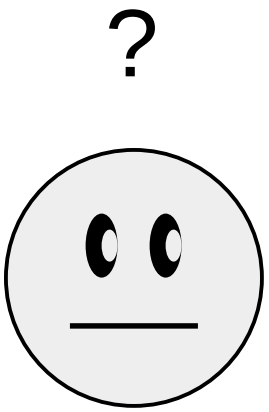
Program, what do you do?
Explaining software

Mark A. Foltz
MIT Artificial Intelligence Lab
March 1, 2001

 mfoltz@ai.mit.edu

Program understanding is hard



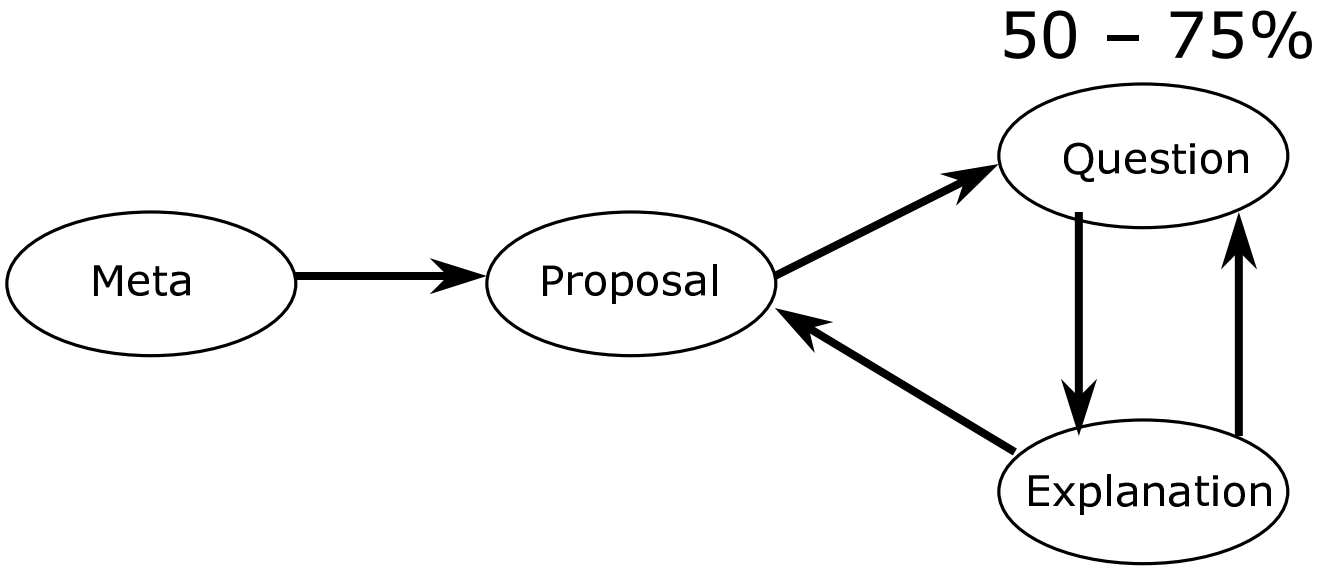
Agent.java

```
public class Agent {
    start()
    doIt()
    stop()
}
```

- **Hard to navigate**
- **Hard to see the “big picture”**
- **Behavior is implicit**
- **Purpose may be lost**

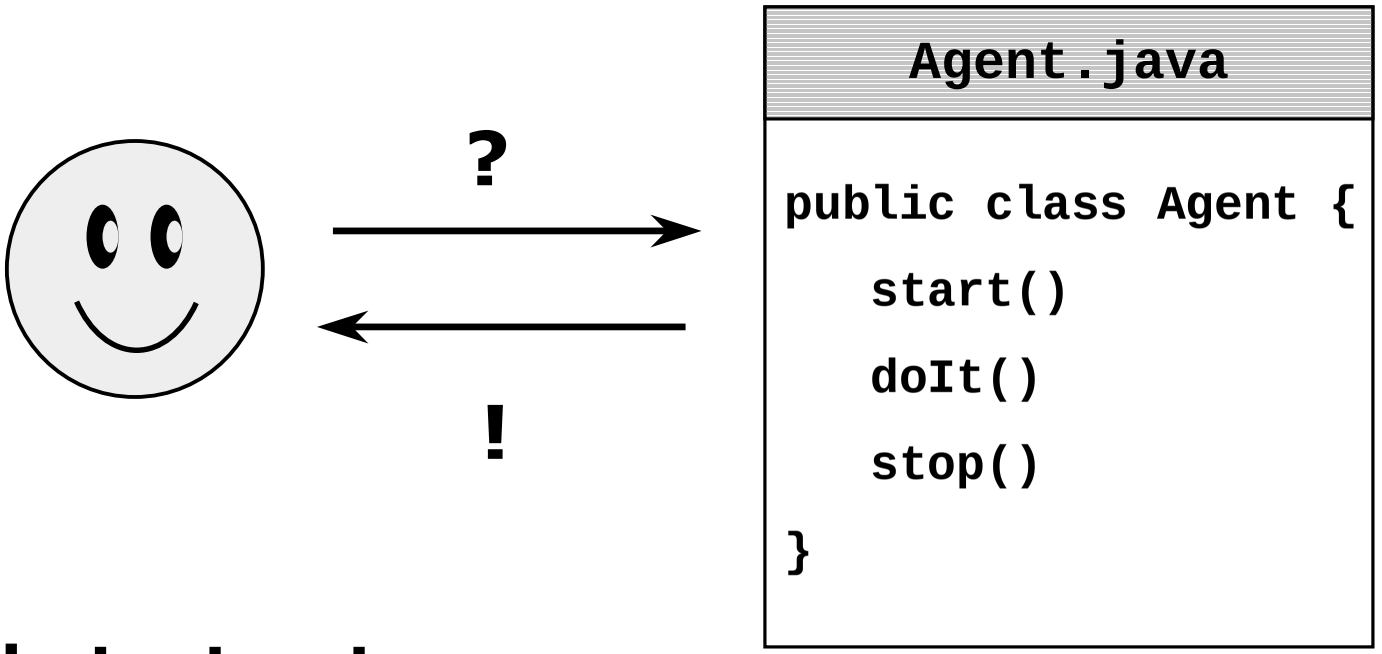
Program explanation is critical

- **Legacy code is everywhere (Y2K)**
- **High turnover rates in software teams**
- **Software teams spend a lot of time explaining code to each other**



The Dream

A "natural" explanation of a program



- **What: structure**
- **How: behavior**
- **Why: purpose, context**
- **From a designer's perspective**

The (Harsh) Reality

- **This is the inverse problem of design**
  - **Reverse engineering**
- **Nobody writes documentation**
- **Self-documenting: a myth?**
  - **Programmers can't explain their own code**
- **Software is huge**
  - **A popular OS: 35M+ lines of code**

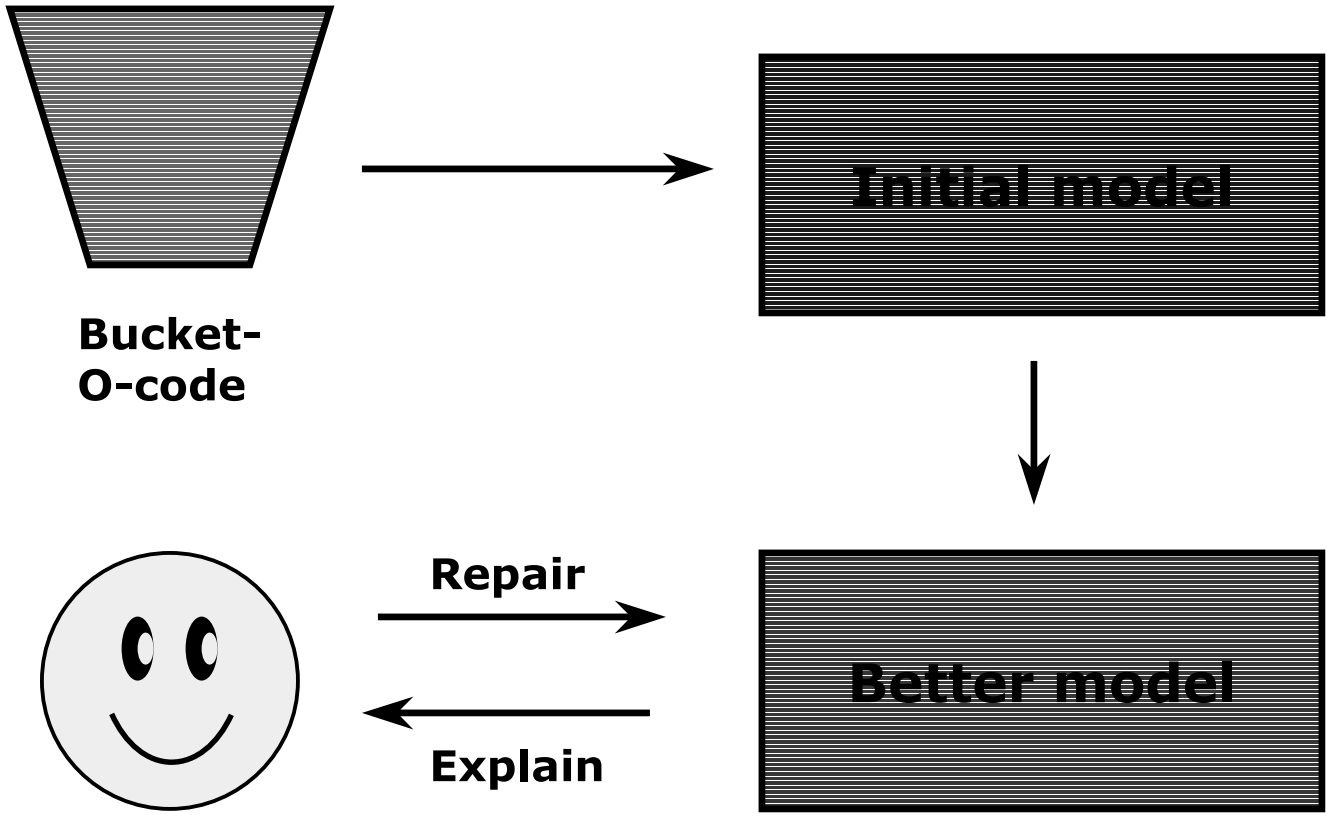
How do people explain their programs?

- **Observe them!**
  - **What they say (vocabulary)**
  - **How they gesture**
  - **What they draw**
  - **What they write (documentation)**
- **What's missing in the code**
  - **Design patterns**
- **What's not described**
  - **Tacit knowledge**

Modes of explanation

- **Investigation**
  - **Dialogue**
  - **Simulation**
- **Documentation**
  - **Mapping**
- **Teaching**
  - **Narratives**
  - **Examples**

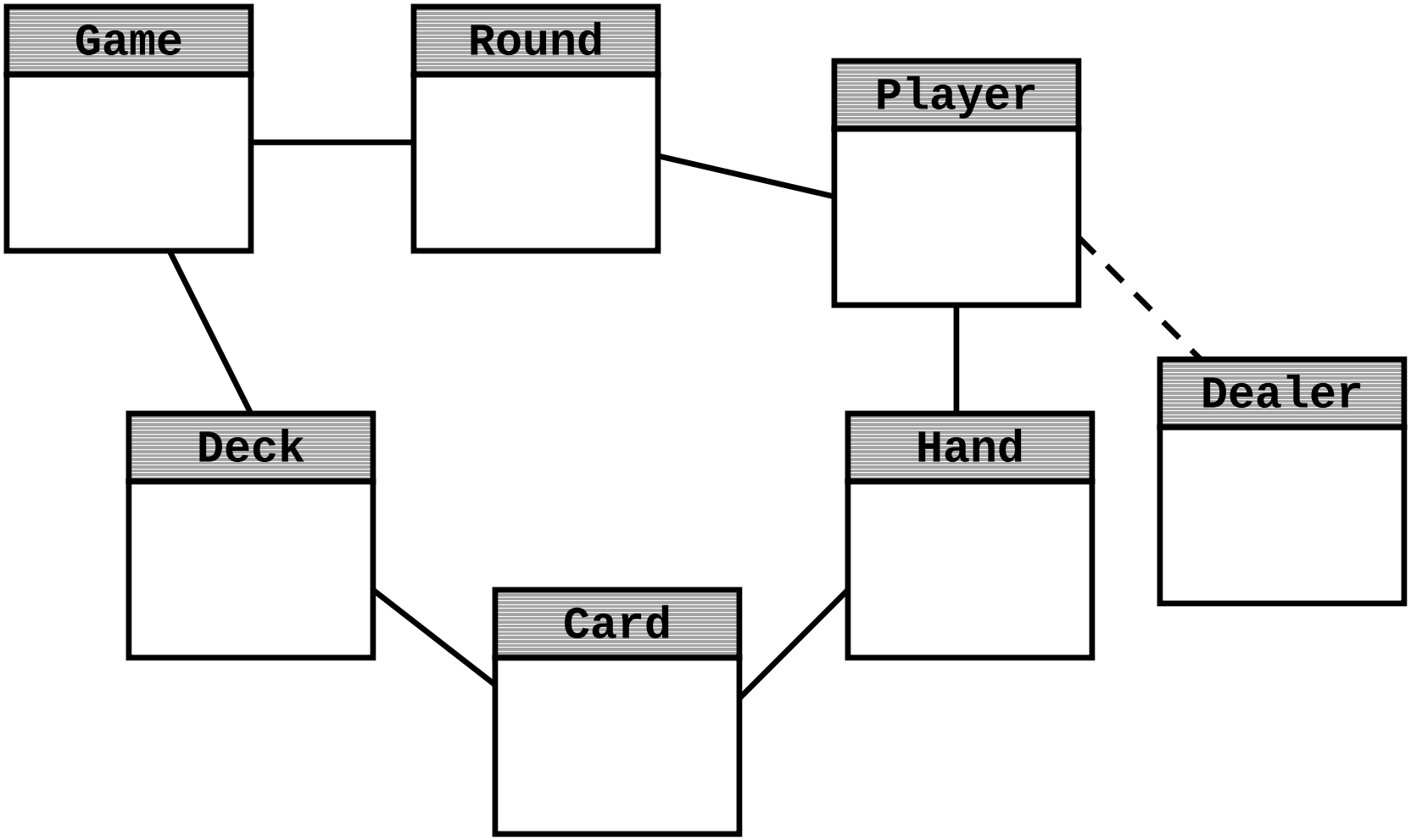
A tool to provide explanations



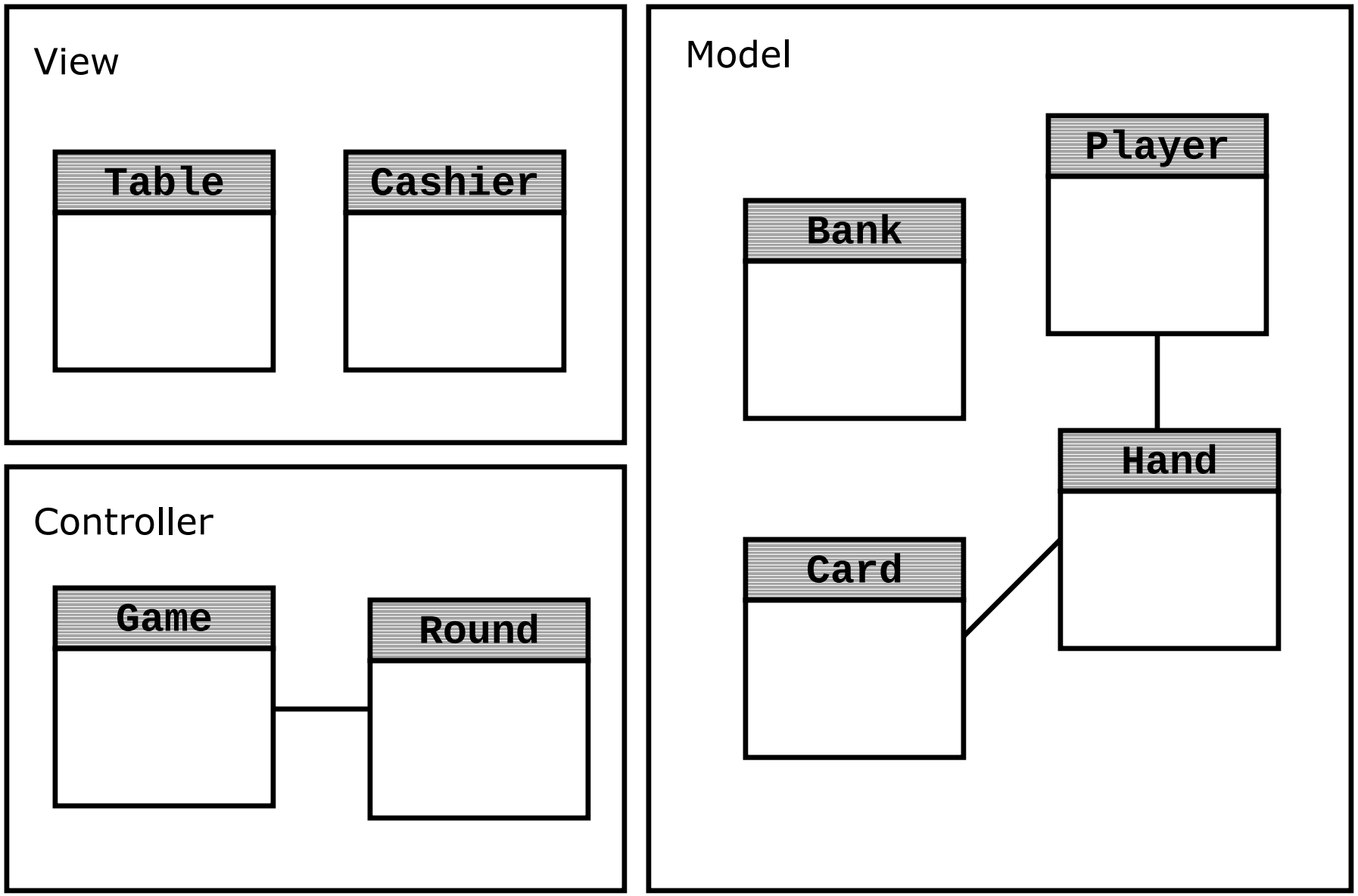
A good software explanation tool should...

- **Provide multiple levels of detail**
- **Present several aspects of the system**
- **Not require extraordinary annotations by the programmer**
- **Scale with the knowledge available about the design**

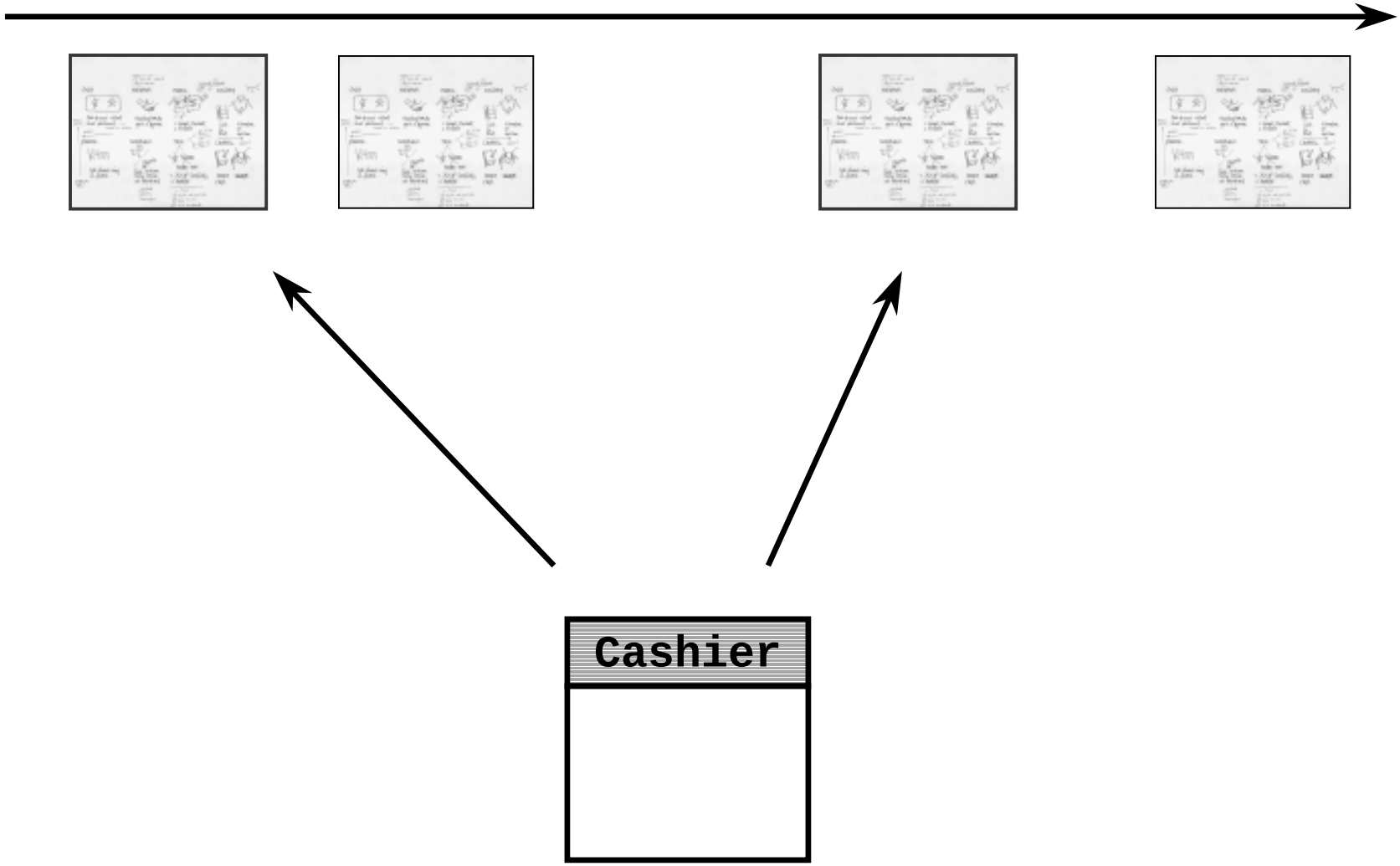
Approach: Visualization



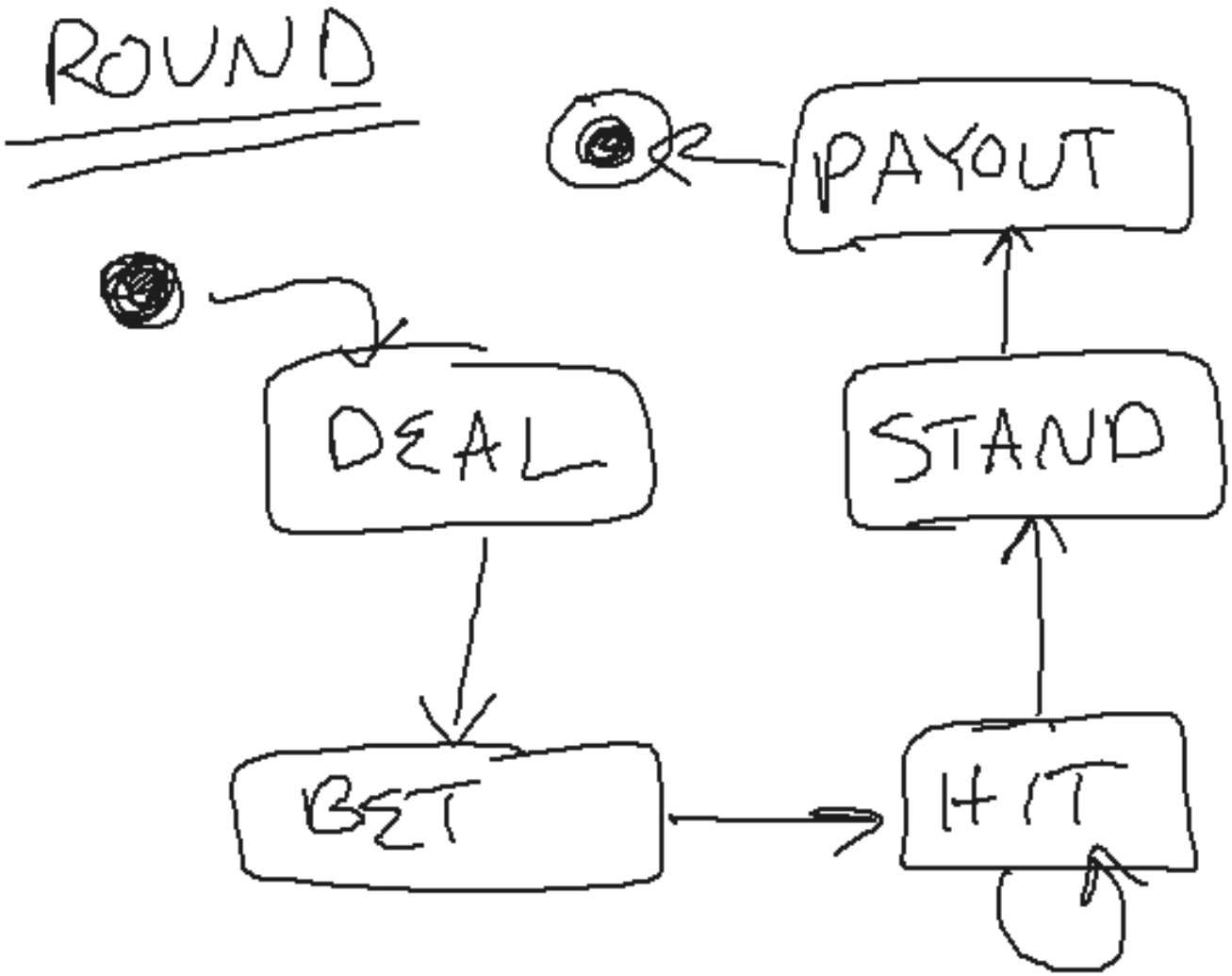
Approach: Cliché Recognition



Approach: Indexing the Design History



Approach: Ask the programmer!

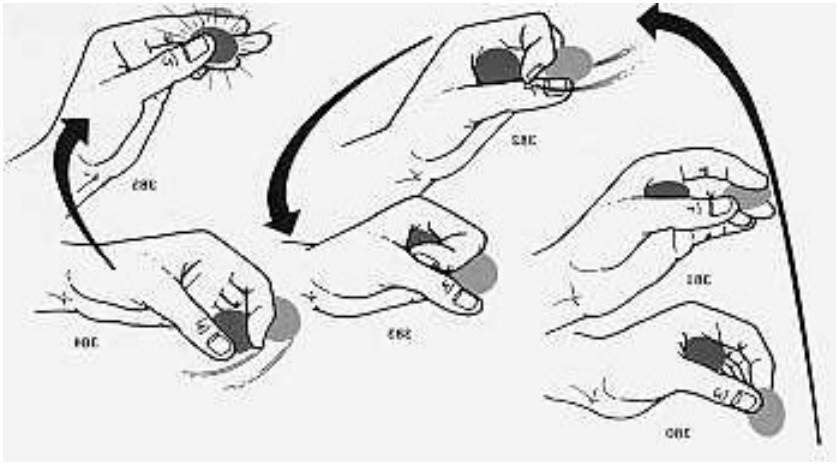


Contributions

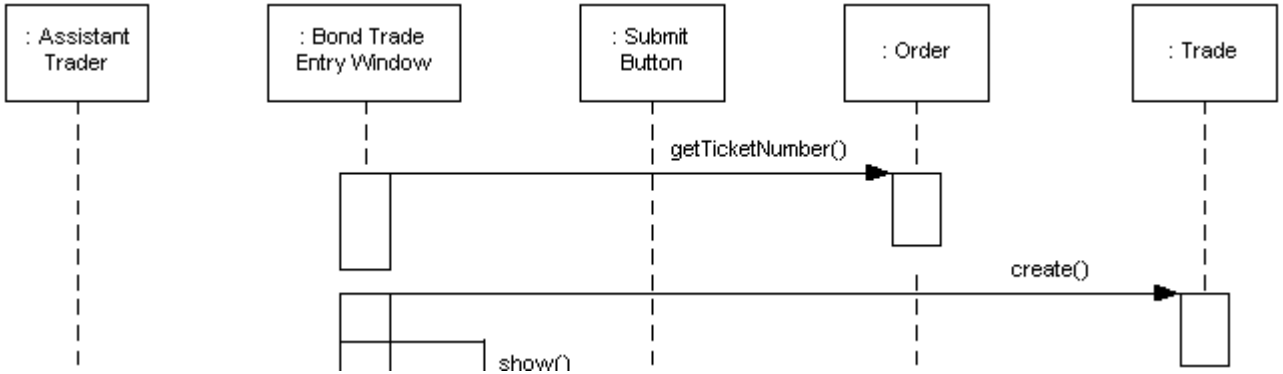
- **AI: What are methods and representations that can produce explanations of complex systems?**
- **Software engineering: How do designers think about their programs?**
- **Design rationale: What should documentation be trying to capture?**

Inspirations

- **Programmer's Apprentice (Rich, Waters, et al., 1986-1990)**
- **Tufte's Visual Explanations**



- **Software diagramming languages (UML)**



Criteria of success

- **Can it generate a natural explanation of a program of moderate complexity?**

  I.e. blackjack

- **Is the explanation helpful?**
  - **versus Javadoc: the 15 minute showdown**