15-413

Functional Modeling

Bernd Bruegge
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213

9999

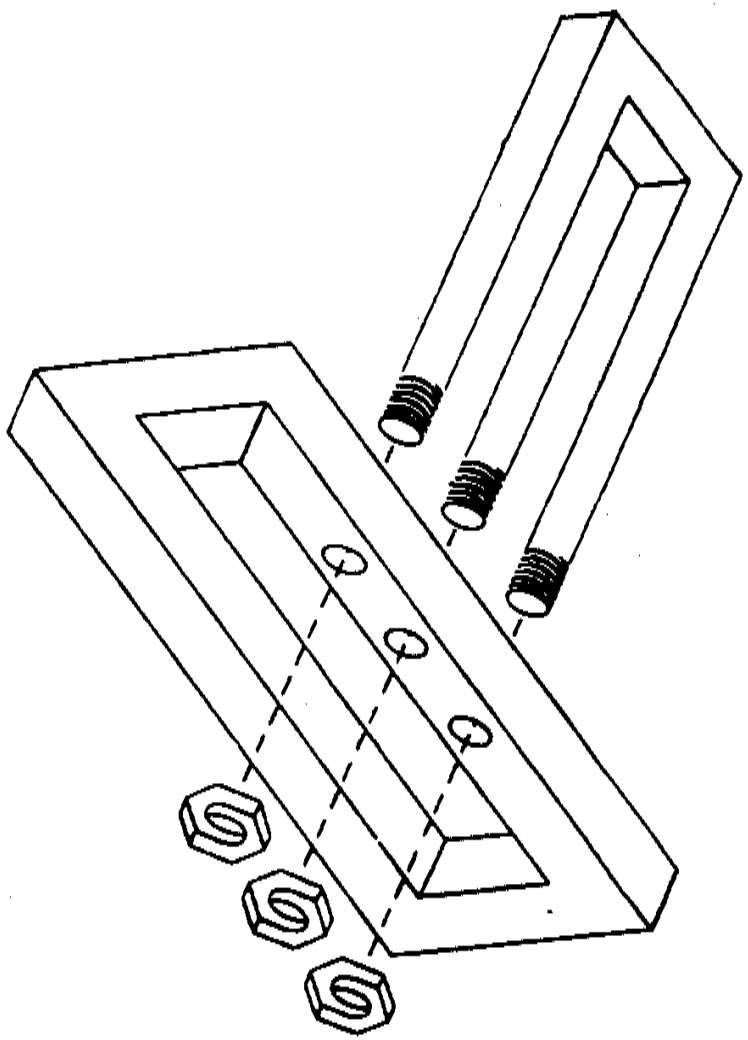
Odds and Ends

- Client is coming
  - **Introduction into AIMSS, ACS and GCS**
  - **Wednesday, Sep 15, Wean Hall 4623, currently scheduled for 4-6 PM (working on a earlier start)**
  - **Authoring and Workflow teams must attend, API liaisons should attend**
- No lecture on Thursday Sep 16
  - **Project Management Lecture is moved to Dec 7**
- Relevant Talk:
  - **Takeo Kanade, Robotics Institute, Virtualized Reality**
  - **"Digitizing a 3D Time-Varying Real Event As Is and in Real Time"**
  - **Friday, Sep 10, 3:30- 5:30 PM , Adamson Wing**
  - **Highly recommended for Augmented Reality and Modeling teams**
- First Architecture Team Meeting
  - **Before Takeo's talk?**

Odds and Ends ctd

- Interface Object (previous slide set) now called Boundary Object in the newest version of UML
- CS account form to be signed by each student

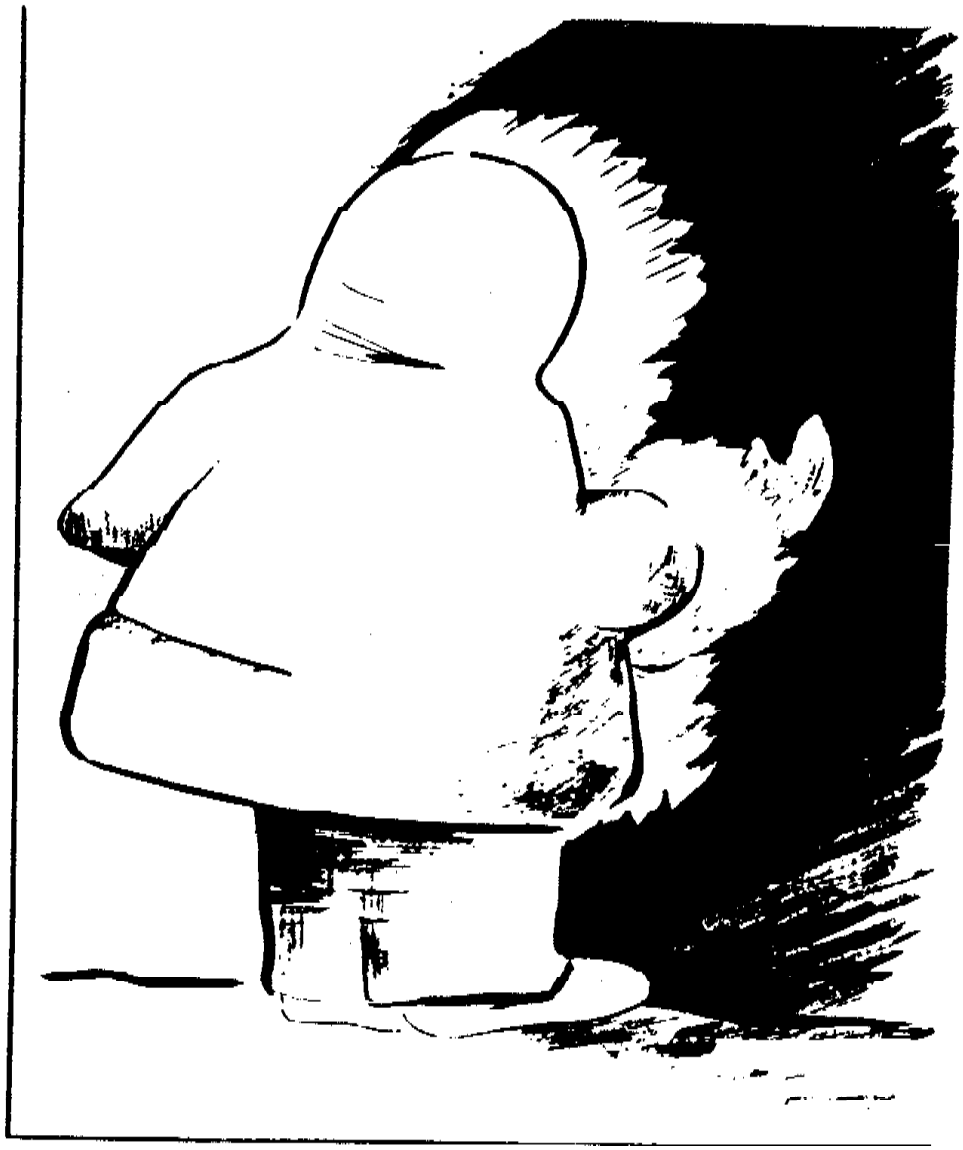
Can you develop this?



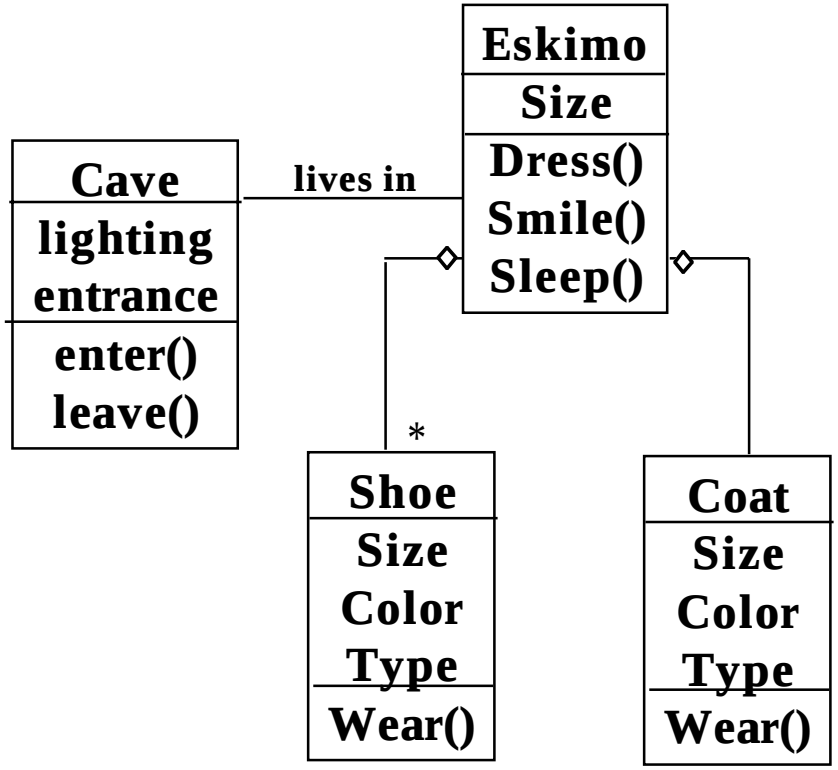
Recap: Types of Scenarios

- As-is scenario:
  - **Used in describing a current situation. Usually used during re-engineering. The user describes the system.**
- Visionary scenario:
  - **Used to describe a future system. Usually described in greenfield engineering or reengineering.**
  - **Can often not be done by the user or developer alone**
- Evaluation scenario:
  - **User tasks against which the system is to be evaluated**
- Training scenario:
  - **Step by step instructions designed to guide a novice user through a system**

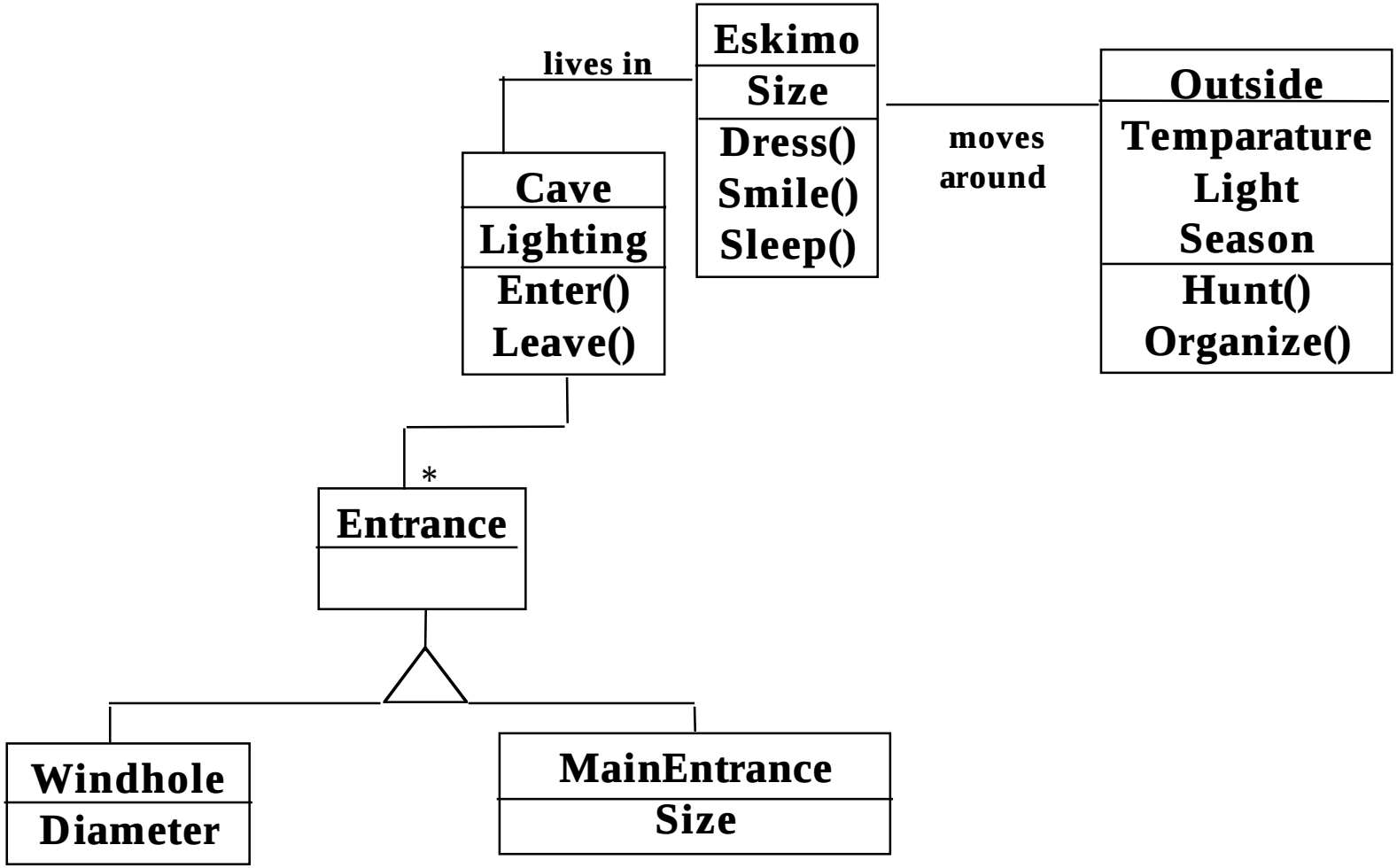
What is This?



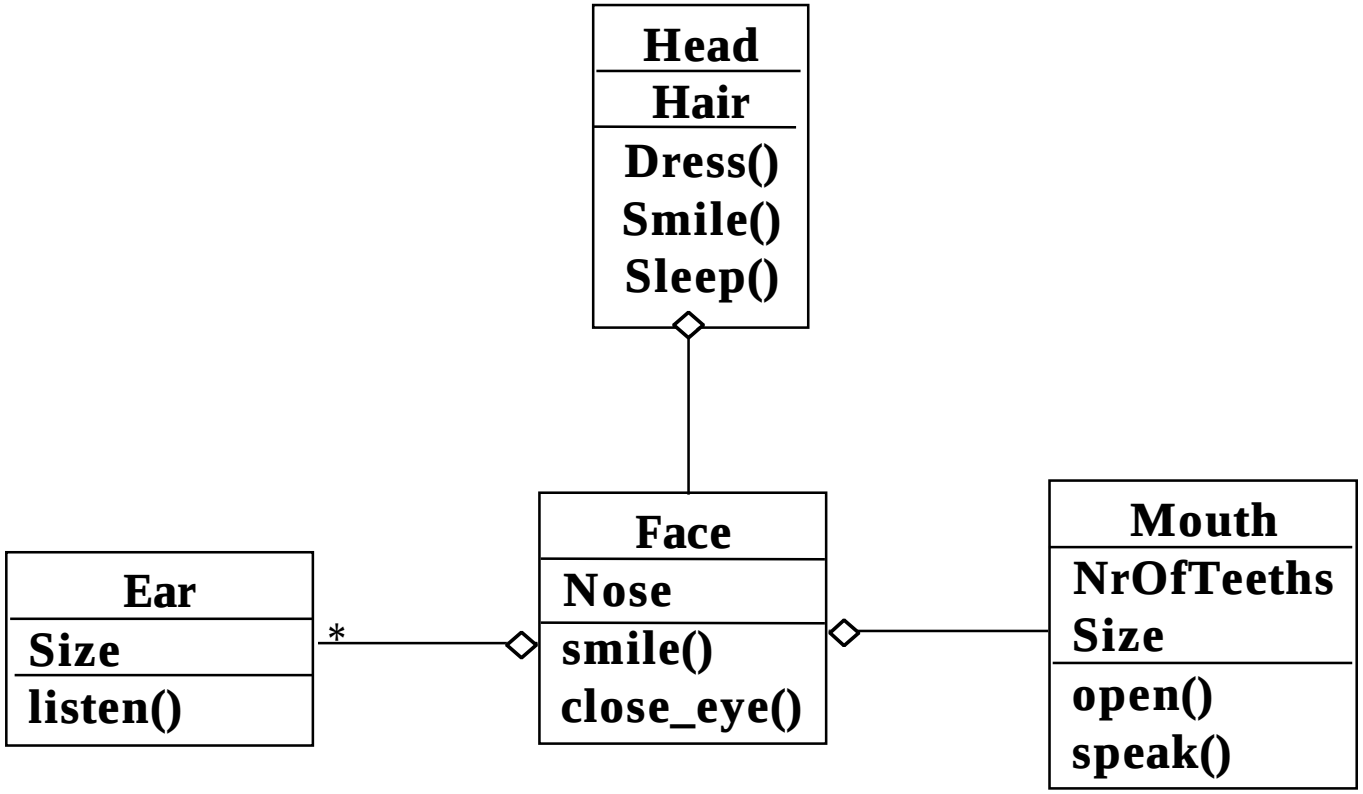
Possible Object Model: Eskimo



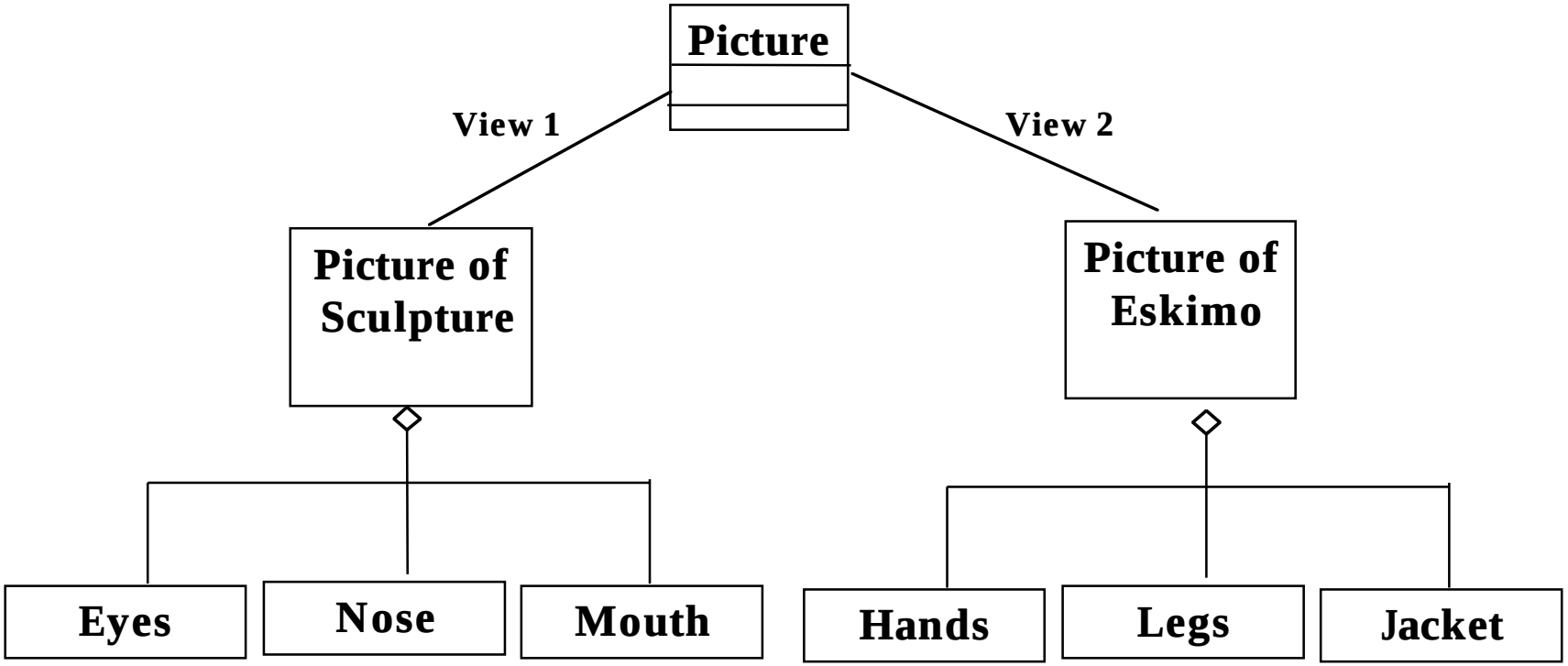
Modeling the Eskimo's Environment



Alternative: Head of an Eskimo



The Artist's View



System and Object identification

- Development of a system is not just done by taking a picture of a scene or domain
- Two important problems during requirements engineering and requirements analysis:
  - **Identification of objects**
  - **Definition of the system purpose**
  - **Depending on the purpose of the system, different objects might be found**
    - **What object is inside, what object is outside?**
  - **How can we identify the purpose of a system?**
    - **Scenarios**
    - **Use cases: Abstractions of scenarios**

Why Scenarios and Use Cases?

- Utterly comprehensible by the user
  - **Use cases model a system from the user users's point of view (functional requirements)**
    - **Define every possible event flow through the system**
    - **Description of interaction between objects**
- Great tools to manage a project. Use cases can form basis for whole development process
  - **User manual**
  - **System design and object design**
  - **Implementation**
  - **Test specification**
  - **Client acceptance test**
- An excellent basis for incremental & iterative development
- Use cases have also been proposed for business process reengineering (Ivar Jacobsen)

How do we find scenarios?

- Don’t expect the client to be verbal if the system does not exist (greenfield engineering)
- Don’t wait for information even if the system exists
- Engage in a dialectic approach (evolutionary, incremental)
  - **You help the client to formulate the requirements**
  - **The client helps you to understand the requirements**
  - **The requirements evolve while the scenarios are being developed**

Example: Accident Management System

- What needs to be done to report a “Cat in a Tree” incident?
- What do you need to do if a person reports “Warehouse on Fire?”
- Who is involved in reporting an incident?
- What does the system do if no police cars are available? If the police car has an accident on the way to the “cat in a tree” incident?
- What do you need to do if the “Cat in the Tree” turns into a “Grandma has fallen from the Ladder”?
- Can the system cope with a simultaneous incident report “Warehouse on Fire?”

Scenario Example: Warehouse on Fire

- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, reports the emergency from her car.
- Alice enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene given that area appear to be relatively busy. She confirms her input and waits for an acknowledgment.
- John, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.
- Alice received the acknowledgment and the ETA.

Observations about Warehouse on Fire Scenario

- Concrete scenario
  - **Describes a single instance of reporting a fire incident.**
  - **Does not describe all possible situations in which a fire can be reported.**
- Participating actors
  - **Bob, Alice and John**

Next goal, after the scenarios are formulated:

- Find a use case in the scenario that specifies all possible instances of how to report a fire
  - **Example: “Report Emergency “ in the first paragraph of the scenario is a candidate for a use case**

- Describe this use case in more detail
  - **Describe the entry condition**
  - **Describe the flow of events**
  - **Describe the exit condition**
  - **Describe exceptions**
  - **Describe special requirements (constraints, nonfunctional requirements)**

Example of steps in formulating a use case

- First name the use case
  - **Use case name: ReportEmergency**
- Then find the actors
  - **Generalize the concrete names (“Bob”) to participating actors (“Field officer”)**
  - **Participating Actors:**
    - **Field Officer (Bob and Alice in the Scenario)**
    - **Dispatcher (John in the Scenario)**
    - **?**
- Then concentrate on the flow of events
  - **Use informal natural language**

Example of steps in formulating a use case

- Formulate the Flow of Events:
  - **The** FieldOfficer **activates the "Report Emergency" function on her terminal. FRIEND responds by presenting a form to the officer.**
  - **The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes possible responses to the emergency situation. Once the form is completed, the FieldOfficer submits the form, at which point, the** Dispatcher **is notified.**
  - **The Dispatcher reviews the submitted information and creates an Incident in the database by invoking the OpenIncident use case. The Dispatcher selects a response and acknowledges the emergency report.**
  - **The FieldOfficer receives the acknowledgment and the selected response.**

Start using Abbot's technique for object identification in parallel to the use case modeling!

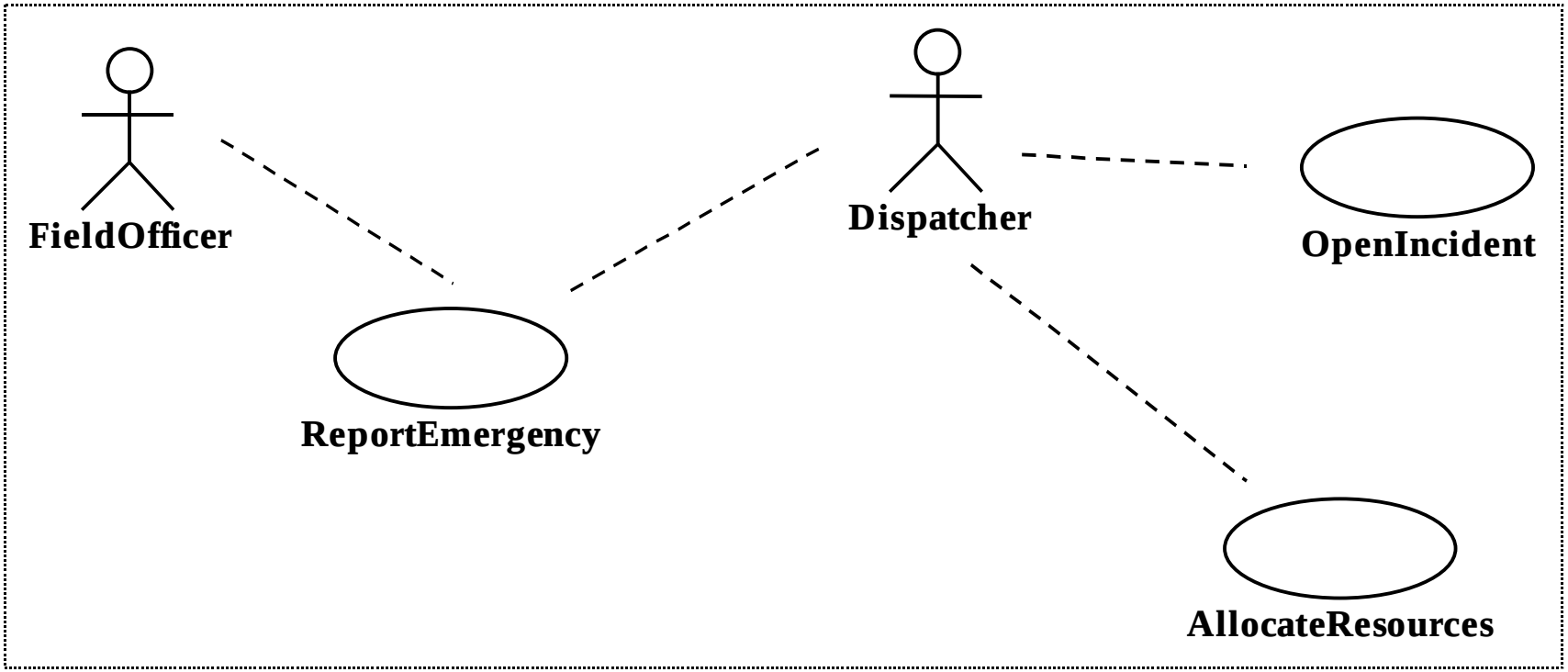
Example of steps in formulating a use case

- Write down the exceptions:
  - **The FieldOfficer is notified immediately if the connection between her terminal and the central is lost.**
  - **The Dispatcher is notified immediately if the connection between any logged in FieldOfficer and the central is lost.**
- Identify and write down any special requirements:
  - **The FieldOfficer's report is acknowledged within 30 seconds.**
  - **The selected response arrives no later than 30 seconds after it is sent by the Dispatcher.**

How to Specify a Use Case (Summary)

- Name of Use Case
- Actors
  - **Description of actors involved in use case**
- Entry condition
  - **Use a syntactic phrase such as "This use case starts when…"**
- Flow of Events
  - **Free form, informal natural language**
- Exit condition
  - **Star with "This use cases terminates when…"**
- Exceptions
  - **Describe what happens if things go wrong**
- Special Requirements
  - **List nonfunctional requirements and constraints**

Use Case Model for Incident Management

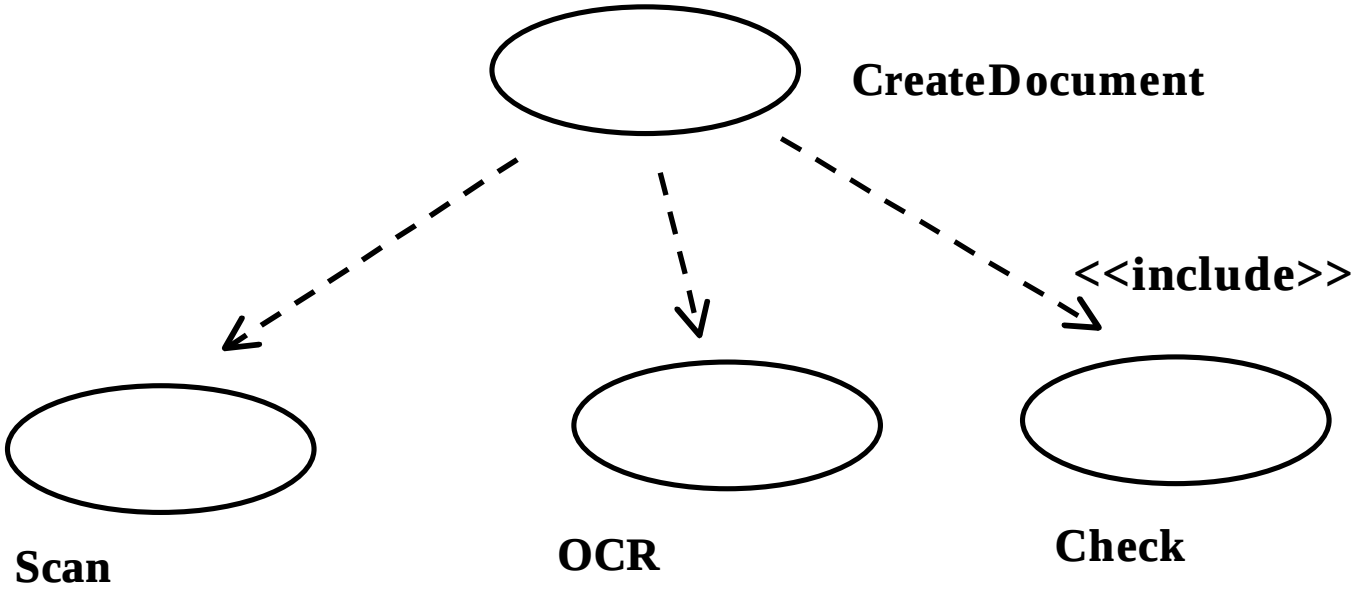


Use Case Associations

- Use case association = relationship between use cases
- Important types:
  - **Extends**
    - **A use case extends another use case**
  - **Include**
    - **A use case uses another use case (“functional decomposition”)**
  - **Generalization**
    - **An abstract use case has different specializations**

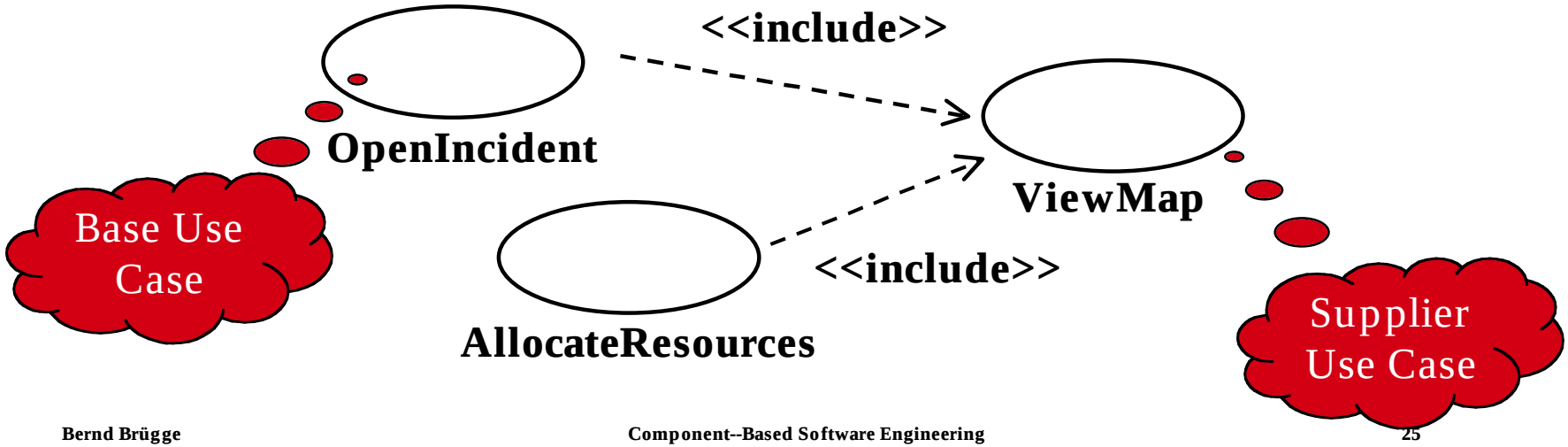
<<Include>>: Functional Decomposition

- Problem:
  - **A function in the original problem statement is too complex to be solvable immediately**
- Solution:
  - **Describe the function as the aggregation of a set of simpler functions. The associated use case is decomposed into smaller use cases**



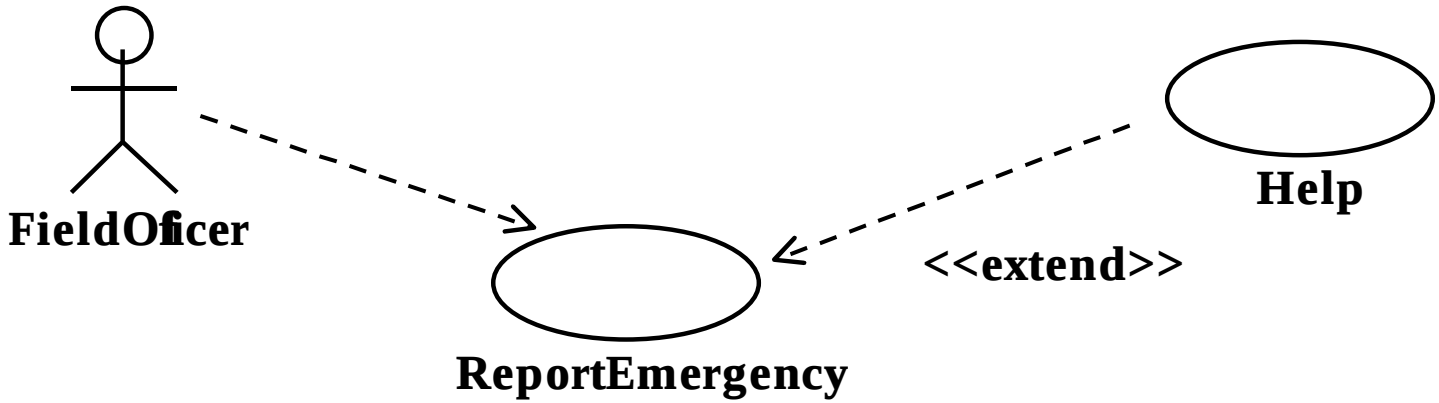
<<Include>>: Reuse of Existing Functionality

- Problem:
  - **There are already existing functions. How can we *reuse* them?**
- Solution:
  - **The *include association* from a use case A to a use case B indicates that an instance of the use case A performs all the behavior described in the use case B ("A delegates to B")**
- Example:
  - **The use case "ViewMap" describes behavior that can be used by the use case "OpenIncident" ("ViewMap" is factored out)**
- **Note: The base case cannot exist alone. It is always called with the supplier use case**



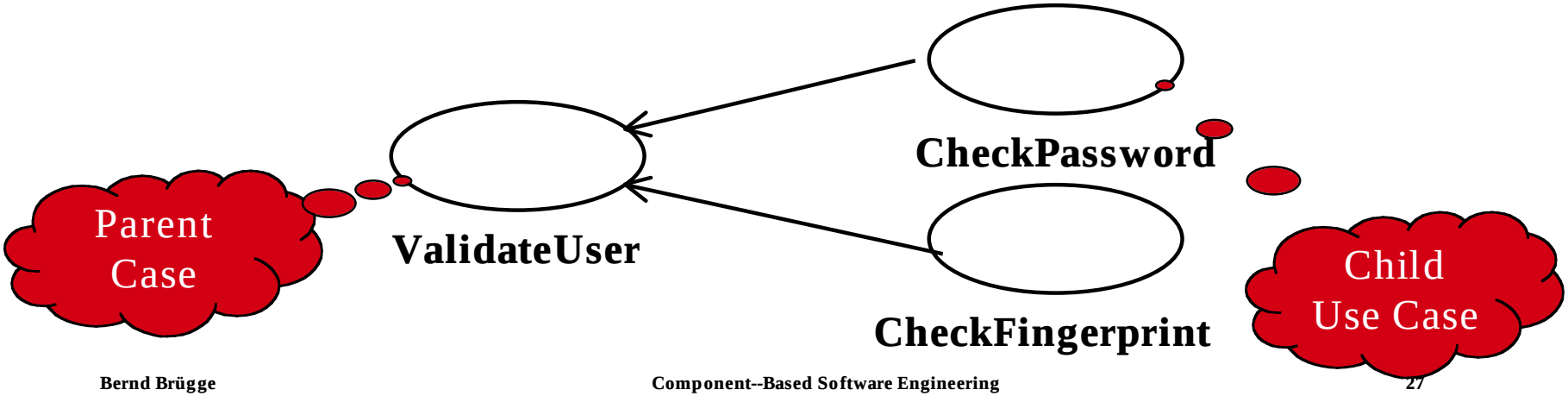
<Extend>> Association for Use Cases

- Problem:
  - **The functionality in the original problem statement needs to be extended.**
- Solution:
  - **An *extend association* from a use case A to a use case B indicates that use case B is an extension of use case A.**
- Example:
  - **The use case "ReportEmergency" is complete by itself , but can be extended by the use case "Help" for a specific scenario in which the user requires help**
- **Note: In an extend assocation, the base use case can be executed without the use case extension**



Generalization association in use cases

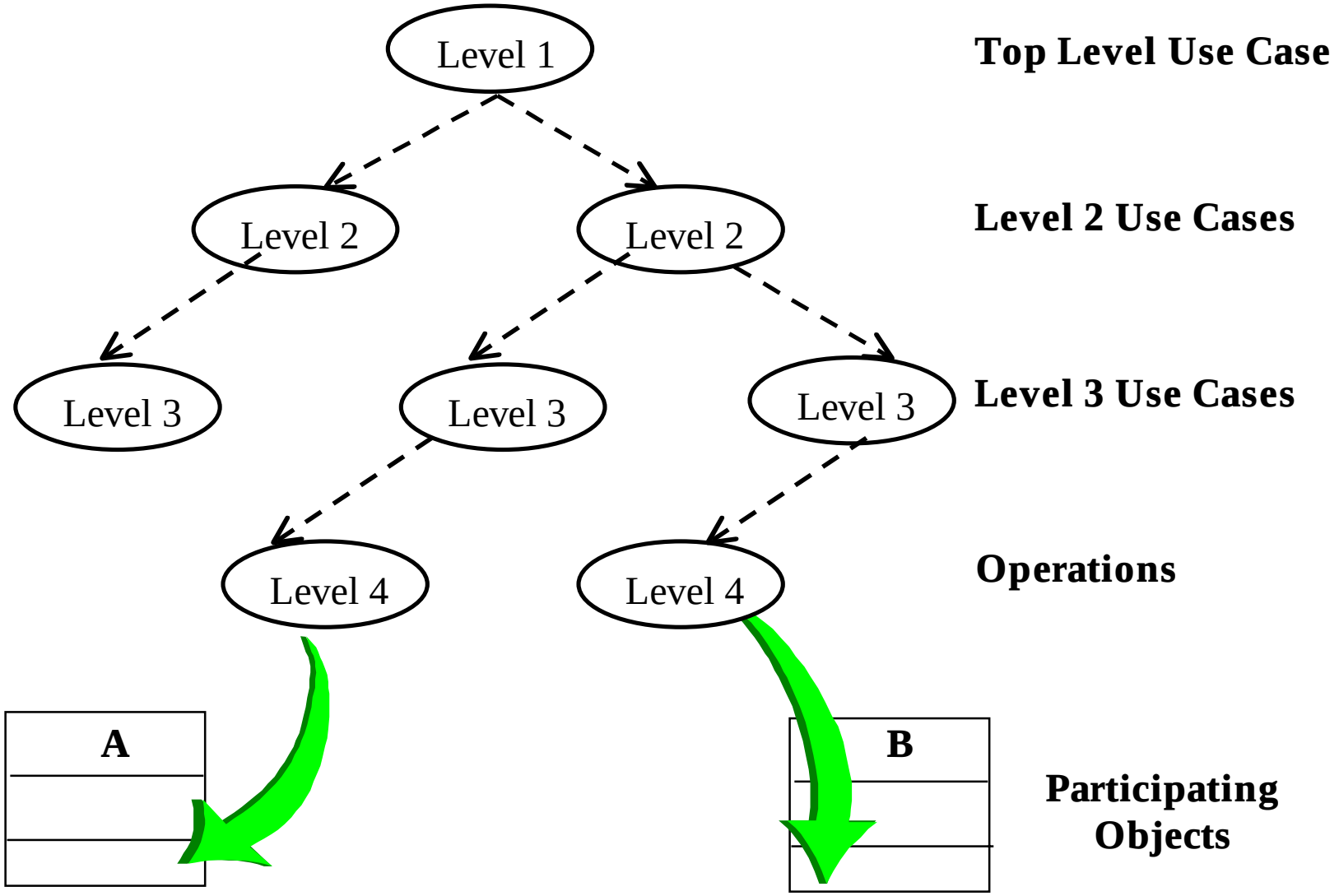
- Problem:
  - **You have common behavior among use cases and want to factor this out.**
- Solution:
  - **The generalization association among use cases factors out common behavior. The child use cases inherit the behavior and meaning of the parent use case and add or override some behavior.**
- Example:
  - **Consider the use case "ValidateUser", responsible for verifying the identity of the user. The customer might require two realizations: "CheckPassword" and "CheckFingerprint"**



How do I find use cases?

- Select a narrow vertical slice of the system (i.e. one scenario)
  - **Discuss it in detail with the user to understand the user's preferred style of interaction**
- Select a horizontal slice (i.e. many scenarios) to define the scope of the system.
  - **Discuss the scope with the user**
- Use mock-ups as visual support
- Find out what the user does
  - **Task observation (Good)**
  - **Questionnaires (Bad)**

From Use Cases to Objects



Finding Participating Objects in Use Cases

- For any use case do the following
  - **Find terms that developers or users need to clarify in order to understand the flow of events**
    - **Always start with the user's terms, then negotiate:**
      - **FieldOfficerStationBoundary or FieldOfficerStation?**
      - **IncidentBoundary or IncidentForm?**
      - **EOPControl or EOP?**
  - **Identify real world entities that the system needs to keep track of. Examples: FieldOfficer, Dispatcher, Resource**
  - **Identify real world procedures that the system needs to keep track of. Example: EmergencyOperationsPlan**
  - **Identify data sources or sinks. Example: Printer**
  - **Identify interface artifacts. Example: PoliceStation**
  - **Do textual analysis to find additional objects (Use Abott's technique)**
  - **Model the flow of events with a sequence diagram**