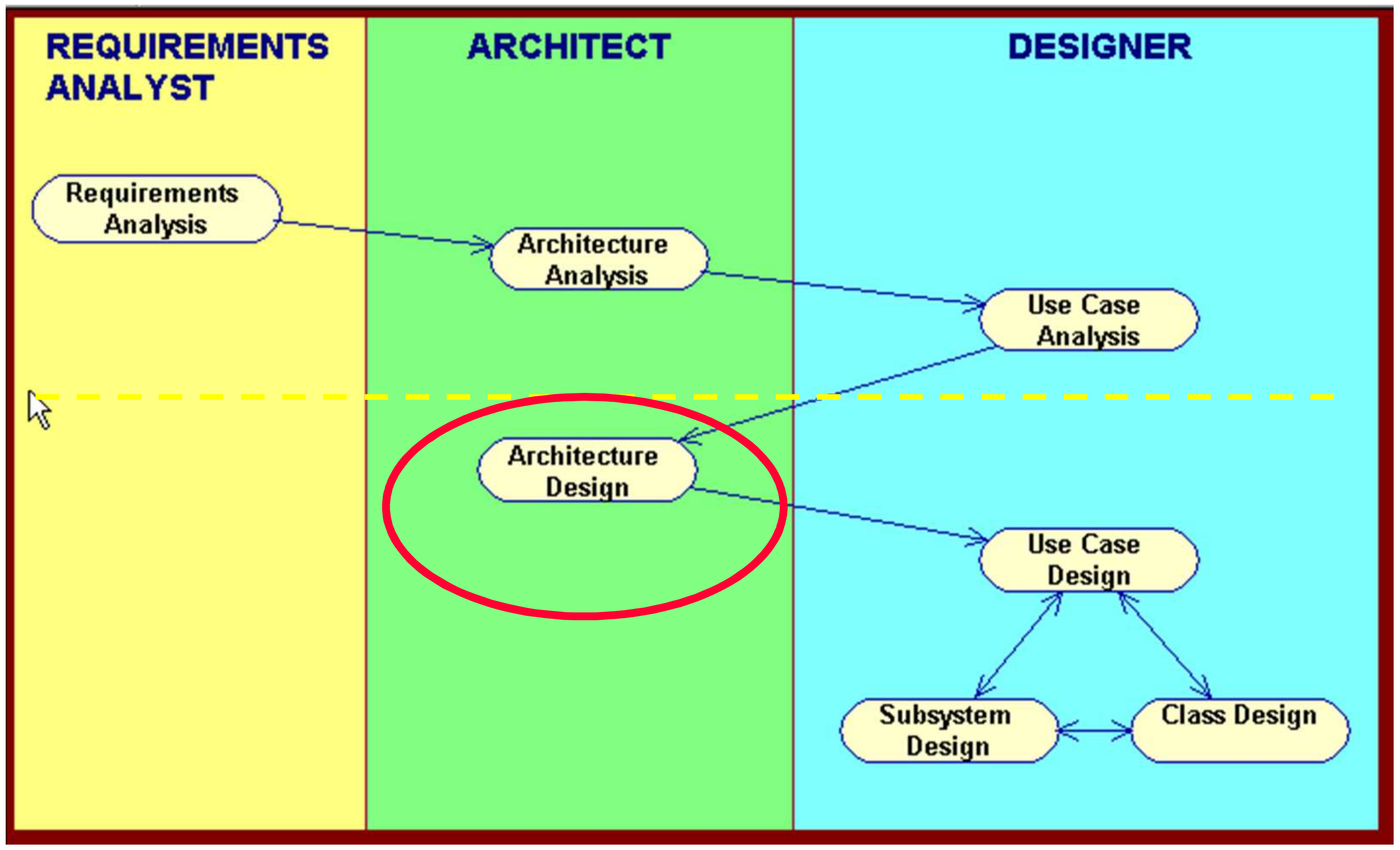
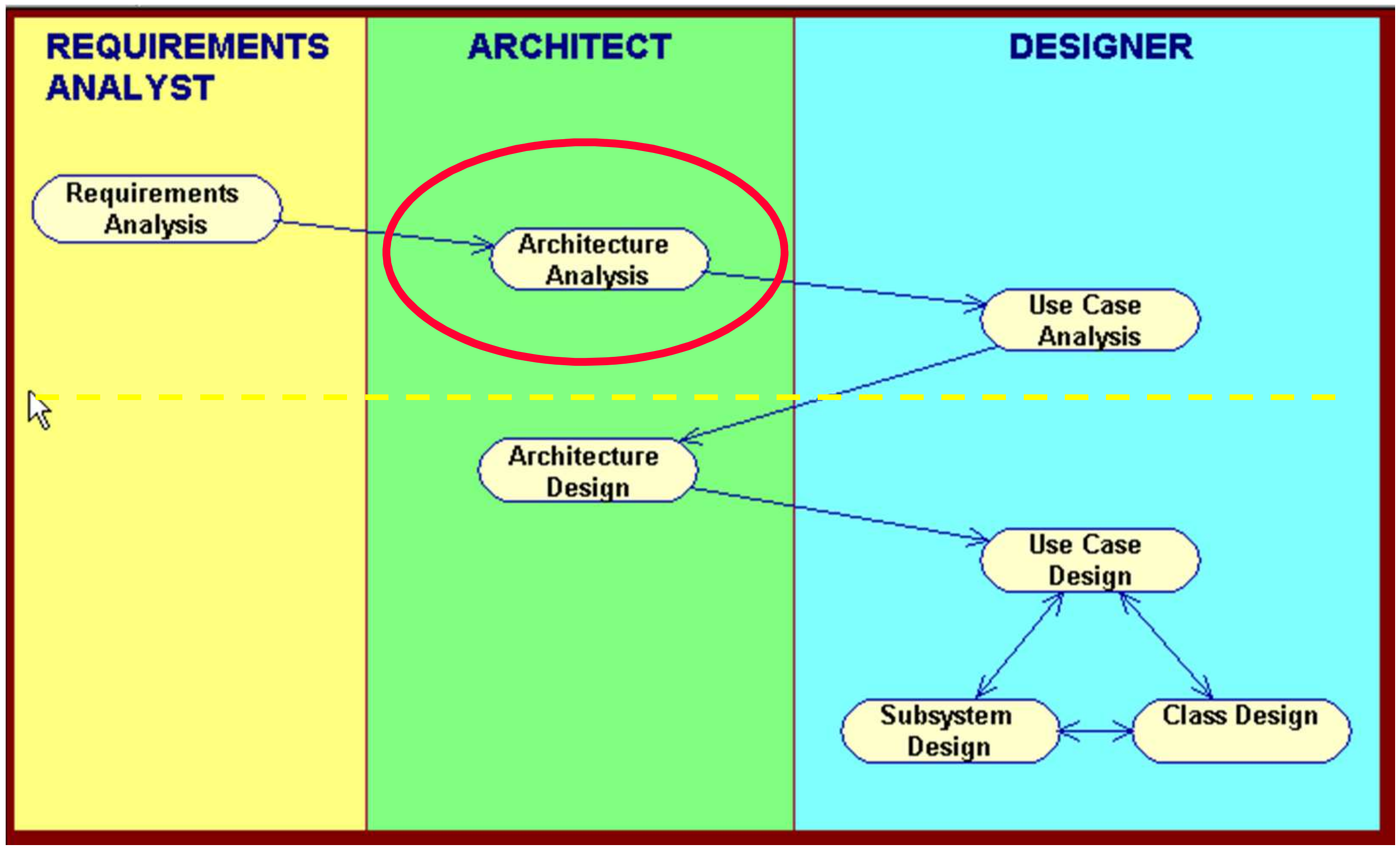
RESTful Web Services

The whole is more than sum of the parts

Basic RUP OOAD Activities



Basic RUP OOAD Activities



Introduction to REST

- REST (Representational State Transfer) was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation.

- REST is an architectural style for designing distributed systems. It is not a standard but a set of constraints, such as being stateless, having a client/server relationship, and a uniform interface.

- REST is not strictly related to HTTP, but it is most commonly associated with it.

Principles of REST

- **Resources** expose easily understood directory structure URIs.
- **Representations** transfer JSON or XML to represent data objects and attributes.
- **Messages** use HTTP methods/verbs explicitly (for example, GET, POST, PUT, and DELETE).
- **Stateless** interactions store no client context on the server between requests. State dependencies limit and restrict scalability. The client holds session state.

HTTP Method GET

- Retrieve information.
- GET requests must be safe and idempotent, meaning regardless of how many times it repeats with the same parameters, the results are the same. They can have side effects, but the user doesn't expect them, so they cannot be critical to the operation of the system.
- Retrieve an address with an ID of 1:

GET /addresses/1

HTTP Method POST

- Request that the resource at the URI do something with the provided entity. Often POST is used to create a new entity:

- Create a new address:

POST /addresses

HTTP Method PUT

- Store an entity at a URI. PUT is used to update an existing entity. A PUT request is idempotent. Idempotency is the main difference between the expectations of PUT versus a POST request. PUT is idempotent.
- Modify the address with an ID of 1:

  PUT /addresses/1

Note: PUT replaces an existing entity. If only a subset of data elements are provided, the rest will be replaced with empty or null.

HTTP Method PATCH

- Update only the specified fields of an entity at a URI. A PATCH request is idempotent. Idempotency is the main difference between the expectations of PATCH versus a POST request.

PATCH /addresses/1

Note: PATCH replaces only a subset of data elements (only elements that are provided), the rest of the elements remain unchanged.

HTTP Method DELETE

- Request that a resource be removed; however, the resource does not have to be removed immediately. It could be an asynchronous or long-running request.

- Delete an address with an ID of 1:

DELETE /addresses/1

HTTP Status Codes

Status codes indicate the result of the HTTP request:

- **1XX** - informational
- **2XX** - success
- **3XX** - redirection
- **4XX** - client error
- **5XX** - server error

Media Types (Representations)

- The Accept and Content-Type HTTP headers can be used to describe the content being sent or requested within an HTTP request.

- Client may set Accept to **application/json** if it is requesting a response in JSON.

- Conversely, when sending data, setting the Content-Type to **application/xml** tells the client that the data being sent in the request is XML.

Understanding JSON

JSON (JavaScript Object Notation) is a lightweight syntax for exchanging data that is designed to be understood easily by humans, and parsed easily by machines.

As the name implies, JSON is based on the JavaScript scripting language; however, JSON itself is completely language independent.

JSON is a popular notation for transmitting data through RESTful web services.

Examples of JSON

This object has three fields, where "name" is a string, "age" is a number, and "member" is a boolean.

```
{
        "name": "John",
        "age": 35,
        "member": false
}
```

Examples of JSON

Arrays contain a list of values, which can be of any type.

This object has two arrays; the "indexes" array contains numbers, and the "names" array contains strings.

```
{
        "indexes": [5, 10, 15, 20],
        "names": ["John", "Elizabeth", "Mary"]
}
```

Examples of JSON

An object containing other objects:

```
{
        "firstName": "John",
        "lastName": "Smith",
        "birthday": "1975-01-31",
        "spouse": {
                "firstName": "Mary",
                "lastName": "Smith"
        }
}
```

Demo 1

- See a real RESTful application in action!