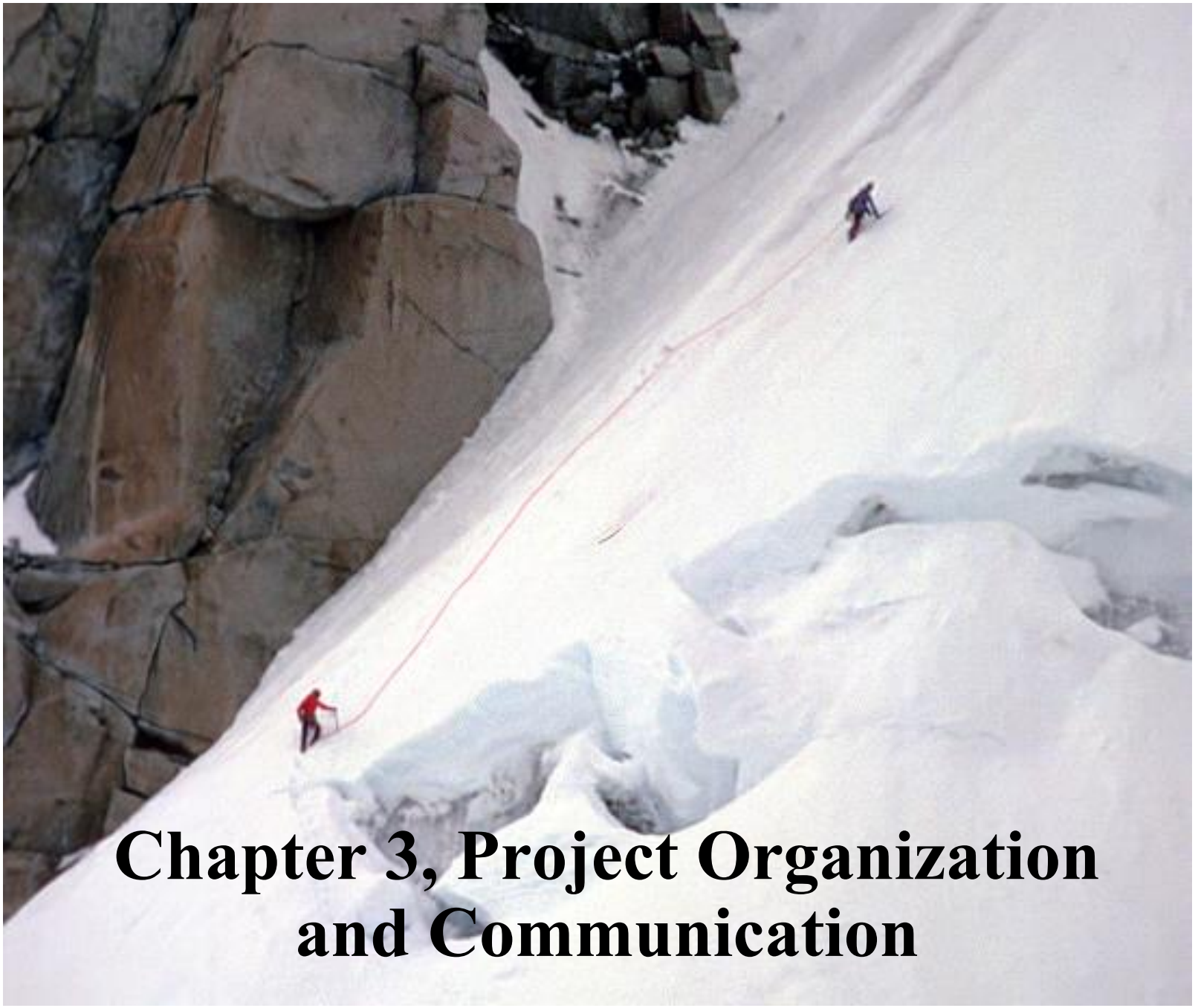
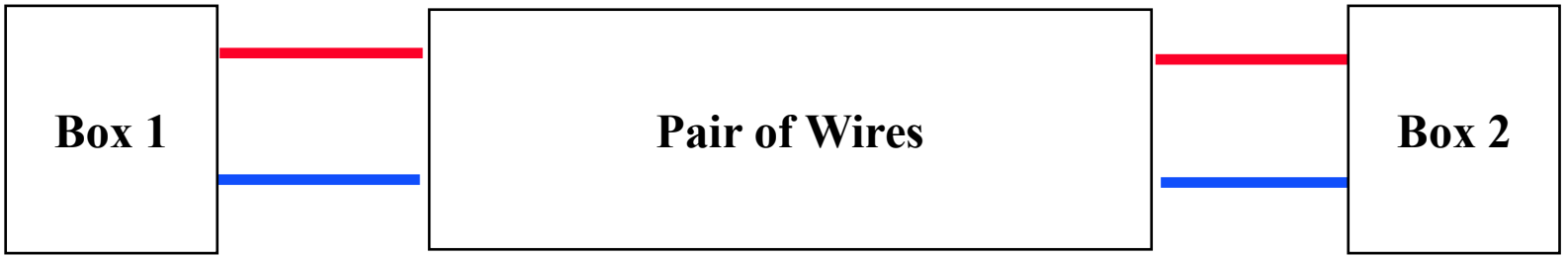
Object-Oriented Software Engineering
Using UML, Patterns, and Java



Chapter 3, Project Organization and Communication

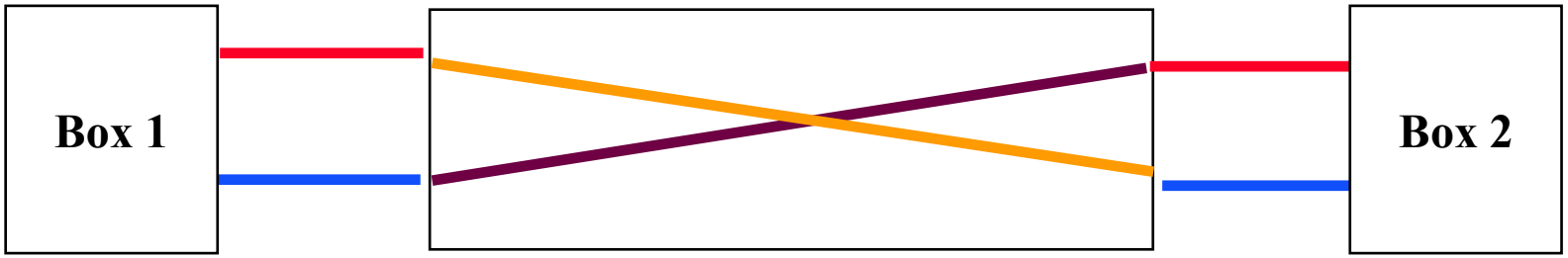
A Communication Example

"Two missile electrical boxes manufactured by different contractors were joined together by a pair of wires.



A Communication Example (continued)

Thanks to a particular thorough preflight check, it was discovered that the wires had been reversed."

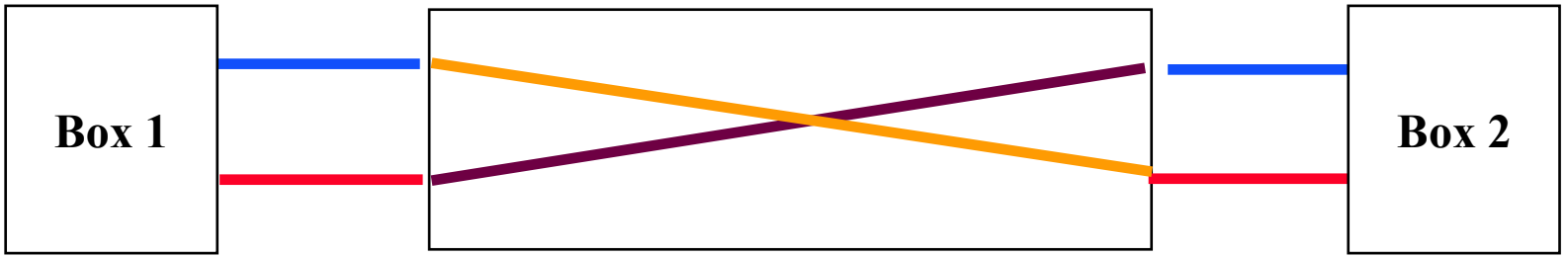


After the Crash...

...

"The postflight analysis revealed that the contractors had indeed corrected the reversed wires as instructed."

- “In fact, both of them had.”



Communication is important

In large system development efforts, you will spend more time communicating than coding

A software engineer needs to learn the so-called soft skills: technical writing, reading documentation, communication, collaboration, management, presentations.

It'd be nice for each of you to (acquire and) demonstrate the following skills:

- Management: Run a team meeting
- Presentation: Present a major aspect of your project during its development phase.
- Collaboration: Negotiate requirements with the client and with members from your team and other teams.
- Technical writing: Write part of the documentation of your software

Definitions

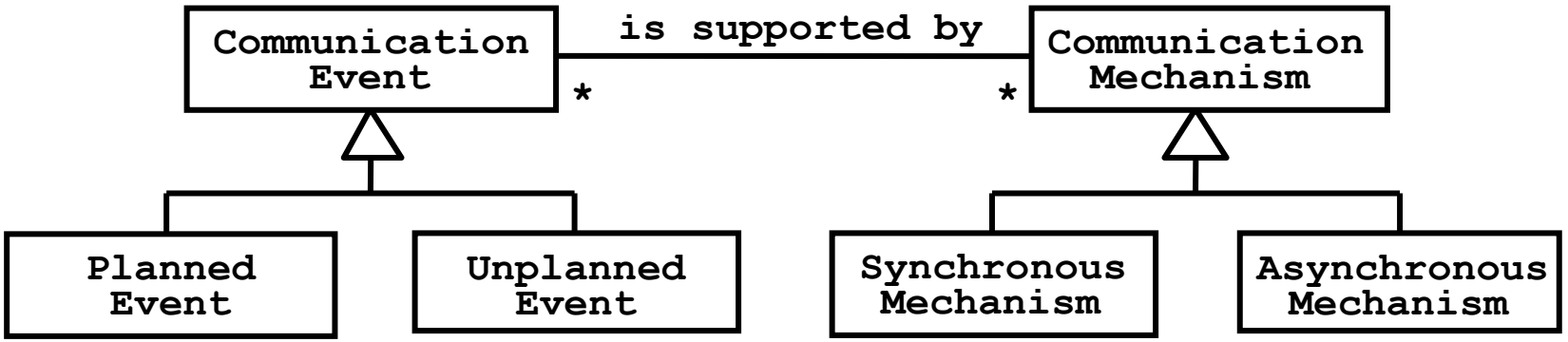
Communication event

- Type of information exchange that has defined objectives and scope
- Scheduled: Planned communication (e.g., review, meeting)
- Unscheduled:Event-driven communication (e.g., request for change, issue clarification, problem report)

Communication mechanism

- Tool or procedure that can be used to transmit information
- Synchronous: Sender and receiver are available at the same time
- Asynchronous: Sender and Receiver are not communicating at the same time.

Classification of Communication



Planned Communication Events

Problem Definition

- **Objective: Present goals, requirements and constraints**
- **Example: Client Presentation**
- **Usually scheduled at the beginning of a project.**

Project Review: Focus on system model

- **Objective: Assess status and review system model, system decomposition, and subsystem interfaces**
- **Examples: Analysis Review, System Design Review**
- **Scheduled around project milestones and deliverables**

Client Review: Focus on requirements

- **Objective: Brief client, agree on requirements changes**
- **Client Review**
- **Usually scheduled after analysis phase**

Client

- In the next project meeting of your group, assign one member as the “client” and practice the client meeting

Planned Communication Events (continued)

Walkthrough (Informal)

- **Objective: Increase quality of subsystem**
- **Example: Developer presents subsystem to team members, informal, peer-to-peer**
- **To be scheduled by each team member**

Inspection (Formal)

- **Objective: Compliance with requirements**
- **Example: Client acceptance test (Demonstration of final system to customer)**
- **To be scheduled by project management**

Planned Communication Events (continued)

Status Review

- Objective: Find deviations from schedule and correct them or identify new issues
- Example: Status section in regular weekly team meeting
- Scheduled every week

Brainstorming

- Objective: Generate and evaluate large number of solutions for a problem
- Example: Discussion section in regular weekly team meeting
- Scheduled every week

Planned Communication Events (continued)

Release

- Objective: Baseline the result of each software development activity
- Software Project Management Plan (SPMP)
- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Object Design Document (ODD)
- Test Manual (TM)
- User Manual (UM)
- Usually scheduled after each phase

Postmortem Review

- Objective: Describe Lessons Learned
- Scheduled at the end of the project

Unplanned Communication Events

Request for clarification

- The bulk of communication among developers, clients and users.
- Example: A developer may request a clarification about an ambiguous sentence in the problem statement.

Request for change

- A participant reports a problem and proposes a solution
- Change requests are often formalized when the project size is substantial.
- Example: A participant reports of a problem the air conditioner in the lecture room and suggests a change.

Issue resolution

- Selects a single solution to a problem for which several solutions have been proposed.
- Uses issue base to collect problems and proposals

Project components (developer's perspective)

- Work product
  - **Piece of code, a Use Case model, a design document, deliverables (to the client) ...**
- Schedule
  - **Intermediate (internal) deadlines**
  - **Alpha, beta, public release dates**
  - **Project management software**
- Participant / project member
  - **Developer, tester, technical writer, product manager ...**
- Task
  - **Design a component, test a component, fix a bug, write User's Guide ...**
  - **Issue tracking software**

Project organizations

- Team based
- Interaction via reporting, decision, and communication
- Example organization
  - **Management team**
  - **User Interface team**
  - **Database team**
  - **Control team**

Roles

- Each member may assume multiple roles
- Role types
  - **Management roles**
    - **Project manager, team leader ...**
  - **Development roles**
    - **System architect, object designer, implementor, tester ...**
  - **Cross-functional roles**
    - **API engineer, document editor, configuration manager, tester ...**
  - **Consultant roles**
    - **Client, end-user, application domain specialist, solution domain specialist ...**

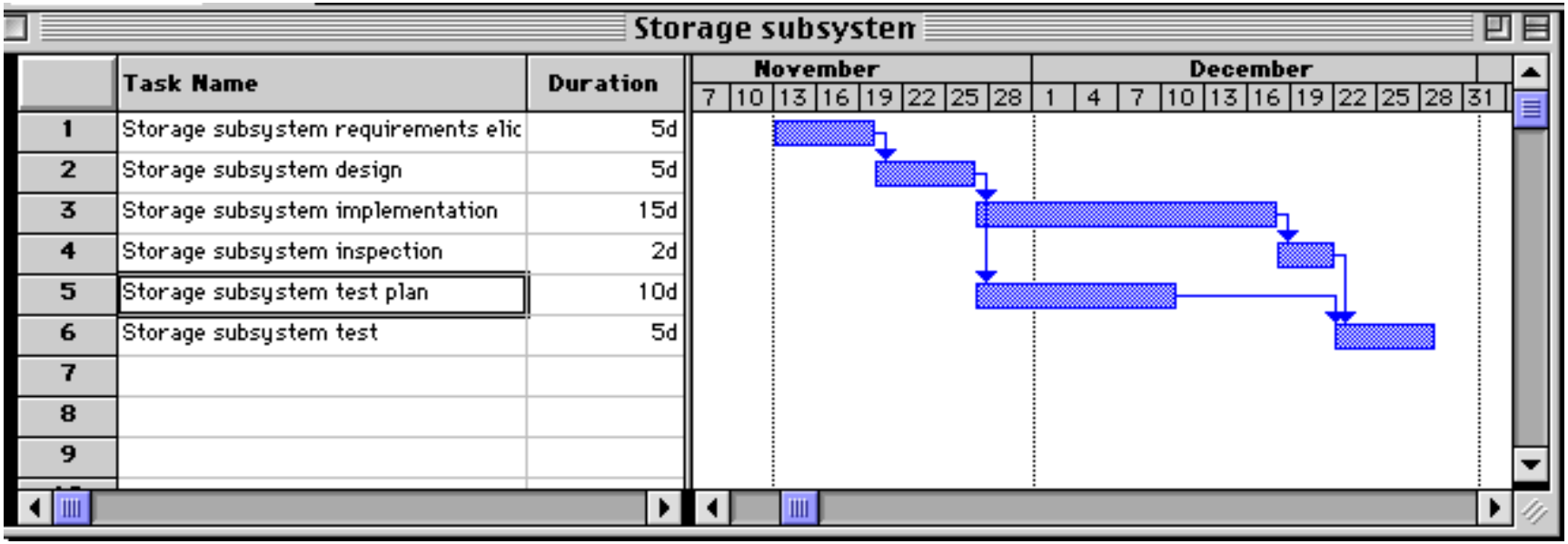
Tasks and work products

- Task: well-defined work assignment for a role
  - **Design a component, test a component, fix a bug, write User's Guide ...**
- Work product: tangible item resulting from a task
  - **Piece of code, a UC model, a design document, deliverables (to the client) ...**

Schedule

- Mapping of tasks onto time with dependencies specified

Schedule – Gannt chart



Schedule – PERT chart

Example of a Problem Statement: Introduction into ARENA

Case Study

A multi-user, web-based system for organizing and conducting game tournaments

ARENA: The Current Situation

- The Internet has enabled virtual communities
- Multi-player computer games now include support for virtual communities
  - **Players can receive news about game upgrades, new game levels, announcement of matches and scores**
- Currently each game company develops such community support in each individual game
  - **Each company uses a different infrastructure, different concepts, and provides different levels of support**
- This redundancy leads to problems:
  - **High learning curve for players joining a community**
  - **Game companies develop the support from scratch**
  - **Advertisers contact each community separately.**

ARENA: The Objectives

- Provide a generic infrastructure to
  - **Support virtual game communities.**
  - **Register new games**
  - **Register new players**
  - **Organize tournaments**
  - **Keeping track of the players scores.**
- Provide a framework for tournament organizers
  - **to customize the number and sequence of matchers and the accumulation of expert rating points.**
- Provide a framework for game developers
  - **for developing new games, or for adapting existing games into the ARENA framework.**
- Provide an infrastructure for advertisers.

ARENA: The Objectives (2)

- Provide a framework for tournament organizers
  - **to customize the number and sequence of matchers and the accumulation of expert rating points**
- Provide a framework for game developers
  - **for developing new games, or for adapting existing games into the ARENA framework**
- Provide an infrastructure for advertisers.

ARENA Functional Requirements

- Spectators must be able to watch matches in progress without prior registration and without prior knowledge of the match
- The operator must be able to add new games.

ARENA Nonfunctional Requirements

- The system must support
  - **10 parallel tournaments,**
  - **Each involving up to 64 players**
  - **and several hundreds of spectators.**
  - **The ARENA server must be available 24 hours a day**
- The operator must be able to add new games

  without modifications to the existing system
- ARENA must be able to dynamically interface to existing games provided by other game developers.

Constraints

- Constraint: Any client restriction on the solution domain and project management
  - **Sometimes also called Pseudo Requirements**
  - **Constraints restrict the solution space**
- Example of constraints
  - **Delivery constraints ("must be delivered before Christmas")**
  - **Organizational constraints ("must have a separate testing team")**
  - **Implementation constraints ("must be written in C++")**
  - **Target platform constraints ("must run on Windows 7")**

ARENA Target Environment

Example:

- Users must be able to run ARENA games as applets in any Web Browser
- The web page must be validated through the *W3C Markup Validation Service*
- Interaction with the ARENA Server must be via HTTP/1.1.

To be distinguished from development environment

- "Prototypes will be built with Revolution 2.6.1"
- "Games will be tested with Internet Explorer and Firefox"
- "The implementation language will be Java 1.4.2."
- "The IDE will be Eclipse 3.2"

Project Schedule

- The project schedule is an optional part of the problem statement
  - **Managerial information**
  - **Often the seed for the schedule in the software project management plan.**
- Lists only major milestones negotiated with the client
  - **3 to 4 dates (fixed dates!)**
- Example:
  - **Project-kickoff April 15**
  - **System review May 15**
  - **Review of first prototype Jun 10**
  - **Client acceptance test July 30**

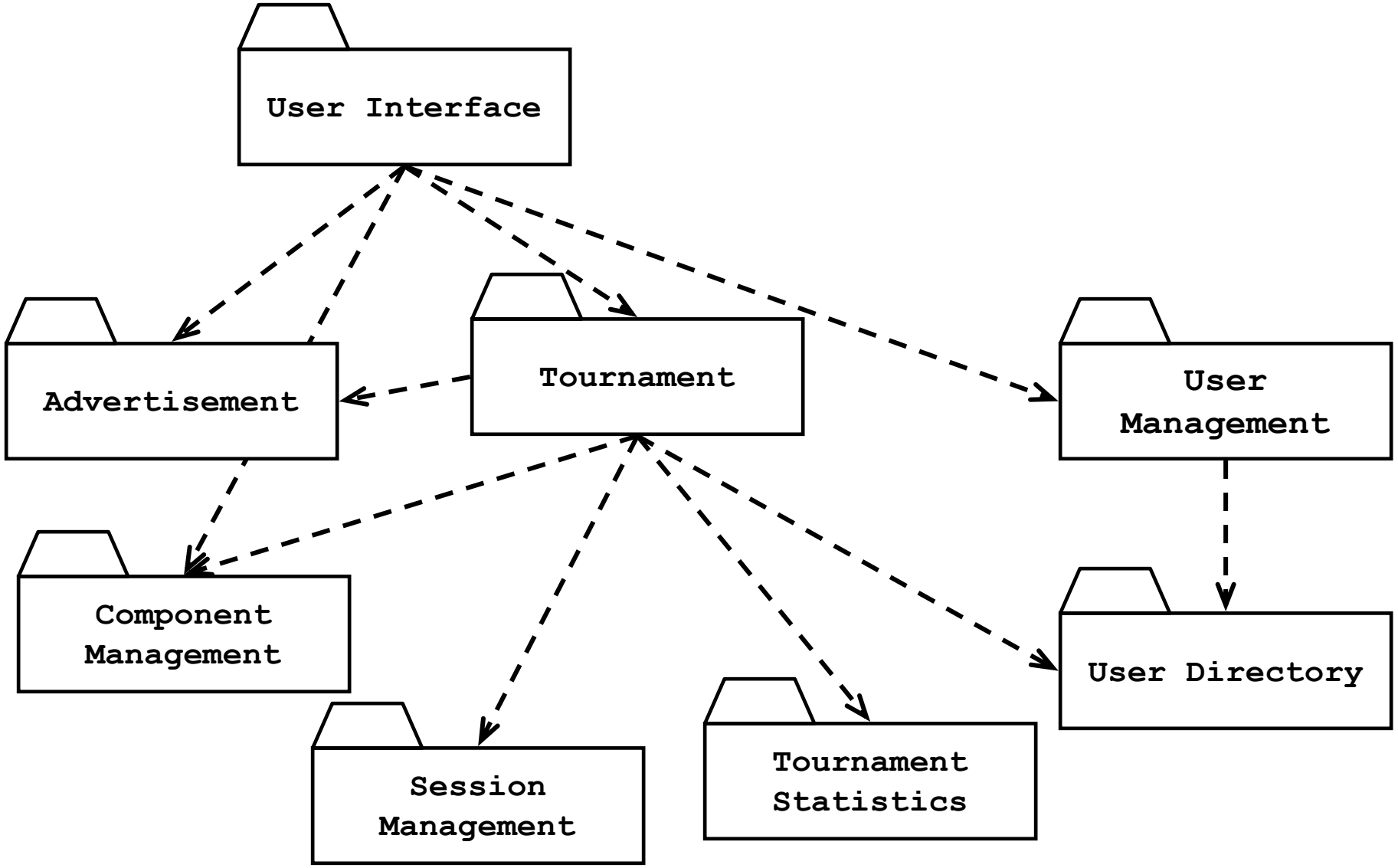
Client Acceptance Criteria

- The system supports 10 parallel tournaments with 64 players and 10 spectators per tournament
- The client supports the games Tic-Tac-Toe and Asteroids
- The average response time for a command issued by a client is less than 1 second
- The average up-time of the ARENA server during one week of testing is 95%.

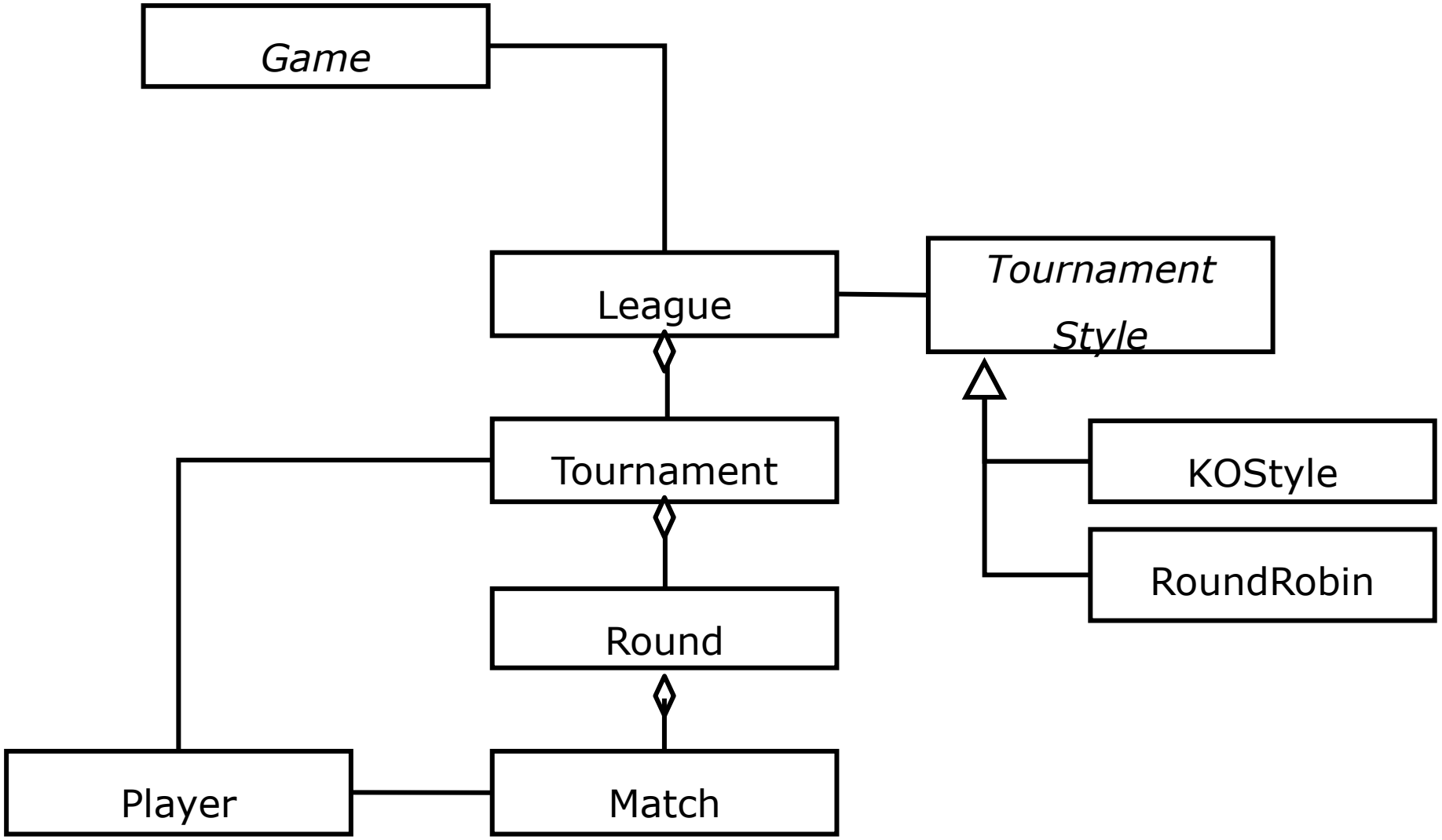
(Initial) ARENA Models

- Subsystem Decomposition
- User Interface of Client
- User Interface of Server
- Object Model

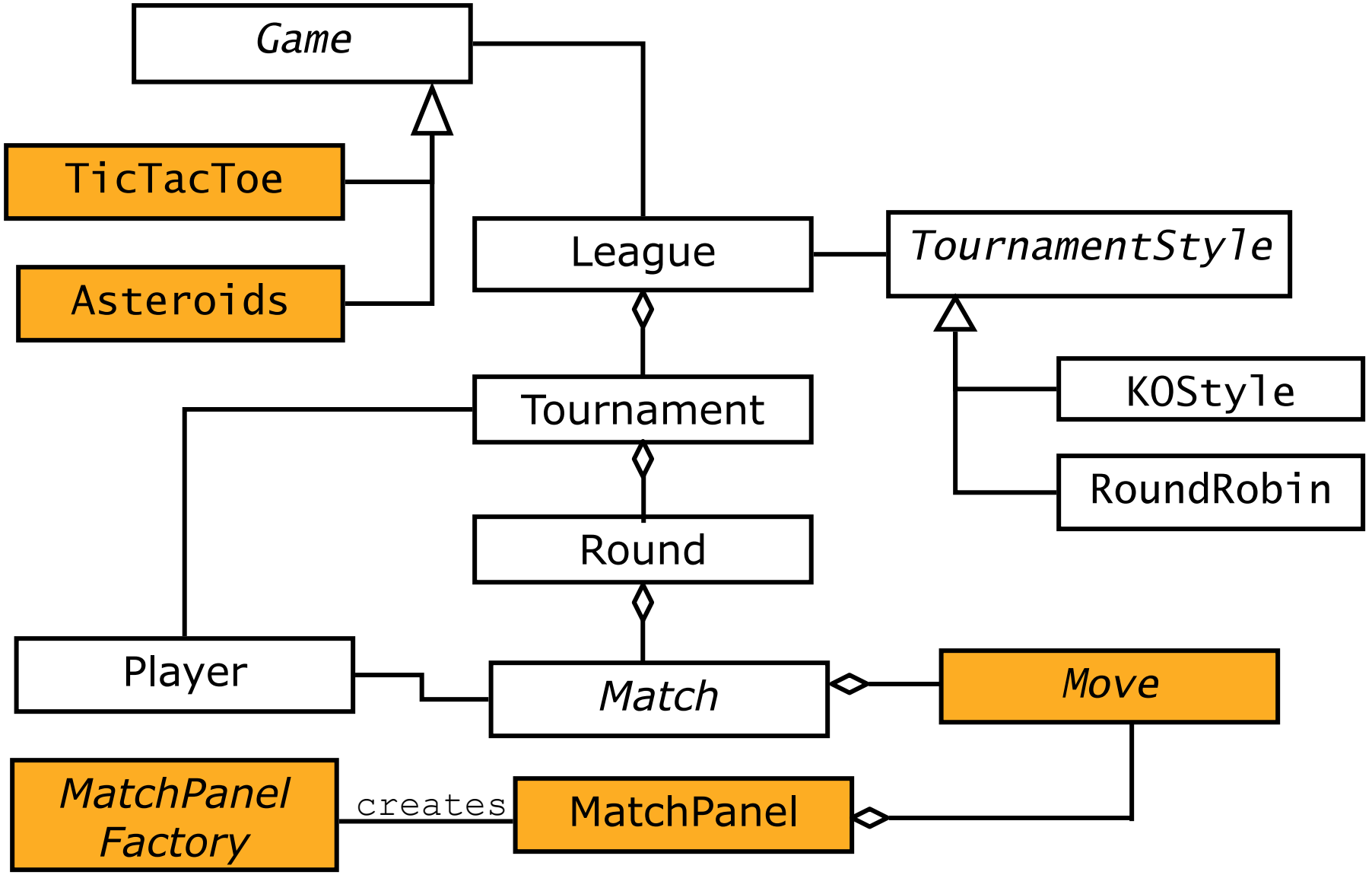
ARENA Subsystem Decomposition



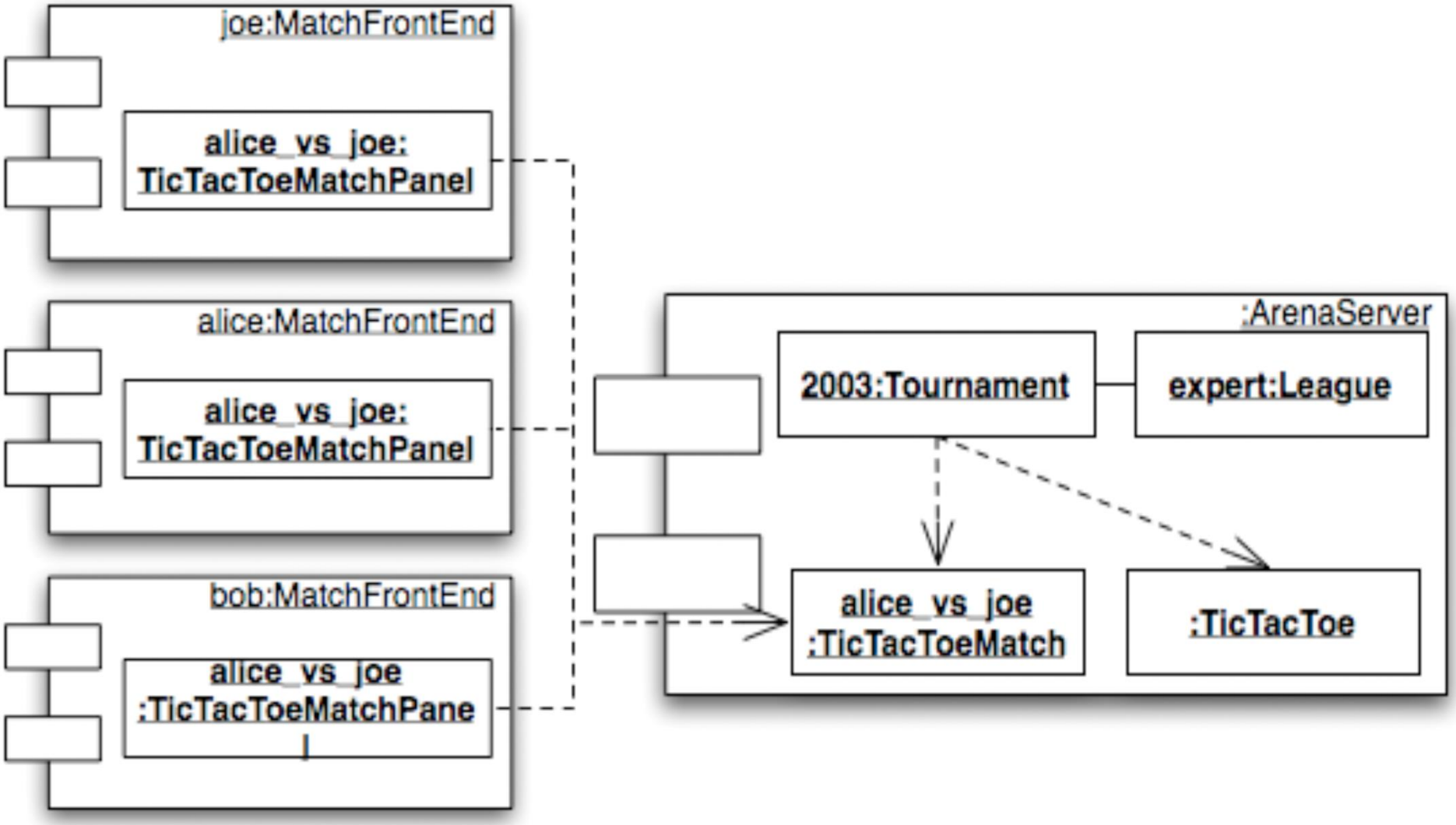
ARENA Object Model



ARENA Object Model (2)



ARENA Instance-Diagram



ARENA User Interface (Client)

ARENA User Interface (Server)