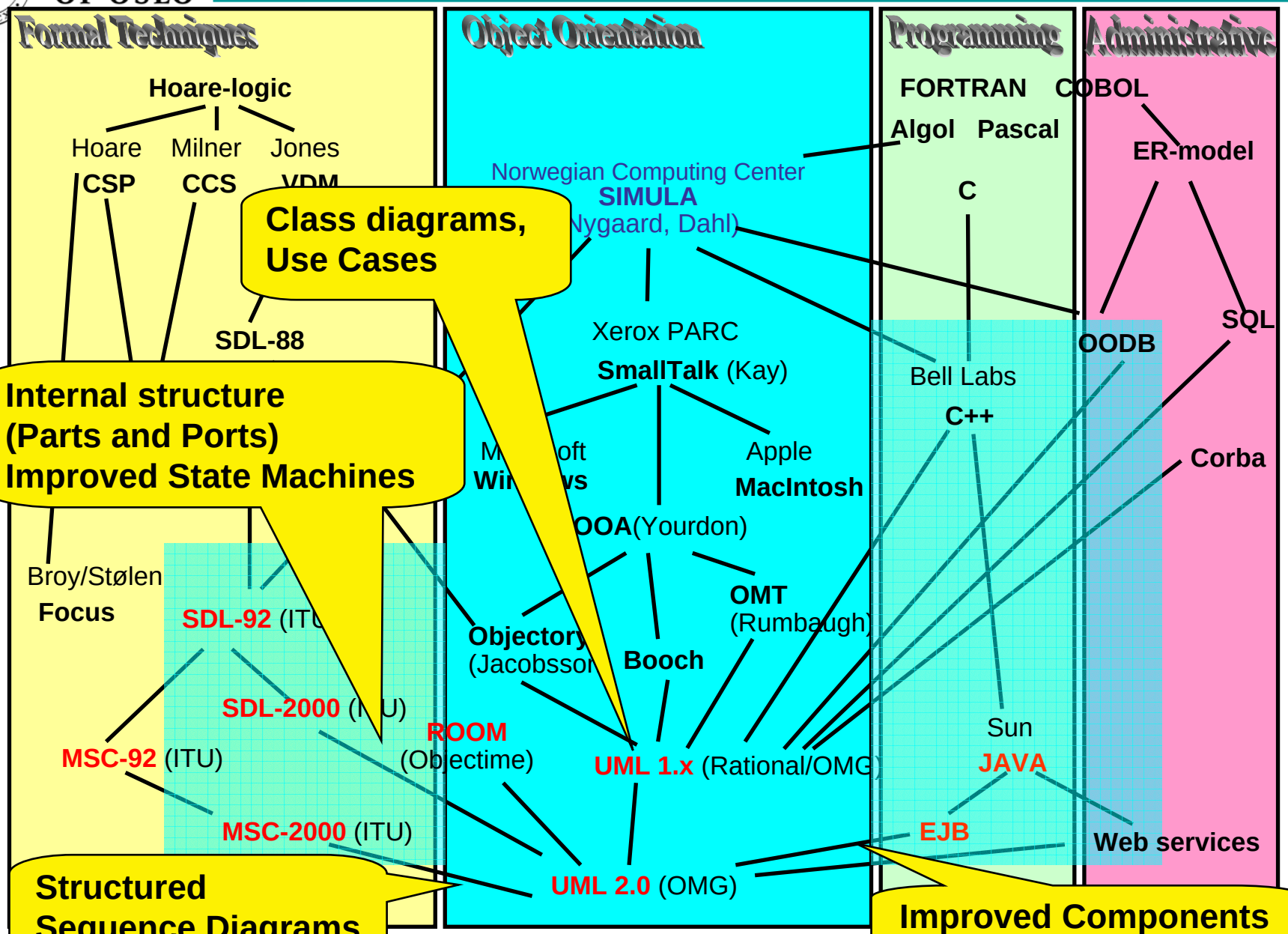
Making OO Popular and Commercial



The Three Amigos

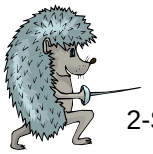


Influences on UML 2.0



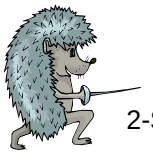
What language(s) to use?

- Requirements
  - used in practice for real engineering
  - expressive
  - visual
  - precise
  - trendy
- Alternatives
  - java (Sun)
    - possibly supplied with selected libraries
  - SDL (ITU)
  - MSC (ITU)
  - UML 1.x (OMG)
  - UML 2.0 (OMG)



Why choosing UML 2.0?

- Pro
  - UML is definitely trendy wrt. modeling languages
  - UML is standardized by open standardization organization (OMG)
  - UML 2.0 has most features of MSC and SDL
  - UML 2.0 is more precise and executable than UML 1.x
  - UML 2.0 is supported by more than one tool, and can be expressed through any drawing tool like Powerpoint, Visio, Framemaker
  - UML 2.0 is now, UML 1.x is history soon
- Con
  - Full UML 2.0 is probably not supported by any dedicated tool, yet
  - Real programmers do not use modeling languages anyway



UML Diagrams

- UML diagrams:
  - Use case diagram
  - Static structure diagrams:
    - Class / object diagram
    - Collaboration
    - Composite structure diagram
  - Behavior diagrams:
    - Sequence diagram
    - Communication diagram
    - State diagram
    - Activity diagram
  - Implementation diagrams:
    - Component diagram
    - Deployment diagram

Use:

Identifying main system functions

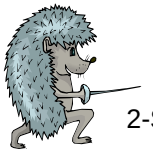
Domain and application modeling

internal structure of objects

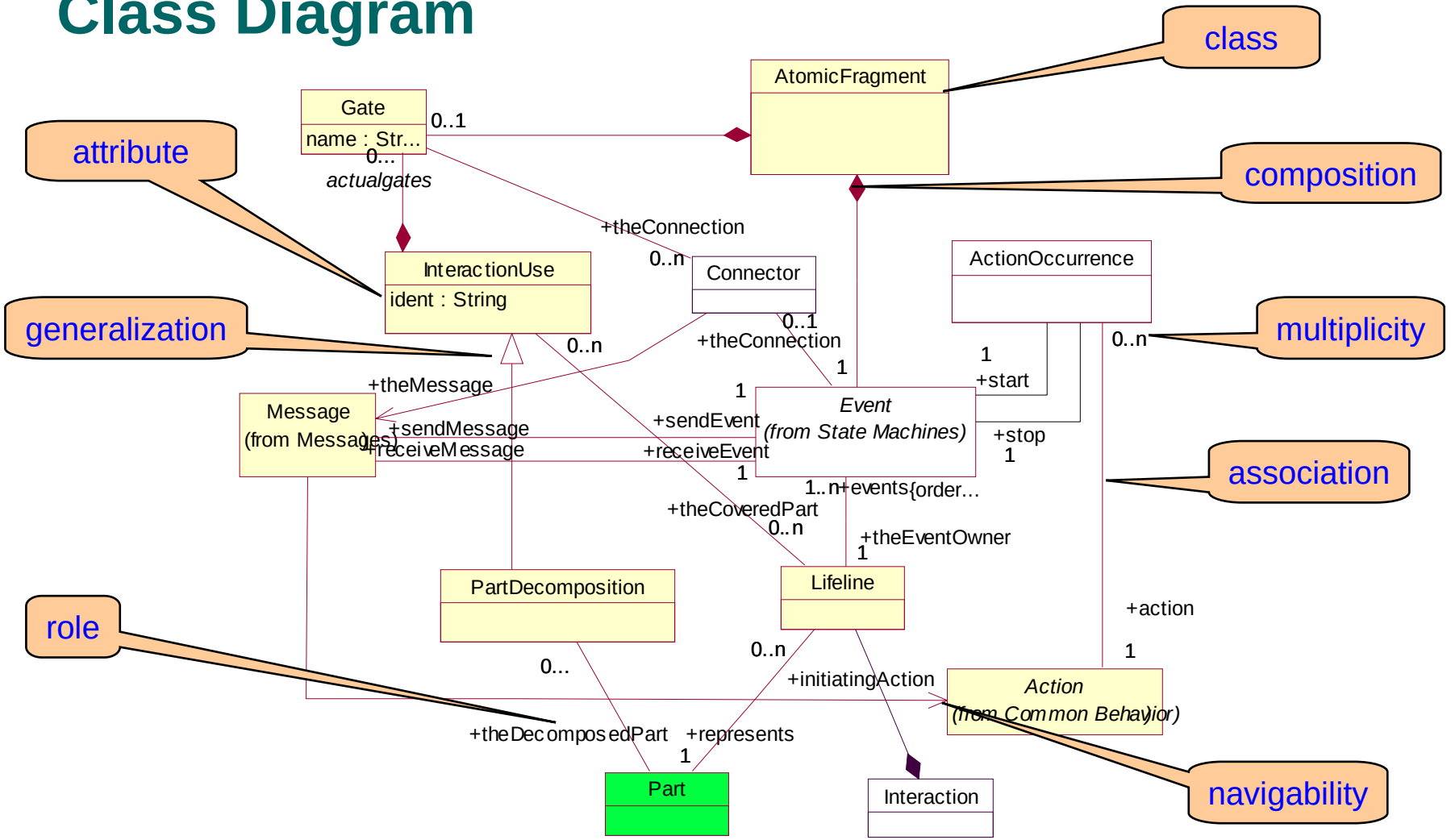
Interactions between objects

Class behaviour (state oriented)
Ditto (action oriented)

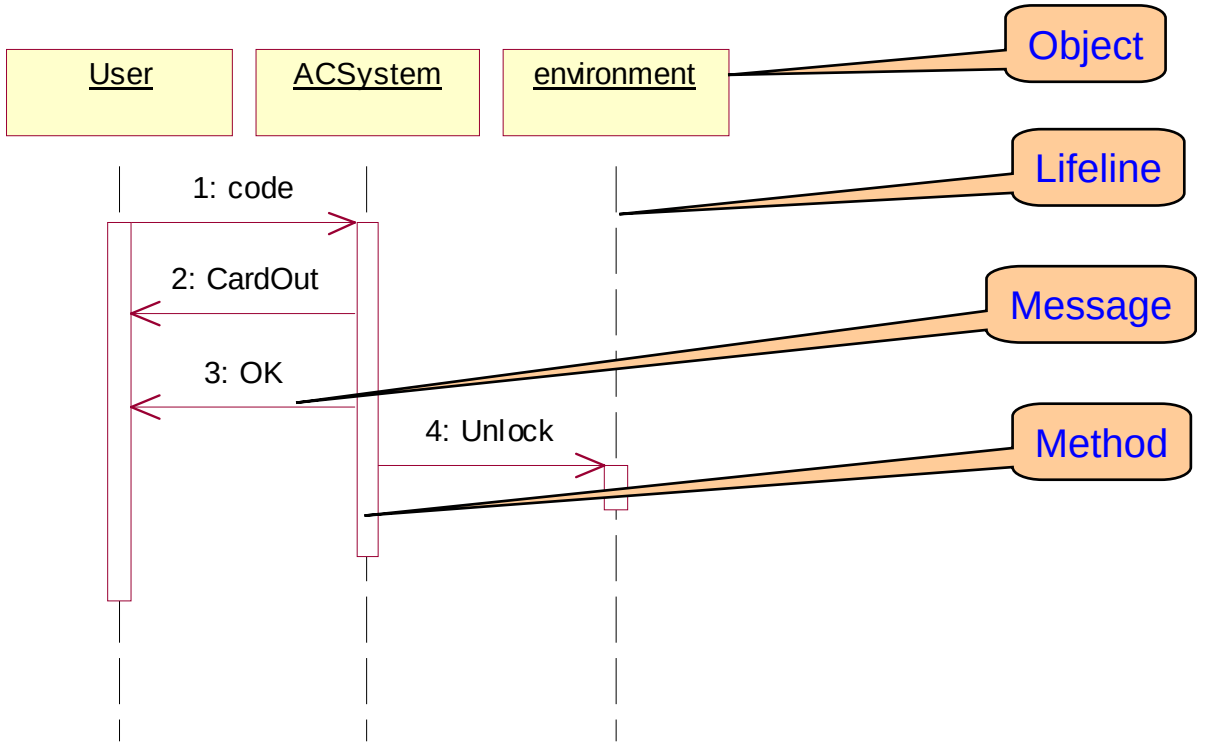
For software structure
For hardware/software structure

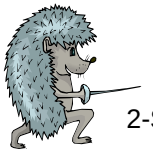


Class Diagram

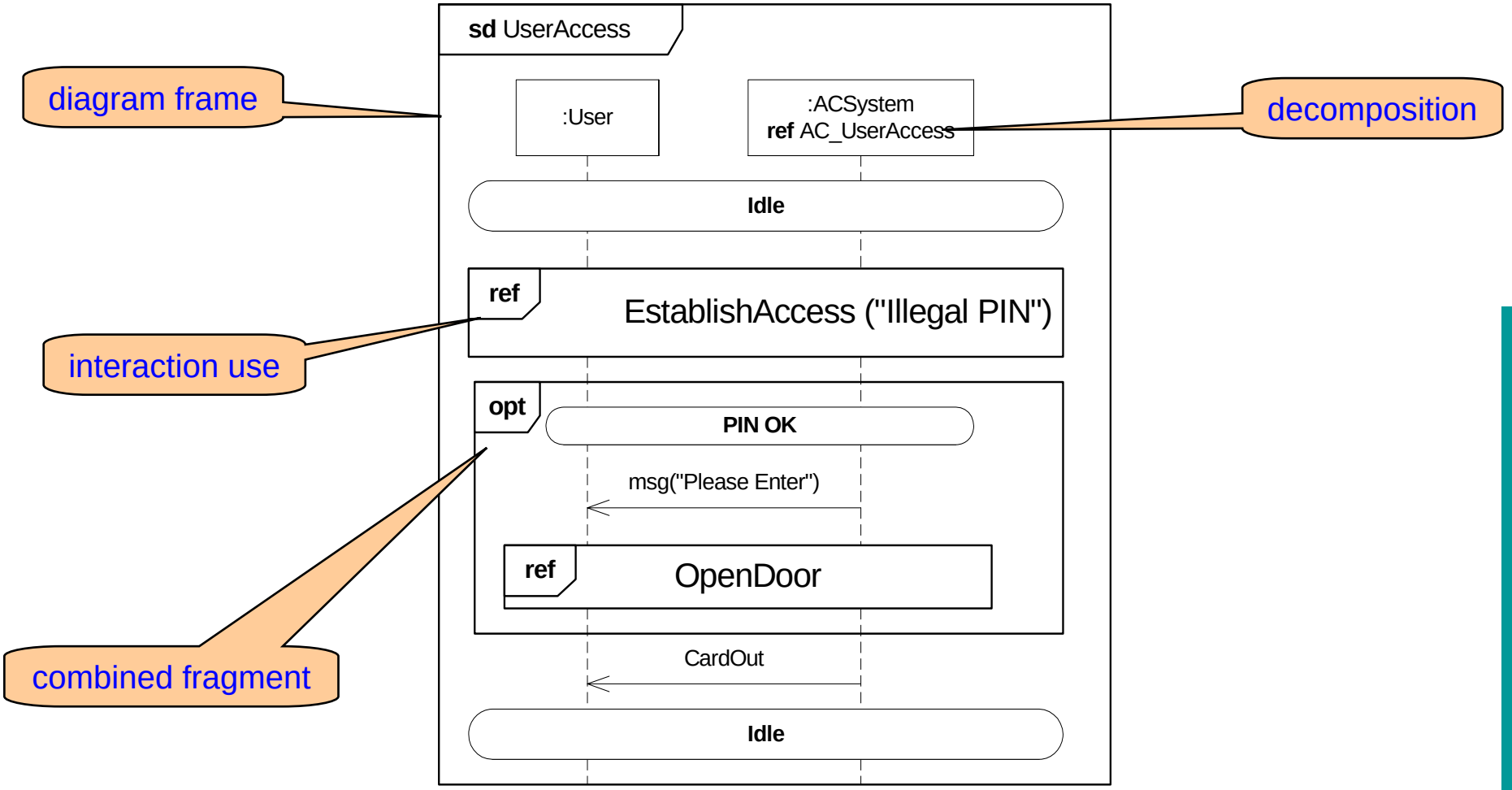


Sequence Diagram (UML 1.x corr. MSC-92)

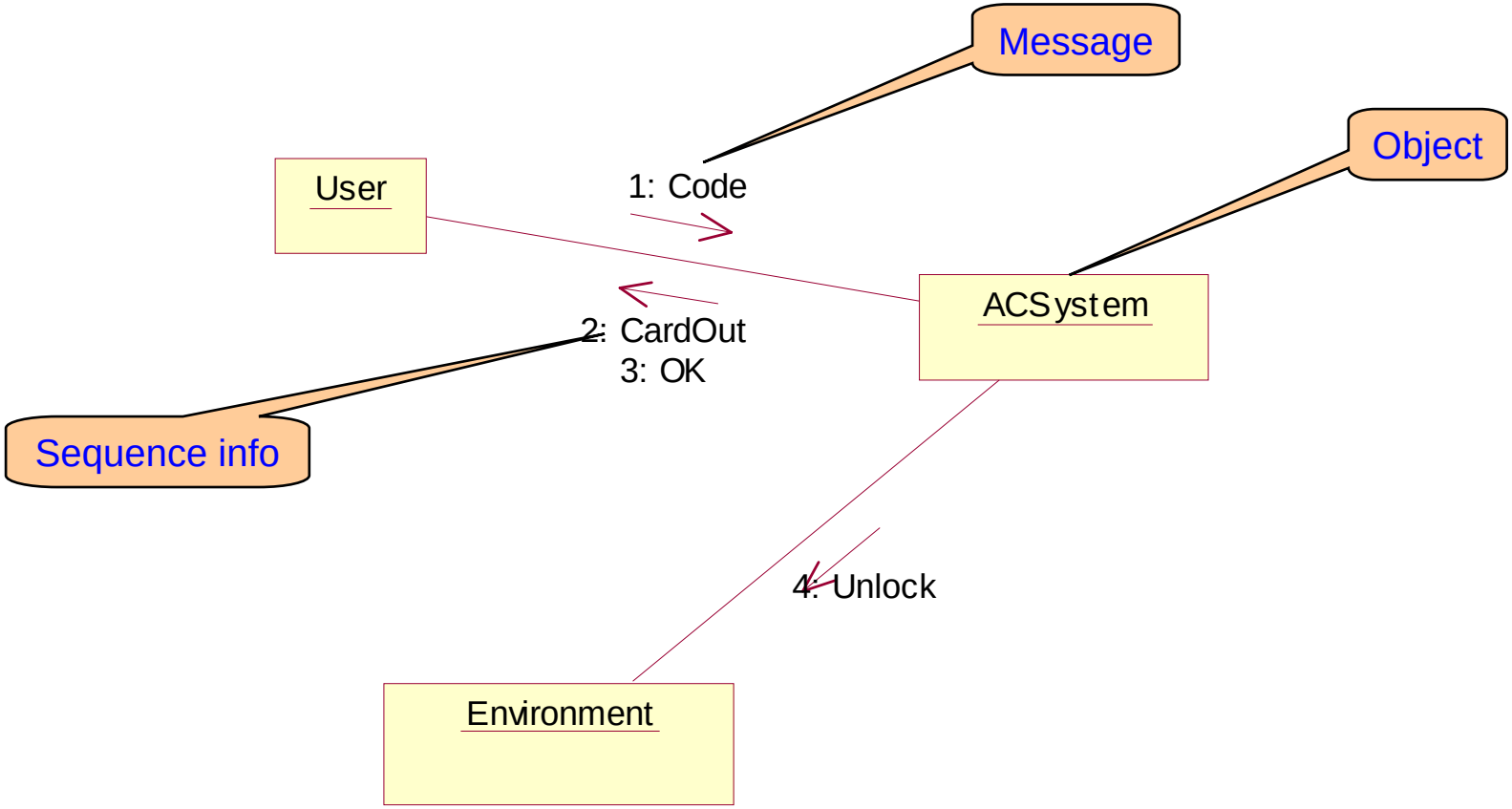




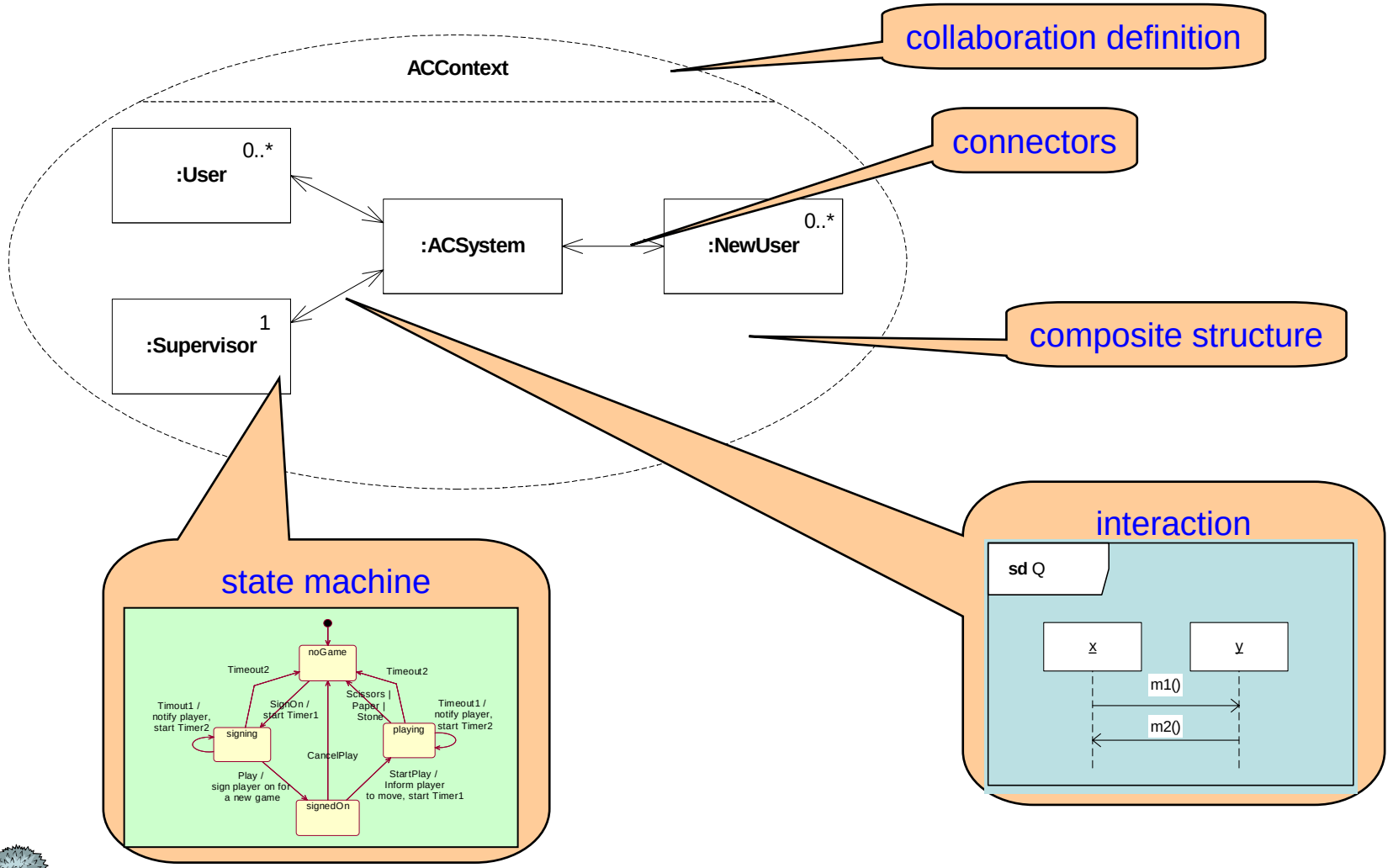
Sequence Diagram (MSC-2000 in UML clothes)



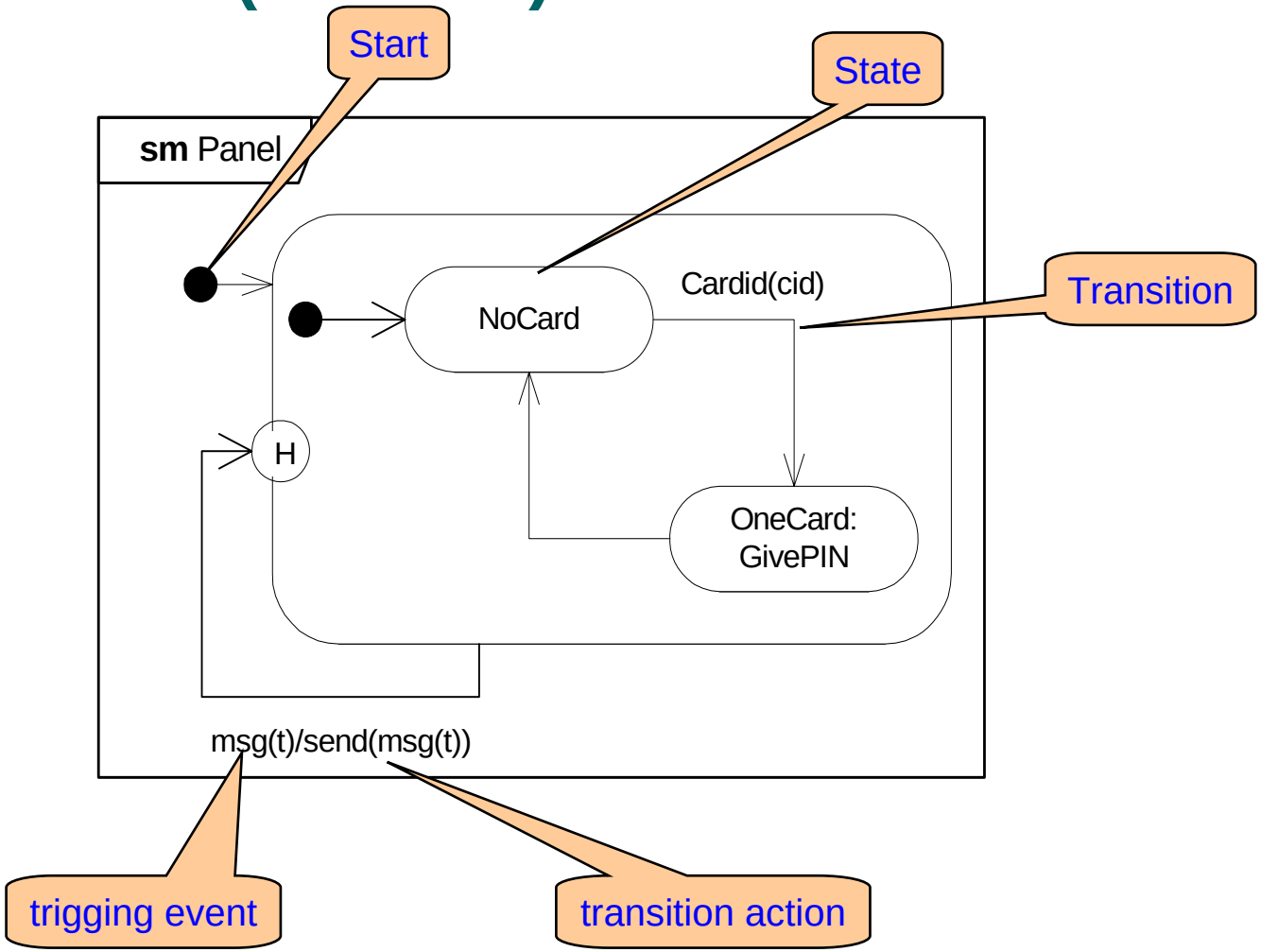
Collaboration diagram (UML 1.x)
Communication diagram (UML 2.0)



Collaborations in UML 2.0 clothes

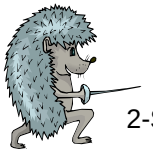


State Machines (UML 2.0)



How important are languages?

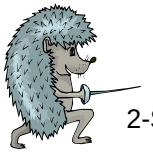
- Not very important
  - "Syntactic sugar"
- Very important
  - "Understanding through describing"



Methodology

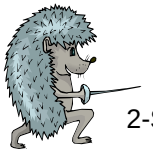
- A good language helps a lot
  - but is hardly sufficient
  - you need to know how to use the language also
- A good method is hard to find
  - easy to understand
  - easy to believe in
  - easy to follow
  - easy to modify
  - easy to get positive effects

  - easy to cheat?
  - easy to overlook?
  - easy to misuse?
  - hard to evaluate?

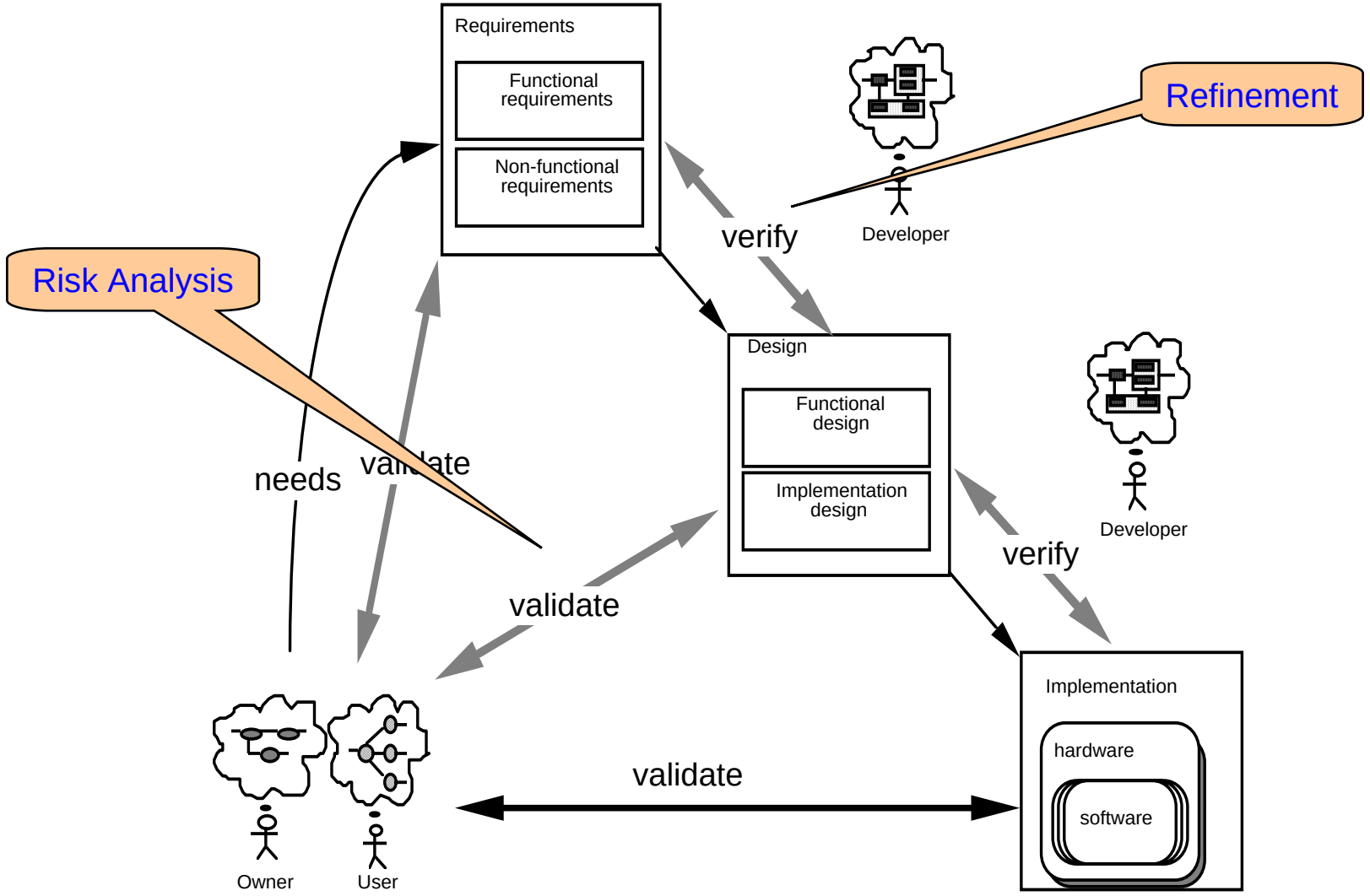


Verification and Validation

- Barry Boehm, 1981:
  - Verification: To establish the truth of correspondence between a software product and its specification (from the Latin veritas, “truth”). Are we building the product right?
  - Validation: To establish the fitness or worth of a software product for its operational mission (from the Latin valere, “to be worth”). Are we building the right product?
- Quality
  - process quality **= meeting the specification**
  - system quality **= playing the role required by the environment.**
- Quality assurance
  - Constructive **methods that aim to generate the right results in the first place**
  - Corrective **methods that aim to detect errors and make corrections.**

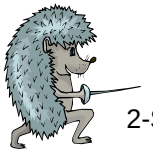


Development model

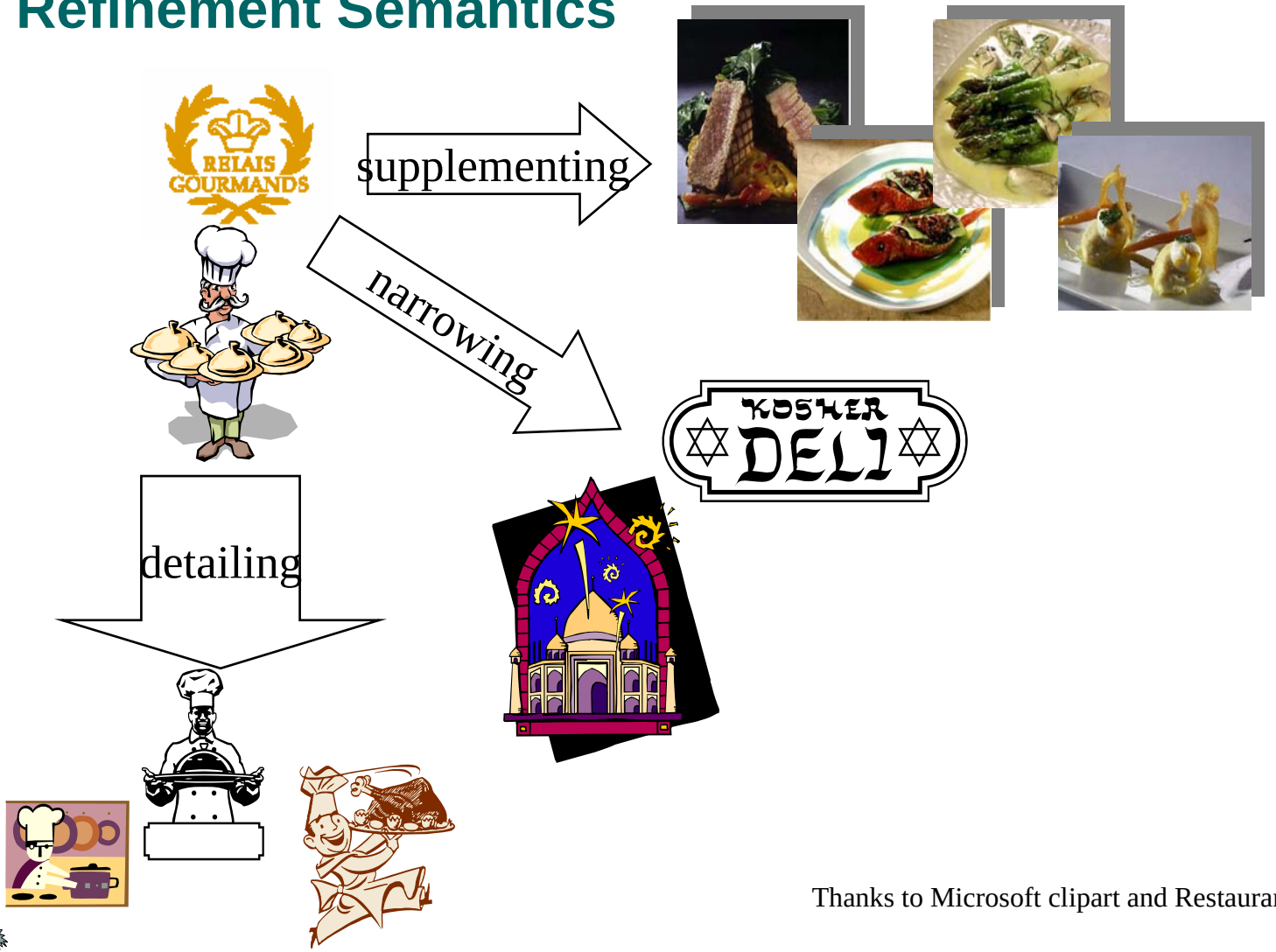


Dialectic Software Development

- Software Development is a process of learning
  - once you have totally understood the system you are building, it is done
- Learning is best achieved through conflict, not harmony
  - discussions reveal problematic points
  - silence hides critical errors
- By applying different perspectives to the system to be designed
  - inconsistencies may appear
  - and they must be harmonized
- Inconsistencies are not always errors!
  - difference of opinion
  - difference of understanding
  - misunderstanding each other
  - a result of partial knowledge
- Reliable systems are those that have already met challenges



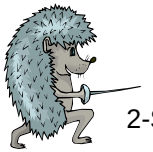
STAIRS – Steps To Analyze Interactions with Refinement Semantics



Thanks to Microsoft clipart and Restaurant Bagatelle's web-site

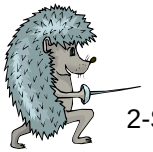
Refinement

- Refine = to free (as metal, sugar, or oil) from impurities or unwanted material
  - here: to make more exact, to reduce the set of legal solutions
  - in particular: to reduce the set of legal histories
- The role of histories
  - Histories model system runs
  - Specifications are modeled by sets of histories
- The need for a precise semantics
  - Syntax, Semantics, Pragmatics
- The assumption/guarantee paradigm
  - The assumption describes the properties of the environment in which the specified component is supposed to run
  - The guarantee characterizes the constraints that the specified component is required to fulfill whenever the specified component is executed in an environment that satisfies the assumption



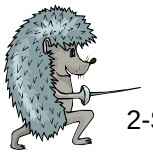
Three main notions of refinement

- Property refinement
  - *requirements engineering:* requirements are added to the specification in the order they are captured and formalized
  - *incremental development:* requirements are designed and implemented in a step-wise incremental manner
- Interface refinement
  - *type implementation:* introducing more implementation-dependent data types
  - *change of granularity:* replacing one step of interaction by several, or the other way around
- Conditional refinement
  - *imposing boundedness:* replacing unbounded resources by implementable bounded resources
  - *change of protocol:* replacing abstract communication protocols by more implementation-oriented communication protocols



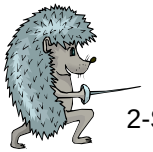
Objectives for the lectures on refinement

- The two lectures on refinement will
  - motivate and explain the basic instruments and principles for defining notions of refinement
    - this includes
      - using histories to model executions
      - the notion of an observer
      - understanding the assumption/guarantee principle
  - explain the following refinement concepts in a UML setting
    - property refinement
    - interface refinement
    - conditional refinement
  - demonstrate refinement in examples

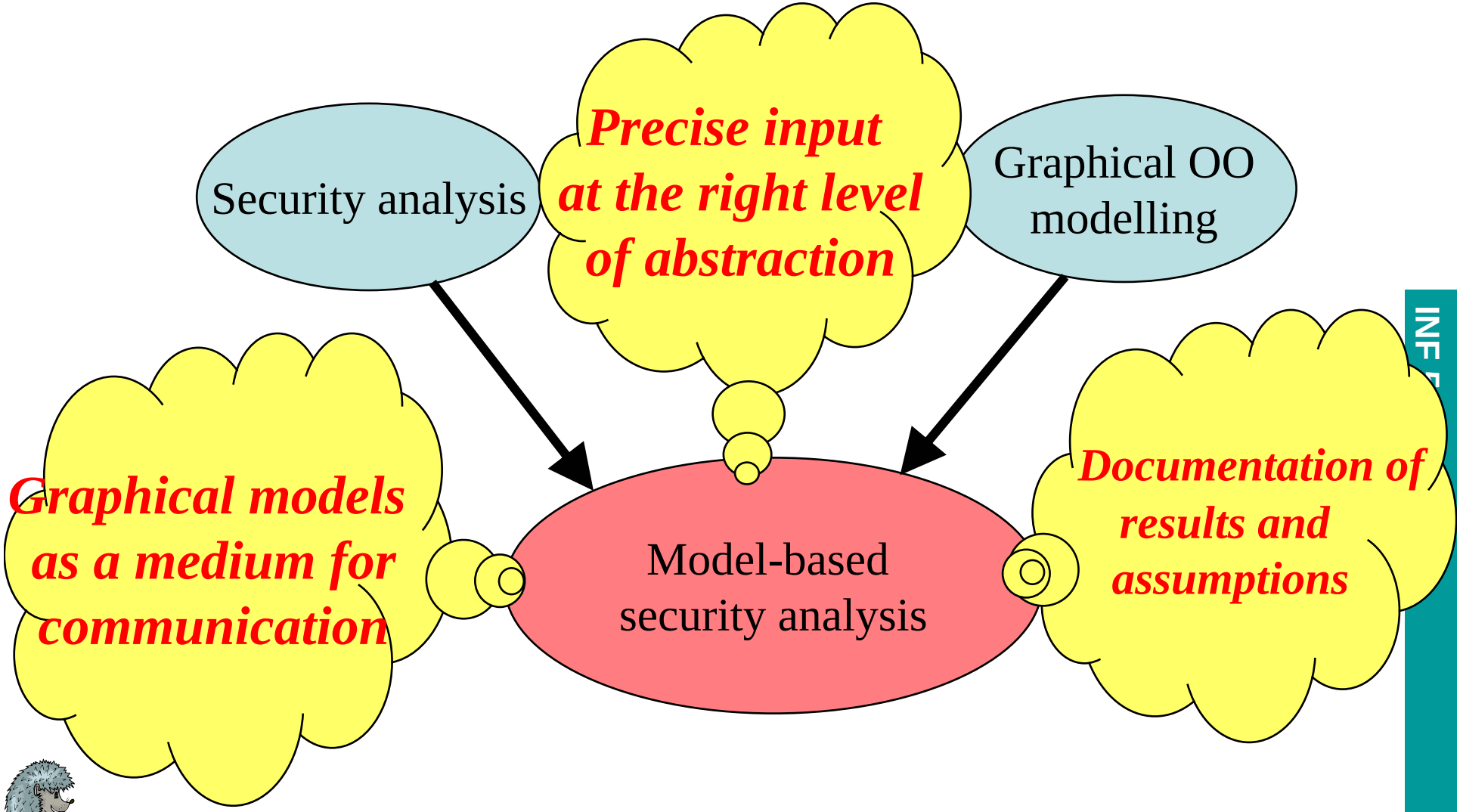


Model-based security analysis

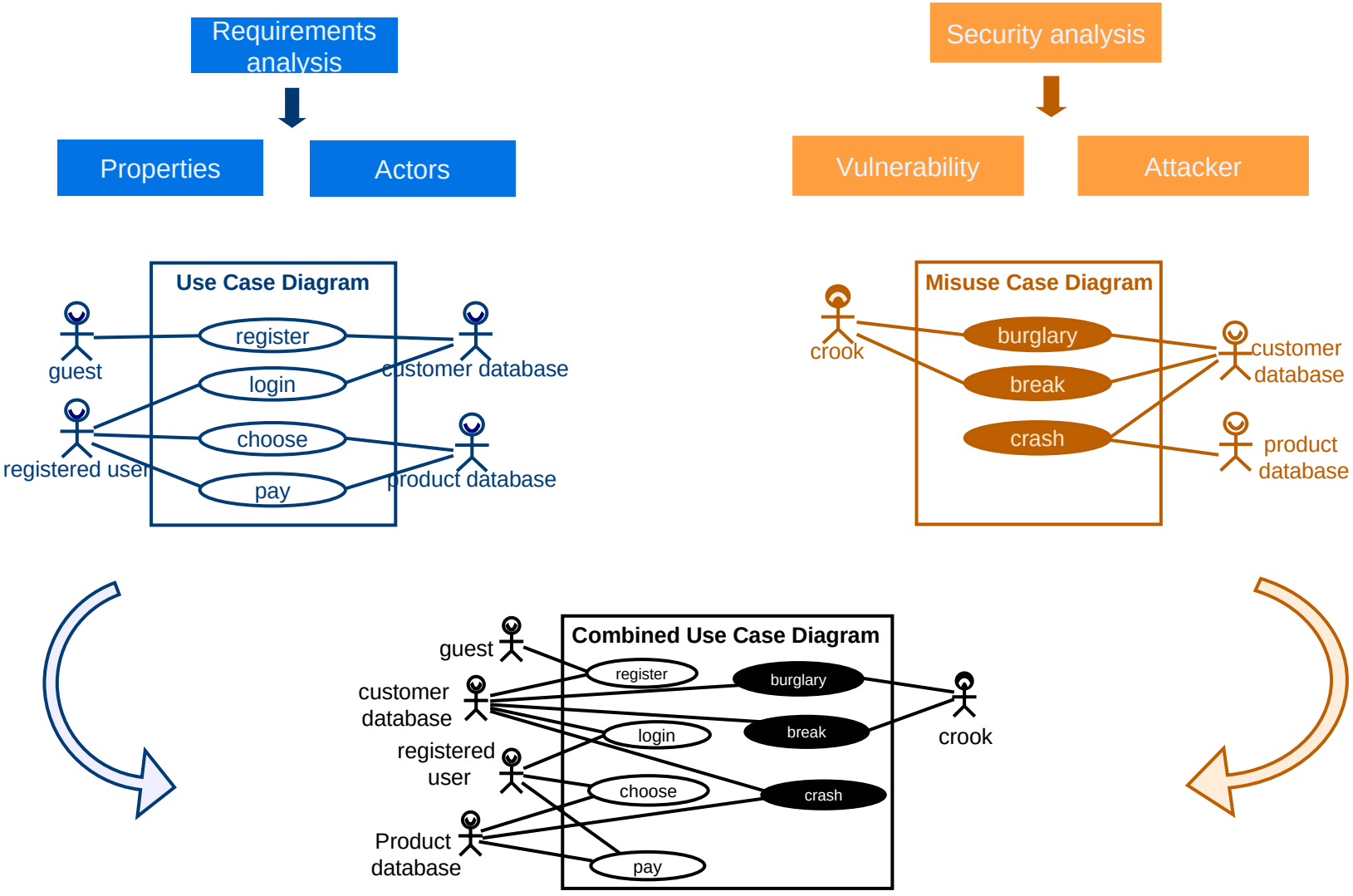
- Risk analysis is a systematic use of available information to
  - determine how often specified events may occur
  - the magnitude of their consequences
- Model-based security analysis is the tight integration of state-of-the art modeling methodology in the security risk analysis process
- Model-based security analysis is motivated by
  - Precision improves the quality of security analysis results
  - Graphical UML-like diagrams are well-suited as a medium for communication between stakeholders involved in a security analysis; the danger of throwing away time and resources on misconceptions is reduced
  - The need to formalize the assumptions on which the analysis depends; this reduces maintenance costs by increasing the possibilities for reuse
  - Provides a basis for tight integration of security analysis in the system development process; this may considerably reduce development costs since undesirable solutions are weeded out at an early stage



Three dimensions of model-based security analysis

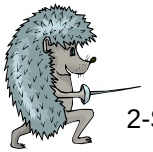


Requirements analysis versus security analysis



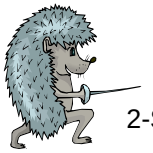
Objectives for the lectures on security analysis

- classify notions of dependability
- introduce, motivate and explain the basic notions and principles for risk management in general and security risk analysis in particular
- relate risk management to system development
- describe the various processes involved in risk management
- motivate and illustrate model-based security analysis
- present relevant standards
- demonstrate the usage of concrete analysis methodology



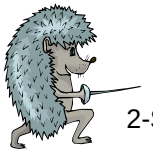
Obligatory Exercise: Blind Group Date

- Meeting a group of people that you do not know
  - based on SMS-message ”subscription”
  - mobile telephone positioning
  - Trafikanten online transportation service
- Specify the services with Sequence Diagrams
  - perform proper refinements
- Design the system with State Machines and Composite Structures
  - prove that the system is a refinement of the requirement
- Perform a risk analysis of the system
- Execute and demonstrate the system!



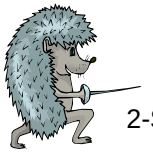
Obligatory Exercise: Tools (1)

- Eclipse
- Rational Software Modeler
  - runs on Linux as well as Windows
  - students can obtain copies for free
- UML to JavaFrame transformer as plug-in to RSM
  - push a button – executable UML! (Asbjørn Willersrud)
- Supplied interfaces to
  - PATS-lab for SMS-sending and positioning
    - Use only Telenor subscription mobiles!
  - Trafikanten online database



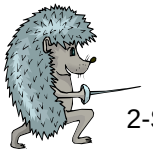
Obligatory Exercise: Tools (2)

- The CORAS-tool available as open source (LGPL-license):
  - http://coras.sourceforge.net/
- Based on other open software (Apache Cocoon, eXist XML database)



Obligatory exercise: Procedures

- Project groups
  - we put them together in a couple of weeks
- Drop 1:
  - Mandatory guidance by the assistants
  - Hard deadline at 23.59
  - Presentation with projector
  - Criticism by another group (written) and by the assistants
  - Public grading by lecturers
    - but this has no effect on the final grade
- Drop 2:
  - A baseline is given by the assistants
  - otherwise it is very much the same story as Drop 1



INFUIT (INF 5150) in a nutshell