https://drive.google.com/file/d/1kXdq-cBlybVMBoqo1cDoGA8OZsxd5KG1/view?usp=sharing

# ARCHITECTURAL ENHANCEMENT

## For ScummVM and SCI Engine

Trevor White (**Group Leader**, Alternatives for Implementation, Use Cases)

Sophia Pagazani (**Presenter**, SEI SAAM analysis)

Aniss Hamouda (**Presenter**, Use Cases, Effect on NFRs)

Claire Whelan (Abstract, Introduction, Potential Risks, Testing Plan)
Nicole Hernandez (Alternatives for Implementation, Lessons Learned)
Nasreen Mir (Current State, Enhancement, Motivation, Value, and Benefit)

# TABLE OF CONTENTS

## 01

### Enhancement Intro

Current state of the system, and our proposed enhancement

## 02

### Implementations

Our two considerations on how to implement our enhancement

## 03

### SEI SAAM Analysis

Our enhancement and its considerations on stakeholders and NFRs

## 04

### Use Cases

A description of how the system would act in specific scenarios

## 05

### Continued Analysis

Testing plans, risks and effect of certain NFRs will be discussed

## 06

### Conclusions

Wrapping everything up, and our lessons learned

# ENHANCEMENT INTRODUCTION

# Current State

**ScummVM** is an open-source software that runs classic point-and-click games across a variety of platforms

- Has a layered style
- Recreates other game engines to run in it
- Maintained by the community
- Includes features such as
  - Save states
  - Remapping controls
  - Fast-forward functionality

Does not have an explicit speedrunning tool…

# Enhancement

Proposed Idea: A **Speedrunning** toolset

- Track best time
- Rewind to previous checkpoints
- A timer that can be started manually or automatically
- Tracks current progress
- Has a practice mode
- Have its own UI

# Motivation, Value, and Benefit

Why?

- Speedrunning has become a popular and competitive activity
- Encourage new categories + games to be competed in
- Proven market for speedrunning already exists

Value?

- Empower current players by making speedrunning accessible
- Enable new ways to interact with old games, creating community
- Provides tools for players who want to master games
- Non-intrusive for casual players

# IMPLEMENTATION 1: ENGINE-INDEPENDENT

# SUMMARY OF ALTERNATIVE 1

**ScummVM Layer:** Most components are added in a new layer of ScummVM, making it work without the SCI engine

**Idea:**  To implement the speedrun functionality in ScummVM, making it easy to port into other engines.

**Changed Components:** Interpreter, User Interface

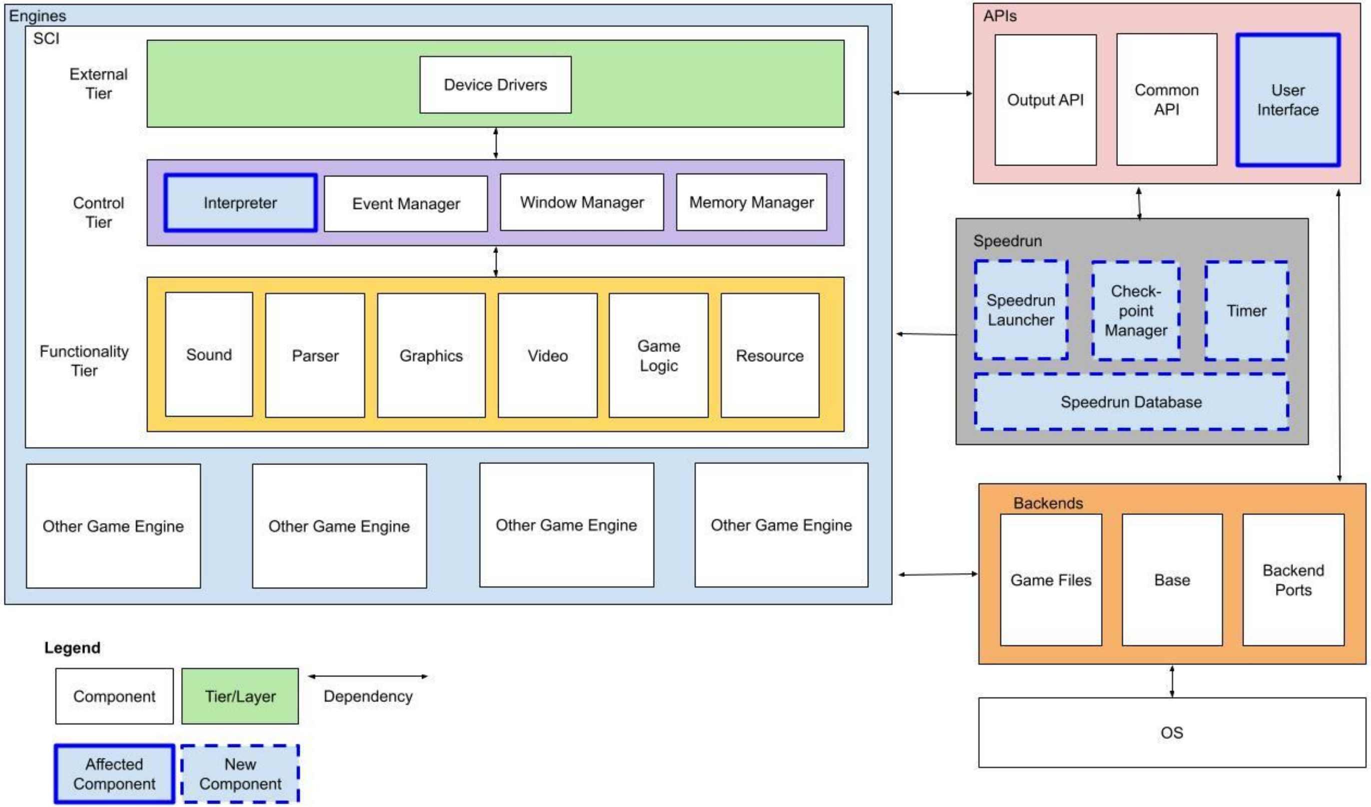
**New Components:** Checkpoint Manager, Speedrun Launcher, Timer, Speedrun Database

# SUMMARY OF ALTERNATIVE 1

**Impact on Architecture:**

- **New layer** added to ScummVM
  - This new layer works in conjunction with the other layers

- This **minimizes** the **dependence** on the engine
- Doing everything in a new layer **minimizes impact** on other layers



# Engine-Independent

# IMPLEMENTATION 2: ENGINE-INTEGRATED

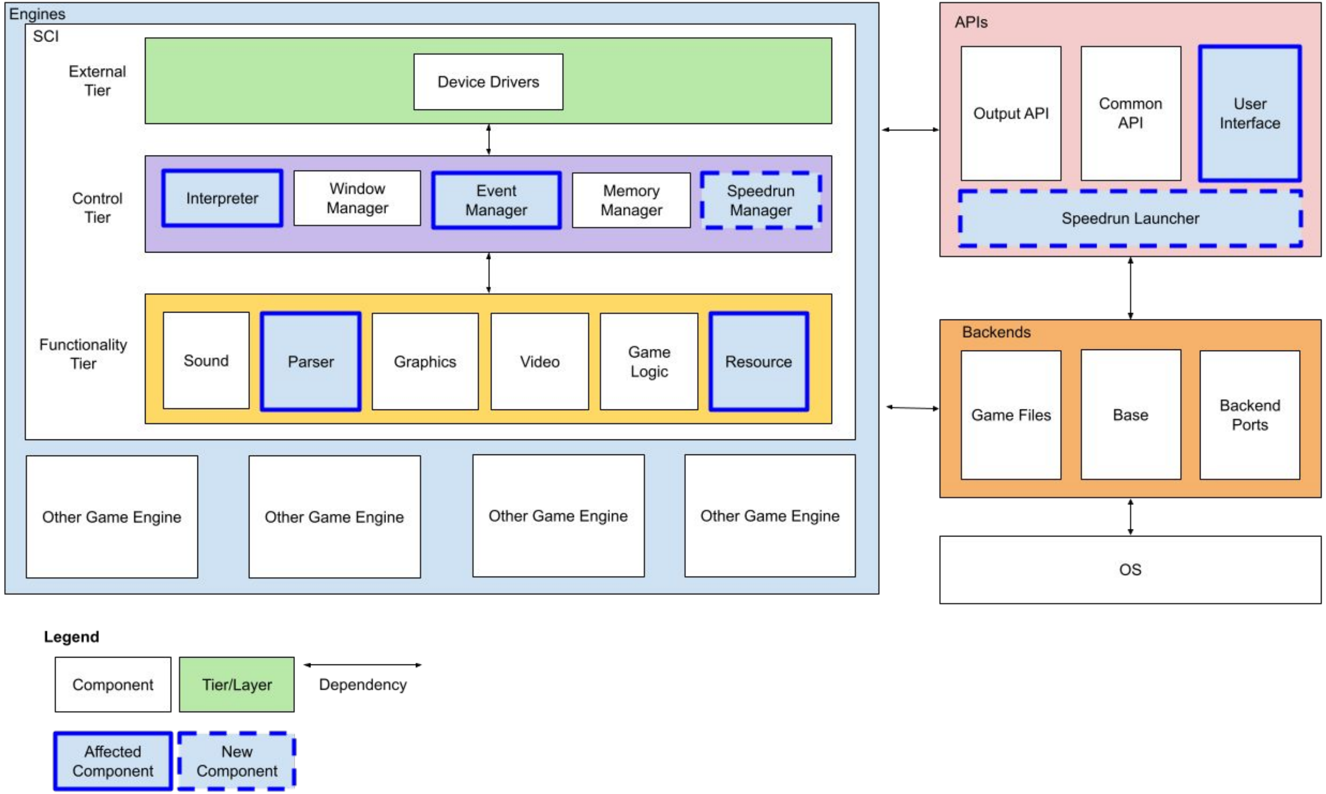
# SUMMARY OF ALTERNATIVE 2

**Engine-Integrated:** Most components are changed or added within the SCI system/architecture, making it highly dependent on it.

**Idea:** To leverage existing components that perform similar tasks to extend functionality for our enhancement

**Changed Components:** Interpreter, Event Manager, Parser, Resource, User Interface

**New Components:** Speedrun Manager, Speedrun Launcher



# Engine-Integrated

# SEI SAAM ANALYSIS

# IDENTIFIED STAKEHOLDERS

**Speedrunners:**

- Players that attempt to complete games with the fastest time within a certain category with specific objectives
- Professional OR just for fun

**Casual Players:**

- Players have little interest in using the speedrunning features but wish to ensure that their gameplay experience is not changed

**Contributors**

- ...to the development of ScummVM and the SCI engine
- new avenue for them to contribute, by creating speedrunning configurations for game engines supported by ScummVM

# ASSOCIATED NFRs

**Speedrunners**

- Accuracy
- Usability
- Performance
- Scalability

**Casual Players**

- Performance
- Usability
- Portability

**Contributors**

- Modifiability

# IMPACTS ON NFRs (1)

| STAKEHOLDER | ENGINE-INDEPENDENT |
|---|---|
| Speedrunner | **Accuracy ✔** - timer is independent and flexible |
| | **Usability ~~** - consistent UI, but inconsistent measures |
| | **Performance ~~** - lots of communication outside SCI |
| | **Scalability ~~** - same as performance |
| Casual Player | **Performance ✔** - remains as it did before enhancement |
| | **Portability ✔** - same as performance |
| | **Usability ✔** - easy to disable before launching a game |
| Contributor | **Modifiability ~~** - open to change because in ScummVM |

# IMPACTS ON NFRs (2)

| STAKEHOLDER | ENGINE-INTEGRATED |
|---|---|
| Speedrunner | **Accuracy** ~~ - relies on internal timer, fast response time |
| | **Usability** ✗ - have to launch engine to view tools |
| | **Performance** ~~ - slow response times, quick reload |
| | **Scalability** ✗ - increased load on the interpreter |
| Casual Player | **Performance** ✗ - memory space held when not in use |
| | **Portability** ✔ - same as before enhancement |
| | **Usability** ~~ - variable for what engines a player is using |
| Contributor | **Modifiability** ✗ - hard to replicate for other engines |

# FINAL IMPLEMENTATION METHOD

We are going with a **Engine-Independent** method for implementation because it is:

- Easier to adapt
- Net positive overall
- Performance is maintained and other NFRs are improved
- Aligns wholly with stakeholders interests

# USE CASES

# RELOADING CHECKPOINT STATE

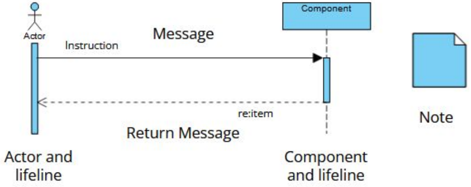
When the player selects to load a checkpoint, the User Interface will pass this to the Checkpoint Manager

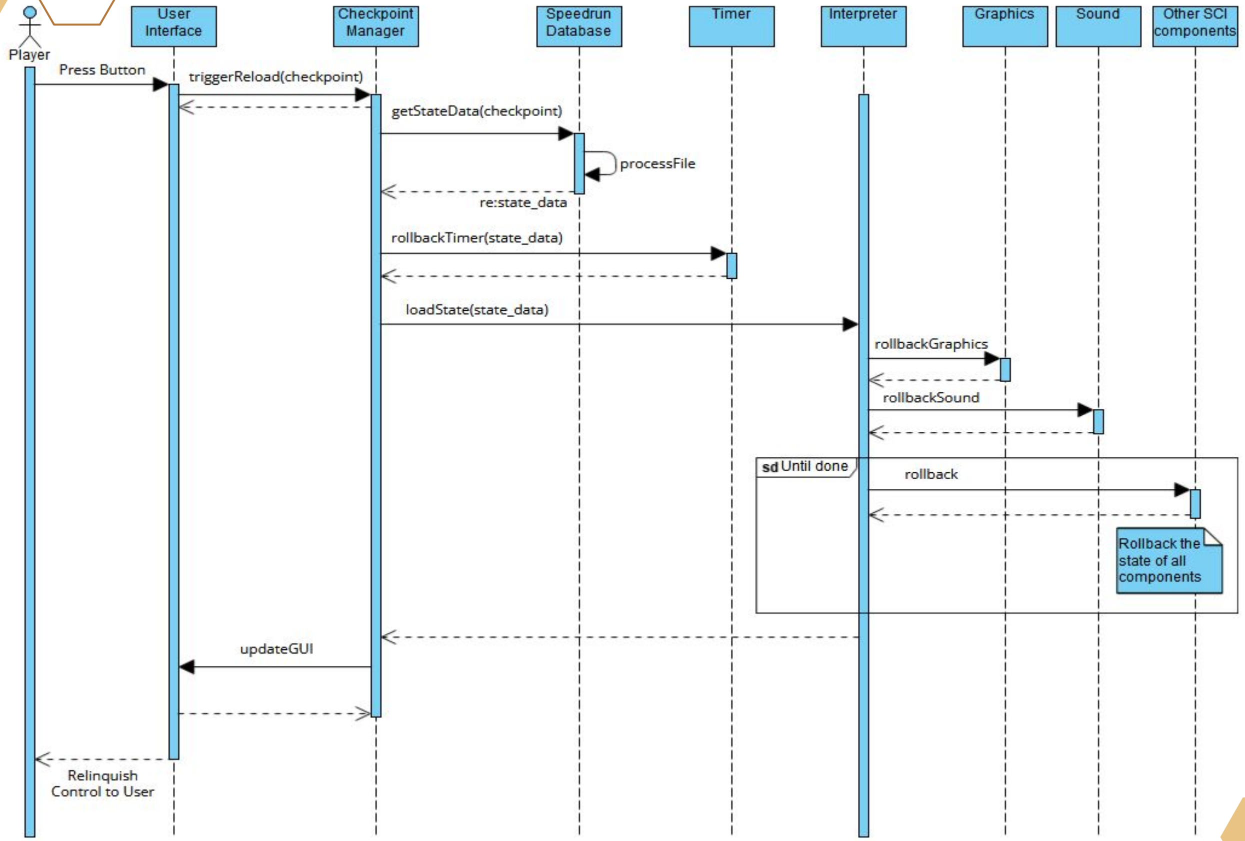
The Checkpoint Manager will:

- Get desired game state from Speedrun Database
- Rollback Timer
- Instruct Interpreter to rollback all components to previous state

Once everything has rolled back, Checkpoint Manager will update User Interface, and then the player will regain control

Legend:


Actor
Instruction
Message
Component
re:item
Return Message
Note
Actor and lifeline
Component and lifeline


Player
User Interface
Checkpoint Manager
Speedrun Database
Timer
Interpreter
Graphics
Sound
Other SCI components
Press Button
triggerReload(checkpoint)
getStateData(checkpoint)
processFile
re:state_data
rollbackTimer(state_data)
loadState(state_data)
rollbackGraphics
rollbackSound
sd Until done
rollback
Rollback the state of all components
updateGUI
Relinquish Control to User

# STARTING A RUN

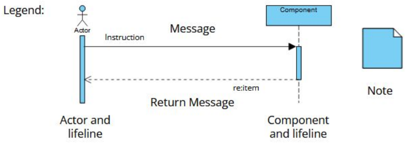
User must enable speedrun mode:

- Select speedrun in User Interface
  - Selection is passed to Checkpoint Manager, which retrieves file from Speedrun Database

When user starts game:

- Interpreter does usual boot sequence
- When done,Speedrun Launcher starts the Timer, and game starts
  - Checkpoint manager monitors game state through Interpreter
  - Gets time and records to Speedrun Database

Legend:



Actor

Instruction

Message

Component

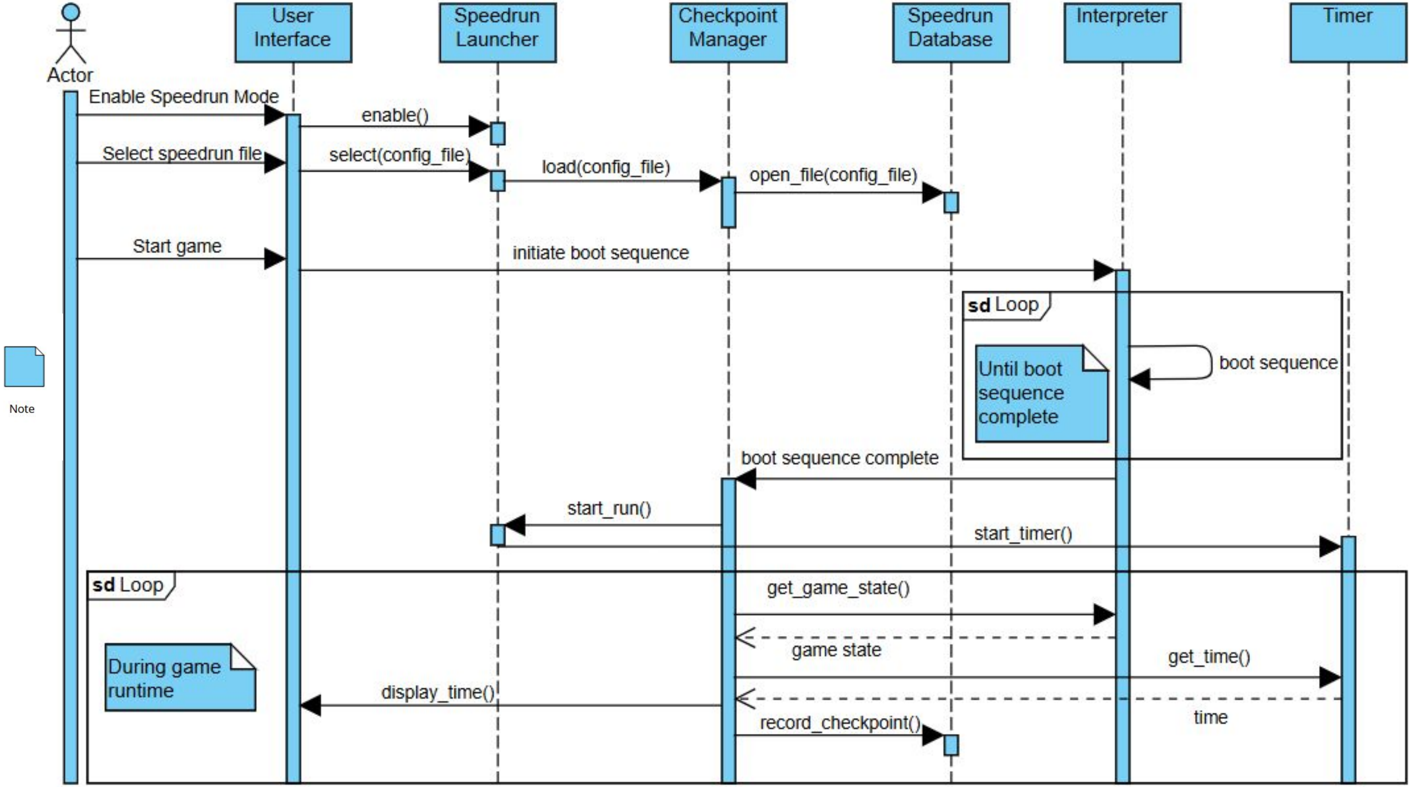
re:item

Return Message

Actor and lifeline

Component and lifeline

Note



Actor

User Interface

Speedrun Launcher

Checkpoint Manager

Speedrun Database

Interpreter

Timer

Enable Speedrun Mode

enable()

Select speedrun file

select(config_file)

load(config_file)

open_file(config_file)

Start game

initiate boot sequence

sd Loop

Until boot sequence complete

boot sequence

boot sequence complete

start_run()

start_timer()

sd Loop

During game runtime

get_game_state()

game state

get_time()

display_time()

time

record_checkpoint()

# POTENTIAL RISKS AND LIMITATIONS

- While our enhancement mainly relies on ScummVM, it still relies on the engine's Interpreter and specific config files
  - Those must be implemented on a case-by-base basis
- Different implementations of engines could cause times to be recorded differently, or might not work at all
- The new components might increase system latency

# PLAN FOR TESTING

Tests will be needed for each engine implementation

A list of use cases will be needed, tests made for each of them

Using the practice mode would help to test. It would be necessary to test that all checkpoints can be loaded correctly when they are saved. It would also help to confirm the system can handle all the checkpoints

# MAINTAINABILITY, EVOLVABILITY, TESTABILITY, AND PERFORMANCE

**Maintainability:**

- Maintenance will take longer, but not much

**Testability:**

- Testability will take longer, but not much

**Evolvability:**

- Harder to add new features that work with the speedrun mode

**Performance:**

- Increased latency, may be negligible on newer machines

# CONCLUSION

.

We considered many different enhancements but settled on speedrunning tools.

There were 2 implementation methods, one in the engine and one in ScummVM.

Doing it in ScummVM satisfied most stakeholders, and was cleaner to implement
This method still has issues:

- Inconsistencies between different engines
- Potential latency or failures

We think that overall it would benefit the system

This experience helped us shape our understanding of software architecture

# LESSONS LEARNED

How **hard** it is to **design new features** for software**.**

Importance of **brainstorming as a team.**

**Box and line** diagrams are very helpful to **understand** a system**.**