

Sryas Software Architecture

Overview

We have a structured process of designing and building software, which comprises of software components, the externally visible properties of those components, and the relationships between them. Technical architecture is a part of software architecture, which focuses on how to deal with certain aspects of the software engineering process. It allows us to design better systems by:

- Meeting system requirements and objectives
- Enabling flexible partitioning of the system: A good architecture enables flexible distribution of the system by allowing the system and its constituent applications to be partitioned among processors in many different ways without having to redesign the distributable component parts.
- Careful attention to the distribution potential of components early in the architectural design process
- Architecture can help minimize the costs of maintaining and evolving a given system over its entire lifetime by anticipating the main kinds of changes that will occur in the system, ensuring that the system's overall design will facilitate such changes, and localizing as far as possible the effects of such changes on design documents, code, and other system work products.
- minimization and control of subsystem interdependencies
- An architecture may be designed to enable and facilitate the (re)use of certain existing components, frameworks, class libraries, legacy or third-party applications, etc

Architecture Description Language and Views

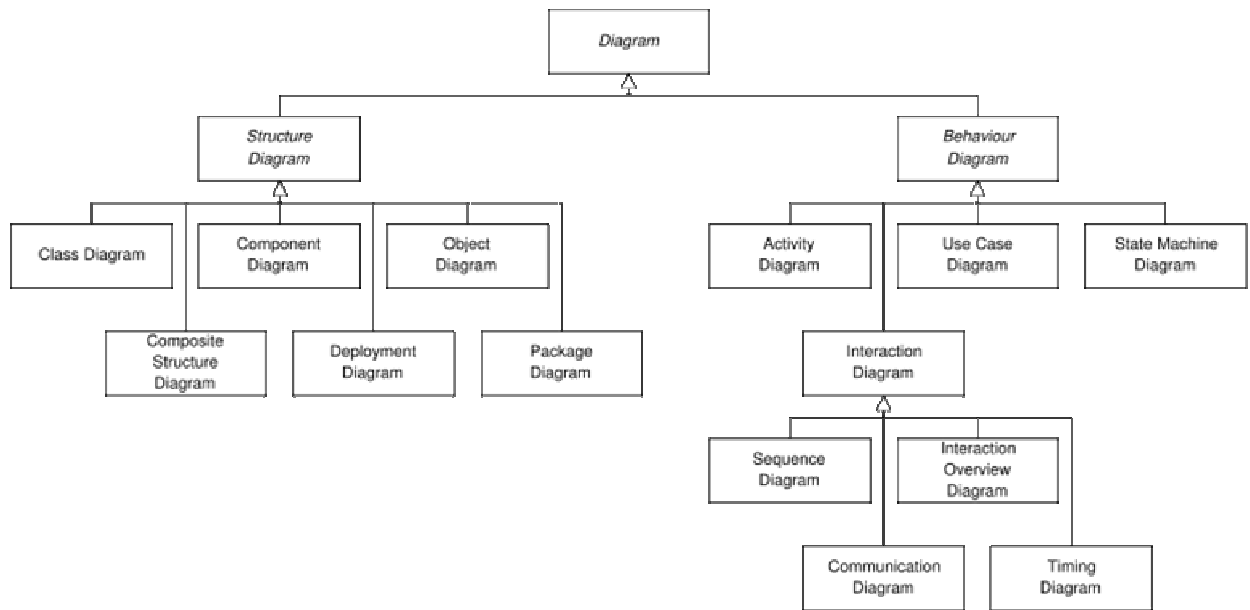
They are languages used to describe software or system architecture. Several different ADL have been developed by different organizations. Some of the inherent features of this language are:

- Be suitable for communicating an architecture to all interested parties
- Support the tasks of architecture creation, refinement and validation
- Provide a basis for further implementation
- Provide the ability to represent most of the common architectural styles
- Support analytical capabilities or provide quick generating prototype implementations
- Graphical syntax with often a textual form and a formally defined syntax and semantics
- Features for modeling distributed systems
- Ability to represent hierarchical levels of detail including the creation of substructures by instantiating templates
- Support the specification of different architectural styles. Few handle object oriented class inheritance or dynamic architectures
- For modeling distributed systems
- Support for automatic generation of software systems

Software architecture is commonly organized in views, which are analogous to the different types of blueprints made in building architecture. Some of common view is:

- Functional/logic view
- Code view
- Development/structural view
- Concurrency/process/thread view
- Physical/deployment view
- User action/feedback view

The UML (unified modeling tool) was established as a standard *"to model systems (and not just software),"* and thus applies to views about software architecture. The Unified Modeling Language (UML) is a standardized specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system



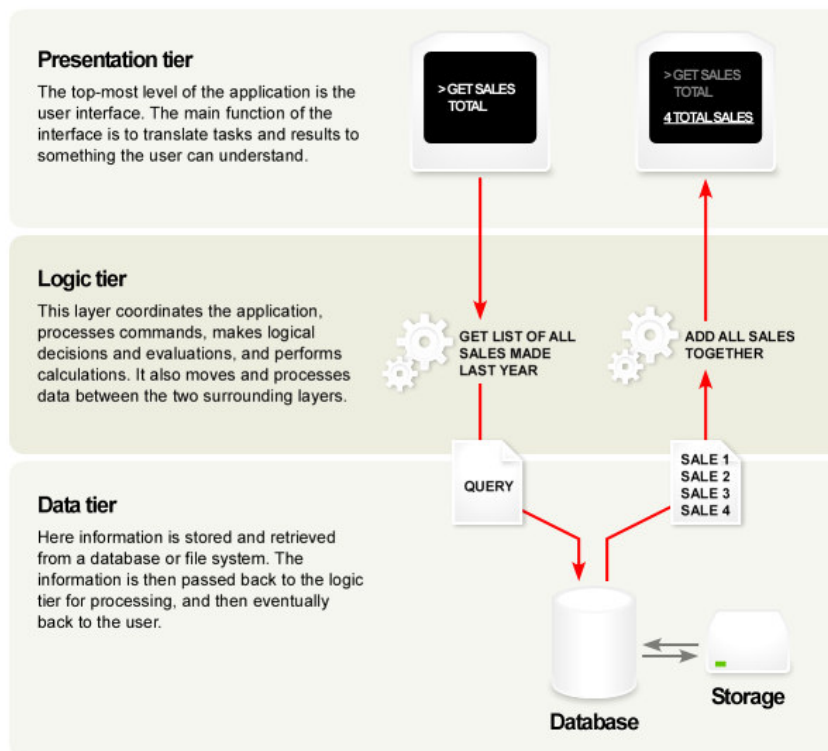
Architecture Styles / Patterns

There are different ways, a system could be designed, developed and deployed. They all follow some design patterns and software architectures. The designs have evolved over a period of time. Few of them have been discussed below

Three-tire Architecture

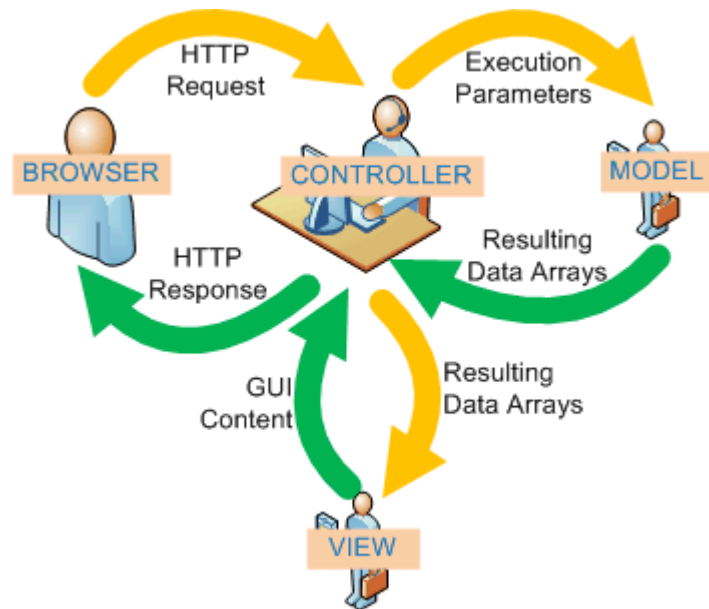
Three-tier' is a client-server architecture in which the user interface, functional process logic ("business rules"), data storage and data access are developed and maintained as independent modules, most often on separate platforms. The three-tier model is considered to be software architecture and a software design pattern. Any of the tiers can be independently upgraded or replaced as requirements or changes in technology. The three-tire architecture comprises of

- Presentation Tier
- Application Tier / Logic Tier / Business Logic Tier
- Data Tier



Model View Controller

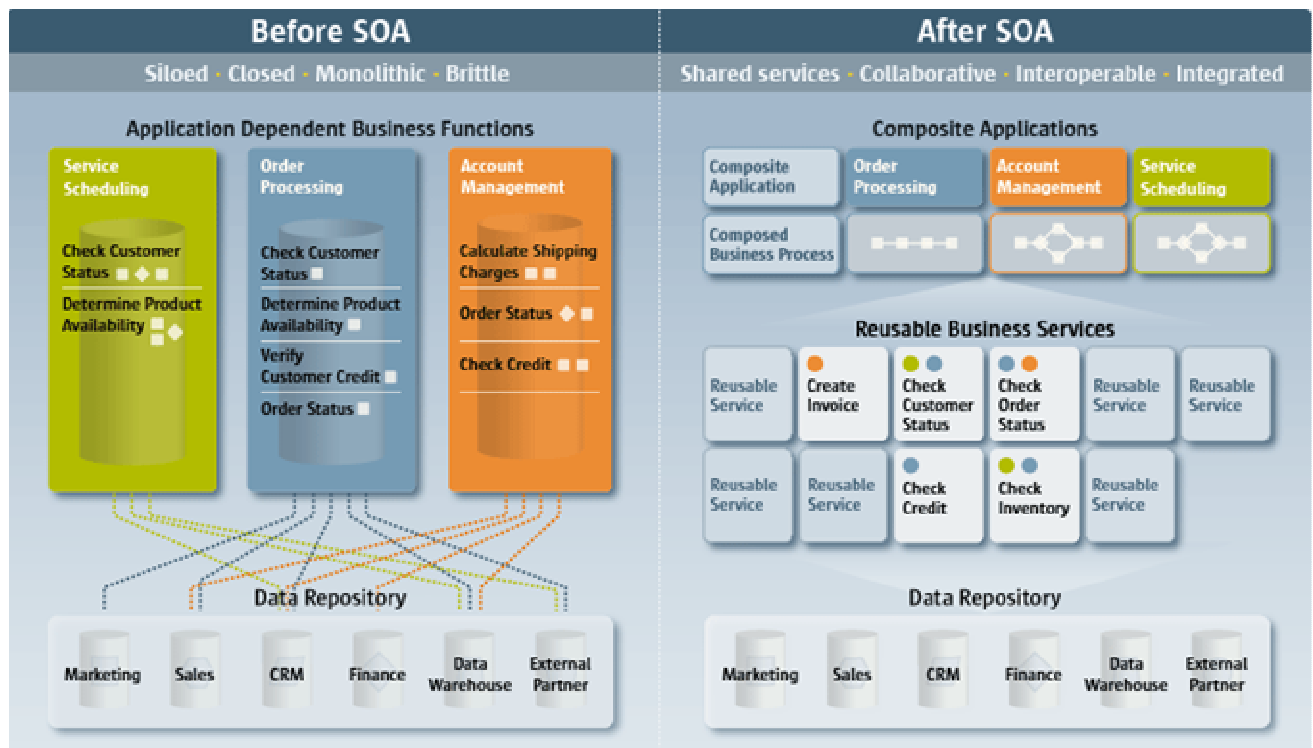
In complex computer applications that present a large amount of data to the user, a developer often wishes to separate data (model) and user interface (view) concerns, so that changes to the user interface will not affect data handling, and that the data can be reorganized without changing the user interface. The model-view-controller solves this problem by decoupling data access and business logic from data presentation and user interaction, by introducing an intermediate component: the controller.



Service Oriented Architecture

Service Oriented Architecture (SOA) is an evolution of distributed computing and modular programming. Services are relatively large, intrinsically unassociated units of functionality, which have no calls to each other embedded in them. They typically implement functionalities most humans would recognize as a service, such as filling out an online application for an account, viewing an online bank statement, or placing an online book or airline ticket order. Instead of services embedding calls to each other in their source code, protocols are defined which describe how one or more services can talk to each other. This architecture then relies on a business process expert to link and sequence services, in a process known as orchestration, to meet a new or existing business system requirement.

Underlying and enabling all of this is metadata which is sufficient to describe not only the characteristics of these services, but also the data that drives them. XML has been used extensively in SOA to create data which is wrapped in a nearly exhaustive description container. Analogously, the services themselves are typically described by WSDL, and communications protocols by SOAP. SOA services are therefore loosely coupled, in contrast, for example, to the functions a linker binds together to form an executable, a DLL, or an assembly. SOA services also run in "safe" wrappers such as the .NET environment, which manages memory allocation and reclamation, allows ad-hoc and late binding, and some degree of indeterminate data typing



Requirements for SOA

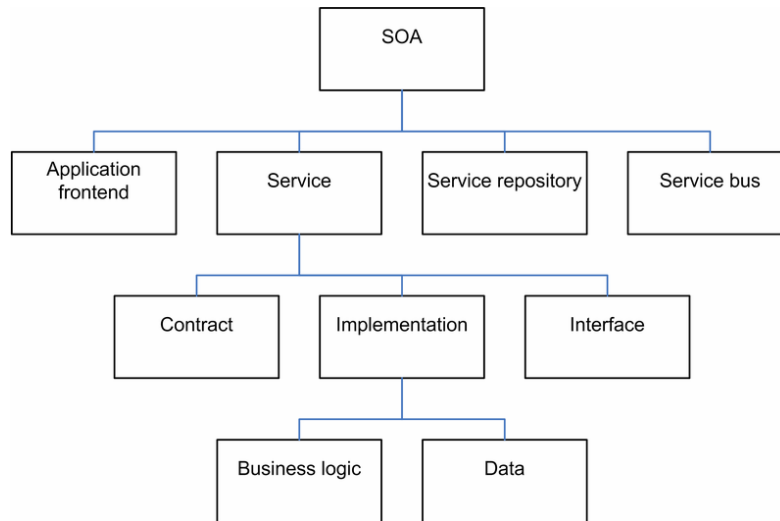
- Interoperability between different systems and programming languages
- Integration between applications on different platforms
- Clear and unambiguous description language
- To allow a convenient integration at design time or even system run time
- Search mechanism required to retrieve suitable services
- Services should be classified as computer-accessible, hierarchical or taxonomies based on what the services in each category do and how they can be invoked.

Web service approach to SOA

Web services implement a service-oriented architecture. A major focus of Web services is to make functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages. These services can be new applications or just wrapped around existing legacy systems to make them network-enabled. Every SOA building blocks serves the following purpose

- Service provider - The service provider creates a Web service and possibly publishes its interface and access information to the service registry. Each provider must decide which services to expose, how to make trade-offs between security and easy availability, how to price the services, or, if they are free, how to exploit them for other value
- Service broker - The service broker, also known as service registry, is responsible for making the Web service interface and implementation access information available to any potential service requestor. The implementer of the broker decides about the scope of the broker. Public brokers are available through the Internet, while private brokers are only accessible to a limited audience, for example, users of a company intranet. The Universal Description Discovery and Integration (UDDI) specification defines a way to publish and discover information about Web services.
- Service requestor - The service requestor or Web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its Web services

SOA Elements



SOA Integration

An SOA consists of service and event interfaces to both legacy systems and newer application that are built on top of platforms like BEA WebLogic, IBM WebSphere, Oracle 10g and Sun Java Enterprise System. A properly designed EAI infrastructure relies on a SOA to create reusable integrated business processes. Developers and non-developers create these processes using graphical tools that model business processes. The SOA serves as the foundation for the integrated enterprise as it shields developers from technical complexities becomes the common platform for business functionality.

