University of
South Australia

INFS 2044

Workshop 2b Answers

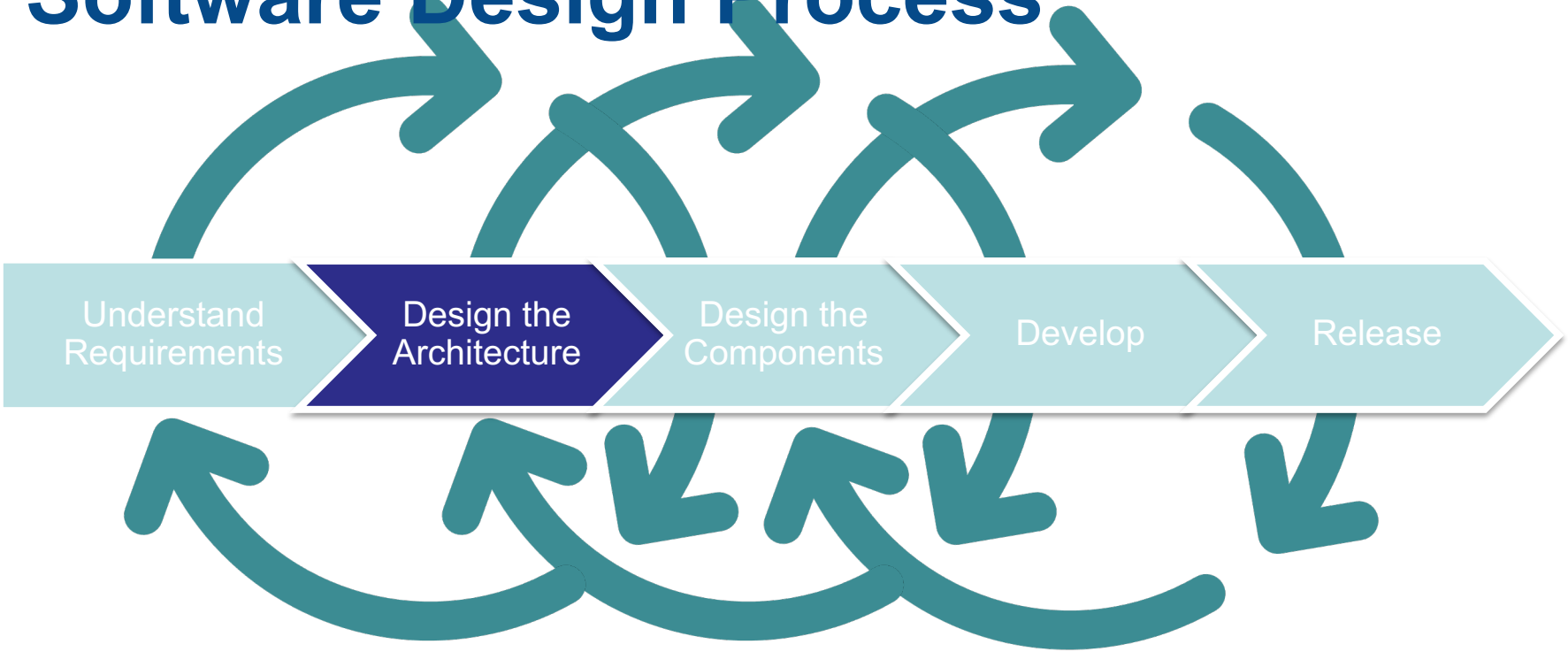
Preparation Already Done

- Read the required readings for this week
- Read and bring a copy of the *Stock Trading System Requirements* to the workshop
- Bring a copy of the workshop instructions (this document) to the workshop

Where We Are At

- Validated requirements and use cases (Week 1)
- Introduction to volatility-based decomposition
- Compositional design to realise use cases

Software Design Process



Learning Objectives

- Apply volatility-based architecture design to complex requirements

Task 1. Assess Decomposition

- Read the *Stock Trading System* case study.
- Discuss potential volatility related to this system.
- What changes in the system and its environment may affect the design?

Stock Trading Use Cases (1)

- The system should enable *in-house traders* to:
  - Buy and sell stocks
  - Schedule trades
  - Issue reports
  - Analyse the trades

Stock Trading Requirements (2)

- Users submit request reports and trades via a web browser
- The system confirms requests and delivers information via email to the users
- Data should be stored in a local database.

Categories of Volatility

- User
- Client application
- Security
- Notification
- Storage
- Connection & Synchronisation
- Duration and device
- Workflow
- Locale
- Regulations
- …

Stock Trading Volatility – User

- Traders serve clients (by maintaining their portfolios)
- Shall customers be able to view the portfolio and trades directly?
- System administrators?

Stock Trading Volatility – Client App

- User volatility often manifests in the type of client application & technology
- Web page sufficient for clients viewing their account
- Some may want to view on a mobile device
- Traders may need a multi-monitor, rich desktop application showing trends, account details, market tickers, newsfeeds, projections, …

Stock Trading Volatility – Security

- Volatility in users implies volatility in how the users authenticate themselves
- There may be few in-house traders, but millions of clients
- For staff, domain authentication is fine
- For internet users, username & password may be sufficient
  - Maybe 2FA
- Authorization is also volatile

Stock Trading Volatility – Notification

- Requirements are not clear
  - What if the email bounces?
  - Send a letter, text message, fax instead?
  - This is a solution masquerading as a requirement
- Real requirement is to notify users
  - The notification mechanism is volatile
- Also volatility on who receives the notifications
  - Clients may prefer email
  - Their accountants may prefer a paper document

Stock Trading Volatility – Storage

- Local Database is a solution, not a requirement.
- Real requirement is **reliable data persistence**
  - System must not lose data
  - Most users will only *read* data
  - Persistence mechanism is volatile (local storage vs cloud; relational vs nosql; disk vs in-memory cache, etc)

Stock Trading Volatility – Connection

- Connectivity and Synchronicity are volatile
- Requirements demand a connected, synchronous, sequential manner of completing a web form and submitting it in-order
  - Traders may want to handle multiple trades concurrently
  - Consider asynchronous submission, out-of-order processing

Stock Trading Volatility – Duration

- Duration of interaction is volatile
  - Some are short, others are long
  - Traders use complicated trades to make money, can take several days to setup the trades
  - Long-running interactions may span several system sessions and devices

Stock Trading Volatility – Trade Items

- Clients may want to trade
  - Stocks
  - Commodities
  - Bonds
  - Currencies
  - Futures

Stock Trading Volatility – Workflow

- Steps for trading different items can be volatile
  - Stock trading works quite differently from trading commodities
  - Trade analysis is done differently from trading items
- The workflow is volatile

Stock Trading Volatility – Locale

- System may be deployed in different regions
  - Trading rules differ
  - Language differs
  - Taxation, regulatory compliance requirements differ

Stock Trading Volatility – Feed

- Source of market data could change over time
- Feeds have a different format, cost rate, and communication protocols
- Feeds can be external or internal to the cmopany

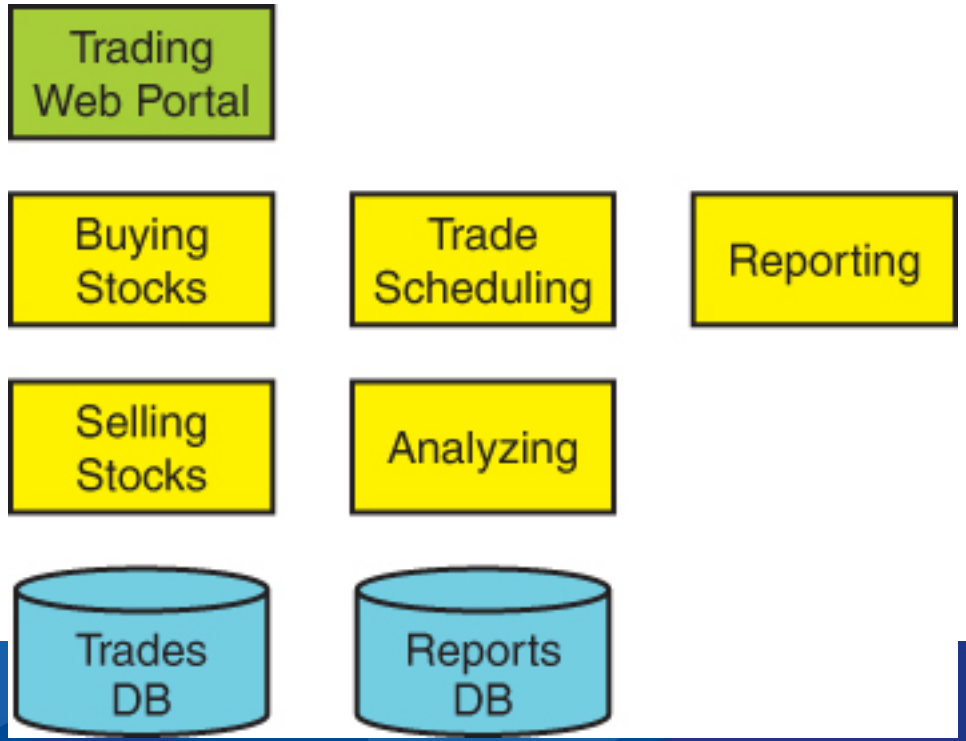
Task 2. Assess Decomposition

- Read the *Stock Trading System* case study on the course site.
- Examine the decomposition given on the next slide.
- Discuss advantages and disadvantages of this design.
- How would changed requirements affect the design?

Recall Stock Trading Use Cases

- Buy and sell stocks
- Schedule trades
- Issue reports
- Analyse the trades

Stock Trading System: Design 1



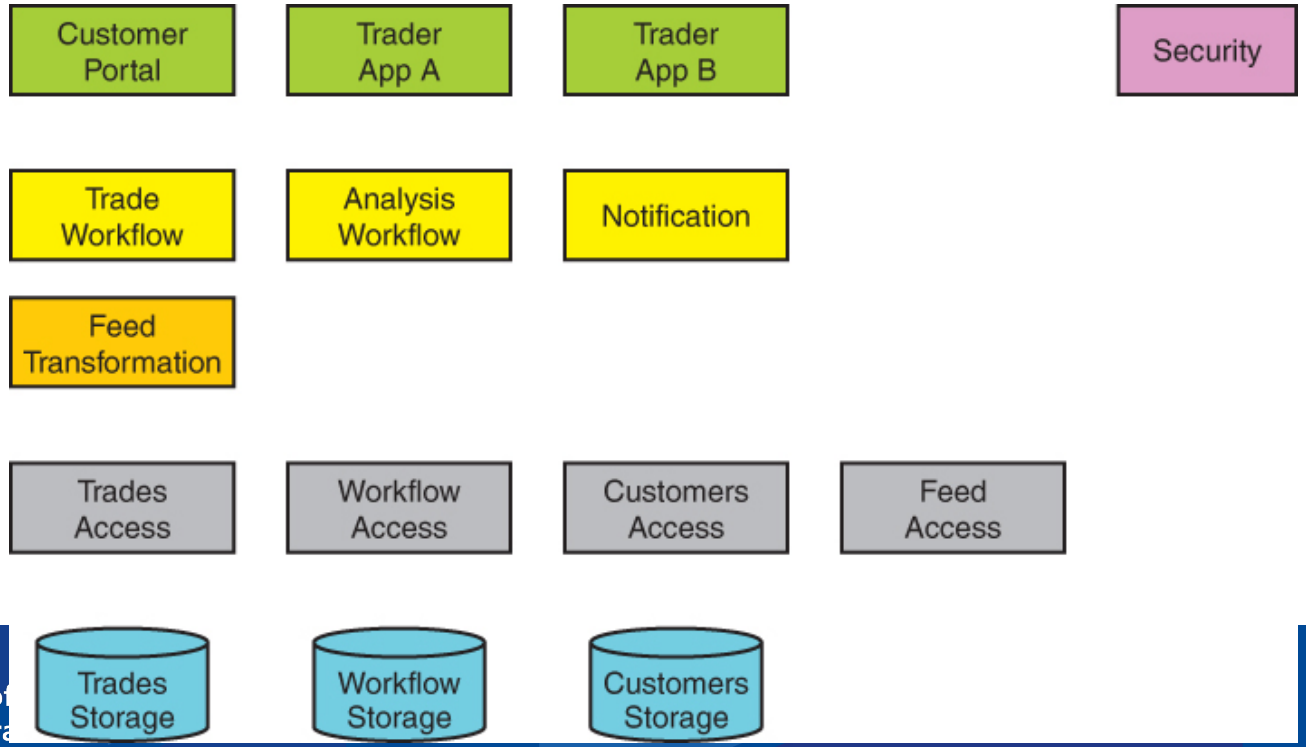
Discussion

- The Client orchestrates use cases
- The business logic resides in the portal
- Volatility in Client Application, Notification, Storage, etc not encapsulated
- Change of interaction to asynchronous not easily possible
- Branching out in trading other financial instruments not easily possible
- Localization not easily possible
- Connecting to new feeds not easily

Task 3: Component Design

- Create a decomposition for the Stock *Trading System* that accounts for the identified volatilities.
- Show how the volatilities map to components.
- Identify strengths and weaknesses of the decomposition.
- Does it isolate change and promote evolution and reuse?

Stock Trading System Decomposition



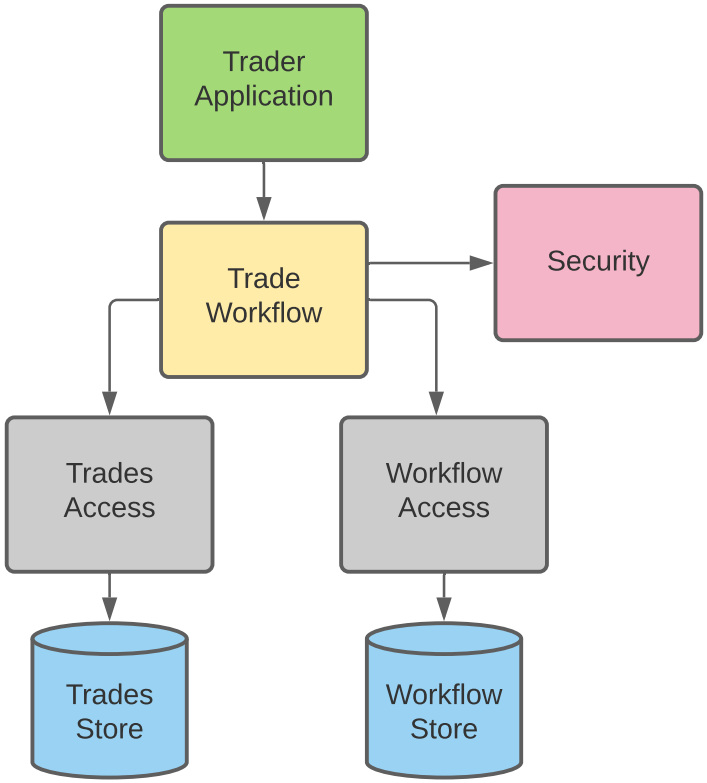
Volatility Mapping to Components

Volatility	Encapsulated in
Data Storage	Resource Access components
Client Notification	Notification component
Trading workflow	Trade Workflow component
Duration & Synchronisation	Trade Workflow component
Trade items	Trade Workflow component
Locale	Trade Workflow component
Market Feed	Feed Access component
Market Feed Content	Feed Transformation component
Authentication & Authorisation	Security component
User	Client Portal, User Apps

Task 4: Validation

- Validate the architecture by creating a Communication Diagram or a Sequence Diagram for use case *Buy Stocks*

Buy Stocks Communication Diagram



Buy Stocks Interaction

1. *Trader* selects stocks and commences a trade in the *Trader Application*.
2. *Trade Workflow* verifies (using the *Security* component) that *Trader* is authorized to trade the item, and loads the corresponding workflow (using *Workflow Access*)
3. *Trade Workflow* guides the *Trader* through the process. *Trader Application* handles the user interface part of this interaction.
4. The details of the trade are stored in the *Trades Store* (via *Trades Access*).
5. At each step of the process, the workflow is persisted in the *Workflow Store* (using the *Workflow Access*) so that it can be resumed later.

What about Reporting?

- Reporting was not identified as volatile
- There is no component for it
- Reporting can be realized using the *TradeWorkflow* component

Completing Architecture Verification

- Need to verify that *each* use case can be realised using the decomposition
- Verify that dependencies exhibit desirable structure
- Verify that each component has only one reason to change

You Should Know

- Identify volatilities in system requirements
- Identify components based on volatility and design principles
- Validate a component design on use cases

Activities this Week

- Complete Quiz 2

University of
South Australia