

Validating Syntactic and Semantic Errors of a Domain-Specific Language for Defining Clinical Pathways in Integrated Care Platforms

Selin Erdem, 02.05.2022, Munich

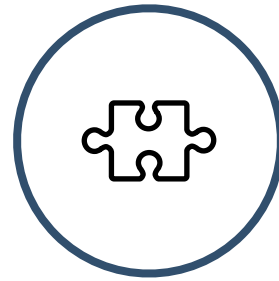
Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
www.matthes.in.tum.de



Motivation



Research
Questions



Results



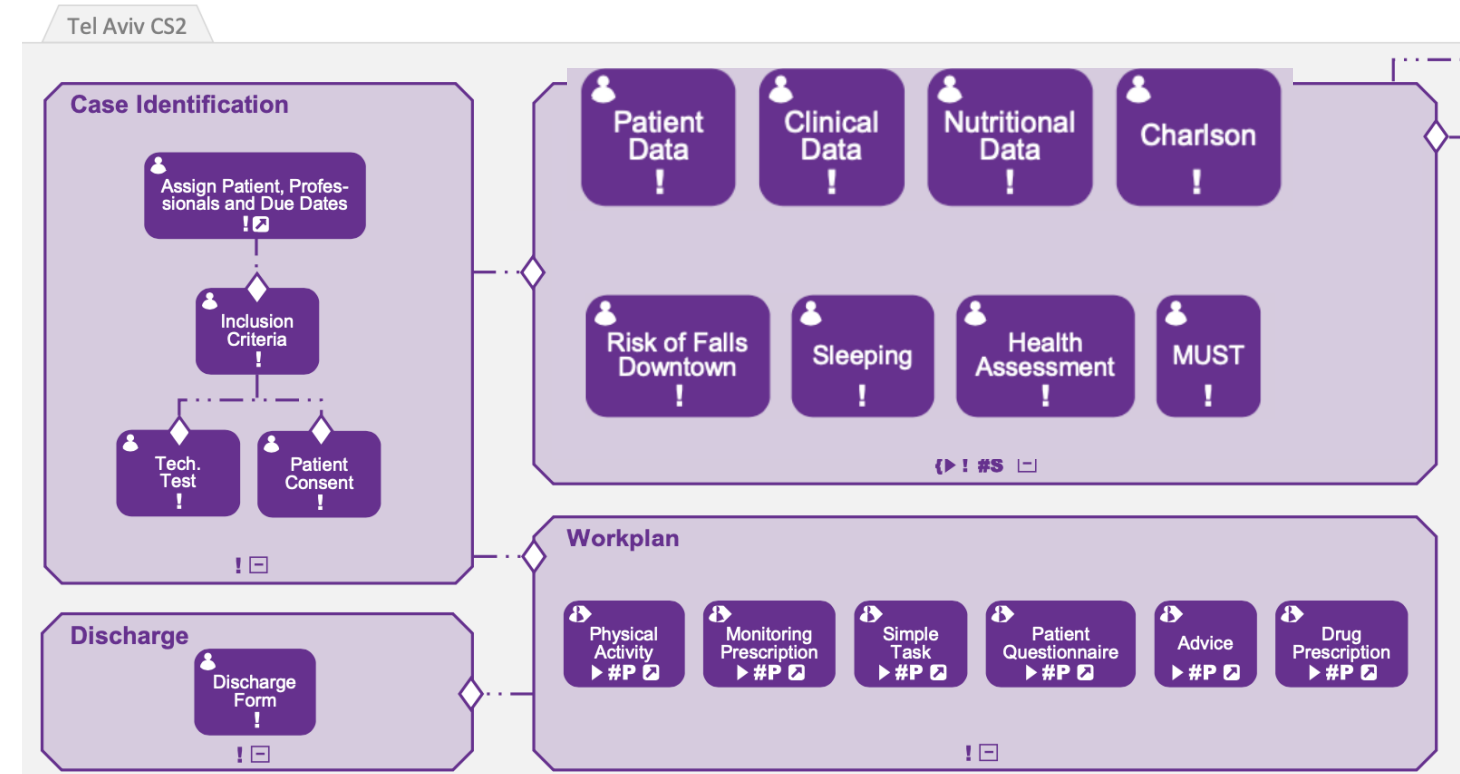
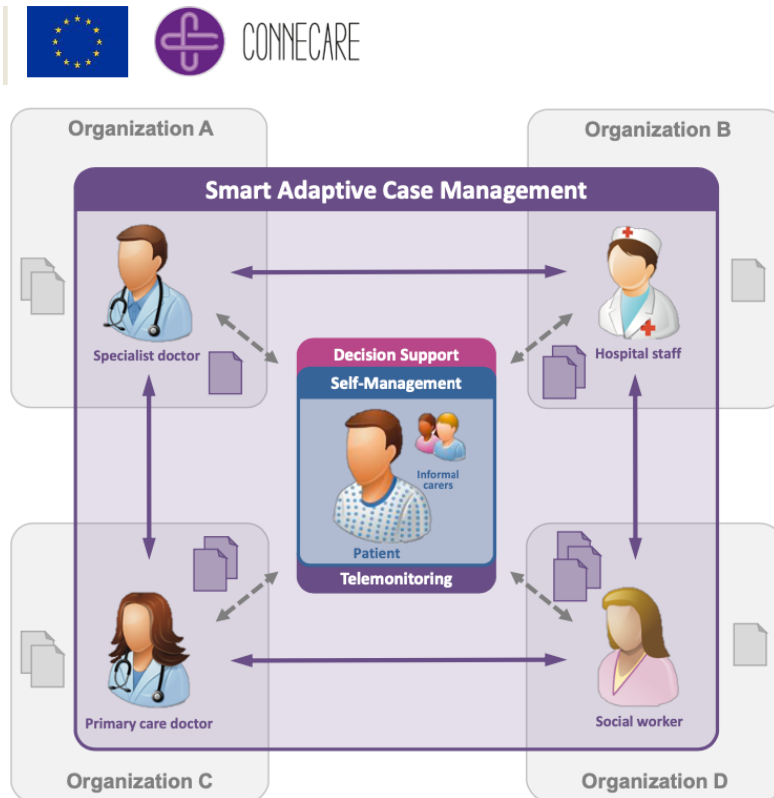
Discussion



Conclusion

Motivation

- Personalised Connected Care for Complex Chronic Patients (CONNECARE)
- Adaptive integrated care system for chronic care management
- Collaborative, meta-model-based, customizable solutions



Sample COPD treatment plan[1]

Current XML based modeling for a Stage

```
1 <StageDefinition
2   id="MCS2_CaseIdentification"
3   description="Case Identification"
4   isMandatory="true"
5   repeatable="ONCE"
6   entityDefinitionId="MCS2_Identification"
7   entityAttachPath="MCS2_Identification">
8
9   <!-- place <HumanTaskDefinition> here--->
10  <!-- place <AutomatedTaskDefinition> here--->
11  <!-- place <DualTaskDefinition> here--->
12
13 </StageDefinition>
14
15 <EntityDefinition id="MCS2_Identification" description="Identification">
16   <AttributeDefinition id="MCS2_Charlson" description="Charlson Comorbidity Index"
17     EntityDefinition.MCS2_Charlson" multiplicity="any" />
18   <AttributeDefinition id="MCS2_Barthel" description="Barthel" type="Link.EntityDef
19     multiplicity="any" />
20   <!-- Place additional <AttributeDefinition> to extend evaluation schema here -->
21 </EntityDefinition>
```



Acadela

(powered by TextX)

```
1 Stage Identification
2   #mandatory
3   #norepeat
4   owner = 'Setting.CaseOwner'
5   label = 'Identification'
```

Current XML based modeling for a Stage

```
1 <StageDefinition
2   id="MCS2_CaseIdentification"
3   description="Case Identification"
4   isMandatory="true"
5   repeatable="ONCE"
6   entityDefinitionId="MCS2_Identification"
7   entityAttachPath="MCS2_Identification">
8
9   <!-- place <HumanTaskDefinition> here--->
10  <!-- place <AutomatedTaskDefinition> here--->
11  <!-- place <DualTaskDefinition> here--->
12
13 </StageDefinition>

1 <EntityDefinition id="MCS2_Identification" description="Identification">
2   <AttributeDefinition id="MCS2_Charlson" description="Charlson Comorbidity Index"
3     EntityDefinition.MCS2_Charlson" multiplicity="any" />
4   <AttributeDefinition id="MCS2_Barthel" description="Barthel" type="Link.EntityDef
5     multiplicity="any" />
6   <!-- Place additional <AttributeDefinition> to extend evaluation schema here -->
7 </EntityDefinition>
```



Acadaela

(powered by TextX)

```
1 Stage Identification
2   #mandatory
3   #norepeat
4   owner = 'Setting.CaseOwner'
5   label = 'Identification'
```



Usability: more readable + user-friendly



Learnability: Intuitive grammar



Error handling in textX is not user friendly

Syntax Errors

- TextX has built-in syntax error handler
- Error messages are non-intuitive, technical-oriented
 - ➡ Error messages needs to be improved

Semantic Errors

- Semantic constraints are not defined e.g. path validations
 - ➡ Constraints should be identified and validated
 - ➡ Need user-friendly error messages

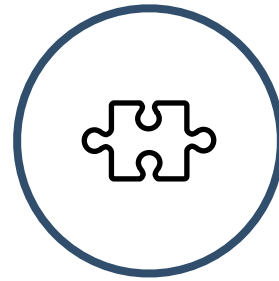
Goal: User-friendly and accurate error handler
Reducing learning and debugging effort



Motivation



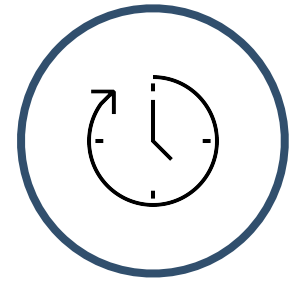
Research
Questions



Results



Discussion



Conclusion

RQ1

What **syntactic** or **semantic errors** can be validated by the DSL during the modeling of clinical pathways?

RQ2

How to **identify and interpret the errors** to modelers in an **accurate, user-friendly manner**?

RQ3

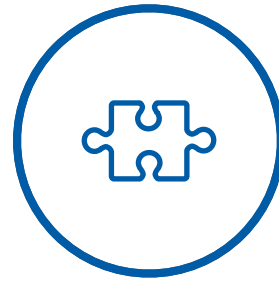
What is the **opinion of modelers** regarding the **usability and accuracy** of the error validator?



Motivation



Research
Questions



Results



Discussion



Conclusion

RQ1

What syntactic or semantic errors can be validated by the DSL during the modeling of clinical pathways?

1. Unexpected Elements

- The elements which are not placed under the correct parent element or not recognised by the grammar

```
TextXSyntaxError: Expected Hash or 'label' or 'Question' or  
'CustomFieldValue' or 'uiRef' or 'externalId' at position /path/to/  
dsl-conneccare/acadela/model_placeholder:(112, 21) => '*labl = 'Sy'.
```

Constraints for Syntax Validation

1. Unexpected Elements

- The elements which are not placed under the correct parent element or not recognised by the grammar

```
TextXSyntaxError: Expected Hash or 'label' or 'Question' or  
'CustomFieldValue' or 'uiRef' or 'externalId' at position /path/to/  
dsl-connecare/acadela/model_placeholder:(112, 21) => '*labl = 'Sy'.
```

2. Typos

- Detect typos, suggest meaningful replacements using a spell-checker
 - Attribuute—>Attribute
 - Owneet---> owner

Constraints for Syntax Validation

3. Explaining Technical Terminology

- Error messages include technical terms such as INT, STRING, EQ

```
TextXSyntaxError: None:175:25: error: Expected STRING at position  
/path/to/dsl-connecare/acadela/model_placeholder:(175, 25)  
=> 'ndition = *Evaluation'.
```

Constraints for Syntax Validation

3. Explaining Technical Terminology

- Error messages include technical terms such as INT, STRING, EQ

```
TextXSyntaxError: None:175:25: error: Expected STRING at position  
/path/to/dsl-connecare/acadela/model_placeholder:(175, 25)  
=> 'ndition = *Evaluation'.
```

4. Validating String Patterns

- Several attributes must follow certain string patterns which are not defined in the original grammar
 - ▶ 'round(Weight / (Height * Height))'
 - ▶ 'colors(5<orange<=18<green<=25<red<100)'

Constraints for Semantic Validation

1. References and Paths

- Elements can be referenced from other elements using a dot notation
 - ▶ `<Stage>.<Task>.<Field>`
 - ▶ `owner = 'Setting.Clinician'`

Constraints for Semantic Validation

1. References and Paths

- Elements can be referenced from other elements using a dot notation
 - ▶ `<Stage>.<Task>.<Field>`
 - ▶ `owner = 'Setting.Clinician'`

2. Unique Element Identifiers

- Some elements of CPs modelled in the system have to have unique identifiers
- Eg: two stages cannot have the same IDs

3. Trusted URLs

- Validating the URL and HTTP method defined in hooks

```
Trigger
  On complete
  invoke 'https://server1.com/api2'
  method POST
  with failureMessage 'Cannot complete the data creation!'
```

Workspace Name	URL	Methods
Umcg	http://127.0.0.1:3001/connecare	POST
Umcg	https://server1.com/api2	POST , GET
Umcg	http://integration-producer:8081/v1/activate	
Umcg	localhost:3001/connecare	

Usability Constraints for Error Messages

Goal: Readability and understandability^[1]

1. Cause of the error

2. Location of the error

3. Suggestions to solve the error



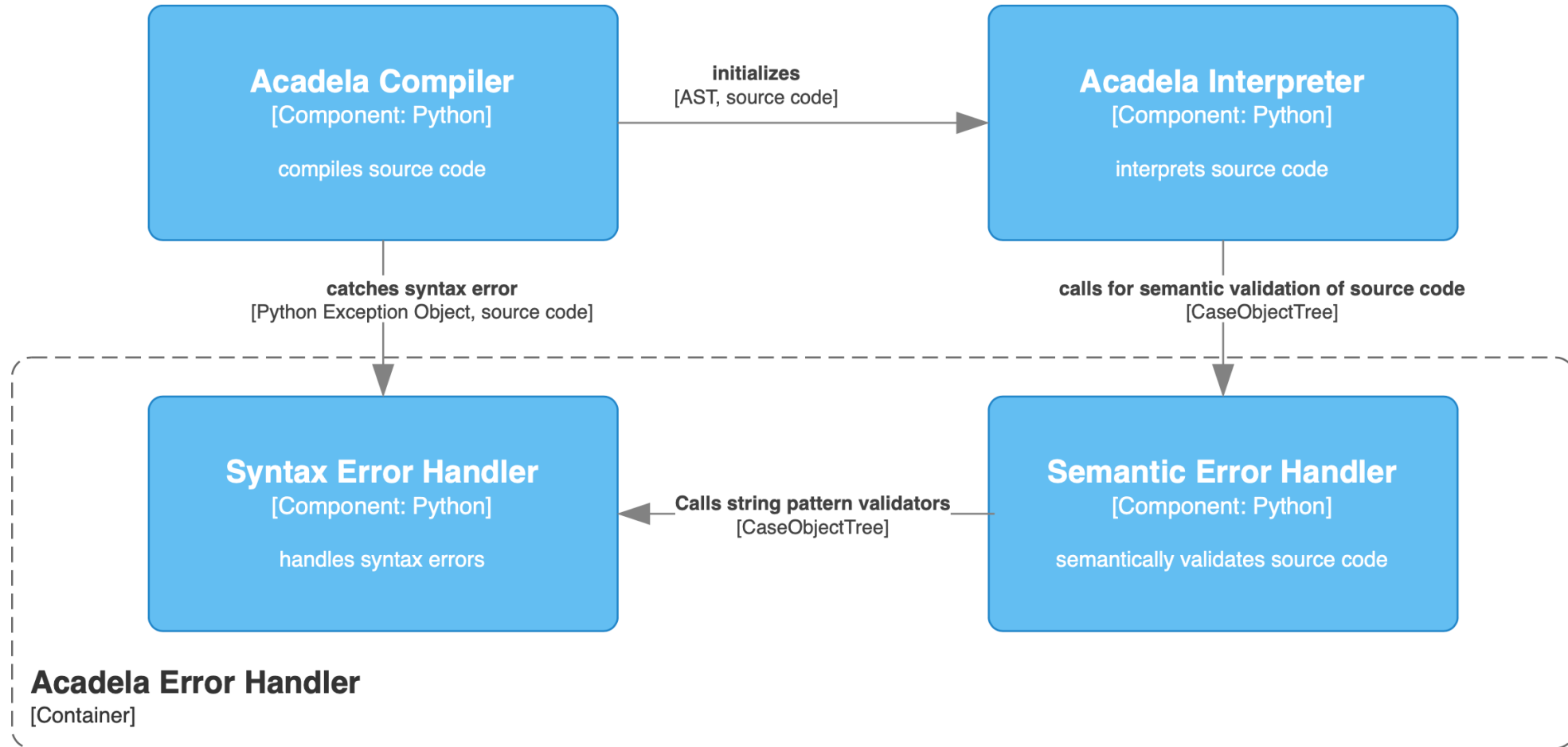
+ Ease of debugging

+ Usability

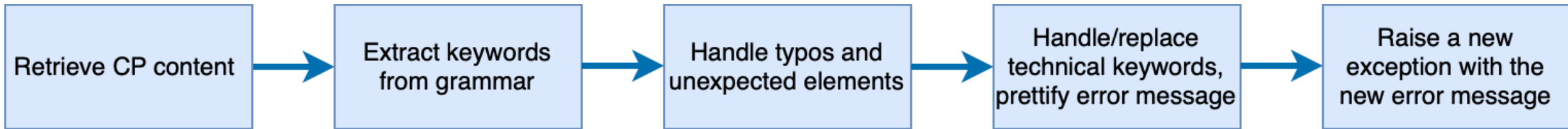
[1] <https://www.nngroup.com/articles/error-message-guidelines/>

RQ2

How to identify and interpret the errors to modelers in an accurate, user-friendly manner?



Unexpected elements, typos and keyword handling:



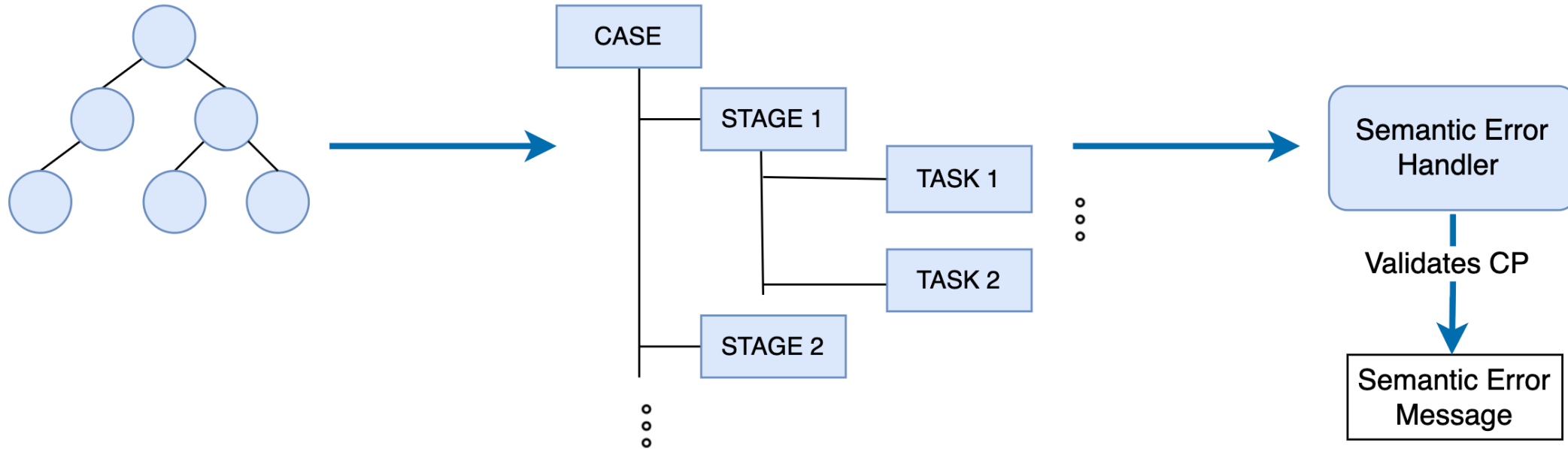
String patterns:

- ➡ New mini textX grammars for each string pattern e.g. if-else statements, functions
- ➡ Validated after compilation, during interpretation

Implementation - Semantic Error Handler

Abstract Syntax Tree(AST)

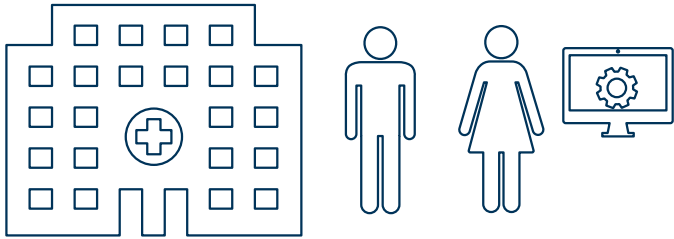
Case Object Tree



Demo

RQ3

What is the opinion of modelers regarding the usability and accuracy of the error validator?



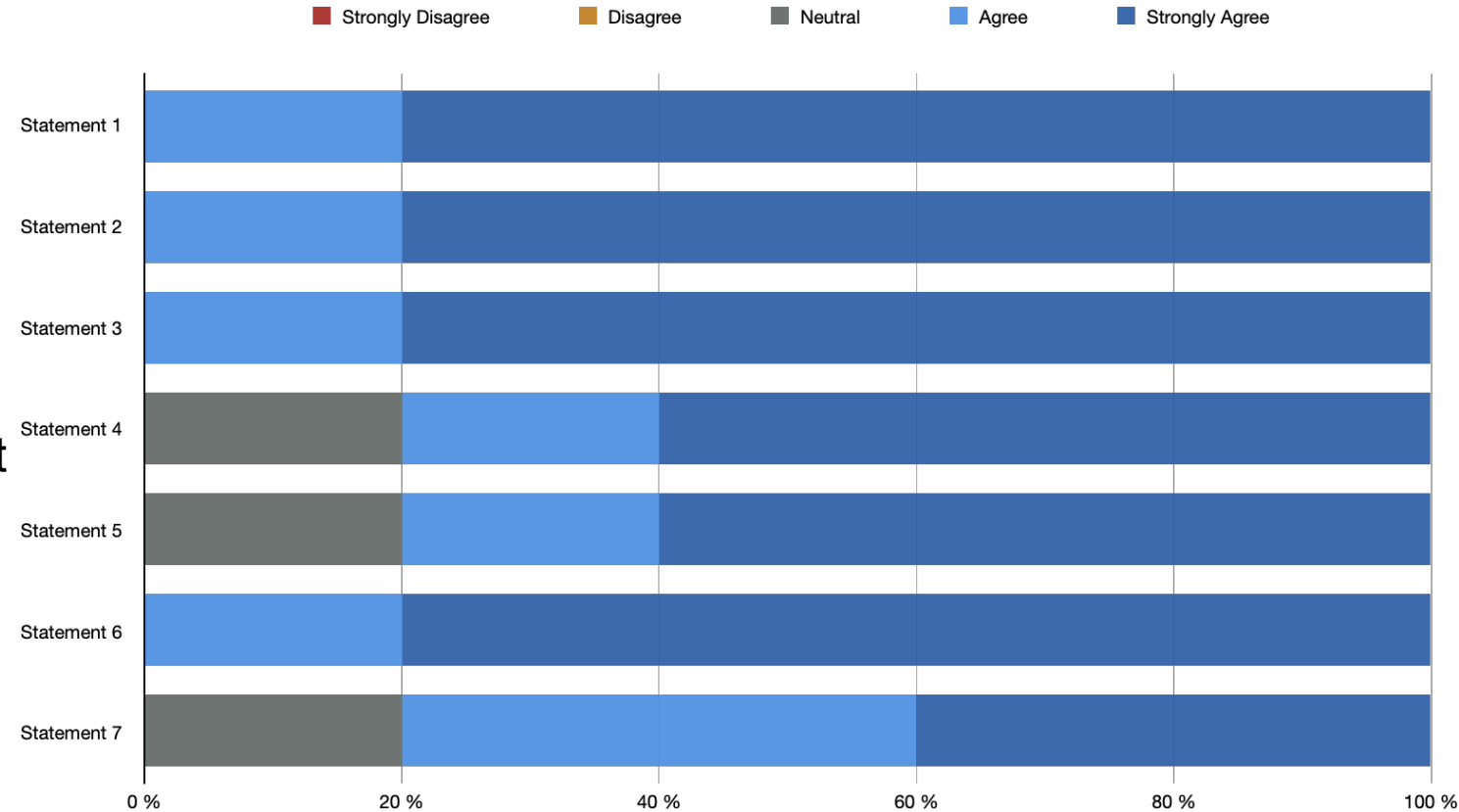
- User studies with 5 participants
- 50-minute training session + online experiment session
 - Part 1:** Solving a small modeling task by completing an existing CP model
 - Part 2:** Fixing errors in an existing model using error messages

	Occupation	Age	YOE	Programming Languages	Modeling Languages
1	Research Assistant	36-45	15	C, C++, Java, JS, Python, R	XML, UML, BPMN, CMMN, SQL
2	Researcher	36-45	2	PlantSimulation	-
3	Research Assistant	26-35	1	C#, Python	BPMN
4	Medical Doctor	46	20	Basic, Pascal	SQL
5	Technician	26-35	9	Java, Python	XML, JSON, SQL

Evaluation of the Error Validator

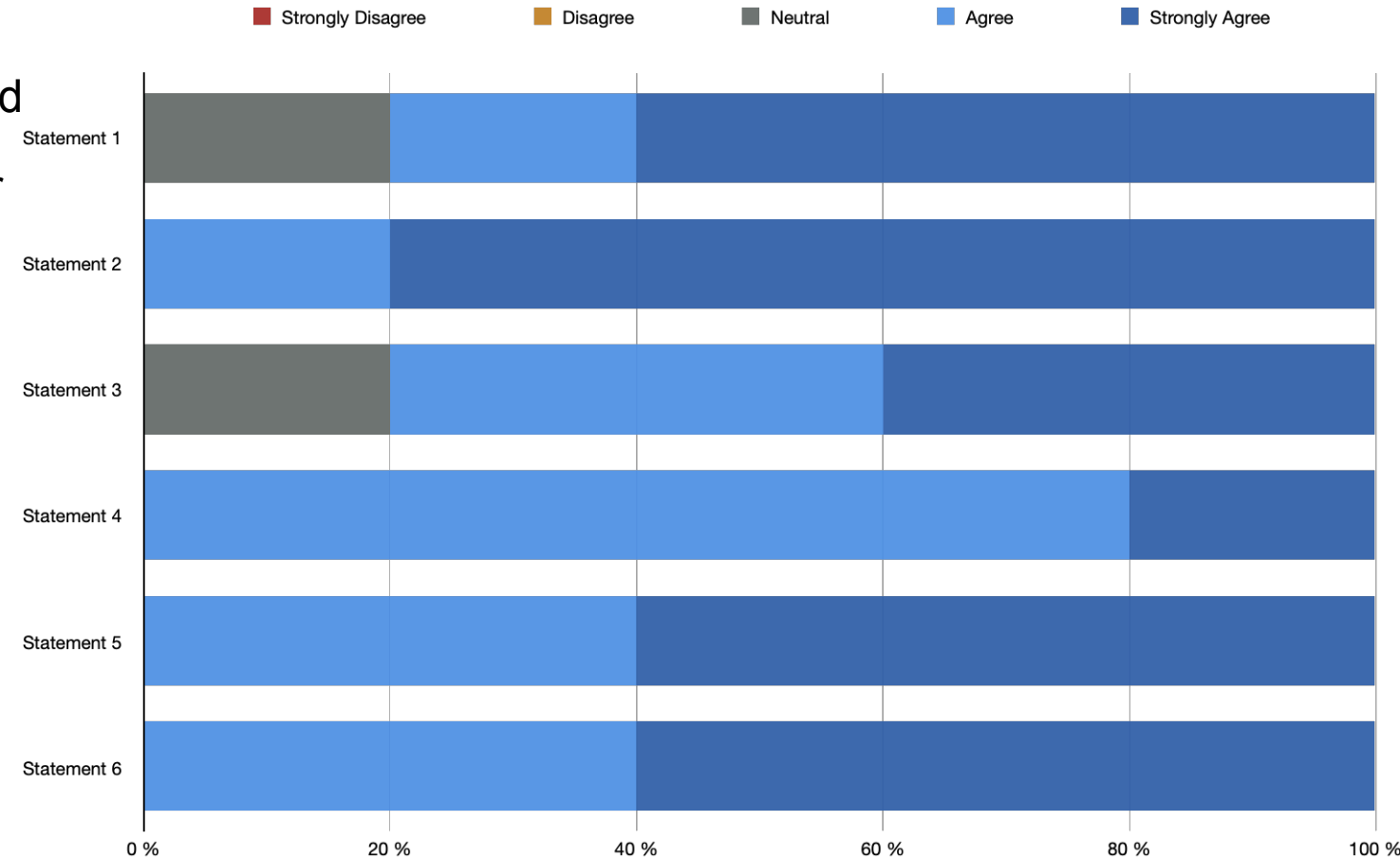
The error messages helped me:

1. to fix unexpected attributes
2. to fix the errors in conditional statements
3. to fix wrong directive errors
4. to fix typos
5. to locate and correct the paths to nonexistent elements
6. to fix duplicate element name
7. to fix the errors related to URLs and HTTP methods



Evaluation of the Error Validator

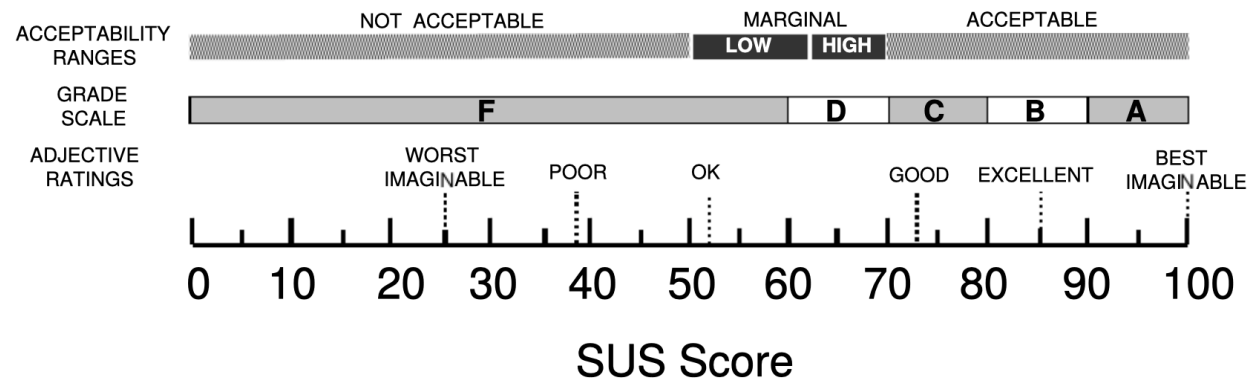
1. The error messages were easy-to-understand
2. I was easily able to locate the source of error using the error messages
3. I was able to fix the errors easily using the error messages
4. The language of the error messages is clear and precise
5. The error messages were accurate
6. The error messages were consistent



Evaluation of the Acadela Language

Usability: System Usability Scale (SUS)

	Profession	System Usability Score
1	Research Assistant	62,5
2	Researcher	55
3	Research Assistant	87,5
4	Medical Doctor	95
5	Technician	77,5
Average Score		77,5

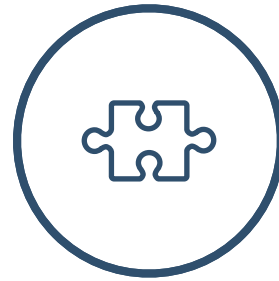




Motivation



Research
Questions



Results



Discussion



Conclusion

Limitations

Limitations of textX

- textX documentation

Limitations of the Proposed Solution

- Edge cases

Limitations of the Evaluation

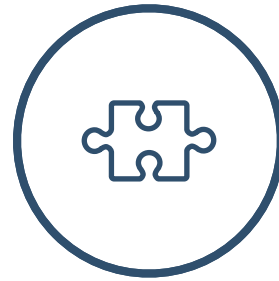
- Sample Size
- Duration



Motivation



Research
Questions



Results



Discussion



Conclusion

Conclusion

- ➡ It is possible to create a error validator and catch variety of semantic and syntactic errors in Acadela
- ➡ Error messages are user friendly and helped the modellers with varying backgrounds to debug clinical pathways

Future Work

- ➡ Covering more edge cases
- ➡ More evaluations with more participants
- ➡ Letting the modellers to construct real clinical pathways



Selin Erdem

Technische Universität München
Faculty of Informatics
Chair of Software Engineering for Business
Information Systems

Boltzmannstraße 3
85748 Garching bei München

Tel +49.89.289.
Fax +49.89.289.17136 17132

www.matthes.in.tum.de
matthes@in.tum.de



1. I think that I would like to use this language frequently.
2. I found this language unnecessarily complex.
3. I think that this language is easy to use.
4. I think that I would need assistance to be able to use this language.
5. I found the various functions such as defining elements, importing modules, and error validations in this language were well integrated.
6. I thought there was too much inconsistency in this language.
7. I would imagine that most people would learn to use this language very quickly.
8. I found this language very cumbersome to use.
9. I felt very confident using this language.
10. I needed to learn a lot of things before I could get going with this language.

System Architecture

