Concept/Topic: *Layers and Client/Server Cuts (CSCs)*

Context

Architects need to divide the wicked overall problem "design architecture for system to satisfy ASRs" into smaller ones. Layers help with that (aand have other benefits). Modern applications are highly distributed; to meet their performance, scalability and reliability goals they are deployed onto multiple hardware nodes (actual or virtual ones) that are organized into tiers. Many systems have more than one frontend and presentation layer technology (including rich standalone clients, single-page Rich Internet Applications (RIAs), mobile phones and tablet app development frameworks, etc.). Many ASRs and, more generally speaking, architectural drivers, govern the choice of an appropriate/adequate selection of layers and their assignment to tiers. The options for this assignment are called *Client/Server Cuts (CSCs)*.

Definition(s)

The definite source on logical layers is PoEAA Fowler (2002); please read pages 17 to 24 in Chapter 1. The 2nd Edition of the Application Architecture guide from Microsoft is rich in content too, for instance see the chapters on Choosing an Application Type[1], Layered Application Guidelines[2], and A Technique for Architecture and Design[3].[4]

We use the following layering scheme (popular in enterprise application development, but also applicable to other application genres):

- *Presentation Layer* (has display and presentation control logic)
- *Business Logic Layer* (might be split up into workflow and computation/entity management logic)
- *Data Access Layer* (or Persistence Layer)

The five CSCs have been captured as *distribution patterns*[5] rather early in the evolution of object-oriented programming and integrated business information systems:

1. *Distributed Presentation* 2 *Remote User Interface*
2. *Distributed Application Kernel*
3. *Remote Database*
4. *Distributed Database*

The roots of this pattern language even predate your lecturer's start in enterprise application development and integration, but also map well to the "Digital age" (i.e., a world of mobile phones, social networks, JavaScript in the browser implementing rich single page applications, Internet of Things, DevOps, etc.).

The following architectural drivers and decision making criteria (or *forces* in pattern terminology) apply:

- Business needs vs. construction complexity
- Processing style: online (transactional) vs. offline (batch processing)
- Distribution vs. performance, security, consistency

[1] https://msdn.microsoft.com/en-us/library/ee658104.aspx
[2] https://msdn.microsoft.com/en-us/library/ee658109.aspx
[3] https://msdn.microsoft.com/en-us/library/ee658084.aspx
[4] While both of these books are rather old already, their conceptual advice is still highly valid in the current Digital era/age. Some of the technology and product examples might be obsolete now, and some patterns have lost some of their relevance; that said, the overall messages including conflicts between architectural drivers are still applicable.
[5] http://hillside.net/plop/plop97/Proceedings/renzel.pdf

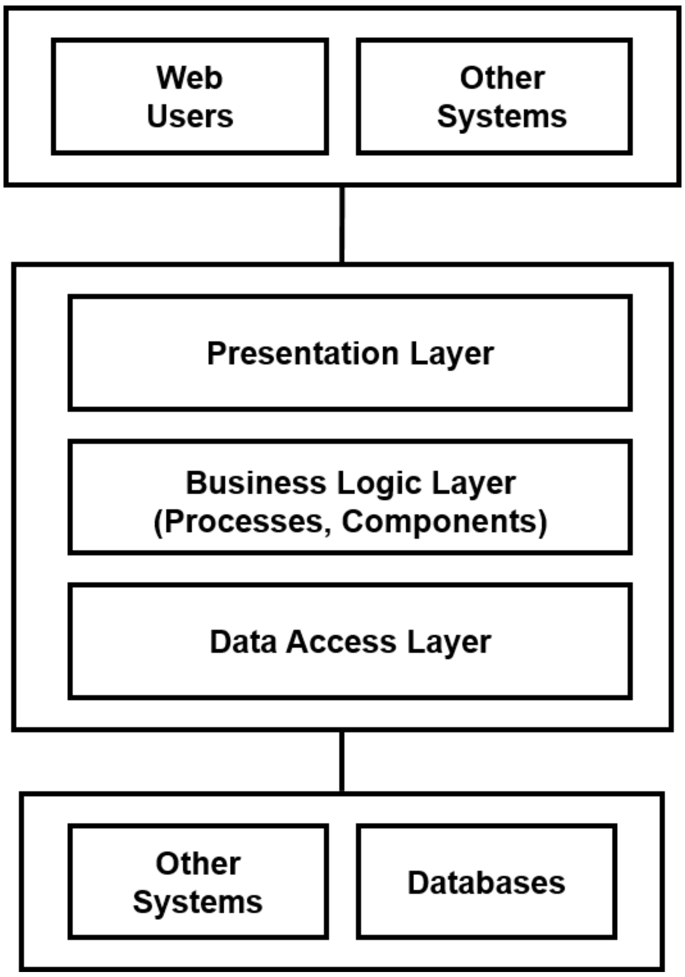

Figure 1: *Logical Layers*

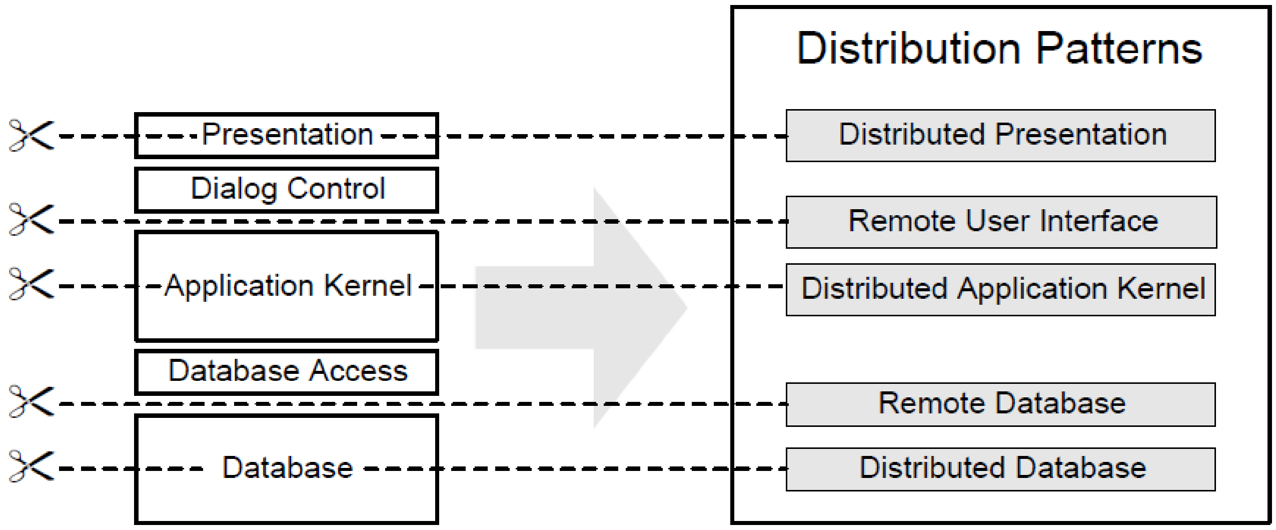

Figure 2: *Distribution Patterns (CSCs)*

- Software distribution cost
- Reusability vs. performance vs. complexity
- Supportability

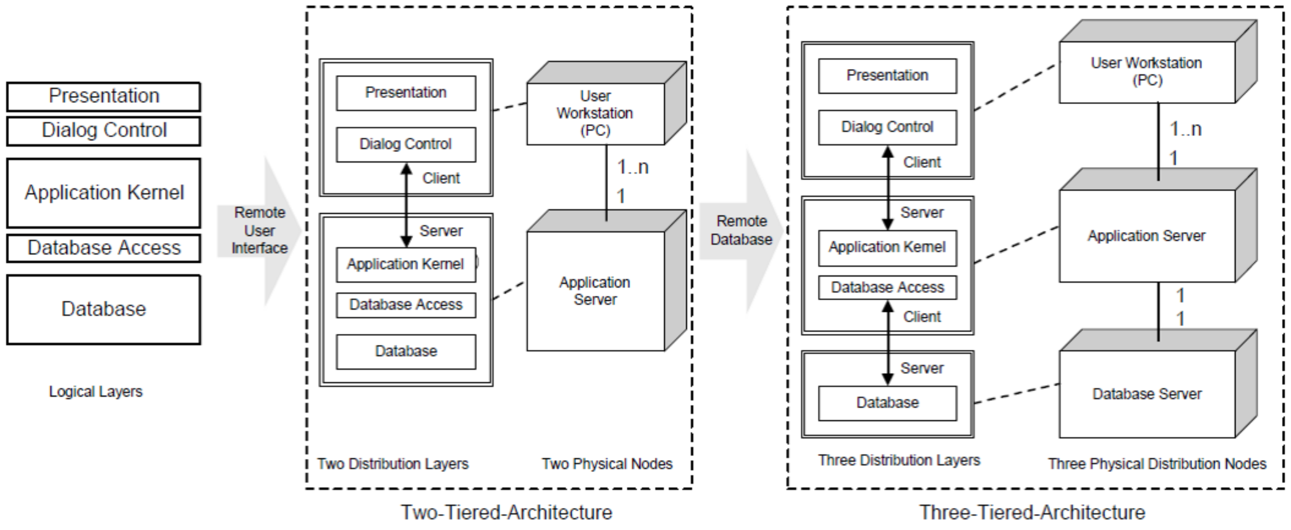

Figure 3: *2-Tier vs. 3-Tier Uage of Distribution Patterns*

Example(s)

Different application archetypes listed in the Microsoft Application Architecture guide[6] are Mobile Application Archetype, Rich Client Application Archetype, Rich Internet Application (RIA) Archetype, Service Archetype, and Web Application Archetype.

Some modern known uses for the patterns include:

1. Distributed Presentation: Thin client on tablet or mobile phone, for instance developed with a cross-platform tool based on HTML.
2. Remote User Interface: App on mobile phone talking to a data-oriented server backend (via HTTP and CRUD resources)
3. Distributed Application Kernel: App on mobile phone talking to a function- or object-oriented server backend (via RESTful HTTP interfaces to compute resources)
4. Remote Database: Remote JDBC
5. Distributed Database: Cassandra and other highly distributed databases supporting eventual consistency

Application of the Concept in Products and Projects

Usage of Concept/Topic:

- The ADD concept catalog in Cervantes and Kazman (2016) picks concepts from the above guide up, but unfortunately is not part of the free sample chapter that is available here[7]. The sample chapter does cover QAS and utility trees though (introduced in lesson 2).
- The SOAD approach introduces meta issues applicable on all layers and tiers, see Table 2 in this article[8]
- ARCUS project, archived here[9].

[6] https://msdn.microsoft.com/en-us/library/ee658104.aspx#RichInternetApplicationArchetype
[7] https://gitlab.dev.ifs.hsr.ch/ZIO/tree/master/ptgmedia.pearsoncmg.com/images/9780134390789/samplepages/9780134390789.pdf
[8] http://soadecisions.org/download/SOAD-SHARK2012v13Final.pdf
[9] http://www.objectarchitects.de/arcus/index.htm

Tips and Tricks

When selecting a layering scheme and CSC the following rules of thumb apply:

- Do not pick the first obvious alternative, but go through the ADD cycle once to find the CSC option that fits your ASRs best.
- Include cross-cutting concerns such a security and systems management (including backup and user management) in your analysis and decision making.
- Also consider n-tier in addition to 2-tier and 3-tier (if justified by requirements).
- Look for framework and middleware support on each layer/tier.
- See this online article[10] for an alternate approach/pattern (functional partitioning over technical).

More Information

Related Topics and Concepts

- Solution Strategy
- SMART NFRs
- C4 Model

Miscellaneous Links:

- More patterns for business information systems (that are still current) have been captured in the ACRUS project[11] that ended in September 1997.
- M. Fowler's website has many articles on design including architecture[12].
- See IFS website Architectural Knowledge Hubs[13] for more links.

References

Cervantes, Humberto, and Rick Kazman. 2016. *Designing Software Architectures: A Practical Approach*. 1st ed. Addison-Wesley Professional.

Fowler, Martin. 2002. *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

[10] https://paulhammant.com/2011/11/29/cookie-cutter-scaling/
[11] http://www.objectarchitects.de/arcus/
[12] https://martinfowler.com/design.html
[13] https://www.ifs.hsr.ch/index.php?id=13193&L=4