

# Visualizing and Documenting SystemVerilog Class Based Testbenches with UML in High Reliability Space Systems

## Authors:

Lucas Roosevelt (Honeywell, ASIC/FPGA Design, [Lucas.Roosevelt@honeywell.com](mailto:Lucas.Roosevelt@honeywell.com))

Brett Oliver (Honeywell, ASIC/FPGA Design, [Brett.Oliver@honeywell.com](mailto:Brett.Oliver@honeywell.com))

John Profumo (Honeywell, ASIC/FPGA Design, [John.Profumo@honeywell.com](mailto:John.Profumo@honeywell.com))

Peter LaFauci (Mentor Graphics, Application Engineer, [Pete\\_LaFauci@mentor.com](mailto:Pete_LaFauci@mentor.com))

## Presenters:

Lucas Roosevelt

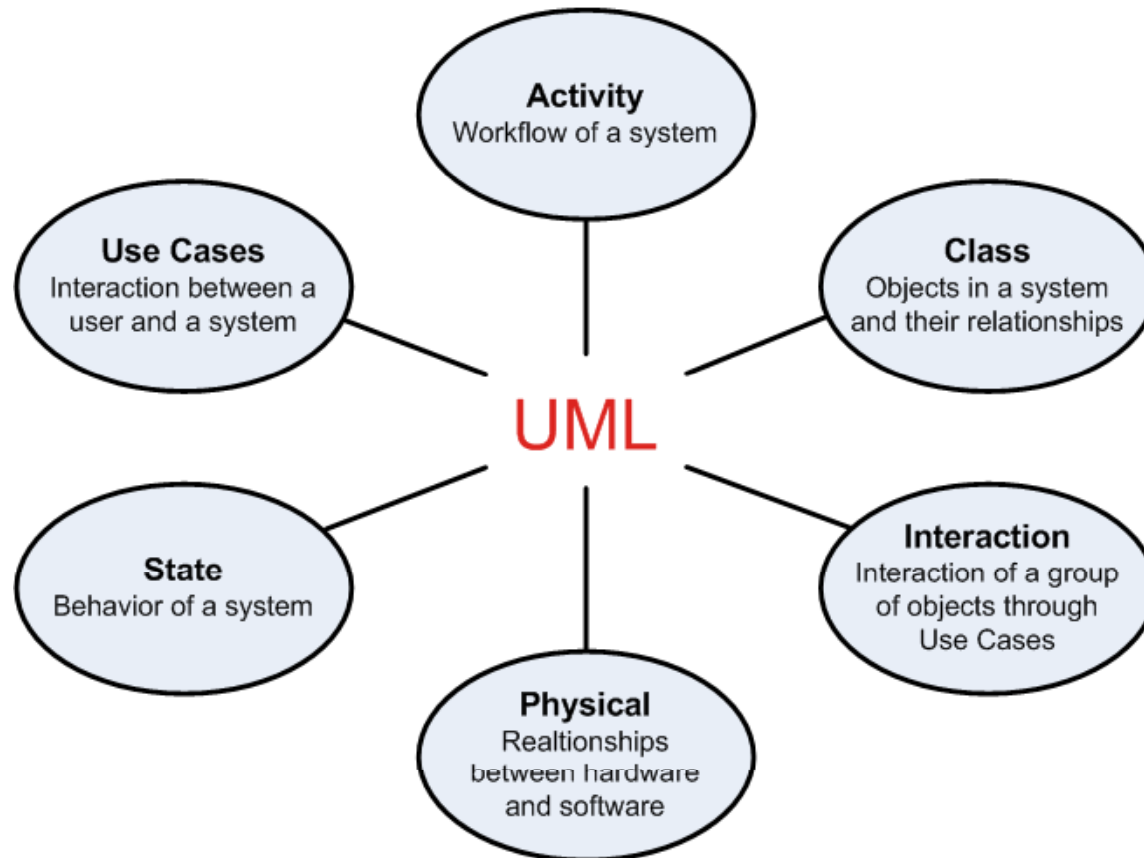
Pete LaFauci



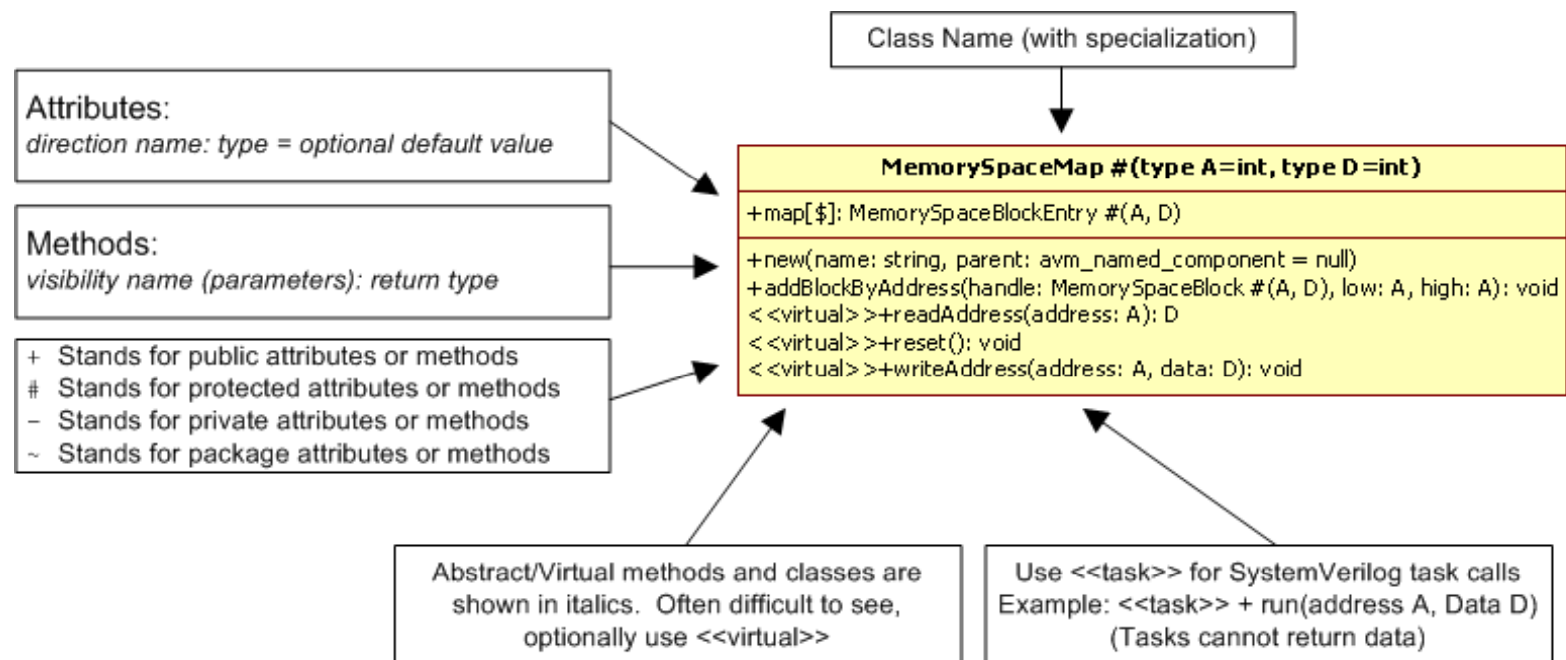
- The “**Unified Modeling Language**” is a standardized general purpose modeling language
  - Describes software systems using visual models
  - Standard was created and is maintained by the Object Management Group
- UML’s modeling can be used throughout any part of the development cycle of an Object-Oriented system
- Space applications require the assurance of conformance to requirements

- **SystemVerilog testbenches are Object Oriented “Systems”**
  - **Require a new approach to documentation**
  - **Block diagrams do not accurately portray Object Oriented Designs and their complex relationships**
  - **Verification of hardware designs is requiring more and more of software engineering techniques to get the job done**
  - **Documentation of SystemVerilog testbenches is where the software industry was 20 years ago**

- **UML is comprised of 14 “Diagrams”**
  - Each diagram type provides a different perspective and degree of abstraction



- UML Class Diagrams describe the types of objects in a system and their relationships to other objects
- The focal point in a Class Diagram is the representation of a class or object



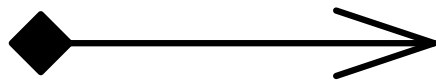
- The relationship of classes with other classes is depicted through the use of specific lines and arrows
  - Lines represent relationships and are called a “*Link*”
  - Arrows indicate the direction of knowledge
  - Different relationships are depicted by different types of lines and arrows



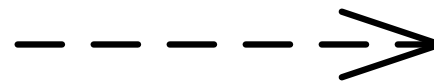
**Association:** Most basic relationship between two classes.



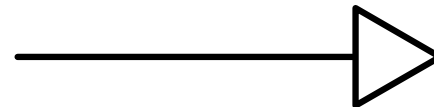
**Aggregation:** Object(s) created at run time. Not destroyed with parent. Typically a component registered as an Observer.



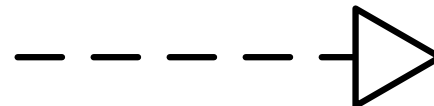
**Composition:** Object(s) created at compile time. Destroyed with parent. Typically a component created by the test environment.



**Dependency:** A class must know specifics about another class but does not create it or typically have a direct handle to it.

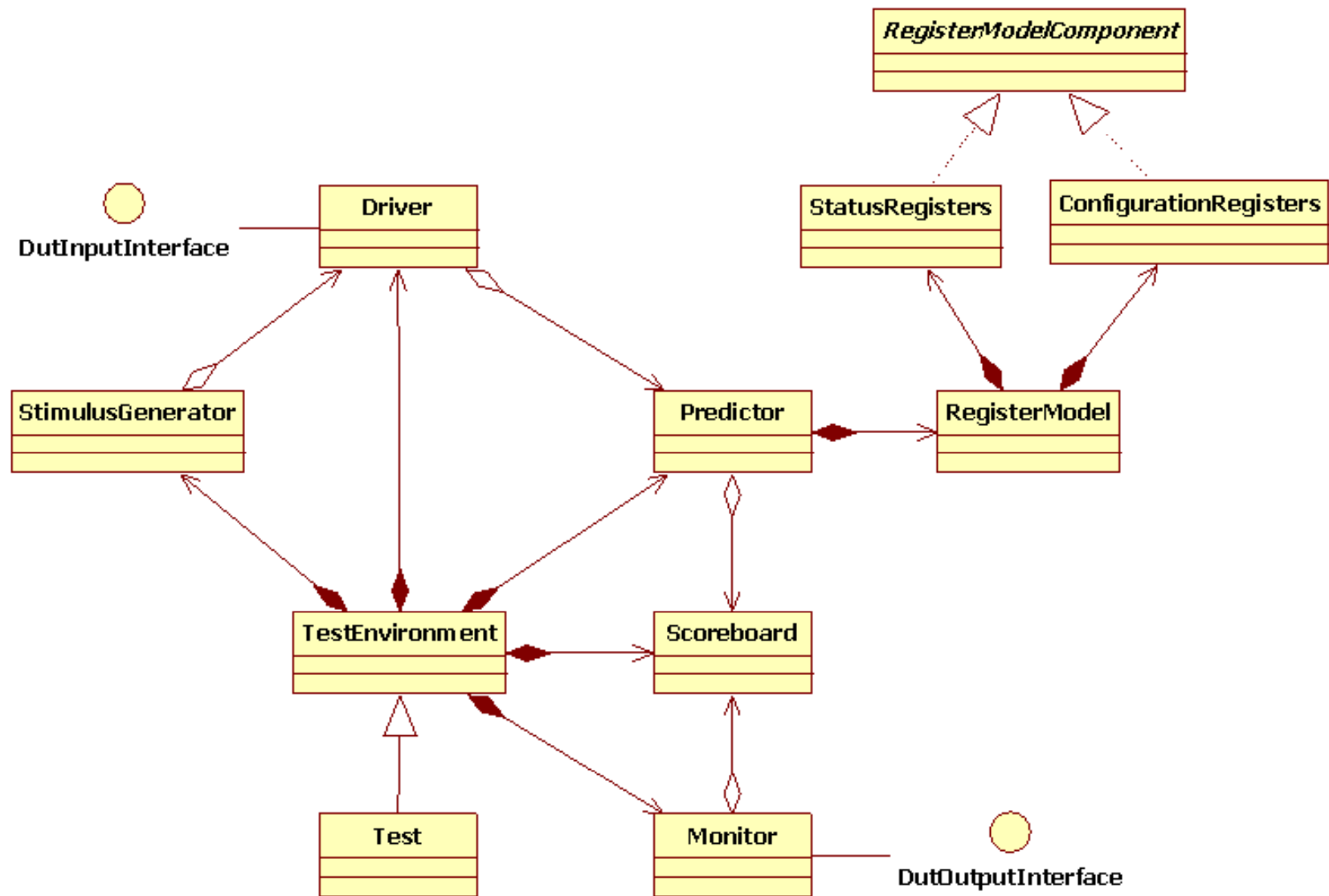


**Generalization:** Adding to or replacing functionality of a parent class.

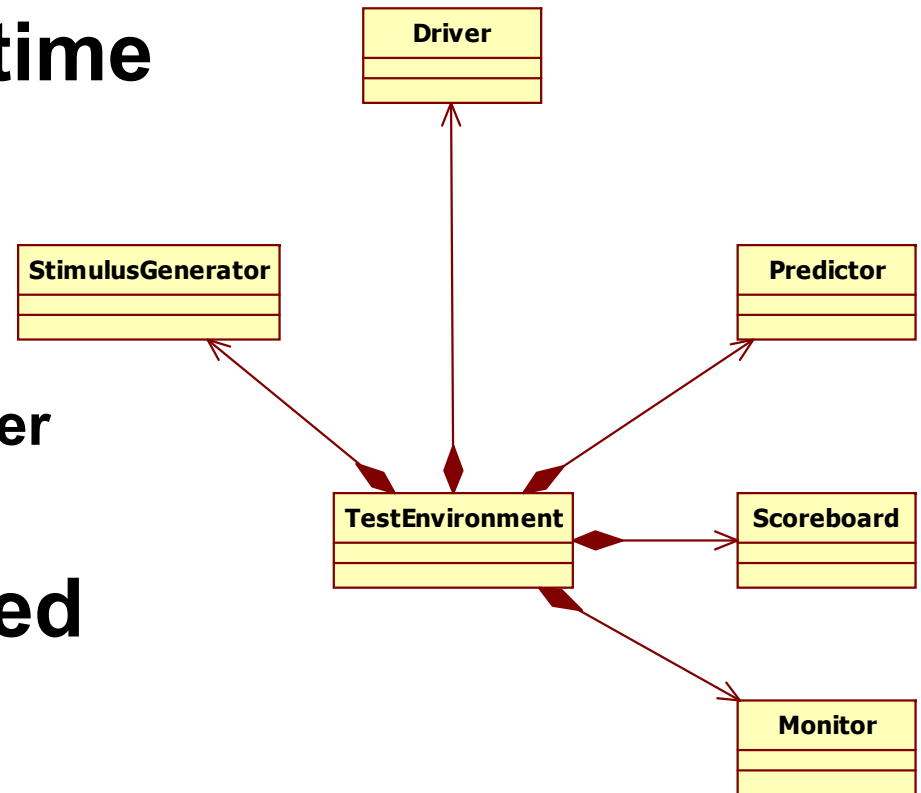


**Realization:** Implementing functionality of a class that is described as being abstract.

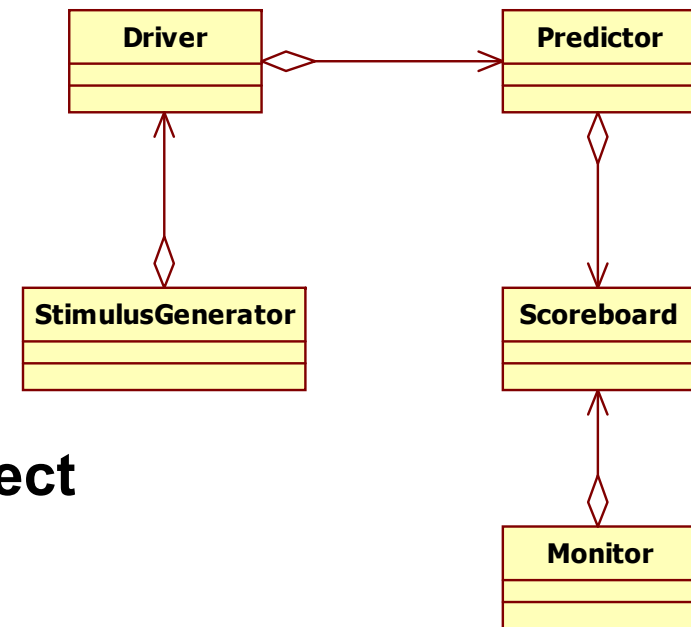
# UML of an Object Oriented Test Environment **Honeywell**



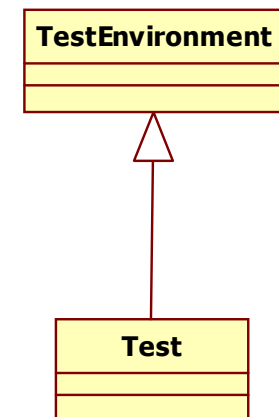
- **Test Environment**  
“creates” the static transactors at compile time
  - Stimulus Generators, Drivers, Scoreboards, Monitors and Predictors
  - “Owns a” relationship
  - Test Environment Owns a Driver
- **Destroyed if the test environment is destroyed**
- **Exist throughout the simulation**



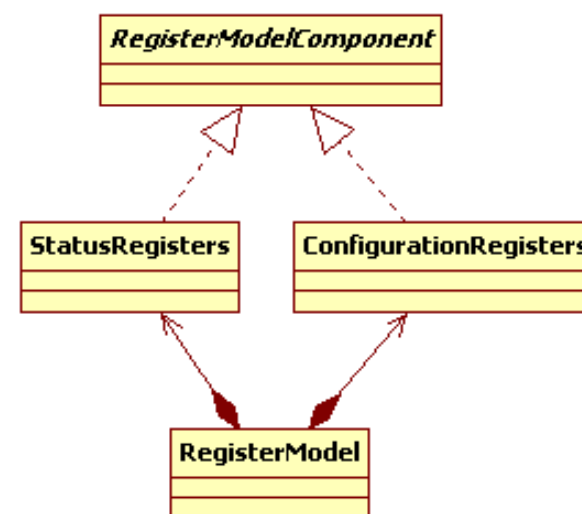
- **Test Environment “connects” the components at run time**
  - Stimulus Generator to Driver
  - Driver to Predictor
  - Predictor to Scoreboard
  - Monitor to Scoreboard
  - “has a” relationship
  - Driver has a Predictor
- **Connections are the result of the Object Oriented “Observer” pattern**
  - Implemented through handle passing or AVM/OVM Analysis ports
- **Objects being pointed at do not get destroyed when the pointer does**
- **Allows for creation of and “connecting” to different components from the test**
  - Bad Drivers, Slaves



- **Test “Extends” the test environment with specifics of that test**
  - Can include added constraints to focus on some specific target area
  - Can include other classes for extending data classes which themselves could also include specific constraints
  - Can include new or “Bad Drivers” which get are swapped with the good drivers to produce different stimulus
  - “is a” relationship
  - Test is a Test Environment



- **Realization or Virtual Classes** define functions that the child classes must implement
  - Allows better object passing between components and the ability to tailor functionality based on that component's specifics
  - Often used to define data transactions so that the drivers, monitors, scoreboards and predictor only need to know the agreed upon function calls.



- **DO-254 and Space Applications**
  - Requires a rigorous, structured verification process
  - Requires that all requirements be formally defined and tested
  - Requires design engineers to PROVE their implementation meets those requirements
- **UML is the gold standard of accurately describing a SystemVerilog Object Oriented testbench environment**
  - Can be time consuming if created manually
  - Mentor Graphics Certe Testbench Studio can automate the UML creation
  - Automating the process removes human error and provides up to the minute diagrams

