OBERON

Scelta del linguaggio per lo sviluppo di Smart Contract

A.A. 2021-2022

Componenti del gruppo:

Casazza Domenico, matr. 1201136

Casonato Matteo, matr. 1227270

Chen Xida, matr. 1217780

Pavin Nicola, matr. 1193215

Poloni Alessandro, matr. 1224444

Scudeler Letizia, matr. 1193546

Stojkovic Danilo, matr. 1222399

Indirizzo repository GitHub:
https://github.com/TeamOberon07/ShopChain

Università degli Studi di Padova

Indice

1 Smart Contracts **2**

2 Linguaggi per lo sviluppo di Smart Contracts (su EVM) **2**
2.1 Solidity 2
2.2 Vyper 3
2.3 Yul e Yul+ 4
2.4 Fe 5

3 Considerazioni Finali **5**

1 Smart Contracts

Gli smart contracts sono programmi salvati all'interno di una Blockchain, eseguiti quando si verificano particolari precondizioni, quali rilascio di fondi tramite un pagamento, registrazione di un veicolo, invio di una notifica o Ticket... Vengono quindi tipicamente utilizzati per automatizzare l'esecuzione di un accordo, o contratto, appunto, senza l'intervento di terze parti, così rendendo tutti i partecipanti immediatamente certi del suo esito; oppure per automatizzare un workflow, scatenando la prossima azione da eseguire. Una volta processata e la blockchain aggiornata, la transazione risulterà immutabile ed accessibile soltanto agli aventi diritto.

I benefici sono dunque:

- Velocità, efficienza e precisione
- Fiducia e trasparenza
- Sicurezza
- Risparmio

2 Linguaggi per lo sviluppo di Smart Contracts (su EVM)

Per sviluppare smart contracts sono disponibili vari linguaggi, in particolare i più famosi ed utilizzati sono i seguenti:

- **Solidity**
- **Vyper**
- **Yul / Yul+**
- **Fe**

Solidity e Vyper sono i più utilizzati ed affermati, mentre gli altri sono più nuovi e particolari. Di seguito si analizzano uno ad uno i vari linguaggi.

2.1 Solidity

Solidity è in assoluto il linguaggio più diffuso ed utilizzato in ambito EVM, il più affermato e quello con più supporto fornito da una vasta community.

Particolarità ed aspetti favorevoli

- Linguaggio di alto livello
- Object-oriented
- Tipizzazione statica
- Simile a C++ e Java
- Molte risorse e tutorial disponibili in rete

- Buon supporto da una vasta e consolidata community
- Disponibilità di vari tools al supporto dello sviluppo, quali
  - Remix, online editor per compilare ed eseguire deploy di smart contracts
  - Truffle e Hardhat, development suites per smart contracts
- Supporto a:
  - Ereditarietà
  - Librerie
  - Tipi definiti da utente
  - Programmazione dinamica
  - Gestione delle eccezioni

Aspetti negativi e/o sfavorevoli

- Più complesso scrivere programmi senza vulnerabilità rispetto agli altri linguaggi, in particolare Vyper

2.2 Vyper

Vyper risulta essere secondo a Solidity, sia per diffusione che per età. È un linguaggio contract-oriented, fortemente ispirato a Python, che presenta varie differenze rispetto a Solidity, sia per stile che per funzionalità. I suoi principi ed obbiettivi sono: Security, Semplicity, Auditability.

Particolarità ed aspetti favorevoli

- Linguaggio di alto livello
- Contract-oriented
- Tipizzazione forte
- Simile a Python
- Possibilità di calcolare un limite superiore preciso sul consumo di gas di ogni chiamata a funzione
- Più semplice scrivere smart contracts più sicuri
- Disponibilità di vari tools al supporto dello sviluppo, quali Brownie ed Etherscan
- Controllo bounds ed overflow su Array e String, limitandone però la dimensione al momento della dichiarazione

Aspetti negativi e/o sfavorevoli

- Meno supporto offerto dalla community, in quanto il linguaggio è meno diffuso ed utilizzato

- Ancora in via di sviluppo. Varie funzionalità già presenti in Solidity non sono ancora state implementate

Per poter garantire una miglior sicurezza e raggiungere gli obbiettivi proposti, Vyper intenzionalmente non supporta varie funzionalità invece presenti in Solidity, tra cui:

- Modifiers
- Chiamate ricorsive
- dati dinamiche
- Overloading di funzioni ed operatori

Vyper afferma infatti che non intende essere al 100% un sostituto per tutto quello che può essere fatto in Solidity, ma deliberatamente non permette o rende particolarmente difficile il fare determinate cose, se lo ritiene opportuno, al fine di aumentare la sicurezza.

2.3 Yul e Yul+

Yul è un linguaggio intermedio che può essere compilato in bytecode per soddisfare i requisiti di diversi back-end. Supporta sia la EVM che Ewasm, ed è pensato per essere un denominatore comune per entrambe le piattaforme. Viene principalmente utilizzato da sviluppatori già familiari con la EVM che sono in cerca di ulteriori ottimizzazioni, per migliorare performance ed utilizzo di Gas.

Principi e obbiettivi:

1. Leggibilità dei programmi, anche se generati da un compilatore Solidity
2. Controllo di flusso semplice da capire
3. Traduzione da Yul a bytecode il più diretta possibile
4. Adeguatezza all'ottimizzazione di interi programmi

Particolarità ed aspetti favorevoli

- Permette agli sviluppatori di avvicinarsi di più alla EVM rispetto agli altri linguaggi
- Permette di migliorare il consumo di Gas.

Aspetti negativi e/o sfavorevoli

- Meno risorse e supporto disponibili
- Linguaggio complicato, richiede una familiarità con la EVM. Sconsigliato come primo linguaggio per sviluppare smart contracts

Yul+ invece è un linguaggio di basso livello molto efficiente. Si tratti di una estensione di Yul, un upgrade sperimentale con vari Quality-of-Life improvements. Viene utilizzato per ulteriori ottimizzazioni, permette di avvicinarsi ancora di più alla EVM e promette di migliorare drasticamente il consumo di Gas.

2.4 Fe

Fe è un linguaggio ancora molto nuovo, che punta ad essere la nuova generazione dei linguaggi per lo sviluppo di smart contracts, aspirando ad essere più sicuro e semplice.

Particolarità ed aspetti favorevoli

- Linguaggio di alto livello
- Tipizzazione statica
- Inspirato a Python e Rust
- Utilizza lo stesso linguaggio intermedio di Solidity (Yul)
- Limita il comportamento dinamico per garantire un costo Gas più preciso
- Presenta di libreria standard
- Punta ad essere semplice da imparare, anche per sviluppatori non familiari con la EVM

Aspetti negativi e/o sfavorevoli

- Linguaggio molto nuovo, ancora in stato di Alpha
- Scarsità di supporto, in quanto la community non è ancora molto diffusa. Sconsigliato come primo linguaggio per sviluppare smart contracts
- Disponibile soltanto per Linux e MacOS

3 Considerazioni Finali

Dopo aver visto e considerato tutti i punti critici sopra riportati, la scelta più sensata ricade tra Solidity e Vyper.
Infatti Yul richiede una familiarità con la EVM che non è posseduta dai membri del Team, mentre Fe è ancora molto, troppo nuovo. Inoltre, entrambi questi linguaggi sono scarsamente utilizzati, e la possibilità di supporto, sia in rete che fornita dall'azienda, è quasi nulla.

Tra Solidity e Vyper invece la scelta definitiva presa dal gruppo, anche sotto consiglio diretto del proponente, risulta essere Solidity. Le sue caratteristiche, tra cui la sua maggiore maturità, diffusione, e semplicità, lo rendono il più adatto all'utilizzo per i nostri scopi.