WICSA/QoSA/CBSE 2016

CAPTURING DESIGN DECISION RATIONALE IN PROGRAM-LEVEL ASPECTS
A.K.A.
ARCHITECTURAL DECISIONS @ RUNTIME

(Paperless) Tool Demonstration

Oliver Kopp, University of Stuttgart

Olaf Zimmermann, Distinguished (Chief/Lead) IT Architect (The Open Group), HSR FHO

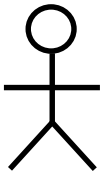
Venice, April 7, 2016

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Software Architectural Knowledge Management (SAKM)

- **G. Booch: "Architectural decisions are the design decisions that are costly to change"**
- **More elaborate definition in *Architectural Decisions as Reusable Assets* (IEEE SW 2011)**
  - Focus on NFRs/QAs, components and connectors, viewpoints
- **Many papers on AKM/AD capturing/modeling/reuse**
  - *SAKM* book, Springer 2009
  - ISO/IEC/IEEE 42010 (2011)
  - *10 years of SAKM*, Capilla et al. (JSS)
  - Tools, e.g. ADMentor (WICSA 2015)
- **Architect vs. developer bridging?**
  - Just enough Software Architecture?
  - Coding the Architecture?

"Let us use AngularJS and Play for our Web App because everybody else does, it will look good on our CVs."

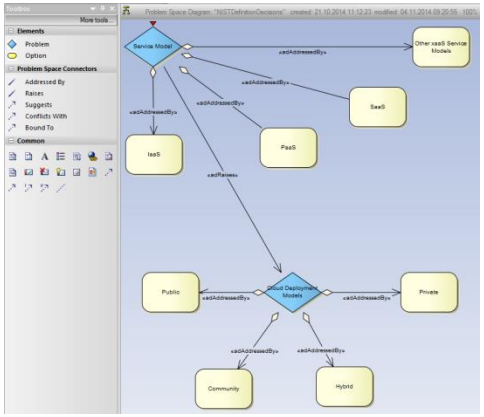


DevArch

"We chose GSON as our JSON Java parser because of its superior performance shown in PoC and the active community support for it."

ADMentor Tool (WICSA 2015): AddIn to Enterprise Architect

- **ADMentor now openly available at https://github.com/IFS-HSR/ADMentor**



Architectural Decision Guidance across Projects

Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge

Olaf Zimmermann, Lukas Wegmann
Institute for Software
Hochschule für Technik (HSR FHO)
Rapperswil, Switzerland
{firstname.lastname}@hsr.ch

Heiko Koziolek, Thomas Goldschmidt
Research Area Software
ABB Corporate Research
Ladenburg, Germany
{firstname.lastname}@de.abb.com

(WH)Y?

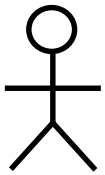
- My version (the Y-approach):
  - *In the context of <use case/user story u>, facing <concern c>, we decided for <option o> to achieve <quality q>*
  - These Y-statements yield a bullet list of open/closed (design) issues (link to project management!)
  - Can go to appendix of software architecture document, notes attached to UML model elements, spreadsheet, team space, or wiki

ABB

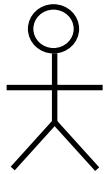
- **Project website http://www.ifs.hsr.ch/index.php?id=13201&L=4**

User Stories to Overcome Architectvs. Developer Dilemma

"As a **development lead**, I would like me and all developers on my teams to **capture** important designs **decisions** and their **rationale** decentrally but **consistently** – in a structured and machine-readable manner so that code maintainers can instantly be informed about the architectural impact of changes they plan to implement, and a consolidated, up-to-date decision log can be extracted for other project stakeholders."



Architect, Development Lead



Developer

"As an **agile developer**, I value working software over comprehensive documentation, and individuals and interactions over processes and tools. To satisfy the request for rationale capturing efficiently and effectively, I would like to **state my decisions directly in the code** – so that I can use code completion and features of my favorite IDE and I do not have to switch to another tool context when I am the flow of programming and unit testing."

Java Annotation to Support the User Stories (Variant 1)

```
public @interface DecisionMade {
    String id() default "AD-nn";
    String solvedProblem() default "[not yet captured]";
    String chosenOption();
    String rationale() default "[not yet justified]";
    String[] relatedDecisions() default {}; }
```

```
@DecisionMade(
id="AD-001",
solvedProblem="Need to select a single consistent
¨ComponentImplementationParadigm' for entire business logic layer",
chosenOption="Decided to implement business logic as a 'POJO')",
rationale="because it is a company-wide policy",
relatedDecisions={"AD-002", "AD-003"})
public class ADAnnotationDemoClass {
       public int sampleBusinessLogic(String input) {
              return 42;}}
```

Java Annotation Definition (Variant 2)

```
public @interface YStatementJustification {
  String id() default "AD-xx";
  String context() default " [functional requirement and
    current design stage]";
  String facing() default "[non-functional requirements
    such as quality attributes and constraints]";
  String chosen() default "[selected solution option]";
  String[] neglected() default {"[alternate solution
    options]"};
  String achieving() default "[positive consequences of
    chosen solution)]";
  String accepting() default "[negative consequences of
    chosen solution]";
  String moreInformation() default "";}
```

- **Usage example shown in actual demo**

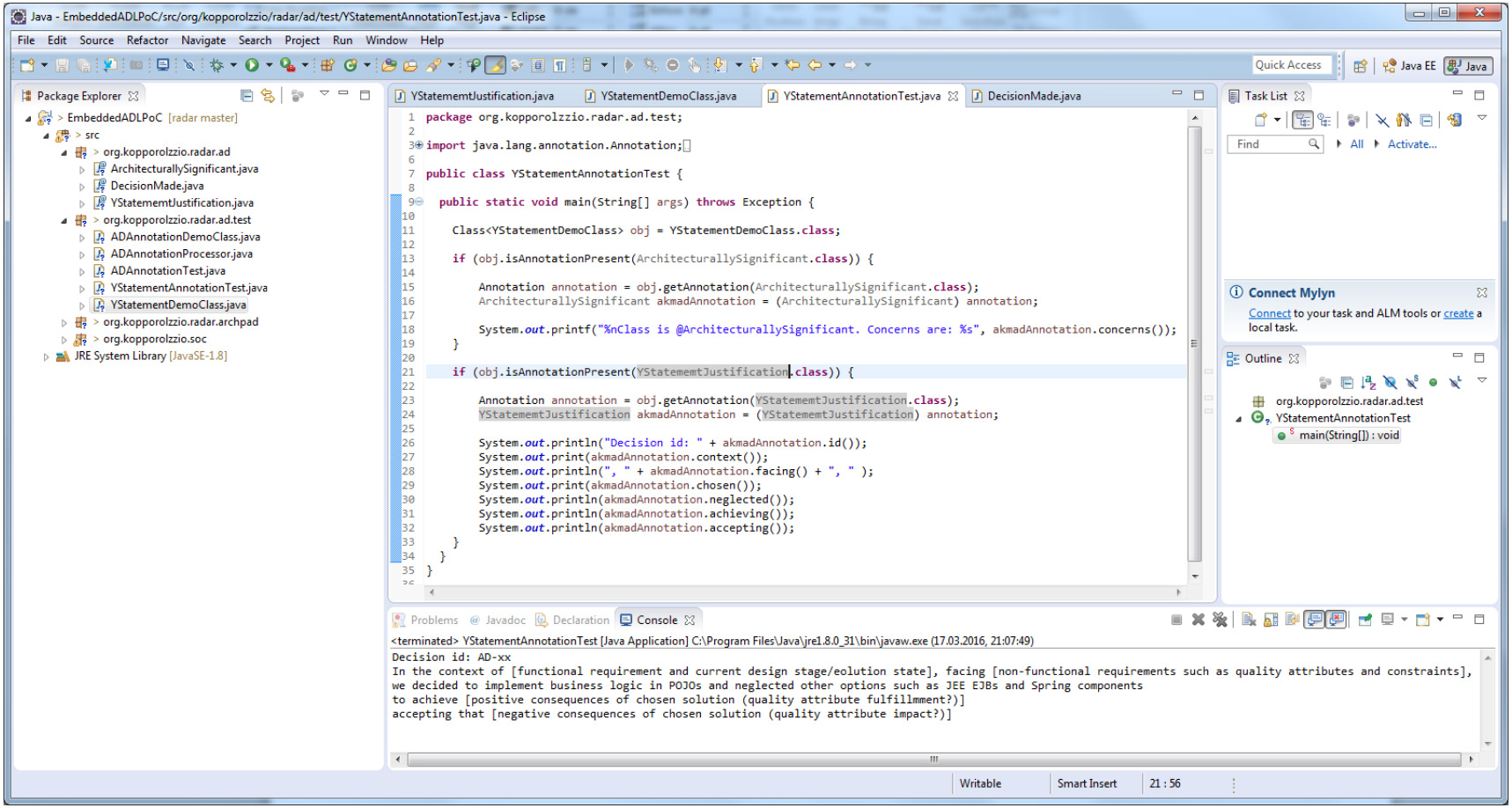
Architectural Decisions at Runtime (ADAR) – Questions (1/2)

- **Is the general approach practical, does it meet a real need?**
  - Which architectural decisions can be captured on the code level (and which ones are less suited for that)?
- **Are the annotations defined/designed well?**
  - Expressive enough to be informative, but also compact enough to be easy to produce and consume (for humans)?
- **Which of the annotation variants 1 and 2 is more appropriate in terms of balancing expressivity and readability?**
  - Flat/basic one or modelled, decomposed one?
- **Where in the code should the annotations be placed?**
  - Should they annotate interfaces and classes only, or methods and variables as well?
  - What about other language elements?

Architectural Decisions at Runtime (ADAR) – Questions (2/2)

- **Is the annotation effort compensated by the benefits of such architecturally evident coding style?**
  - Does it indeed bring architects and developers closer together?
  - What are the technical and organizational risks (e.g., can an “annotation proliferation syndrome” occur?)
- **What are the alternatives to aspects/annotations that suit developer needs?**
  - Are predefined comments such as custom task tags in Eclipse (e.g., // CAPTURE [AD] or // DOCUMENT_DECISION sufficient to solve the problem?
- **Which other annotations should be defined en route to an architecturally evident coding style?**
  - [TODO tbc]
- **ADAR demo code available on GitHub:**
  - https://github.com/koppor/embedded-adl

Demo Setup (in Eclipse): Annotation Usage & Runtime Access



https://github.com/koppor/embedded-adl