Refactoring Architecture Models

ACM/IEEE MODELS 2018, København, Denmark

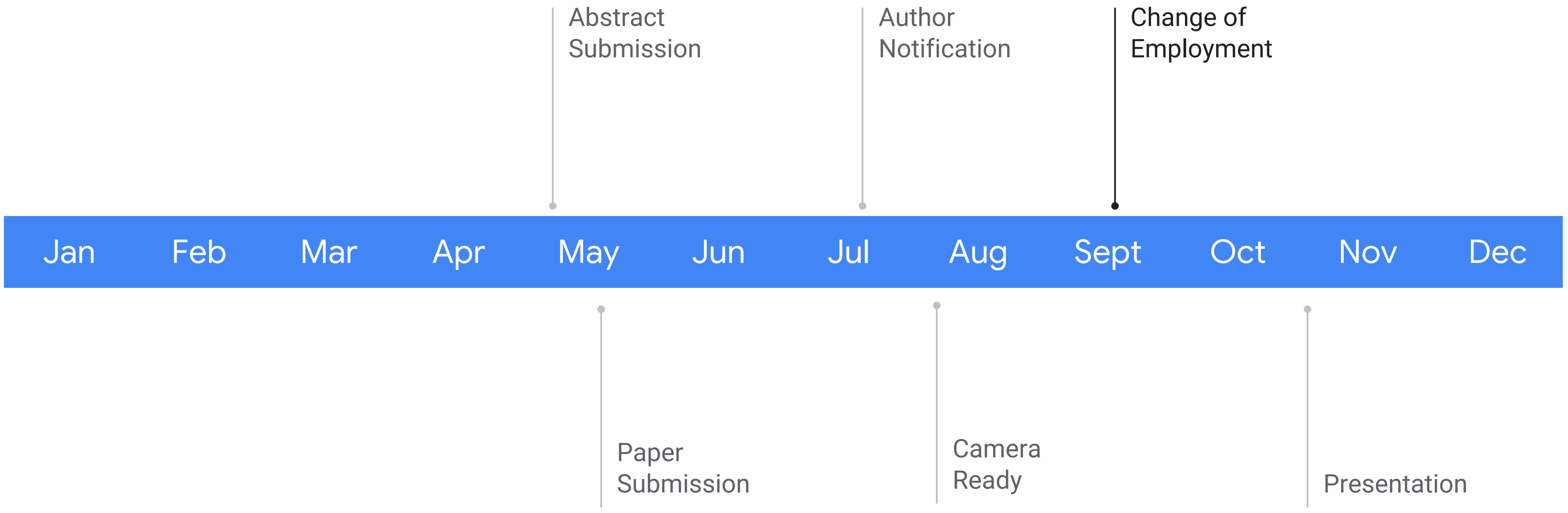
Ta'id Holmes

New Ideas and Vision Paper @ Foundations Track

Motivation	Customization of Architecture Models
Vision Idea	Automated Model Refactoring
Application	Cloud Application Orchestration
Discussion	Assumptions and Opportunities

Credits

Publication Process



Acknowledgements

- Work was carried out at **Deutsche Telekom** in collaboration with the **University of Vienna** (cf. author affiliations of published paper).
- It was partially supported by:
  - **Austrian Science Fund** (FWF)
    Project ADDCompliance (no. I – 2885)
  - **Austrian Research Promotion Agency** (FFG)
    Project DECO (no. 864707)
- Conference participation and presentation of the contribution is conducted with **Google**.

Google

Motivation

Customers have Individual Requirements

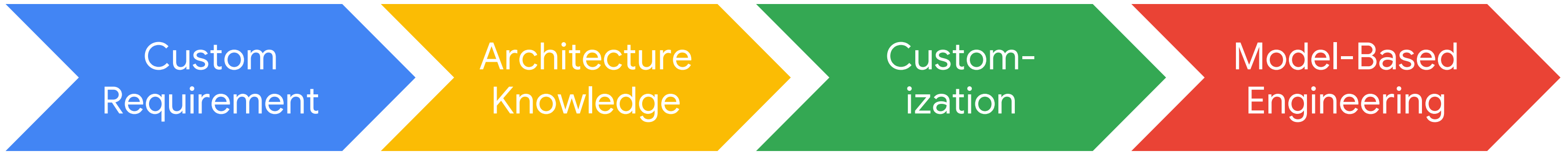
Not Always Known @ Design Time

... and Subject to Change

Often, New Requirements Emerge

⇒ Architecture Models Need to be Customized

Customizing Software-Intensive Systems



Security Policies

Operational Requirements

Local Legislations

Vision Idea

Compliance Through Refactorings

“a **change** to the internal **structure** [...] to **improve** [...] characteristics without changing its [...] behavior.”

Martin Fowler

Refactoring: Improving the Design of Existing Code. 1999, Addison-Wesley

1. Deriving AK Out of Requirements First

2. Formalizing AK in Model Transformations

Application

Cloud Application Orchestration

Cloud Application Orchestration **Models** describe Cloud Applications:

- Resources (e.g., Networking, Compute, Memory, Storage)
- Dependencies

Consumed by an **Orchestration Engine**

- Automates Deployment of an Application

Deriving Architectural Knowledge from Requirements

- **Architecture:** high availability of services
- **Legislator:** API for Legal Interception
- **Operations**
    - Secure Shell access over bastion host to instances
    - unified solutions, e.g., for logging or monitoring
- **Security:** firewalls mandated for protecting services

Capturing Architectural Knowledge in Transformations

Model refactorings can be realized using, e.g.,

- **Epsilon Object Language (EOL)**

In particular:

- Epsilon Wizard Language (EWL)
- **Epsilon Validation Language (EVL)**

Protecting a Service with a Firewall (EWL)

```
wizard Firewall_Protection {
          guard : self.isKindOf(Model!ao::ConnectionPoint) and
                    self.isPublic() and not self.isProtected()
          title : "inserting a firewall for protecting " + self.name
          do { self.owningModel.insertFirewall(self) }
}
```

Establishing High Service Availability (EVL)

```
context High_Availability {
        guard : self.isKindOf(Model!ao::ConnectionPoint)
        constraint C2_derived_from_R2 {
                check : self.ha
                message : "R2 is not met by " + self.name
                fix {
                        title : "new server instance with keepalived"
                        do {
                                var service : new Service("keepalived");
                                …
}       }       }       }
```

Discussion

Customization Delivery Scenarios

Manual Customization

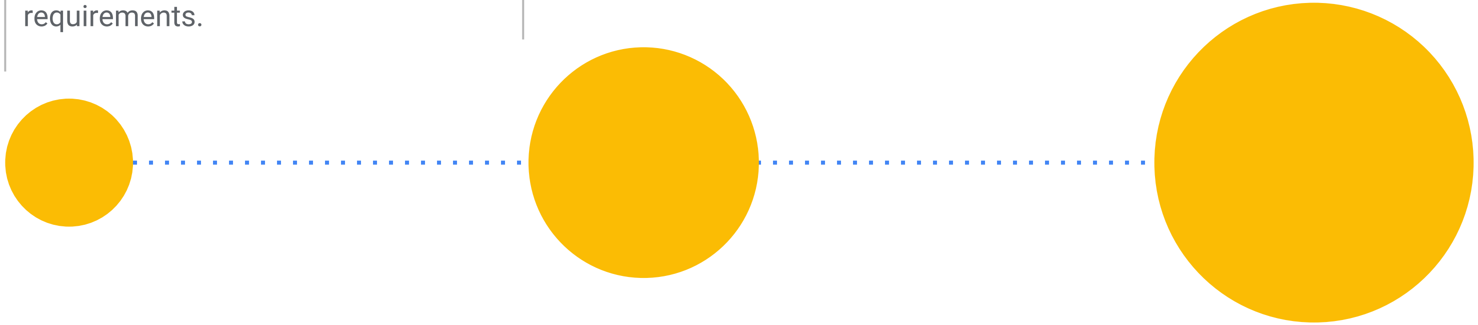
Each application is customized individually while respecting requirements.

Architectural Knowledge

Architectural knowledge is derived from requirements prior to any customization.

Model Refactoring

Transformations capturing architectural knowledge automate customizations.



Source: Holmes, T.; Zdun, U.: Refactoring Architecture Models for Compliance with Custom Requirements, ACM/IEEE MODELS, **2018**, 267-277, ACM

Evaluating Approaches

Effort	Manual	Centralized AK	Model Refactoring
Req. Disclosure (E1)	required	**not** required ✓	**not** required ✓
Arch. Knowledge (E2)	every ISV	**once** ✓	**once** ✓
Implementation (E3)	for each application	for each application	**automated** ✓
Verification (E4)	for each application	for each application	**automated** ✓
Total	E1+E2***n(ISV)**+(E3+E4)***n(Apps)**	E2+(E3+E4)***n(Apps)**	E2(Refactoring)

Key Takeaway: automated model refactoring approach scales best.

Google

Assumptions

- Architectural knowledge can be
  - derived from requirements;
  - formalized and expressed as model transformations.
- Conflicting requirements have been identified and resolved.

Further Work

- Systematic methods

  for deriving model transformations from requirements.

- Idempotency, transitivity, and order of transformations.
- Non-destructive transformations.
- Protecting parts of the architecture prior to transformations.

Thank You