A Smart Contract Debugger for Flint

Noel Lee (kyl116)

What is Flint?

Smart contract programming language

Developed by an MEng student, then continued (since 2018)

Supports **Ethereum**

What is Flint?

Safer alternative to Solidity (the most popular smart contract language for Ethereum)

Language features for safety:

- **Caller protection** protects contract from unauthorized callers
- **Type states** prevent unexpected state changes

Debugging 101

Debugging without a debugger?

Debugging 101

Debugging without a debugger?

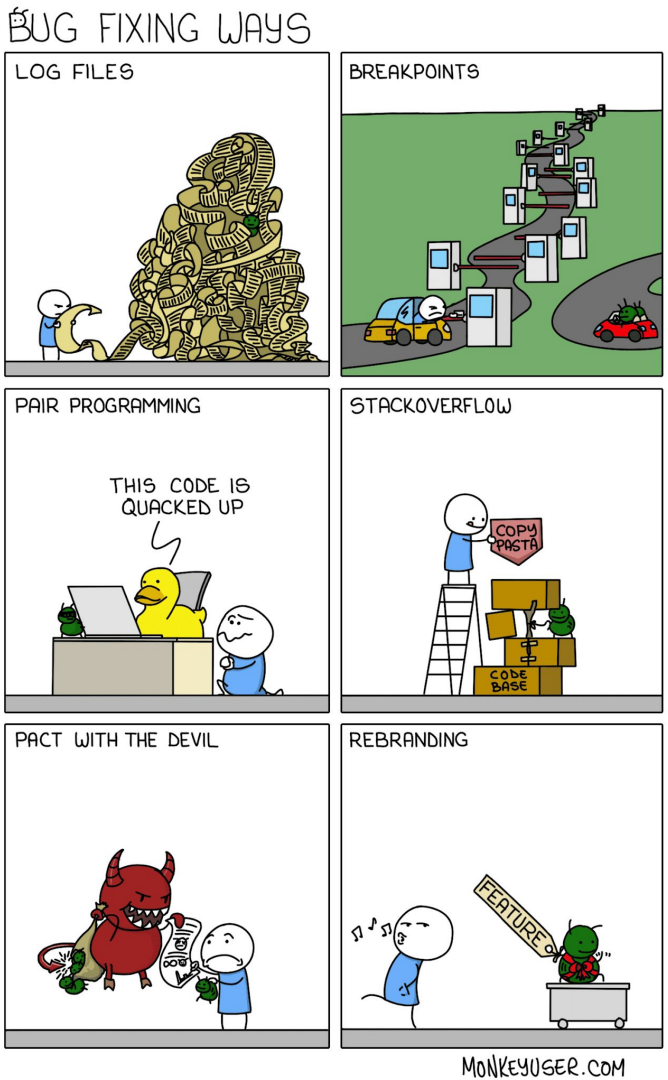
- Rubber duck

Debugging 101

Debugging without a debugger?

- Rubber duck
- Print debugging

```
print("here")
if something {
  print("i'm here")
  doSomething()
} else {
  print("testing”)
  a = 42 / 0
  print(something, a)
  print("the universe is broken")
}
```



Debugging 101

Debugging without a debugger?

- Rubber duck
- ~~Print debugging~~

```
print("here")
if something {
  print("i'  ere")
  doSomethi
} else {
  print("tes
  a = 42 /
  print(some ing, a)
  print("the universe is broken")
}
```

Debugging 101

Debugging without a debugger?

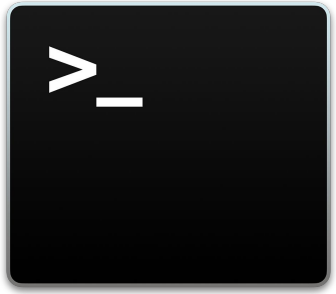
- Rubber duck
- ~~Print debugging~~
- Binary search

```
Error: Returned error: VM Exception while processing transaction: revert
```

https://vmexceptionwhileprocessingtransactionrevert.com/

Flint debugger

Provides a **command line interface (CLI)** and
a **VS Code extension**



Trace-based

Features:

- Step through source code
- Breakpoints
- Variable inspection (+ type state)
- Reverse debugging

```
970 LT
971 OR
972 AND
973 PUSH1
    ^^^^^
975 DUP2
976 EQ
977 PUSH2

file:///Users/noel/fyp/evaluation/flint/Supermarket.flint:

51:      cashRegister.transfer(source: &value)
52:
53:      if amount <= 2 && amount <= stock {
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
54:        stock -= amount

fdb> n

977 PUSH2
980 JUMPI
985 JUMPDEST
986 PUSH2
    ^^^^^
989 DUP5
990 PUSH1
992 PUSH1

file:///Users/noel/fyp/evaluation/flint/Supermarket.flint:

52:
53:      if amount <= 2 && amount <= stock {
54:        stock -= amount
           ^^^^^^^^^^^^^^^
55:      }

fdb>
```

[Extension Development Host] - Supermarket.flint — evaluation

RUN Ask for transaction h

Supermarket.flint | supermarket.js | txs.txt | runner.js | package.js

flint > Supermarket.flint

VARIABLES

- stack
- memory
- storage
  - cashRegister: 1000020 Wei
  - stock: 1005011
- flint
  - state: Opened
- evm
  - gas: 54401
  - gasCost: 3
  - jump: -
  - op: SSTORE
  - pc: 1006

WATCH

CALL STACK — PAUSED ON BREAKPOINT

frame Supermarket.flint 54:7

BREAKPOINTS

- Supermarket.flint flint 50
- Supermarket.flint flint 54

```
39    }
40
41    public func employ(position: Int, employee: Address) mutates (staff) {
42      staff[position] = employee
43    }
44  }
45
46  FlintSupermarket @(Opened) :: (any) {
47    @payable
48    public func buy(amount: Int, implicit value: Wei) mutates (stock, customers) {
49      let enoughCash: Bool = value.getRawValue() == amount * 10
50      assert(enoughCash)
51      cashRegister.transfer(source: &value)
52
53      if amount <= 2 && amount <= stock {
54        stock -= amount
55      }
56
57      incrementCustomers(count: 1)
58    }
59  }
60
61  FlintSupermarket @(Opened) :: (manager) {
62    public func close() mutates (customers) {
63      resetCustomers()
64      become Closed
65    }
66  }
67
68  FlintSupermarket @(Closed) :: (manager) {
69    public func open() {
```

master* | 0 0 | Ask for transaction hash (evaluation) | Compilation completed successfully! | 0 | -- NORMAL -- | Spaces: 2 | UTF-8 | LF | Flint

Demo

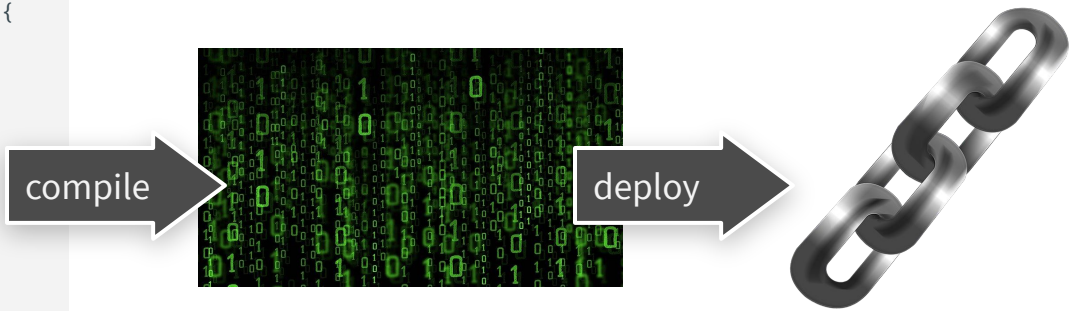
“Supermarket” Contract

```
contract FlintSupermarket (Opened, Closed) {
  var staff: Address[3] = []
  let manager: Address

  var stock: Int = 0

  var customers: Int = 0

  var cashRegister: Wei
}
// more code ...
```

Typical smart contract development process

```
contract FlintSupermarket (Opened, Closed) {
  var staff: Address[3] = []
  let manager: Address

  var stock: Int = 0

  var customers: Int = 0

  var cashRegister: Wei
}
// more code ...
```

Flint contract



EVM bytecode

Test blockchain
(e.g. Ganache)

“Supermarket” Contract

Example #1

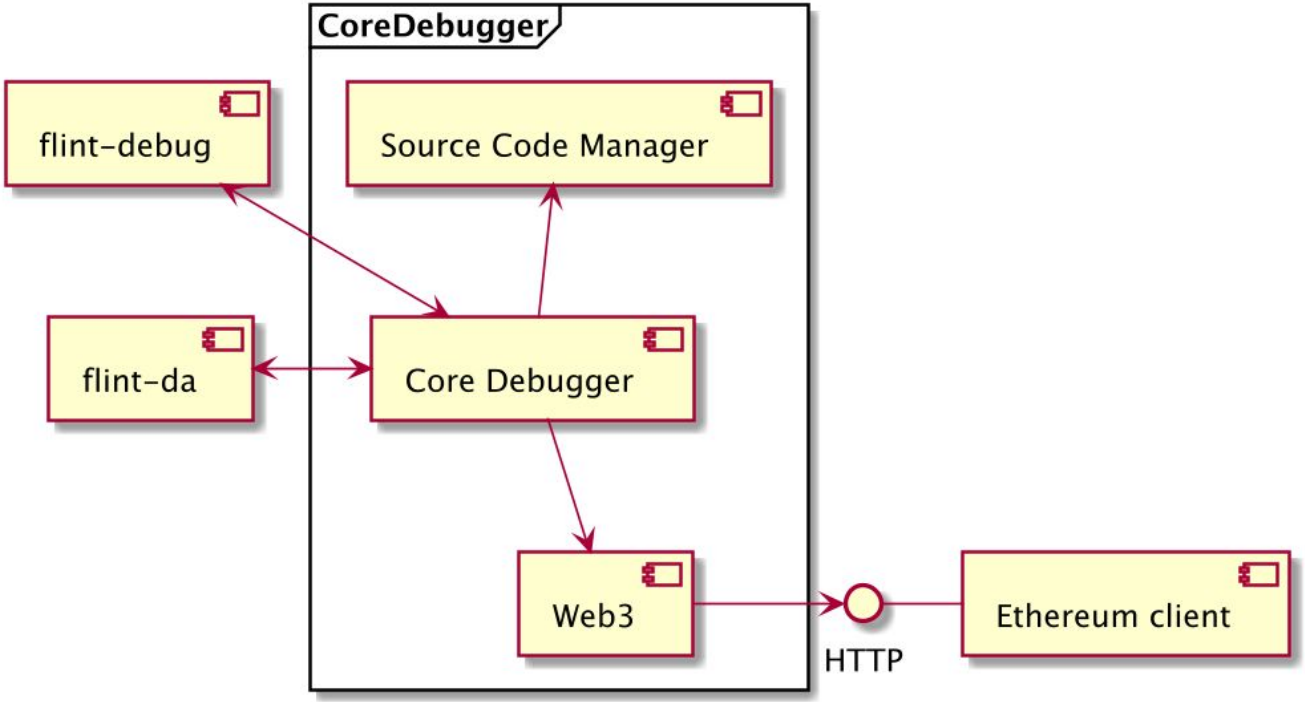
```
supermarket.buy(amount: 2, value: 20)
```

Example #2 (type states)

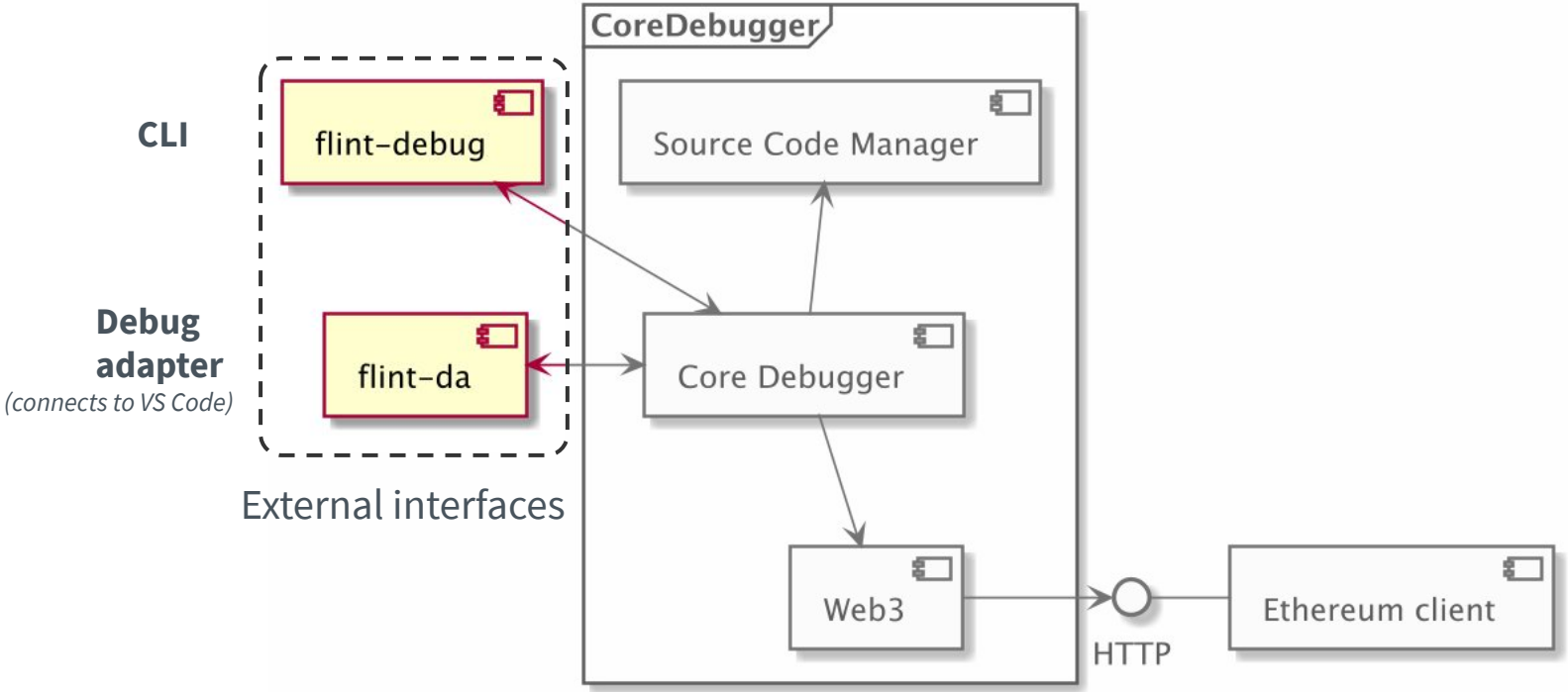
```
supermarket.close()
```

Implementation

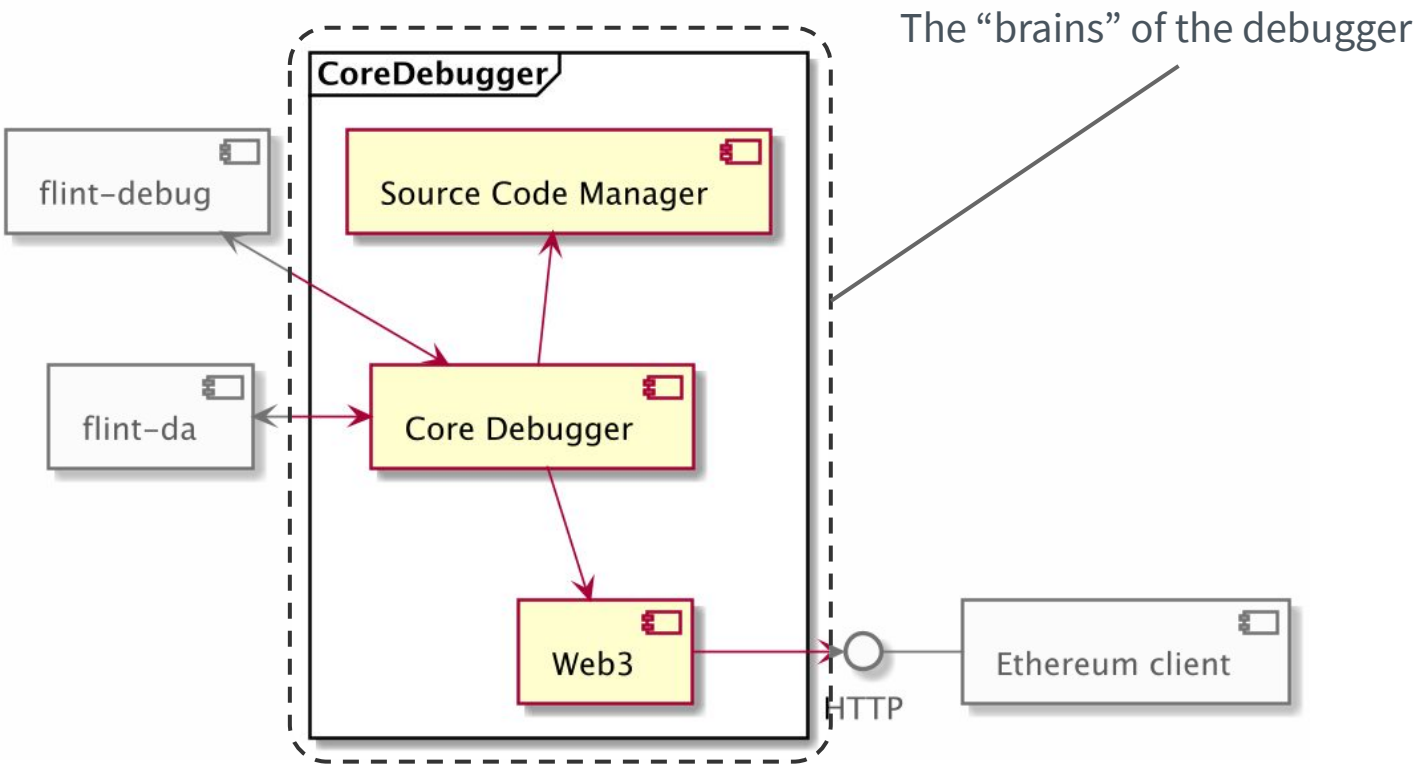
Flint Debugger architecture



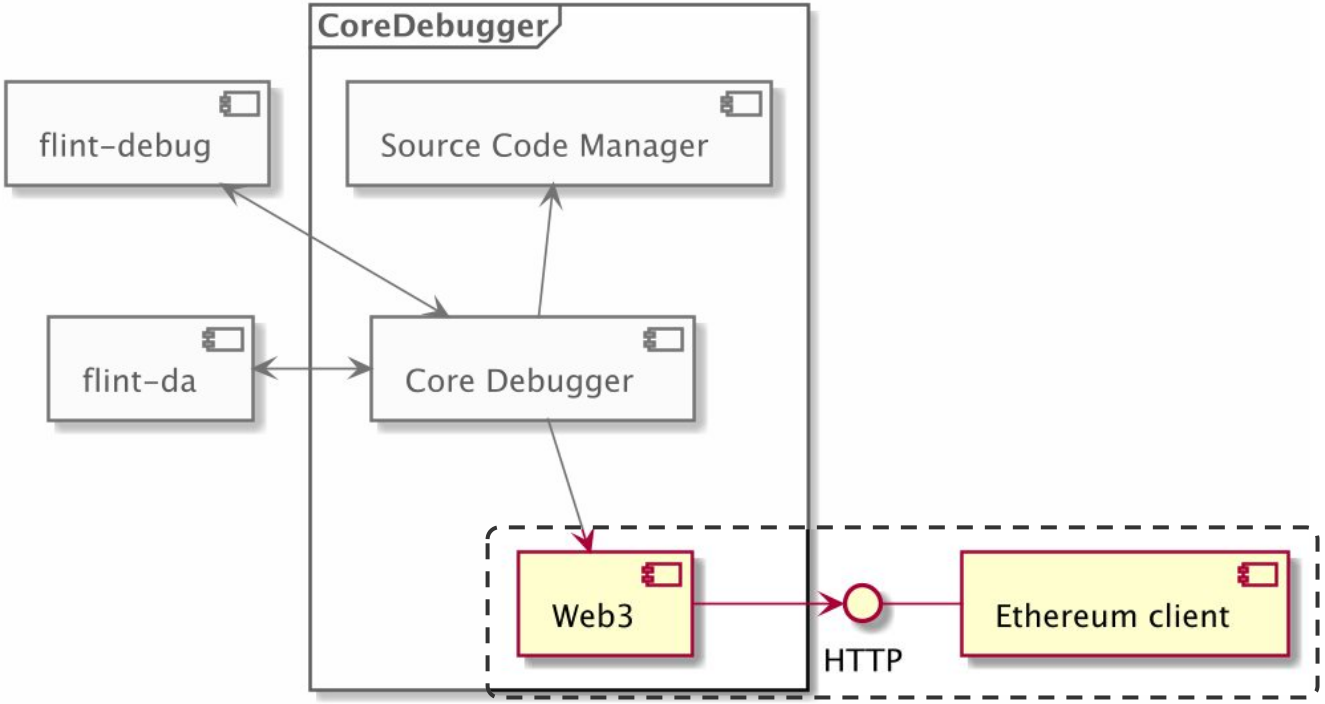
Flint Debugger architecture



Flint Debugger architecture



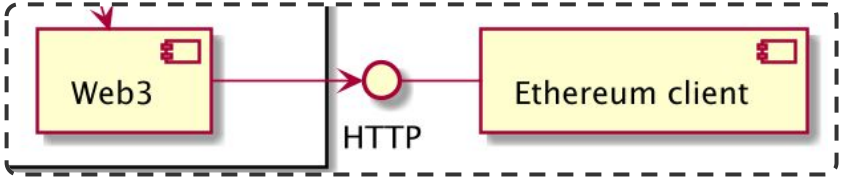
Flint Debugger architecture



Web3

- **Connects** to Ethereum client
- Retrieve data on the blockchain
  - **Transaction trace**
  - Deployed contract info

- *Library used:*
  - https://github.com/Boilertalk/Web3.swift + our own extension
  - (to support API method **`debug_traceTransaction`**, which we needed)



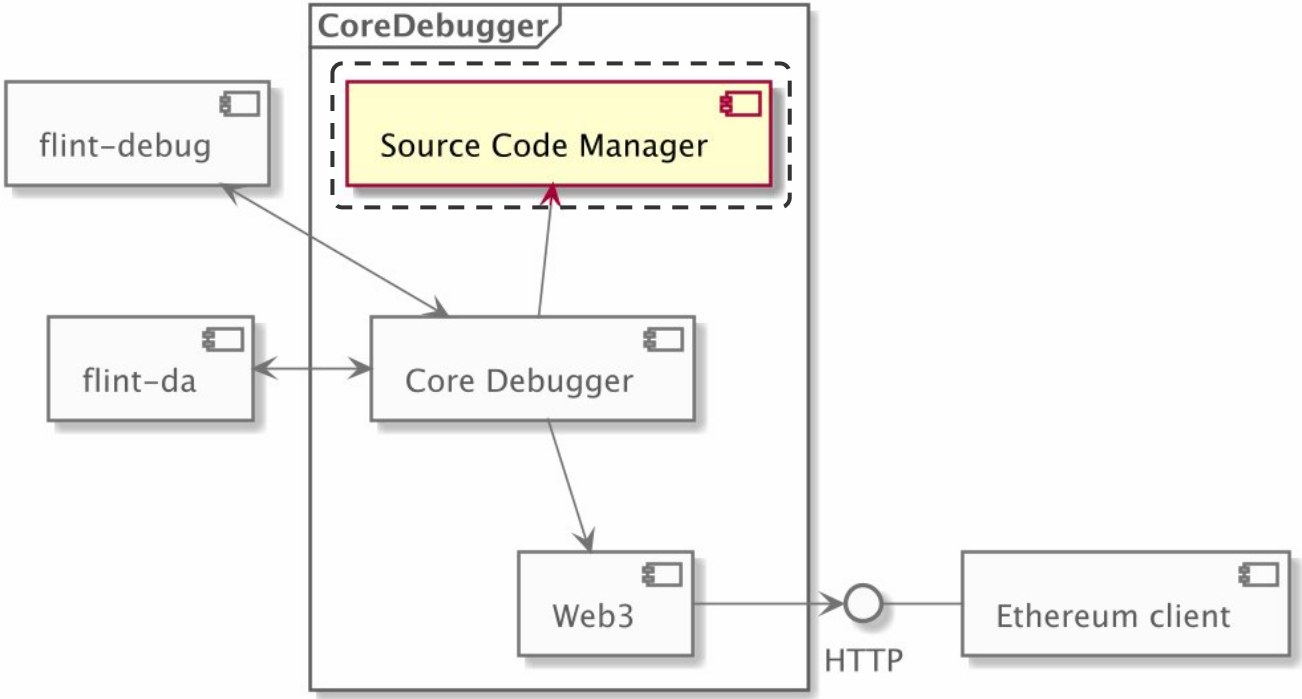
```
{
   "structLogs":[
      {
         "gas":6700613,
         "pc":5,
         "op":"PUSH1",
         "stack":[
            "0000000000000000000000000000002710",
            "0000000000000000000000000000000112"
         ],
         "memory":[
            "0000000000000000000000000000000000",
            "0000000000000000000000000000000000",
            "0000000000000000000000000000000080"
         ],
         "storage":[
            "00000000":"0000000000000000000000012345678",
            "00000020":"000000000000000000000000deadbeef"
         ]
      },
      // ...
   ]
}
```

Where are we in the contract?

Which contract variables are these?

Example transaction trace *(some fields omitted)*

Flint Debugger architecture



Source code manager

Q: How do we know where each instruction is generated from, i.e. its **source location**?

A: ***Source maps!***

```
...

if true {
  a = a + 1
}
...
```

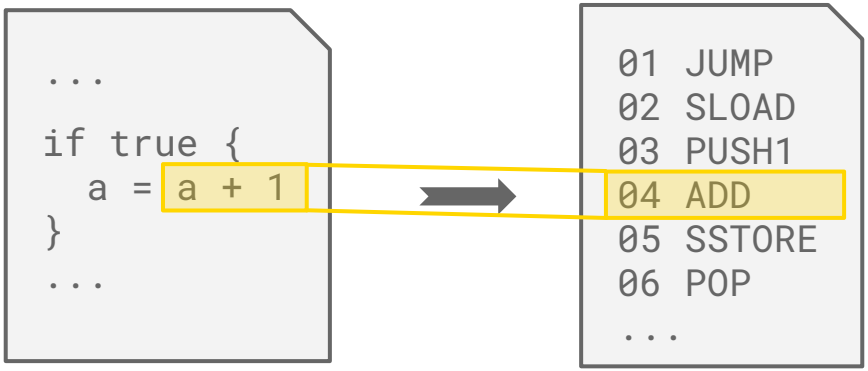
example.flint

```
01 JUMP
02 SLOAD
03 PUSH1
04 ADD
05 SSTORE
06 POP
...
```

example.bin
(EVM bytecode)

Source map

- It **maps** generated code to its source location
- Flint compiler is extended to generate a source map as a **compiler artifact**

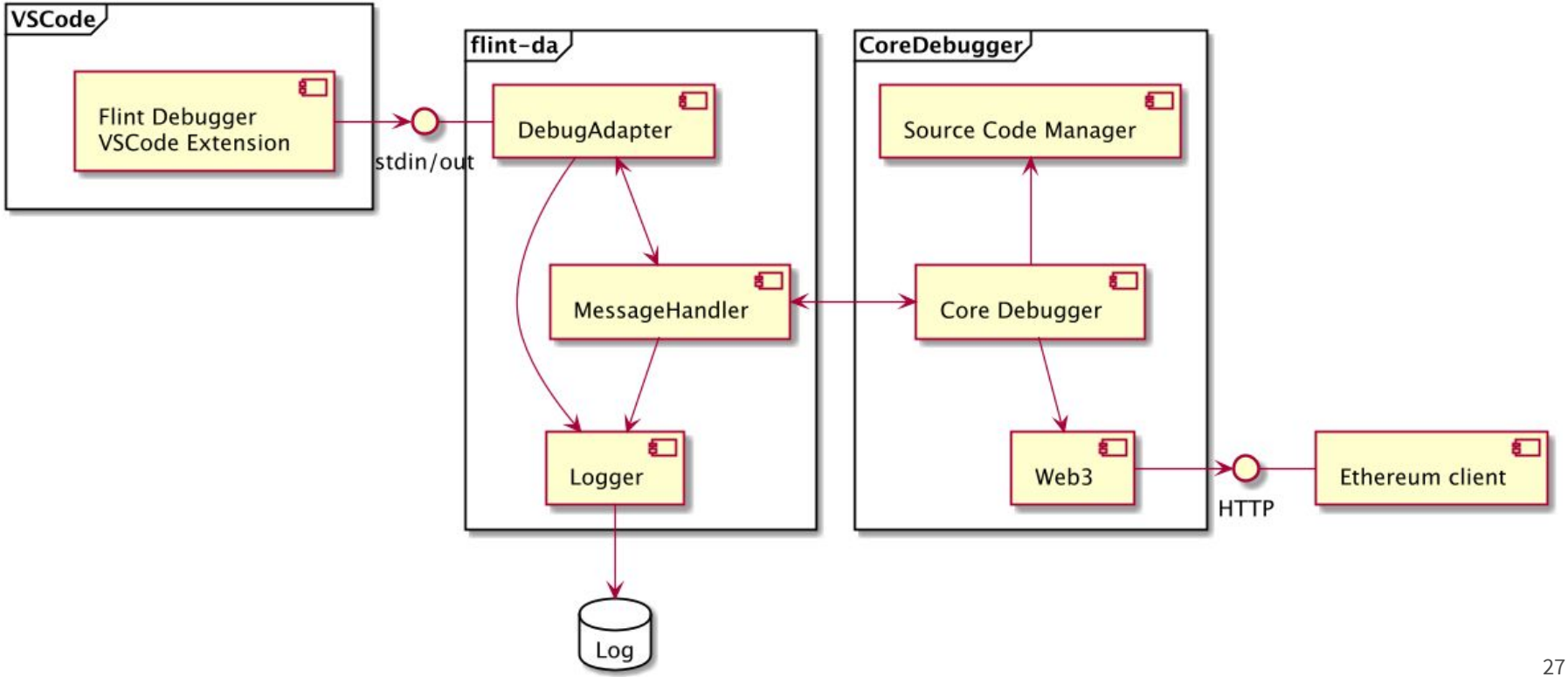


example.flint

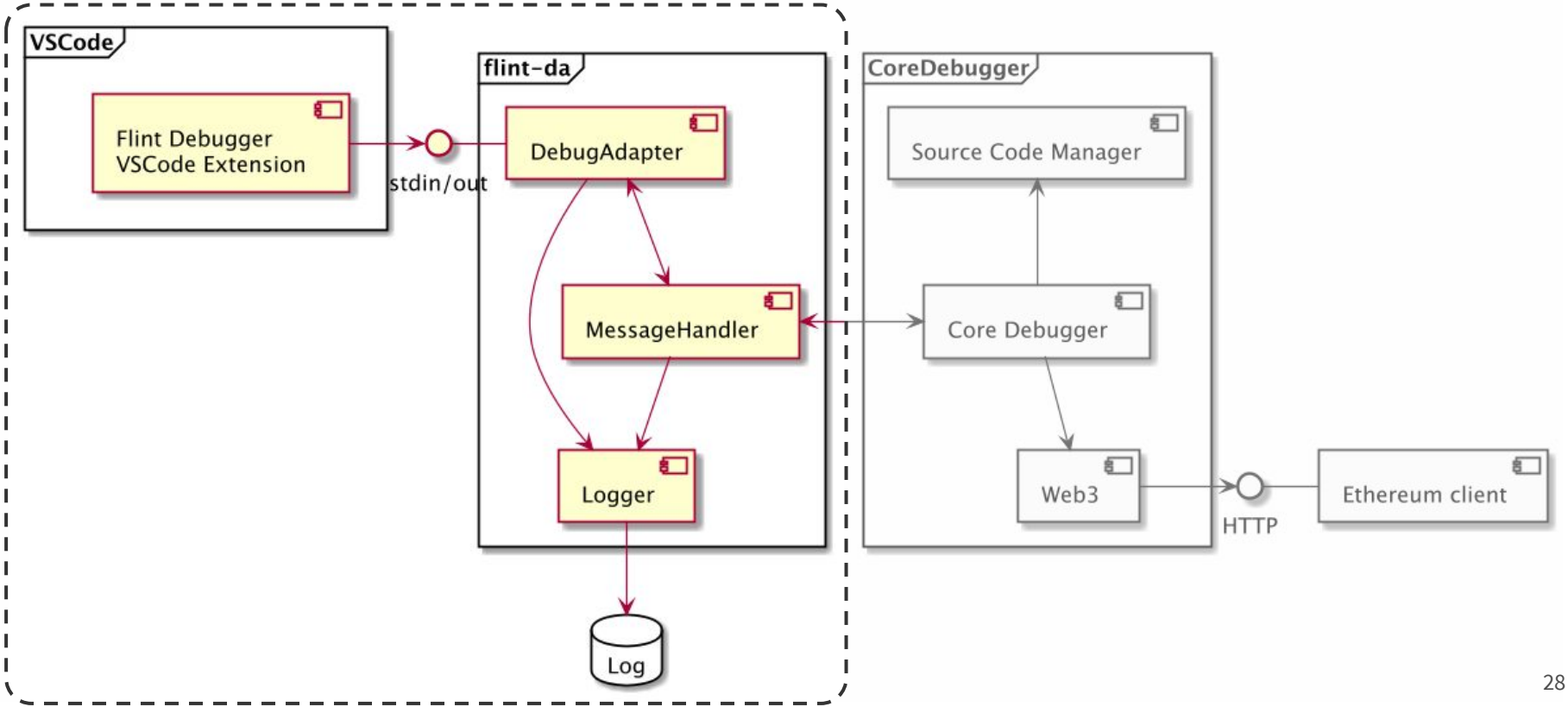
example.bin
(EVM bytecode)

VS Code extension

Debug adapter/VS Code extension

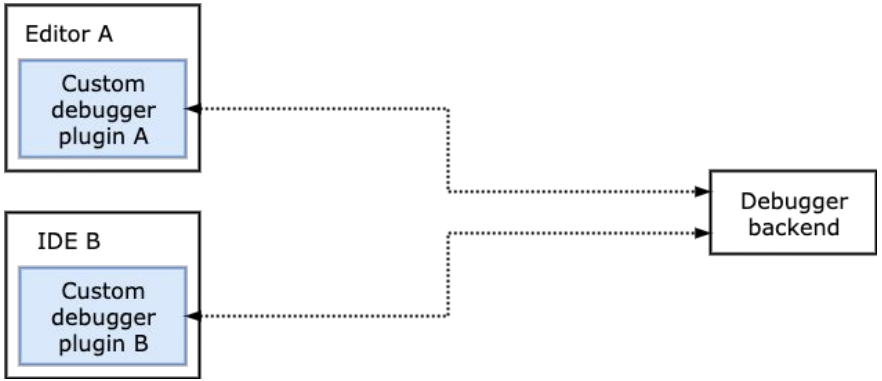


Debug adapter/VS Code extension

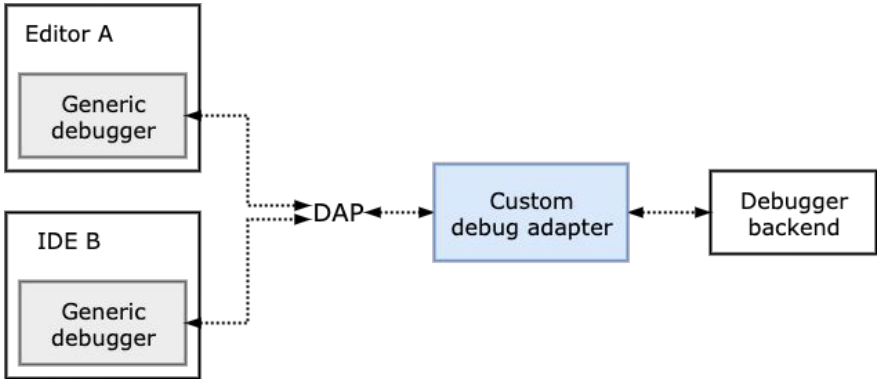


Debug Adapter Protocol (DAP)

- Developed by Microsoft
- **Standardizes** the communication between editors/IDEs and debuggers



without DAP



with DAP

Evaluation

Compiler overhead

How does **changing code generation** impact the Flint compiler's performance?
0.3074s -> 0.3264s (+ 6.18%) overhead

How does **source map generation** impact the Flint compiler's performance?
+ 2.6s (+ 800%) overhead

Tests

✔ Pass

Travis CI Dashboard Changelog Documentation Help

noellee / flint build passing

Current Branches Build History Pull Requests More options

debugger CodeFragment: added new tests #26 passed Restart build

Commit 77c93c4
Compare 3ea710e..77c93c4
Branch debugger
Noel

Ran for 27 min 20 sec
15 minutes ago

no language set
AMD64

Job log View config

Remove log Raw log

```
$ bash -c 'echo $BASH_VERSION'                                                    0.09s
  1 Worker information                                                            worker_info
  6                                                                               0.01s
  7 Build system information                                                      system_info
167
168                                                                               3.69s
                                                                                  docker_mtu
                                                                                  resolvconf
169 $ git clone --depth=50 --branch=debugger https://github.com/noellee/flint.git noellee/flint   git.checkout   1.18s
179
180 $ git submodule update --init --recursive                                     git.submodule   15.33s
190 4.4.20(1)-release
```

Flint debugger

vs. Solidity debuggers

Existing smart contract debuggers

Truffle

- Solidity development framework
- Provides CLI tool **`truffle debug`**
- Trace-based

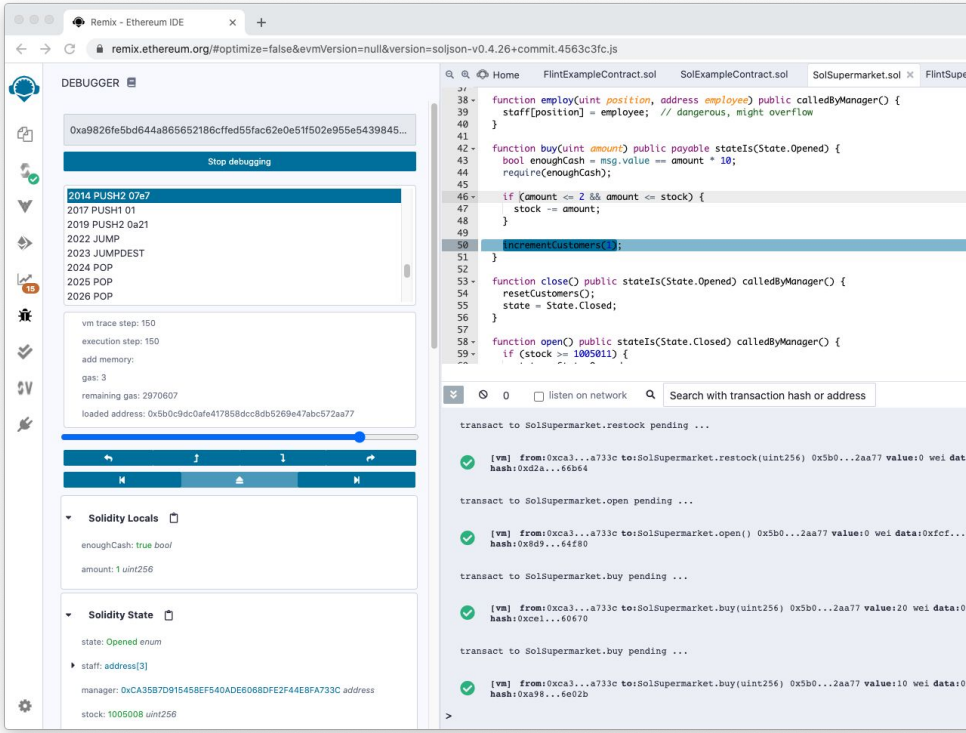
```
➜  truffle-example truffle debug 0xc0dc498cb48cf7e649891da466336912a52b6a206229daa04fdf80fd7df09b28
Starting Truffle Debugger...
✔ Compiling your contracts...
✔ Gathering information about your project and the transaction...

Addresses called: (not created)
 0x2563923049eE9f9601e4ebB3715FE2384c9D6baa - Counter

Commands:
(enter) last command entered (step next)
(o) step over, (i) step into, (u) step out, (n) step next
(;) step instruction (include number to step multiple)
(p) print instruction, (l) print additional source context
(h) print this help, (q) quit, (r) reset
(t) load new transaction, (T) unload transaction
(b) add breakpoint, (B) remove breakpoint, (c) continue until breakpoint
(+) add watch expression (`+:<expr>`), (-) remove watch expression (-:<expr>)
(?) list existing watch expressions and breakpoints
(v) print variables and values, (:) evaluate expression - see `v`


Counter.sol:

1: pragma solidity >=0.4.21 <0.7.0;
2:
3: contract Counter {
   ^^^^^^^^^^^^^^^^^^

debug(development:0xc0dc498c...)>
```

TRUFFLE

Existing smart contract debuggers

Remix

- Browser-based IDE (Solidity, Vyper)
- Provides a Solidity debugger
- Trace-based
- Supports reverse debugging



How our Flint debugger compares

	Flint debugger	Truffle	Remix
Stepping	✔	✔	✔
Breakpoints	✔	✔	✔
Reverse debugging	✔	✘	✔
Contract variable inspection	✔	✔	✔
Raw state inspection	✔	✔	✔
Expression evaluation	✘	✔	✘
Type states	✔	N/A	N/A

Conclusion

What we've done

Debugger for Flint

CLI and **VS Code extension**

Extended open source library

Web3.swift

Built open source Swift library

SwiftDAP for the Debug Adapter Protocol

Repositories summary

Flint

54 changed files, with 2,237 additions and 114 deletions.

Web3.swift

3 changed files, with 64 additions and 0 deletions.

SwiftDAP

(new repo) 1375 loc.

vscode-flint-debug

(new repo) 53 loc.

Challenges

Legacy codebase

How to change code generation mechanism without refactoring too much?

Background research

Only way to understand how existing smart contract debuggers work: read their code

Limited library support

Why not JavaScript/TypeScript?

Future work

Improve source map generation **performance**

Support **expression evaluation**

Support **Ewasm**

Q&A