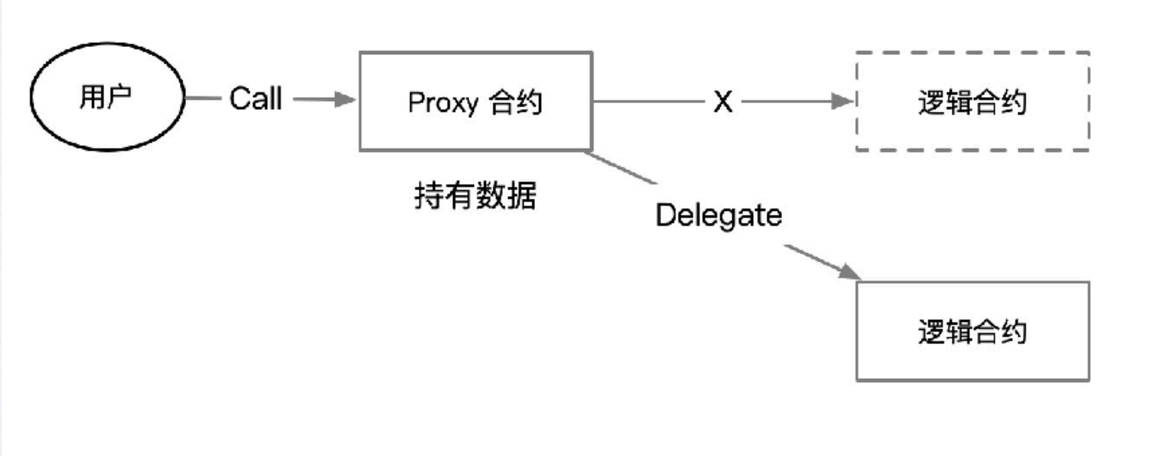
WEEK 4
DAY 4

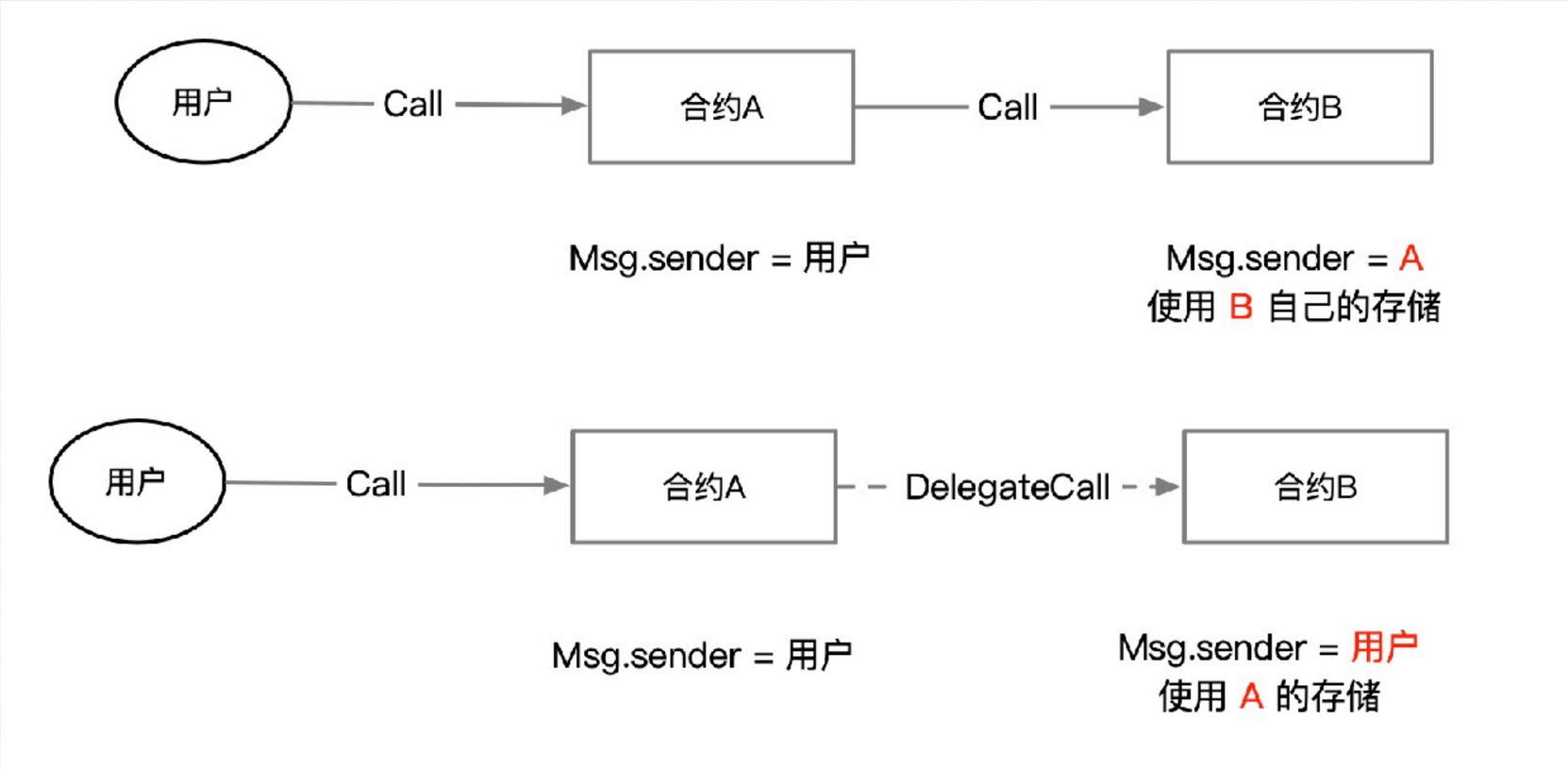
登链社区 - Tiny熊

合约升级

- 合约一旦部署，便不可更改
- 合约中有错误如何修复?
- 要添加额外功能，要怎么办?



Call 与 delegateCall 回顾



testCall_Delegate.sol

尝试给Counter升级

```
pragma solidity ^0.8.0;

contract Counter {
    uint private counter;

    function add(uint256 i) public {
        counter += 1;
    }

    function get() public view returns(uint) {
        return counter;
    }
}
```

升级尝试 1 – 使用代理

```
pragma solidity ^0.8.0;

contract CounterProxy {
    address impl;
    uint private counter;

    function add(uint256 i) public {
        bytes memory callData = abi.encodeWithSignature("add(uint256)", n);
        (bool ok,) = address(impl).delegatecall(callData);
        if(!ok) revert("Delegate call failed");

    }

    function get() public view returns(uint) {
        bytes memory callData = abi.encodeWithSignature("get()");
        (bool ok, bytes memory retVal) = address(impl).delegatecall(callData);
        if(!ok) revert("Delegate call failed");
        return abi.decode(retVal, (uint256));
    }
}
```

升级尝试 1 – 使用代理

- 问题 1：代理和逻辑合约的存储布局不一致发生无法预期的错误
- 问题 2：升级后，想添加新方法，怎么办?

CounterProxy.sol

CounterProxy

Storage	
slot	变量
0	impl
1	counter

Counter.sol

Storage	
slot	变量
0	counter

升级添加的变量，必须在末尾添加

实现地址槽位：bytes32(uint(keccak256("eip1967.proxy.implementation")) - 1)

升级尝试 2 – 统一委托

- 如何委托未来添加的函数及获取返回值?
  - fallback 统一委托
  - 用汇编魔法获取返回值

CounterFallback.sol

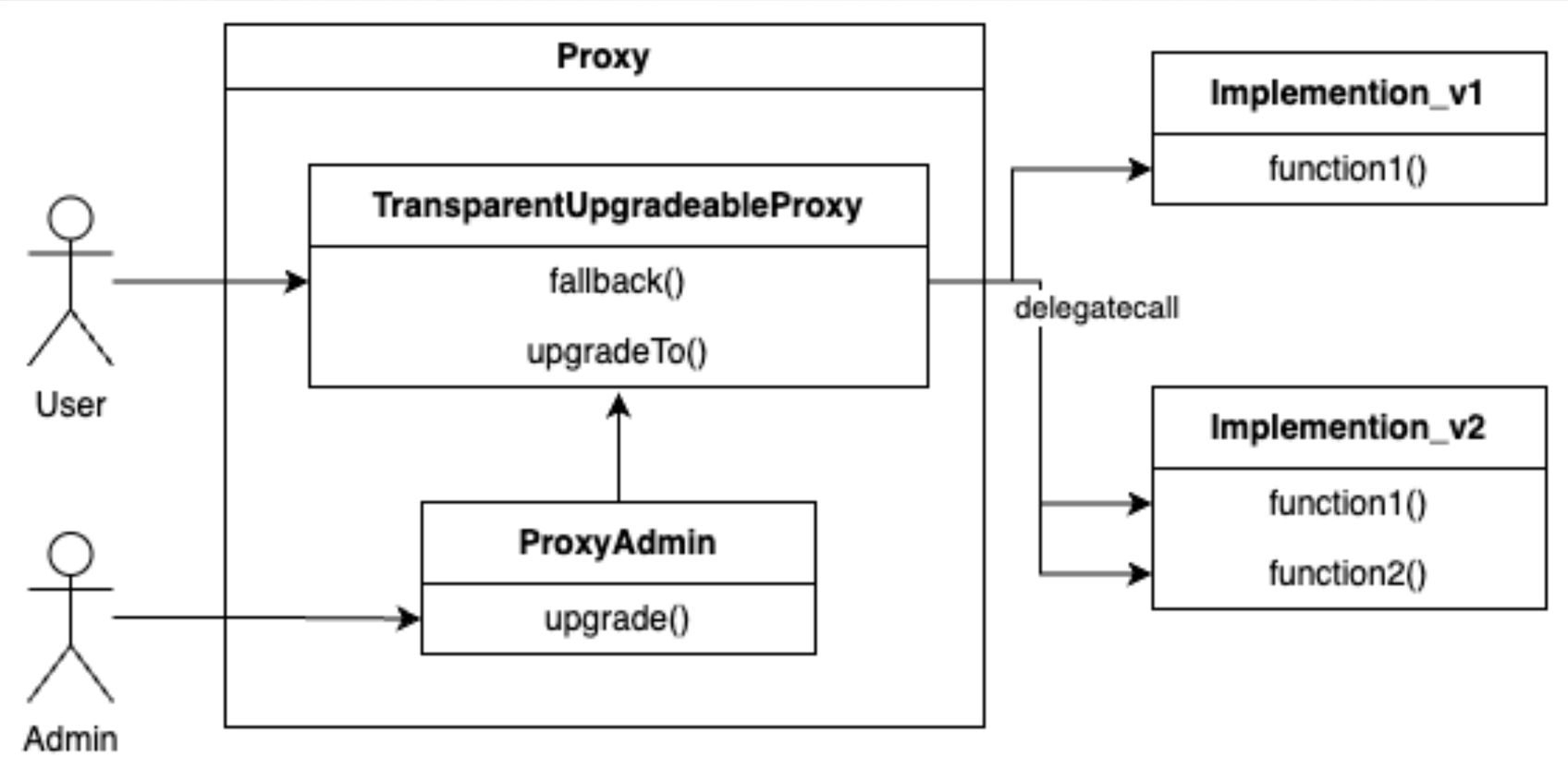
```
        assembly {
            //获得自由空闲指针
            let ptr := mload(0x40)
            //将返回值从返回缓冲去copy到指针所指位置
            returndatacopy(ptr, 0, returndatasize())

            //根据是否调用成功决定是返回数据还是直接revert整个函数
            switch success
            case 0 { revert(ptr, returndatasize()) }
            default { return(ptr, returndatasize()) }
        }
```

升级尝试 – 通用升级

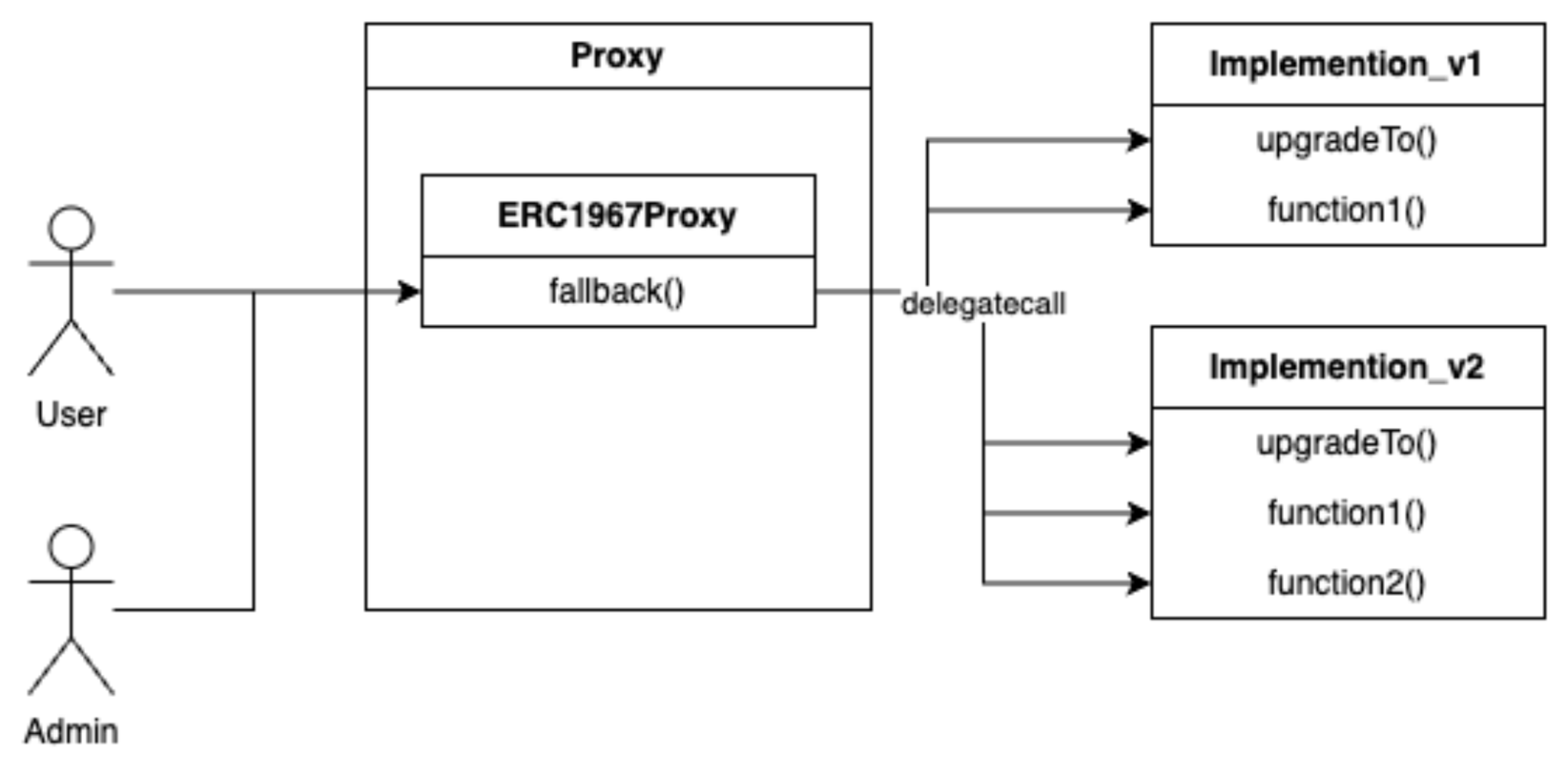
- 函数冲撞问题，若某个函数与upgradeTo() 函数选择器一样会出现什么问题?
- 透明代理（Transparent Proxy） - ERC1967Proxy
- UUPS（universal upgradeable proxy standard） - ERC-1822

透明代理



TransparentProxy.sol

UUPS



UUPSProxy.sol

使用 DelegateCall 要注意的点

- 代理和逻辑合约的存储布局需要一致。
- delegateCall 返回值
  - (bool success, bytes memory returnData) = address.delegatecall(payload);
  - Bytes 需转化为具体的类型
- 不能有函数冲撞
- 初始化问题？ - 实现合约中构造函数无效

合约升级


合约升级
重温 Call/DelegateCall
解决1. 代理和逻辑合约存储布局问题
解决2： 函数调用及返回值泛化
解决 3： 函数冲撞

开发中使用升级（Hardhat）

- hardhat-upgrades
  - npm install --save-dev @openzeppelin/hardhat-upgrades
- contracts-upgradeable
  - npm install —save-dev @openzeppelin/contracts-upgradeable

https://docs.openzeppelin.com/contracts/5.x/upgradeable

https://github.com/xilibi2003/training_camp_2/tree/main/w3_2_code

开发中使用升级Foundry

- foundry-upgrades
  - forge install OpenZeppelin/openzeppelin-foundry-upgrades
- openzeppelin-contracts-upgradeable
  - forge install OpenZeppelin/openzeppelin-contracts-upgradeable

https://github.com/OpenZeppelin/openzeppelin-foundry-upgrades

https://github.com/OpenSpace100/blockchain-tasks/tree/upgrade/w3_permit

合约的创建

SOLIDITY – 创建合约

- 创建合约的几个方法：
  - 外部部署（Remix/Hardhat/Foundry）Web3.js / Ethers.js / viem.js
  - 合约使用New 关键字（create 操作码）
  - 最小代理合约（复用代码）：
    - https://eips.ethereum.org/EIPS/eip-1167
    - https://github.com/optionality/clone-factory
  - Create2 （New + salt）
    - C c = new C{salt: _salt}();

SOLIDITY – 合约地址

- create 地址不易控制
  - 根据创建者（sender）的地址以及创建者发送过的交易数量（nonce）来计算确定
    - keccak256(rlp.encode([normalize_address(sender), **nonce**]))[12:]

- Create2 确定性的创建合约， 提前确定地址有什么好处?
    - keccak256(0xff ++ sender ++ **salt** ++ keccak256(init_code))[12:]

DEMO

CreateC.sol

```
function createContract1() public returns (address) {
    C  c = new C();
    return address(c);
}

function createContract2(address impl) public returns (address) {
    return createClone(impl);
}

function createContract3(uint _salt) public returns (address) {
    C c = new C{salt: keccak256(abi.encode(_salt))}();
    return address(c);
}
```

练习题

- 在以太坊上用ERC20 模拟铭文铸造，创建可升级的工厂合约：
  - 方法1： deployInscription(string symbol, uint totalSupply, uint perMint)
  - 方法 2： mintInscription(address tokenAddr)
  - 第 2 个版本加入铸造费用（price）

https://decert.me/quests/ac607bb0-53b5-421f-a9df-f3db4a1495f2

练习题

- 部署一个可升级的 NFT 市场合约
  - 第一版本普通合约
  - 第二版本，加入离线签名上架 NFT 功能方法（签名内容：tokenId， 价格），实现用户一次性 setApproveAll 给 NFT 市场合约，每次上架时使用签名上架。
  - 部署到测试网，并开源到区块链浏览器（在 Readme.md 中备注代理合约及两个实现的合约地址）

https://decert.me/quests/ddbdd3c4-a633-49d7-adf9-34a6292ce3a8