Chapter 7
System Design

Software Engineering

Software Architectures

SW architecture community has codified architecture knowledge in

- Architectural styles or patterns
- Architectural tactics
- Reference architectures

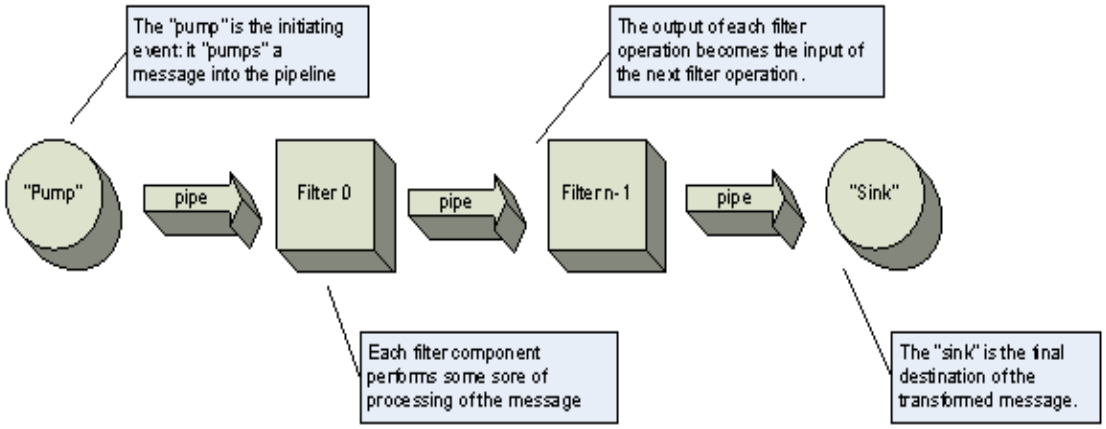
Pipes and filters

The **filter** transforms or filters the data it receives via the pipes with which it is connected. A filter can have any number of input pipes and any number of output pipes.

The **pipe** is the connector that passes data from one filter to the next. It is a directional stream of data, that is usually implemented by a data buffer to store all data, until the next filter has time to process it.

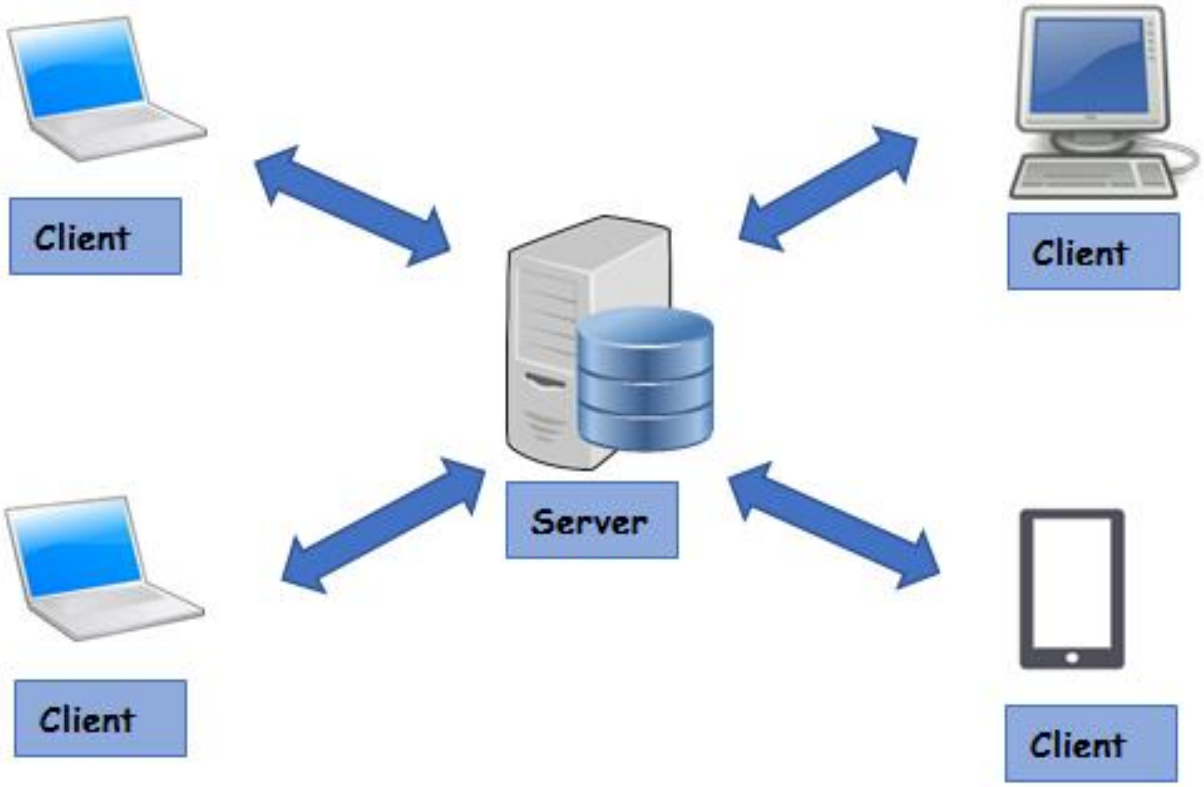
The **pump** or producer is the data source. It can be a static text file, or a keyboard input device, continously creating new data.

The **sink** or consumer is the data target. It can be a another file, a database, or a computer screen.

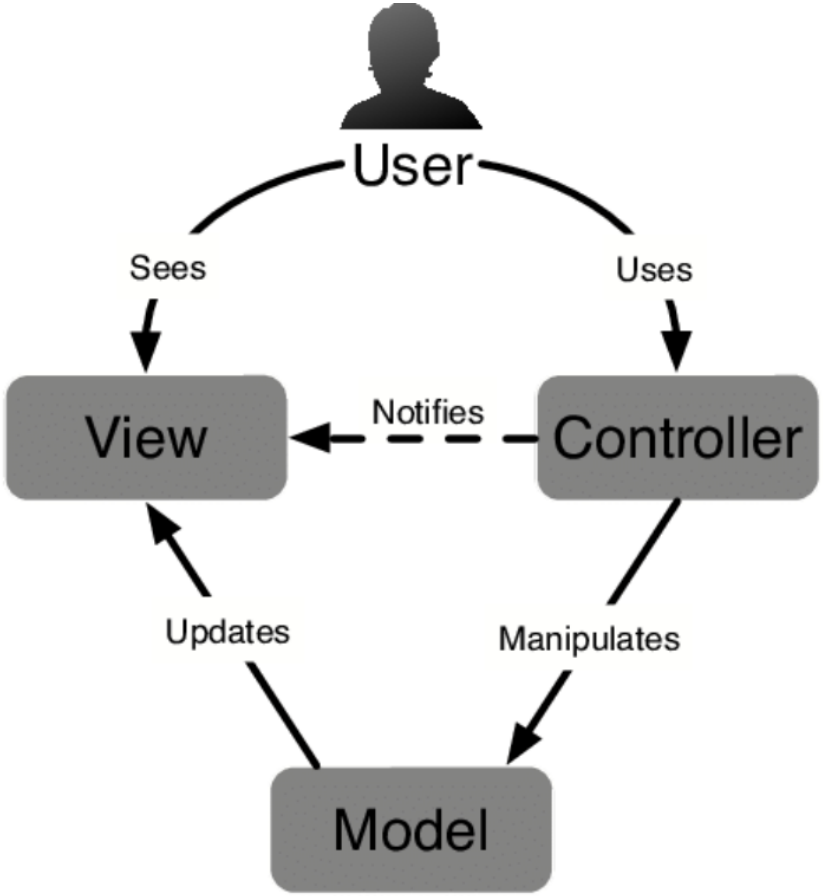


Event driven

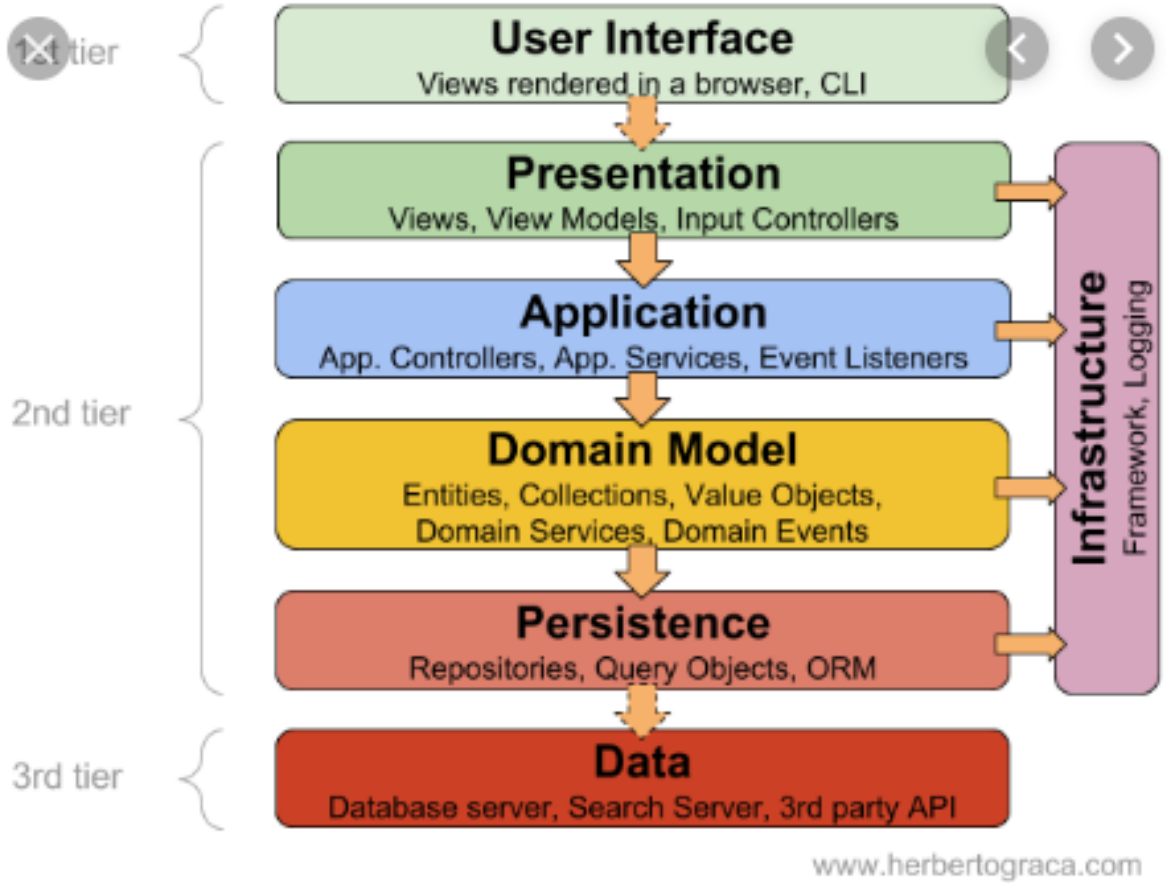
Client-Server



Model-View-Controller



Layered


1st tier
User Interface
Views rendered in a browser, CLI
2nd tier
Presentation
Views, View Models, Input Controllers
Application
App. Controllers, App. Services, Event Listeners
Domain Model
Entities, Collections, Value Objects,
Domain Services, Domain Events
Persistence
Repositories, Query Objects, ORM
Infrastructure
Framework, Logging
3rd tier
Data
Database server, Search Server, 3rd party API
www.herbertograca.com

Database-centric

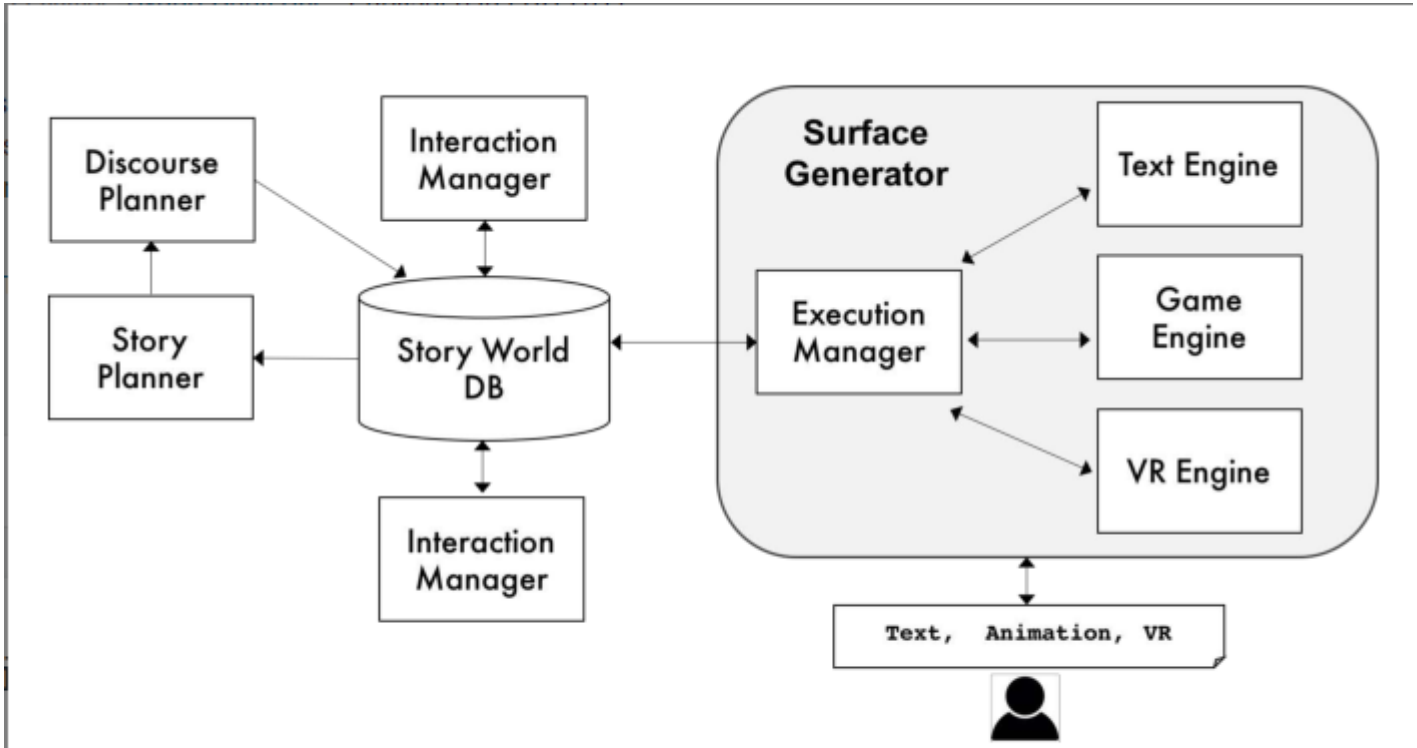
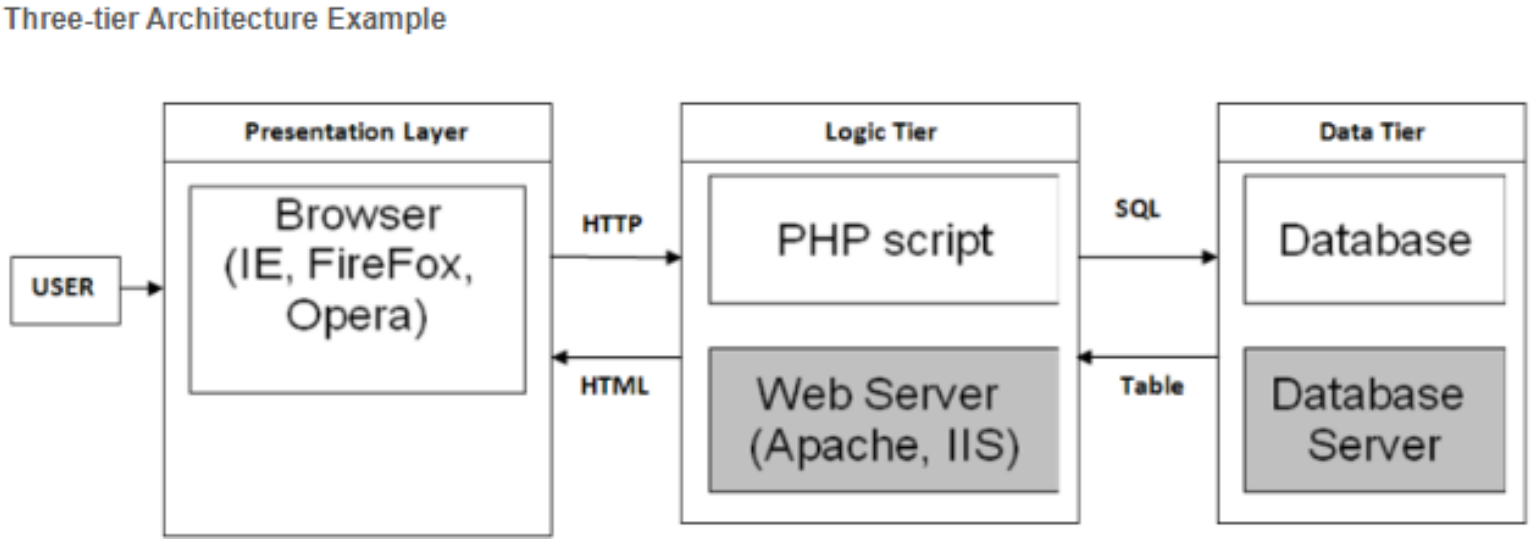
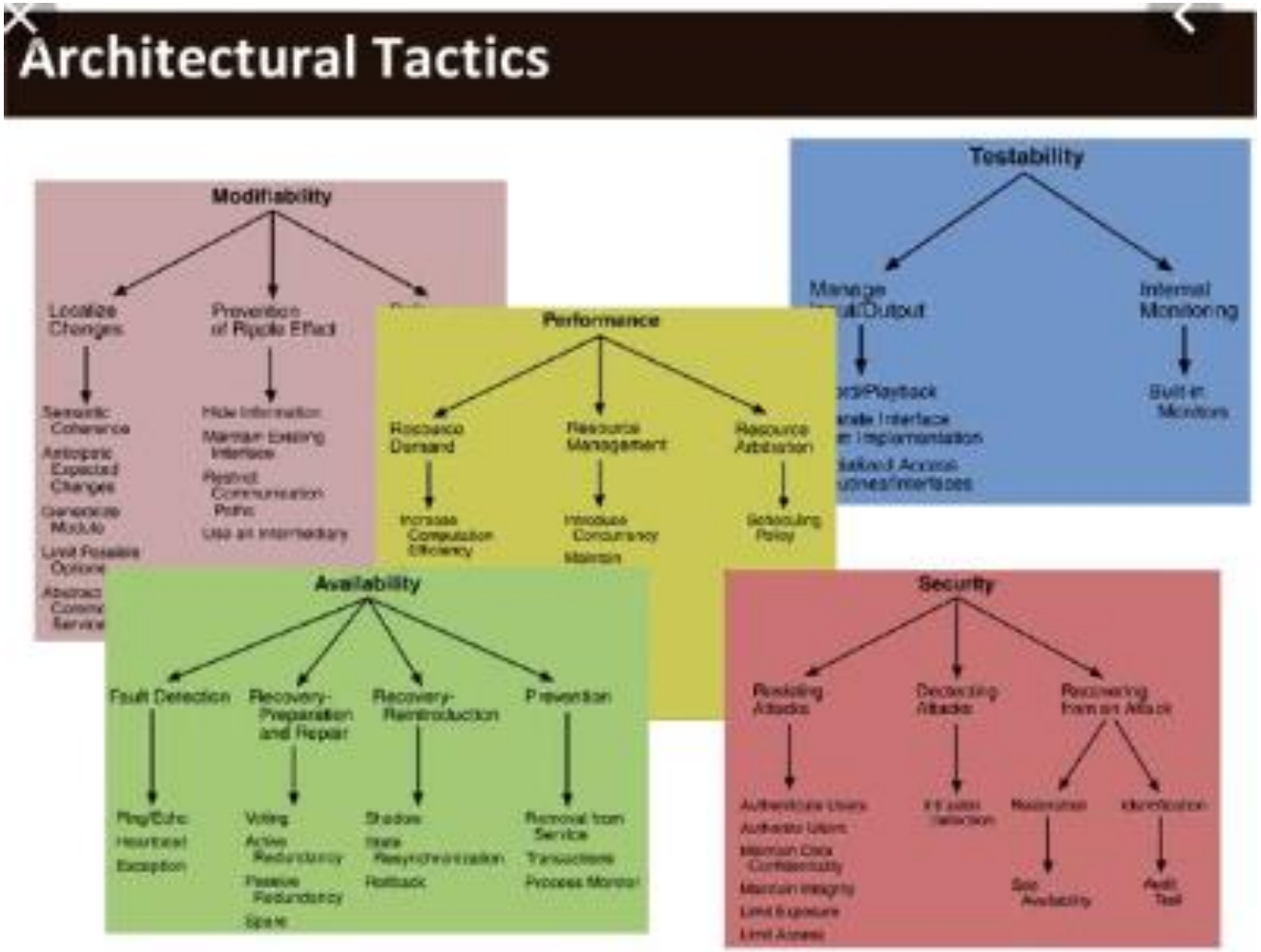


Figure 1: The design of interactive storytelling system employing a database.

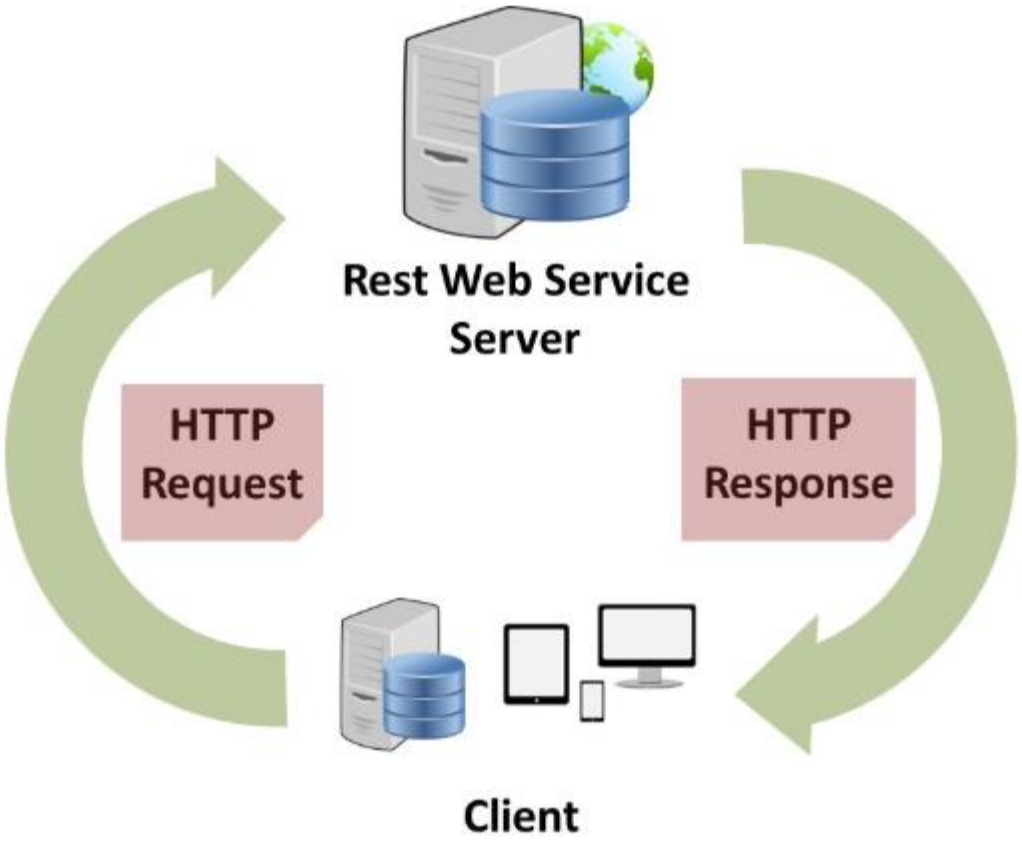
Three-tier


Three-tier Architecture Example
Presentation Layer
Logic Tier
Data Tier
USER
Browser
(IE, FireFox,
Opera)
HTTP
HTML
PHP script
Web Server
(Apache, IIS)
SQL
Table
Database
Database
Server

Architectural Tactics



Reference Architecture - REST



RESTful Web Services Architecture

https://medium.com/@sagar.mane006/understanding-rest-representational-state-transfer-85256b9424aa

Representational State Transfer, REST Architectural Style

Well-designed Web application

- Network of web pages (a virtual state-machine)
- User progresses through an application by selecting links (state transitions)
- Resulting in the next page (representing the next state of the application

Roy Fielding Ph.D. dissertation, 2000

REST Architectural Style

REST style has formal and informal “constraints:

- Client-server
- Stateless
- Cacheable
- Uniform interface
- Layered system
- Code on demand (optional)

https://medium.com/@sagar.mane006/understanding-rest-representational-state-transfer-85256b9424aa

REST Architectural Style

Steps

- Define the domain and data.
- Organize the data into groups.
- Create URI to resource mapping.
- Define the representations to the client (XML, HTML, CSS, …).
- Link data across resources (connectedness or hypermedia).
- Create use cases to map events/usage.
- Plan for things going wrong.

Detailed Design

Traditional

- design is created and documented
- programmers translate design to code

Agile

- programmers are more likely to develop the design

Functional Decomposition

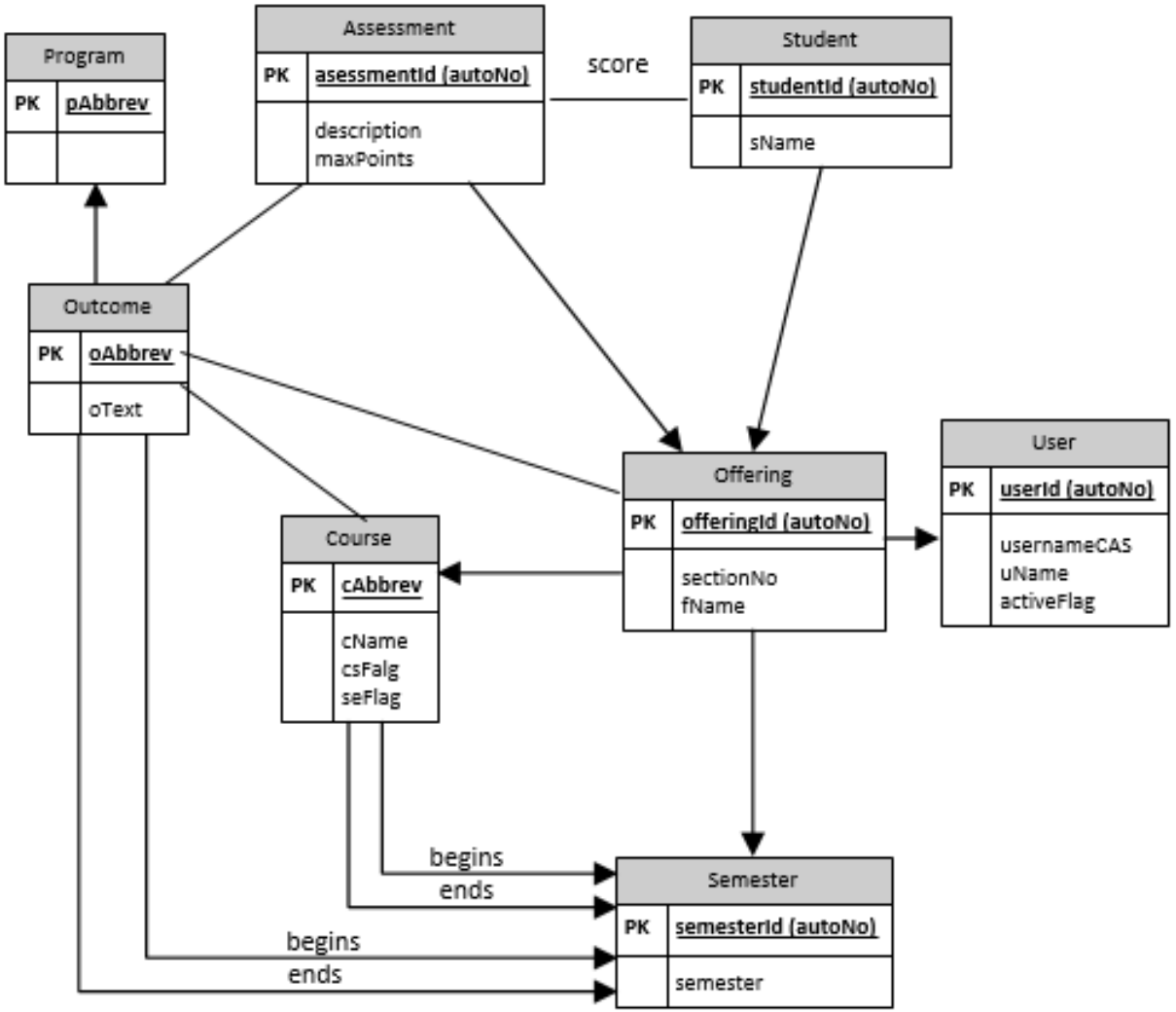
Functional decomposition – decompose a function or module into smaller modules

Relational DB Design

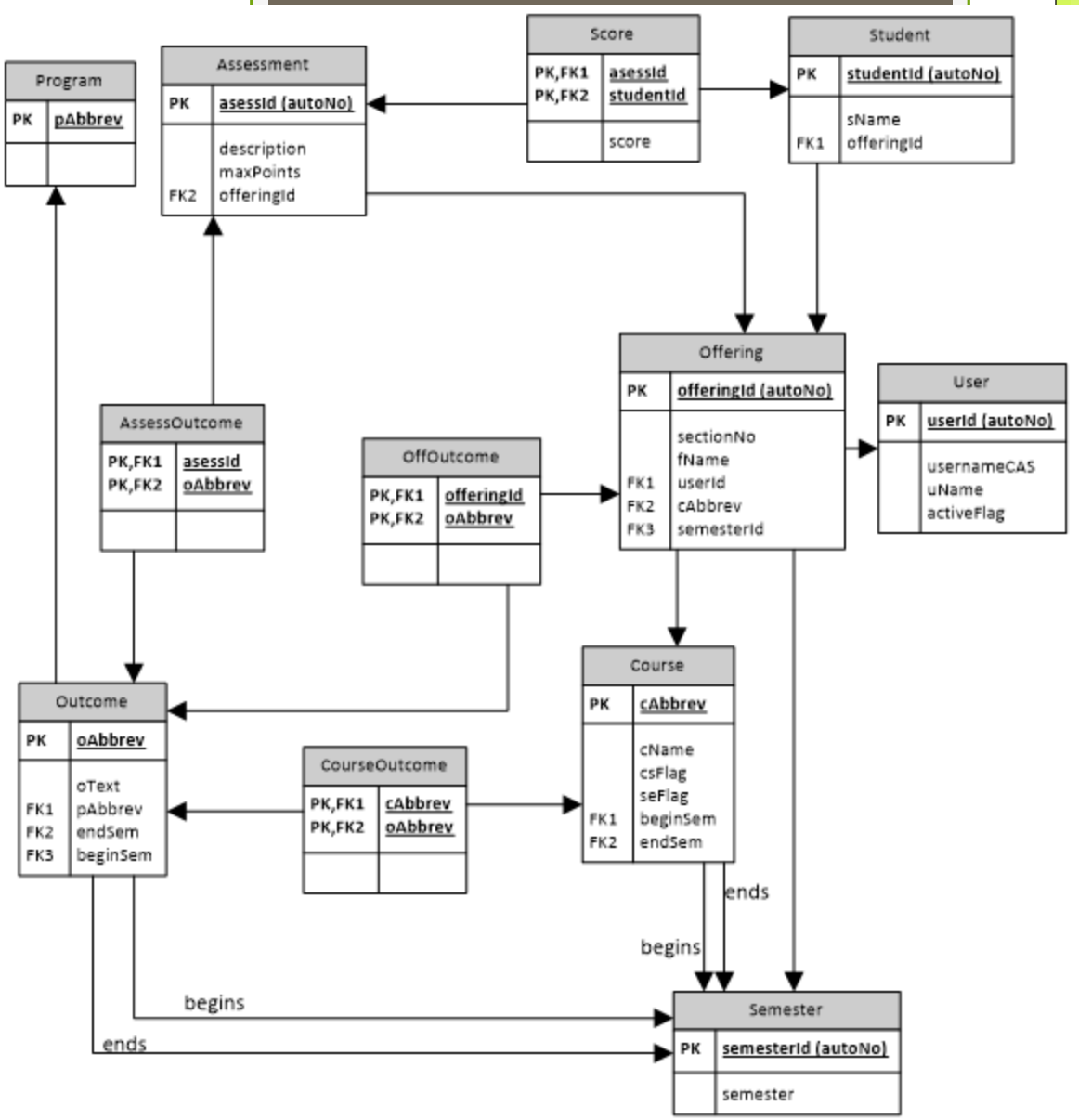
Relational db design

- Data modeling – attributes may be multivalued
- Logical db design – normalization and foreign keys
- Physical - select data types and indices
- Deployment and maintenance - where db resides, which tables go which hard drives

Data Modeling



Logical DB Design



NoSQL

NoSQL – "Not only SQL"

- Key-value pairs
  - Redis
- Values can be complex objects
  - MongoDB
  - Cassandra (Apache)
  - Dynamo DB (Amazon)
  - Azure
- Hadoop (Apache)– handles huge amount of data as text files
  - Hadoop
  - Impala
  - Hbase
  - Hive

Sharding

Sharding – storing different keys of a key-value store or different rows of a database in different computers

- Provides horizontal scalability

User-Interface Design

User-interface design is based on psychology, cognitive science, aesthetics and art. Two main issues:

- Flow of interactions (most important)
- Look and feel

User-Interface Design

Cognitive model:

1. Form goal
2. Form intension
3. Specific action
4. Execute actions
5. Perceive system state
6. Interpret feedback
7. Evaluate feedback

User-Interface Heuristics

UI Heuristics:

- Consistency
- Put the user in control
- Reduce the user's memory load
- Making the system status visible
- Metaphors are helpful

Object-Relational Impedance Mismatch

Object-relational impedance mismatch – mismatch between code (classes) and relational database tables