**Introduction**

Perbedaan SOA dan SAAS.

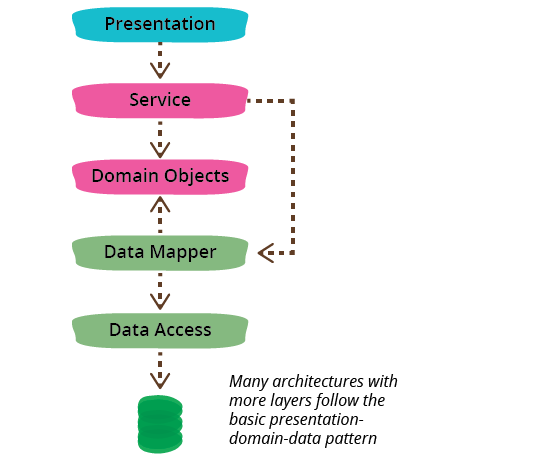
SOA adalah sebuah bentuk arsitektur yang menerapkan konsep service-oriented (berorientasi service). Pendekatan SOA adalah pendekatan penyelesaian masalah yang besar dengan membaginya. SAAS adalah bisnis system dimana semua resource di hosted di cloud. Enterprise applications include: patient records, shipping tracking, cost analysis, automated billing systems, supply chain, enterprise resource planning (ERP). Enterprise applications don’t include automobile fuel injection, word processors, elevator controllers, chemical plant controllers, telephone switches, operating systems, compilers, and games. Response time is the amount of time it takes for the system to process a request from the outside. Responsiveness is about how quickly the system acknowledges a request as opposed to processing it.Latency is the minimum time required to get any form of response, even if the work to be done is nonexistent. Throughput is how much stuff you can do in a given amount of time. For enterprise applications a typical measure is transactions per second (tps). Load is a statement of how much stress a system is under, which might be measured in how many users are currently connected to it. Efficiency is performance divided by resources. A system that gets 30 tps on two CPUs is more efficient than a system that gets 40 tps on four identical CPUs. Scalability is a measure of how adding resources (usually hardware) affects performance. A scalable system is one that allows you to add hardware and get a commensurate performance improvement, such as doubling how many servers you have to double your throughput.

**Layering**

Layering is one of the most common techniques that software designers use to break apart a complicated software system. OSI 7 layer: physical layer, Data link layer, Network Layer, Transport Layer, Session Layer, Presentation Layer, Application Layer. Layered System In this scheme the higher layer uses various services defined by the lower layer, but the lower layer is unaware of the higher layer. How it works? 1. The user calls a function on an object in the upper layer. 2. This object calls functions in the layer below. 3. These functions in turn approach the layer below and the layer above. Etc. 4. Eventually the function is performed and control is returned to the user. All this is usually done sequentially, in the same thread. Benefit 1. You can understand a single layer as a coherent whole without knowing much about the other layers. 2. You can substitute layers with alternative implementations of the same basic services. 3. You minimize dependencies between layers. 4. Layers make good places for standardization. 4. Once you have a layer built, you can use it for many higher-level services. Thus, TCP/IP is used by FTP, telnet, SSH, and HTTP. The downside 1. Layers encapsulate some, but not all, things well. The classic example of this in a layered enterprise application is adding a field that needs to display on the UI, must be in the database, and thus must be added to every layer in between (cascading changes). 2. Extra layers can harm performance. The three layers 1. presentation layer, knows about handling http requests and rendering HTML. Display information to the user and to interpret commands from the user into actions upon the domain and data source. 2. domain layer, a business logic layer that contains validations and calculations. Responsibility: This is the work that this application needs to do for the domain you're working with 3. data layer, data access layer that sorts out how to manage persistent data in a database or remote services. Communication with Databases, Communication with other applications, Communication with Messaging systems, Communication with Transaction managers.

The domain and data source should never be dependent on the presentation! What does it mean? There should be no subroutine call from the domain or data source code into the presentation code.

Benefit: It will be easier for us to substitute with other/different presentations.



Choosing where to run your layer. Simple case: Run everything on servers ! Advantages: 1. Every thing is easy to upgrade and fix. 2. You don't have to worry about deployment to many desktops. 3. You don't have to worry about compatibilities with other desktop software. Problems: Problems on responsiveness and disconnected operations. Alternatives: we can look at the options layer by layer. Data Source . 1. The data source pretty much always runs only on servers. 2. The exception is where you might duplicate server functionality onto a suitably powerful client, usually when you want disconnected operation. Alternatives: we can look at the options layer by layer. Presentation. 1. It depends mostly on what kind of user interface you want 2. Running a rich client pretty much means running the presentation on the client. 3. Running a Web interface pretty much means running on the server. Alternatives: we can look at the options layer by layer. Domain Logic 1. You can run business logic all on the server or all on the client, or you can split it. 2. Again, all on the server is the best choice for ease of maintenance 3. The demand to move it to the client is for either responsiveness or disconnected use.