WIKIPEDIA

Multitier architecture

In software engineering, **multitier architecture** (often referred to as ***n*-tier architecture**) or **multilayer architecture** is a client–server architecture in which presentation, application processing and data management functions are physically separated. The most widespread use of multitier architecture is the **three-tier architecture**.

N-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application. A three-tier architecture is typically composed of a *presentation* tier, a *logic* tier, and a *data* tier.

While the concepts of layer and tier are often used interchangeably, one fairly common point of view is that there is indeed a difference. This view holds that a *layer* is a logical structuring mechanism for the elements that make up the software solution, while a *tier* is a physical structuring mechanism for the system infrastructure.[1][2] For example, a three-layer solution could easily be deployed on a single tier, such as a personal workstation.[3]

Contents

Layers

The "Layers" architectural pattern has been described in various publications.[4]

Common layers

In a logical multilayer architecture for an information system with an object-oriented design, the following four are the most common:

- **Presentation layer** (a.k.a. UI layer, view layer, presentation tier in multitier architecture)
- **Application layer** (a.k.a. service layer[5][6] or GRASP Controller Layer [7])

- **Business layer** (a.k.a. business logic layer (BLL), domain logic layer)
- **Data access layer** (a.k.a. persistence layer, logging, networking, and other services which are required to support a particular business layer)

The book *Domain Driven Design* describes some common uses for the above four layers, although its primary focus is the domain layer.[8]

If the application architecture has no explicit distinction between the business layer and the presentation layer (i.e., the presentation layer is considered part of the business layer), then a traditional client-server (two-tier) model has been implemented.

The more usual convention is that the application layer (or service layer) is considered a sublayer of the business layer, typically encapsulating the API definition surfacing the supported business functionality. The application/business layers can, in fact, be further subdivided to emphasize additional sublayers of distinct responsibility. For example, if the model–view–presenter pattern is used, the presenter sublayer might be used as an additional layer between the user interface layer and the business/application layer (as represented by the model sublayer).

Some also identify a separate layer called the business infrastructure layer (BI), located between the business layer(s) and the infrastructure layer(s). It's also sometimes called the "low-level business layer" or the "business services layer". This layer is very general and can be used in several application tiers (e.g. a CurrencyConverter).[9]

The infrastructure layer can be partitioned into different levels (high-level or low-level technical services).[9] Developers often focus on the persistence (data access) capabilities of the infrastructure layer and therefore only talk about the persistence layer or the data access layer (instead of an infrastructure layer or technical services layer). In other words, the other kind of technical services are not always explicitly thought of as part of any particular layer.

A layer is on top of another, because it depends on it. Every layer can exist without the layers above it, and requires the layers below it to function. Another common view is that layers do not always strictly depend on only the adjacent layer below. For example, in a relaxed layered system (as opposed to a strict layered system) a layer can also depend on all the layers below it.[4]

Three-tier architecture

Three-tier architecture is a client-server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms.[10] It was developed by John J. Donovan in Open Environment Corporation (OEC), a tools company he founded in Cambridge, Massachusetts.

Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently in response to changes in requirements or technology. For example, a change of operating system in the *presentation tier* would only affect the user interface code.

Typically, the user interface runs on a desktop PC or workstation and uses a standard graphical user interface, functional process logic that may consist of one or more separate modules running on a workstation or application server, and an RDBMS on a database server or mainframe that contains the computer data storage logic. The middle tier may be multitiered itself (in which case the overall

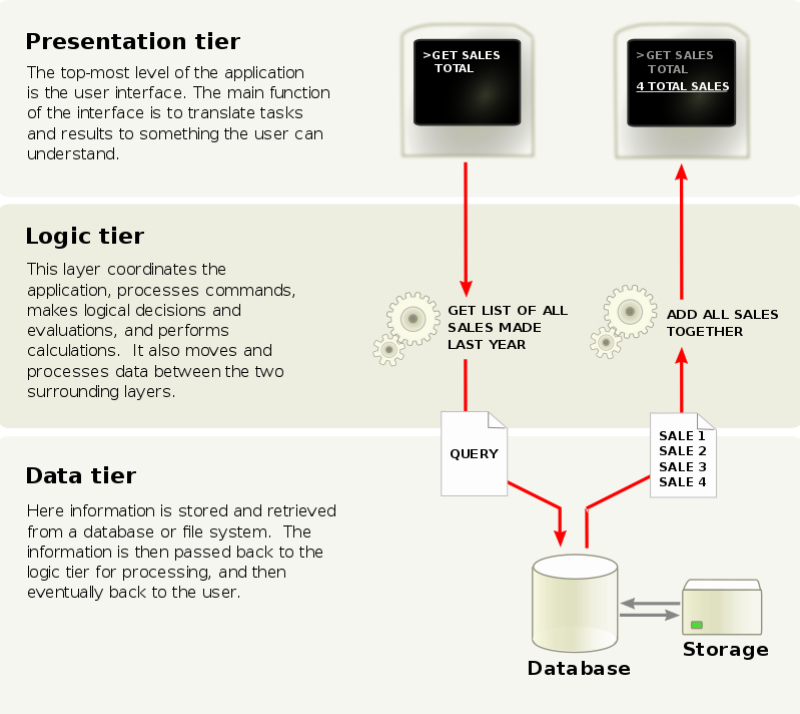
architecture is called an "*n*-tier architecture").

Presentation tier

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).

Application tier (business logic, logic tier, or middle tier)

The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.



Overview of a three-tier application.

Data tier

The data tier includes the data persistence mechanisms (database servers, file shares, etc.) and the data access layer that encapsulates the persistence mechanisms and exposes the data. The data access layer should provide an API to the application tier that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms. Avoiding dependencies on the storage mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change. As with the separation of any tier, there are costs for implementation and often costs to performance in exchange for improved scalability and maintainability.

Web development usage

In the web development field, three-tier is often used to refer to websites, commonly electronic commerce websites, which are built using three tiers:

1. A front-end web server serving static content, and potentially some cached dynamic content. In web-based application, front end is the content rendered by the browser. The content may be static or generated dynamically.
2. A middle dynamic content processing and generation level application server (e.g., Symfony, Spring, ASP.NET, Django, Rails, Node.js).
3. A back-end database or data store, comprising both data sets and the database management system software that manages and provides access to the data.

Other considerations

Data transfer between tiers is part of the architecture. Protocols involved may include one or more of SNMP, CORBA, Java RMI, .NET Remoting, Windows Communication Foundation, sockets, UDP, web services or other standard or proprietary protocols. Often middleware is used to connect the separate tiers. Separate tiers often (but not necessarily) run on separate physical servers, and each tier may itself run on a cluster.

Traceability

The end-to-end traceability of data flows through *n*-tier systems is a challenging task which becomes more important when systems increase in complexity. The Application Response Measurement defines concepts and APIs for measuring performance and correlating transactions between tiers. Generally, the term "tiers" is used to describe physical distribution of components of a system on separate servers, computers, or networks (processing nodes). A three-tier architecture then will have three processing nodes. The term "layers" refers to a logical grouping of components which may or may not be physically located on one processing node.

See also

- Abstraction layer
- Client–server model
- Database-centric architecture
- Front-end and back-end
- Hierarchical internetworking model
- Load balancing (computing)
- Open Services Architecture
- Rich web application
- Service layer
- Shearing layers
- Web application

References

1. Deployment Patterns (Microsoft Enterprise Architecture, Patterns, and Practices) (http://msdn.microsoft.com/en-us/library/ms998478.aspx)
2. Fowler, Martin "Patterns of Enterprise Application Architecture" (2002). Addison Wesley.
3. Deployment Patterns (Microsoft Enterprise Architecture, Patterns, and Practices) (http://msdn.microsoft.com/en-us/library/ms998478.aspx)
4. Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael (1996-08). Pattern-Oriented Software Architecture, Volume 1, A System of Patterns. Wiley, August 1996. ISBN 978-0-471-95869-7. Retrieved from http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471958697.html.
5. Martin Fowler's Service Layer (http://martinfowler.com/eaaCatalog/serviceLayer.html)
6. Martin Fowler explains that Service Layer is the same as Application Layer (http://martinfowler.com/bliki/AnemicDomainModel.html)
7. Comparison/discussion of the GRASP Controller Layer vs. Application/Service Layer (http://tech.groups.yahoo.com/group/domaindrivendesign/message/7582)

8. Domain-Driven Design, the Book pp. 68-74. Retrieved from http://www.domaindrivendesign.org/books#DDD.
9. *Applying UML and Patterns*, 3rd edition, page 203 (http://www.craiglarman.com/wiki/index.php?title=Books#Applying_UML_and_Patterns) ISBN 0-13-148906-2
10. Eckerson, Wayne W. "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications." Open Information Systems 10, 1 (January 1995): 3(20)

External links

- Linux journal, *Three Tier Architecture* (http://www.linuxjournal.com/article/3508)
- Microsoft Application Architecture Guide (http://msdn.microsoft.com/en-us/library/ee658109.aspx)
- Example of free 3-tier system (http://webebenezer.net/build_integration.html)
- *What Is the 3-Tier Architecture?* (http://www.tonymarston.net/php-mysql/3-tier-architecture.html)
- Description of a concrete layered architecture for .NET/WPF Rich Client Applications (http://waf.codeplex.com/wikipage?title=Architecture%20-%20Get%20The%20Big%20Picture&referringTitle=Home)