Domain-Specific Smart Contract Languages: Why and How

13-11-‘18

Tijs van der Storm
storm@cwi.nl / @tvdstorm

About me

- Senior researcher at CWI in the SWAT team
- Professor software engineering at University of Groningen
- Expertise: language design, language engineering, language workbenches, … META stuff ;)
- Currently running 1 project with blockchain, 2 about to start in 2019

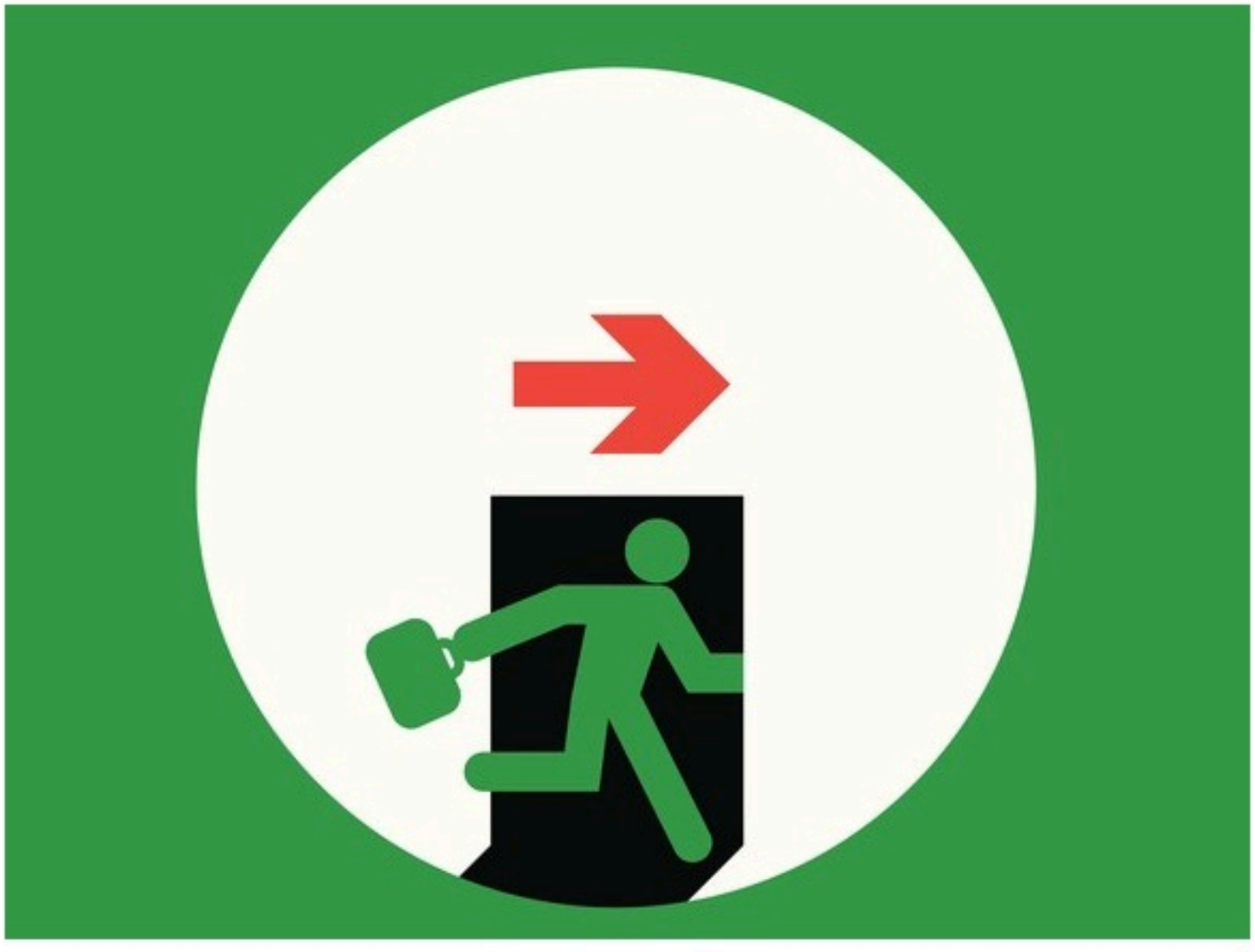
CWI
Centrum Wiskunde & Informatica

university of groningen

www.rascal-mpl.org

KLINT FINLEY BUSINESS 06.18.16 04:30 AM

A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN



Thoughts on The DAO Hack

Emin Gün Sirer

dao ethereum

June 17, 2016 at 09:45 AM

← Older

Newer →

We just lived through the nightmare scenario we were worried about as we called for a moratorium on The DAO: someone exploited a weakness in the code of The DAO to empty out more than 2M ($40M USD) ether.

The exploit seems to have targeted the *reentrancy problem* in the 'splitDAO' function. The reentrancy problem is related to but distinct from the unchecked-send problem that was discussed on this blog yesterday. Both problems are well-known, identified by Least Authority's audit of the Ethereum virtual machine as problems that can affect applications, as well as Peter Vessenes's recent blog post. In essence, a call that looks like a regular call can easily be turned into a recursive call, and unless the application is coded very carefully, it can be used to make multiple withdrawals when only one should be allowed. It looks like the attacker took advantage of it to withdraw substantial sums.

https://www.coindesk.com/hacks-scams-attacks-blockchains-biggest-2017-disasters/

2. Parity Wallet Breach

Issues began in July when the U.K.-based startup discovered a vulnerability in version 1.5 of its wallet software, resulting in at least 150,000 ethers being stolen from user accounts.

4. Parity Wallet Freeze

Occurring suddenly this November, a Parity user accidentally found a bug in the software code, freezing more than $275 million in ether in the wallet's second major incident of 2017.

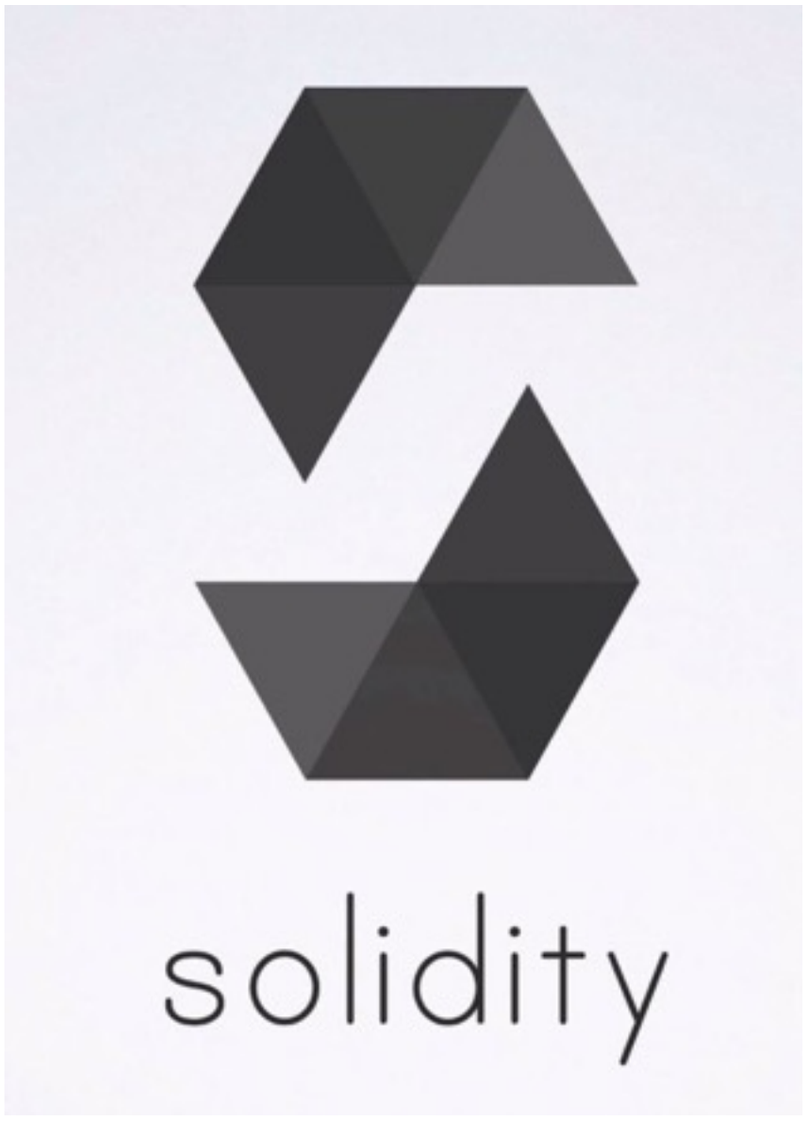

BUG

```
/* Allow another contract to spend some tokens in your behalf */
function approve(address _spender, uint256 _value)
    returns (bool success) {
    allowance[msg.sender][_spender] = _value;
    return true;
}

/* Approve and then comunicate the approved contract in a single tx */
function approveAndCall(address _spender, uint256 _value, bytes _extraData)
    returns (bool success) {
    tokenRecipient spender = tokenRecipient(_spender);
    if (approve(_spender, _value)) {
        spender.receiveApproval(msg.sender, _value, this, _extraData);
        return true;
    }
}
```

```
/* A contract attempts to get the coins */
function transferFrom(address _from, address _to, uint256 _value) returns (bool success) {
    if (balanceOf[_from] < _value) throw;                 // Check if the sender has enough
    if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
    if (_value > allowance[_from][msg.sender]) throw;   // Check allowance
    balanceOf[_from] -= _value;                           // Subtract from the sender
    balanceOf[_to] += _value;                             // Add the same to the recipient
    allowance[_from][msg.sender] -= _value;
    Transfer(_from, _to, _value);
    return true;
}

/* This unnamed function is called whenever someone tries to send ether to it */
function () {
    throw;     // Prevents accidental sending of ether
}
```

Features of Solidity

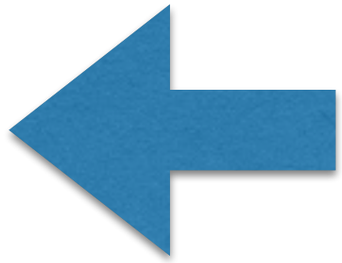
- Javascript - objects + “stuff”
  - functional abstraction
  - conditionals
  - recursion
  - loops
  - execution handling
  - side-effects



☆ MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts

OOPSLA

Session: Security Chair(s): Tobias Wrigstad



!!!!

Ethereum is a distributed blockchain platform, serving as an ecosystem for smart contracts: full-fledged inter-communicating programs that capture the transaction logic of an account. Unlike programs in mainstream languages, the execution of an Ethereum smart contract is restricted by a gas limit: execution proceeds as long as gas is available. Thus, gas is a valuable resource that can be manipulated by an attacker to provoke unwanted behavior in a victim's smart contract (e.g., wasting or blocking funds of said victim). Gas-focused vulnerabilities exploit undesired behavior when a contract (directly or through other interacting contracts) runs out of gas. Such vulnerabilities are among the hardest for programmers to protect against, as out-of-gas behavior may be uncommon in non-attack scenarios and reasoning about it is far from trivial. In this paper, we classify and identify gas-focused vulnerabilities, and present MadMax: a static program analysis technique to automatically detect gas-focused vulnerabilities with very high confidence. Our approach combines a control-flow-analysis-based decompiler and declarative program-structure queries. The combined analysis captures high-level domain-specific concepts (such as "dynamic data structure storage" and "safely resumable loops") and achieves high precision and scalability. MadMax analyzes the entirety of smart contracts in the current Ethereum blockchain in just 10 hours (with decompilation timeouts in 8% of the cases) and flags contracts with a current (though highly volatile) monetary value of over $5B as vulnerable. Manual inspection of a sample of flagged contracts shows that 81% of the sampled warnings do indeed lead to vulnerabilities, which we report on in our experiment.

Link to Publication: http://www.nevillegrech.com/madmax-oopsla18.pdf

Neville Grech
University of Athens

Anton Jurisevic
University of Sydney

Bernhard Scholz
The University of Sydney
Australia

Michael Kong
University of Sydney

Lexi Brent
University of Sydney

Yannis Smaragdakis
University of Athens
Greece

https://2018.splashcon.org/track/splash-2018-OOPSLA#About

SMART CONTRACTS ARE NEITHER SMART NOR CONTRACTS

FRITZ HENGLEIN
UNIVERSITY OF COPENHAGEN

http://hjemmesider.diku.dk/~henglein/smart-contracts-are-neither.pdf

- Smart contracts = self-executing contracts (programs) in complex Turing-complete programming language

- Rules and actions intermixed:
  Not contracts

- Hard to analyze, low-level programs:
  Not smart

Programming

```
.text:131411EF                  ; FUNCTION CHUNK AT .text:13141239 SIZE 000
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143000 SIZE 000
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143522 SIZE 000
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13145C99 SIZE 000
.text:131411EF
.text:131411EF 68 F1 AF 5B FF                   push    0FF5B4FF1h
.text:131411F4 E9 A0 4A 00 00                   jmp     loc_13145C99
.text:131411F4                  start           endp
.text:131411F4
.text:131411F4                  ; ---------------------------------------------------------------------------
.text:131411F9 64 69 31         byte_131411F9   db 64h, 69h, 31h        ; D
.text:131411FC 60 16 A2 75 9A C4+               dd 75A21660h, 0CACDC49Ah, 7
.text:131411FC CD CA FA 7B 9D 7B+               dd 6EEE544Dh, 10D69610h, 0F
.text:131411FC D5 D4 48 2B 85 4B+               dd 0EB6323E4h, 82B58465h, 0
.text:13141238                  ; ---------------------------------------------------------------------------
.text:13141238 58                               pop     eax
.text:13141239                  ; START OF FUNCTION CHUNK FOR start
.text:13141239
.text:13141239                  loc_13141239:                           ; C
.text:13141239 52                               push    edx
.text:1314123A 55                               push    ebp
.text:1314123B 53                               push    ebx
.text:1314123C 51                               push    ecx
.text:1314123D 9C                               pushf
.text:1314123E 57                               push    edi
.text:1314123F 50                               push    eax
.text:13141240 56                               push    esi
.text:13141241 51                               push    ecx
.text:13141242 68 00 00 00 00                   push    0
.text:13141247 8B 74 24 28                      mov     esi, [esp+2Ch+var_4
.text:1314124B BF F9 11 14 13                   mov     edi, offset byte_13
.text:13141250
.text:13141250                  loc_13141250:                           ; C
.text:13141250 89 F3                            mov     ebx, esi
.text:13141252 03 34 24                         add     esi, [esp+2Ch+var_2
.text:13141255
.text:13141255                  loc_13141255:                           ; C
.text:13141255                                                          ; s
```

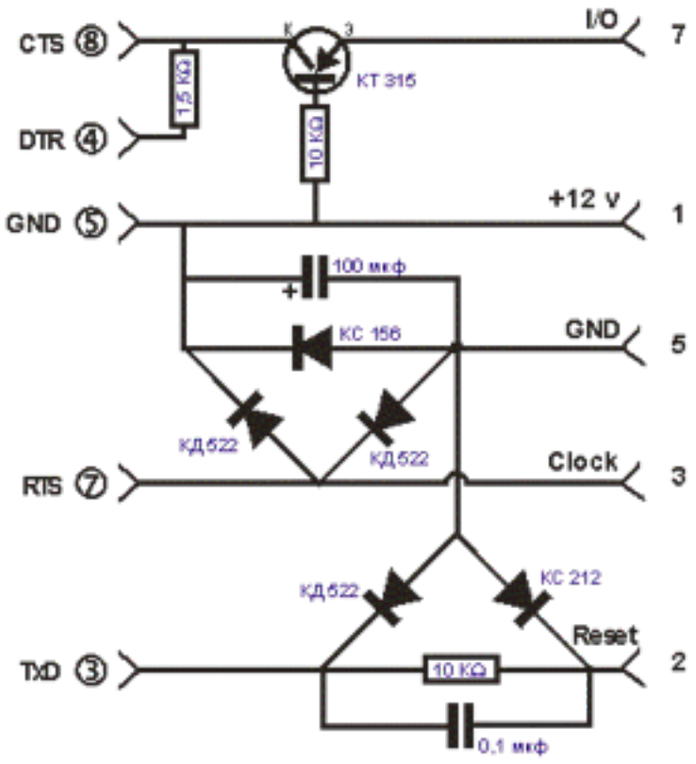

```
.text:131411EF                  ; FUNCTION CHUNK AT .text:13141239 SIZE 0000009F BYTES
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143000 SIZE 00000122 BYTES
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143522 SIZE 000000F1 BYTES
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13145C99 SIZE 0000000A BYTES
.text:131411EF
.text:131411EF 68 F1 AF 5B FF                  push    0FF5BAFF1h
.text:131411F4 E9 A0 4A 00 00                  jmp     loc_13145C99
.text:131411F4                 start           endp
.text:131411F4
.text:131411F4                  ; ---------------------------------------------------------------------------
.text:131411F9 64 69 31        byte_131411F9   db 64h, 69h, 31h        ; DATA XREF: start+5C↓o
.text:131411FC 60 16 A2 75 9A C4+              dd 75A21660h, 0CACDC49Ah, 7B9D7BFAh, 2B48D4D5h, 687E4B85h
.text:131411FC CD CA FA 7B 9D 7B+              dd 6EEE544Dh, 10D69610h, 0FE39DFC2h, 0ED0EB3C7h, 5CFE8AADh
.text:131411FC D5 D4 48 2B 85 4B+              dd 0EB6323E4h, 82B58465h, 0D0947C7Bh, 0C80CD20Dh, 30DE6A19h
.text:13141238                  ; ---------------------------------------------------------------------------
.text:13141238 58                              pop     eax
.text:13141239                  ; START OF FUNCTION CHUNK FOR start
.text:13141239
.text:13141239                 loc_13141239:                           ; CODE XREF: start+4AAF↓j
.text:13141239 52                              push    edx
.text:1314123A 55                              push    ebp
.text:1314123B 53                              push    ebx
.text:1314123C 51                              push    ecx
.text:1314123D 9C                              pushf
.text:1314123E 57                              push    edi
.text:1314123F 50                              push    eax
.text:13141240 56                              push    esi
.text:13141241 51                              push    ecx
.text:13141242 68 00 00 00 00                  push    0
.text:13141247 8B 74 24 28                     mov     esi, [esp+2Ch+var_4]
.text:1314124B BF F9 11 14 13                  mov     edi, offset byte_131411F9
.text:13141250
.text:13141250                 loc_13141250:                           ; CODE XREF: start+D9↓j
.text:13141250 89 F3                           mov     ebx, esi
.text:13141252 03 34 24                        add     esi, [esp+2Ch+var_2C]
.text:13141255
.text:13141255                 loc_13141255:                           ; CODE XREF: start+98↓j
.text:13141255                                                         ; start+A8↓j ...
```

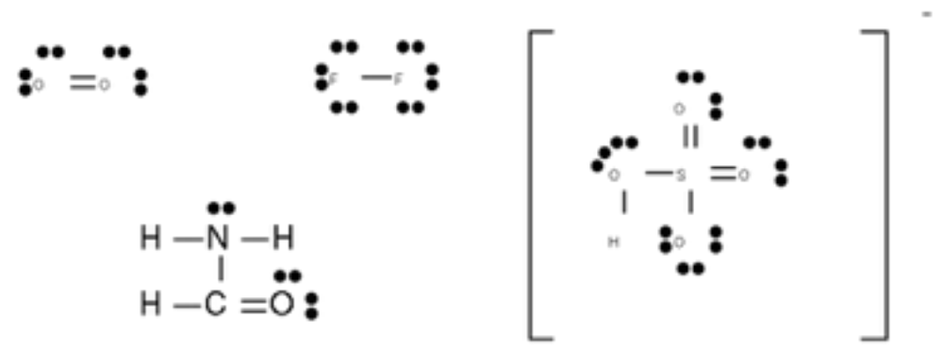
What is high-level?

- Language employs concepts and terms in its notation from problem domain instead of solution domain (= computers ;)


```
.text:131411EF                  ; FUNCTION CHUNK AT .text:13141239 SIZE 0000009F BYTE
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143000 SIZE 00000122 BYTE
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143522 SIZE 000000F1 BYTE
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13145C99 SIZE 0000000A BYTE
.text:131411EF
.text:131411EF 68 F1 4F 5B FF                   push    0FF5B4FF1h
.text:131411F4 E9 A0 4A 00 00                   jmp     loc_13145C99
.text:131411F4                  start           endp
.text:131411F4
.text:131411F4                  ; ---------------------------------------------------------------------------
.text:131411F9 64 69 31         byte_131411F9   db 64h, 69h, 31h        ; DATA XREF:
.text:131411FC 60 16 A2 75 9A C4+               dd 75A21660h, 0CACDC49Ah, 7B9D7BFAh,
.text:131411FC CD CA FA 7B 9D 7B+               dd 6EEE544Dh, 10D69610h, 0FE39DFC2h,
.text:131411FC D5 D4 48 2B 85 48+               dd 0EB6323E4h, 82B58465h, 0D0947C7Bh,
.text:13141238                  ; ---------------------------------------------------------------------------
.text:13141238 58                               pop     eax
.text:13141239                  ; START OF FUNCTION CHUNK FOR start
.text:13141239
.text:13141239                  loc_13141239:                           ; CODE XREF:
.text:13141239 52                               push    edx
.text:1314123A 55                               push    ebp
.text:1314123B 53                               push    ebx
.text:1314123C 51                               push    ecx
.text:1314123D 9C                               pushf
.text:1314123E 57                               push    edi
.text:1314123F 50                               push    eax
.text:13141240 56                               push    esi
.text:13141241 51                               push    ecx
.text:13141242 68 00 00 00 00                   push    0
.text:13141247 8B 74 24 28                      mov     esi, [esp+2Ch+var_4]
.text:1314124B BF F9 11 14 13                   mov     edi, offset byte_131411F9
.text:13141250
.text:13141250                  loc_13141250:                           ; CODE XREF:
.text:13141250 89 F3                            mov     ebx, esi
.text:13141252 03 34 24                         add     esi, [esp+2Ch+var_2C]
.text:13141255
.text:13141255                  loc_13141255:                           ; CODE XREF:
.text:13141255                                                          ; start+A0↓j
```

Domain-specific languages

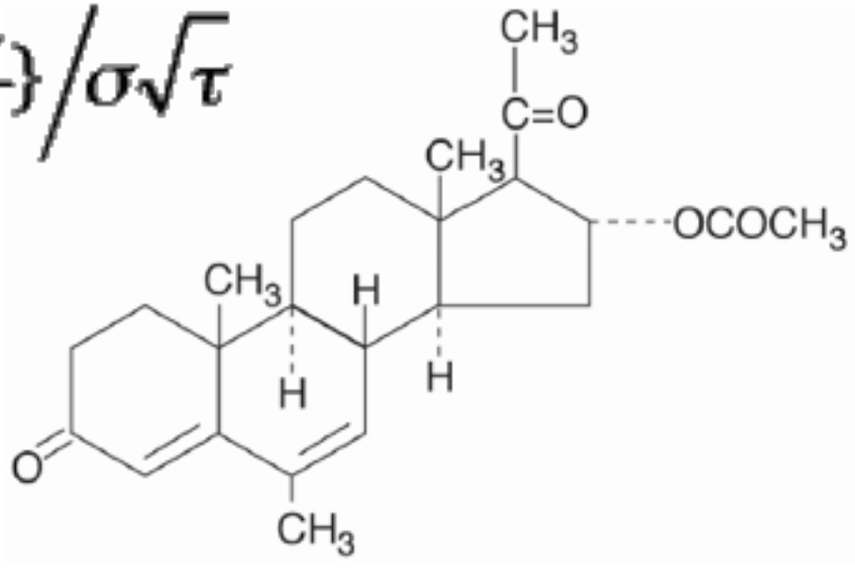




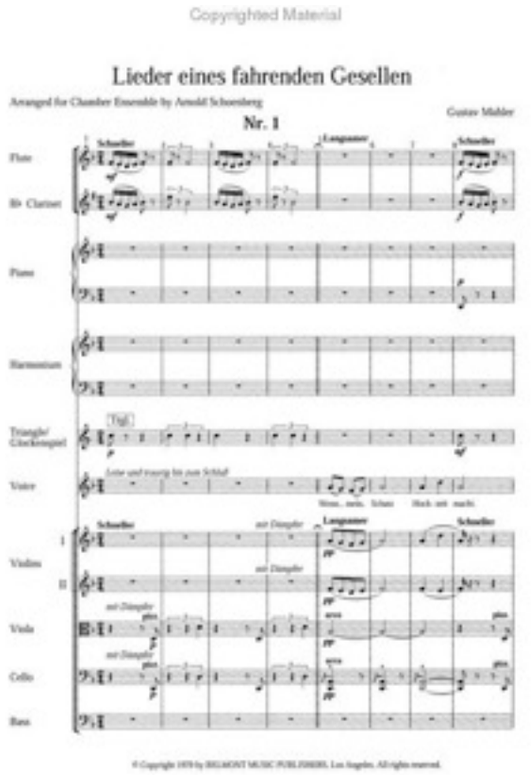
$$c_t = S_t N(h) - Xe^{-rt} N(h - \sigma\sqrt{\tau})$$

where

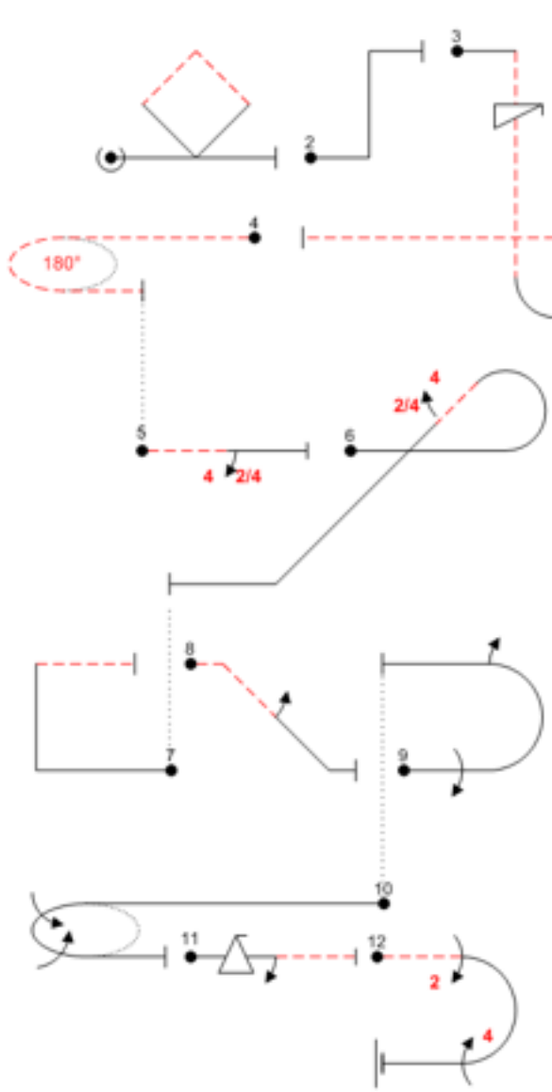
$$h = \{\ln(\frac{S}{X}) + r\tau + \frac{\sigma^2\tau}{2}\} \Big/ \sigma\sqrt{\tau}$$

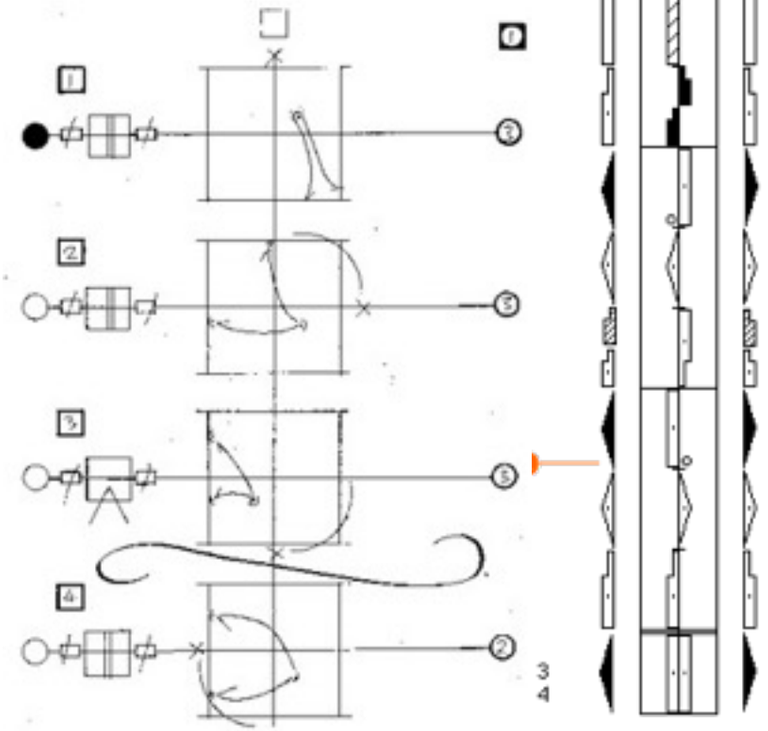


```
E|-------------|-------------|-------------|-------------|
B|-------------|-------------|-3---3--0----|-1---1------|
G|-0-2--0------|-0-2-0-------|-------------|--------0---|
D|--------2----|--------2----|-------------|-------------|
A|-------------|-------------|-------------|-------------|
E|-------------|-------------|-------------|-------------|
```

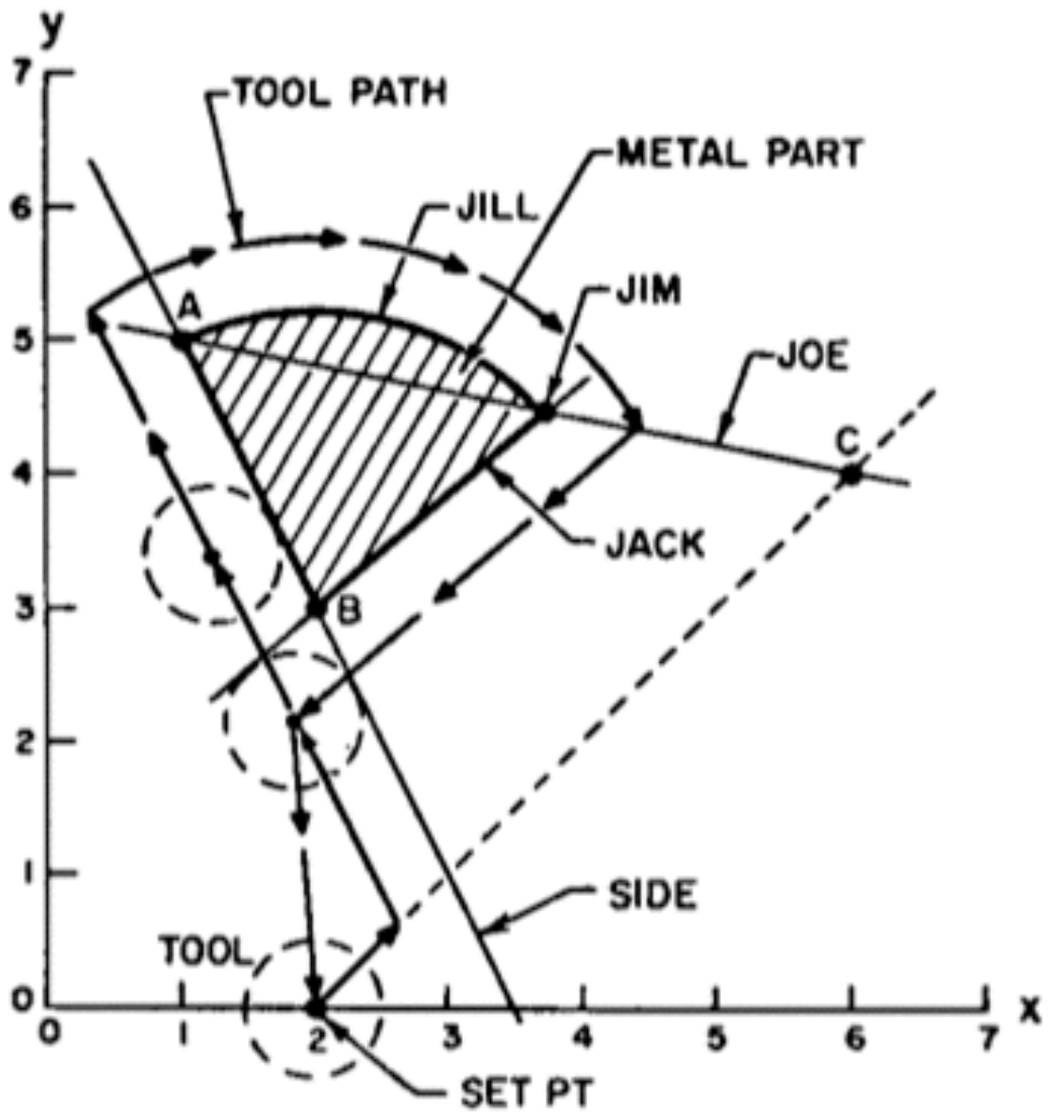



```
White:  G. A. Anderssen
Black:  J. Dufresne
Opening:  Evans Gambit
Location: Berlin, 1854
    White          Black
   -------        -------
 1. P-K4           P-K4
 2. Kt-KB3         Kt-QB3
 3. B-B4           B-B4
 4. P-QKt4         BxKtP
 5. P-B3           B-R4
 6. P-Q4           PxP
 7. O-O            P-Q6
 8. Q-Kt3          Q-B3
 9. P-K5           Q-Kt3
10. R-K1           KKt-K2
11. B-R3           P-Kt4
12. QxP            R-QKt1
13. Q-R4           B-Kt3
14. QKt-Q2         B-Kt2?
15. Kt-K4          Q-B4?
16. BxQP           Q-R4
17. Kt-B6 ch!      PxKt
18. PxP            R-Kt1
19. QR-Q1!         QxKt
20. RxKt ch        KtxR
21. QxP ch!        KxQ
22. B-B5 dbl ch    K-K1
23. B-Q7 ch        K-B1
24. BxKt mate
```





APT




```
A   = POINT / 1, 5
B   = POINT / 2, 3
C   = POINT / 6, 4
      TL DIA / +1.0, INCH
      FEDRAT / 30, IPM
SET PT = FROM, POINT / 2, 0
      IN DIR, POINT / C
SIDE = GO TO, LINE / THRU, A, AND, B
      WITH, TL LFT, GO LFT, ALONG / SIDE
JILL = GO RGT, ALONG, CIRCLE / WITH, CTR AT, B, THRU, A
JOE = LINE / THRU, A, AND, C
JIM = POINT / X LARGE, INT OF, JOE, WITH, JILL
JACK = LINE / THRU, JIM, AND, B
      GO RGT, ALONG / JACK, UNTIL, TOOL, PAST, SIDE
      GO TO / SET PT
      STOP, END, FINI
```

A domain

Domain Knowledge

Hard labour

Code

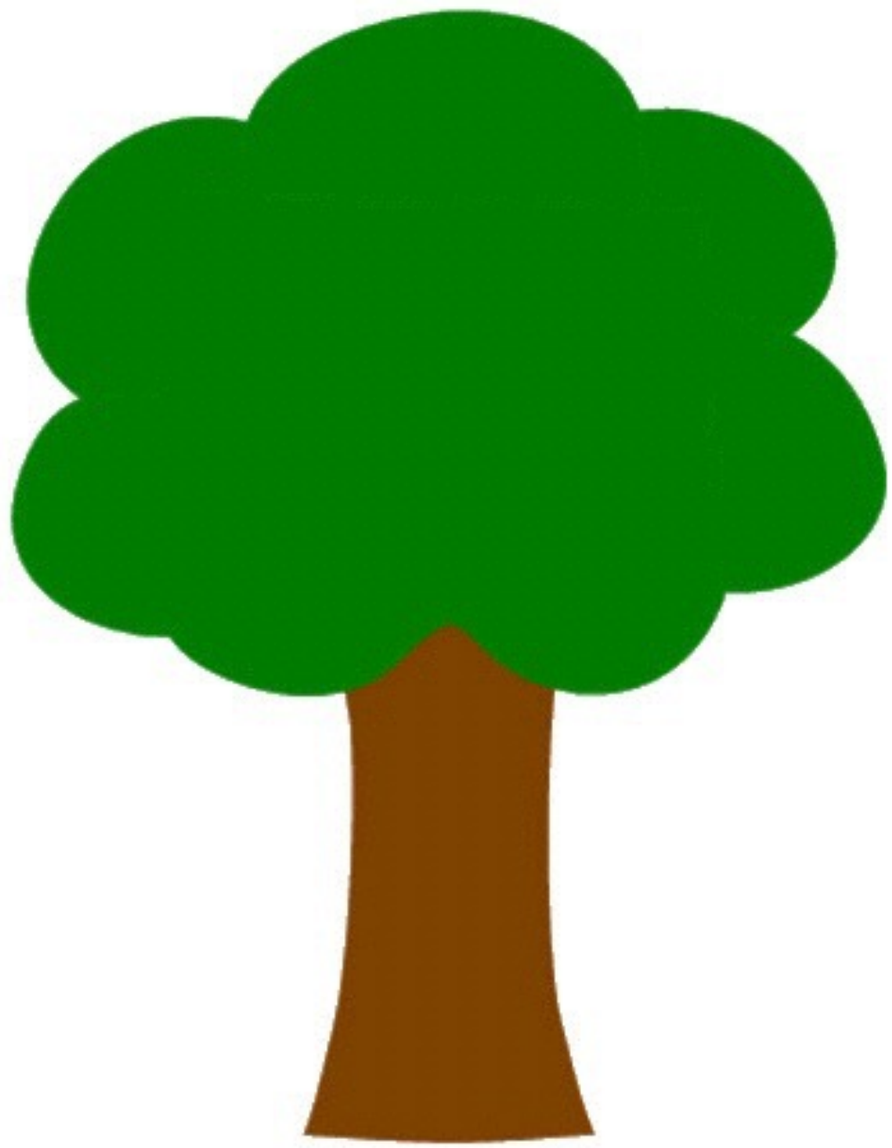
Domain Knowledge

Hard labour

Code

Change/verify/etc.

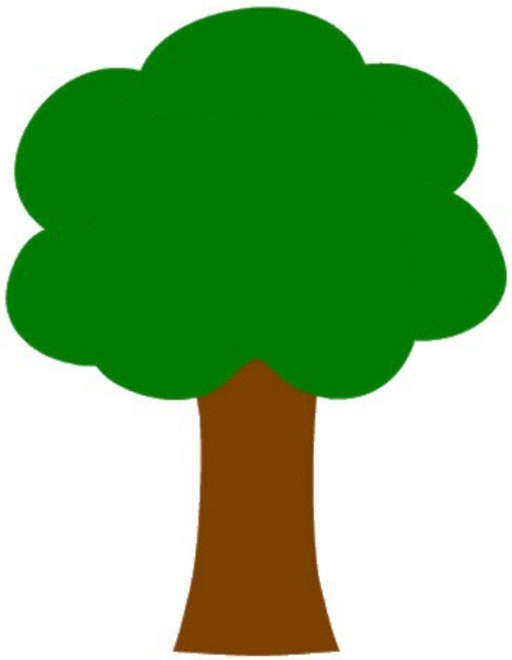
Modeling the domain



Domain-specific Language

Automation

Code



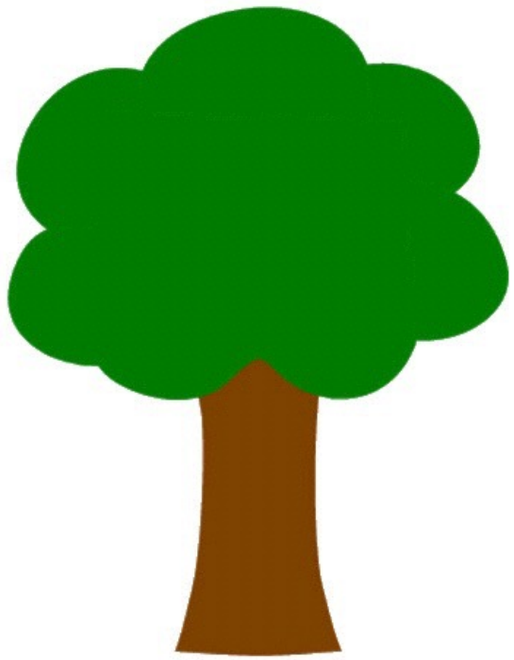
“What”
formalized
requirements

“How”
formalized design
and architecture

Domain-specific Language

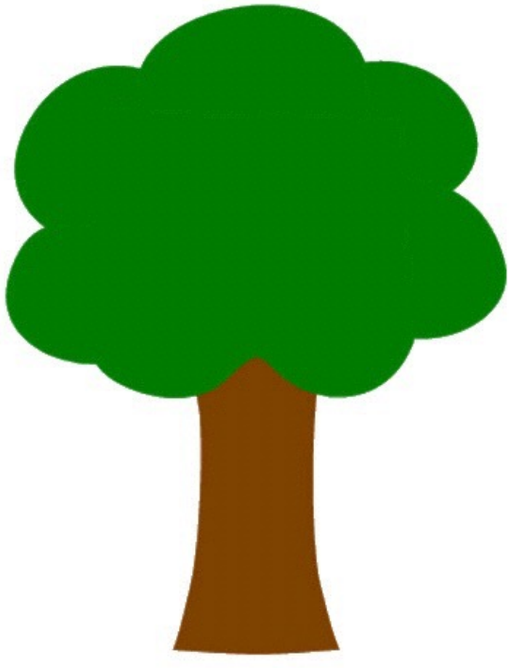
Automation

Code



Change/verify etc.

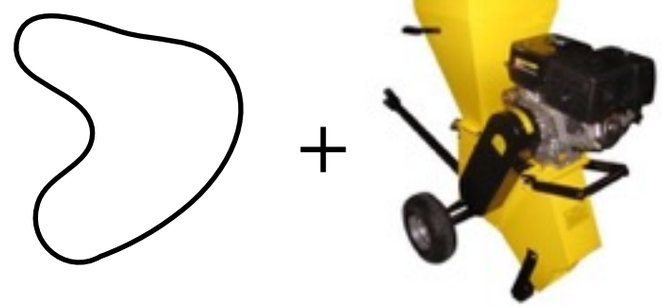
#hoedan?



CWI SWAT

- SWAT: Software Analysis & Transformation
- Tools for understanding existing software better (reverse engineering)
- Designing better languages to improve software construction (forward engineering)
- Key tech: Rascal (www.rascal-mpl.org)
- Meta programming language and **language workbench**

Rascal



www.rascal-mpl.org

DSLs @ SWAT

- Derric: digital forensics
- QL: questionnaires
- Machino: router software
- Halide*: image processing
- AlleAlle: relational modeling
- MicroMachinations: game economies

swat.engineering

A Formal Language for Analyzing Contracts

```
future(rightA="1 round lot pork bellies",
       rightB="$1,500.00",
       p = "for delivery in July 2002") =

    when withinPeriod(p)
        to Holder rightA   with   to Counterparty rightB
    then terminate
```

```
callOptionAmerican (rightA="1 round lot XYZ Corp.",
                    rightB="$2,000/lot",
                    time="end of trading on last trading day of August") =
    when beforeTime(time)
        when choiceOf(Holder)
            to Holder rightA with to Counterparty rightB
    when afterTime(time)
        terminate
```

```
loanPayments(payment, schedule) =
    for schedule
        when withinPeriod(schedule.next)
            payment
carPurchase(car, downPayment, monthlyPayment, schedule)  =
    to Counterparty getTitle(car)   with   to Holder downPayment
    then to Holder loanPayment(monthlyPayment, schedule)
```

https://nakamotoinstitute.org/contract-language/

Nick Szabo

Other domains

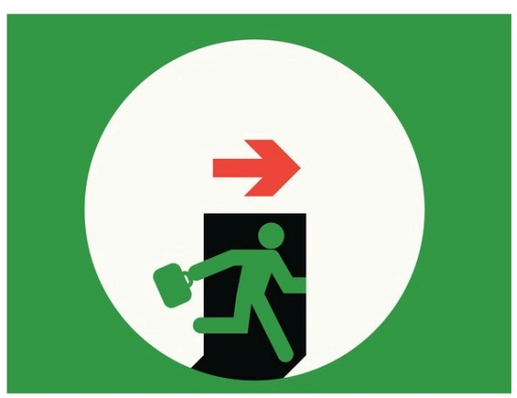
- Financial contracts
- Supply chain
- Logistics
- Identity management
- Food? ;)
- etc.

Domain-specific smart contract languages

```
/* Allow another contract to spend some tokens in your behalf */
function approve(address _spender, uint256 _value)
    returns (bool success) {
    allowance[msg.sender][_spender] = _value;
    return true;
}

/* Approve and then comunicate the approved contract in a single tx */
function approveAndCall(address _spender, uint256 _value, bytes _extraData)
    returns (bool success) {
    tokenRecipient spender = tokenRecipient(_spender);
    if (approve(_spender, _value)) {
        spender.receiveApproval(msg.sender, _value, this, _extraData);
        return true;
    }
}

/* A contract attempts to get the coins */
function transferFrom(address _from, address _to, uint256 _value) returns (bool success) {
    if (balanceOf[_from] < _value) throw;                 // Check if the sender has enough
    if (balanceOf[_to] + _value < balanceOf[_to]) throw;  // Check for overflows
    if (_value > allowance[_from][msg.sender]) throw;   // Check allowance
    balanceOf[_from] -= _value;                           // Subtract from the sender
    balanceOf[_to] += _value;                             // Add the same to the recipient
    allowance[_from][msg.sender] -= _value;
    Transfer(_from, _to, _value);
    return true;
}

/* This unnamed function is called whenever someone tries to send ether to it */
function () {
    throw;     // Prevents accidental sending of ether
`
```

KLINT FINLEY BUSINESS 06.18.16 04:30 AM

A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN



vs

http://www.rascal-mpl.org