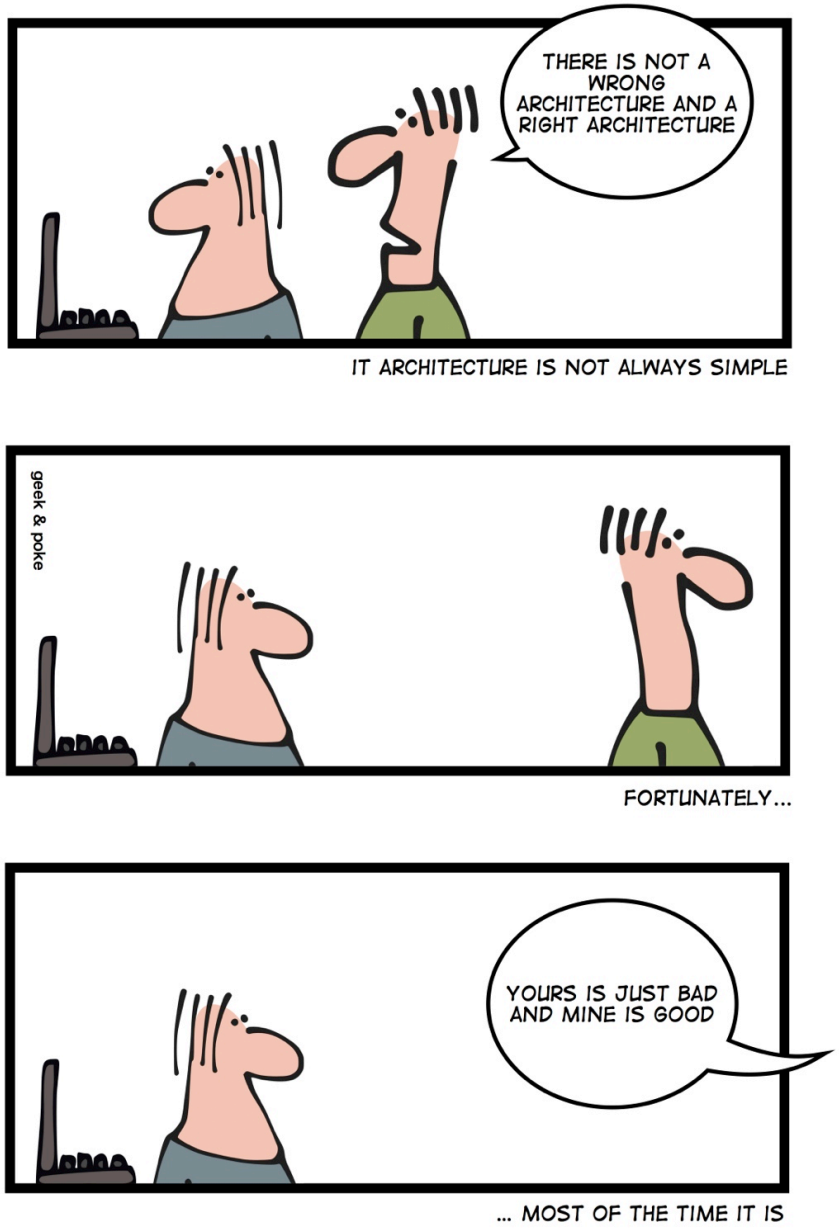
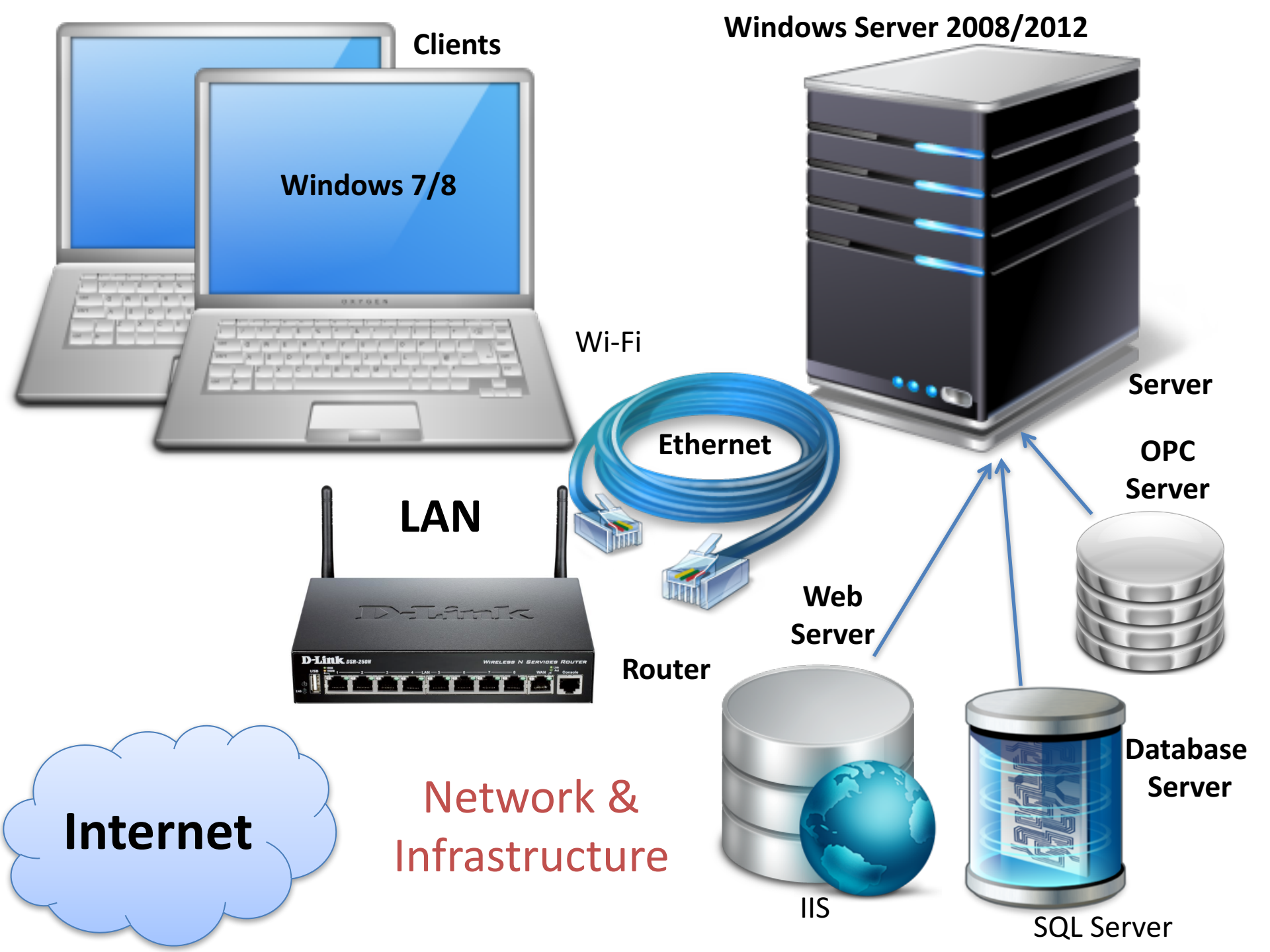
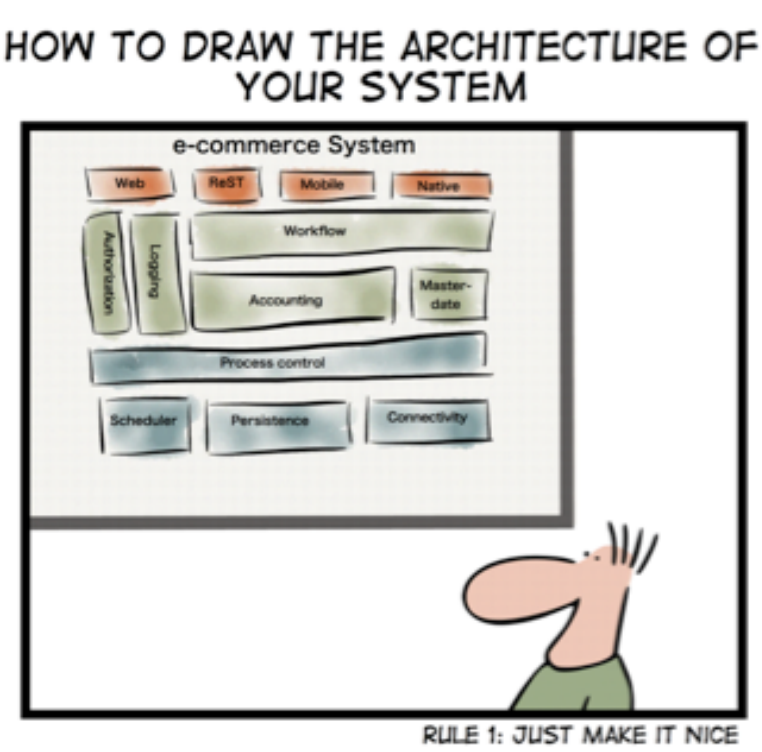


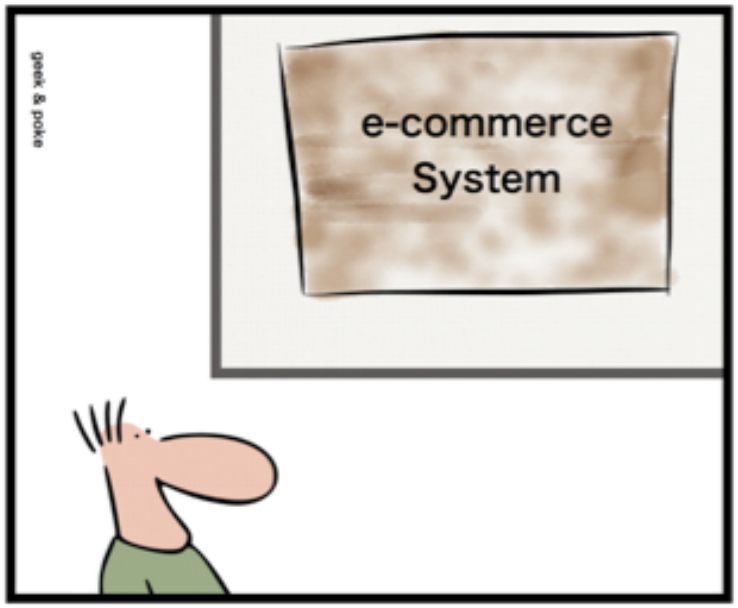
O. Widder. (2013). *geek&poke*. Available: http://geek-and-poke.com

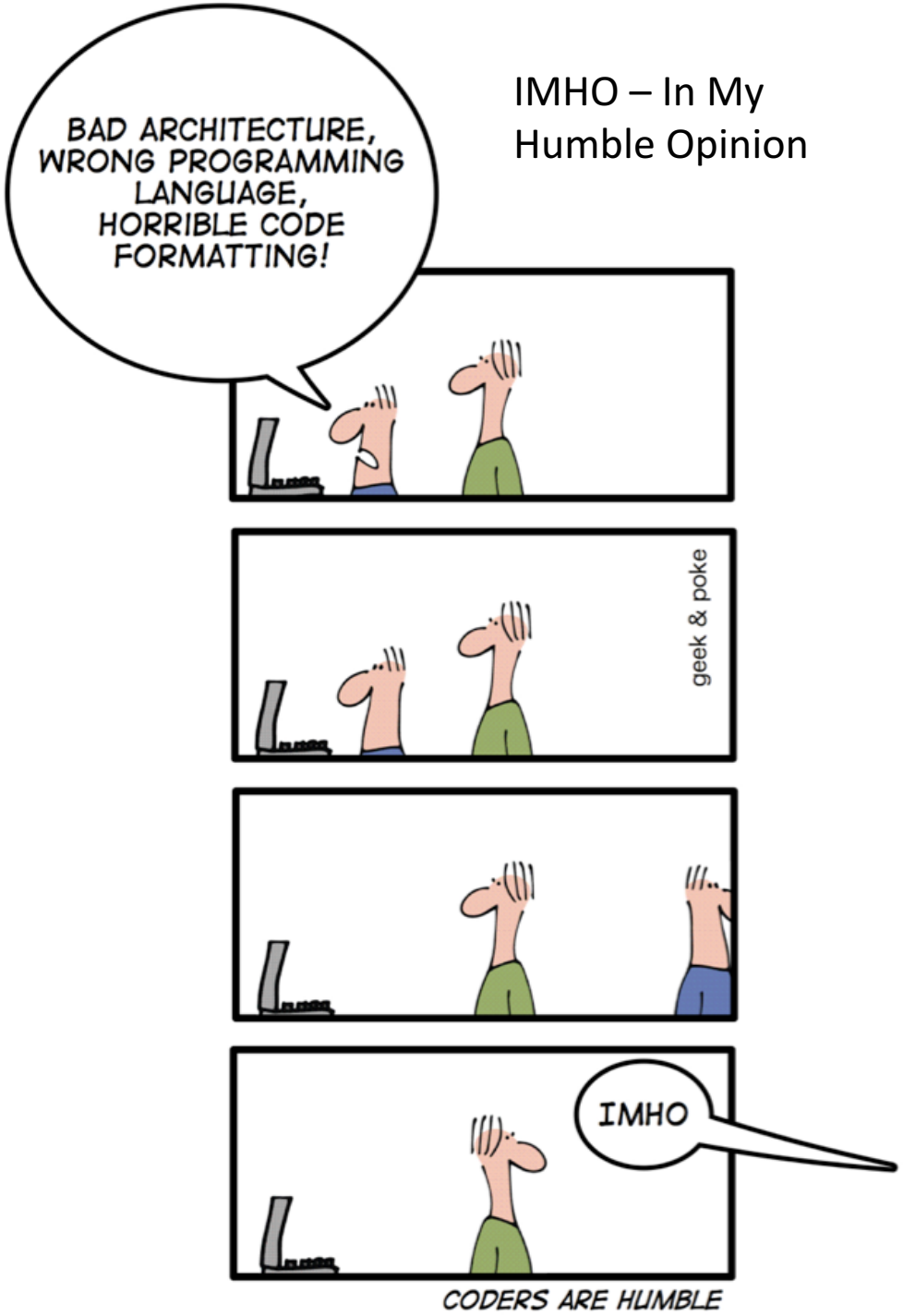
Software Architecture

Hans-Petter Halvorsen


Clients
Windows 7/8
Windows Server 2008/2012
Wi-Fi
Server
Ethernet
OPC
Server
LAN
Web
Server
Router
Internet
Network &
Infrastructure
Database
Server
IIS
SQL Server



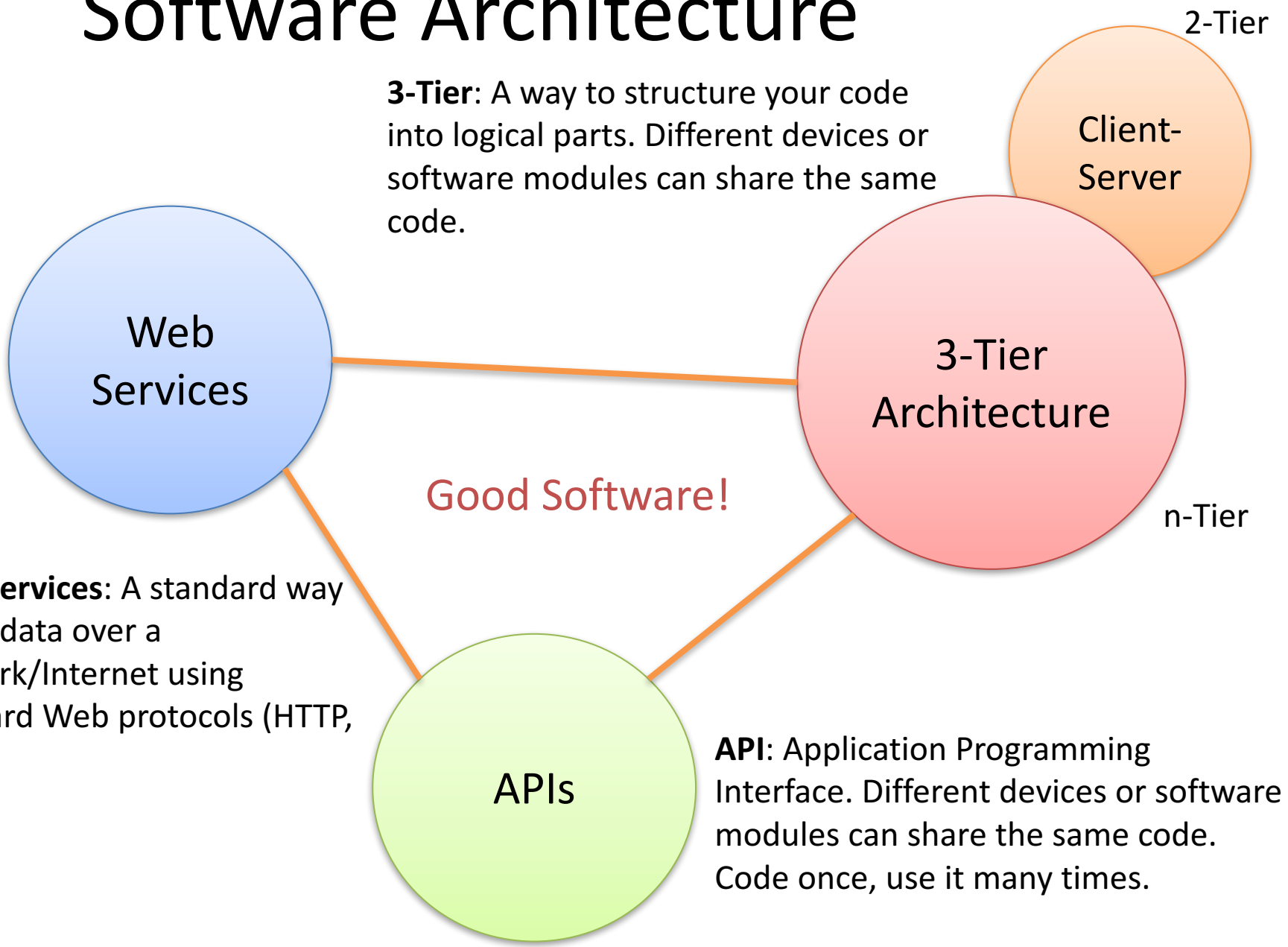




IMHO – In My Humble Opinion

O. Widder. (2013). *geek&poke*. Available: http://geek-and-poke.com

Software Architecture



3-Tier: A way to structure your code into logical parts. Different devices or software modules can share the same code.

Web Services: A standard way to get data over a network/Internet using standard Web protocols (HTTP, etc.)

API: Application Programming Interface. Different devices or software modules can share the same code. Code once, use it many times.

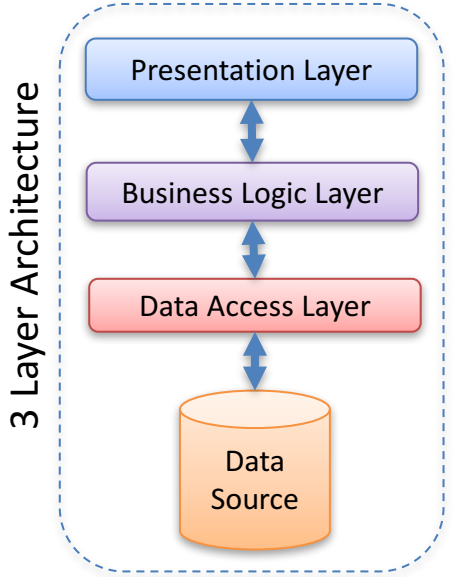
Network/Software Architecture

Client/Server Architecture 3 Layer Architecture SOA Architecture

Windows 8

iOS

RDC/TeamViewer

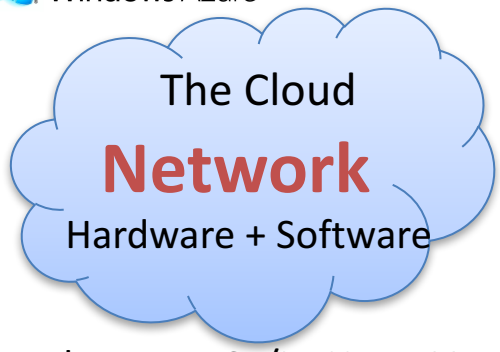


Virtualization!

VMware HyperV

amazon web services

Windows Azure



Port 8080

HTTP

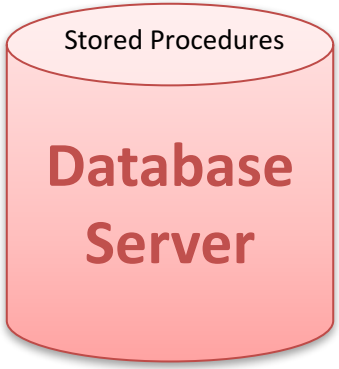
OPC Tunneller Software

Internet, Ethernet, TCP/IP, HTTP, VPN, Routers, Switches, Computers, Protocols, OSI, XML, SOAP, etc.

RDC/TeamViewer

Windows Server 2012

Port 1433



3-tier+WebService Architecture - Example

Installed on one or more Windows Servers in your LAN or in the Cloud

Microsoft IIS Internet Information Services

Web Server

Web Services

Business Logic

Stored Procedures

Data Source

Data Layer

Microsoft SQL Server

Team Foundation Server

TFS Cient

Presentation Logic

Visual Studio

Software Architecture

- Client-Server
- N-tier/3-tier
- SOA – Service Oriented Architecture
  - Web Services

- APIs
- etc.

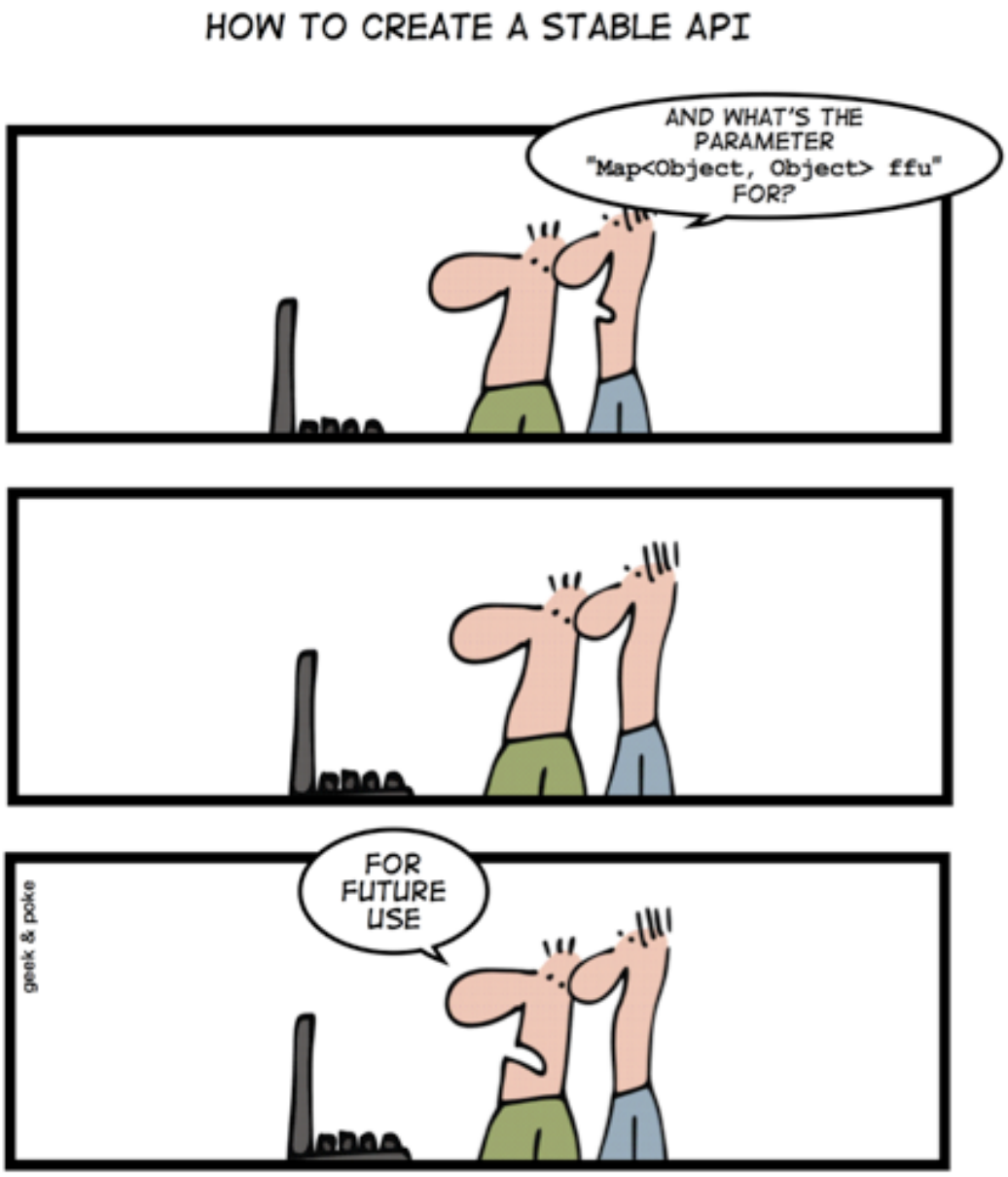
.NET vs. Java

http://www.youtube.com/watch?v=8Px-GHPxB4I

API

Hans-Petter Halvorsen

API


HOW TO CREATE A STABLE API
AND WHAT'S THE PARAMETER "Map<Object, Object> ffu" FOR?
FOR FUTURE USE
geek & poke

API

- API - Aplication Programming Interface
- A specification of how some software components should interact with each other.
- A library with functions, etc. you can use in your code
- Examples:
  - Windows API
  - Java API
- But you can also create your own API that you use internally in the team or expose to others

Software Design without APIs

Pros

- Fast to implement in small projects.
- Agile – can serve as a starting point for API design.
- No need to consider how code interfaces with other software.
- Can be appropriate for small “dead end” projects.

Cons

- Inappropriate for large projects.
- Code has a limited (as opposed to general) functionality.
- Code is not reusable.
- Code is hard to maintain/modify.
- Prone to errors and bugs.

Why a Good API is hard to Design

- Forces designer to anticipate future usage of code.
- Requirements are incomplete (may never be complete).
- Requires abstraction.
- Requires modularization.
- Requires skills in programming languages.
- Requires code rewrites – time consuming and labor intensive.

The Benefits of API Driven Design

When an API is used in a project, it

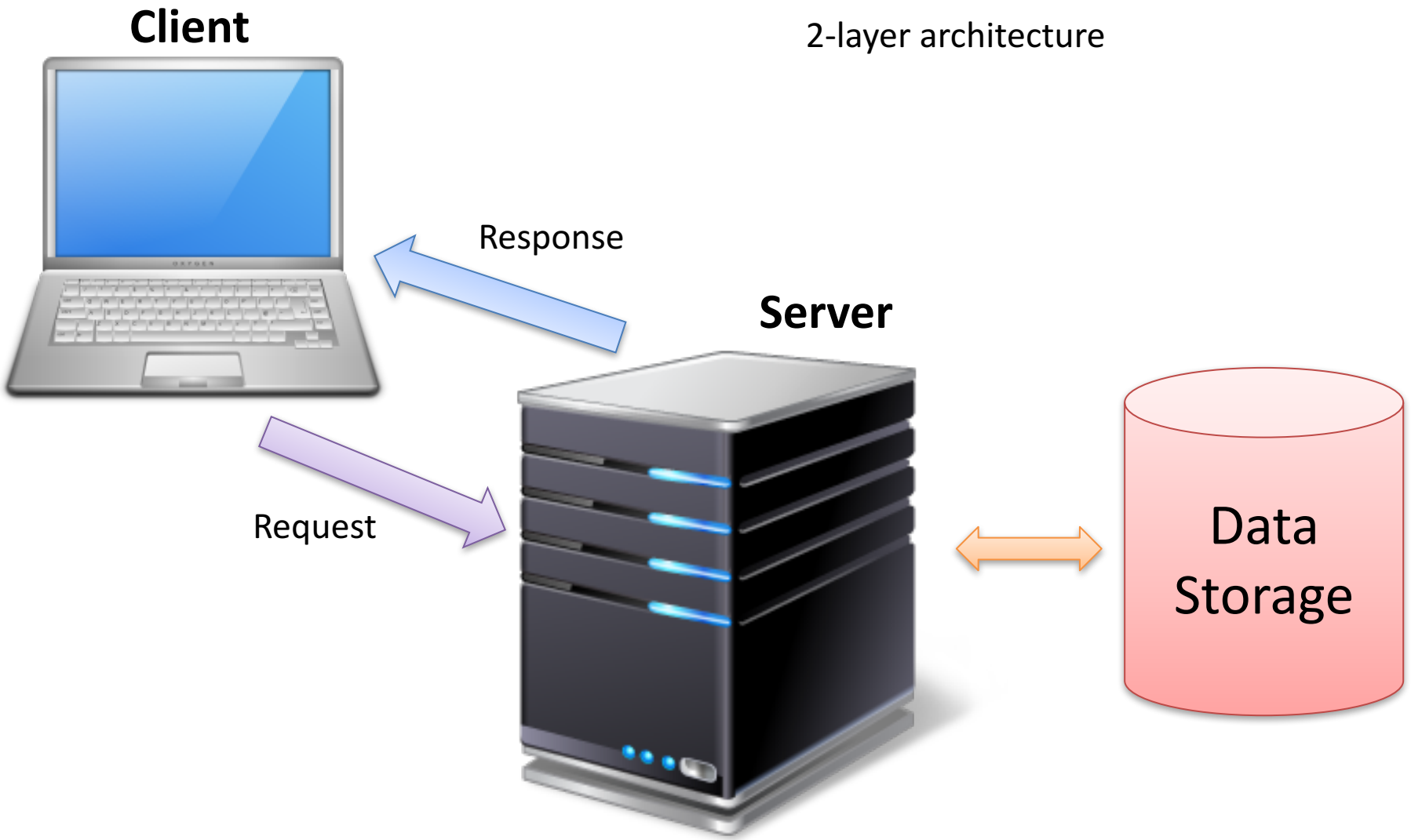
- Allows to focus on the project.
- Saves development time.
- Reduces errors and debugging.
- Facilitates modular design.
- Provides a consistent development platform.

➔ API driven design requires planning and programming skills. API driven design is costly initially, but it pays in the long run. So, obviously, creating APIs is good software practice in most cases.

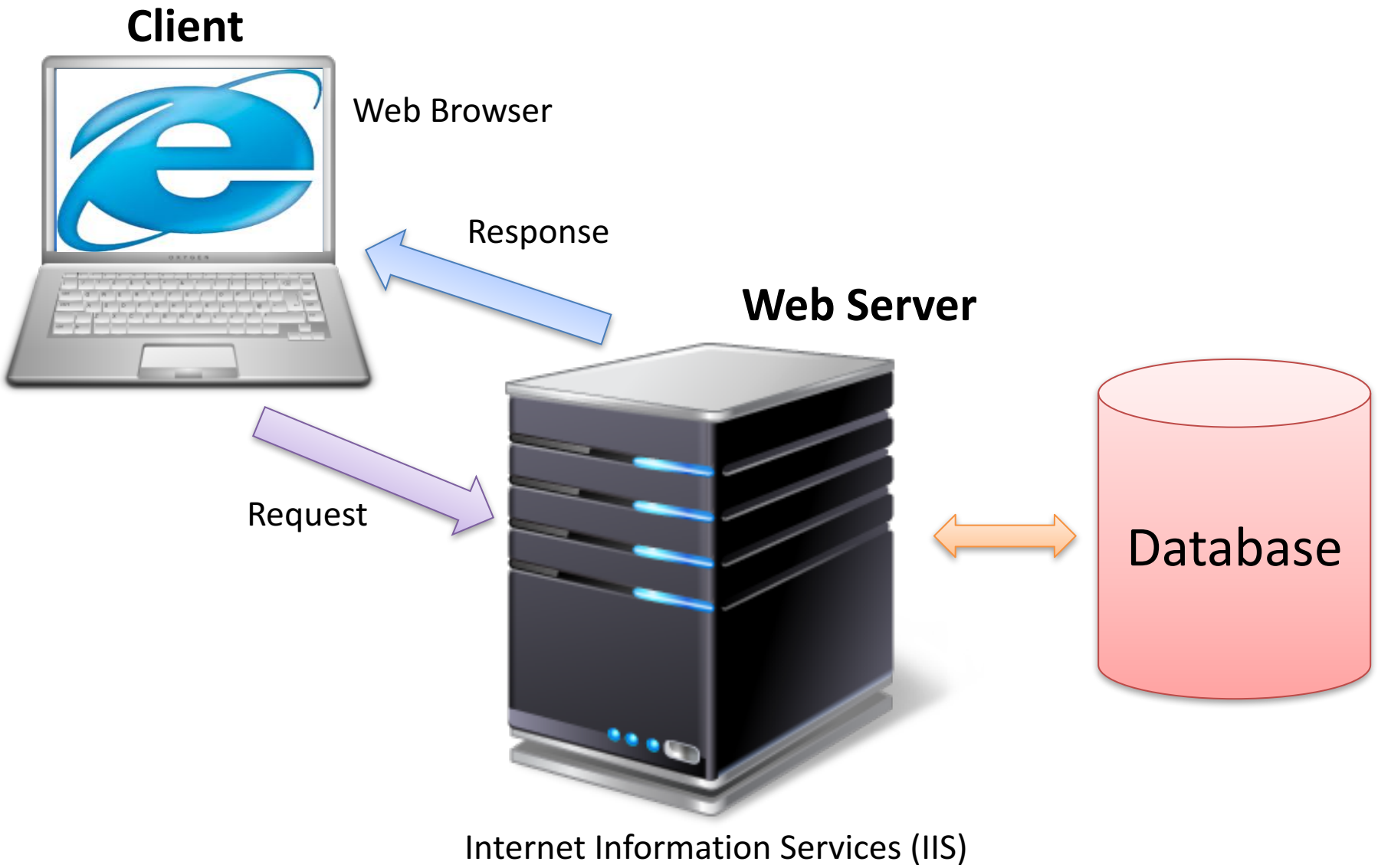
Client -Server

Hans-Petter Halvorsen

Client-Server



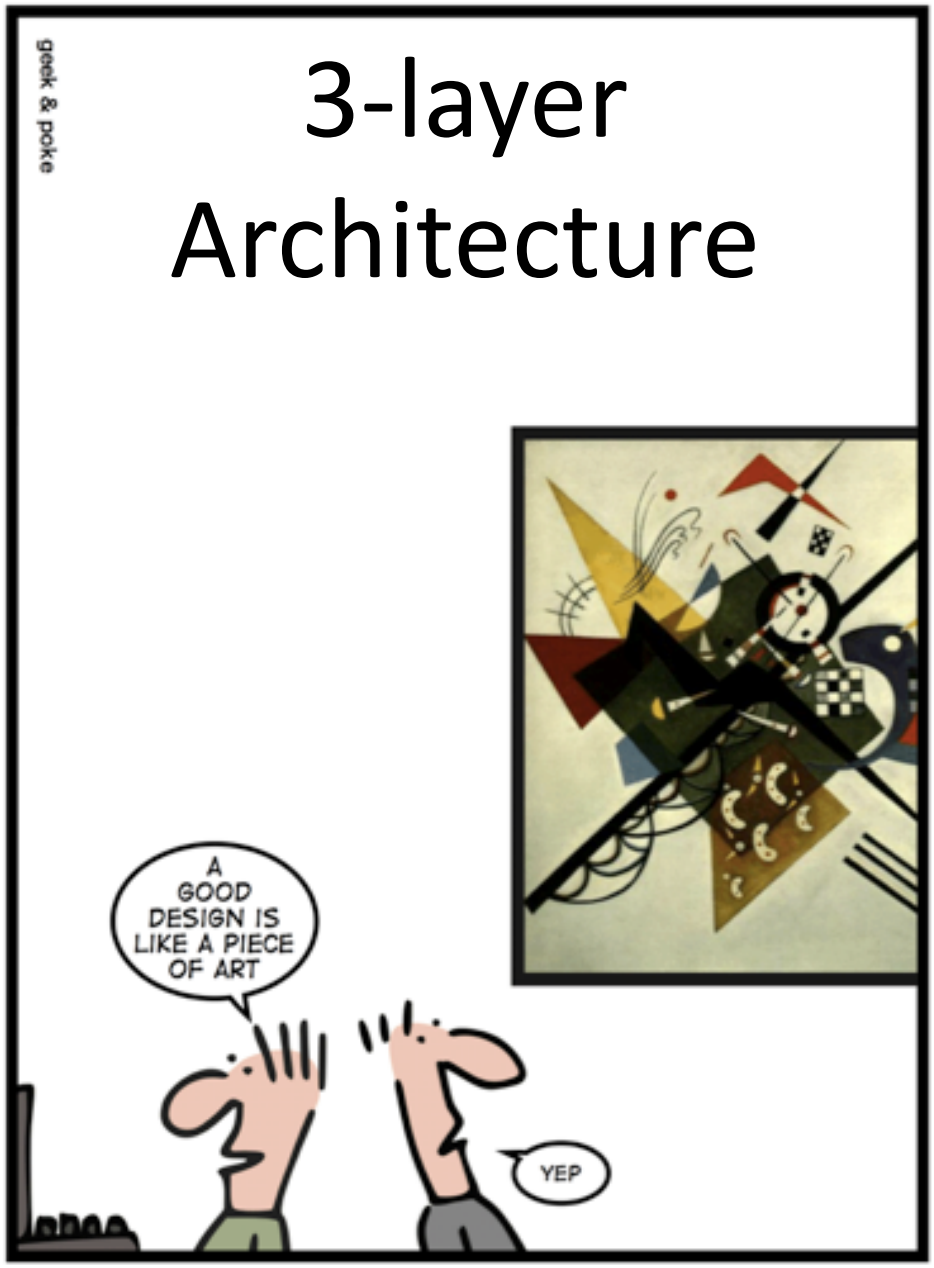
Client-Server Example

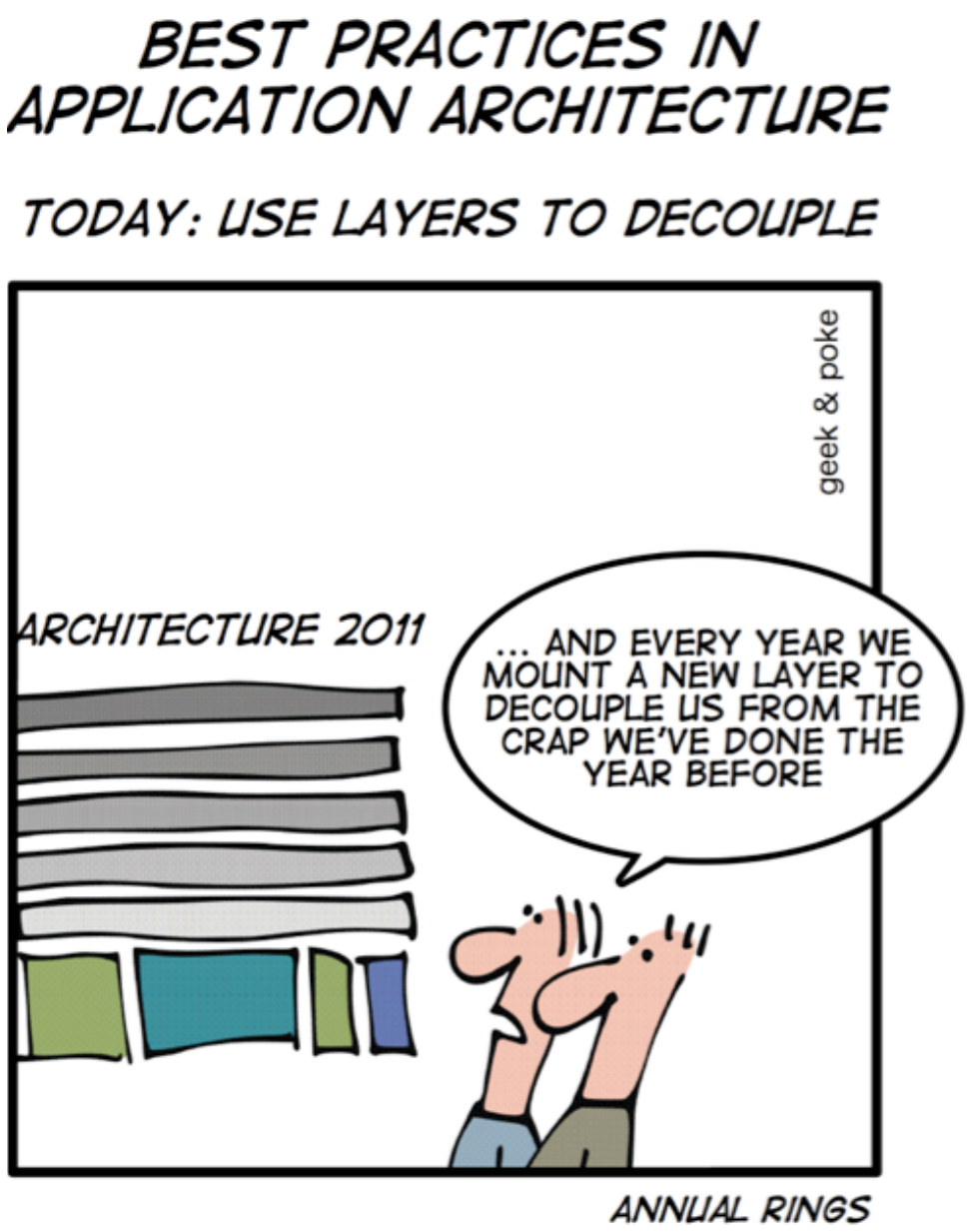


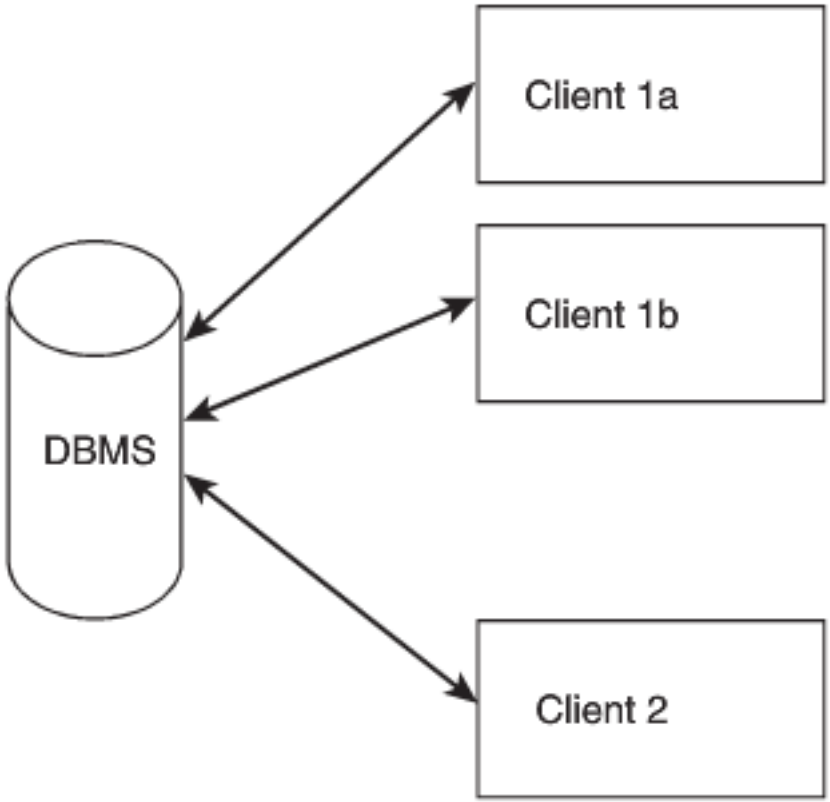
3-layer Architecture

Hans-Petter Halvorsen

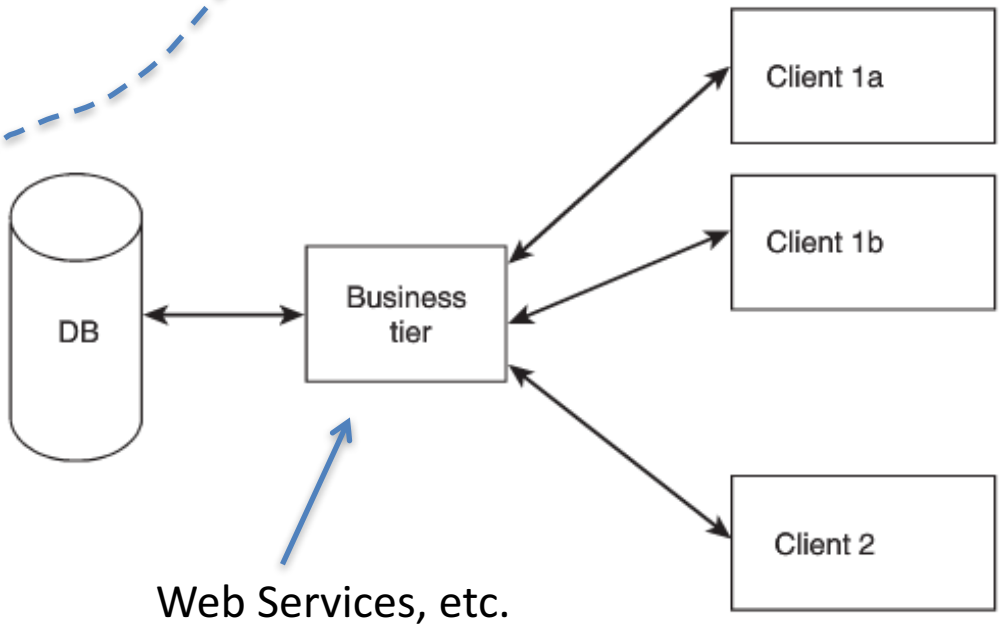
3-layer Architecture





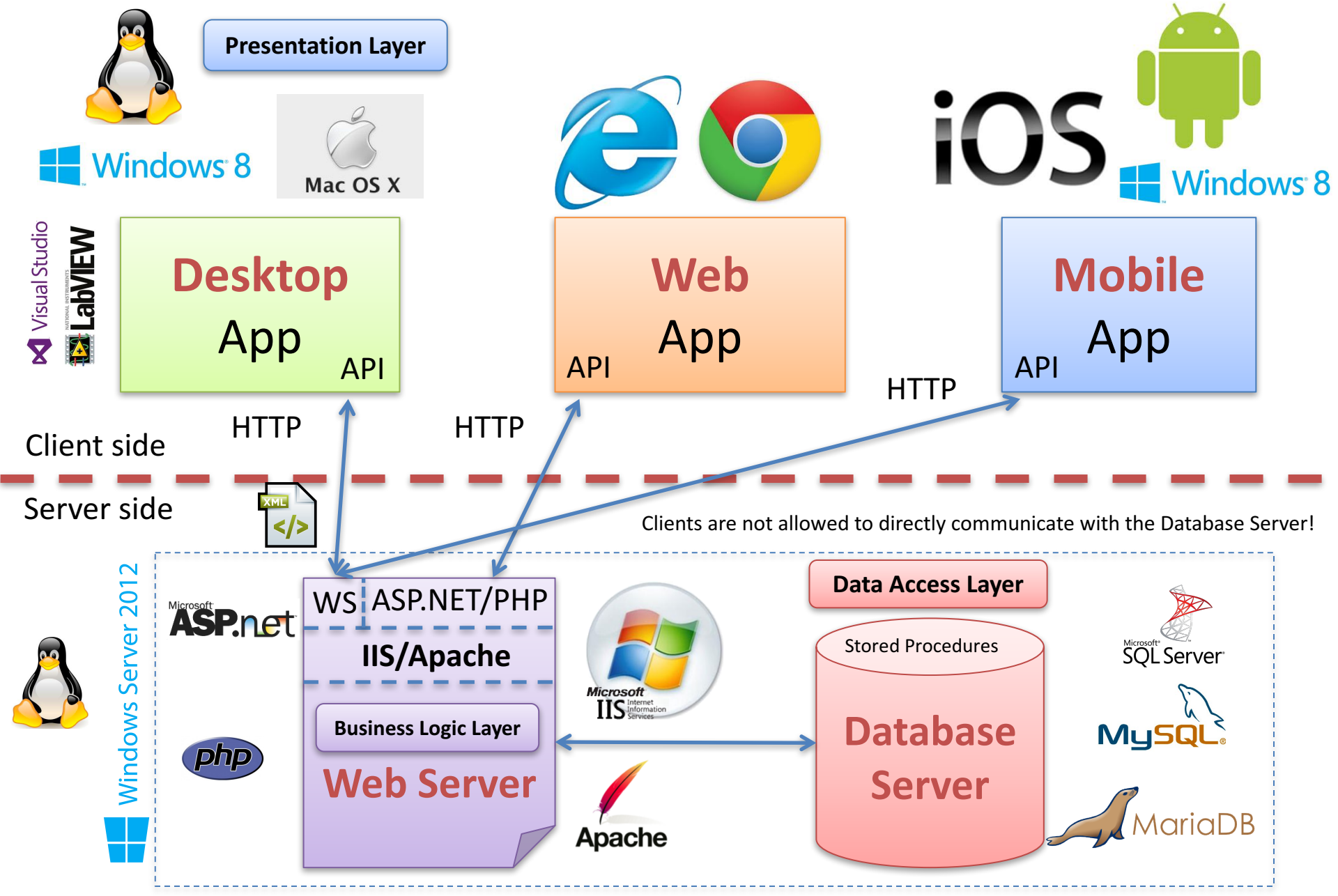


The database-centric style. Typically, the clients communicate directly with the database.

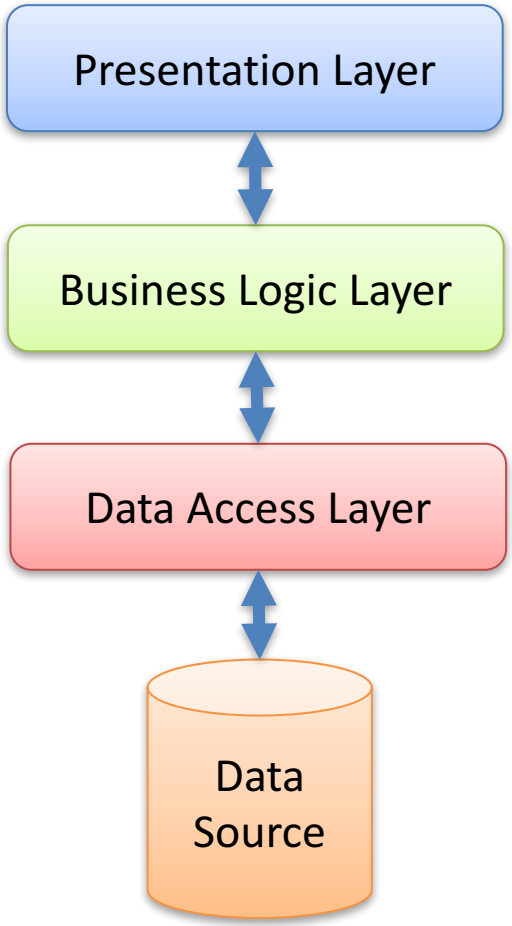


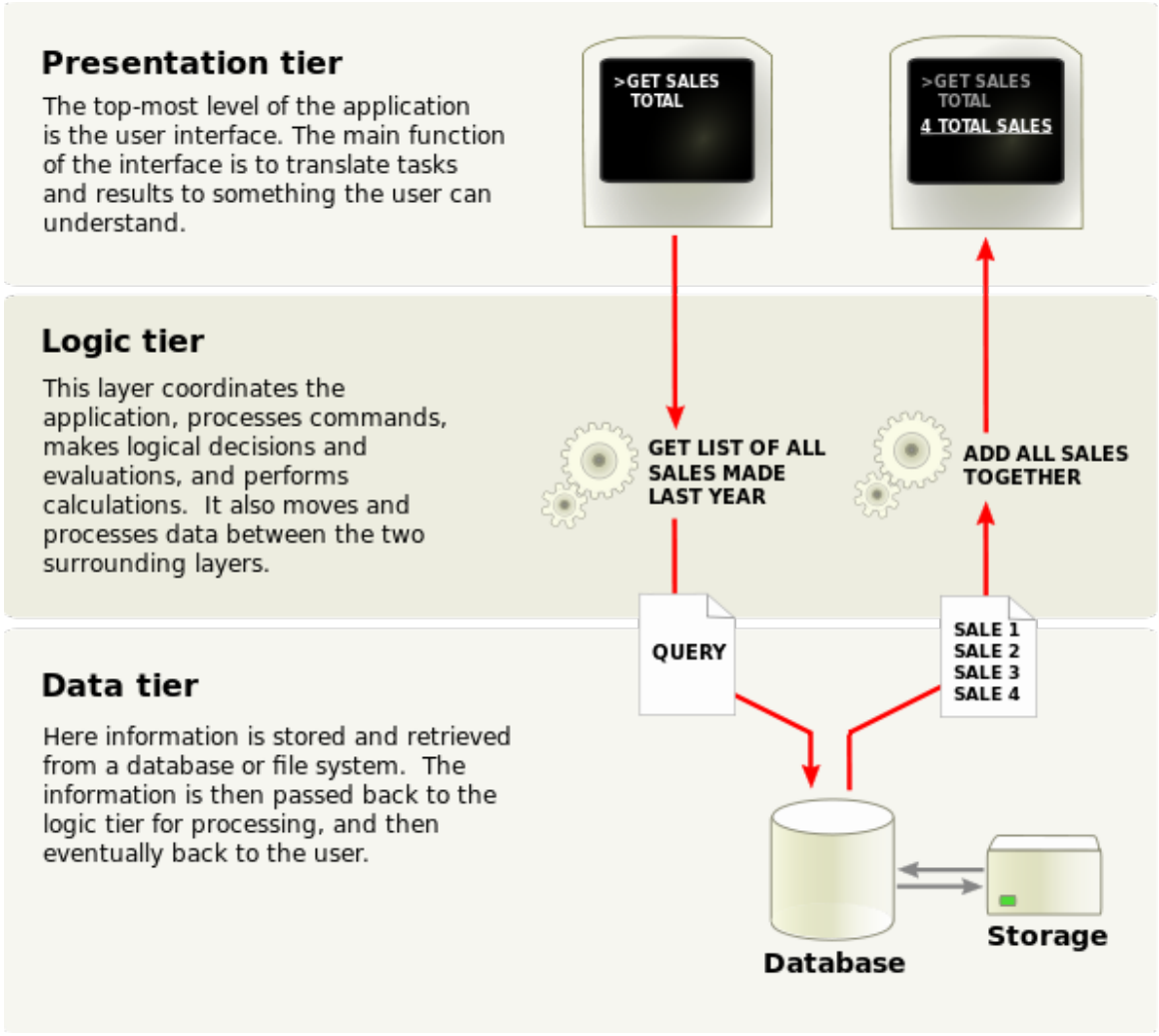
A three-tier style, in which clients do not connect directly to the database.

3 Layer Network/Software Architecture



3-tier Architecture





Example of 3 Layer Architecture

Clients

Visual Studio LabVIEW

Desktop App

- Visual Studio/C#
- WinForm/WPF
- LabVIEW

Client #1

Web Browser

Microsoft ASP.net

Web App

- ASP.NET
- PHP
- JavaScript, HTML

Client #2

Tablet or Smartphone

Mobile App

Client #3

Windows 8

- iOS (Xcode, Objective-C)
- Android (Eclipse, Java)
- Windows 8 (Visual Studio/C#)

Presentation Layer

HTTP

Web Services or OPC

Web Services

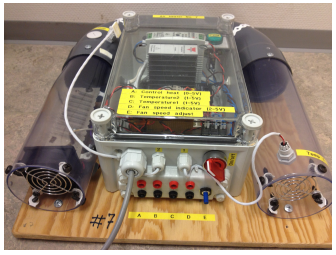
Process

I/O Module

USB-6008

Examples:

4 Tank



Air Heater

2 Tank

Weather Station

Server-side Logic

Web Server

Internet Information Services (**IIS**) - or Apache

3 Layer Architechture

Business Layer (Logic)

Data Layer(Logic)

Stored Procedures

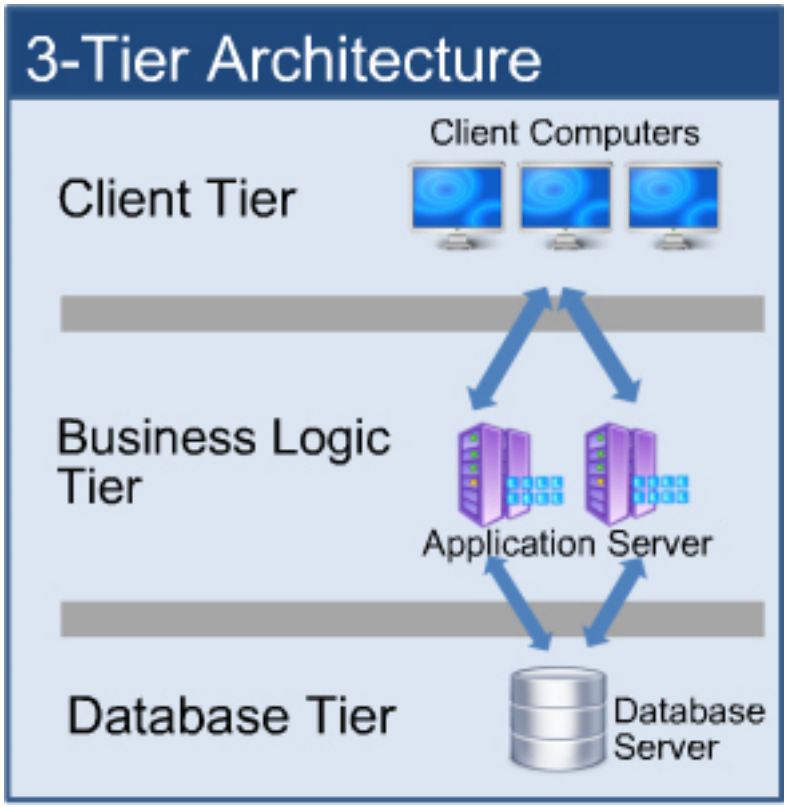
Database

Microsoft SQL Server

SQL Server (or MySQL, SQLite, Oracle)

Windows Server 2012

Server



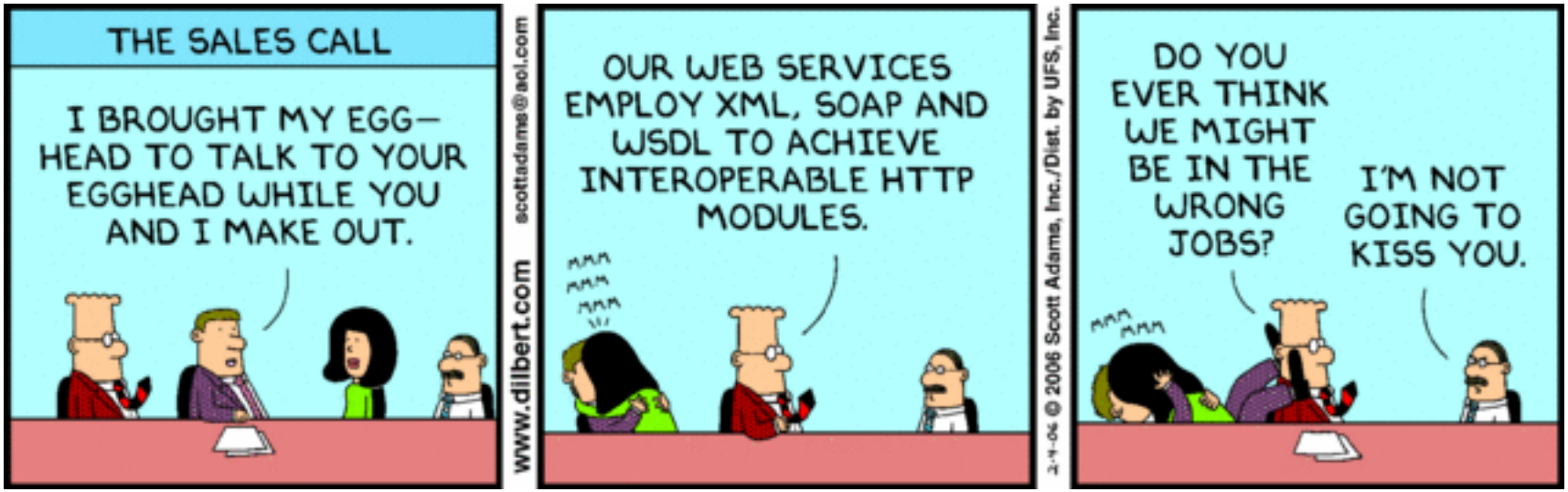
Client/Presentation Layer/Tier

Business/Logic Layer/Tier

Data Layer/Tier

Note! The different layers can be on the same computer (Logic Layers) or on different Computers in a network (Physical Layers)

Web Services

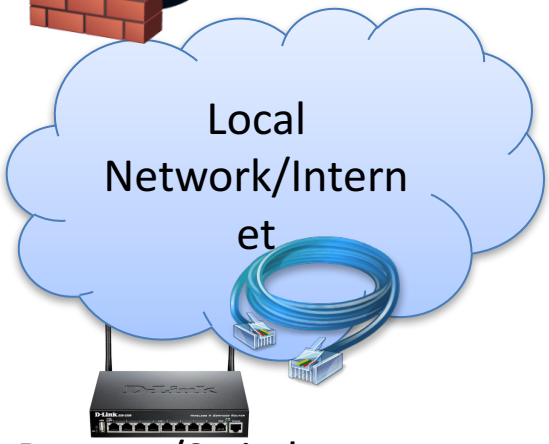


S. Adams. *Dilbert*. Available: http://dilbert.com

Hans-Petter Halvorsen

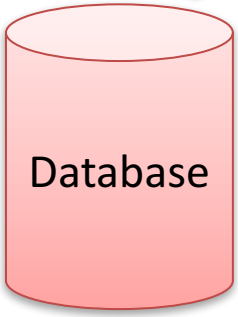
Problem

How to Share Data between Devices in a Network?



Routers/Switches, etc.

Clients



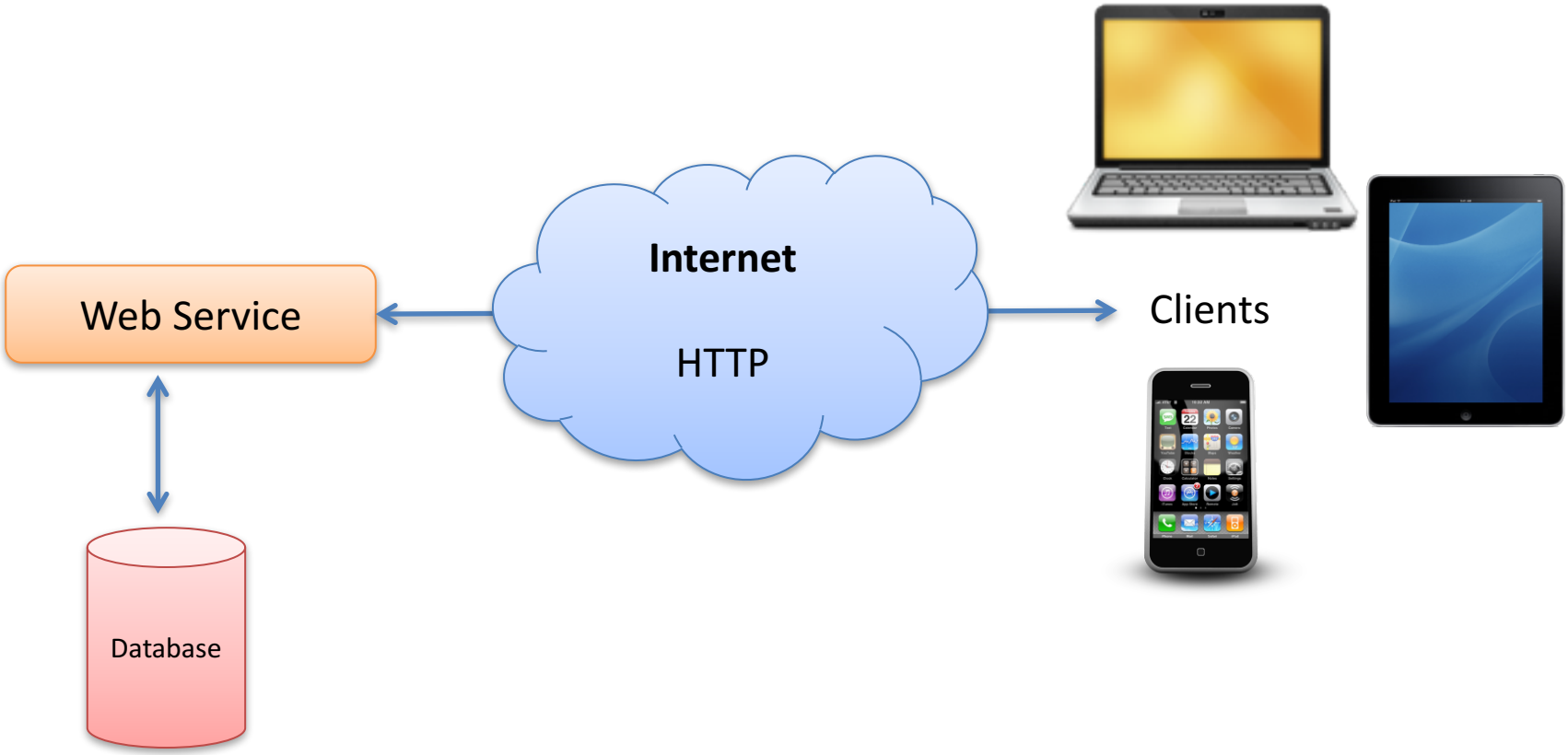
Problem

How to Share Data between Devices in a Network?

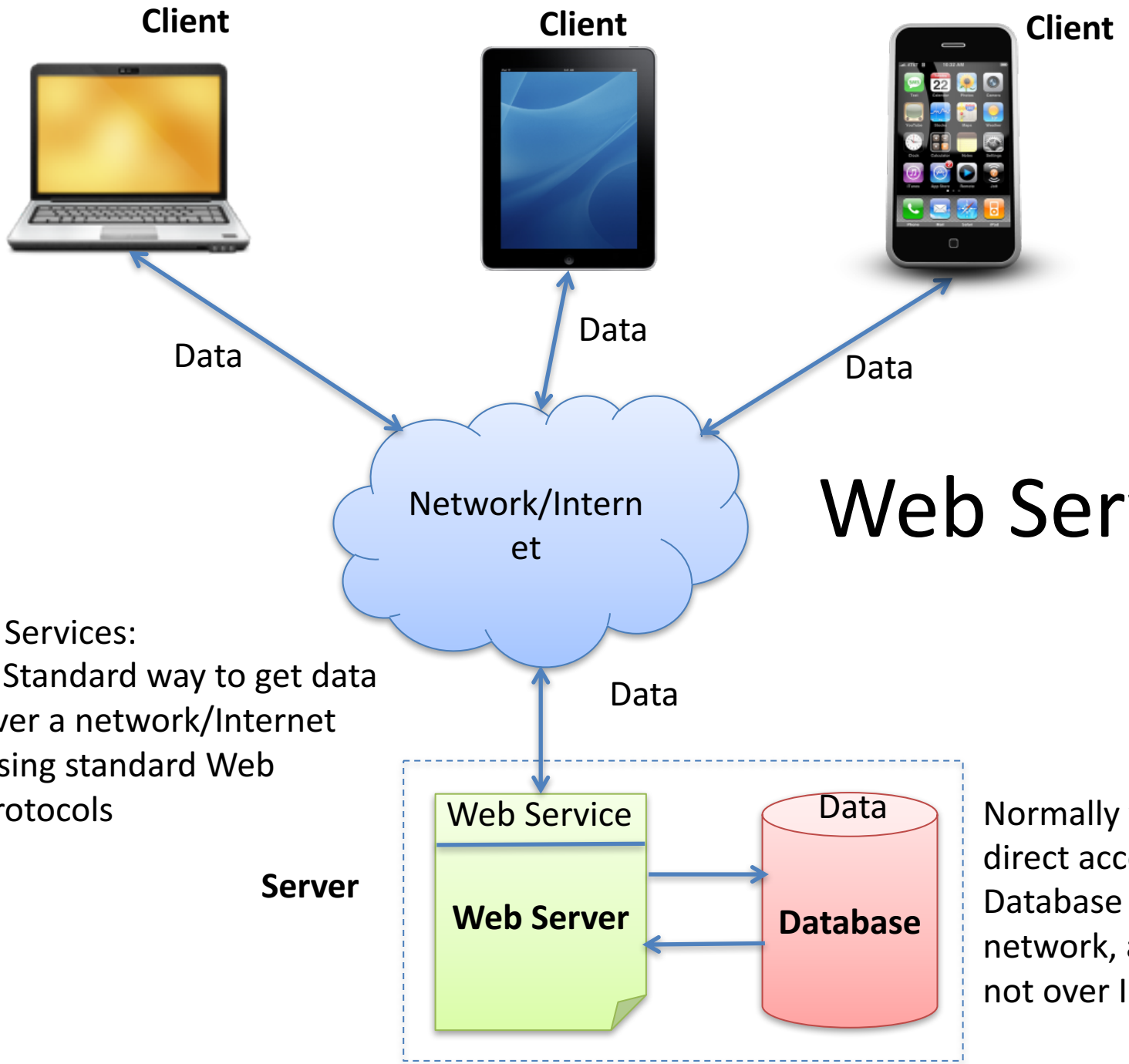
Direct Connection between the Database and the Clients that need the Data is normally not possible, due to security, compatibility issues, etc. (Firewalls, Hacker Attacks, etc.)

Direct Connection in a Local Network (behind the Firewall) is normally OK – but not over the Internet

Solution: Web Service



Web Services uses standard web protocols like HTTP, etc.
HTTP is supported by all Web Browser, Servers and many Programming Languages



Web Services

Web Services:

- A Standard way to get data over a network/Internet
- Using standard Web protocols

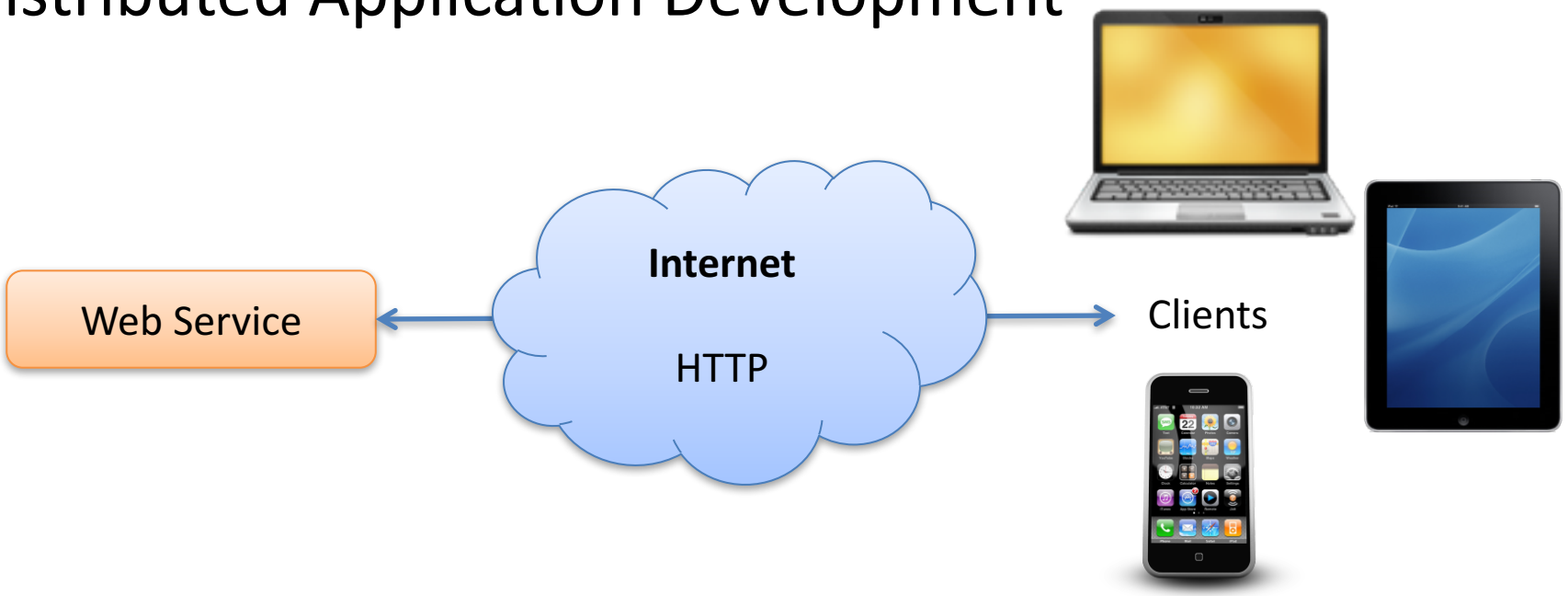
Normally you dont have direct access to a Database over a network, and espesially not over Internet

Web Services

- A Web service is a method of communications between two devices over the World Wide Web.
- Web API
- Standard defined by W3C
- Cross-platform
- Web Services can be implemented and used in most Programming Languages (C#/ASP.NET, PHP, LabVIEW, Objective-C, Java, ...)
- Uses standard Web technology (Web protocols)
  - HTTP, REST, SOAP, XML, WSDL, JSON, ...

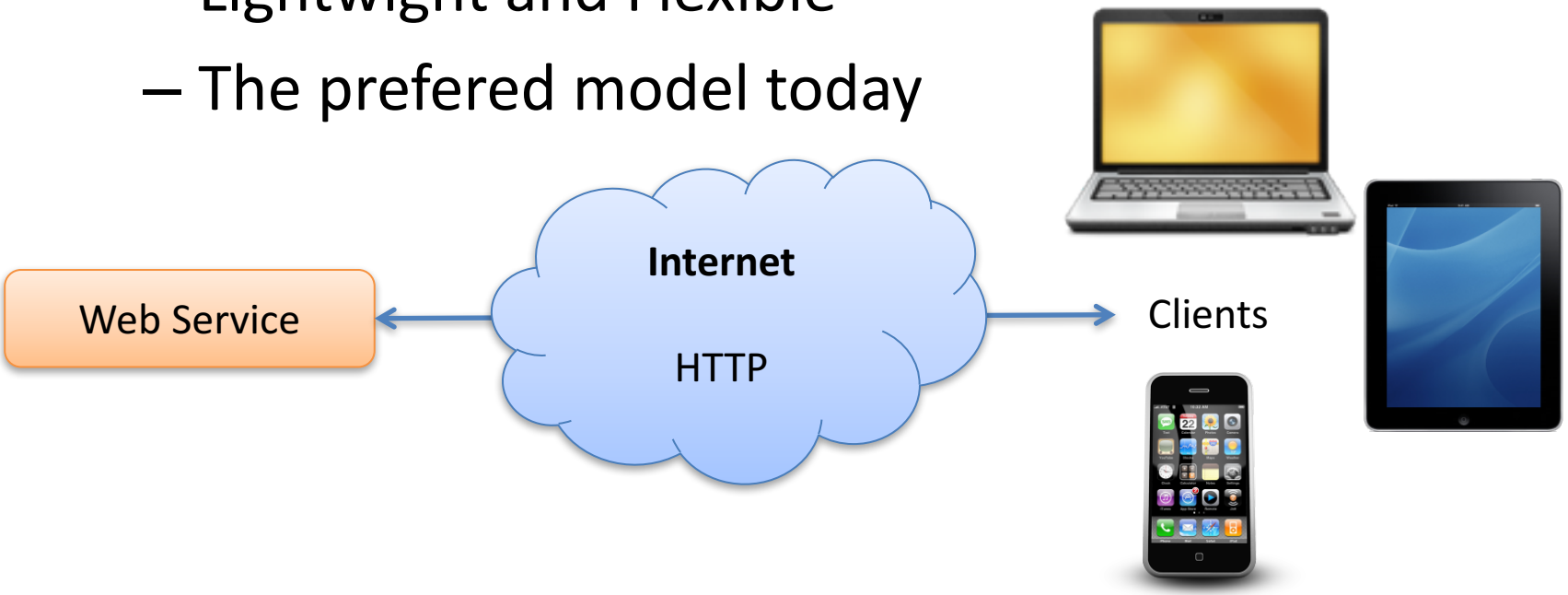
Why Web Service?

- Today Web Services have been very popular
- Easy Data sharing over Internet
- Platform-independent Communication
- Makes it possible of integration of different systems and platforms
- Distributed Application Development



Web Services

- Web Services 1.0: Uses SOAP
  - "Complex"
- Web Services 2.0: Uses REST
  - Less Complex than using SOAP
  - Lightwight and Flexible
  - The prefered model today



References

- I. Sommerville, *Software Engineering*: Pearson, 2010.
- E. J. Braude and M. E.Bernstein, *Software Engineering. Modern Approaches*, 2 ed.: Wiley, 2011.
- Wikipedia. (2013). *SOA*. Available: http://en.wikipedia.org/wiki/Service-oriented_architecture
- NTNU. (2013). *TDT4140 Systemutvikling*. Available: http://www.ntnu.no/studier/emner/TDT4140
- UiO. (2013). *INF1050 - Systemutvikling*. Available: http://www.uio.no/studier/emner/matnat/ifi/INF1050/
- O. Widder. (2013). *geek&poke*. Available: http://geek-and-poke.com
- B. Lund. (2013). *Lunch*. Available: http://www.lunchstriper.no, http://www.dagbladet.no/tegneserie/lunch/
- S. Adams. *Dilbert*. Available: http://dilbert.com

Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway

www.usn.no

E-mail: hans.p.halvorsen@hit.no

Blog: http://home.hit.no/~hansha/