Unified Process

Use-Case Driven

Rolf-Helge Pfeiffer - Anders Kalhauge

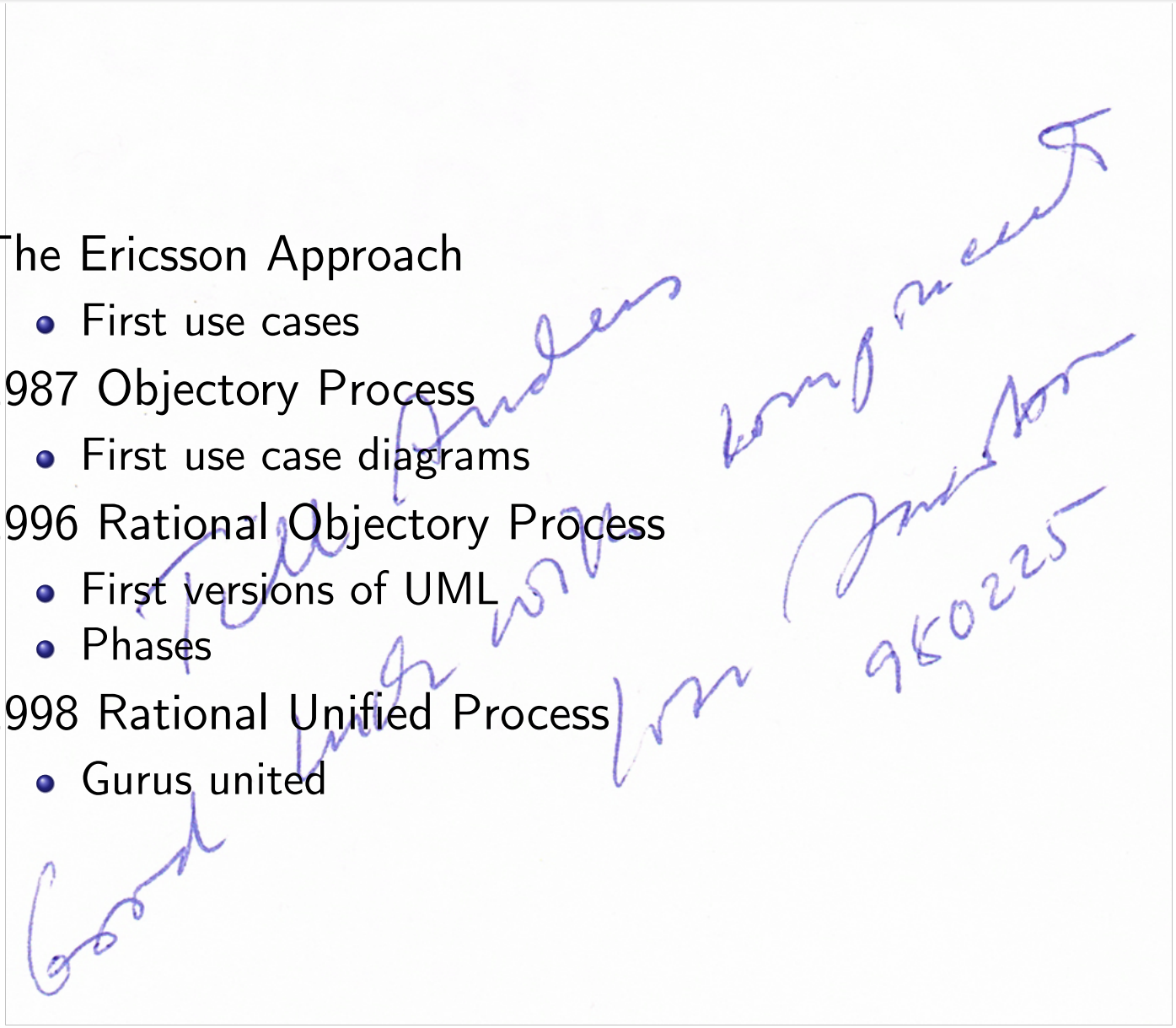
Fall 2016

Agenda

- Unified Process overview
- Unified Modeling Language
- Use cases

History of the Unified Process

- The Ericsson Approach
  - First use cases
- 1987 Objectory Process
  - First use case diagrams
- 1996 Rational Objectory Process
  - First versions of UML
  - Phases
- 1998 Rational Unified Process
  - Gurus united

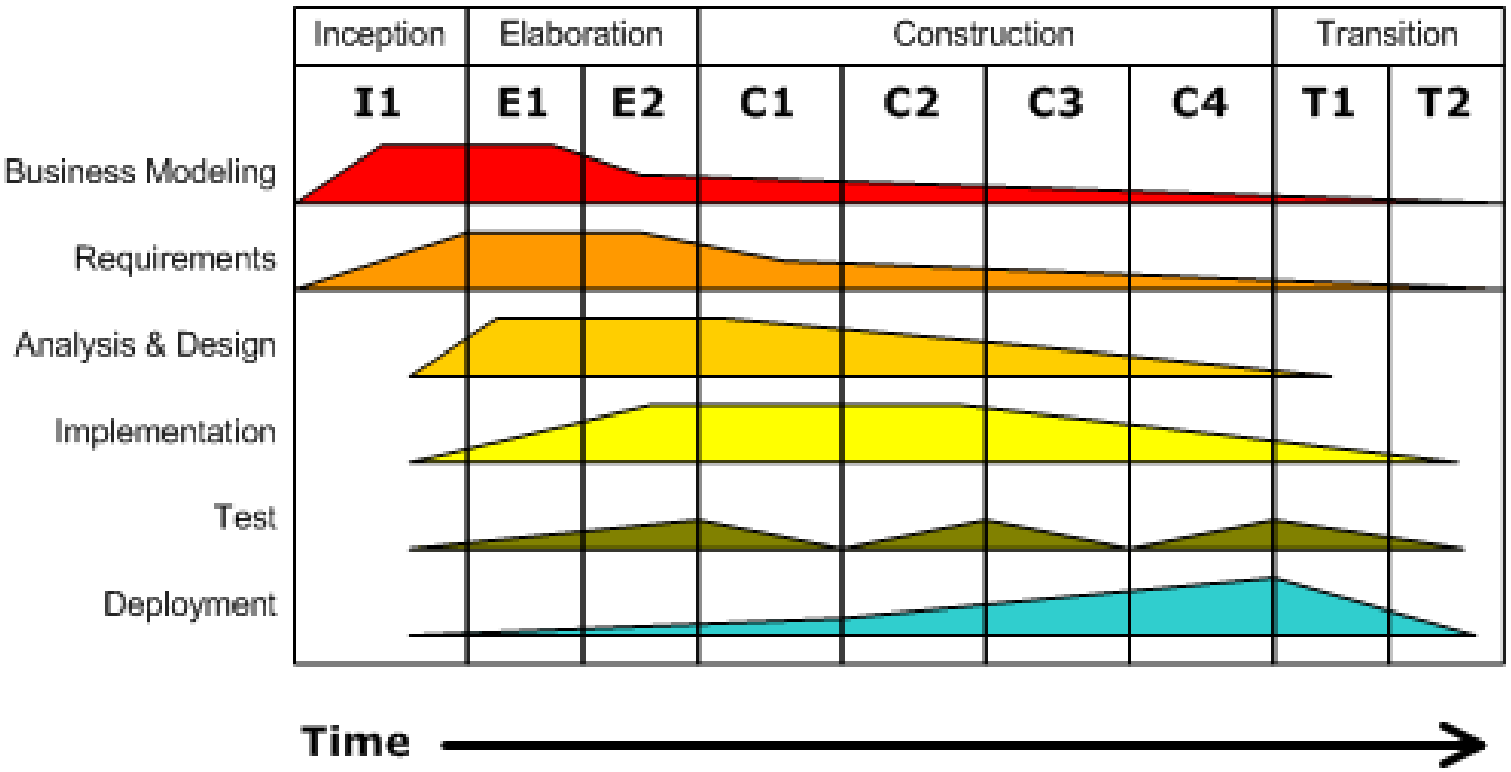


The Unified Process is

- **Use-Case Driven**
- Architecture-Centric
- Iterative and Incremental

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



- Business modeling
- Requirements Capture: From Vision to Requirements
- Capturing the Requirements as Use Cases
- Analysis
- Design
- Implementation
- Test

The Phases

- Inception
  Launches the Project
- Elaboration
  Makes the Architectural Baseline
- Construction
  Leads to Initial Operational Capability
- Transistion
  Completes Product Release

The goal in the inception phase is to make the business case to the extend necessary to justify lauching the project.

- Cost/benefit
  Will the gains accruing from the use or sale of the software product more than offset the cost of developing it?
- Time to market
  Will it reach the market (or internal application) in time to obtain these gains?
- Resolve system scope
- Resolve ambiguities in the requirements needed in this phase
- Establish a candidate architecture
- Mitigate the critical risks

- A feature list.
- A first version of a business (or domain) model that describes the context of the system.
- A first cut of the models representing a first version of the use-case model, the analysis model, the design model. Of the implementation model and test model, there may be something rudimentary. There is also a first version of the supplementary requirements.
- A first draft of a candidate architecture description with outlines of views of the use case, analysis, design, and implementation models.
- Possibly a proof-of-concept exploratory prototype, demonstrating the use of the new system.
- An initial risk list and a use-case ranking list.
- The beginnings of a plan for the entire project, including a general plan for the phases.
- A first draft of the business case, which includes: business context and success criteria (revenue projection, market recognition, project estimate).

The goal in the elaboration phase is to capture most of the remaining requirements, formulating the functional requirements as use cases.
Also a sound architectural foundation - the architectural baseline - must be established.

- Monitor remaining risks
- Fill in further details of the project plan.
- Judge the Worth of the Business Case

- Preferably a complete business (or domain) model which describes the context of the system.
- A new version of all models: use cases, analysis, design, deployment, and implementation. (At the end of the elaboration phase these models will be complete to less than 10% apart from the use case and analysis model that may include more (in some cases up to 80%) use cases to ascertain that the requirements have been understood. The majority of all use cases have been understood to make sure that no architecturally important use cases are left aside and that we can estimate the costs of introducing them.)
- An executable architectural baseline.
- An architecture description, including views of the use case, analysis, design, deployment, and implementation models.
- Updated risk list.
- Project plan for the construction and transition phases.
- A preliminary user manual (optional).
- Completed business case, including business bid.

The team working in the construction phase, starting from an executable architecture baseline and working through a series of iterations and increments, develops a software product ready for initial operation in the user environment.

- Project plan for the transition phase.
- The executable software itself—the initial-operational-capability release. This is the final build from construction.
- All artifacts, including models of the system.
- Maintained and minimally updated architecture description.
- Preliminary user manual in enough detail to guide beta users.
- Business case, reflecting situation at end of phase.

This phase focuses on establishing the product in the operational environment.

- Find out whether the systme really does what the business and its users request.
- Discover unanticipated risks.
- Note unresolved problems.
- Find failures.
- Fix ambiguities and gaps in the user documentation.
- Focus on areas where users appear to be deficient and in need of information or training.

- The executable software itself, including installation software.
- Legal documents such as contracts, license documents, waivers, and warranties.
- Completed and corrected product release baseline including all models of the system.
- Completed and updated architecture description.
- Final user, operator, and system administrator manuals and training materials.
- Customer support references and web references on where to find more information, to report defects, and to find information on fixes and upgrades.

Consider the following classification of prototypes by Christiane Floyd!

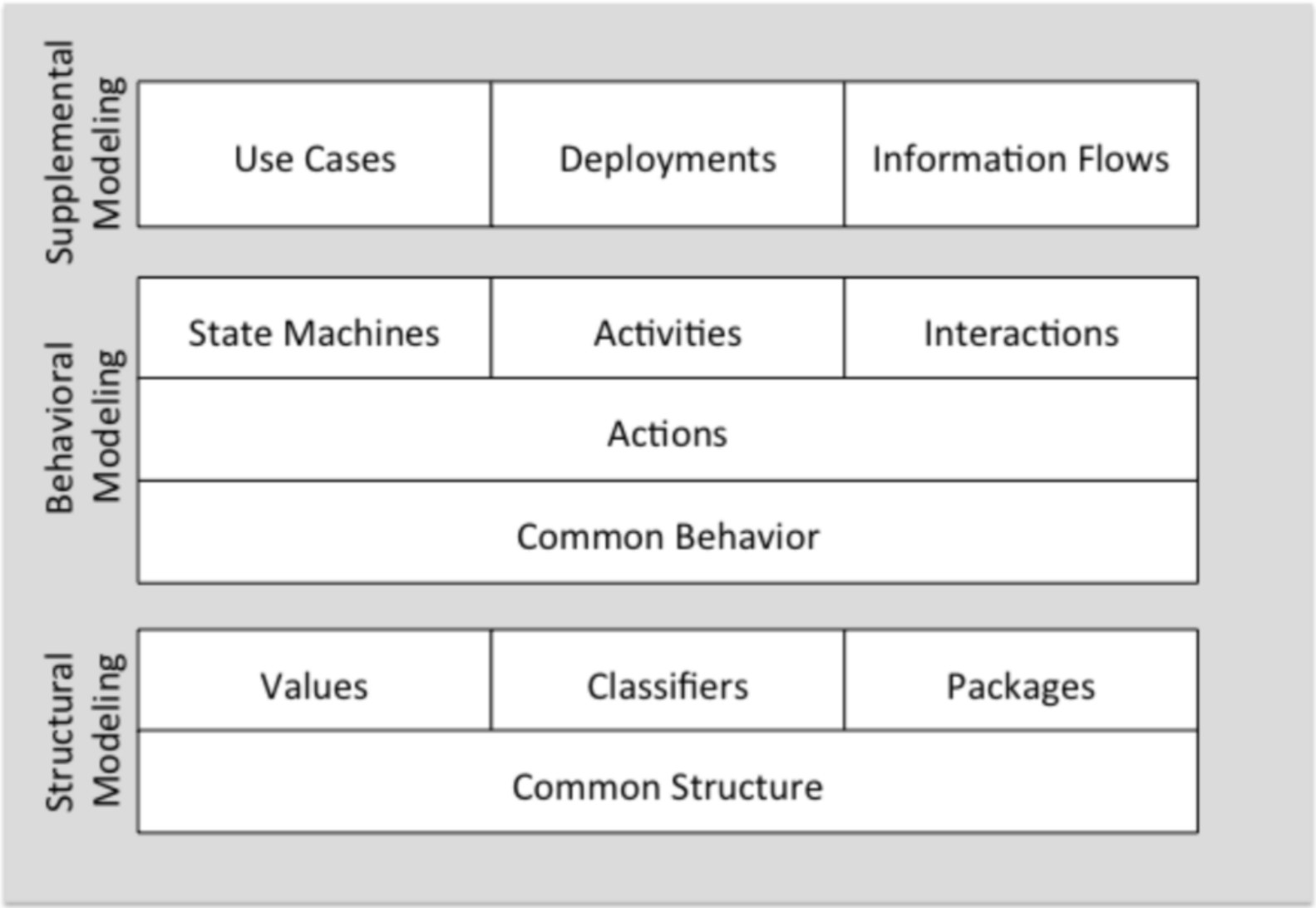
3. Approaches to Prototyping

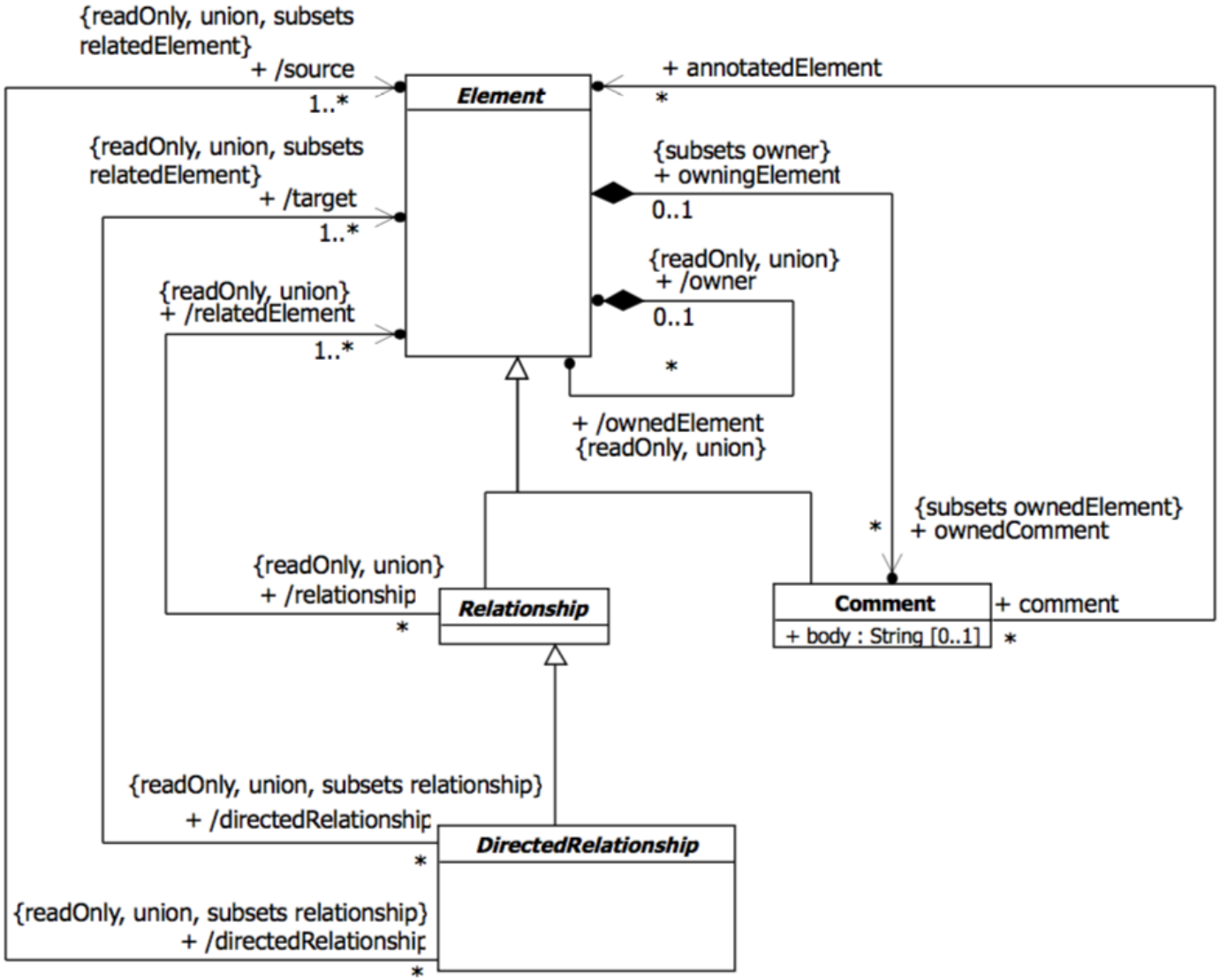
Depending on the goals we wish to achieve, we can distinguish three broad classes of prototyping:

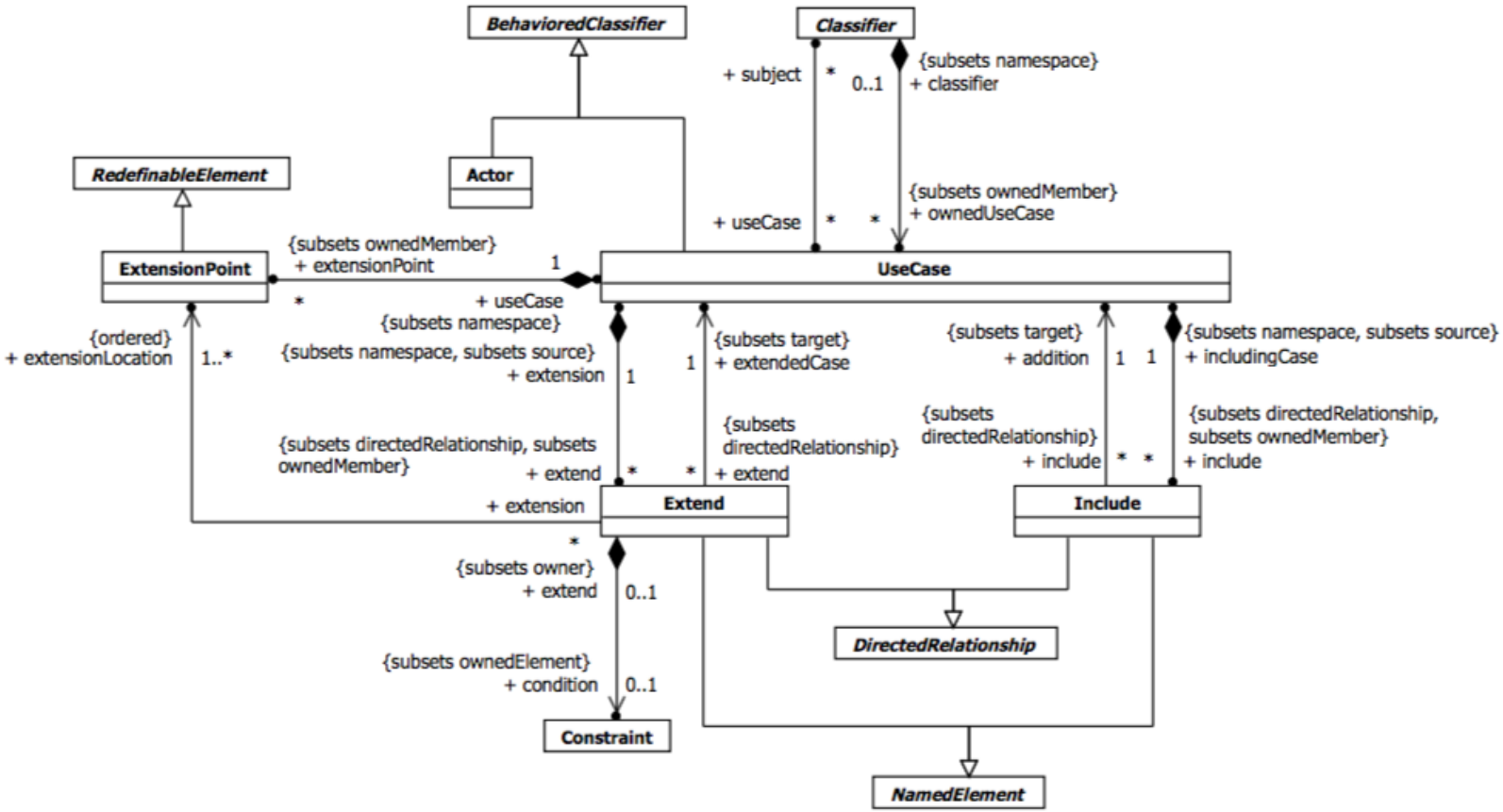
- prototyping for exploration, where the emphasis is on clarifying requirements and desirable features of the target system and where alternative possibilities for solutions are discussed,
- prototyping for experimentation, where the emphasis is on determining the adequacy of a proposed solution before investing in large-scale implementation of the target system,
- prototyping for evolution, where the emphasis is on adapting the system gradually to changing requirements, which cannot reliably be determined in one early phase.

Can you in any way relate them to the four UP phases?


TM


Supplemental Modeling
Use Cases
Deployments
Information Flows
Behavioral Modeling
State Machines
Activities
Interactions
Actions
Common Behavior
Structural Modeling
Values
Classifiers
Packages
Common Structure


{readOnly, union, subsets relatedElement}
+ /source
1..*
Element
+ annotatedElement
*
{readOnly, union, subsets relatedElement}
+ /target
1..*
{subsets owner}
+ owningElement
0..1
{readOnly, union}
+ /owner
0..1
*
{readOnly, union}
+ /relatedElement
1..*
+ /ownedElement
{readOnly, union}
{subsets ownedElement}
+ ownedComment
*
{readOnly, union}
+ /relationship
*
Relationship
Comment
+ body : String [0..1]
+ comment
*
{readOnly, union, subsets relationship}
+ /directedRelationship
*
DirectedRelationship
{readOnly, union, subsets relationship}
+ /directedRelationship
*


BehavioredClassifier
Classifier
+ subject
*
0..1
{subsets namespace}
+ classifier
RedefinableElement
Actor
{subsets ownedMember}
+ ownedUseCase
+ useCase
* *
{subsets ownedMember}
+ extensionPoint
1
ExtensionPoint
UseCase
* + useCase
{subsets namespace}
{ordered}
+ extensionLocation
1..*
{subsets namespace, subsets source}
+ extension
1
1
{subsets target}
+ extendedCase
{subsets target}
+ addition
1 1
{subsets namespace, subsets source}
+ includingCase
{subsets directedRelationship, subsets ownedMember}
+ extend
* *
{subsets directedRelationship}
+ extend
{subsets directedRelationship}
+ include
* *
{subsets directedRelationship, subsets ownedMember}
+ include
+ extension
Extend
Include
*
{subsets owner}
+ extend
0..1
DirectedRelationship
{subsets ownedElement}
+ condition
0..1
Constraint
NamedElement

A user story describes functionality that will be valuable to either a user or purchaser of a system or software.

- User stories are written by users.
- User stories focuses on details.
- User stories cannot be guaranteed to be complete.
- User stories does not define system boundaries.

A use case is a generalised description of a set of interactions between the system and one or more actors, where an actor is either a user or another system.

- Use cases are written by developers.
- A use case model will describe the complete (sub)system.
- A use case defines the system boundary.
- Any use case scenario will bring the system from one consistent state to another.

Alistair Cockburn has proposed a template for use cases, which can be found at:

`http://alistair.cockburn.us/Basic+use+case+template`

The template is used in the example later in this presentation.

Use cases can be at different detail levels:

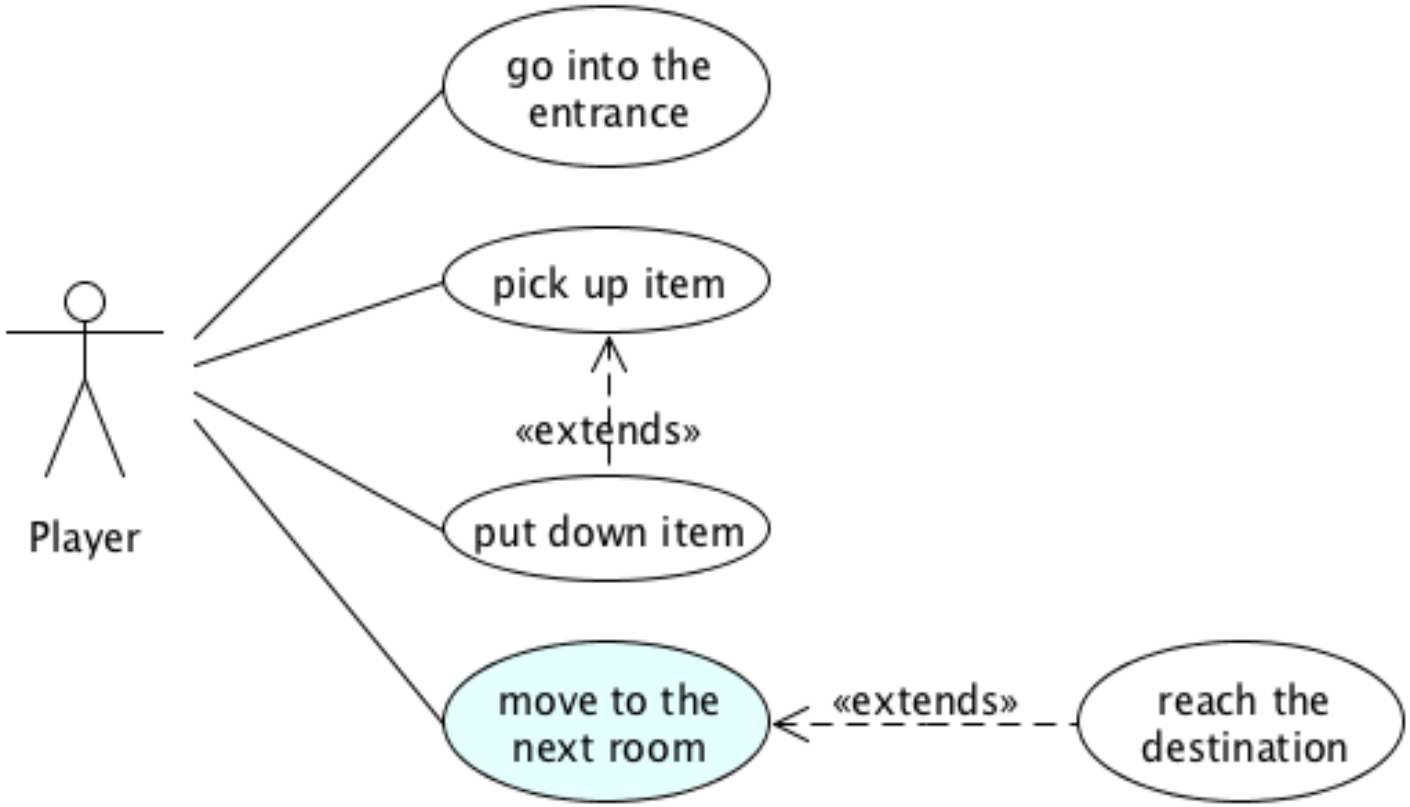
- Brief use case
  The use case is merely more than the title of an action
- Casual use case
  Typically casual use cases are described in more detail
- Fully dressed use case
  Includes all entries from the template

Actors are outside the system. For this reason, it is important to describe their responsibilities in detail. Actor resposibilities are not and should not be covered by the system.
Actors are not concrete people or systems, rather abstract roles of these users or systems.

Dungeon game example
Requirements

cphbusiness
COPENHAGEN BUSINESS ACADEMY

We want a Dungeon game. The scenario of the game is a number of connected rooms or dungeons in a mountain. The player enters the mountain from the entrance room, and he/she should travel from dungeon to dungeon until he/she reaches the treasury room. All dungeon has doors that leads to at least one other dungeon. A dungeon can contain an unlimited number of items. Items have values. When a player is in the room he/she can see the items in the room, and he/she can see the doors leading from the room to other dungeons. The player can pick up and lay down items when he/she is in a room. But he/she can keep at most five items at a time. The quest is to reach the treasury room with the most expensive items through the shortest path. The game should run on a central server, and played throug a mobile phone connected to the server.


go into the entrance
pick up item
«extends»
put down item
Player
move to the next room
«extends»
reach the destination

- **Name** Move to the next room
- **Scope** System under design (SuD)
- **Level** Goal: Move to the next room
- **Primary Actor** Player
- **Precondition** The player is in a room
- **Main succes scenario** ...
- **Success guaratees** The player is in a new room
- **Extensions** Reach the destination if room is treasury room
- **Special Requirements** NONE

- **Name** Move to the next room

. . .

- **Main succes scenario**
  1. Player chooses “move”
  2. System shows a list with all doors to other rooms
  3. Player selects the door he/she wants to move through
  4. System shows the room name, and asks the player to confirm
  5. Player confirms the selection of door
  6. System moves the player to the room behind the selected door

. . .

Dungeon game example
Use Case Model - Subsystem Sequence Diagram