

# 中山大学数据科学与计算机学院本科生实验报告

## (2019 年秋季学期)

课程名称：区块链原理与技术

任课教师：郑子彬

年级	17 级	专业（方向）	软件工程
学号	17343123	姓名	吴宗原
电话	15013029107	Email	1120902512@qq.com
开始日期	12.11	完成日期	12.13

### 一、项目背景



#### 传统供应链金融：

某车企（宝马）因为其造车技术特别牛，消费者口碑好，所以其在同行业中占据绝对优势地位。因此，在金融机构（银行）对该车企的信用评级将很高，认为他有很大的风险承担的

能力。在某次交易中，该车企从轮胎公司购买了一批轮胎，但由于资金暂时短缺向轮胎公司

签订了 1000 万的应收账款单据，承诺 1 年后归还轮胎公司 1000 万。这个过程可以拉上金

融机构例如银行来对这笔交易作见证，确认这笔交易的真实性。在接下里的几个月里，轮胎

公司因为资金短缺需要融资，这个时候它可以凭借跟某车企签订的应收账款单据向金融结构

借款，金融机构认可该车企（核心企业）的还款能力，因此愿意借款给轮胎公司。但是，这

样的信任关系并不会往下游传递。在某个交易中，轮胎公司从轮毂公司购买了一批轮毂，但由于租金暂时短缺向轮胎公司签订了 500 万的应收账款单据，承诺 1 年后归还轮胎公司 500 万。当轮毂公司想利用这个应收账款单据向金融机构借款融资的时候，金融机构因为不认可轮胎公司的还款能力，需要对轮胎公司进行详细的信用分析以评估其还款能力同时验证应收账款单据的真实性，才能决定是否借款给轮毂公司。这个过程将增加很多经济成本，而这个问题主要是由于该车企的信用无法在整个供应链中传递以及交易信息不透明化所导致的。

### 区块链+供应链金融：

将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

### 实现功能：

功能一：实现采购商品—签发应收账款 交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。

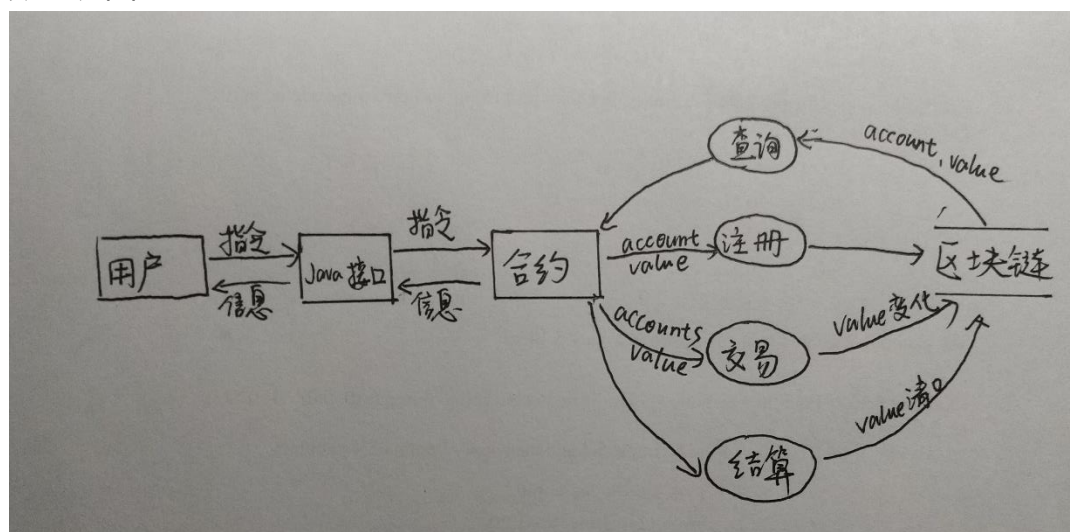
功能二：实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。

功能三：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。

功能四：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

## 二、 方案设计

### 数据流图：



最开始的想法是在数据库中再建立一个表 DebtsTable，存储核心企业下放的应收账款。这样四个功能就可以这样实现：

1. 给下游企业在 DebtsTable 中增加 Debts 金额；
2. 下游企业之间的 Debts 交易在 DebtsTable 中进行；
3. 向银行申请融资，以 Debts+Value 为基准；
4. 银行支付所有 Debts。

这样四个功能能够实现，但是在之后和同学的交流中被告知由于现金的交易在这个供应链中是不被作为核心来交互的，所以可以用应收账款来作为流通的“货币”，即四个功能的实现能够用如下方式：

1. 核心企业下放应收金额给下游企业；
2. 应收金额作为货币流通在下游企业；
3. 应收金额流通至银行，相当于银行给予现金给下游企业，即为融资；
4. 结算应收金额，核心企业的外债回收，其余下游企业和流通至银行融资的应收金额归零。

核心代码：

1. 智能合约中清除下游和银行中的应收账款：

```
35     function settlement() public returns(int256){
36         int256 ret_code = 0;
37         // 打开表
38         Table table = openTable();
39
40         for(uint i = 0; i < accountDebts.length;i++)
41         {
42             Entry entry = table.newEntry();
43             entry.set("account", accountDebts[i]);
44             entry.set("asset_value", int256(0));
45             int count = table.update(accountDebts[i], entry, table.newCondition());
46             if (count == 1) {
47                 // 成功
48                 ret_code = 0;
49             } else {
50                 // 失败
51                 ret_code = -2;
52             }
53         }
54         emit SettlementEvent(0);
55         return 0;
56     }
```

2. java 后端中命令行执行 settlement 时调用的函数

```

197 public void settlementAllAccount() {
198     try {
199         String contractAddress = loadAssetAddr();
200         Asset asset = Asset.load(contractAddress, web3j, credentials, new StaticGasProvider(gasPrice, gasLimit));
201         TransactionReceipt receipt = asset.settlement().send();
202         List<SettlementEventResponse> response = asset.getSettlementEventEvents(receipt);
203         if (!response.isEmpty()) {
204             if (response.get(0).ret.compareTo(new BigInteger("0")) == 0) {
205                 System.out.printf(" settlement success! \n");
206             } else {
207                 System.out.printf(" settlement failed, ret code is %s \n",
208                     response.get(0).ret.toString());
209             }
210         } else {
211             System.out.println(" event log not found, maybe transaction not exec. ");
212         }
213     } catch (Exception e) {
214         // TODO Auto-generated catch block
215         // e.printStackTrace();
216
217         logger.error(" registerAssetAccount exception, error message is {}", e.getMessage());
218         System.out.printf(" register asset account failed, error message is %s\n", e.getMessage());
219     }
220 }

```

### 三、 功能测试

创建 t1: 核心企业, t2: 交易下游企业, t3: 银行 (更下游企业)

```

fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh register t1
100
register asset account success => asset: t1, value: 100
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh register t2
0
register asset account success => asset: t2, value: 0
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh register t3
0
register asset account success => asset: t3, value: 0

```

功能一：实现采购商品—签发应收账款 交易上链

```

fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh transfer t1
t2 10
transfer success => from_asset: t1, to_asset: t2, amount: 10

```

实现应收账款的转让上链 (利用应收账款向银行融资上链)

```

fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh transfer t2
t3 5
transfer success => from_asset: t2, to_asset: t3, amount: 5

```

应收账款支付结算上链, 应收账款归零 (代码中如果把添加 accountDebts 账户放在交易目标中, 则核心企业 value 不会归零)。

```

fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh settlement
settlement success!

```

检查应收账款是否归零。

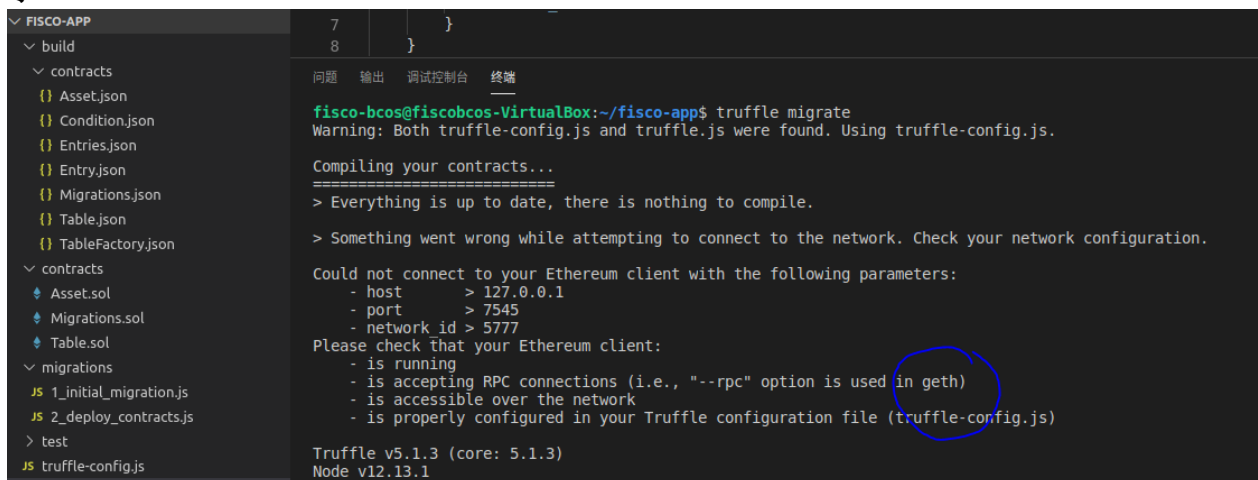
```

fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh query t2
asset account t2, value 0
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh query t3
asset account t3, value 0
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh query t1
asset account t1, value 0

```

## 四、 界面展示

原打算用 truffle 编写前后端的，但是后来发现 FISCO 链端应该是不支持的，所以就搁浅了。



```
FISCO-APP
├── build
│   ├── contracts
│   │   ├── Asset.json
│   │   ├── Condition.json
│   │   ├── Entries.json
│   │   ├── Entry.json
│   │   ├── Migrations.json
│   │   ├── Table.json
│   │   └── TableFactory.json
│   ├── contracts
│   │   ├── Asset.sol
│   │   ├── Migrations.sol
│   │   └── Table.sol
│   └── migrations
│       ├── 1_initial_migration.js
│       └── 2_deploy_contracts.js
└── test
    └── truffle-config.js

问题 输出 调试控制台 终端

fisco-bcos@fiscobcos-VirtualBox:~/fisco-app$ truffle migrate
Warning: Both truffle-config.js and truffle.js were found. Using truffle-config.js.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

> Something went wrong while attempting to connect to the network. Check your network configuration.

Could not connect to your Ethereum client with the following parameters:
- host      > 127.0.0.1
- port      > 7545
- network_id > 5777
Please check that your Ethereum client:
- is running
- is accepting RPC connections (i.e., "--rpc" option is used in geth)
- is accessible over the network
- is properly configured in your Truffle configuration file (truffle-config.js)

Truffle v5.1.3 (core: 5.1.3)
Node v12.13.1
```

## 五、 心得体会

这次大作业还是做得很艰难的，首先是由于自身的能力不足，对于框架了解的不多，没有什么经验，在网上查找区块链使用的框架的时候，找到了在以太坊中应用广泛的 truffle 框架，因为当时想的是 FISCO 应该也支持，但是结果是不能。浪费了大量的时间之后，重新回到文档上，才发现原来官方文档中有完整介绍的只有 Java，且已经给出了示例工程以及 SDK 配置方法，以及接口的调用方法，绕了弯路之后才回到正轨。之后编写智能合约的时候，一开始想的在数据库中再建立一个表 DebtsTable，存储核心企业下放的应收账款去实现四个功能，之后实现的时候发现比想象的复杂许多，在问了同学之后，得知由于现金的交易在这个供应链中是不被作为核心来交互的，所以可以用应收账款来作为流通的货币，简化了合约的实现，但是最后还是没能完整地这次作业，停留在了后端和链端刚完成的阶段，还是有很大遗憾的，应该早一些开始就不会这么紧迫。总之，FISCO 作为国产的区块链底层技术，虽然走在了前列，但是感觉还是有很多不太完善的地方，也没有一个开发生态，导致除了官方文档没有其余可以纠错学习的地方，开发起来并不是很友好，希望 FISCO 能够渐渐地完善起来吧。