# EE382C: Verification and Validation
# Project Update
# Spring 2014

**Name of Team Members**: David Liu, Olamide Kolawole
(Note: Albert Bustamante has dropped the class)

**Project Title:** Modeling of Dual-Clutch and Semi-Automatic Transmission Logic in Alloy

**Project Updates:** After the project proposal, a base repository for organization was done. It was created on Github, and is available here:

**https://github.com/triskadecaepyon/DCT_Alloy_Model**

We separated out the tasks for the project into the following sub-tasks:

- Create Transmission Model (Tasked to: David)
- Solve for Predicates and configurations (Tasked to: David)
- Pipe output to graphical display or compile ready model (Tasked to: Olamide)

Here is each of the tasks broken out into details:

***Create Transmission Model***

- Car is a type of vehicle
- Car is driven by a transmission, powered by one or more engines (i.e. if you have a hybrid, or a Tesla S it might be different)
- Transmission has X gears
- 1st, 2nd, 3rd gear are types of gears
- Gear ratio to vehicle speed relations
- Shift schedules (i.e. when your car decides at what speed to shift up/down)
- Transmission inputs: Gas pedal, brakes, vehicle speed, shift up/down command, engine RPM
    - Each of these can be modeled as a relation of atoms: i.e. 0%, 50%, 100% of gas pedal is a type of gas pedal input.
    - Shift input is a type, and has child relations of up/down
    - Engine RPM can be modeled as : Low threshold, power band, red-line.
- Outputs: Gear selection, Clutches (i.e. clutch/de-clutch), Clutch line pressure, layshaft speed

***Solve for Predicates and configurations***

- Toyota's "unintended acceleration" - i.e. if brakes are applied at 100%, regardless of gas, declutch the system.

    - if Both brakes and throttle are put down, de-clutch
- Make sure the system can't shift into the wrong gear, aka shifting such that the gear ratio is now putting the engine at redline.
- Ensure the engine remains in the powerband, or close to it if throttle is set past a given value (i.e. > 50%)
- Knowing when to ignore inputs, like shifting up when no gear exists, or when the system cannot physically do so.
- Making sure the model holds up for different configurations: i.e. 6 or 7 speed gearbox, single or dual clutch systems.

***Pipe output to graphical display or compile ready model***

Alloy's output is non-standard for most users in the Automotive industry, or even those looking to understand the system from say, a stakeholder perspective.

The goal for this task is to take the XML output of an Alloy model, and display it properly for the specific domain (i.e. automotive ). This task can be accomplished by looking for display software, frameworks, or other projects to supplement the Alloy output. Since there is many outputs that process XML into graphical formats for UML, there should be at least a few options.

**Updates on Tasks**

The first task of creating a transmission model is proving to be the most difficult. As said before, the abstraction, concepts, and syntax require a very strong background in the Alloy language, which is quite hard to do in the time allotted for a semester. That being said, some progress in the relations and atoms have been created that seem to work.

Examples of the car relations can be found on the full file on Github, but:

```
open base_transmission_model

abstract sig Vehicle {}

sig Car extends Vehicle{
     drives: one Transmission,
     powered: one Engine
}

sig Engine {}

pred show (g: Gear) {
  SingleCar[]
}

pred SingleCar() {
     Car.drives = Transmission
```

```
     Car.powered = Engine
     one Car
}

run show
```

Note that we are importing an alloy model, and representing the initial relation a transmission has with an engine. The engine itself still has to be modeled, but the base transmission relation is nearly complete. Examples of some of the predicates are below:

```
pred restrictedGearBox (t: Transmission) {
     //Says that the atoms of shiftschedules and shiftpoints
reside in the Transmission
     // and cannot be "loose" atoms in the world.
     all s: ShiftSchedule | s in t.restricted
     all p: Shiftpoints | p in t.restricted.threshold
     all g: Gear | g in t.gears
}

pred noSameGear {
     all g: Gear |  g.nextGear !in g.previousGear
     //Says that no nextgear can be the same as its previous
gear.
     //This could be a problem, as it can restrict things such as
first gear.
}

fact connected {
     all g: Gear | Gear in g.^nextGear
     all g: Gear | Gear in g.^previousGear
     //Every Gear has a previous and nextgear, and is reachable
through
     //those relations somehow.
}
```

Interestingly enough, writing the relation of a transmission in alloy has also shown that the concept of *order* needs to be in it, so the alloy file will need to be re-tooled with the module util/ordering soon.

In the creation of the transmission and car classification files, came the representations of the predicate solving task.

Examples:

- `Check for 3 upshift but 2 downshift`
    - Looks to see if it lands on correct gear based on circumstances

- `Check for 8 Gear`
    - Different gearbox configurations to see minimum relations needed
- `run show restrictedBySevenRelations`
    - where it has 10 gears and 7 shift point restrictions

For many of these predicates, the order relation must be working, or unusual instances are found.

***Piping Alloy output to graphical display***

Alloy's visualization module is not the best solution when it comes to displaying the relational output and instances of the transmission model. Positioning and organization of the relations is key when trying to understand the generated solutions.

We have implemented another visualization tool called “yED"(http://www.yworks.com/). It takes an exported output (in XML) from Alloy and transposes it to to be usable in the display software. We then take an XML import in conjunction with an XSLT stylesheet. The style sheet is required to display the graph, as yED is not aware of the Alloy XML formatting.

yED provides a library that we use to tap into advanced diagramming toolset in a Java program. The output of our program is GraphML(http://graphml.graphdrawing.org/). This means that we can potentially use other diagramming tools, not just yED. More work will need to be done to visualize inaccuracies in our model, and improved visual cues and categorization of relationships will be at the forefront.

Currently, we do not have a specification for the XML export from Alloy. We can guarantee that the display output will work for our model, but unsure of other models outside our current project scope.

(Example image below)

*Test Alloy output from yED of current model:*