A Framework for Remote Usability Evaluation on Mobile Devices

Bachelorarbeit in Informatik

Daniel Bader

Introduction

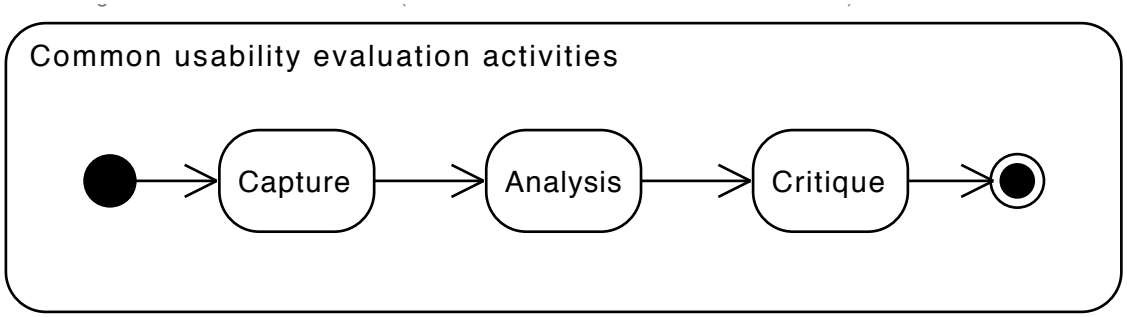
Usability

- An indicator for the ease of use and acceptability of a system

Usability Evaluation

- Methods for determining the usability of a system
- Usability data are collected and analyzed by human evaluators

Common activities (Ivory and Hearst, 2001)



Introduction

Automated Usability Evaluation

- Some or all phases of an Usability Evaluation are automated
- Various degrees of automation exist

Remote Usability Evaluation

- Evaluators and test users are separated in *space* and/or *time*

Automated Remote Usability Evaluation

- One of the "*ultimate goals*" in the field of usability evaluation

Introduction

Usability Evaluation is especially important on mobile devices

- User interfaces on mobile devices became more complex
- Mobile context creates new challenges:
    - different environments, distractions, ...

Current mobile devices have a rich ability to track their surroundings

- Microphone, GPS, accelerometer, video camera, ...

Usability Evaluation can be (partly) automated

Problem statement

A lack of software support for Usability Evaluation on current mobile platforms

- No automation support in existing tools

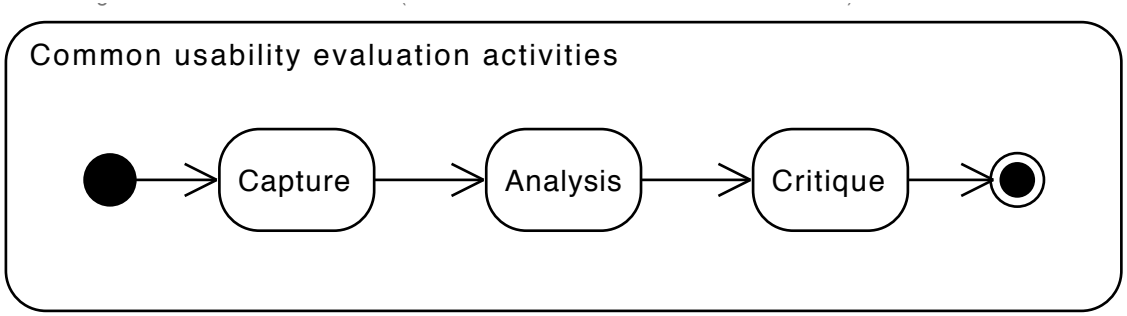
- No support for built-in tracking capabilities (microphone, accelerometer, GPS, ...)

- No support for collecting user interactions graphically (screenshots, video)

Proposed solution

The **muEvaluationFramework** (mobile usability Evaluation Framework)

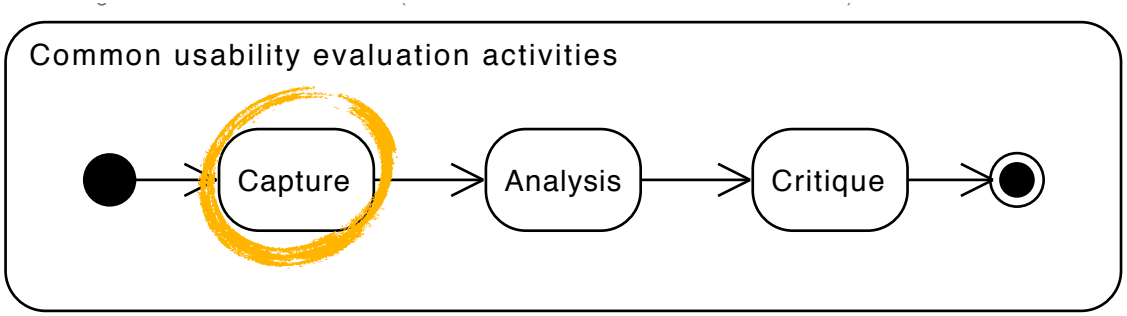
An application-independent software framework for remote usability evaluation on mobile platforms which supports automation in all phases of a usability evaluation.



Outline

- ✓ **Introduction**
- ✓ **Problem statement**
- **Proposed solution**
  - Requirements specification
  - System design
  - Object design
- **Prototype demo**
- **Future work**

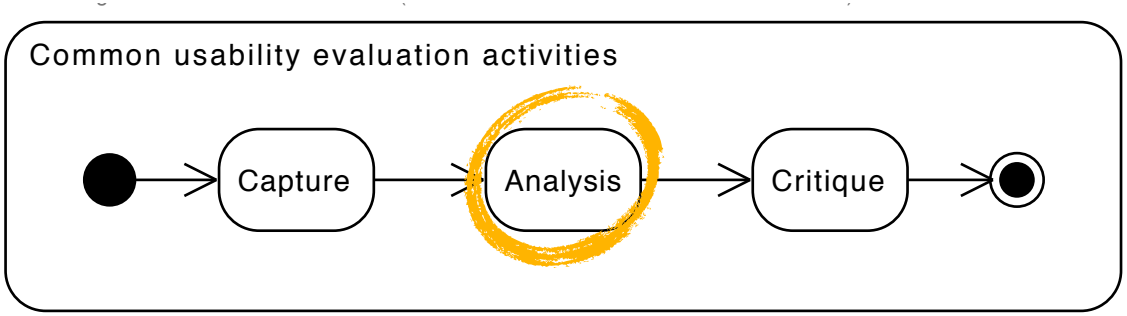
Functional requirements



Capture phase

- Session-based usability data collection
- Support for multiple usability-data sources
  - User input events
  - Application events
  - Device sensor events
- Live preview for collected events

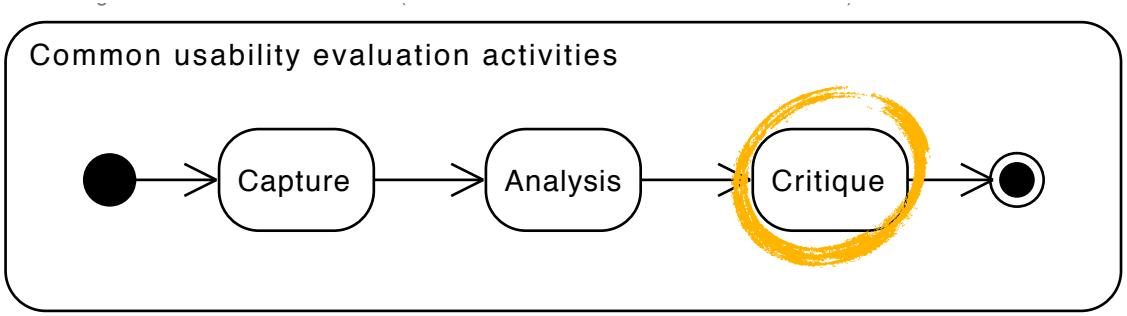
Functional requirements



Analysis phase

- Interpretation of the captured data
- Detection of usability problems
  - e.g. Human Interface Guidelines (HIG) violations
- Usability summary generation
  - Most used views and UI widgets
  - Heat maps for user input events
  - ...

Functional requirements



Critique phase

- Report generation
  - Reports summarize the interpretation results
  - Reports can be viewed using a web browser
- Report configuration
  - Selection of the report sections

Nonfunctional requirements

Minimal setup work required by the developer

- Low entry barrier
- Quick setup for existing projects
- Goal: "*link against one library and make one method call*"

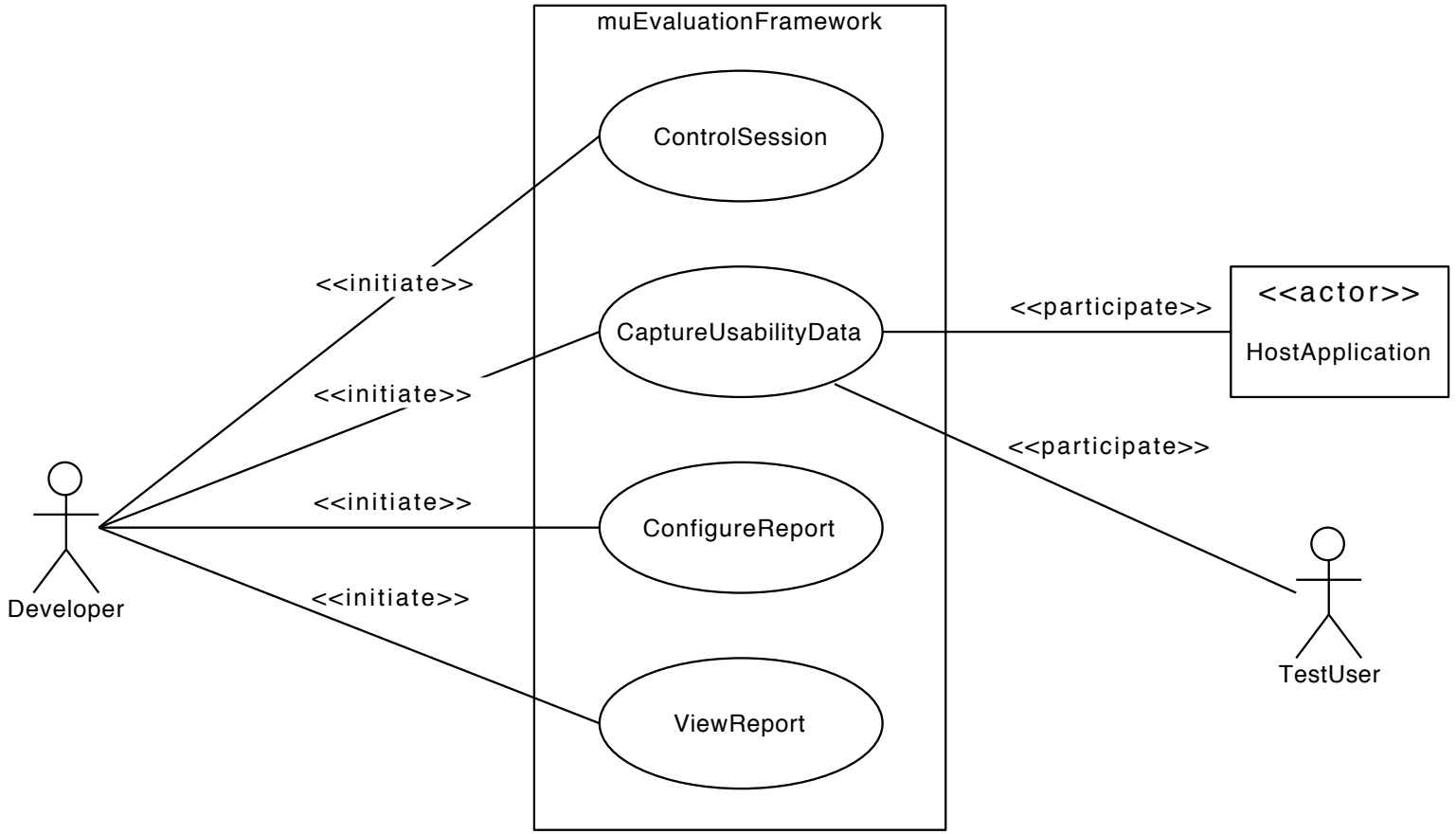
Wireless communication with the mobile device

- Test users can move freely and do not feel restricted
- Evaluation can be performed outside of a laboratory

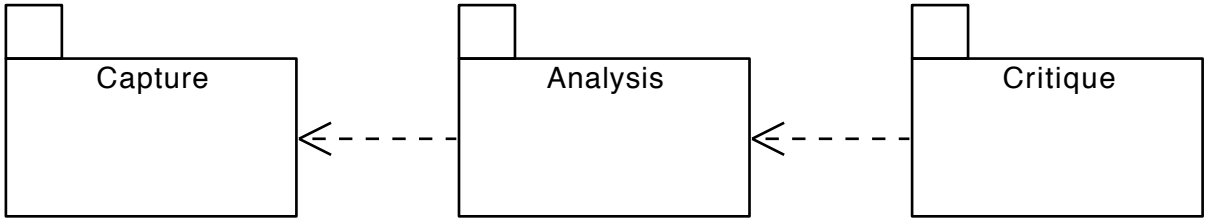
Extensibility

- Easily add new sensors, interpreters, and sections
- Framework intended as a research platform

Use case model

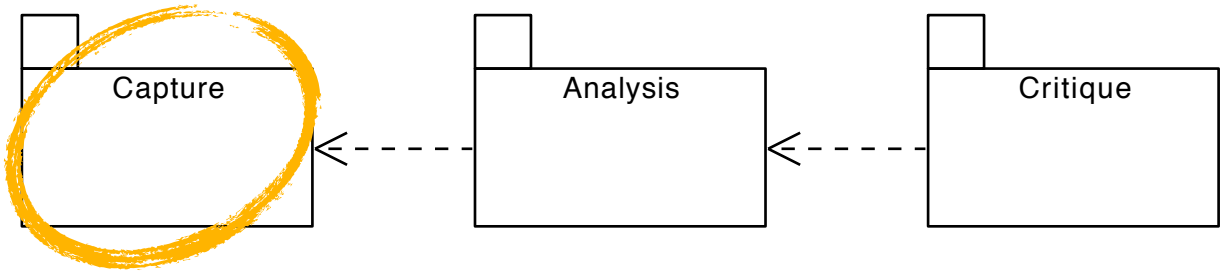


Object model and Dynamic model



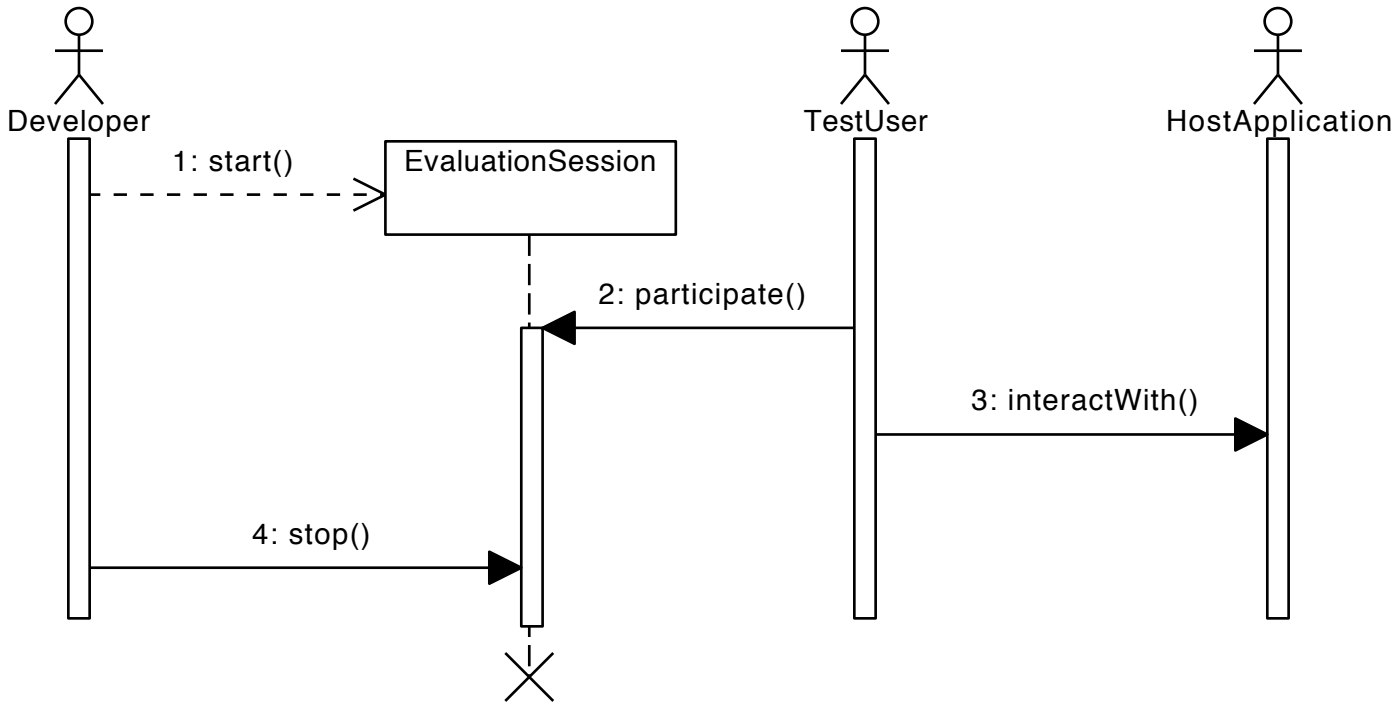
Conceptual sub-models for each of the three phases

Object model and Dynamic model



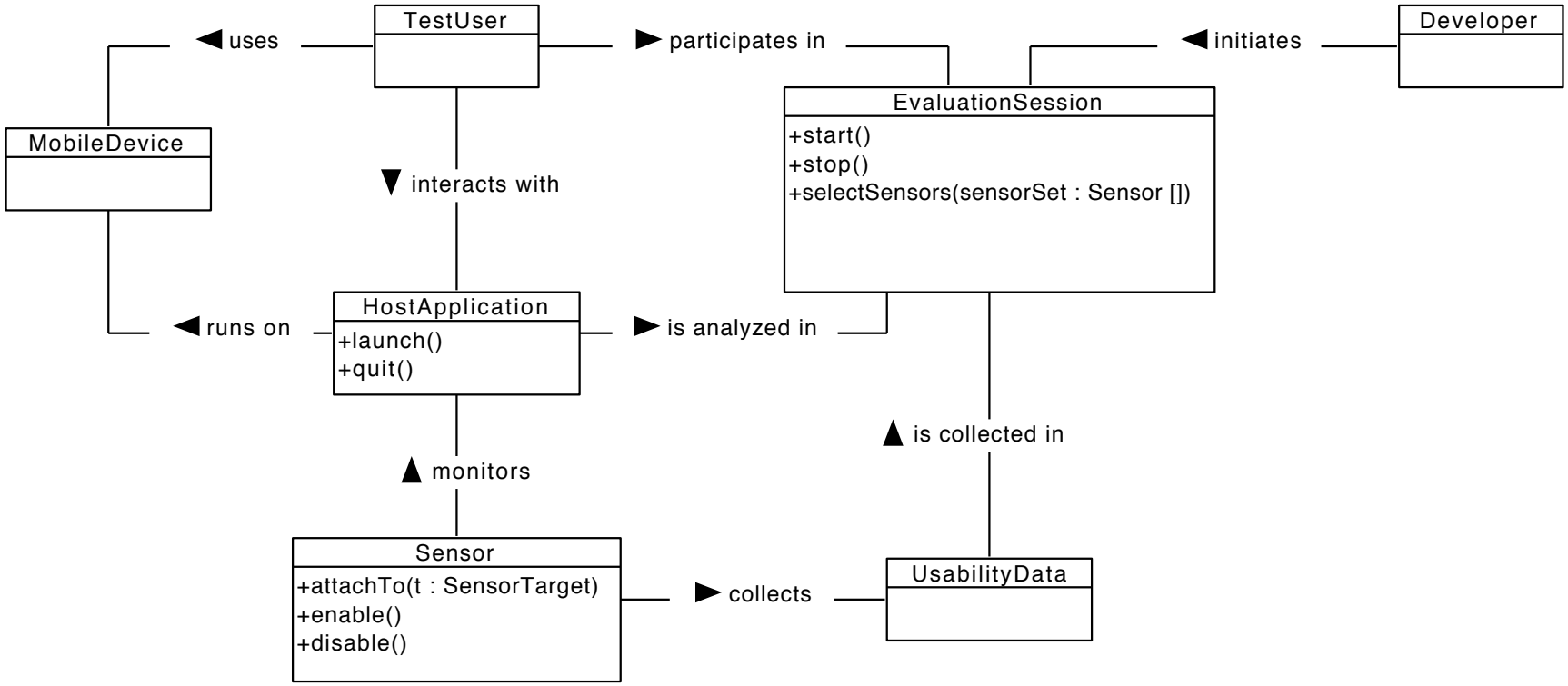
Conceptual sub-models for each of the three phases

Capture phase

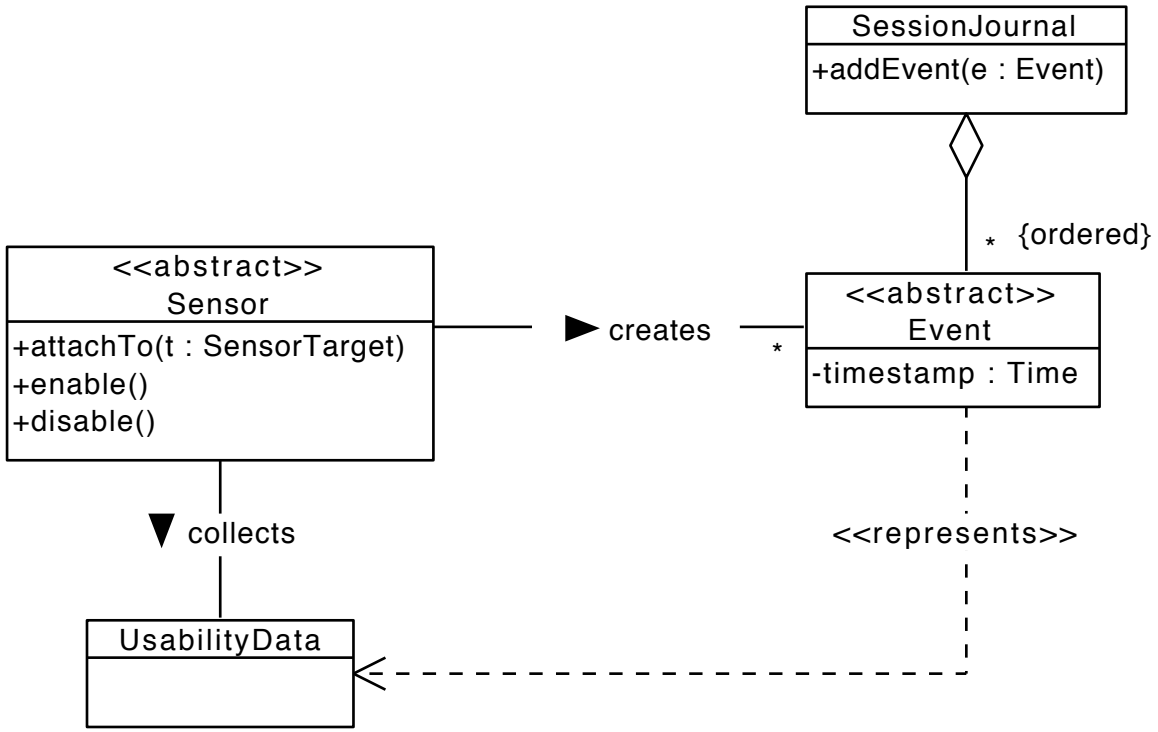


The life cycle of an **EvaluationSession**

Capture phase

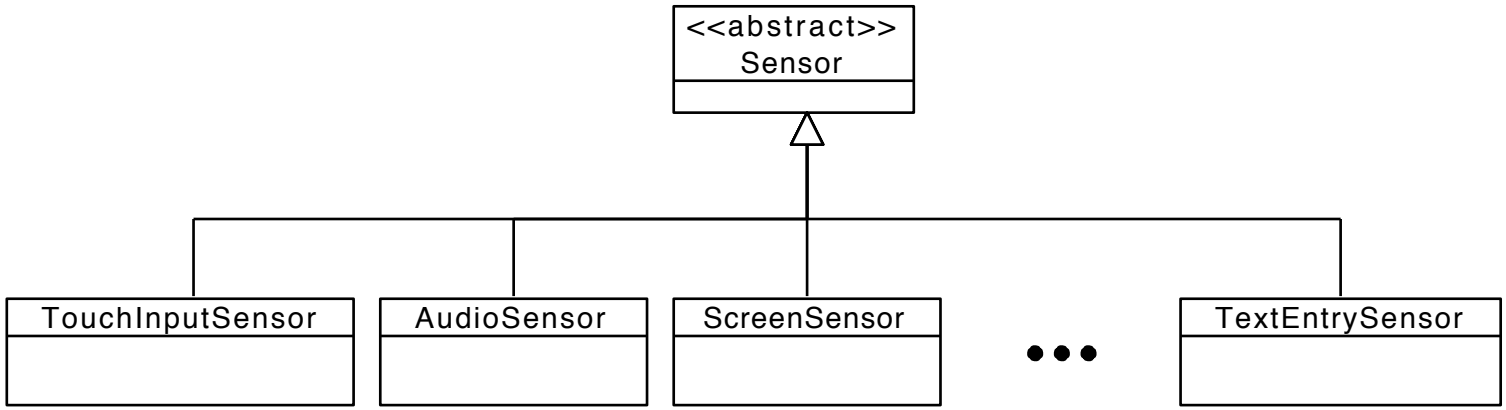


Capture phase



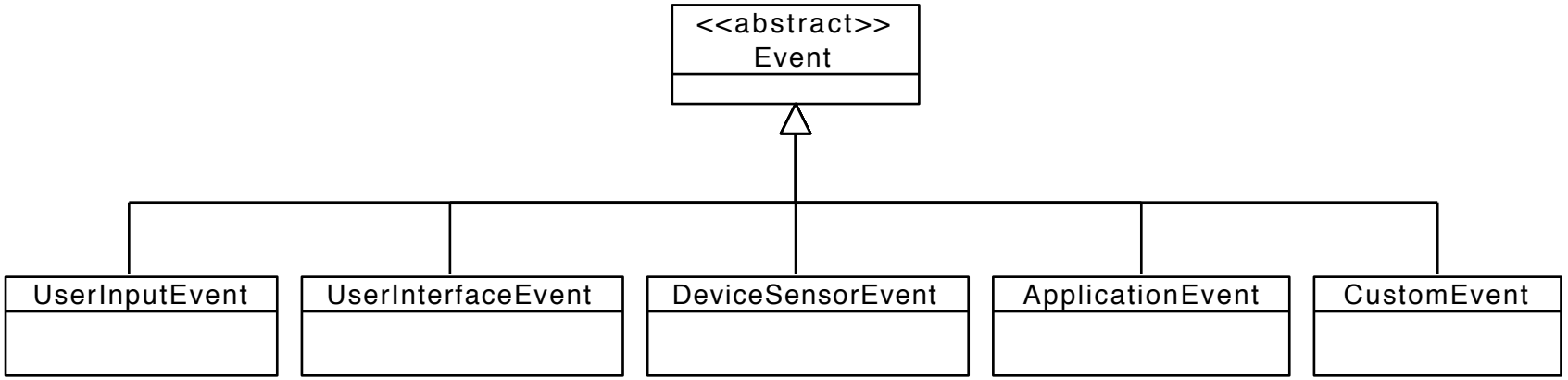
Sensors and **Events**

Capture phase



Sensor specializations

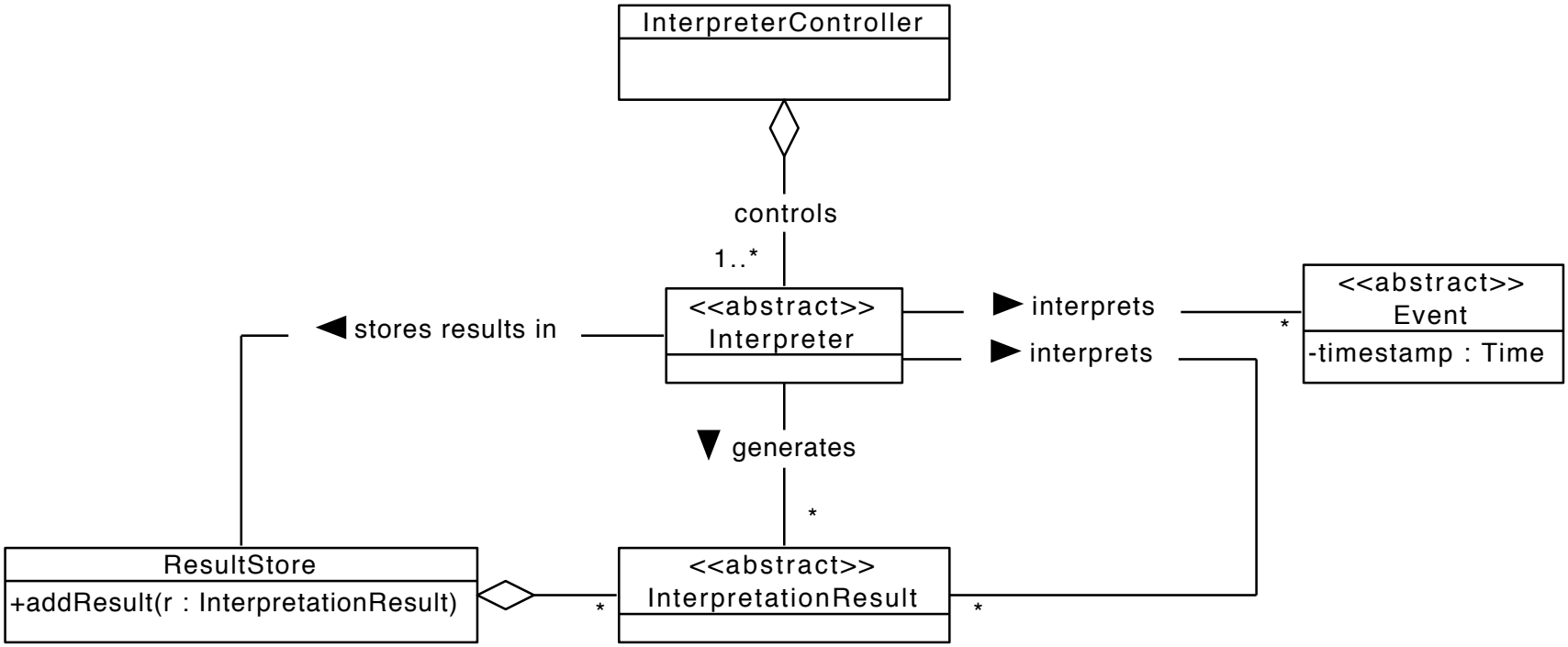
Capture phase



Event specializations

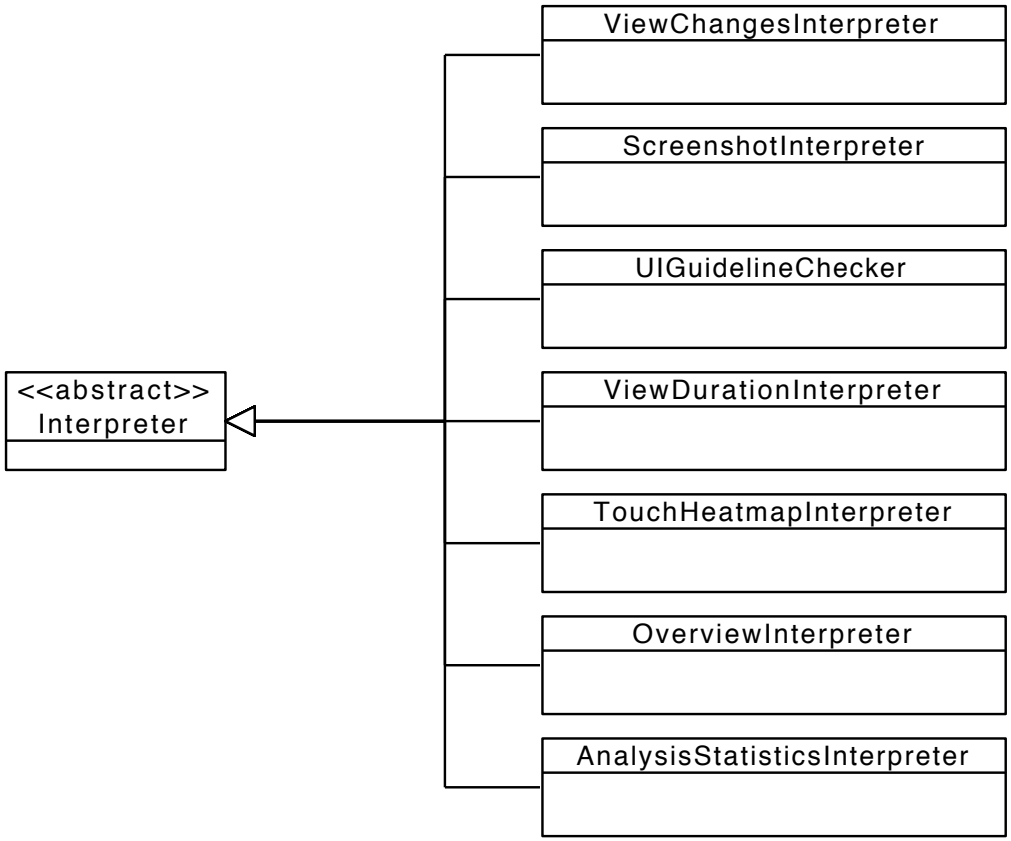
Object model and Dynamic model

Analysis phase



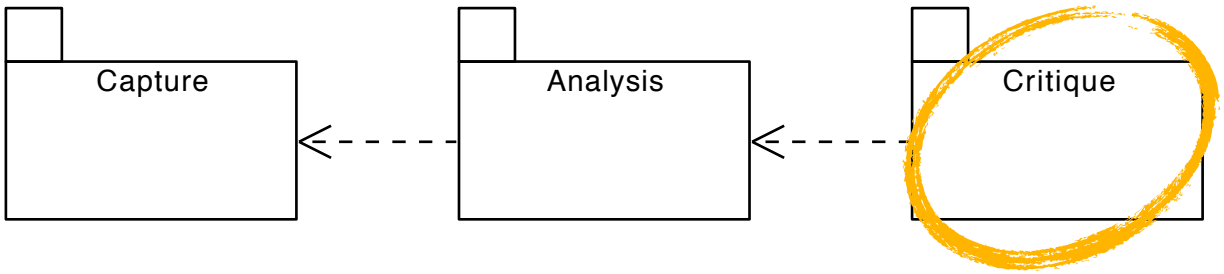
Interpreters work on data collected in the *Capture* phase and generate **InterpretationResults.**

Analysis phase

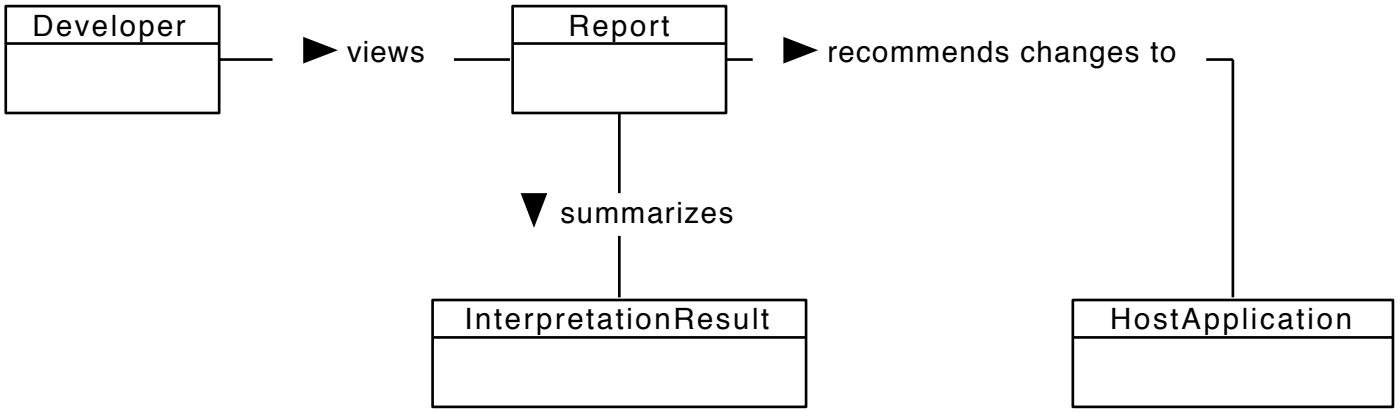


Interpreter specializations

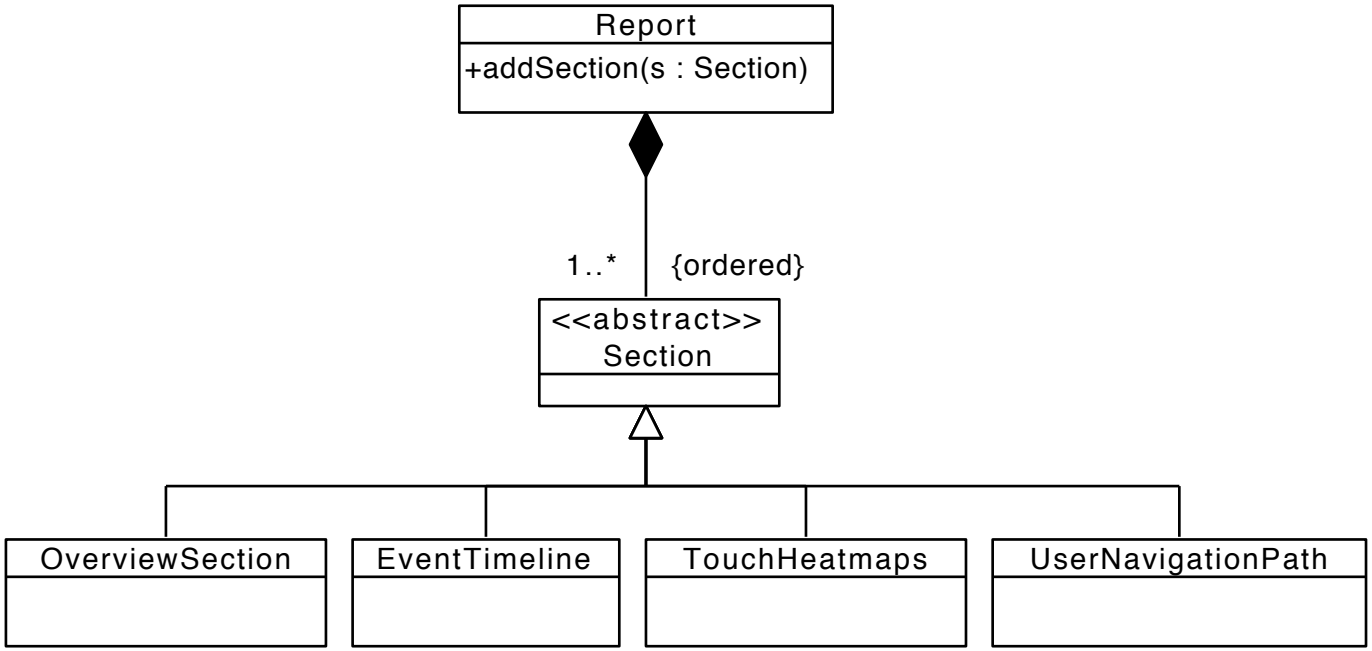
Object model and Dynamic model



Critique phase

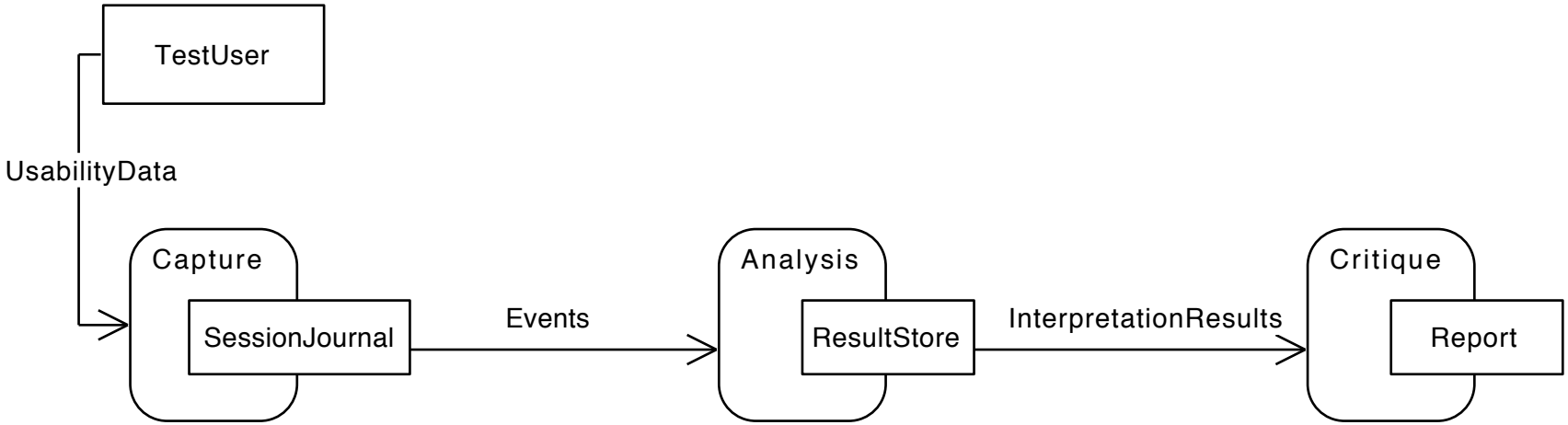


Critique phase



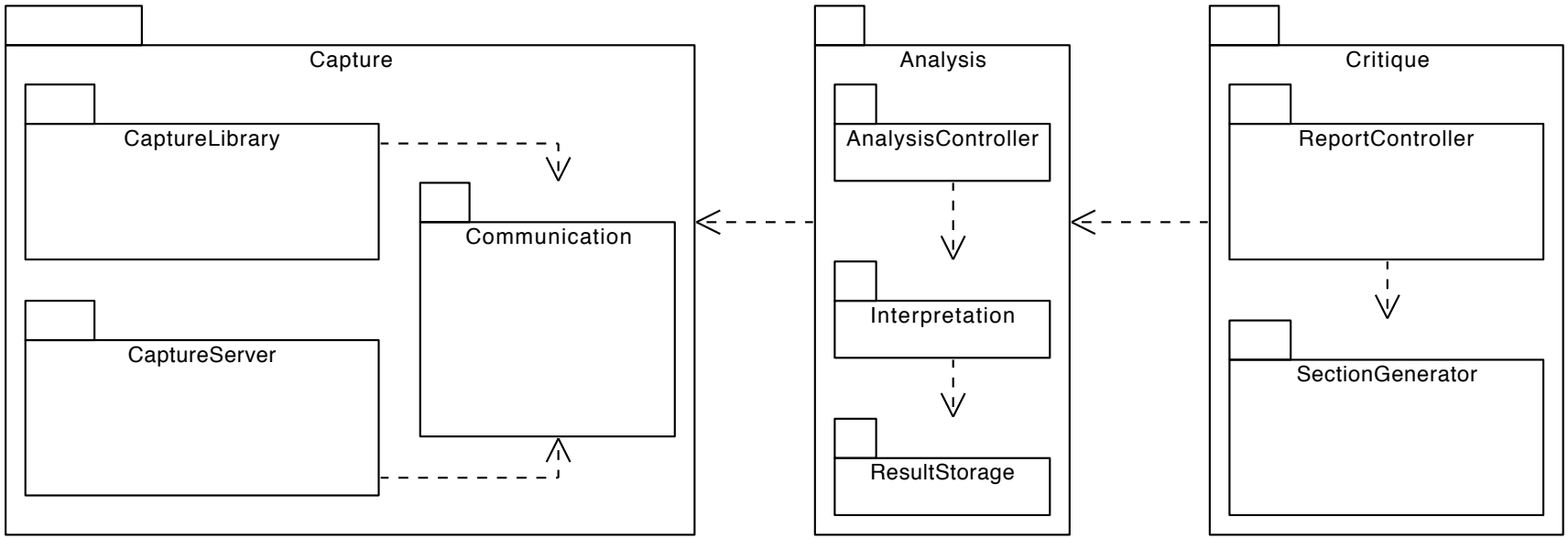
Report document model

High-level dynamic model



Inputs and **outputs** of the three phases

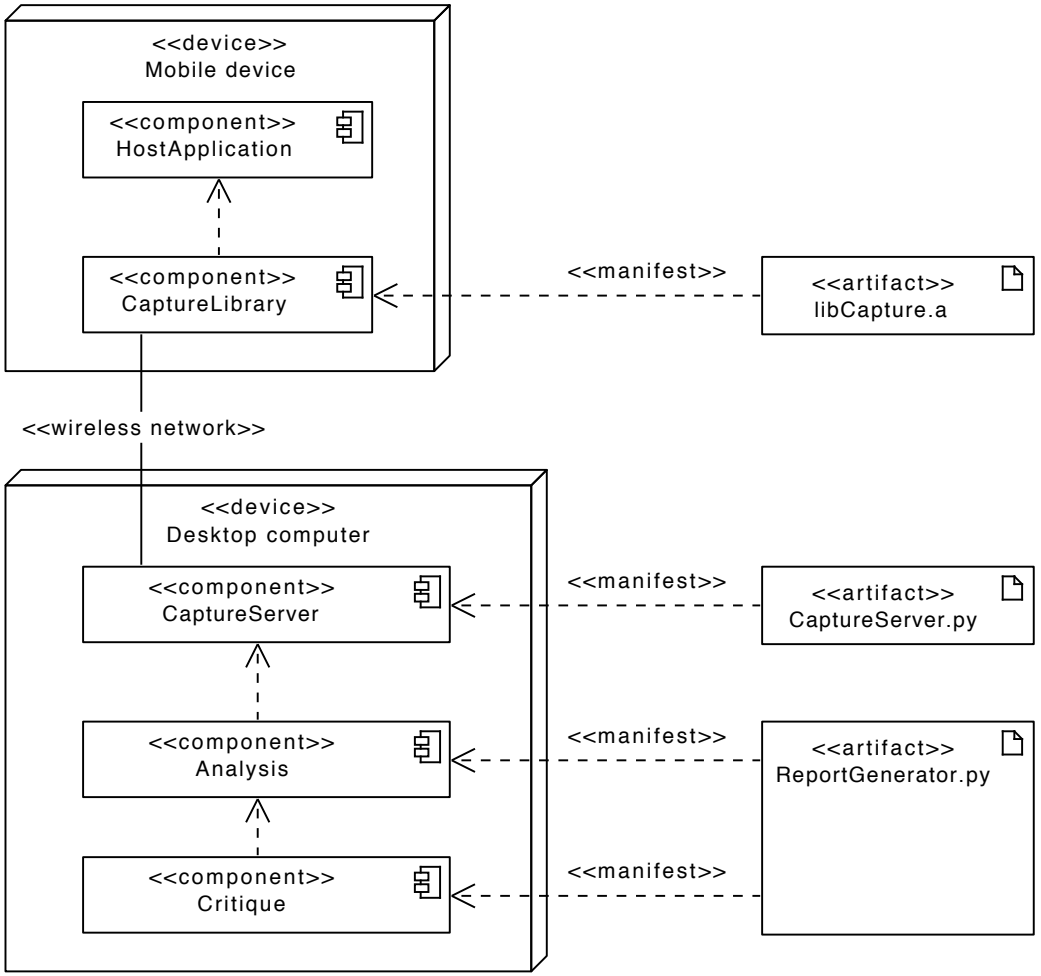
System design



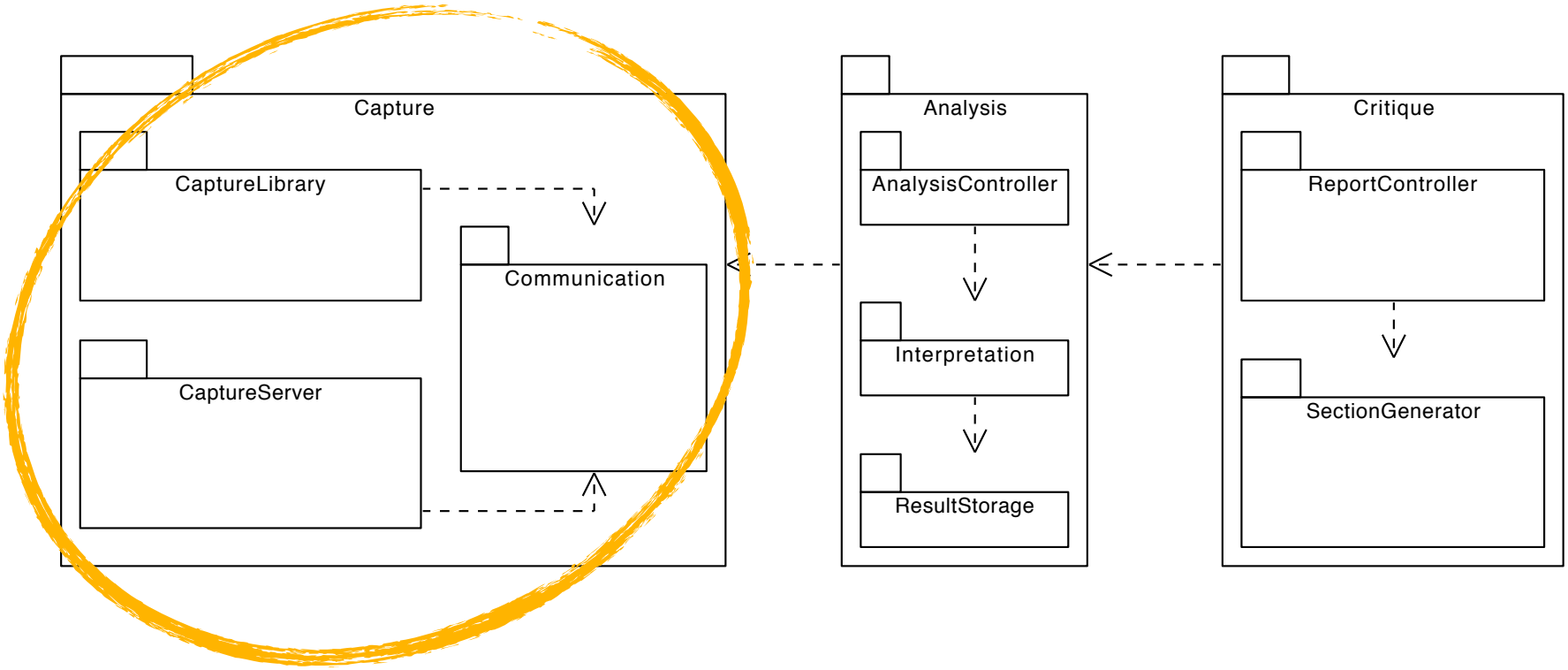
Subsystem decomposition of the framework

System design

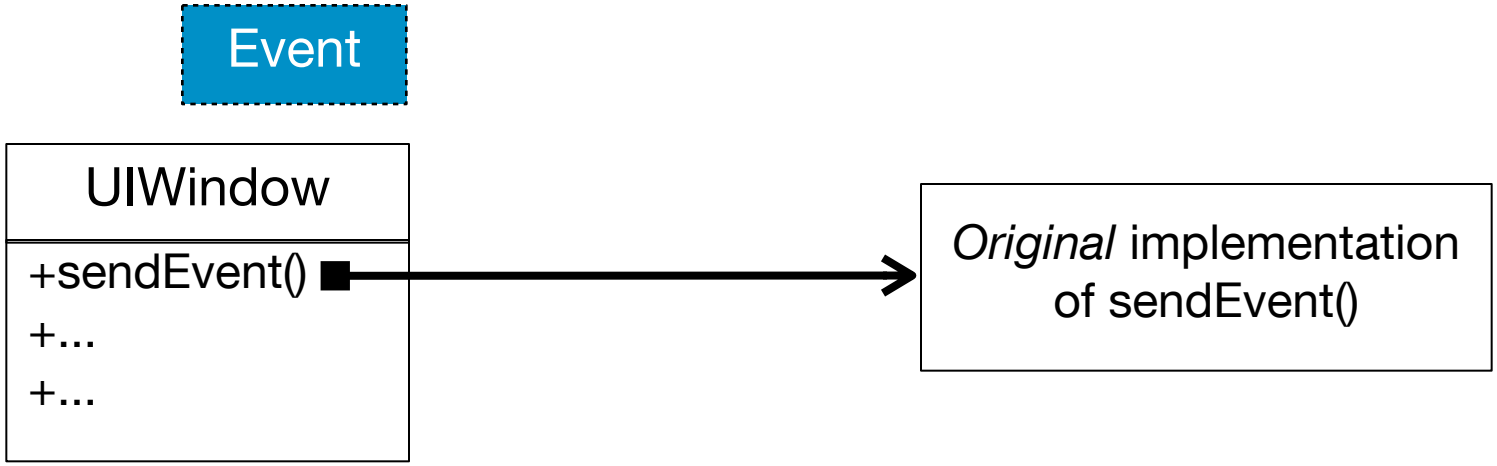
Deployment of the framework



Object design

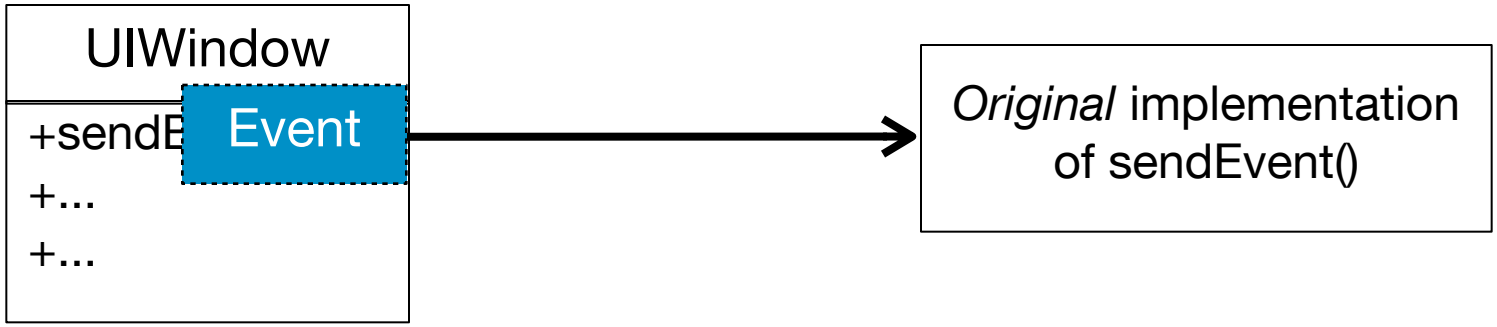


CaptureLibrary - Method interception



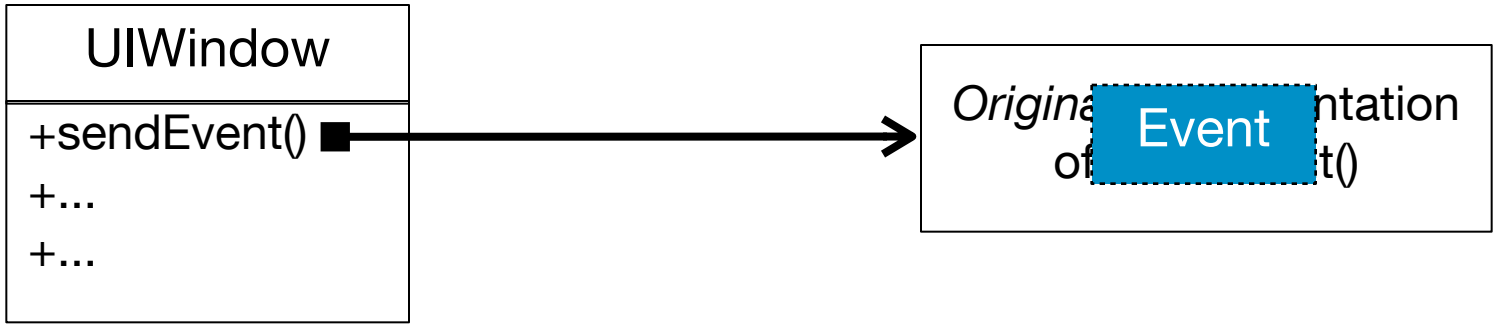
UIWindow sendEvent() behavior *before* method interception

CaptureLibrary - Method interception



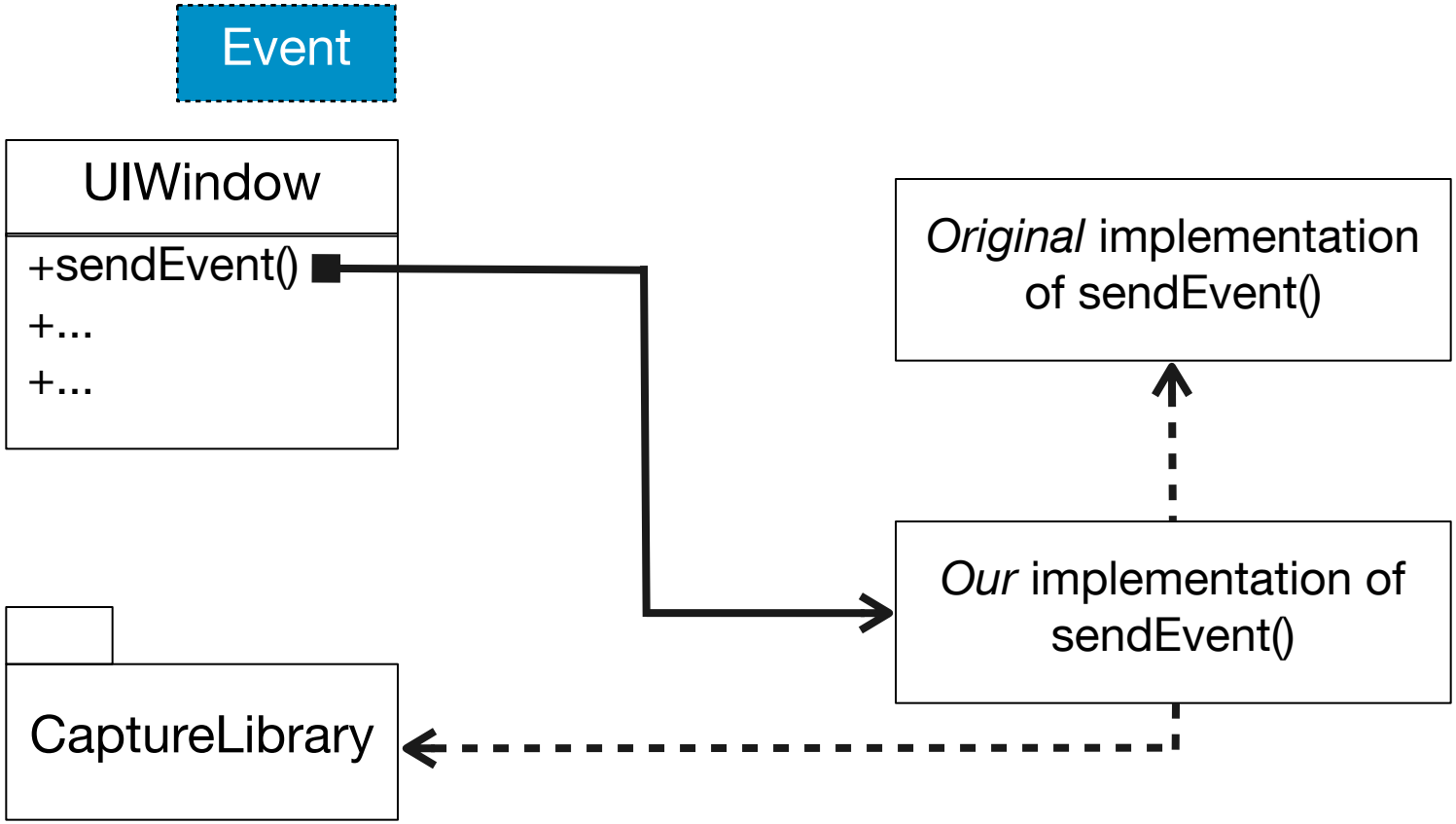
UIWindow sendEvent() behavior *before* method interception

CaptureLibrary - Method interception



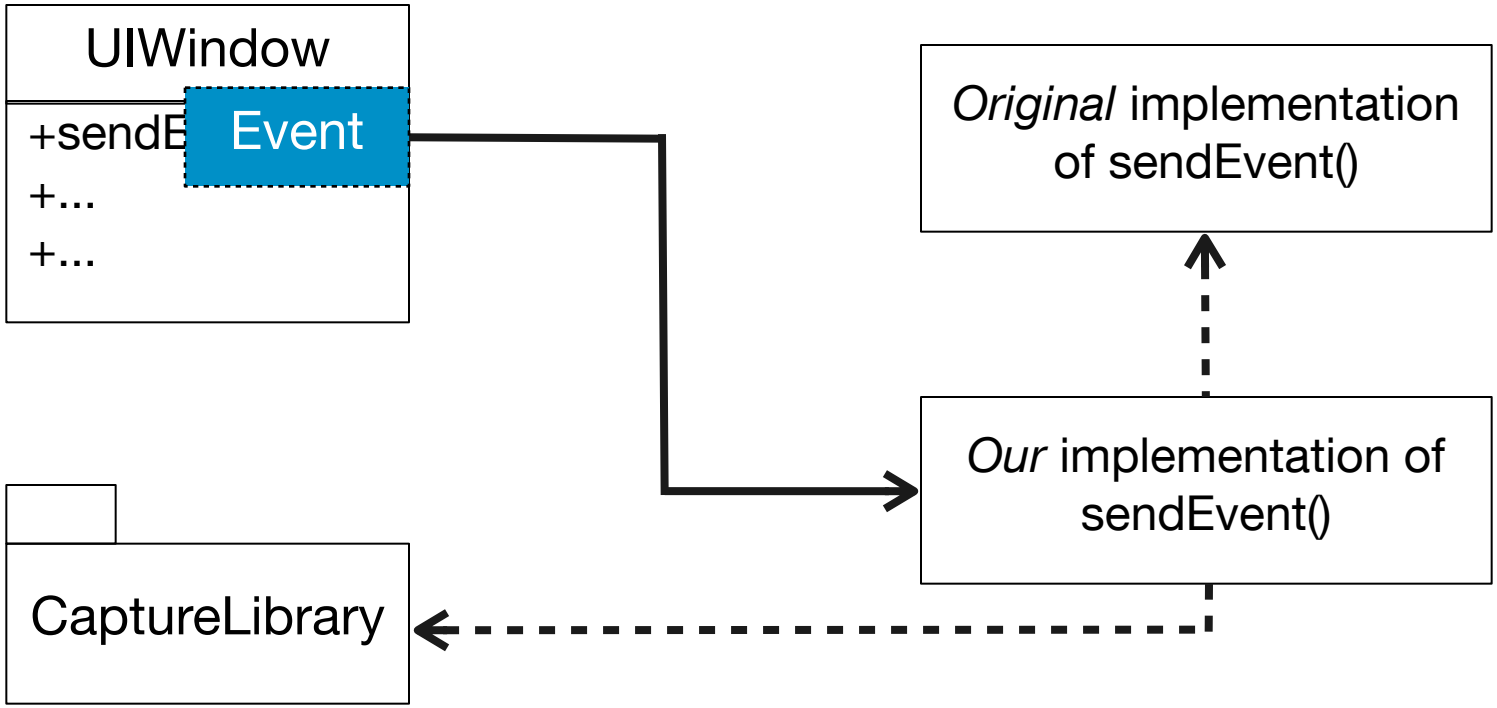
UIWindow sendEvent() behavior *before* method interception

CaptureLibrary - Method interception



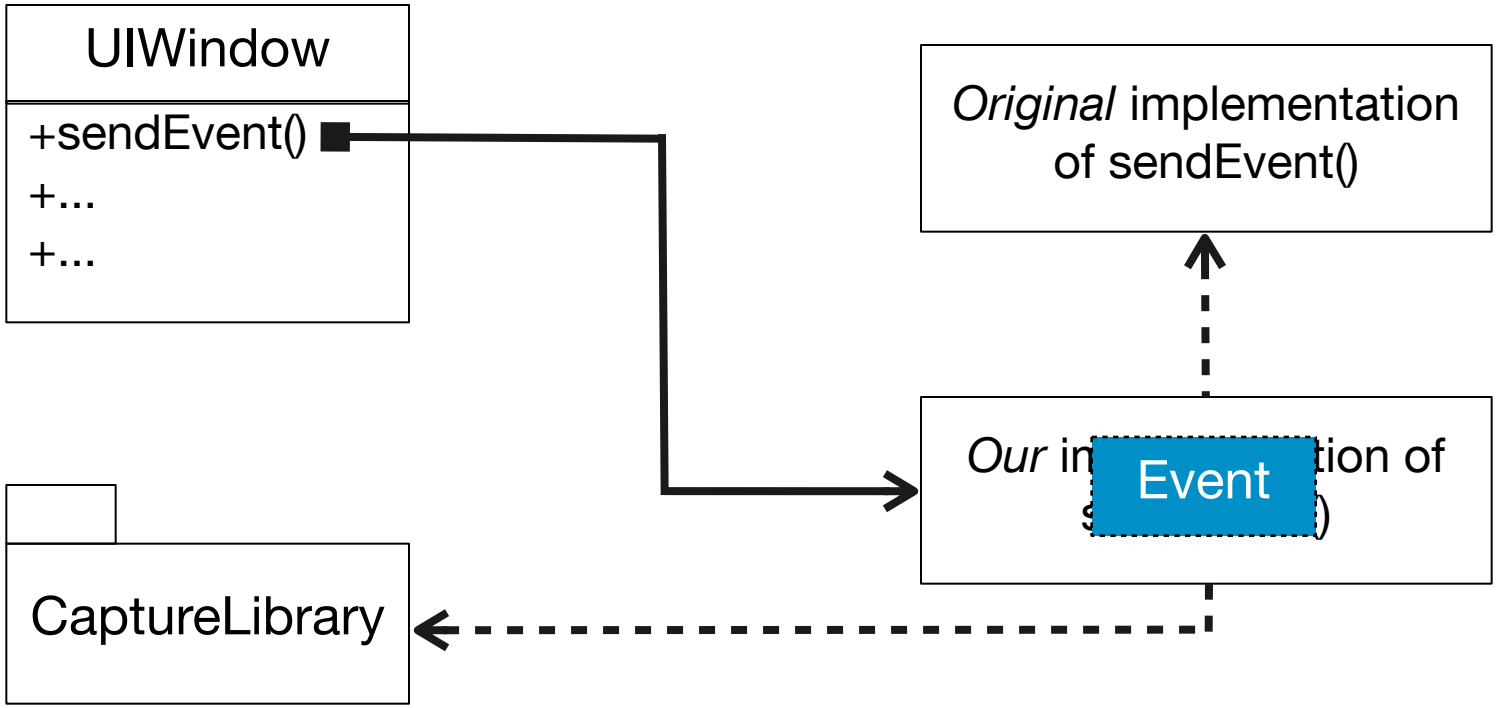
UIWindow sendEvent() behavior *after* method interception

CaptureLibrary - Method interception



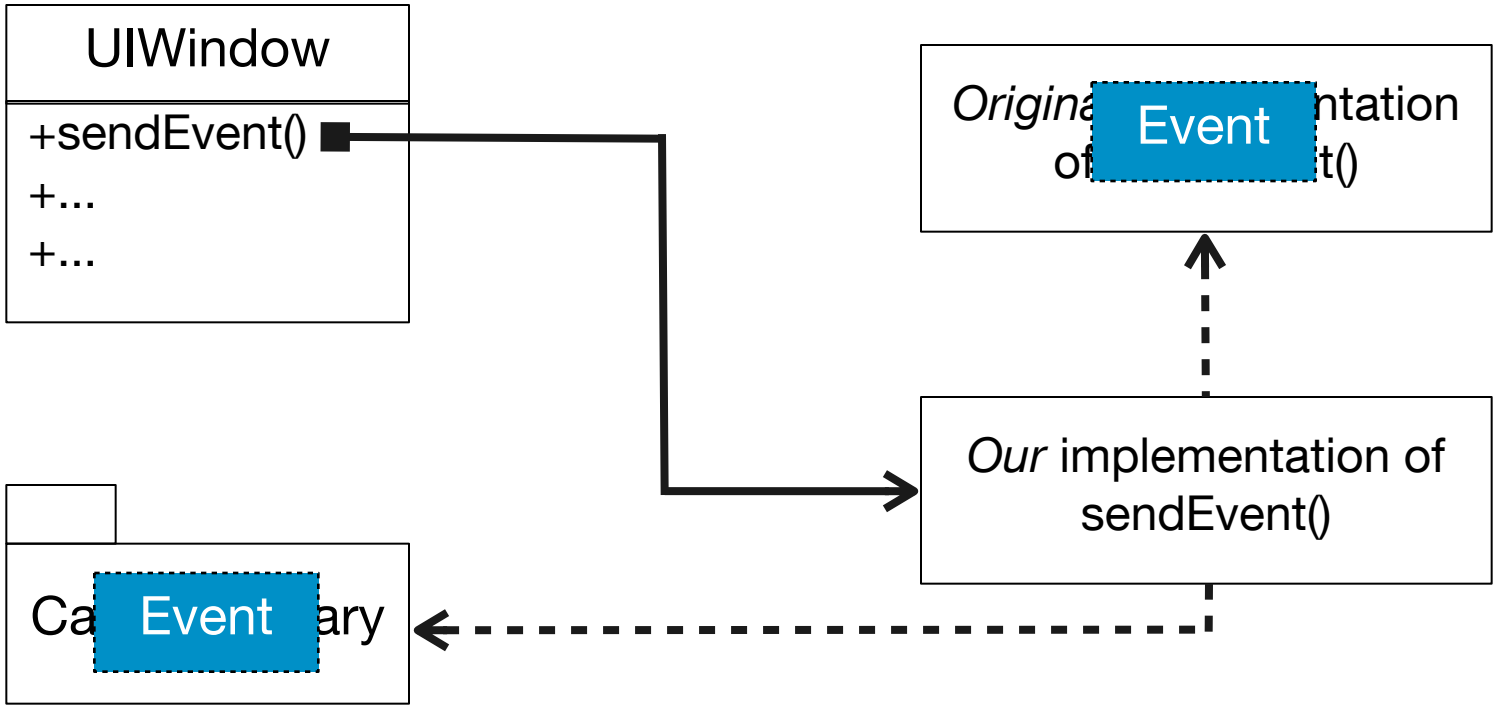
UIWindow sendEvent() behavior *after* method interception

CaptureLibrary - Method interception



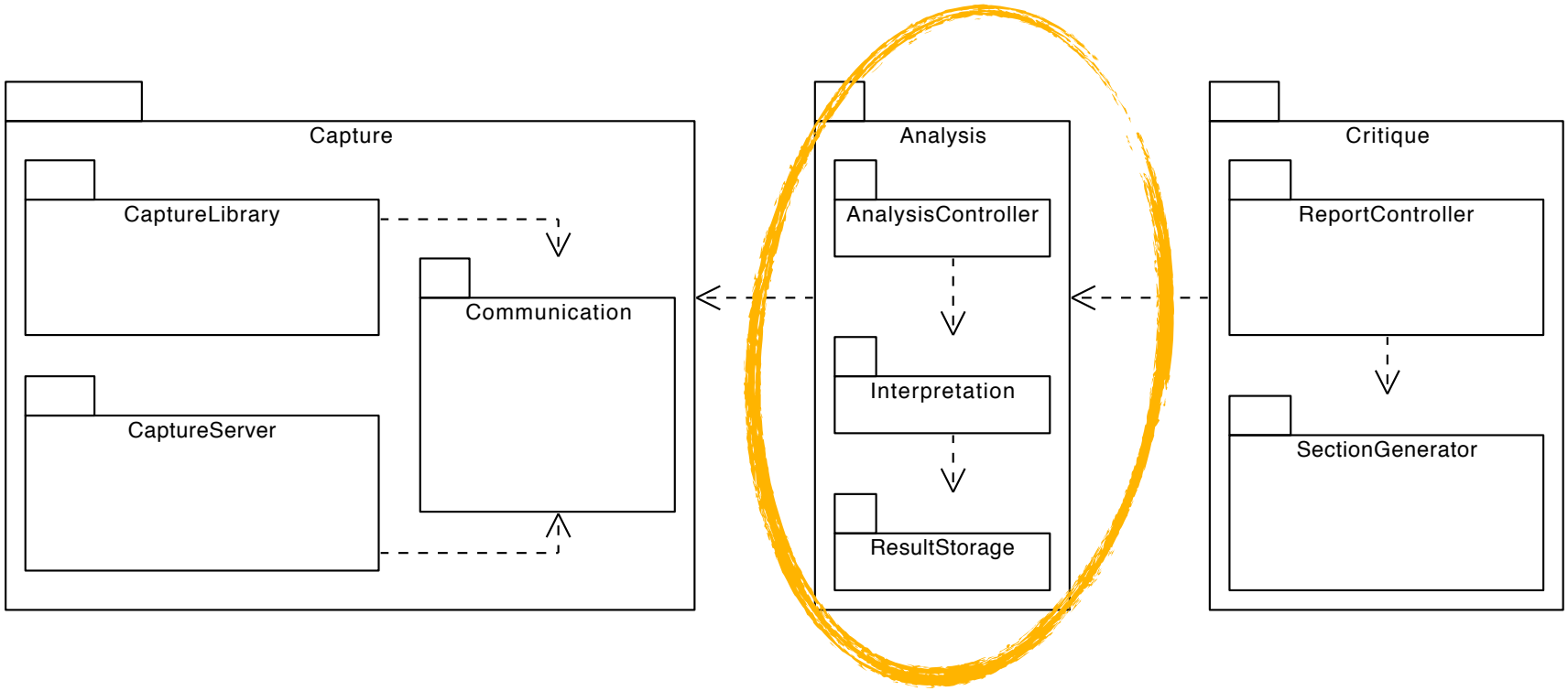
UIWindow sendEvent() behavior *after* method interception

CaptureLibrary - Method interception

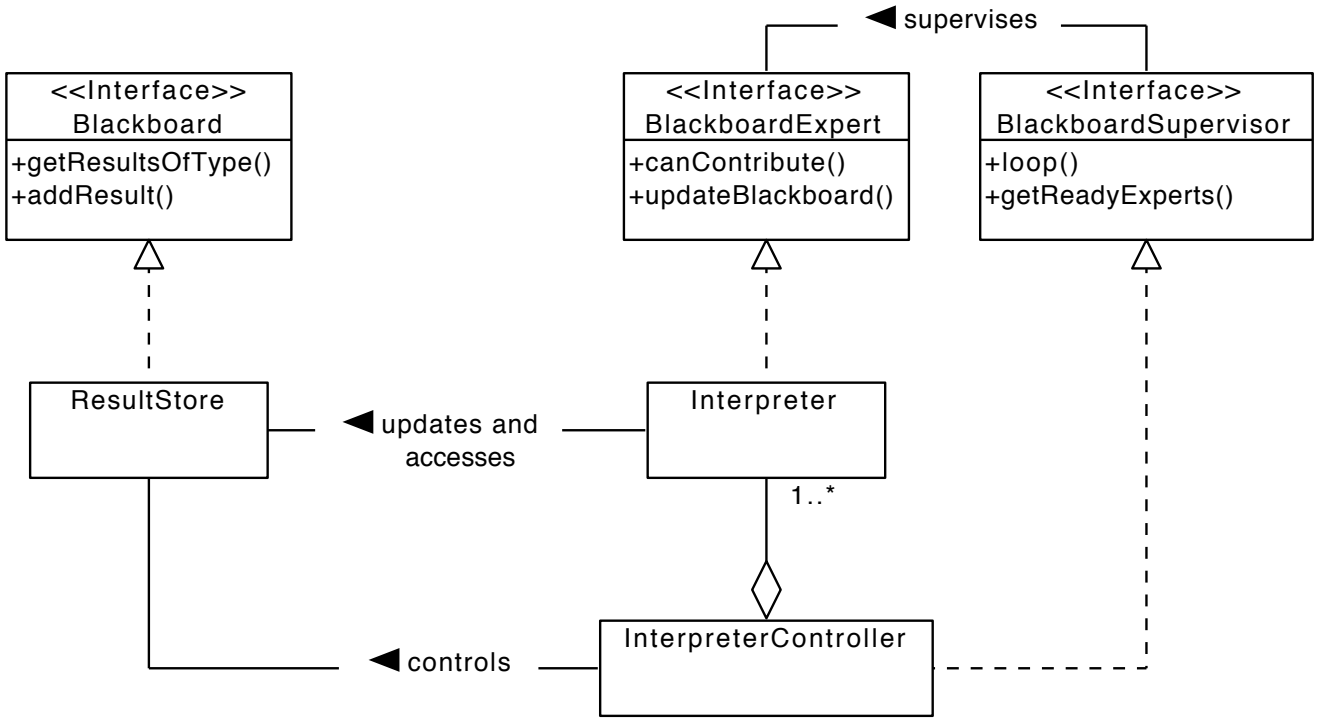


UIWindow sendEvent() behavior *after* method interception

Object design



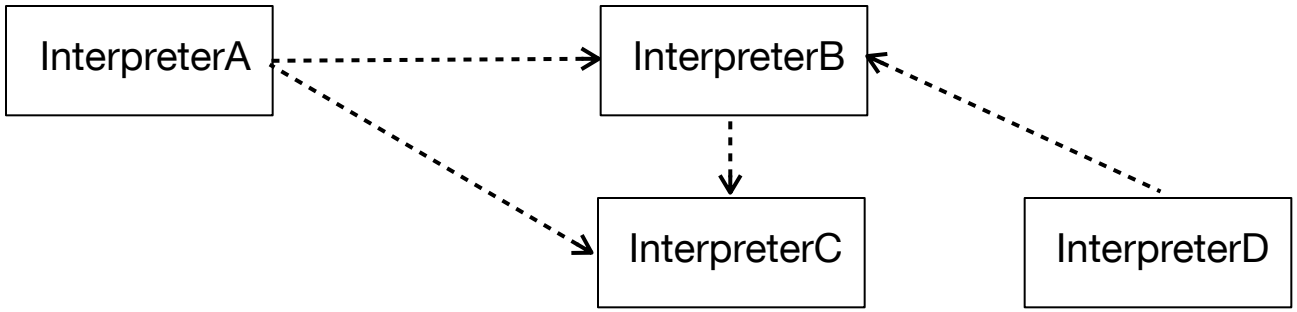
Analysis subsystem



Usage of the **Blackboard design pattern** in the Analysis subsystem

Analysis subsystem

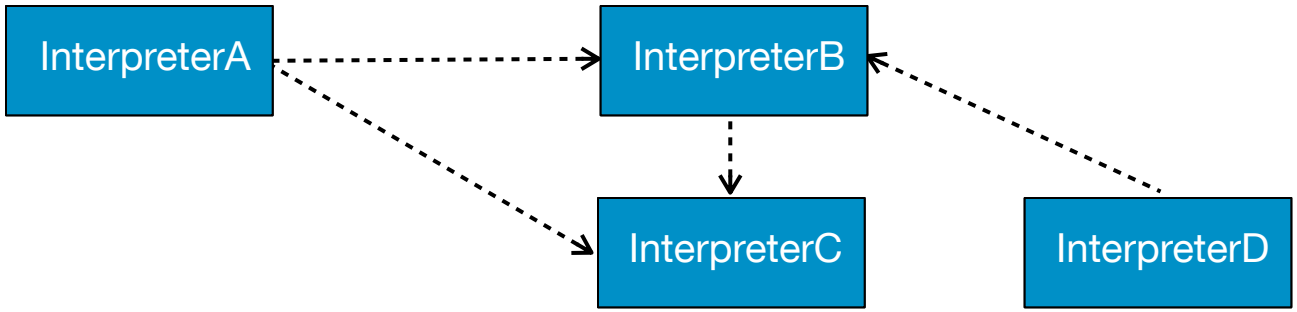
Dependencies between **Interpreters** are resolved automatically



Generated order of execution:

Analysis subsystem

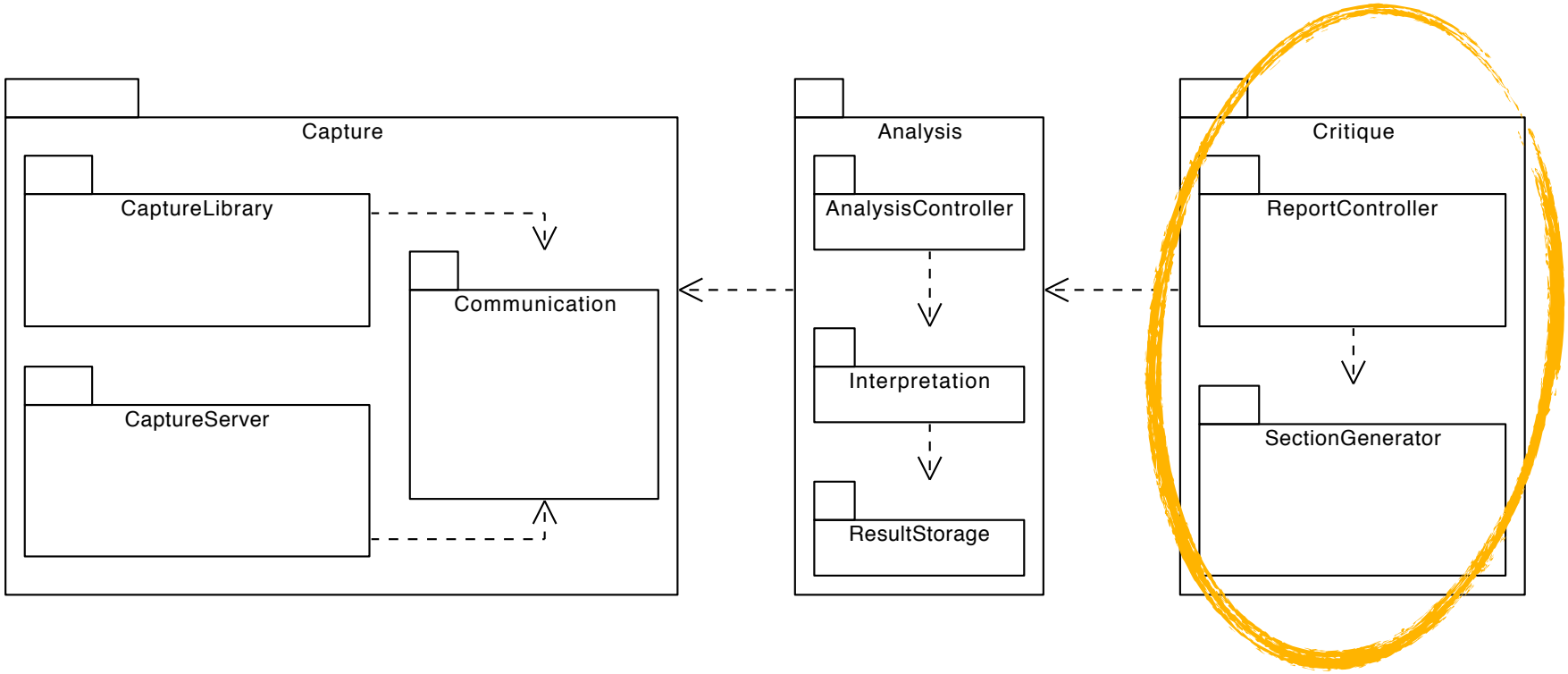
Dependencies between **Interpreters** are resolved automatically



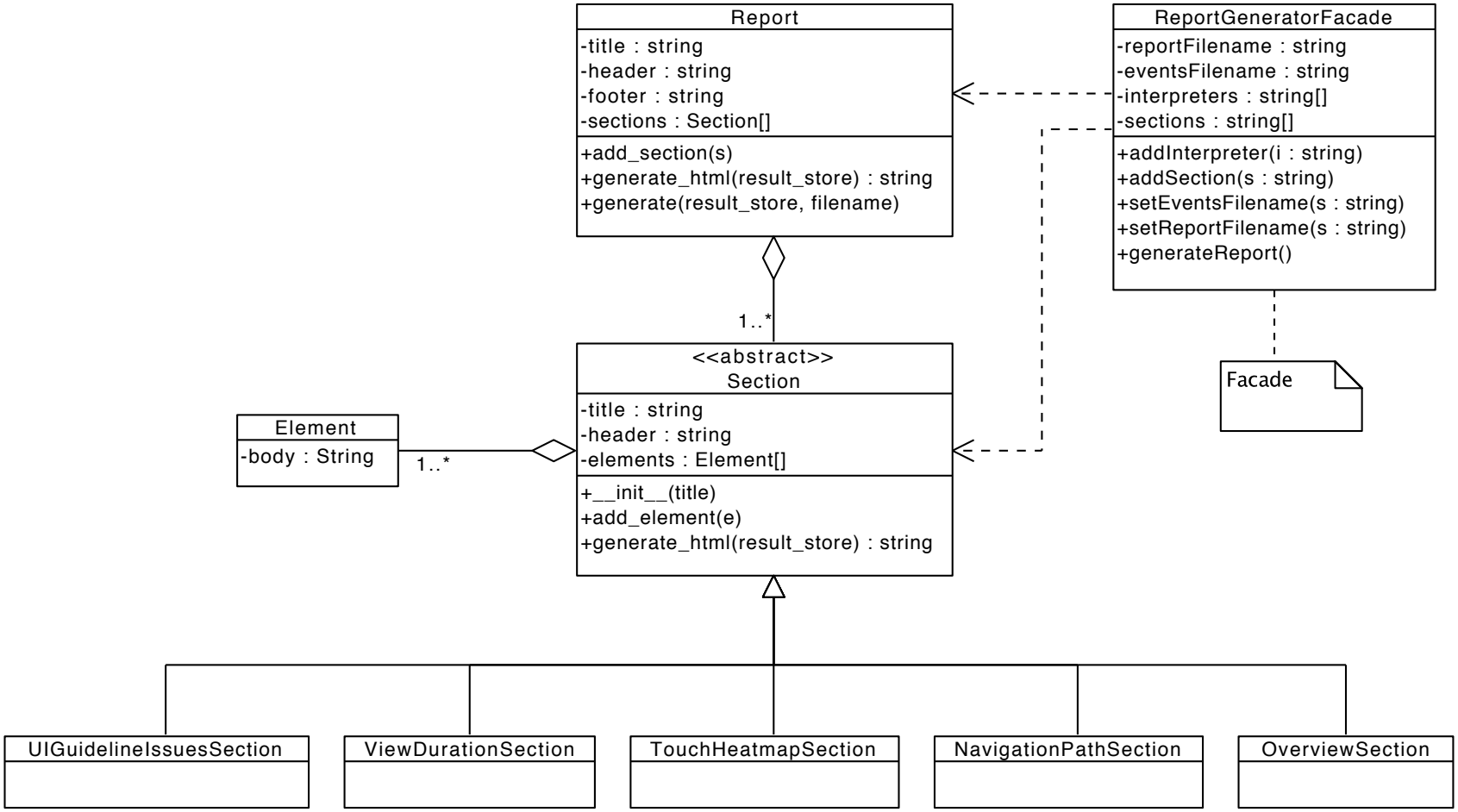
Generated order of execution:

{C} → {B} → {A, D}

Object design



Critique subsystem



Prototypical implementation

Full vertical prototype for Apple iOS

- Implemented in Objective-C and Python
- Tested with the applications *Wordpress for iOS* and *PlainNote*

Capture support

- Automatic network setup using Zeroconf/Bonjour
- Custom, stateless XML-based network protocol
- Data transmission is performed on background thread

Analysis support

- Six interpreters were implemented

Critique support

- Generates HTML output
- Five section types were implemented

Demo

Future work

Add more sensors and interpreters

- e.g. audio and video recording

Use the framework in a real world scenario

Add support for Google Android

- Java has rich support for reflection

Add ability to playback sessions

- Play back captured user interactions on the device

Integrate framework in FastFix

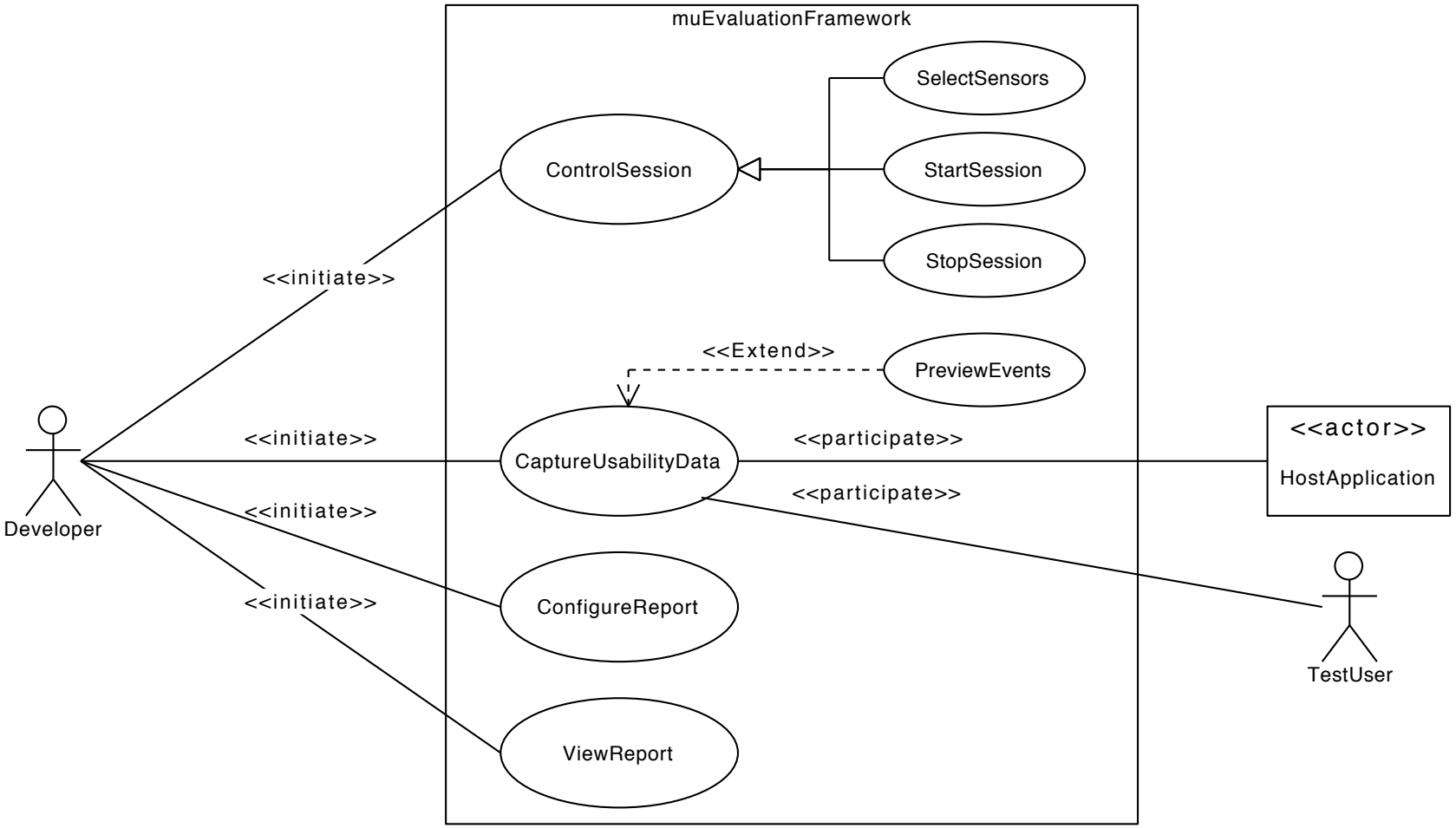
- Perform usability error detection
- Provide access to user interaction data and context information

Thank you!

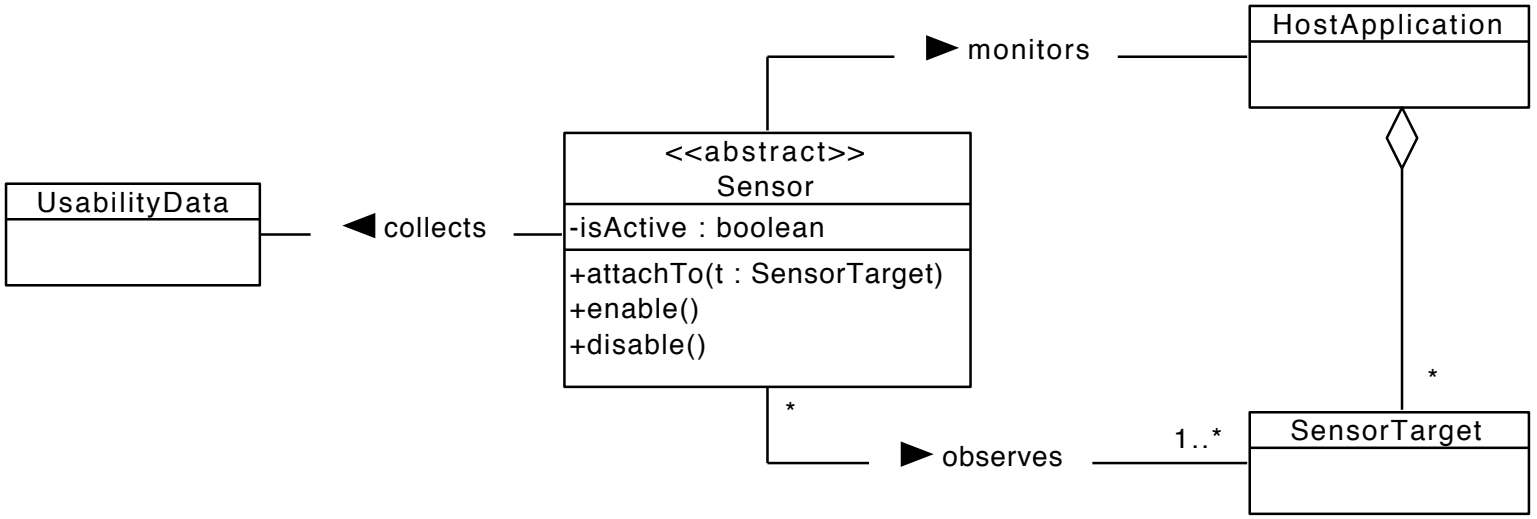
Questions?

Backup slides

Use case model (with refinements)

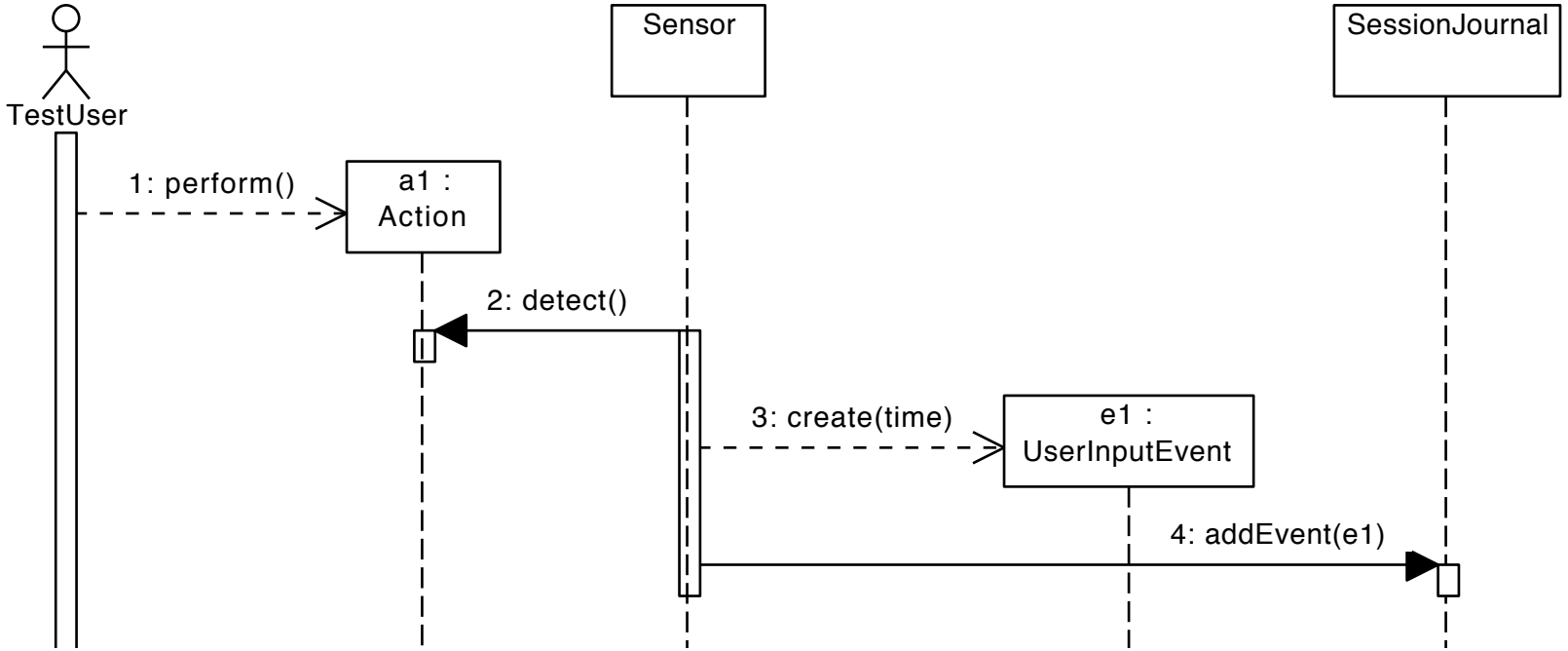


Capture phase



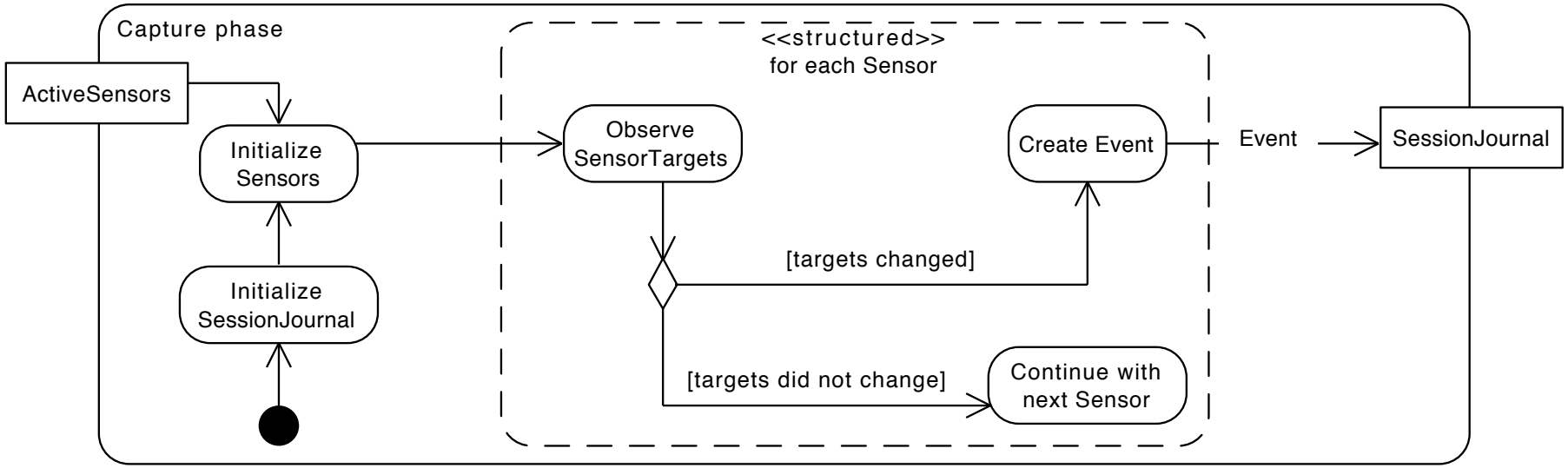
Sensors and **SensorTargets**

Capture phase

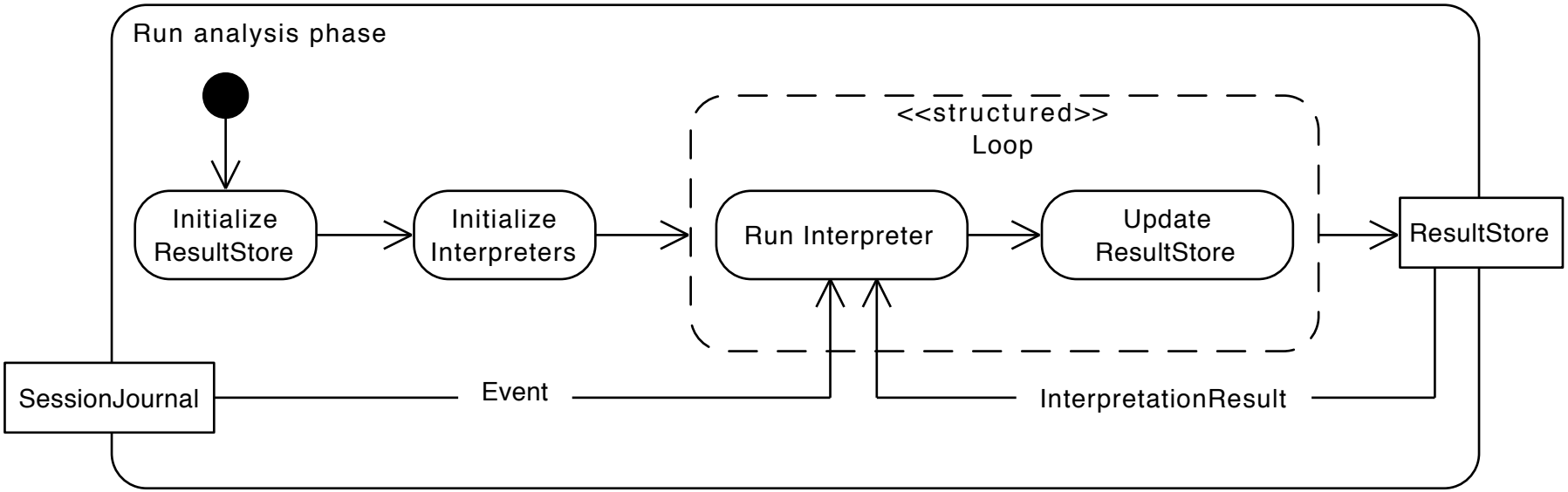


Event detection example

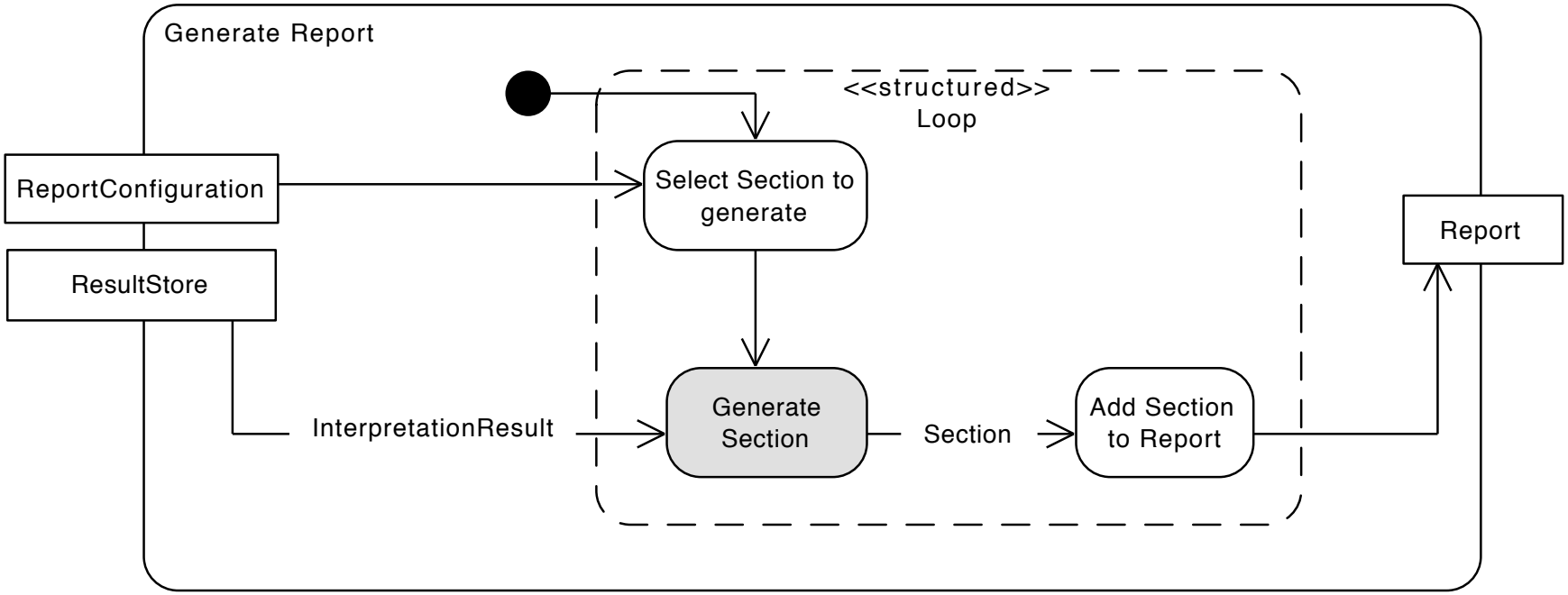
Capture phase



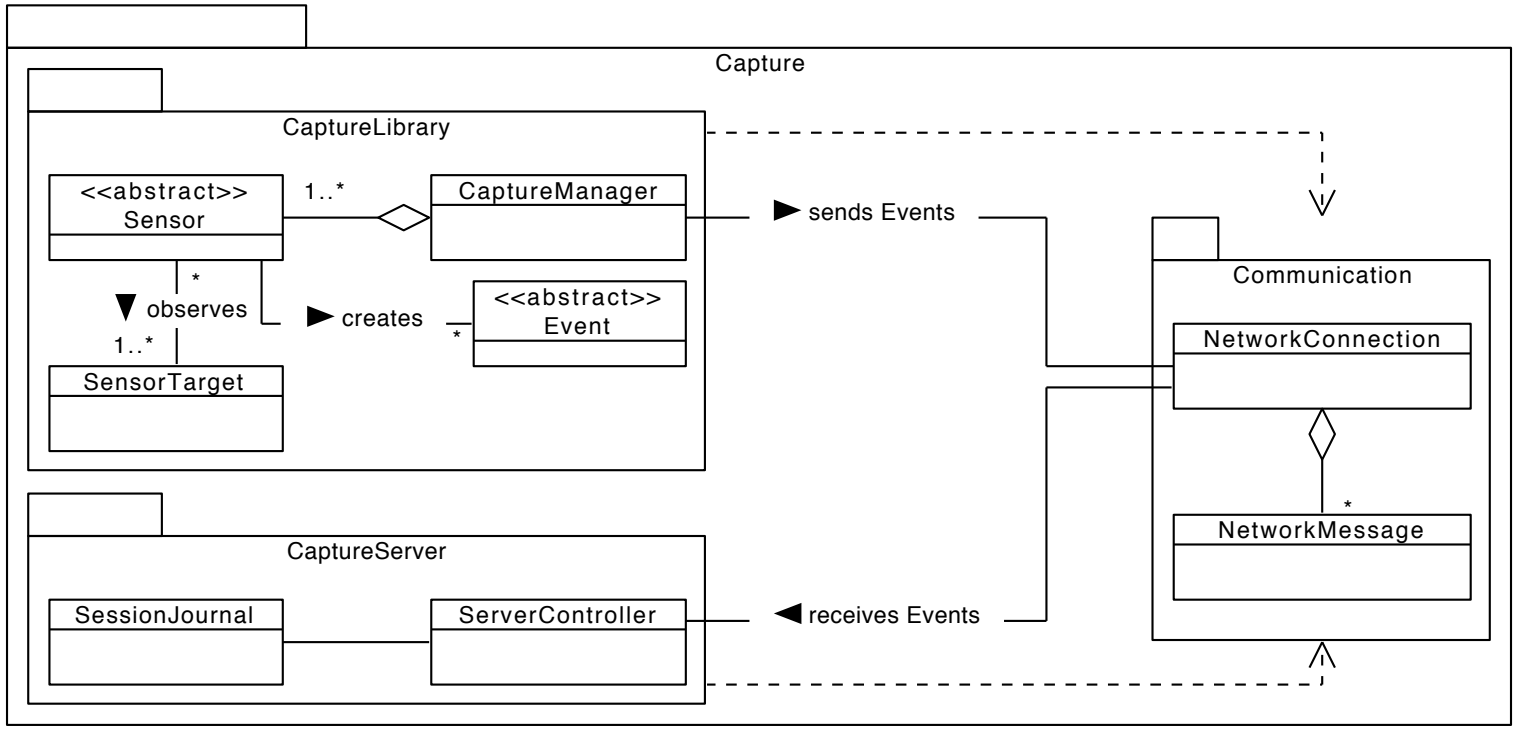
Analysis phase



Critique phase

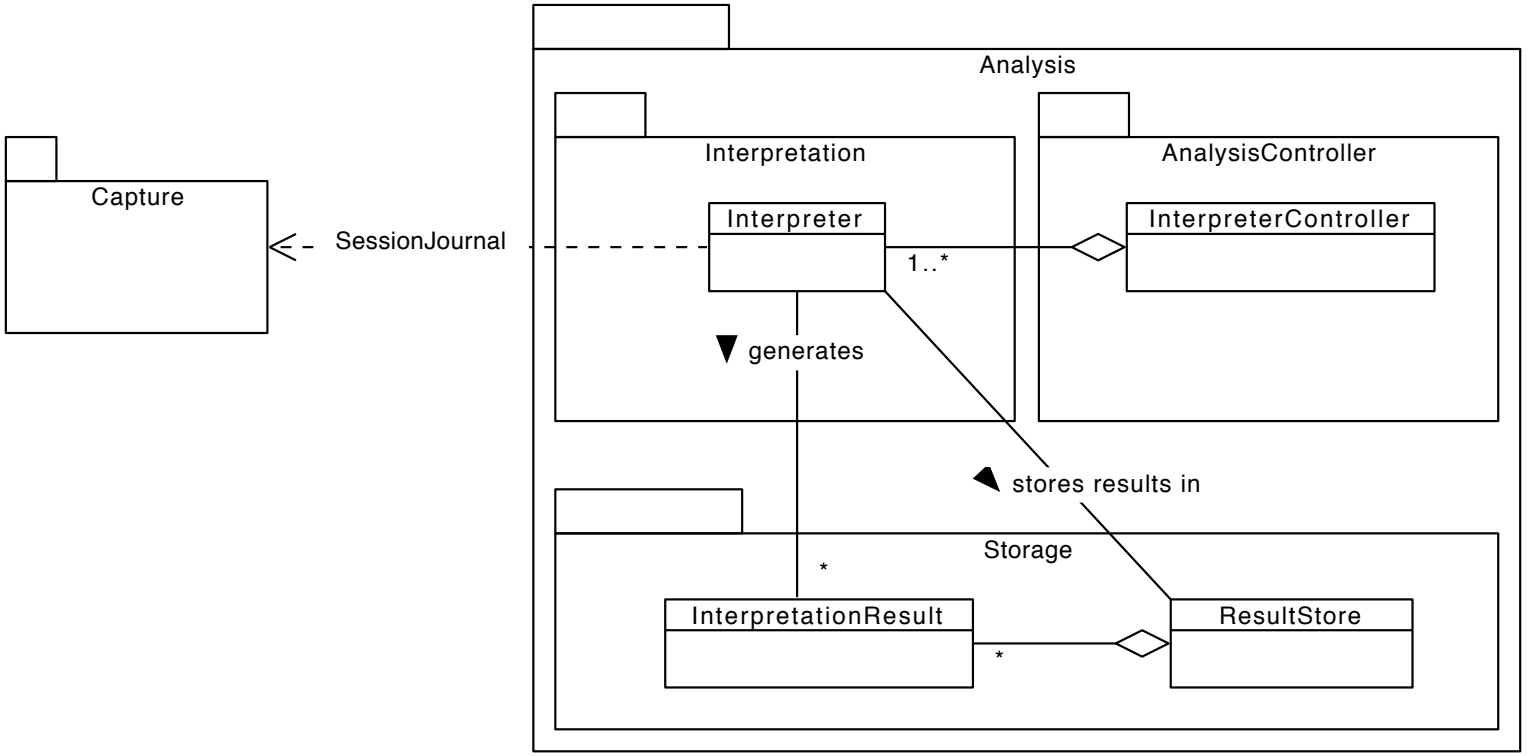


System design



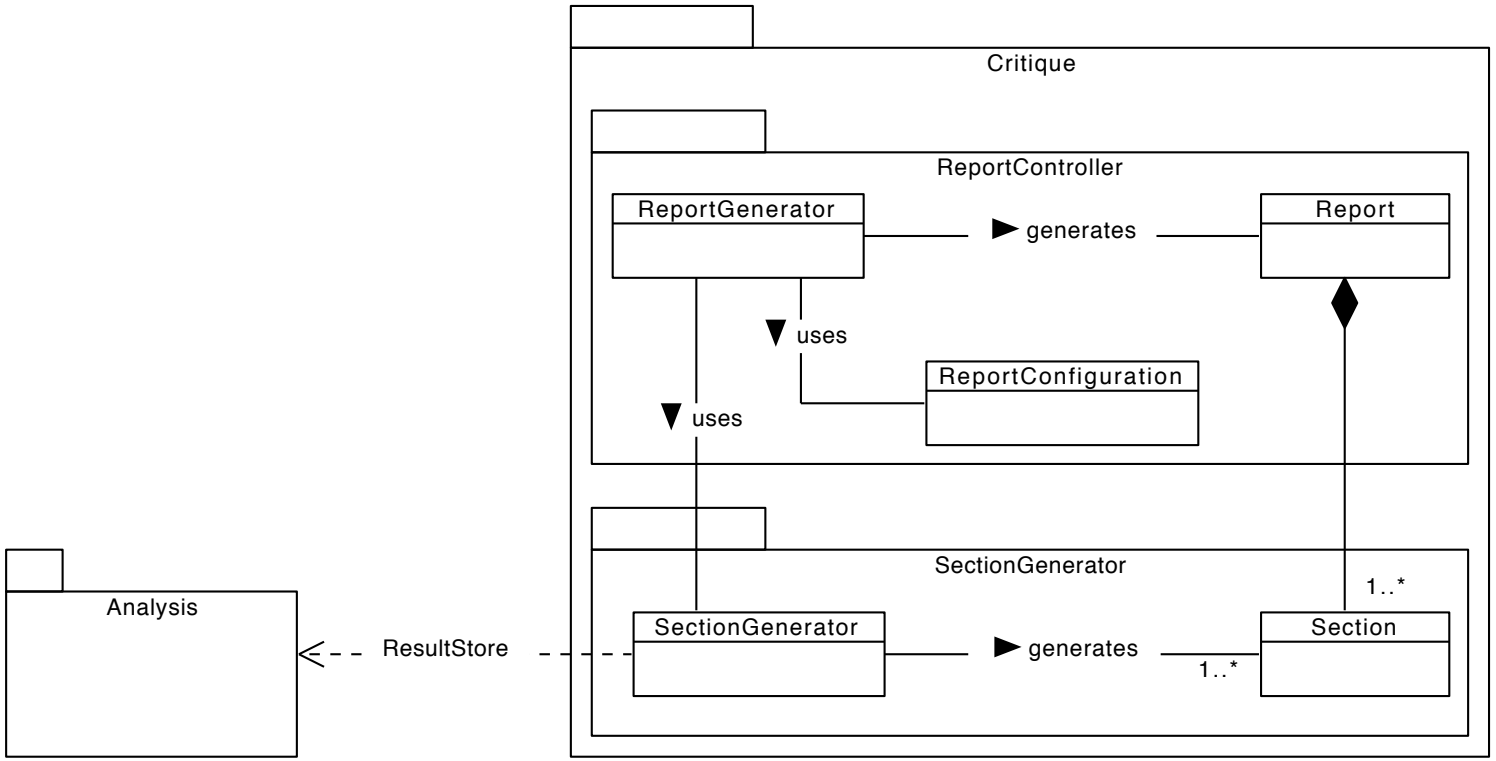
The **Capture** package

System design



The **Analysis** package

System design



The **Critique** package