**Course Syllabus**

**Course Name:** Construction Programming - by Correct

**Course Code:** 5941-1-202

**Type:** Course Selection

**Credit:** 4.0

**Instructor:** Dr. Ren Atininger

**Prerequisites:**
- 0201-1-202: Introduction to Logic and Set Theory
- 202-1-2051: Programming Languages
- 202-1-2054: Principles of Programming

**Syllabus**

One of the significant challenges faced by software developers is the need to find ways to define algorithms without errors and produce clean programs.

The goal is to achieve full functionality, but achieving it is crucial, especially in systems where inadequate performance can lead to human life expectancy, cause significant environmental damage, or result in severe business failure.

When the software system is developed correctly, testing and verification are used to ensure the correctness of the program. However, tests are not guaranteed to reveal all errors due to practical difficulties (and sometimes lack of ability).

A third approach to verify the correctness of the software code is to build the code in a disciplined manner, directly from the specification (specification), through a series of refinement steps (refinement). Invariant

The algorithms developed in this course are very short, typically, as the development process is not trivial.

The course is based on Carol Morgan's book "Specifications" and takes place using the Dafny programming language, integrated into Visual Studio. This environment allows a combined statement of functional specification and imperative code for implementation.

Throughout the software development process, there is also an automated verification mechanism for the correctness of the program, allowing the execution of programs only after they are proven. (The execution is possible by translating the code from Dafny to C#.)
A major challenge software developers are facing is how to design correct algorithms and develop computer programs without bugs. This is certainly not a trivial goal, and finding ways to achieving it is highly important, especially in the development of life-critical, safety-critical, or mission-critical systems.

When focusing on a program’s functional correctness (such that for any valid input the program generates the expected output), one could resort to common practices such as testing or verification. However, successful testing does not guarantee the absence of errors, whereas verification is difficult to perform. A third option is to construct the proof of correctness alongside the development of code.

This is an advanced course teaching how to design algorithms and programs that are guaranteed to meet their specification. Starting with a mathematical description of the program’s requirements, the course presents a formal method for turning such specifications into actual code, in a stepwise approach known as refinement. Techniques of algorithm refinement are presented, for the derivation of loops and recursive procedures from invariants.

The developed algorithms are typically very short, but challenging, as we aim to construct correct and efficient code. The taught material is mainly based on the textbook "Programming from Specifications" by Carroll Morgan. The programming throughout this course is done in the language Dafny, using its integration into Microsoft Visual Studio. This environment enables the annotation of programs with their specifications. Moreover, it includes an automatic verifier, such that a program can be executed only after its functional correctness has been established. (The execution is facilitated by a translation of the Dafny code into C#.)

At the end of the course students are expected to be able to construct correct programs. More concretely, you will be able to:

- Specify program requirements abstractly.
- Perform rigorous and formal derivations of efficient programs from their abstract specifications.
- Understand the criteria for algorithm refinement.

The following topics will be covered, along with a range of examples and case studies:

- Program specification using predicates and assertions: predicate notation, preconditions and postconditions, specification statements.
- The language of guarded commands (the assignment statement, sequential composition and conditional statements, blocks, local variables, and arrays) with proof rules for each program construct and the corresponding syntax in the programming language Dafny (see https://github.com/Microsoft/dafny, http://rise4fun.com/Dafny).
- Basic techniques for finding invariants.
- Constructed types: from sets, bags, and lists to functions and relations.
- Procedures and parameter passing.
- Recursive procedures: rigorous derivation of sorting and search algorithms.
- Recursive types: linked lists and binary trees.
Students taking this course will be required to have good programming skills, yet they will not be expected to possess the logical-reasoning skills needed for specifying the algorithms, as those will be acquired throughout the course.

The final grade will be determined by one homework assignment (20%), a must-pass midterm examination (20%), and a final project (60%). The assignment and project are to be prepared in (small) teams.

The main textbook of this course is:

Carroll Morgan. *Programming from Specifications (2nd edition)*

Additional textbooks:

Anne Kaldewaij. *Programming: The Derivation of Algorithms*
Roland Backhouse. *Program Construction: Calculating Implementations from Specifications*
Edsger W. Dijkstra. *A Discipline of Programming*