UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

**CS4590(SJ14), Projektpraktikum SSE**
**CS5198, Projektpraktikum Programmierung** WS 2015/2016

Torben Scheffel, Malte Schmitz
{torben.scheffel, malte.schmitz}@isp.uni-luebeck.de 5. Oktober 2015

# Software Architecture for an Automata Library

## Abstract

LamaConv[1] is a library for representing and translating temporal logics and automata. LamaConv is continuously developed at the ISP and it can handle more and more different automata types. The library is implemented in Scala and the data structures for the logics and the automata as well as the algorithms for their translations are already implemented in a prototypical way. This project is about designing and applying a suitable software architecture based on common design patterns.

## Problem Statement

LamaConv is capable of many different automata types and logics (compare for example [1] for a theoretical introduction). For example it supports alternating Büchi automata (ABA) and alternating parity automata (APA). As these automata only differ in their acceptance condition their data structures in Scala could share many common elements. Operations on these automata—like removing unnecessary states—share a common base, too, but also need to take care of the different acceptance conditions. The current prototypical implementation does not benefit from this shared elements in data structures and algorithms.

A good logic and automata library written in a functional and object oriented language like Scala should follow some principles of software design: For example data structures and operations must not be mixed and should be loose coupled. And yet the implementation should share and reuse common code as much as possible. To achieve these goals common design patterns [2] should be taken into account while designing the architecture.

The goal of this project is to design and implement a software architecture for LamaConv that makes it easier to add new automata types and translations that differ only slightly from the existing ones. Furthermore integrating the library with existing tools like jUnit$^{RV}$ [3] and JavaMOB should be more straight forward.

## Goals

- Get an idea of the logics, automata and translations implemented in LamaConv.
- Design a new software architecture for an automata library like LamaConv.
- Apply the new architecture to the existing Scala implementation of LamaConv.

## Requirements

- You have a good knowledge of software engineering and design patterns, e.g. from the course software construction.
- You are at least interested in formal methods like temporal logics and automata theory.
- You already know Scala or want to learn it.

## Literature

[1] J. E. Hopcroft et al. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 2001.

[2] E. Gamma et al. *Design Patterns. Elements of Reusable Object-Oriented Software.* Prentice Hall, 1994.

[3] N. Decker, M. Leucker, D. Thoma. *jUnit$^{RV}$—Adding Runtime Verification to jUnit.* NFM, 459–464, LNCS, 2013.

[1] `http://www.isp.uni-luebeck.de/lamaconv`