# Enterprise Application Development

## Lab 4 – Midterm Review

## Jake Young – C14706715

# Describe the Software Selection Maturity Scale? What is its relationship to technology adoption in the Enterprise?

The Software Selection Maturity Scale is a mechanism used to evaluate the state of your organizations selection process. It is a roadmap for improvements.

The SSMS is a five-level scale which measures the maturity of a given enterprise for its technology evaluation and acquisition process.

The five steps are:

* 1. Initial
  2. Basic
  3. Proactive and defined
  4. Verified and Adjusted
  5. Optimised, Tested and Approved

Technology is not adopted well in the enterprise as there are always new solutions developed daily which enterprises struggle to integrate due to capital and inefficiency.

# Compare and contrast the monolithic and SOA models of enterprise application software. Describe are the benefits of a SOA composition approach to application construction.

Monolithic enterprise application software cost very highly in terms of maintenance as they are such large complex systems.

Whereas, Service oriented architectures are less cost effective as they create a specific set of protocols at the beginning of setting up the software to ensure the operation runs smoothly.

Monolithic software is relatively easy to hook up its components to those cross-cutting. Monolithic software can become quite confusing as they get very much tangled together. Whereas, SOA software are loosely couple pieces of functionality.

# Why have enterprises moved towards web technologies for service software construction? What are the principal benefits?

Enterprises have adopted a lot of the web technology stack for services implementation as the stack has plenty of benefits for its usage which business’ always strive to have also such as cost-saving solutions.

Another benefit for using web technologies is that the technology in open sourced, this means that it will always be up to date and tested by many world-wide. Therefore, the enterprise knows that the technology in reliant and usable.

# What are the advantages and disadvantages of HTTP statelessness as the basis of a service application protocol tier?

The main advantage of HTTP being stateless is that you can clearly identify each request being sent or received by the client or server.

Another advantage is that each request and response can use their own unique TCP/IP connection.

However, HTTP being stateless also comes with disadvantages, such as the client or server cannot make any guarantee or promise about staying in the same state for any duration.

# Describe the architectural constraints of the REST architectural pattern

The REST architecture comes with some constraints such as uniform interface, client-server, stateless, cacheable, and a layered system.

Since the REST architecture has a uniform interface, the developer must decide on the API’s interface. The REST architecture ensures also that the client application and the server application must be completely independent. They must not correlate in any way.

Like HTTP, the REST architecture is also a stateless architecture. Each client and server request are separated.

The use of caches in the REST architecture allows for performance improvement as the user’s data is already saved. Lastly, the REST architecture allows to use a layered system which allows for the server to deploy API’s, and the client to store the data.

# Explain the relationship between resources, models and views? What is meant by view aggregation?

A model is the name given to some abstraction for internal persistent service state such as a database table. Resources are often known as associations as they are based off the models.

The view is the representational state of the model. Views use serialisation formats such as JSON or XML.

When a resource is based on a model, the view also becomes the representation of the resource.

# Describe the five RESTful operations, giving examples using HTTP. What is meant by idempotence? Mention which of the REST operations are idempotent and why.

Idempotence is the property of an operation such that the operation can be applied multiple times to some value without changing the outcome beyond its first application.

An operation may be known as being idempotent after the first application of its operation.

There are five RESTful operations which are the following; POST, GET, PUT, PATCH, and DELETE

The POST operation takes effect when new data is added to current data. GET operation is used to read resources from the databases.

The PUT operation is used to fully update a resource which is considered to be idempotent.

The PATCH command is used to partially update a resources but is not considered to be idempotent.

Finally, the DELETE operation is used to remove an item from a resource.

The POST operation is not idempotent as each call creates a new resource.

The GET operation is considered to be idempotent as it is reading a resource.

The PUT operation is considered to be idempotent as each call doesn’t alter the state.

The PATCH operation is not considered to be idempotent as it only partially updates a resource.

Lastly, the DELETE operation is considered to be idempotent because once the item is removed from the resource it doesn’t matter how many requests are made thereafter.

# Explain the problem of failure propagation in SOA systems. What are the desirable characteristics of an API versioning system? What are the two kinds of API compatibility?

The main issue with failure propagation in SOA systems occurs when an API doesn’t fully appreciate some of its behaviours being depended on by consumers a good API versioning system would have the following characteristics; API stability, Major changes, Minor changes, and build identifier.

API stability means how likely the system is to change and be relied on. Major changes of a system include features which have been added or existing ones changed.

Minor changes include existing features being updated. Finally, the build identifier pinpoints the precise origins of the API version.

There are two kinds of API compatibility; backward compatibility and forward compatibility. The backward compatibility means changes to the API to be ineffective to existing consumers. Whereas, forward compatibility means the API is designed in such a way that it will transparently interoperate with a future version of itself allows consumers using the new version of the API to work with the legacy API.

# Describe the major elements of the logical data model. Describe how it abstracts the details of database access in the application tier.

The logical data model is typically implemented in a middleware tier implemented in an object-oriented language such as Java, Python or Ruby.

The logical view model of an architecture service is typically composed of three elements, the models, the Language bindings for SQL, and the Database vendor driver.

The logical view model abstracts the details of a database access in the application tier by going through the database vendor driver which connects to the database directly. From here, data can be extracted with ease.

# Describe in detail the pathology of a SQL injection exploit. What should the application developer to avoid this kind avoid this kind of vulnerability.

When an attacker wants to exploit a query formation vulnerability. They can do so by repeatedly sending queries to the service with malformed information input in the hope that it will find a flaw.

Once the flaw is found, the attacker can potentially mount arbitrary attacks on the system to learn more about the schema, the data and extract or modify critical values.

In order to avoid exploitation, the developer should use a pre-parser function which checks the validity of the query before it is executed. Another method is to use a prepared statement or parameterised queries. Also isolating the execution within a tight security sandbox using database privileges is another great method.