CST4125: Blockchain Development
Week: 8
Title: Remove and Trader

Dr Ian Mitchell

Bedfordshire,
UK

2023

Lecture Objectives

Knowledge

- Trade
- Remove
- CLI composer

Disclaimer

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

Contact and Office Hours

Contact Details

- Name: Dr Ian Mitchell
- Room: TG10
- Address: Middlesex University, Computer Science, London, NW4 4BT
- email: smerf.net

Contact and Office Hours

Contact Details

- Name: Dr Ian Mitchell
- Room: TG10
- Address: Middlesex University, Computer Science, London, NW4 4BT
- email: smerf.net

Office Hours

- During term time only
- When: Autumn Term: Mondays 1100-1300hrs
- Please read notifications or emails
- There are occassions that these could be arranged online, e.g., due to industrial action or inclement weather

Deadlines

				Feedback	
Description	Submission	Weight	Deadline	Formative	Summative
1. Hyperledger	MyLearning	50%	18th December 2022	LW11-12	12/01/2023
2. Ethereum	MyLearning	50%	2nd April 2023	LW23-24	24/04/2023
Resits	MyLearning	50-100%	1st July 2023	None	None
Deferals	MyLearning	50-100%	1st July 2023	None	None

Module Feedback

Module Survey

Learn from Mistakes

Mistakes

We can learn a lot from bad design, as long as we don't emulate it. Sometimes it is necessary to make mistakes in order to learn. Here we look at implementation of Arrays of items in registries. Warning: Data duplication is a bad thing. It creates inconsistencies. It causes high maintenance code. Data redundancy is a good thing. It ensures consistency and causes low maintenance code. **Data redundancy should be used at all times.**

Bad Examples

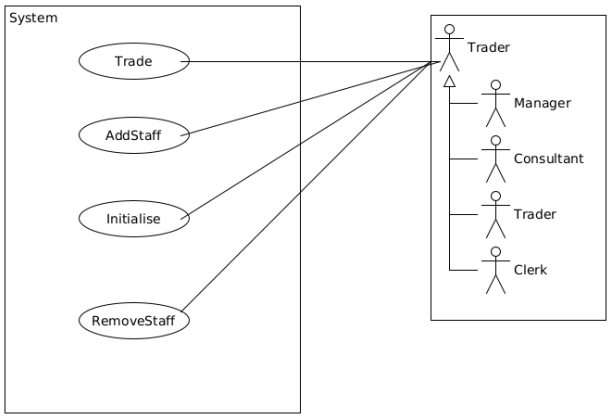
- Trader example
- keep tabs on trader commodities
- restricted view
- add trader
- remove trader

Problem Definition

- Remove Staff from Trader Network
- Remove Staff from Trader Registry
- Update assets with nominated member of staff
- Update nominated member of staff asset ownership

Trader

Use Case Diagram



Remove

Current State

Remove

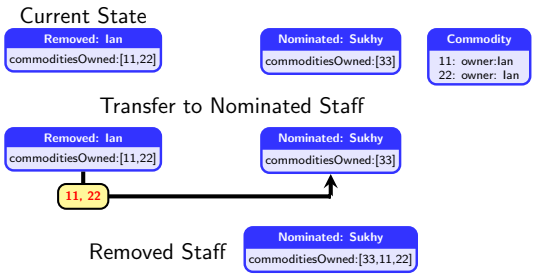
Current State

Remove

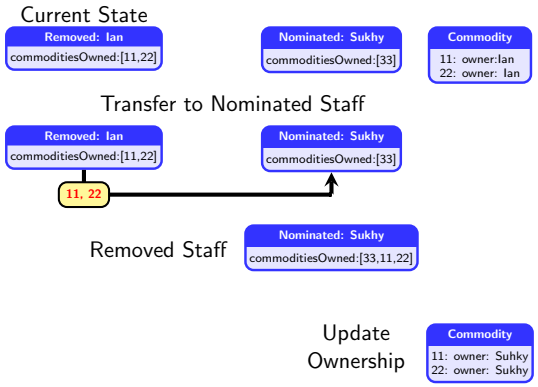
Current State

Transfer to Nominated Staff

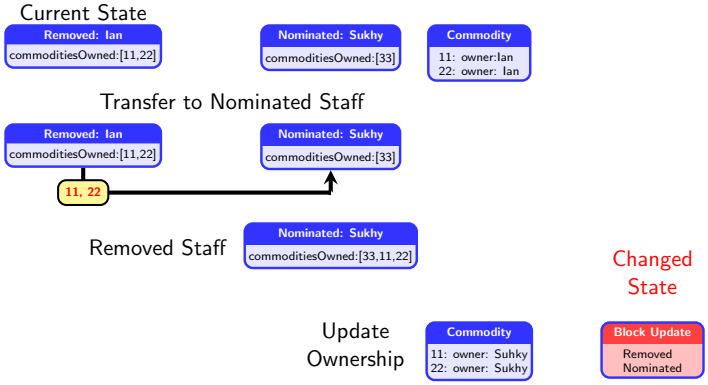
Remove



Remove



Remove



Reflections

- Can this transaction be completed with a Database?

Reflections

- Can this transaction be completed with a Database?
- Why use blockchain?

Reflections

- Can this transaction be completed with a Database?
- Why use blockchain?
- Immutability - append-only ledger

Reflections

- Can this transaction be completed with a Database?
- Why use blockchain?
- Immutability - append-only ledger
- Tamper resistant

Reflections

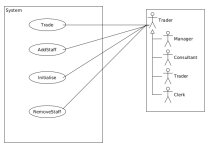
- Can this transaction be completed with a Database?
- Why use blockchain?
- Immutability - append-only ledger
- Tamper resistant
- Non-repudiation

CTO I
Assets

```
11 asset Commodity identified by tradingSymbol {
12   o String tradingSymbol
13   o String description
14   o Double quantity
15   --> Trader owner
16 }
```

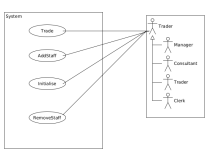

CTO I
Participants

```
17 participant Trader identified by tradeId {
18   o String tradeId
19   o String firstName
20   o String lastName
21   o Grade grade
22   o String[] commoditiesOwned
23 }
```



CTO I
Transactions

```
4  transaction Trade {
5    --> Commodity commodity
6    --> Trader newOwner
7  }
8  transaction removeStaff {
9    -->Trader removedStaff
0    -->Trader nominatedStaff
1  }
2  transaction initialise{}
3  transaction initialiseAll{}
4  transaction removeAll{}
```



Permissions I

```
1 rule traderToSubmitTX{
2   description: "Any trader can submit a transaction"
3   participant: "org.trader.net.Trader"
4   operation: ALL
5   resource: "org.trader.net.*"
6   action: ALLOW
7 }
```

JS - Trade Commodity I

```
var ns='org.trader.net';
/**
 * transaction of a commodity from one trader to another
 * @param {org.trader.net.Trade} tx - tx to be processed
 * @transaction
 */
async function tradeCommodity(tx) {
    let buyerAndSeller = new Array();
    let traderReg = await getParticipantRegistry(ns+'.Trader');
    let exist = await traderReg.exists(tx.newOwner.getIdentifier());
    //test if buyer exists
    if (exist){
  //buyer
      let buyer = await traderReg.get(tx.newOwner.getIdentifier());
    //seller
      let seller = await traderReg.get(tx.commodity.owner.getIdentifier());
      // add to the buyer
      buyer.commoditiesOwned.push(tx.commodity.getIdentifier().toString());
      buyerAndSeller.push(buyer);
      //remove from the seller
      let needle = tx.commodity.getIdentifier().toString();
      let haystack = seller.commoditiesOwned;
      let filteredHaystack = haystack.filter((item)=>item!=needle);
      seller.commoditiesOwned = filteredHaystack;
      buyerAndSeller.push(seller)
      //updateAll traders
      await traderReg.updateAll(buyerAndSeller);
      //update the commodity
      tx.commodity.owner = tx.newOwner;
      let commodityReg = await getAssetRegistry(ns+'.Commodity');
```


JS - Trade Commodity II

```
      await commodityReg.update(tx.commodity);
    } else
      throw new Error('Buyer does not exists');
}
```


JS - Remove I

```
/**
 * transaction to remove staff
 * @param {org.trader.net.removeStaff} staff leaving
 * @transaction
 */
async function removeStaff(tx){
  let traderReg = await getParticipantRegistry(ns+'.Trader');
  let selectedTrader = await traderReg.get(tx.nominatedStaff.getIdentifier());
  let removedTrader = await traderReg.get(tx.removedStaff.getIdentifier());
  let commodityReg = await getAssetRegistry(ns+'.Commodity');
  let commodityRegAll = await commodityReg.getAll();
  let commodityArray = new Array();
  removedTrader.commoditiesOwned.forEach(item =>selectedTrader.commoditiesOwned.push(item
      ));
  await traderReg.update(selectedTrader);
  commodityRegAll.forEach( function (item) {
    if(item.owner.getIdentifier() == tx.removedStaff.getIdentifier()){
      let factory = getFactory();
      let newCommodity = factory.newResource(ns, 'Commodity', item.getIdentifier());
      let ownerRel = factory.newRelationship(ns, 'Trader', tx.nominatedStaff.
     getIdentifier());
      newCommodity.owner = ownerRel;
      newCommodity.description = item.description;
      newCommodity.quantity = item.quantity;
      commodityArray.push(newCommodity);
      }
    });
  await traderReg.remove(removedTrader);
  await commodityReg.updateAll(commodityArray);
}
```


JS - Remove II

, basicstyle=

JS - Remove I

```
async function removeStaff(tx){
    let traderReg = await getParticipantRegistry(ns+'.Trader');
    let selectedTrader = await traderReg.get(tx.nominatedStaff.getIdentifier());
    let removedTrader = await traderReg.get(tx.removedStaff.getIdentifier());
    let commodityReg = await getAssetRegistry(ns+'.Commodity');
    let commodityArray = new Array();
    removedTrader.commoditiesOwned.forEach(async function(item) {
            selectedTrader.commoditiesOwned.push(item);
            let newCommodity = await commodityReg.get(item.toString());
            newCommodity.owner = tx.nominatedStaff;
            commodityArray.push(newCommodity);
        });
    await traderReg.update(selectedTrader);
    await traderReg.remove(removedTrader);
    await commodityReg.updateAll(commodityArray);
}
```

JS - Initialise I

```
/*
* @param {org.trader.net.initialise} no param
* @transaction
*/
async function initialise(){
  let ids = ['111', '222', '333'];
    let firstNames = ['Ian', 'Sukhvinder', 'Xiaochun'];
        let lastNames = ['Mitchell', 'Hara', 'Cheng'];
  let grades = ['Manager', 'Trader', 'Trader'];
    let staff = new Array();
      for(let i=0; i<ids.length; i++){
        let factory = getFactory();
          let newStaff = factory.newResource(ns, 'Trader', ids[i]);
          newStaff.firstName = firstNames[i];
          newStaff.lastName = lastNames[i];
          newStaff.grade = grades[i];
          newStaff.commoditiesOwned = new Array();
          staff.push(newStaff);
      }
      let traderReg = await getParticipantRegistry(ns+'.Trader');
    await traderReg.addAll(staff);
```


JS - Initialise All I

```
/*
* @param {org.trader.net.initialiseAll} no param
* @transaction
*/
async function initialiseAll(){
  //values to populate staff
  let ids = ['1000', '2000', '3000'];
    let firstNames = ['Mohamed', 'Pauline', 'Sharon'];
  let lastNames = ['Sherrif', 'Tyler', 'Brown'];
      let grades = ['Manager', 'Consultant', 'Trader'];
  //values to populate commodities
    let commodityIDs= ['51', '52', '53', '54', '55'];
      let commodityOwner = ['1000', '2000', '2000', '3000', '1000'];
    let commodityDesc = ['tea', 'coffee', 'milk', 'chocolate', 'grain'];
    let commodityQuan = [100.0, 140, 200.0, 55.5, 400];
  // arrays for staff and commodities - push new item every iteration and then addAll to
     registry
    let staff = new Array();
    let commodities = new Array();
    //have to update staff first since there is a relationship of owner in commodity
      for(let i=0; i<ids.length; i++){
        let factory = getFactory();
          let newStaff = factory.newResource(ns, 'Trader', ids[i]);
          newStaff.firstName = firstNames[i];
          newStaff.lastName = lastNames[i];
          newStaff.grade = grades[i];
          newStaff.commoditiesOwned = new Array();
          let needle = ids[i];
          let haystack = commodityOwner;
          //filteredHaystack is commoditiesOwned array
```


JS - Initialise All II

```
          haystack.filter((element,index)=>{if (needle==element) newStaff.
    commoditiesOwned.push(commodityIDs[index]) });
          staff.push(newStaff);
      }
    let traderReg = await getParticipantRegistry(ns+'.Trader');
    await traderReg.addAll(staff);
    commodityIDs.forEach( (commodityValue, commodityIndex) => {
      let factory = getFactory();
      let newCommodity = factory.newResource(ns, 'Commodity', commodityValue);
      let ownerRel = factory.newRelationship(ns, 'Trader', commodityOwner[commodityIndex
    ]);
        newCommodity.owner = ownerRel;
        newCommodity.description = commodityDesc[commodityIndex];
        newCommodity.quantity = commodityQuan[commodityIndex];
        commodities.push(newCommodity);
      });
    let commodityReg = await getAssetRegistry(ns+'.Commodity');
    await commodityReg.addAll(commodities);
}
/*
```


JS - Remove All I

```
/*
 * @param {org.trader.net.removeAll} no param
 * @transaction
 */
async function removeAll(){
  let traderReg=await getParticipantRegistry(ns+'.Trader');
  let comReg=await getAssetRegistry(ns+'.Commodity');
  let traderAll=await traderReg.getAll();
  let comAll=await comReg.getAll();
  await traderReg.removeAll(traderAll);
  await comReg.removeAll(comAll);
}
```


Code Summary

- Await
- Var/Let
- Array methods
- Factory
- Update / UpdateAll
- Add / AddAll
- Get / GetAll
- Asset / Participant Registry
- **Remove**
- Arrow functions

YO

- yo command can create basic network files
- `yo hyperledger-composer:businessNetwork`
- Composer project generator
- Creates:
  - `package.json`
  - `README.md`
  - `models/file.cto`
  - `lib/logic.js`
  - `permissions.acl`

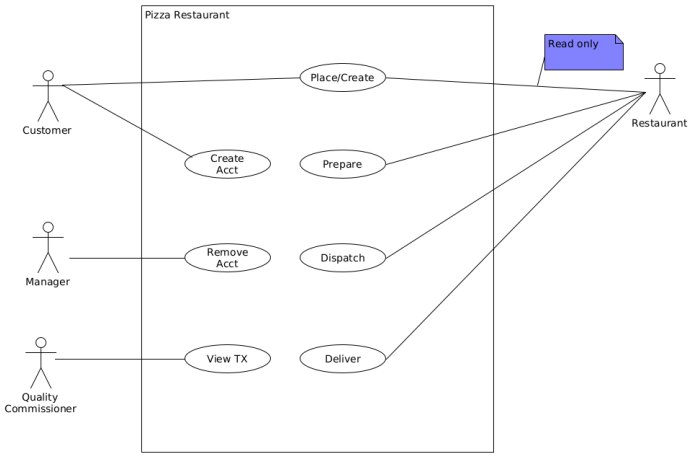
Package

- `composer archive create -t dir -n .`
- create a business network archive
- upload to web interface

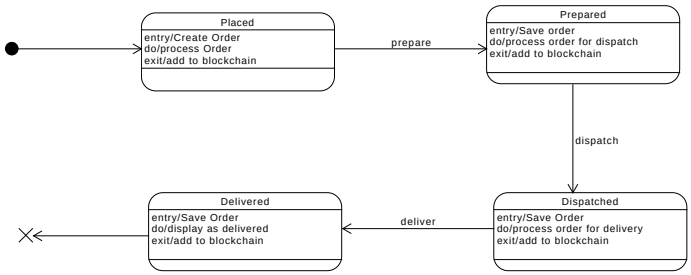
Exercise

- Complete the trader example
- Commodity
- Trader
- Enum
- Transactions
  - Initialise all
  - Initialise
  - Trade
  - Remove trader

Pizza



Pizza State



Problem Definition: 12%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
Problem Definition, PD (12%)	Specification	No Spec.	Spec., present	Spec. is not conducive to BC.	Unrelated or missing spec. components	Spec. conducive to BC, all components explained and coherent	1	/4
	Flowchart, FC	No use of FC in [4]	FC applied, no explanation.	All components of FC applied, some explanation.	All components of FC applied correctly but does not match spec/UCD.	All components of FC applied correctly and matches spec/UCD	1	/4
	Use Case Diagram, UCD.	No UCD	Incoherent UCD	Misaligned UCD and PD. Assumptions left uncommented	No include or extend relationships. Assumptions commented	Aligned and complete UCD with comments and assumptions	1	/4

Data Modelling: 16%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
Data Model (16%)	Participants	No participants	Lacking and/or incorrect participants. Incorrect data types used. Unidentified.	Irrelevant participants. Correct identification. Lacking any assumptions. Opportunities to use more appropriate data types missed.	Participants lacking UCs and incomplete assumptions. Structurally sound.	Correct participants, data structures, assumptions and matching UCs	1	/4
	Assets	No assets	Lacking and/or incorrect assets	Irrelevant assets. No enum or concepts.	Assets unrelated to participants or no assets with the capability of state change	Some of the assets must at least be 3 of the following: have a state capable of change, relevant, complete and related to participants	1	/4
	Transactions, TX	No TX	Vague TX	TX not updating state	TX without ownership	participant specific TX	1	/4
	Comments	No comments	Auto-generated comments only (headers only), no clarifying comments	Vague, incorrectly placed and/or unexplanatory comments	Explanatory and identifiable comments, but incomplete. Too verbose and high comment to code ratio	Complete, concise and succinct comments	1	/4

Access Control Language: 12%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
Access Control Language, ACL, (12%)	Participants	No ACL. Basic ACL, admin access only & automatically generated code	ACL has too few rules	ACL has contradictions or allows unauthorised access to transactions or assets. There is no difference between participant access	ACL order is incorrect	ACL is implemented correctly	1	/4
	Ordering, Comments and listing	No listing or basic ACL, admin access only & automatically generated code	Syntax errors for ACL.	Rules are disorganised and need re-ordering. Inclusion of commented out rules	Rules are in correct order, but lack ideal names, descriptor values and comments. No line numbers.	Correct order and appropriate names, descriptors values and comments	1	/4
	Conditions	Auto-generated rules only. Admin access to all.	No conditions and simple rules only	Conditions applied incorrectly.	Identifier conditions applied correctly	Conditions to check status or lists and of a higher order of difficulty	1	/4

Business Logic: 32%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
Business Logic (32%)	Queries	No Queries	Queries but don't execute	Irrelevant Queries	Relevant Queries without relationships	Relevant Queries with relationships	1	/4
	Transactions	No Transactions	BL - run time execution	BL code accessing assets and participants, with no restriction, or comments directing to ACL	BL code accessing TX with restrictions, but not acknowledged	Acknowledged rules and restrictions and code accessing both assets and participants correctly	2	/8
	API	No use of promises	BL code not executing	BL code duplicating ACL	No extensive use of API and promises	Extensive use of API and Promises and complexity used to aid the update of state correctly	3	/12
	Initialise	No initialisation or automatic population of values in registry	Initialisation present but not working	Initialisation only partial, e.g., only completes assets and not participants	All assets and participants populated but incorrectly, e.g., data is misaligned	All assets and participants populated correctly	1	/4
	Comments	No comments	Non-explanatory comments	Partial explanatory comments	Overly commented	Fully explanatory comments	1	/4

Presentation: 12%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
Presentation (12%)	Slide Content	No slides	Incoherent presentation and not demonstrating the understanding of the coursework. Cluttered and/or illegible slide content	Coherent but poor content coverage. Less than 5 mins in length. Uncluttered. Some illegible slide content, especially screenshots	Less than 9 mins or greater than 10 mins. Clear figures and screenshots. Coherent but not explaining all points required	Between 9-10 mins in length, clear and readable slides and addresses all items	1	/4
	Transaction, TX	No Demonstration	Demonstration of successful TX	Demonstration of unsuccessful TX due to ACL	Demonstration of unsuccessful Demonstration due to BL	All demonstrations completed	1	/4
	Structure	No structure	No headers *and* footers, slide numbers	No headers *or* footers, slide numbers	Headers, Footers and numbers but incorrect	All slides consistent with correct information in headers and footers.	1	/4

Documentation: 8%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
Report (8%)	English	Many sentences rendered nonsensical and many misspellings	Some sentences rendered nonsensical and a few misspellings	Sentences with poor grammar, written in first or second person, and a few misspellings	Good grammar, not written in third person. A few grammatical and spelling mistakes.	Written in third person. A few grammatical or spelling mistakes	1	/4
	Template	No structure followed	No numbering but structure present	Incorrect frontmatter or backmatter, but main matter correct structure. No figure, listing or table captions.	No citations or references, or incorrect bibliography style applied	Correct template, citations/references, numbering and template compliance.	1	/4

Business Network Archive: 8%

Criteria	Sub-criteria	0	1	2	3	4	W	Σ
BNA (8%)	Execution	Errors	Run-time errors	No errors (4)			1	/4
	BNA format	None	ACL	Node.js	CTO	Structure	1	/4

References I

[1] The Linux Foundation. *Hyperledger Architecture, Volume 1*. hyperlink. [Accessed: Jan 2021]. 2017.

[2] The Linux Foundation. *Hyperledger Architecture, Volume 2*. hyperlink. [Accessed: Jan 2021]. 2018.

[3] Nitin Gaur et al. *Hands-on Blockchain with Hyperledger: Building Decentralised Applications with Hyperledger Fabric and Composer*. Packt, 2018. ISBN: 9781788994521.

[4] Dylan Yaga et al. *Blockchain technology overview*. Tech. rep. National Institute of Standards and Technology, 2018.

Web Resources

- http://hyperledger.org
- https://nodejs.org
- https://hyperledger.github.io/composer/latest/api/runtime-factory
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://github.com/hyperledger/composer-sample-networks
- https://hyperledger.github.io/composer/latest/business-network/bnd-create