



G

E

N

T

E

S

T

Silu Shen Tristan Miller Zane Fink Joshua Johnson

# Why is Unit Testing Important?

- Only 16.2% of all software **projects completed on-time and on-budget** in 2014 [1]
- Delayed and Canceled software projects cost companies a combined total of **\$152 billion** [1]
- Companies need better ways to ensure project success
- Unit testing can reduce software defects from **20.9%** to **91%** in software projects. [2]

Unit testing can help **companies save time and money.**

# Client Introduction

---



Associate  
Professor Alex  
Groce

- DeepState is a unit testing framework created Alex Groce and Trail of Bits.
- It provides developers easier access to tools such a fuzzing and symbolic execution.
- While manual testing techniques are viable, DeepState provides a more comprehensive way to identify errors code.

# The Current DeepState Workflow

DeepState.hpp

```
#define TEST(...  
#define TEST_F(...  
#define ASSUME(...  
#define EXPECT(...  
#define ASSERT(...  
#define LOG(...
```



Tests.cpp

```
#include <DeepState  
  
TEST(Unit, Name1) {  
    symbolic_int x;  
    ASSUME(x > 0);  
    ...  
}  
  
TEST(Unit,
```


libdeepstate.a

```
DeepState_Assume:  
    ...  
DeepState_Pass:  
    ...  
DeepState_SoftFail:  
    ...  
DeepState_
```




Tests.o

```
Unit_Name1_Test:  
    call  
DeepState_Int  
    cmp eax, 0  
    jg .L0  
    call DeepState_..  
.L0:  
    ...
```



MANTICORE



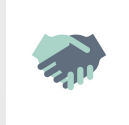
DynamRIO  
The DR. is in.

Tests

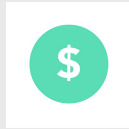
```
0101010  
1010101  
0010101  
0110101  
0100101  
0101010  
1010101  
0101010  
1010101  
0101010
```

# The Problem

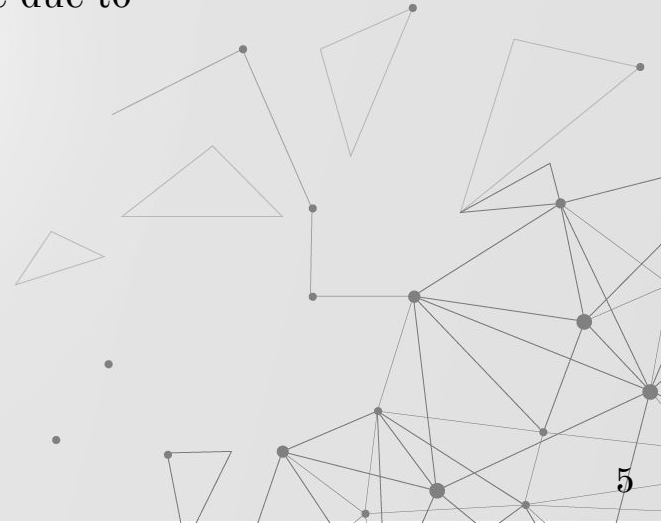
---



It can be hard to integrate with software projects because of pre-established frameworks.



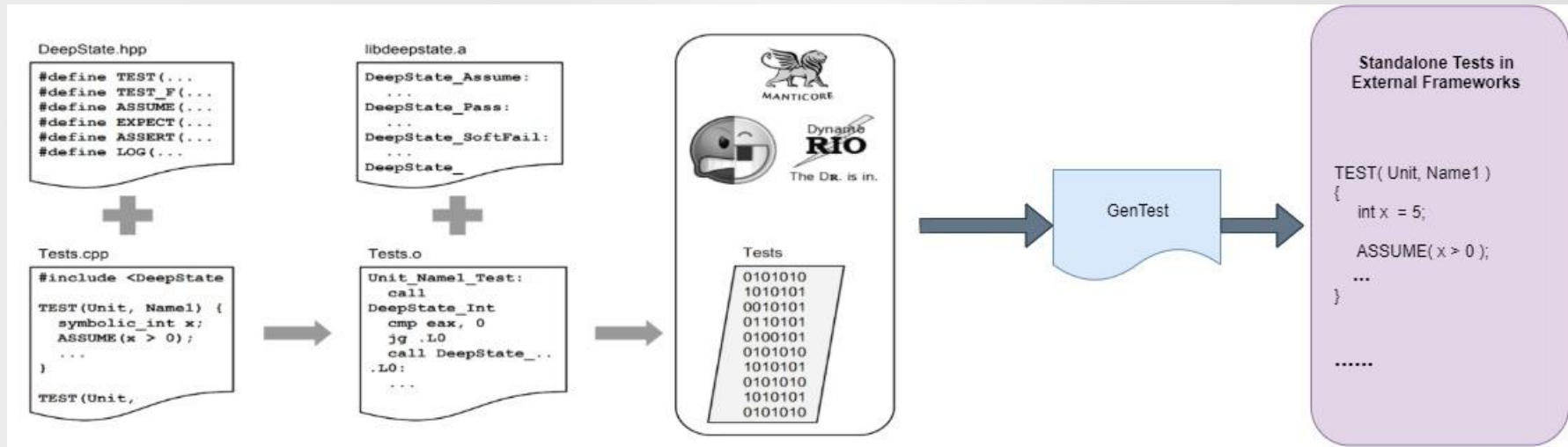
Projects who may want to use the tools provided by DeepState are unable due to migration costs.



# Solution Overview

The solution envisioned by team GenTest involves:

- Creating a module which will translate DeepState tests to other frameworks.
- Still provide support for auto-generated test cases.
- Requires no DeepState framework functions after translation.



# Requirements Acquisition Process

---

- ➔ **Bi-weekly Interviews with client**
- ➔ **Reading DeepState source code**
- ➔ **Working through DeepState tutorials**



# Key Requirements

---

- **Extension to DeepState functionality that enables the creation of standalone unit tests.**
- **Support for translation to user-defined target testing framework (Google Test, Catch2, etc.)**





# Top Level Requirements

---

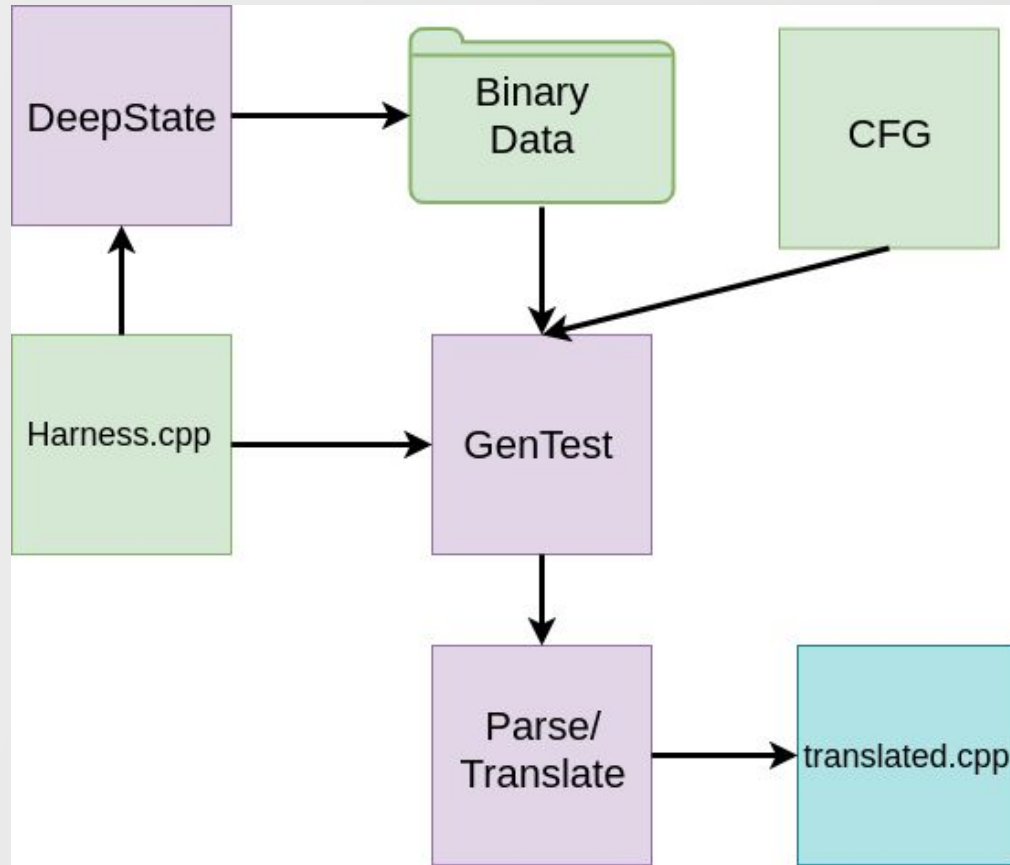
- **Functional Requirements**
  - **Translation from DeepState testing framework syntax into user-specified testing syntax**
  - **Insertion of values generated by DeepState into standalone test files.**
  - **Support for nested structures and looping constructs**

# Top Level Requirements (Cont.)

---

- **Performance**
  - **Simple command-line interface with no more than 5 required arguments**
  - **DeepState users can produce standalone tests within 3 minutes.**
  - **Translated tests must be semantically equivalent to those in the test harness**
- **Environmental**
  - **No libraries outside of C++ STL can be used.**
  - **Generated code is in C/C++**
  - **Cross-Platform support for Linux and MacOS**

# A Breakdown of the Test Generation Requirement



# Risks and Feasibility

## Risk Assessment of GenTest

Risk	Probability	Severity
Failure To Compile	Orange	Red
Semantic Inequivalence	Green	Red
Unsupported Data Types and Structs	Green	Green

# Risks and Feasibility (Cont.)

---

- Failure To Compile
  - Standalone tests are generated by GenTest
  - Utilizes a Context-Free Grammar (CFG)
  - Probability of failure of compilation in external framework
  - **Mitigation:** Use of logging will allow developers to debug the GenTest process faster and easier



# Risks and Feasibility (Cont.)

---

- Semantic Inequivalence Between DeepState Tests and Standalone Test Output
  - Abnormal behavior of generation and assembly process
  - **Mitigation:** Using a variety of test files comparing DeepState and standalone test outputs

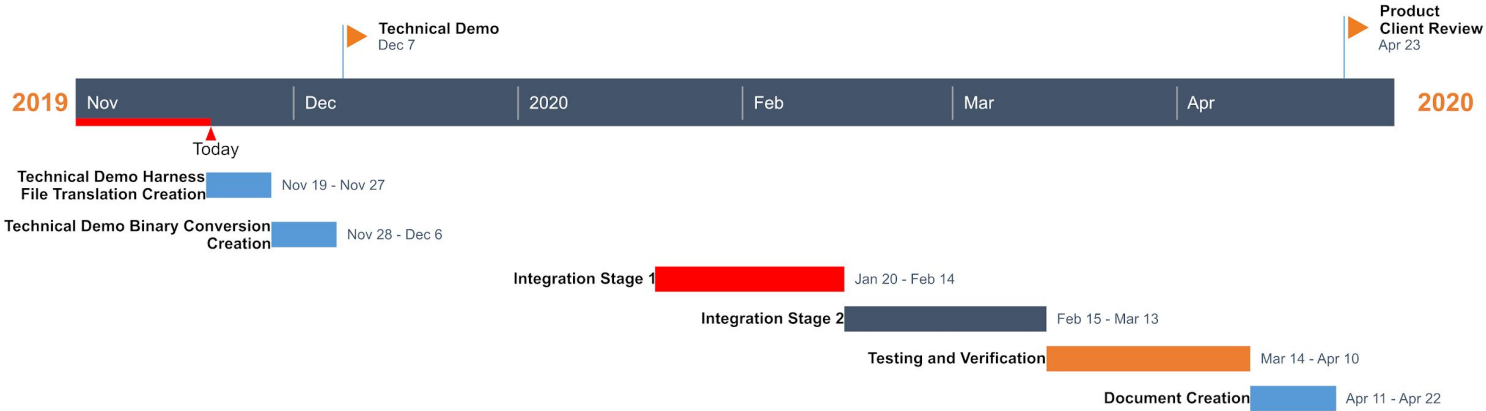


# Risks and Feasibility (Cont.)

- Incomplete or Sparse Support for Data Types and Structs
  - Currently plan to support
    - The listed primitive data types, which DeepState can support
    - Basic Structs
  - Stretch Goals:
    - Nested and Recursive Structs
    - Pointers
  - **Mitigation:** Explicitly detailing supported features in documentation

Char, String  
Short  
Int  
Unsigned  
Long  
Int8\_t  
UInt8\_t  
Int16\_t  
UInt16\_t  
Int32\_t  
UInt32\_t  
Int64\_t  
UInt64\_t  
Double, Floats

## Project Plan Overview

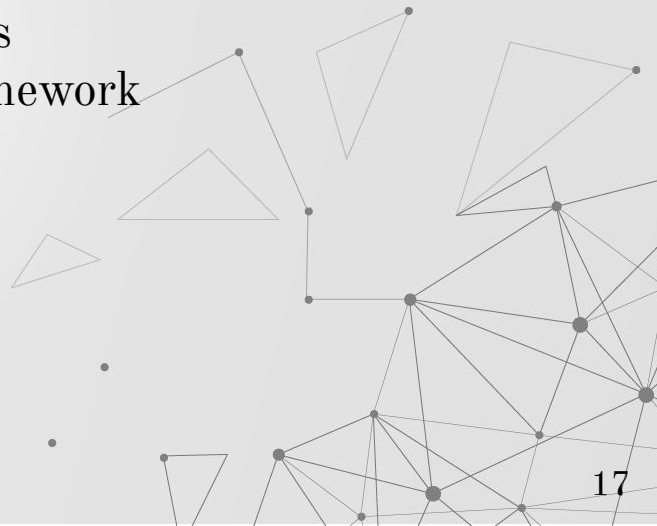




# Conclusion

---

- The Problem
  - DeepState-generated tests are highly coupled with DeepState itself.
  - Need a way to increase portability of tests.
- Our Solution
  - Extension to DeepState
    - Creation of standalone, self-contained tests
    - Translation into user-specified testing framework syntax
- Our Plan Moving Forward
  - Phase 1: Translation of simple tests



# Conclusion

---

- Thank you
- Questions?

