Self-describing Agents

MAS^2 2008

P. Reiß, T. Seutter, B. Schiemann, G. Görz, B. Ludwig

Department of Computer Science 8
University Erlangen-Nuremberg

28. 02. 2008

Outline

Initial Situation

Nowadays, computers are ubiquitous

- Applications become more and more complex
- Users hardly know about all of the features of a program

Users have to learn to become familiar:

- Manuals, textbooks
- Online help systems
- Training courses

The MAS case

Agents are

- autonomous
  $\Longrightarrow$ not necessarily feedback to the user
- flexible
  $\Longrightarrow$ new behaviours users might not know about
- social
  $\Longrightarrow$ communication without user interaction

JPAPA

Java Platform Annotation Processing Architecture

- Main idea: software should be able to describe itself
- Synchronous explanations about what is going on „inside“ the system

How ist works:

- Application programmers put explanations in the source code
- System runs in introspective mode
- User is informed if „explained“ code is reached

Java annotations

- Provide data about a program that is not part of the program itself
- No direct effect on the operation of the code they annotate
- Used by pre-processor
- Visible at runtime

```
@Doku(id="setup()",
   doku="Some explanation about the method")
protected void setup() {
   ...
}
```

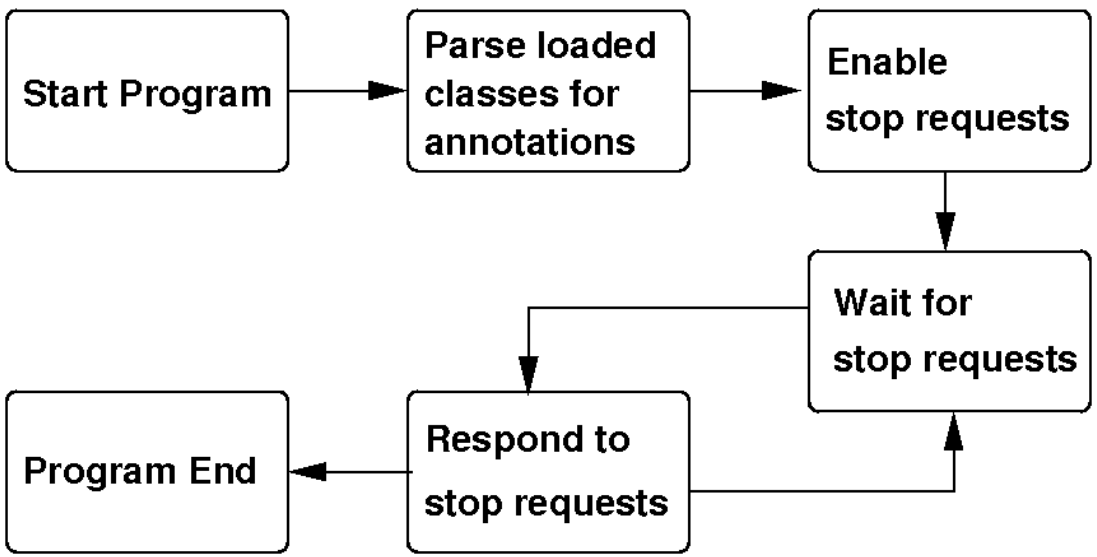
Java Reflection API

- Used by programs which require the ability to examine or modify the runtime behavior of applications
- Information about classes, methods, member variables, ...
- Examination of annotations

```
if(field.isAnnotationPresent(Doku.class)) {
  Annotation doc = field.
                   getAnnotation(Doku.class);
  String doku = doc.doku();
  ...
}
```

Java Debugging Interface

- Part of the Java Platform Debug Architecture (JPDA)
- High level Java API providing information useful for debuggers
- Every annotated element is marked with a stop request
- Execution is suspended whenever annotated code is to be executed
- Explanation is presented to the user



MAS for supply chain management

- A^4: **A**daptive **A**genten **A**nwendungen und **A**utonomie
- MAS for tracking and tracing supply chains
- 3 types of agents:
  - Discourse agent
  - Surveillance agent
  - Coordination agent
- Proven outcome of DFG SPP 1083: „Agentensysteme in betriebswirtschaftlichen Anwendungen“

Focus on coordination agent

- Management of tracking and tracing
- Starts and stops surveillance agents
- Decision support tool for management (CEOs)
- Business rules
  - Rule engine
  - Fuzzy logic
- Autonomy is adjustable:
  - Meta rules for user notification and intervention
  - But: no explanation when and why a rule matches

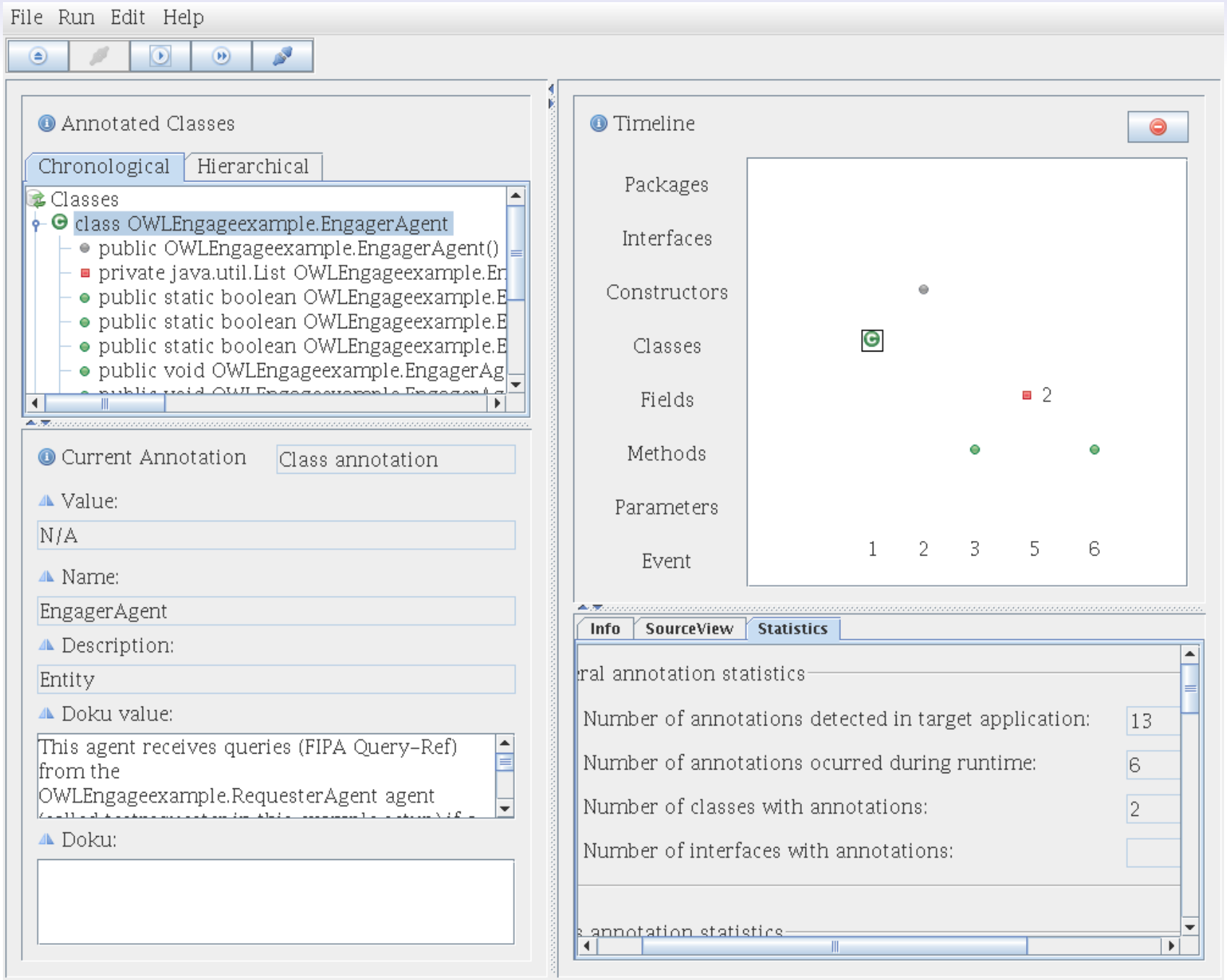
Self-explaining coordination agent

Introspection for

- Explanation of rules and meta rules
  $\Longrightarrow$ decision making becomes traceable
- Domain experts are able to learn how the coordination agent works
- Flexible supervised adjustment of the autonomy
- Dynamic well-grounded adjustment of business rules

Thank You

- For your attention!
- Any questions?


File Run Edit Help
Annotated Classes
Chronological
Hierarchical
Classes
class OWLEngageexample.EngagerAgent
public OWLEngageexample.EngagerAgent()
private java.util.List OWLEngageexample.En
public static boolean OWLEngageexample.E
public static boolean OWLEngageexample.E
public static boolean OWLEngageexample.E
public void OWLEngageexample.EngagerAg
Current Annotation
Class annotation
Value:
N/A
Name:
EngagerAgent
Description:
Entity
Doku value:
This agent receives queries (FIPA Query-Ref) from the OWLEngageexample.RequesterAgent agent
Doku:
Timeline
Packages
Interfaces
Constructors
Classes
Fields
Methods
Parameters
Event
2
1 2 3 5 6
Info
SourceView
Statistics
ral annotation statistics
Number of annotations detected in target application: 13
Number of annotations ocurred during runtime: 6
Number of classes with annotations: 2
Number of interfaces with annotations:
annotation statistics