# Engineering Software Correctness

**Faculty Member:** Rex Page

## Objectives
- Software components with guaranteed properties
- Education in practices that can lead to software guarantees

## Background
- Some software needs to be free of certain defects
  
  You don’t want your car to speed up when you press brake
  
  You don’t want your ailerons stuck in “roll”
  
  You don’t want your word processor to lose your edits
- Conventional software cannot have certified properties
  
  Why? Because classical logic fails on conventional software
  
  No machine-checked verification beyond driver-level SW
- What is needed?
  
  Mechanized logic, for checking details down to the bit level
- Tools exist ... We want to put them into practice

## How to Design a Reliable Software Component
- Imagine that you already have the component
- Express basic properties as equations (“axioms”)
- Make sure axioms cover all possible cases
- Make sure circular axioms move toward non-circular ones
  
  **It’s as simple as that!**
- All other properties derive from the basic axioms
- Computation based on substituting equals for equals
- Verification relies only on classical logic
- Machines check all details

**Engineers can know behavior of designs**

## Example and References
- Simple properties of a multiplexor, expressed as equations
  
  \[ x_1 : [x_2, x_3, \ldots x_n] = [x_1, x_2 \ldots x_n] \]
  
  \[ \text{mux} ([] []) = [] \]
  
  \[ \text{mux} (x : xs) (y : ys) = x : y : (\text{mux} xs ys) \]
- Derived property
  
  \[ \text{mux} [x_1, x_2, \ldots x_n] [y_1, y_2, \ldots y_n] = [x_1, y_1, x_2, y_2 \ldots x_n, y_n] \]
- Derivation: proof by mathematical induction
  
  Fully verified by machine
- Methods scale to large components
- References
  
  
  Discrete Math with a Computer 2nd Ed, O’Donnell/Hall/Page (Springer 2006)
  
  DoubleCheck Your Theorems, Carl Eastlund, ACL2 2009 (Boston)