340CT
Software Quality and Process Management

Dr. Yih-Ling Hedley
Email: aa0817@coventry.ac.uk

Architectural Patterns

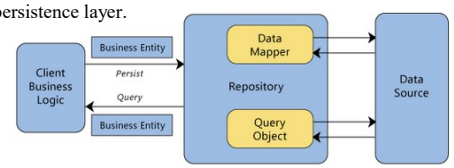
- Architecture design and patterns review
  - describes a fundamental and **structural** organization for software systems with a set of predefined subsystems
  - specifies the **relationships of subsystems**, and include the **rules & guidelines** for organizing their relationships
  - Examples: design patterns
    - Repository (layered architecture)
    - Publisher-Subscriber (event driven architecture)

Patterns: Repository – Review 1

- Repository pattern
  - Context: to deal with the problems with the applications where the **business logic requires accesses data from data stores**.
  - Solution: to use a repository to separate the logic that retrieves the data and maps it to the entity model from the business logic that acts on the model. A repository **separates the business logic from** the interactions with **the underlying data source.**

Patterns: Repository Review 2

- The repository **queries the data source** for the data, maps the data from the data source to a **business entity**, and persists changes in the business entity to the **data source**.
- A **separation between the database access and the application**: a Repository encapsulates the set of objects persisted in a data store and the operations performed over them in the persistence layer.



Source: The Repository Pattern (MSDN)

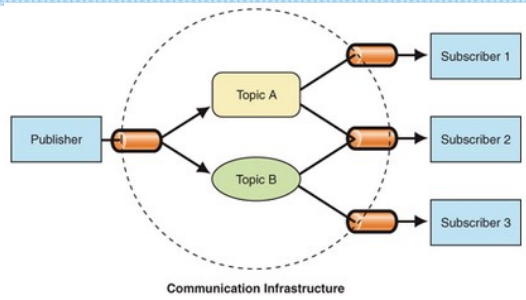
Patterns: Publisher-Subscriber 1

- Publisher-Subscriber pattern
  - Context: an integration architecture of several applications in which **an applications can send messages to the applications that are interested in receiving the messages** without knowing the identities of the receivers.
  - Solution: to set up communication infrastructure which creates a mechanism that **sends messages to all interested subscribers** and enables listening applications to subscribe to specific messages.

Patterns: Publisher-Subscriber 2

- Publisher-Subscriber pattern
  - **List-Based**: to identify a pre-defined subject and to maintain a list of subscribers for that subject. The subject notify each subscriber on the subscription list for an event.
  - **Broadcast-Based**: based on a pre-defined subject, all messages are sent to all listening nodes, and each node is responsible for filtering unwanted messages.
  - **Content-Based**: messages are intelligently routed to their final destination based on the content of the message.

Patterns: Publisher-Subscriber 3



Source: The Publisher-Subscriber Pattern (MSDN)

YL Hedley

Quality Attributes/Criteria 1

- Quality attributes review
  - Overall features of the architecture that affect **run-time behaviour, system design, and user experience**, such as: usability, performance, reliability, and security which indicates the success of the design and the overall quality of the software application.
  - When designing applications to meet any of these qualities, it is also important to consider the **impact on other requirements and the tradeoffs between multiple quality attributes.**

YL Hedley

Quality Attributes/Criteria 2

- Quality Attributes
  - The importance or **priority of each quality attribute differs** from system to system
    - In some cases, system performance, scalability, security, and usability will be more important than interoperability. Interoperability is likely to be more important in off-the-shelf software

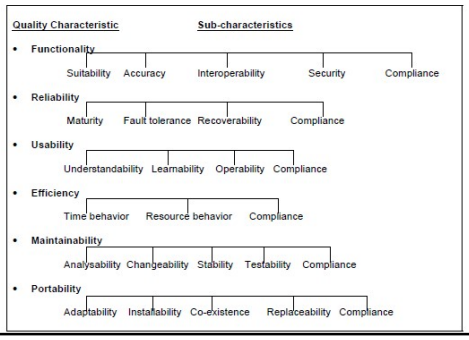
YL Hedley

Quality Attributes/Criteria 2.1

- **System qualities**: overall qualities of the system when considered as a whole, e.g. supportability, testability.
- **Run-time qualities**: qualities of the system directly expressed at run-time - availability, interoperability, manageability, performance, reliability, scalability, and security.
- **Design qualities**: qualities reflecting the design of the system - conceptual integrity, flexibility, maintainability, and reusability.
- **User qualities**: usability of the system.

YL Hedley

Quality Models: ISO 25010



YL Hedley

ISO 25010: Example

Quality Criteria Applied at Architecture Level 1

- **Functionality**
  - **Suitability**: provides the adequate functions for the required tasks.
    - Identification of system's functionalities.
    - Provision of an architecture specification with the **specified functionality decomposed into functions** associated to components which will meet the functional requirements of the system

YL Hedley

ISO 25010: Example

Quality Criteria Applied at Architecture Level 2

- **Functionality**
  - **Accuracy**: provides the right or agreed results or effects with the needed degree of precision.
    - Identification of the components with the functions responsible of the **computations** (functional components)

ISO 25010: Example

Quality Criteria Applied at Architecture Level 3

- **Functionality**
  - **Interoperability**: the ability to interact with one or more specified systems.
    - Identification of the connectors communicating with external specified systems. For example, to require Web services compatibility implies the existence of components.

ISO 25010: Example

Quality Criteria Applied at Architecture Level 3.1

- **Functionality**
  - **Interoperability**:
    - Determined by the presence or not of corresponding middleware components.(such as Web servers, application servers, content management systems) – e.g. Secure Sockets Layer (SSL) in the Sun Java System Application Server for authenticated and encrypted communication between clients and servers.

ISO 25010: Example

Quality Criteria Applied at Architecture Level 4

- **Functionality**
  - **Security**: the ability to prevent unauthorized access to programs or data.
    - Provision of a mechanism or device (software or hardware) to perform explicitly this task, may be a component (e.g., a service provided by the middleware) or a functionality integrated into a component

ISO 25010: Example

Quality Criteria Applied at Architecture Level 5

- **Reliability**
  - **Fault tolerance**: the ability to maintain a specified level of performance in case of software fault or of infringement of its specified interface.
    - Provision of a mechanism or software device, which may be a component or integrated into a component, e.g. **exception handling**

ISO 25010: Example

Quality Criteria Applied at Architecture Level 6

- **Reliability**
  - **Recoverability**: Capability to re-establish the level of performance and to recover the data and required time/effort.
    - Provision of a mechanism or software device, which may be a component or integrated into a component, to re-establish the level of performance or to recover the data, e.g. redundancy.

ISO 25010: Example

Quality Criteria Applied at Architecture Level 7

- **Portability**
  - **Installability**: the capability of the software product to be installed in a specified environment.
    - Provision of an install mechanism

More on ISO 25010 Quality Model: S. Wagner, Software Product Quality Control, DOI 10.1007/978-3-642-38571-1 2, © Springer-Verlag Berlin Heidelberg 2013

YL Hedley

ISO 25010: Case Study

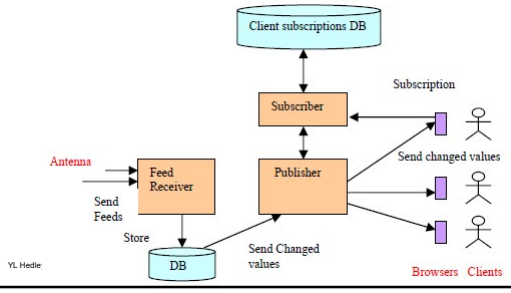
Market Stock Exchange Monitoring System (MSEMS)

- **Requirements:** a real-time data provider, for monitoring stock exchanges for brokers and independent investors. An antenna (**feed server**), external to the system, provides the data (feed) to the **data server**. A feed contains the relevant information of a stock exchange transaction. **The clients** (brokers), distributed in different geographical locations, subscribe with the data server. When a change on the feed to which a client has subscribed occurs, the feed is broadcasted to the client by the data server, according to a strict time delay. The time delay will depend on the network structure. Requirements for the system are high security, availability, platforms heterogeneity, distribution of clients, reliable information with strict deadlines.

YL Hedley

ISO 25010: Case Study

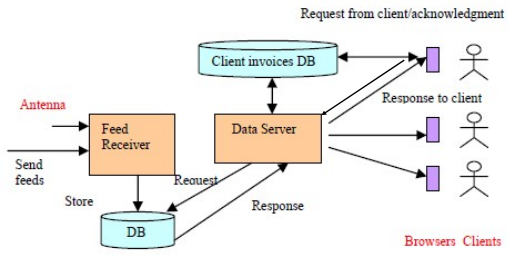
MSEMS Architecture Model 1: Publisher-Subscriber



YL Hedley

ISO 25010: Case Study

MSEMS Architecture Model 2: Repository



YL Hedley

Activity

- Evaluate the Publisher-Subscriber and Repository approach to the MSEMS system, which is better?
  - Security
  - Efficiency
    - in time behaviour (time spent from the data reception to the data delivery)
    - in resource utilization (usage of time)
  - Reliability

ISO 25010: Case Study

MSEMS Architecture Design 1

- Architecture designs in relation to quality characteristics
  - Model 1: **Publisher-Subscriber**, achieving better results in:
    - **Security**: a subscription mechanism , a better mechanism
    - **Efficiency in Time behaviour (time spent from the data reception to the data delivery):** better as less time required

YL Hedley

ISO 25010: Case Study

MSEMS Architecture Design 2

- Architecture designs in relation to quality characteristics
  - Model 2: **Repository** achieving better results in:
    - **Reliability (**maturity i.e. the capability of the software product to avoid failures**):** fewer components in the architecture
    - **Efficiency in resource utilization (time):** better as the Browser displays only on request

YL Hedley

Activity

- Evaluate the architecture solution based on the requirements for the MSEMS case study:
  - Publisher-Subscriber (Model 1) or Repository (Model 2) ?

ISO 25010: Case Study

MSEMS Architecture Design Comparison 3

- Overall architecture assessment for software quality:
  - Model 1 is better solution to Model 2 as given in the requirements of the MSEMS case study: **security and efficiency in time behaviour** should be considered to be the priority and are thus more important, than reliability (maturity) and efficiency in resource utilization (time)

YL Hedley