Runtime Repair of Software Faults using Event-Driven Monitoring

Chris Lewis, Jim Whitehead
{ cflewis, ejw } @ soe.ucsc.edu

Introduction

Video games have **emergent behaviors** due to **complex subsystems** (physics, AI, graphics)

The complexity of possible states makes games almost **untestable**

We specify **invariants** about the game using a **rule engine** to stop the game staying in an **undesirable state**

This makes it easier to **verify** a game design is **correct**

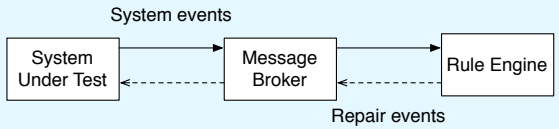
What are Rule Engines?

Rule engines are optimized to match **facts** against **conditions**

We insert **facts** about the **game events** into the rule engine

Designers can **specify conditions** that indicate game design bugs

Fixes can be applied to the game state



Why Rule Engines?

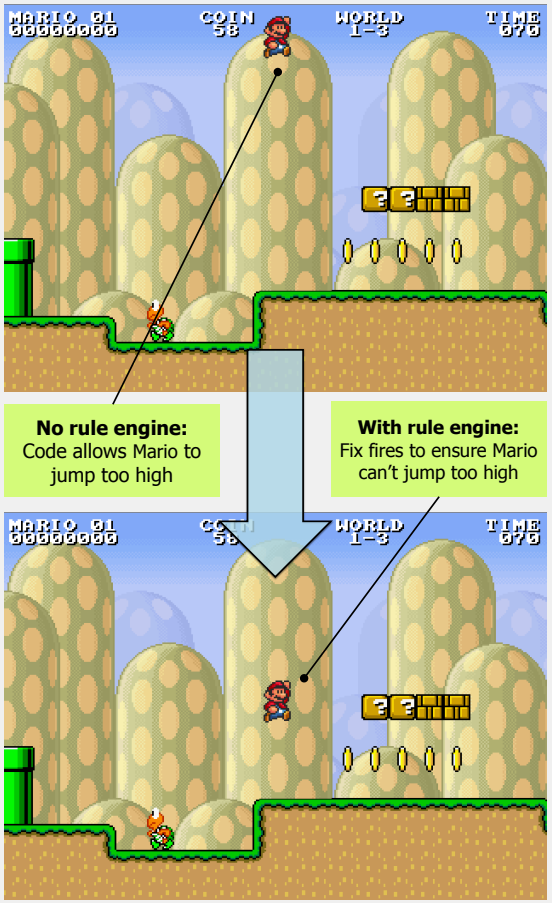
Code is **declarative** and **readable** by non-programmers

Conditions are **optimized**

Works with **cross-cutting concerns**

Complex Event Processing allows simple expression of conditions **over time**

Example Situation: Buggy Lakitu



Example Rule: Mario can only jump 5 blocks high

```
rule "marioTooHigh"
when
    $jump: Jump(jumpTime > 5)
then
    print("Mario jumped too high");
    send(new MarioMovement(false, 10f,
        $jump.getXAcceleration() * 1.5f));
end
```

Read as: "When Mario is set to jump higher than 5 blocks, disable his jump flag and accelerate him towards the ground in an arc."

Example Rule: Mario can only jump for 3 seconds

```
rule "marioJumpTooLong"
when
    $jump: Jump($mario : mario)
    not(Landing(this after[0s,2s] $jump))
then
    print("Mario jumped too long");
    send(new MarioMovement(false, null, null);
end
```

Read as: "When there is a jump fact, but no landing fact within 2 seconds after Mario jumps, turn off his jump flag and let the game design choose his downward acceleration."

Broader Relevance

Super Mario World is the simplest non-trivial case, the technique can **scale** to **current commercial games**

Shows rule engines can be used as an **enforced specification language**

Capable of detecting problems with any software of **complex state**

Event monitoring could be used to collect important event sequences and **automatically create rules** that define the general/non-buggy case