Automated Configuration in Adaptive IoT Software Ecosystems to Reduce Manual Device Integration Effort: Application and Evaluation of a Novel Engineering Method

Fabian Burzlaff (University of Mannheim), Steffen Jacobs, Christian Bartelt

Contact: burzlaff@es.uni-mannheim.de

Short Resume: Fabian Burzlaff

Fabian Burzlaff works since 2016 as a scientific assistant at the Institute for Enterprise Systems (InES) and is currently in the final phase towards his PhD in Computer Science.

From 2010 until 2016, he earned his Bachelor and Master degree in Business Informatics at the University of Mannheim.

Aside from research, he has worked as a freelancer in the field of IT consulting in multiple projects. Since 2017, he is an elected member of the advisory council for a German student consultancy firm.

In his free time, Fabian spends most of his time in nature doing all kind of activities. His favorites are hiking, running and swimming.

Research Interests and Selected Projects

Presenters Research Interests

- Software Architecture
- Internet of Things
- AI-based Service Systems
- Smart Systems
- System Integration
- Web Services
- Logic and Reasoning
- Declarative Languages
- Software Ecosystems
- Microservices and REST
- Empirical Software Engineering
- Knowledge Management

Selected Projects@InES

- ARBAY: Augmented-Reality platform for selling full of variety products
  - https://arbay-projekt.de/de/
- VanAssist: Interactive and intelligent System for autonomous vans
  - https://www.vanassist.de/
- CrowdMyRegion: A social delivery network
  - https://crowdmyregion.de/

Outline of this talk

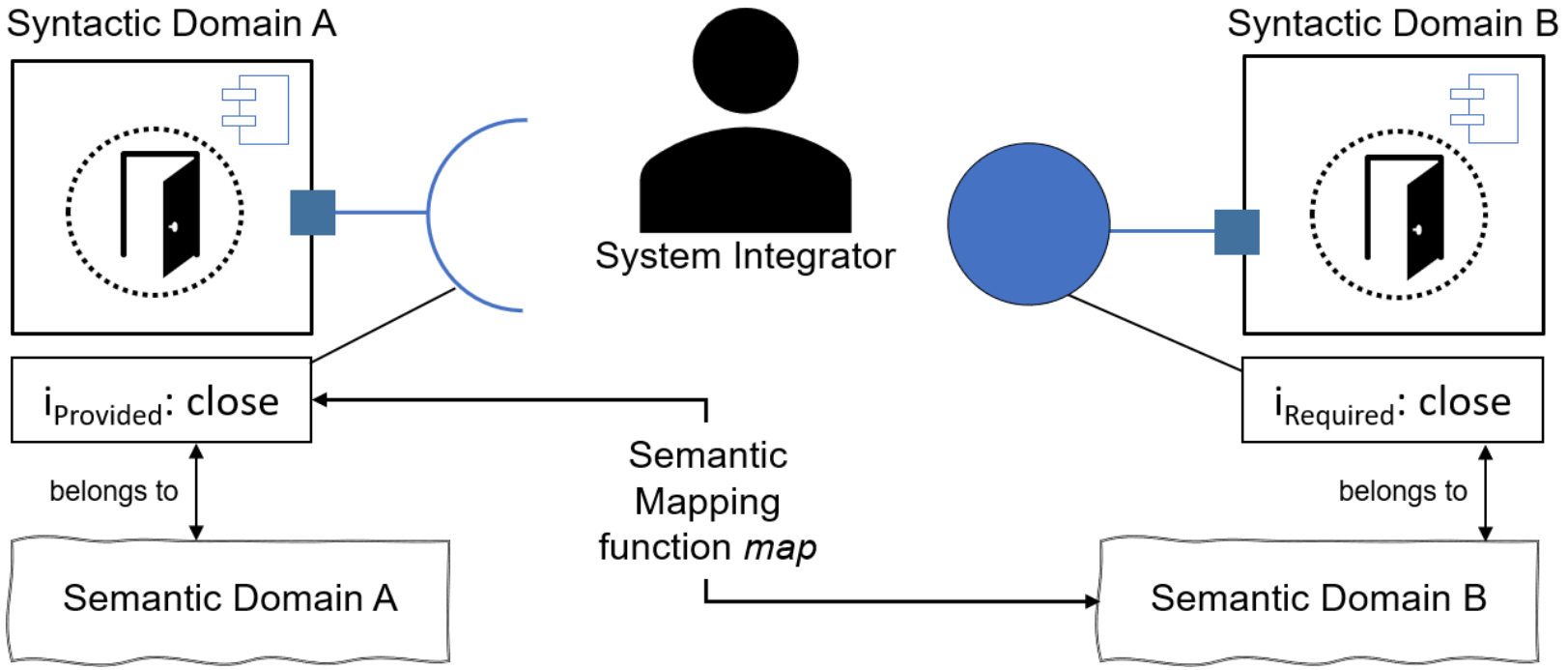
1. Use Case and Problem Statement
2. Novel Integration Method
3. Related Work
4. Evaluation Design and Results
5. Discussion

1/5 Use Case and Problem Statement

First of all, lets introduce some basic terms that we are using throughout our presentation

- Software Component: A piece of software that conforms to a component model and provides and requires functionality by the concept of an interface
- Interface Description Language: An Interface Description Language defines a set of functionality in a programming-language independent way.
- Software Component Interoperability: Software components are compatible if there exists a contract between their interfaces that maps all necessary interface description elements (i.e., they can be integrated). Semantic interoperability in distributed systems is mainly achieved by establishing semantic correspondences (i.e., mappings) between vocabularies of different services
- Semantic Component Interoperability: Semantic Interoperability ensures that services and data exchanged between a provided and a required interface makes sense - that the requester and provider have a common understanding of the "meaning" of services and data

Next, these definitions are applied on a use-case where a smart door should be replaced within an automation rule



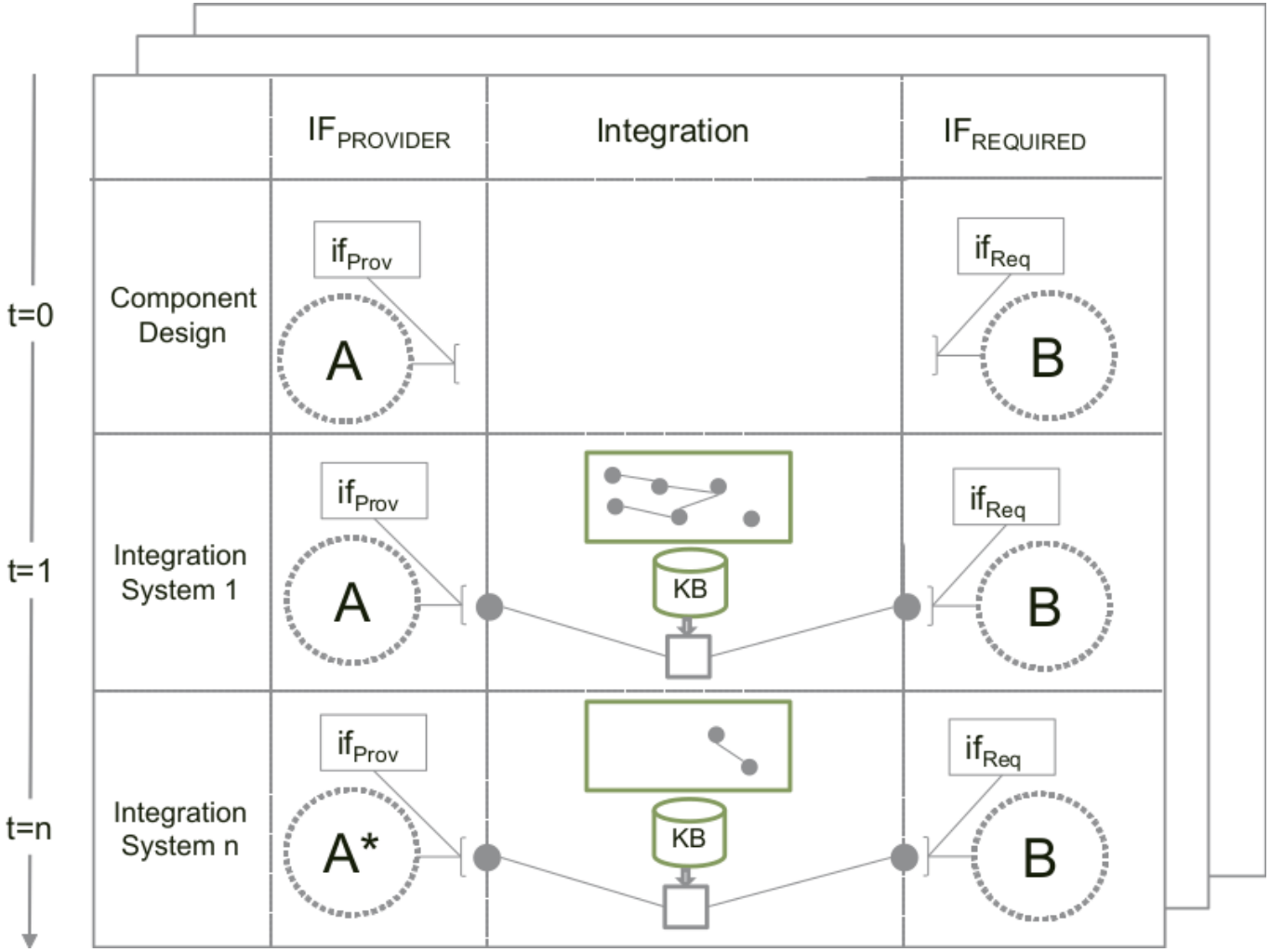
Is „close" to be translated with „next to (another object)" or with „close/opposite to open"?

For a human, this is an easy task. For a software component, this is hard.

2/5 Novel Integration Method

Semantic interoperability in distributed systems is mainly achieved by establishing semantic correspondences (i.e., mappings) between vocabularies of different services

We call our novel Knowledge-driven Architecture Composition



3/5 Related Work

In contrast to existing work, we believe that our method is novel due to the following differences

Software Engineering

- 28 years of component-based software engineering
  - Vale 2015
- Component adaptation
  - Hummel 2008, Becker 2004
  - Using test-cases for creating (semantically) correct adapter
- Fuzzy Service matching
  - Platenius 2016
  - Combination of Fuzzy matching techniques

Artificial Intelligence

- Semantic Web Service Description Languages
  - e.g. WSDL-S 2004, SAWDSL 2007
  - Attach formalized „semantics“ to interface
- Ontology Matching
  - Niepert 2011, Meilicke 2011
  - Match knowledge bases

Our Method

- Bottom-up instead of top-down
  - No predefined ontology at system design time
- Evolution instead of revolutionary
  - Incompleteness of integration knowledge is explicitly allowed
- Case-based instead of process between humans
  - Formalization effort is lowered as practitioners only formalize what they actually use
- Reuse allows for automated integration

4/5 Evaluation Design and Results

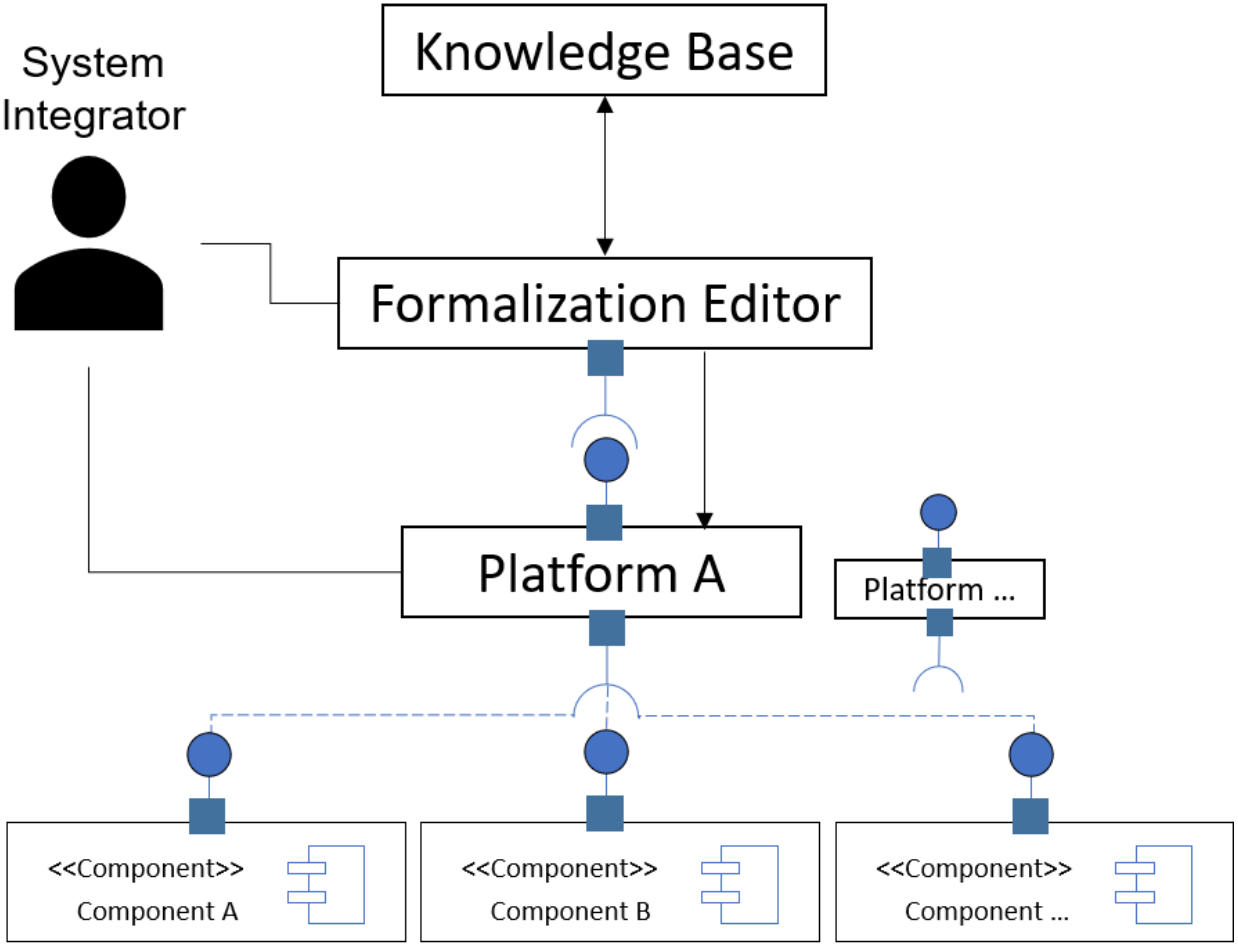
We evaluated our method with students by providing the following storyline

Storyline

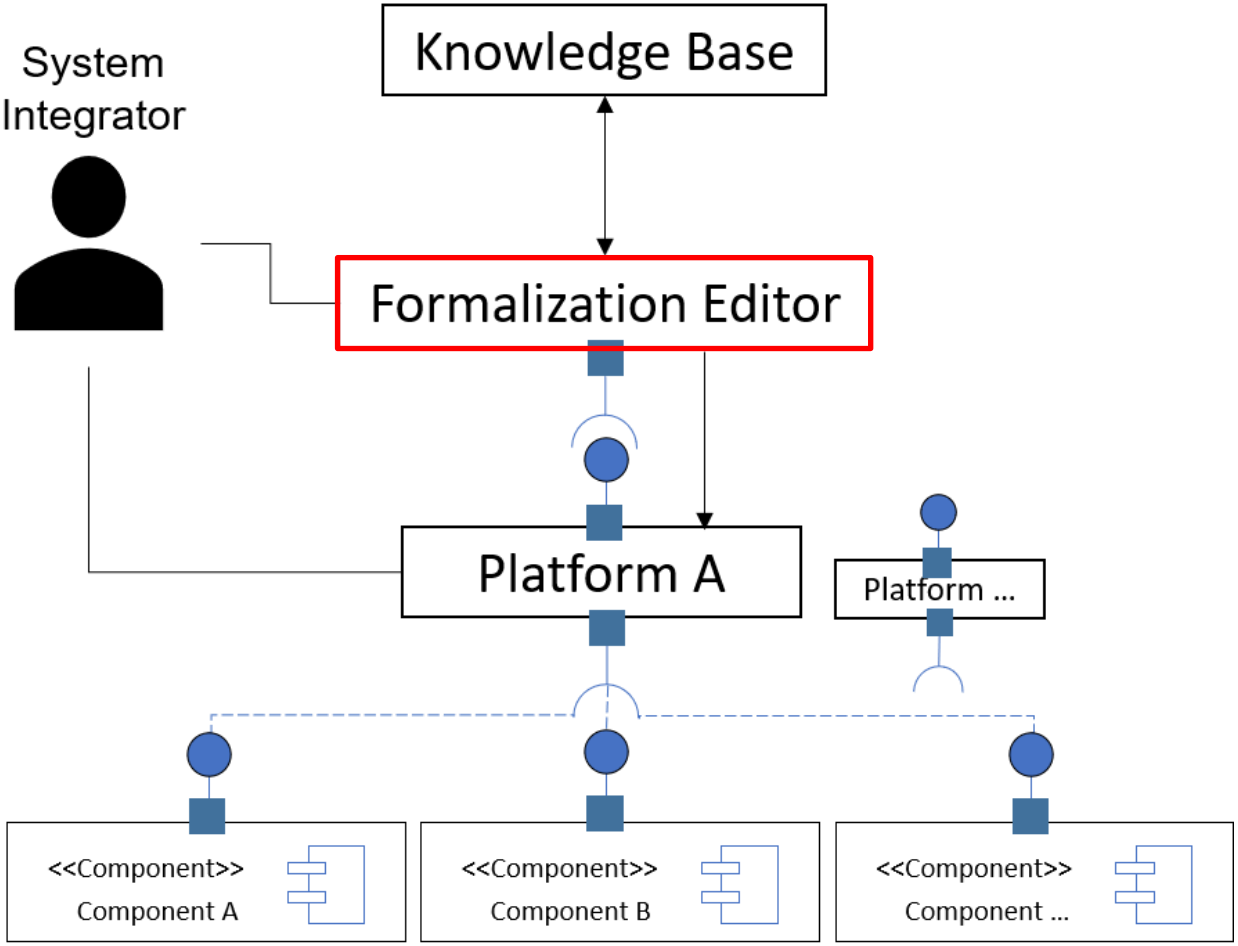
A new automation rule has been downloaded to your home automation platform. However, the rule is not working as other devices have been initially used.

Your task is to replace all data-channels until the graphical state visualization provided by the home automation platform of each device is acting accordingly to the meaning conveyed by the displayed automation rule

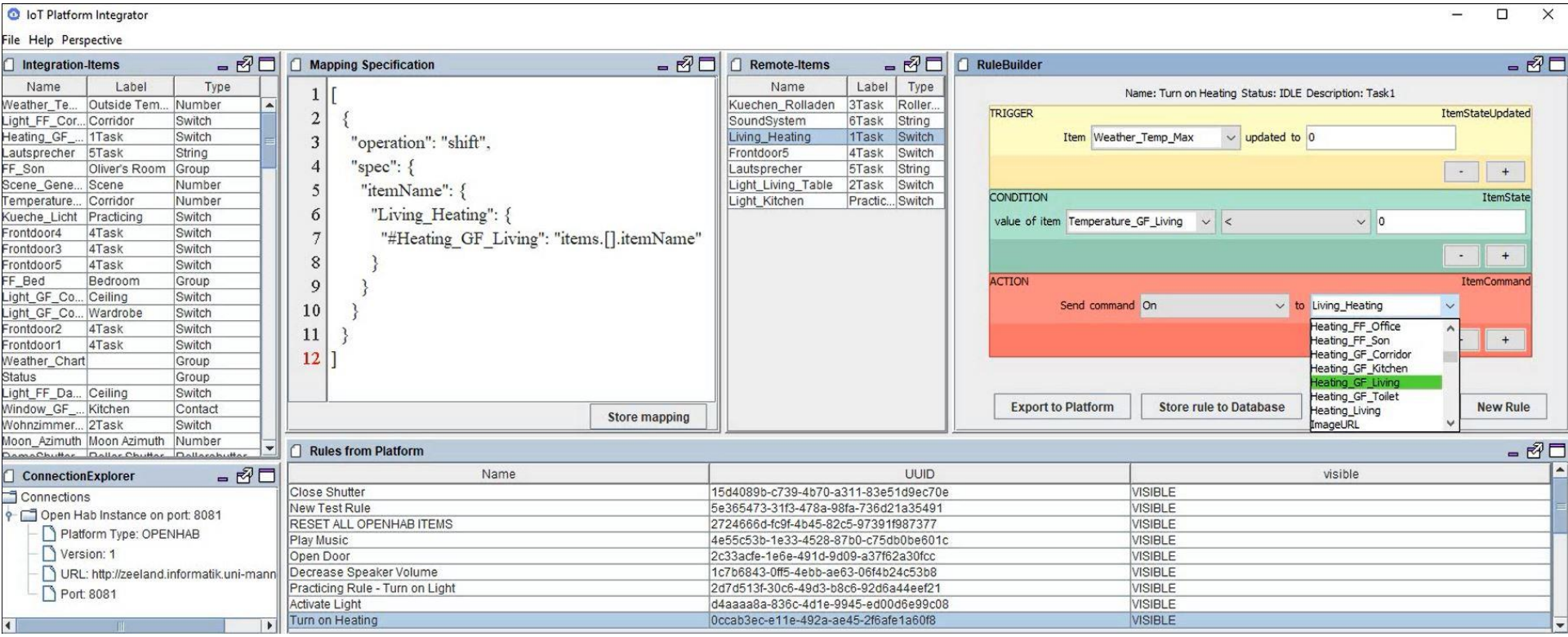
The following logical architecture was implemented to support the knowledge-driven architecture composition method



The following logical architecture was implemented to support the knowledge-driven architecture composition method



The user interface for formalizing and storing integration knowledge as well as exporting adapted automation rules was kept simple



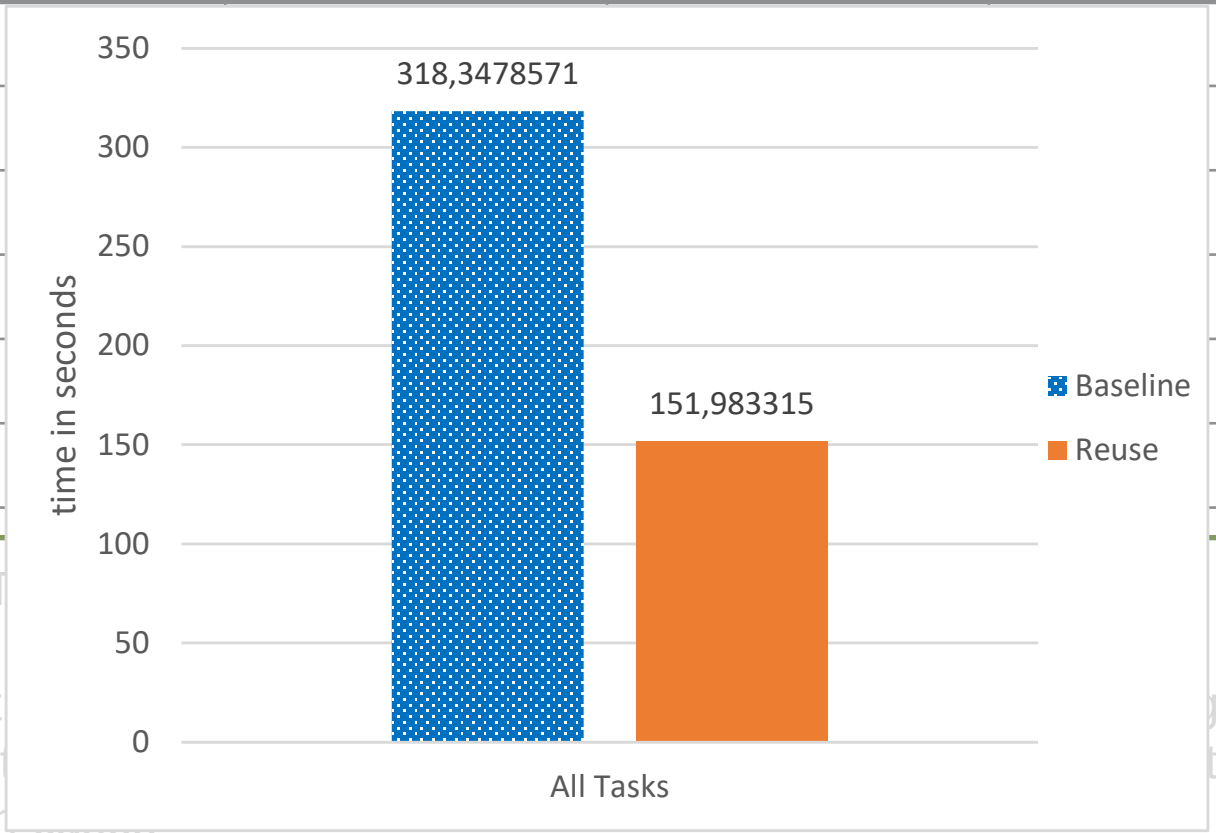
During the evaluation, overall integration time for six automation rules were measured

	Baseline	Specification	Reuse
Rule 1	42 : 12	57 : 30	38 : 22
Rule 2	38 : 10	59 : 26	0 : 0
Rule 3	32 : 9	54 : 29	25 : 13
Rule 4	31 : 10	43 : 18	0 : 0
Rule 5	31 : 10	40 : 20	30 : 16
Rule 6	34 : 16	51 : 19	0 : 0

- Average Time : Variance in seconds
- Baseline: Manual Integration
- Specification: Method-specific formalization of integration knowledge
- Reuse: Integration knowledge retrieved from knowledge base due to previous integration cases

During the evaluation, overall integration time for six automation rules were measured

	Baseline	Specification	Reuse
Rule 1			
Rule 2			
Rule 3			
Rule 4			
Rule 5			
Rule 6			



- Average T
- Baseline:
- Specificat ge
- Reuse: Int to previous integration

5/5 Discussion

Overall, the presented results can be regarded as evidence that the

Internal Threats to Validity

- Confounding variable: Use Interface Design
- Reusability of mappings was ensured early
- Only data mappings instead of services

External Threats to Validity

- Small population size (15 students) does not allow for generalization
- Situational factors: Students have been instructed in skype sessions and sometimes used multiple monitors
- Only one evaluation run

Overall, the presented results can be regarded as evidence that the

Internal Threats to Validity

- Confounding variable: Use Interface Design
- Reusability of mappings was ensured early
- Only data mappings instead of services

External Threats to Validity

- Small population size (15 students) does not allow for generalization
- Situational factors: Students have been instructed in skype sessions and sometimes used multiple monitors
- Only one evaluation run

Executive Summary of this paper

We applied and evaluated a novel integration method called Knowledge-driven Architecture Composition that explicitly allows for manual integration effort without sacrificing reliability.

Therefore, we propose an architecture, provide tool support and empirically evaluate them with students.

In the future, we plan to expand our method towards HTTP services and more complex reusability principles

End – Thank you for your attention!

Backup

Overall Average Participant Performance