Service Orientation and WCF

Miguel A. Castro
@miguelcastro67

……. SOA …….

"The decomposition of a system into autonomous or nearly autonomous units of responsibility, and exposure of said units in a secure and controlled fashion."

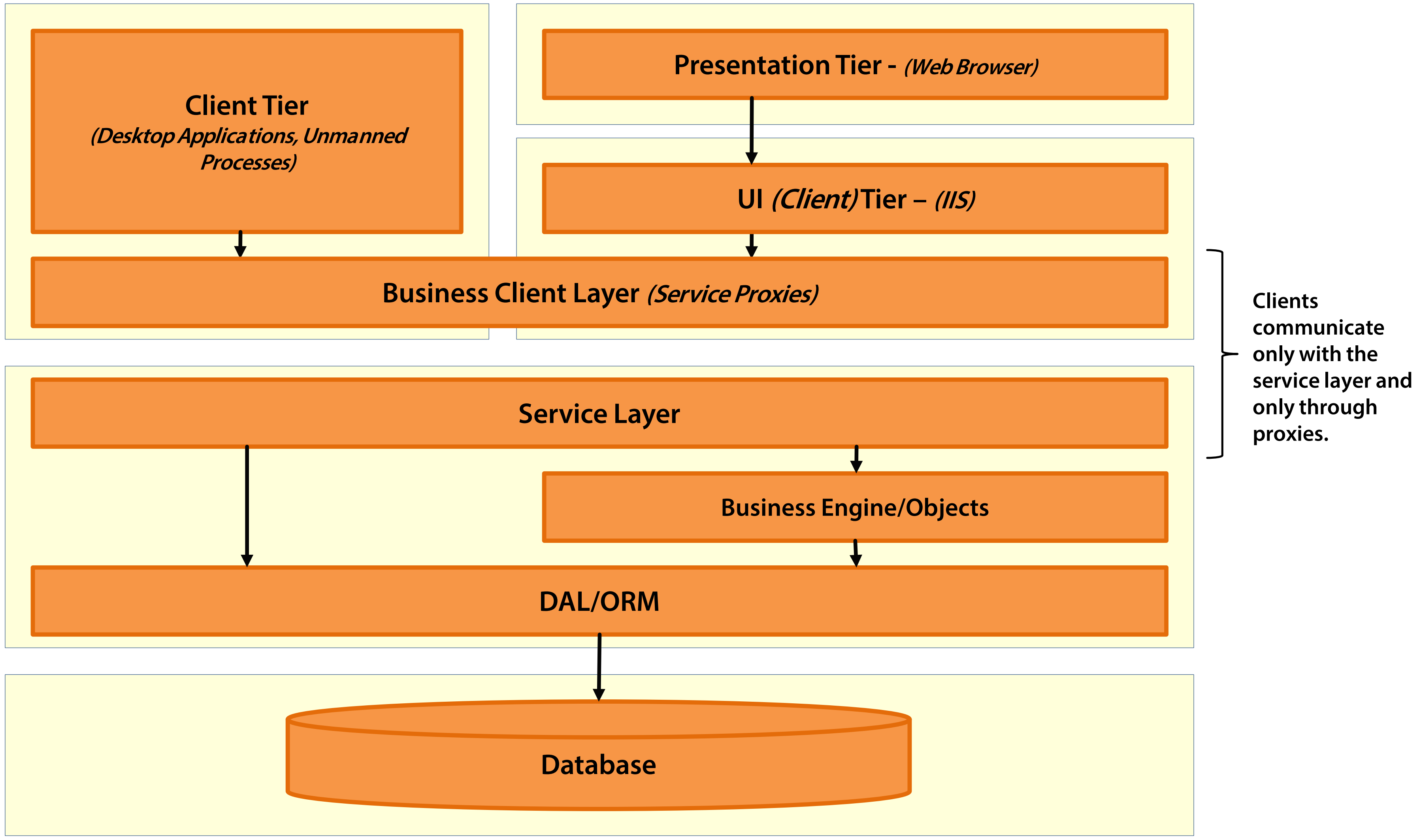
Now in Plain English

- **The exposure of an API for your system**
- **All clients access functionality by making service calls**
- **SOA has also been used for Service Oriented Application**
  - Probably a bit more accurate in real-world use
- **An application's business layers are governed by a set of loosely coupled services**
- **Not a replacement for OOP**
- **An evolution**
  - Procedural
  - Object Oriented
  - Component Oriented
  - Service Oriented

What Is a Service?

- **Collection of related units of responsibility (operations)**
  - Rock solid at handling said responsibility
- **An orchestration (entry) layer for the client atop rest of architecture**
  - Manages calls to down-level layers
- **Secure, leaves system in consistent state, varies in instantiation, thread aware, gracefully handles faults**

Service Oriented Architecture


Client Tier
(Desktop Applications, Unmanned Processes)
Presentation Tier - (Web Browser)
UI (Client) Tier – (IIS)
Business Client Layer (Service Proxies)
Service Layer
Business Engine/Objects
DAL/ORM
Database
Clients communicate only with the service layer and only through proxies.

Service Oriented Applications

- **Applications whose potentially volatile areas are wrapped in a simple service call exposed to a client.**
  - This is known as an operation
  - Client is not exposed to any details of how an operation is fulfilled
- **Details of processing can evolve or change over time, while client should not need to change.**
- **Hide away anything client should not have access to.**
  - Prevents a client from unwanted intrusion into other part of a system
  - Removes most application configuration from a client’s computer

Service Oriented Applications

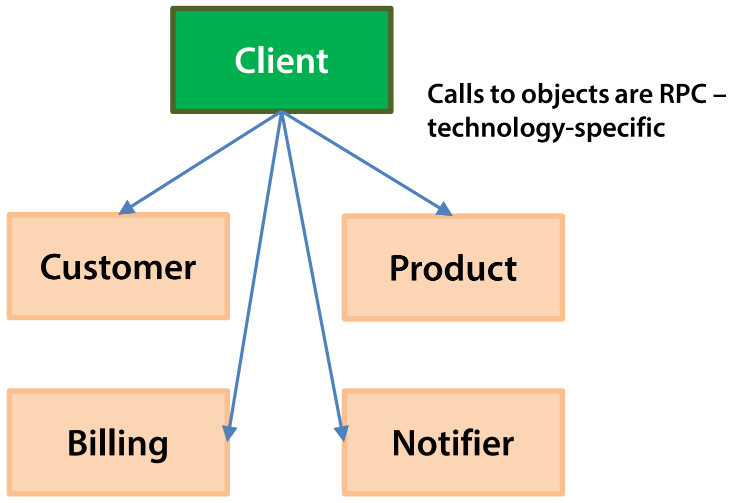
- **In object oriented programming, it’s all about the “object”**
  - Objects contain state and behavior
  - Objects may be persisted in memory for a long time
  - Clients have direct access to business objects
  - Objects physically deployed to every client
  - Client need to know how to use the exposed object model
  - Client typically needs to perform all the necessary baby steps a business operation may need

Service Oriented Applications

- **In service oriented programming, it’s all about the “call”**
  - Clients make a call containing all data needed for processing
  - Calls are in and out (usually assumed stateless)
  - Service receives data, performs processing, and returns response
    - Processing can include more down-level calls
    - Can open and close a database connection
  - Next call starts process over
  - Services and all down-level resources physically exist on service-side only
  - Much more scalable architecture
  - Clients need only know what calls are available
  - Details of processing are hidden from the client behind the call

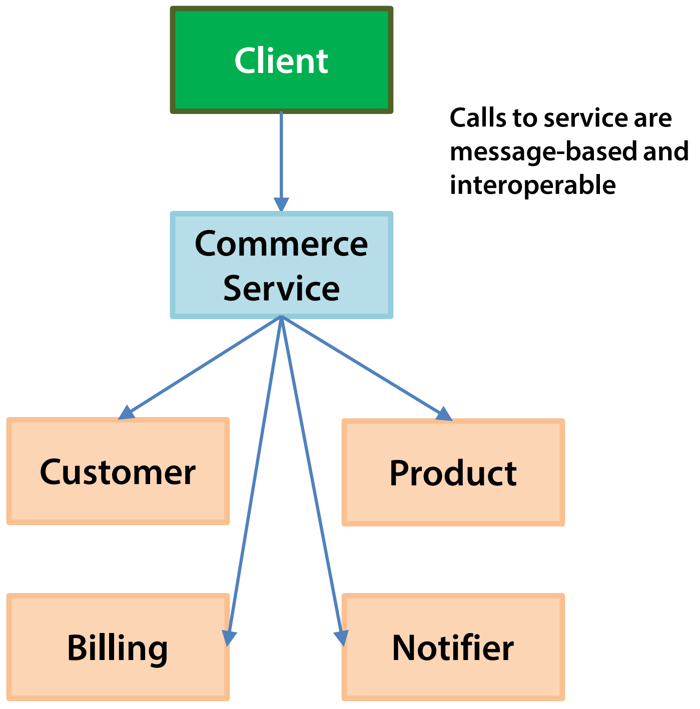
Service Oriented Applications

Object Oriented E-Commerce Task



Business objects are filled with the data they need and the methods called in order to perform necessary processing.

Service Oriented E-Commerce Task



Service orchestrates all down-level calls in a single transaction.

Service Oriented Technologies

- **Facilitate the task of writing an application's service layer and exposing it using industry-standard protocols**
- **Encapsulate plumbing for low-level tasks**
  - Communications (first and foremost)
  - State management
  - Transaction facilitation
- **Provide tooling necessary**
  - Service side
  - Client side

In the Old Days

- **.NET Remoting**
  - TCP-Compatible
  - Fast but .NET to .NET only
  - Cryptic configuration
- **Web Services**
  - HTTP only
  - Too forgiving
  - Promoted bad practices
  - Required WS* for advanced functionality

In the Old Days

- **Enterprise Services**
  - Provided advanced services to components
    - Transactions, Security, etc.
  - PITA to setup and work with
- **MSMQ**
  - Great for reliable message transportation
  - Its own API (yet another thing to learn)

In the Old Days

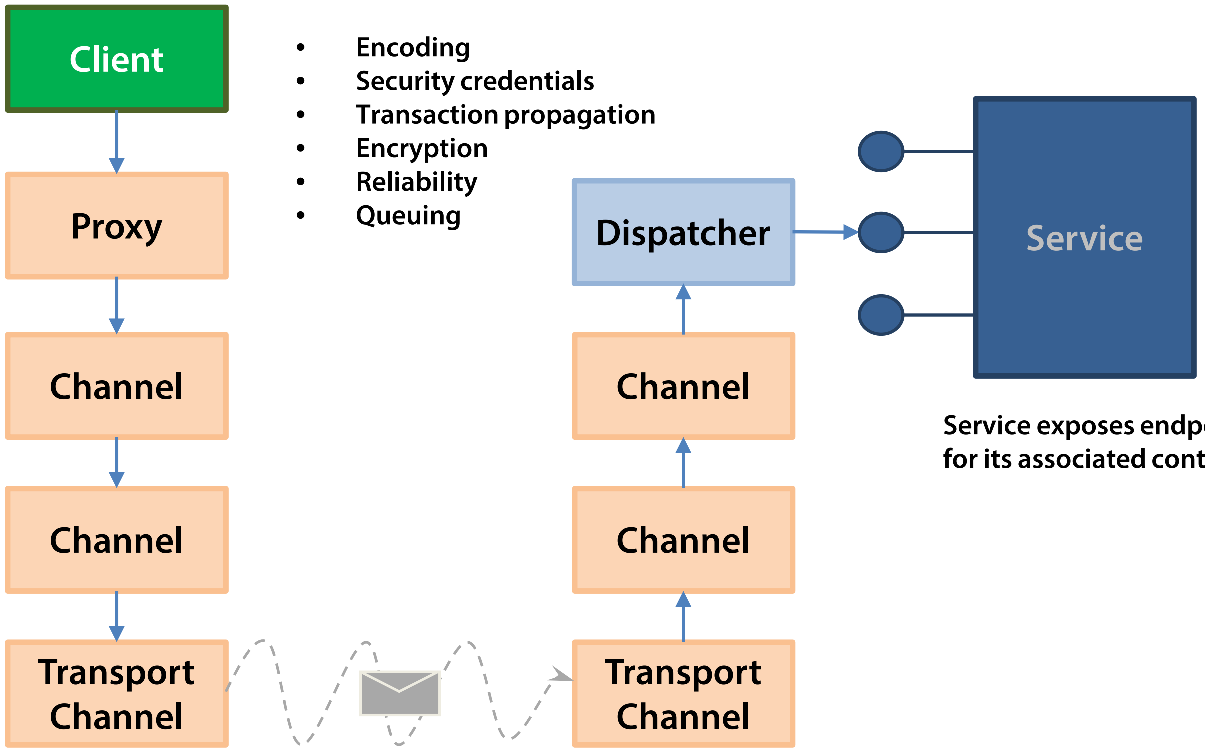
- **Sockets**
  - Good for point-to-point
  - Code heavy
  - Lots of manual details
    - Buffer size, ports, address, chunks, etc.

Windows Communications Foundation

- **Predominantly SOAP based, but REST capable**
- **Provides unified programming model for several technologies**
- **Abstracts the details of communications**
- **Developer writes services one way, despite underlying transport used**
- **Provides a declarative and aspect-oriented approaches for managing service characteristics**
  - Transactions
  - Reliability
  - State
  - Security
  - Instantiation
  - Others

WCF Architecture

- **Known sometimes as an interception-architecture**
  - Also referred to as a Pipeline architecture
  - Very similar to that of ASP.NET
  - Loosely based on the GoF Chain of Responsibility pattern



Channels check for and act upon a serialized message:

- Encoding
- Security credentials
- Transaction propagation
- Encryption
- Reliability
- Queuing

Service exposes endpoints for its associated contracts.

WCF vs. Web API

- One is NOT a replacement for the other
- One is NOT better than the other
- Different technologies
- WCF is WAY more feature rich
- Web API is WAY more interoperable
- WCF is based on the SOAP protocol
- Web API is based on the REST architecture
- WCF requires tooling
- Web API just requires ability make an HTTP request
- WCF’s binding choices make it faster in the firewall

WCF Components

- **WCF is very explicit**
- **Contract-based**
  - An API designed with WCF is very well defined
- **WCF uses**
  - Data Contracts
  - Service Contracts
  - Services
  - Hosts
  - Client Proxies
  - Configuration
- **Note to Web API advocates:**
  - Most of these concepts are represented in one way or another in Web API

So When Do I Want to Use WCF?

- **Most systems today have many client components**
  - Desktop
  - Browser
  - Mobile App
  - Mobile Browser
- **Inside the firewall, WCF still offers the best set of features**
  - .NET to .NET is almost a no-brainer
  - Browser apps are still “Intranet” apps (remember where IIS lives)
- **Mobile apps (not mobile browser apps) will access REST-based services better**
  - Web API becomes a client to WCF and “extends” reach for only those services needed by mobile clients
- **Moral of the Story:**
  - Most systems will benefit from both technologies today

Let’s go get serviced !