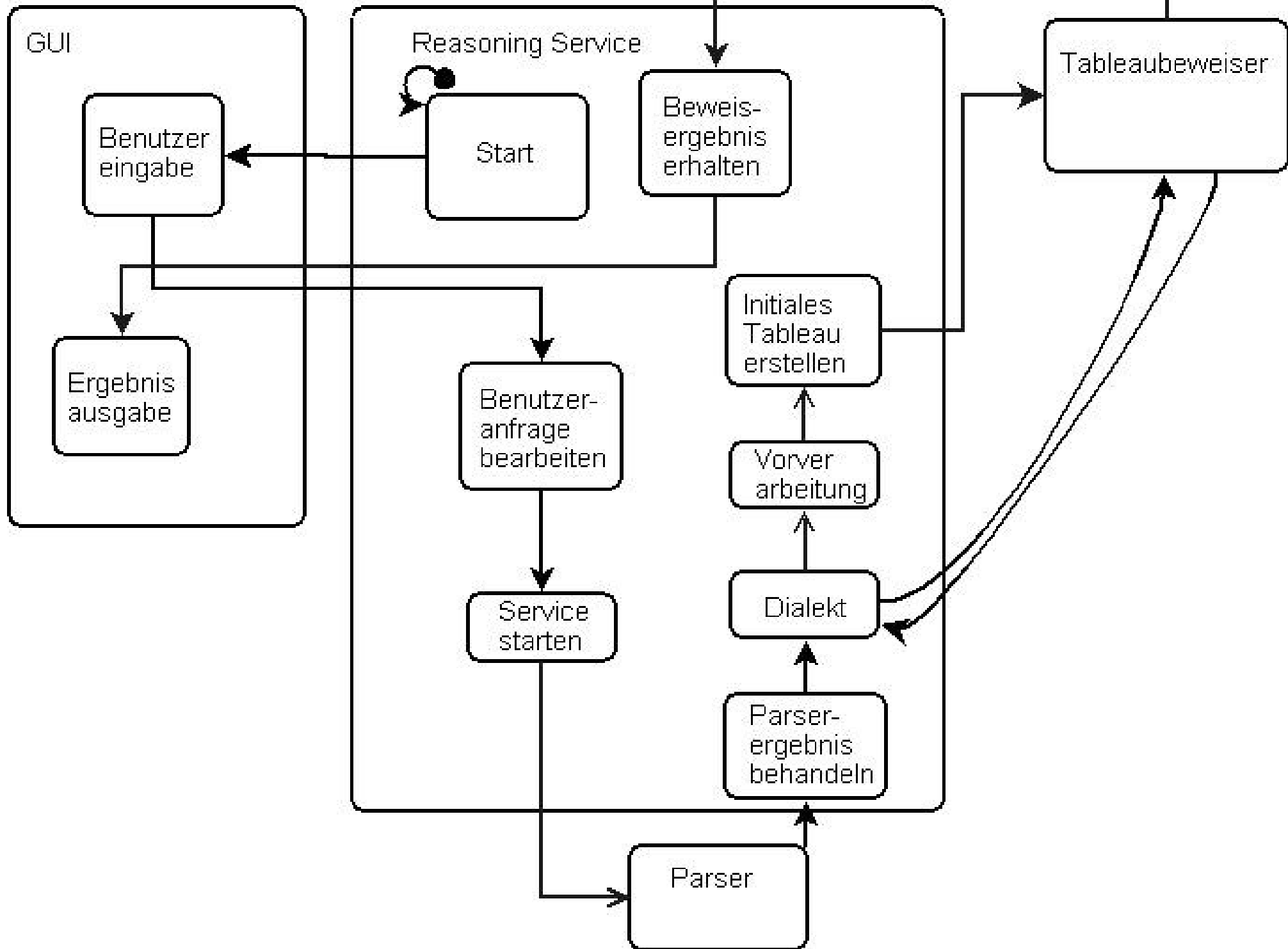


Reasoning-Services

Felix Lindner, Oliver Gries



Klassenübersicht

Package userInterface

- GuiComponentUtil

Package main

- TableauBeweiserApplet
- TableauBeweiser
- Service
- AbstractService
- ServiceSubsumption, -Satisfiability, -Disjointness, -Equivalence, -TBoxSatisfiability, -InstanceCheck
- Neue Exceptions

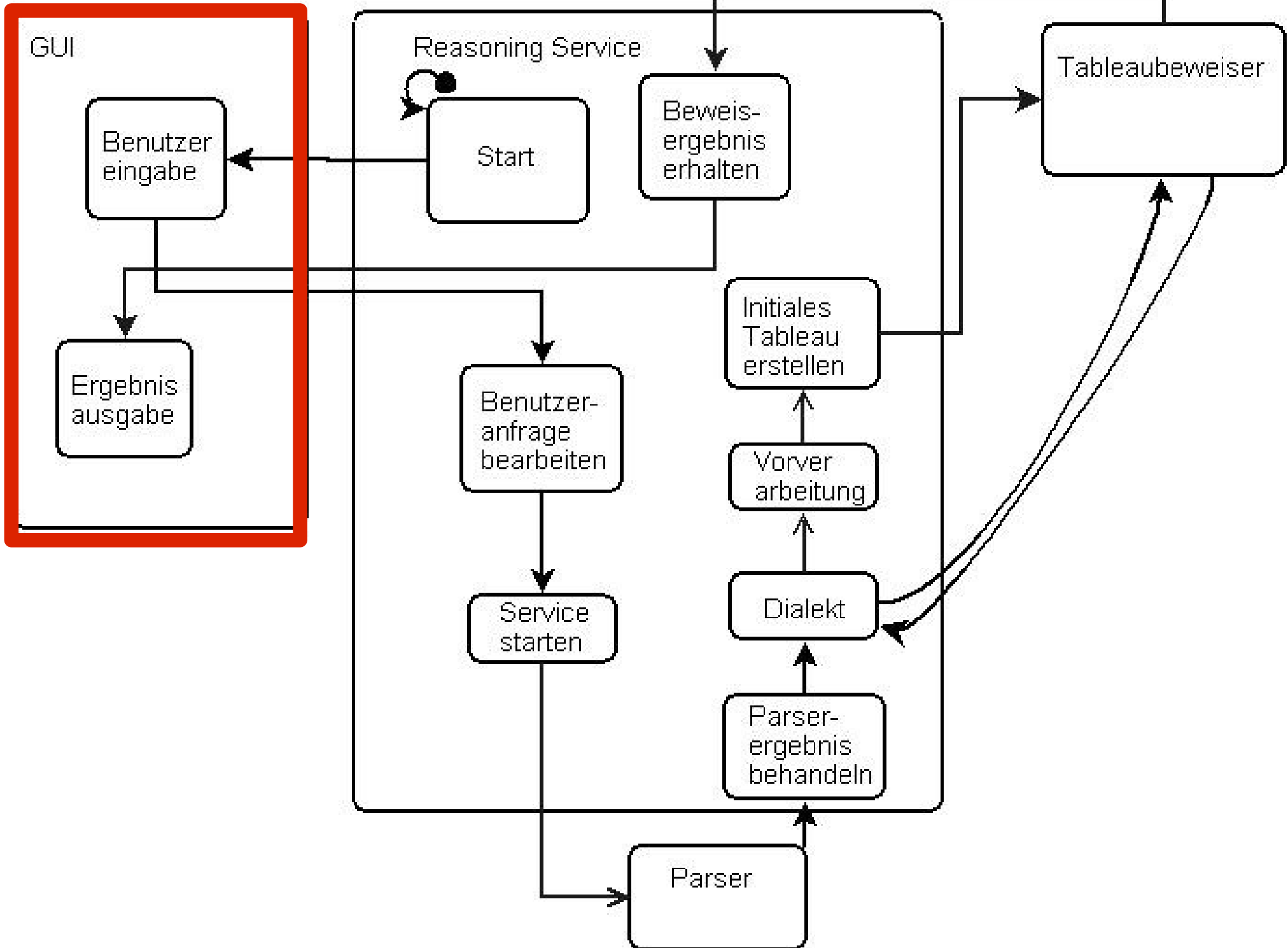
Neue Exceptions

- **ParserExceptionExpanded**

Enthält `ParserException` und Information darüber, in welchem Feld der Fehler durch den Cursor angezeigt werden soll.

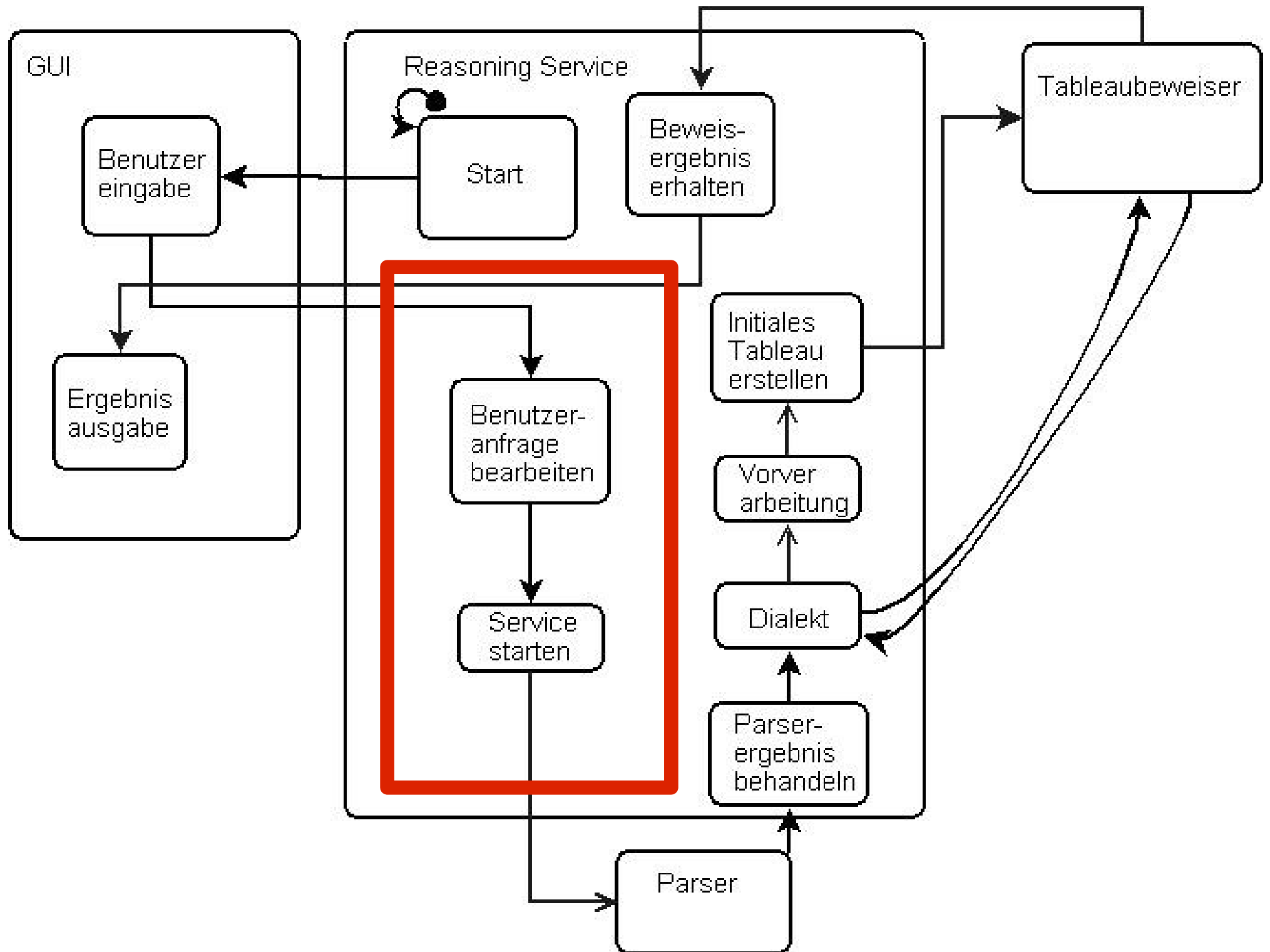
- **TBoxException**

Wird geworfen, falls die `TBox` nicht definitiv ist.



GuiComponentUtil.java

- `String[] reasoningServices = {Satisfiability, ...};`
- `JComboBox reasoningChoice;`
- `JCheckBox tBoxOnOff;`
- `JScrollPane wTextPane, eTextPane;`
- `JLabel tBoxLabel, aBoxLabel, wTextPaneLabel, eTextPaneLabel, whiteLabel;`



TableauBeweiser.java

neue Instanzvariablen

- Service myService;
- boolean isDL;
- (GUI-Elemente)

erweiterte Methoden

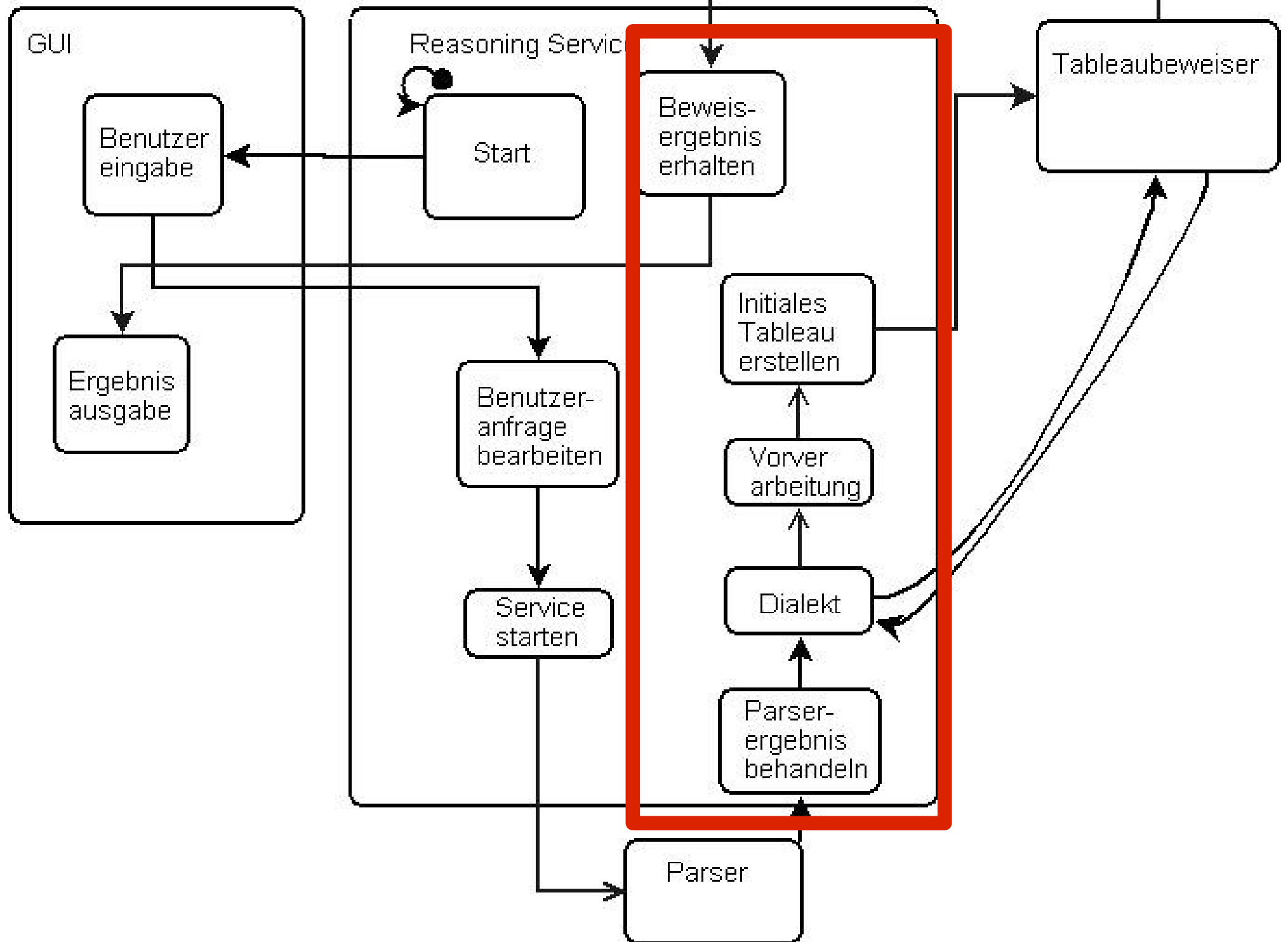
- actionPerformed();
- execute(); (früher validTest(), führt die Services aus)
- toALGUI();
- toDLGUI();

actionPerformed();

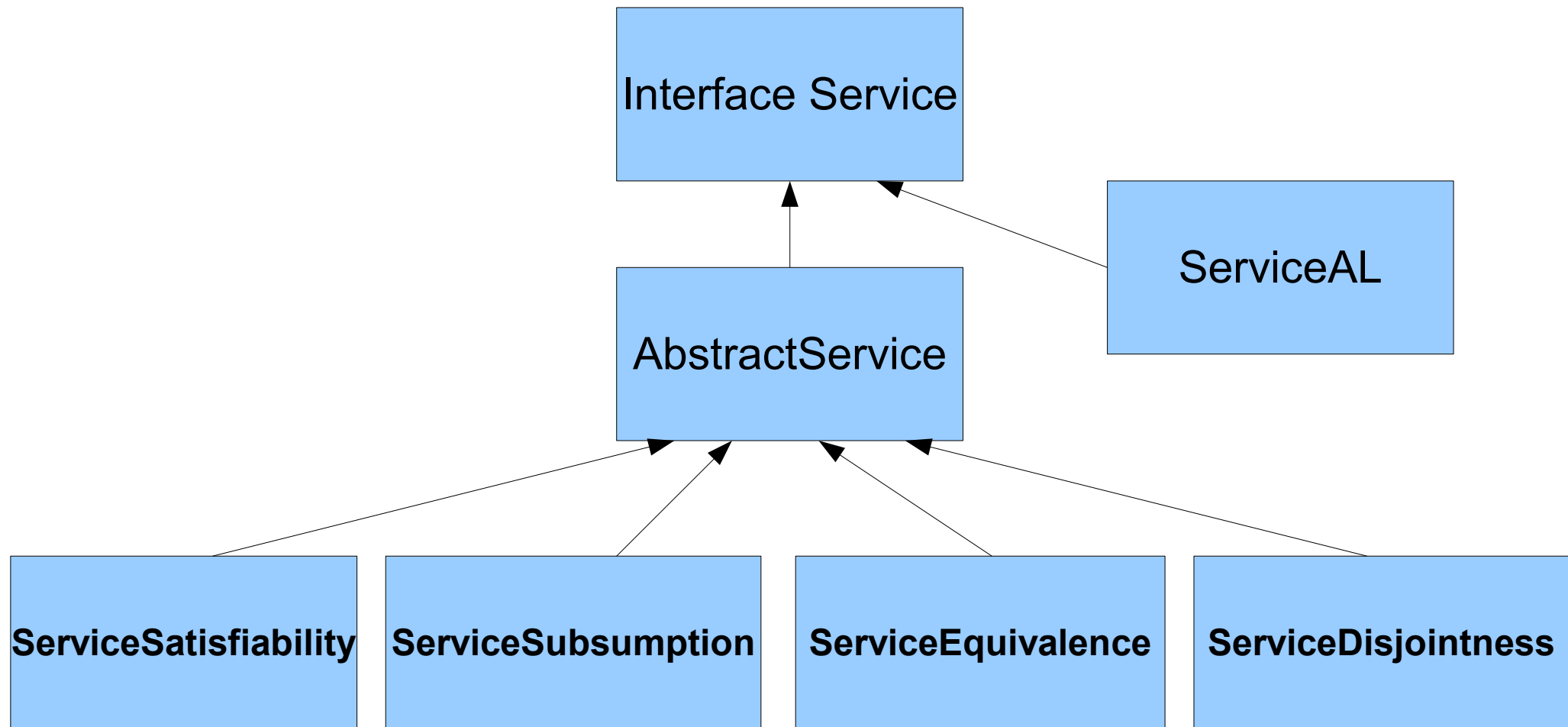
- Setzen des Services
- auf JCheckBox reagieren
- Ein- und Ausblenden des zweiten Eingabefeldes
- ToolTipText und Labels werden angepasst
- bei Einstellung des Parsers / Provers auf AL bzw. DL wird die GUI automatisch angepasst

execute();

- Strings aus Textfeldern holen
- ausgewählten Service der myService-Instanz zuweisen
- Ausführen der AL-/DL-Services mit
'myService.runService();'
- Behandlung von Exceptions, insbesondere das
Positionieren des Cursors bei ParserExceptions
- Ausgabe des Ergebnisses



Service-Hierarchie



Interface Service

'runService()' wird von allen Services
implementiert

AbstractService extends Service

Enthält die Methoden, die von allen DLReasoning-Services benutzt werden:

- `parse(String, DLType);` ruft den Parser auf
- `parseTBox();`
- `expandConcept();`
- `expandTBox();`
- `checkDialect();`
- `prove();` ruft `prover.satisfiableTest()` auf

ServiceSatisfiability

```
public ServiceSatisfiability(String concept, Parser parser,  
    Prover prover, String TBox, boolean wrtTBox)
```

- implementiert 'runService()';
- Konzept parsen
- evtl. TBox parsen und Konzept expandieren
- Dialekt checken
- 'initTableau()' : Tableau mit einem Zweig, TEntry enthält das Konzept und eine neue Konstante
- DLProver aufrufen

ServiceSubsumption

```
public ServiceSubsumption(String concept_left, String  
concept_right, Parser parser, Prover prover, String tBox,  
boolean wrtTBox)
```

Wie ServiceSatisfiability, aber:

- beide Konzepte werden geparkt und evtl. expandiert
- 'initTableau()' erstellt ein Tableau mit einem Zweig mit jeweiligem TEntry fuer das erste und der Negation des zweiten Konzeptes

ServiceEquivalence

Konstruktor wie ServiceSubsumption

- führt zwei Subsumption-Services aus:
C ist äquivalent zu D, wenn C von D subsummiert wird und D von C.

ServiceDisjointness

Konstruktor wie bei der Subsumption

- Parsen und Expandieren wie bei dem Subsumptions-Service
- 'initTableau()' erstellt ein Tableau mit einem Zweig mit den beiden Konzepten als Einträgen

Ausblick

- Services: TBox-Satisfiability, InstanceCheck
- ABox-Behandlung: UNA