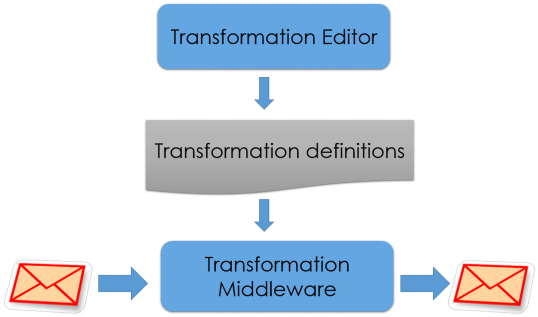
Lukas Hofmaier

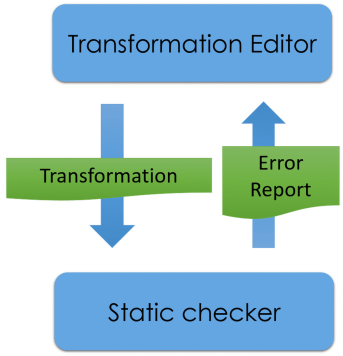
Students	Lukas Hofmaier
Lecturers	Prof. Dr. Farhad D. Mehta
Advisors	Johannes Rieken , Microsoft , Zürich , ZH
Topic	Software and Systems
Project Partners	Incentage AG , Fehraltorf , ZH

Static Checking For A Mapping DSL In The Field Of Financial Messaging

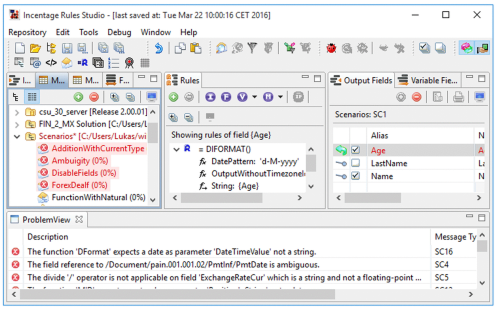
Detecting type errors and ambiguous references in transformation definitions at design-time.



Incentage Middleware Suite translates messages according to transformation definitions.



The static checker finds inconsistencies in a transformation definition which are then reported to Incentage Rules Studio.



Screenshot of the graphical editor Incentage Rules Studio with the integrated static checker.

Problem: In the financial service industry, financial institutions exchange messages electronically amongst themselves. To integrate systems that use different message formats, Incentage AG provides a message transformation middleware named Incentage Middleware Suite. The Incentage Middleware Suite translates messages according to transformation definitions written in the Incentage transformation language. To create and modify transformation definitions in the Incentage transformation language the company provides the graphical editor Incentage Rules Studio. A transformation definition may contain errors in form of inconsistent and ambiguous instructions. Erroneous transformation definitions cause failures or unexpected behavior during the message translation in Incentage Middleware Suite. It would therefore be advantageous to check transformation definitions for possible errors without the need to test them.

Objective: The goal of this thesis was to design and extend Incentage Rules Studio with a new feature that detects errors in a transformation definition while the user is editing it. To detect errors a software component named the static checker finds inconsistencies in transformation definitions. These are then reported to the user. Having immediate feedback both increases the user's productivity as well as the trust and satisfaction of working with the Incentage Rules Studio. The user must additionally not be burdened with providing additional information for the static checker to work.

Result: We implemented a static checker based on a defined set of rules as a separate library. Our static checker implementation detects two specific categories of errors. The first category of errors is found by using a type system. A type system defines the range of values an operator or a function may be applied to. A violation of this constraint constitutes a type error. The second category of errors are ambiguous definitions. Ambiguity can arise in transformation rules that contain a reference to another message field. Due to the possibility of repeating sections within a message, there may be many occurrences of the referenced field. The static checker reports ambiguous field references to the user. Additionally, it supports the user in avoiding ambiguous field references by highlighting them in the corresponding selection dialog.