Software Architectures

From Design Patterns to Enterprise Architecture

Enterprise Architectures I

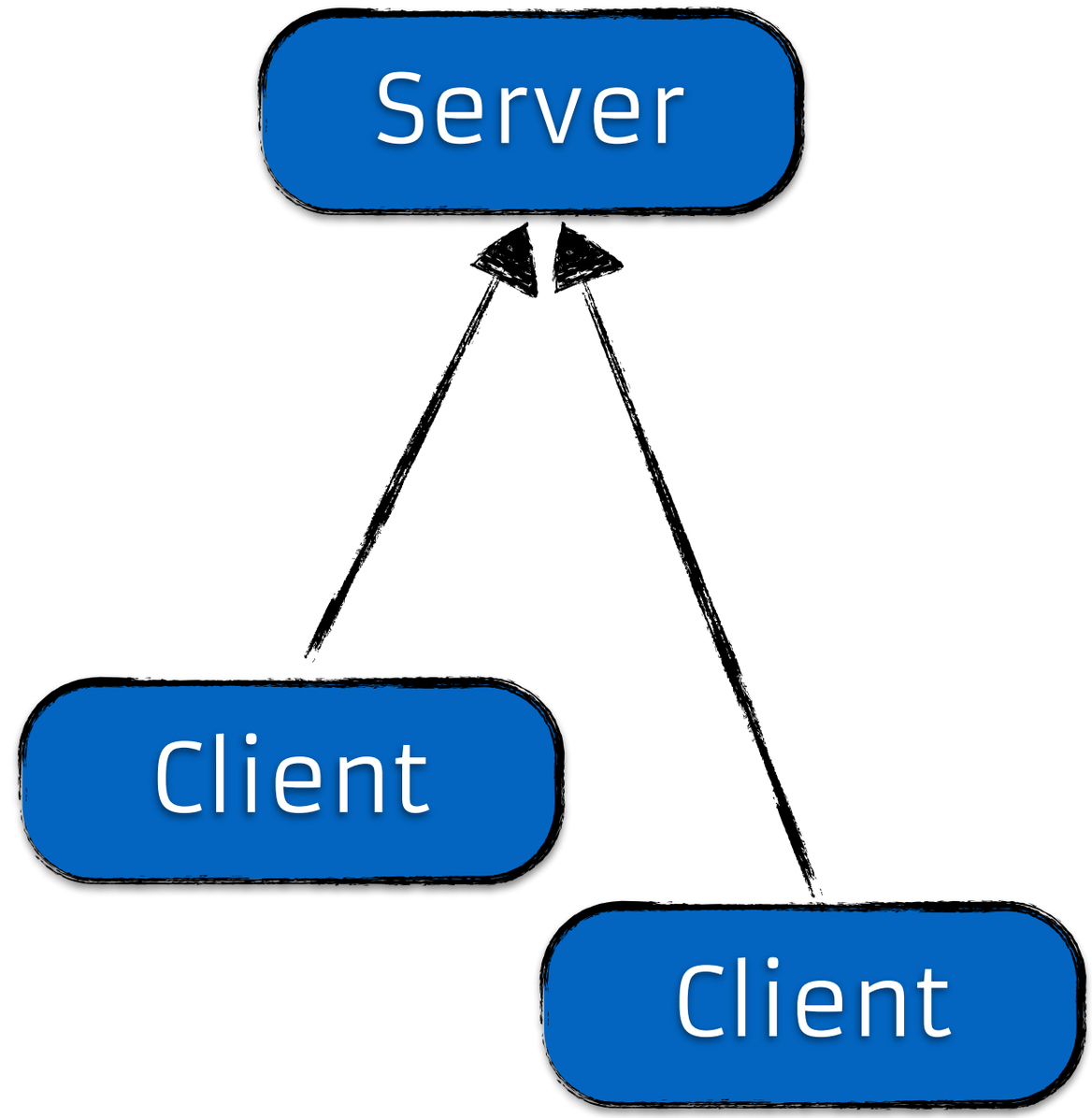
Last week on "Software Architectures"

- Visitor Pattern
- Strategy Pattern
- Factory Method Pattern
- Singleton Pattern
- Exercise: Recognizing Patterns in Architecture

Network architectures

Client-server model

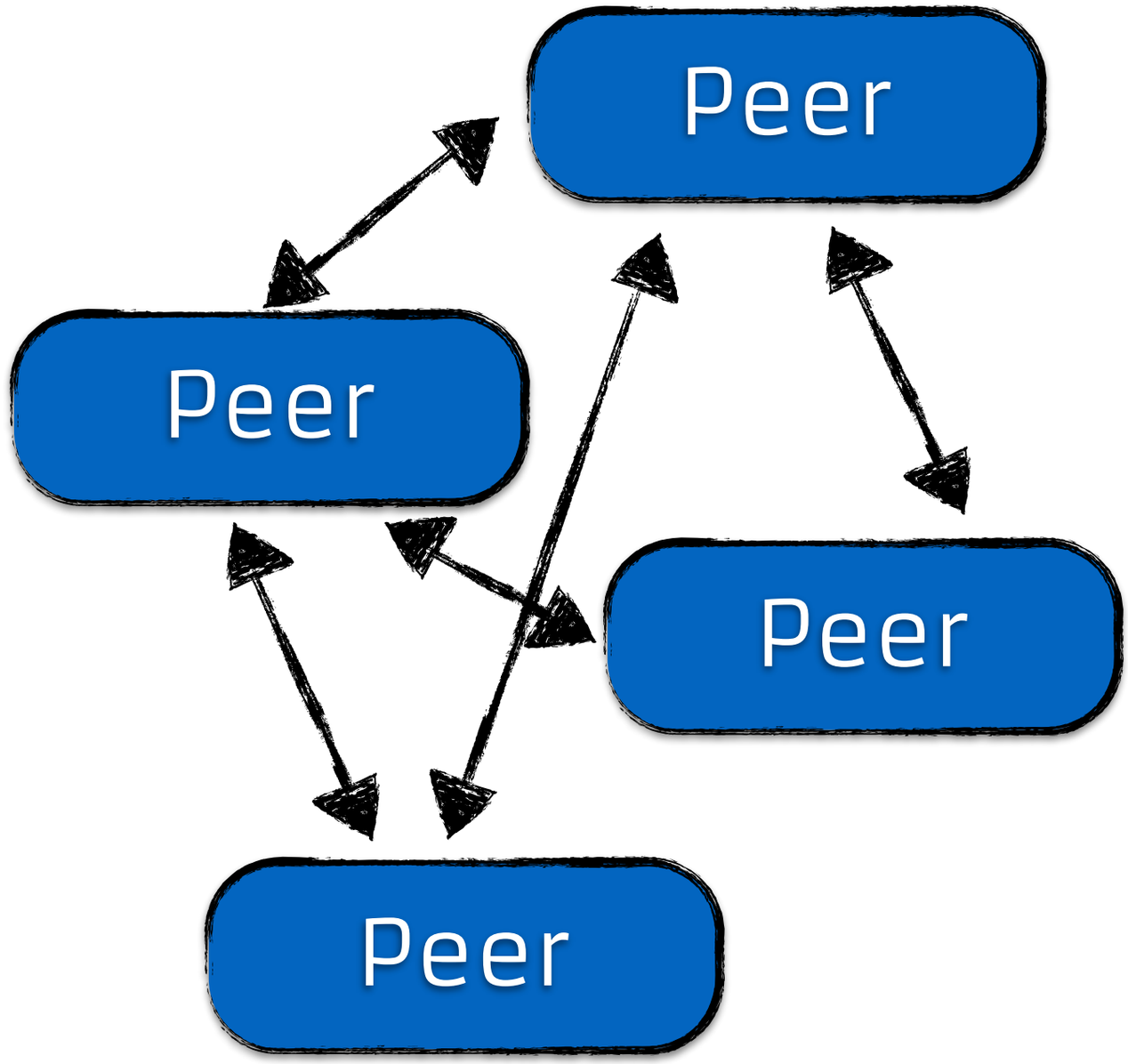
- Invention of XEROX PARC (as so many things)
- Request-Response message pattern
- An application becomes distributed
- A client “borrows” computation time from the server
- The server shares its computation time



Network architectures

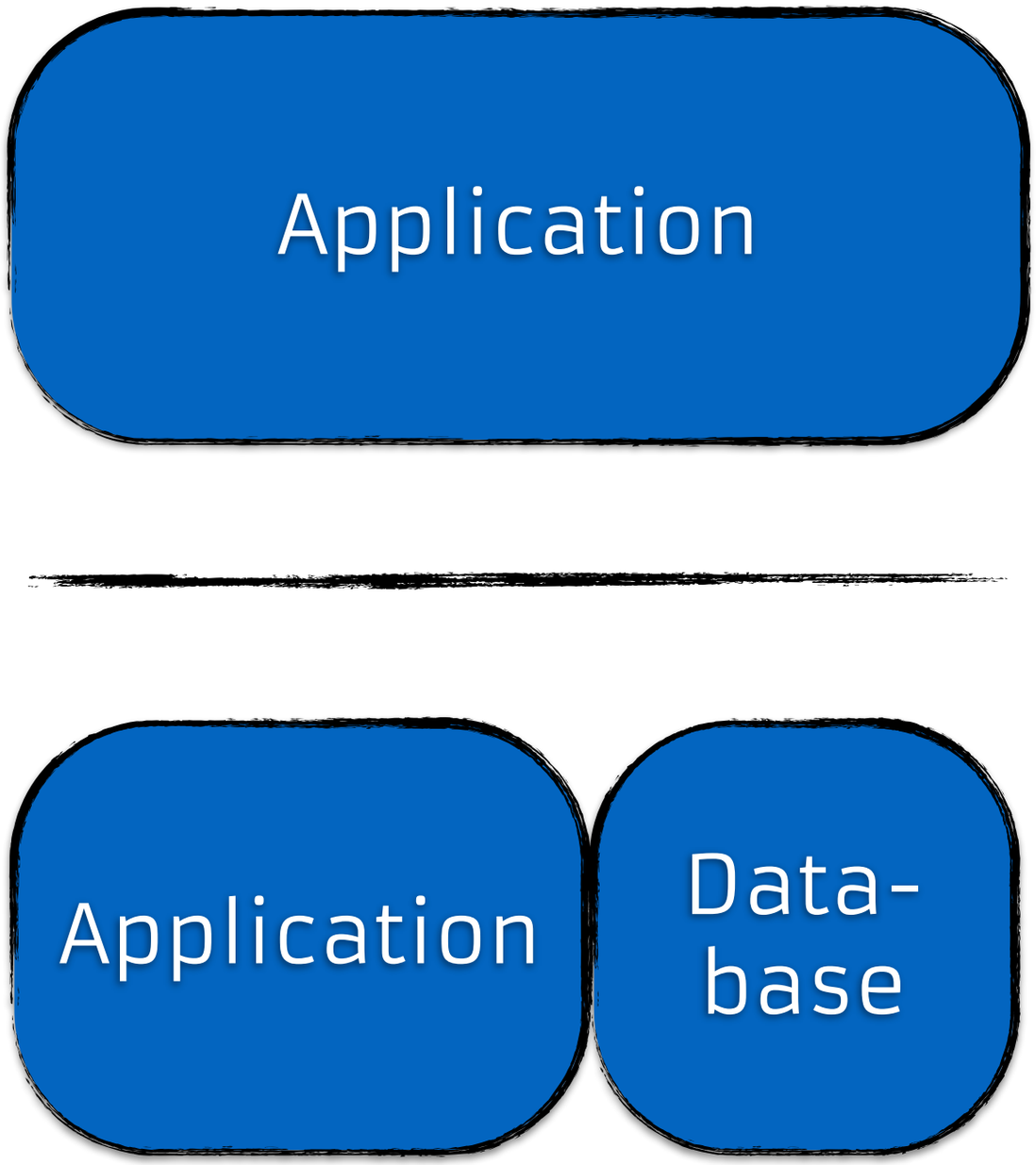
Peer-to-peer model

- Decentralized network strategy
- Peers may share processing time, disk storage or network bandwidth
- Peer discovery can be hard
- Networks can be structured, unstructured or semi-structured
- Are you using peer-to-peer networks?



Multi-tier Architectures

- First applications were monolith in their design
- They did everything themselves from handling user input to storing data
- It turned out that certain parts are common to multiple applications
- For example: data storage

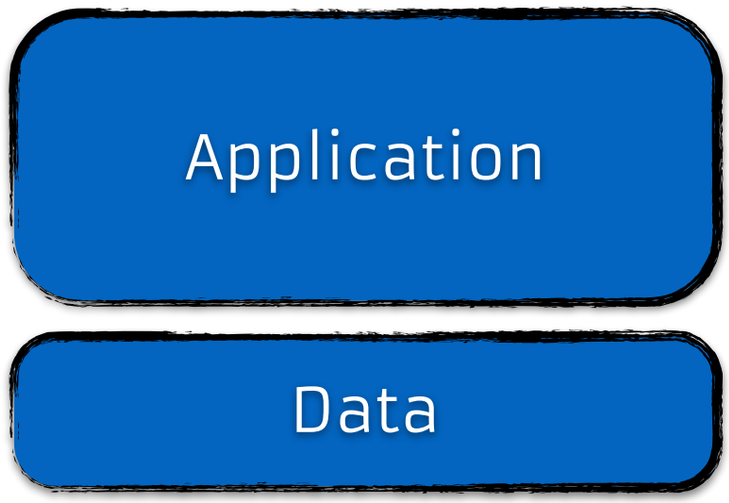


1-tier

- But first… a look back:
- A computer was a machine for a single purpose
- Changing the programming took time
- A program took care of everything itself
- This stayed true until the 1960ties

2-tier

- The first thing that moved out of the monolithic architectures were data storage mechanisms
- 2-tier architectures for systems emerged
- Different data storage techniques evolved:
  - Navigational databases & ISAM
  - Relational database management systems
  - SQL
  - Object-oriented databases
  - NoSQL & XML databases



2-tier

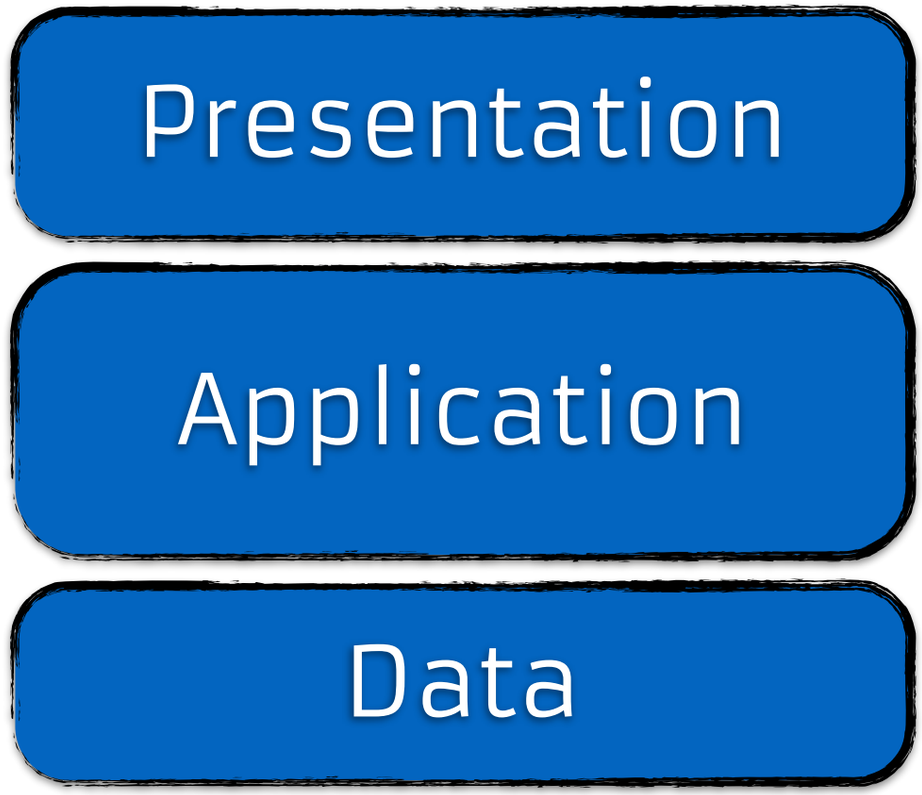
- The first
  architect
- 2-tier arc
- Different
  - Navigat
  - Relatior
  - SQL
  - Object-
  - NoSQL & XML databases

plication

Data

3-tier

- A third tier emerged
- Multiple views on the same application
- Parallel development to distributed systems
  - Tiers do not need to be on the same computer
  - Protocols more important that operating systems and programming languages



n-tier?

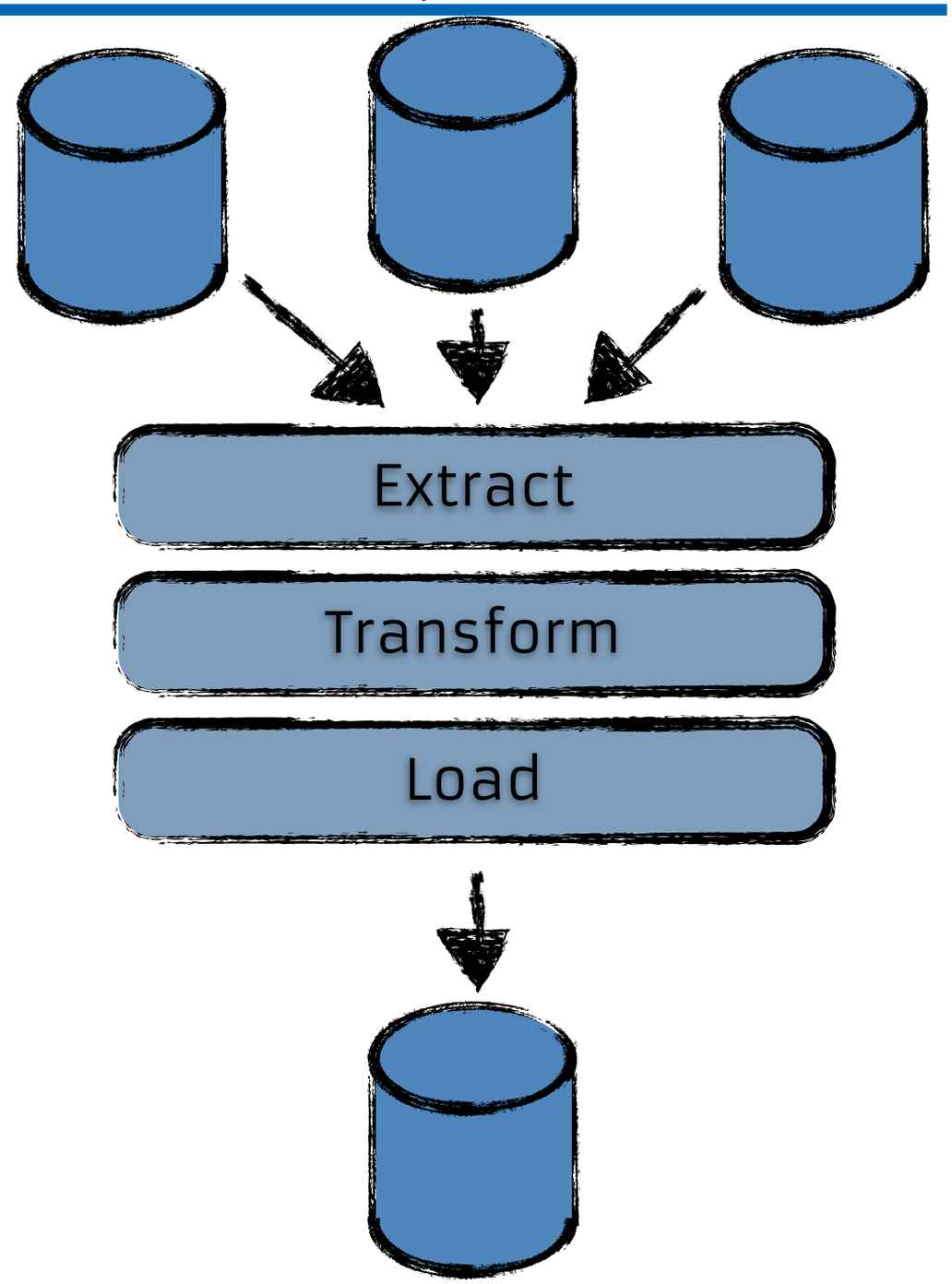
- Same thing…
- …but the middle tier splits up in multiple parts
- That’s why it’s called middleware
- We will inspect more of this next week

- But now we have multiple systems

Integration architectures

Extract Transform Load

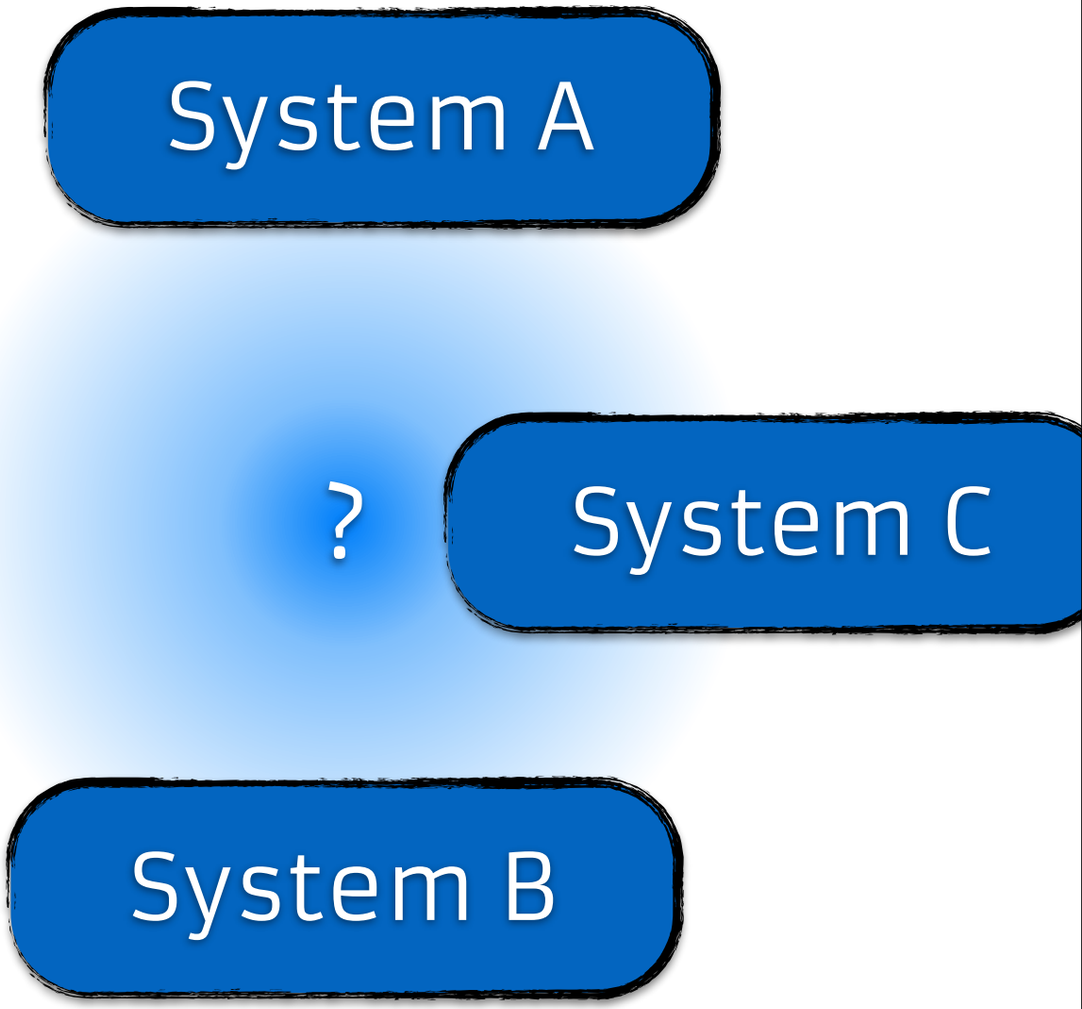
- Extract data from sources
- Transform to fit current needs
- Load to final target
- Usually more than three steps
- Can be used to integrate data from different systems



Integration architectures

Enterprise Application Integration

- Integrate different applications
- Do not build islands or information silos
- Why?
  - Data integration
  - Vendor independence
  - Common facade
- Patterns
  - Mediation (Intra-communication)
  - Federation (Inter-communication)



Integration architectures

Service-Oriented Architectures

- Services are unassociated, loosely coupled units of functionality that are self-contained
- Highly governed architecture, highly rated standard compliance
- Fosters reuse, granularity, modularity, composability, componentization and interoperability
- Decompose by services and not by systems

Rich Internet Applications

- Client-Server architectures put a lot of stress on servers
- Especially in case of web applications
- But client machines are quite strong machines mostly nopping
- RIAs make more use of the client
- Sometimes, special software has to be installed on the client
- Nowadays: HTML5

Cloud

Magic happens here…

Infrastructure as a service

- Hypervisors
  - Xen
  - VMWare ESX/ESXi
- Amazon EC2
- Anything that gives you a computer in the cloud…

Storage as a service

- Services like
  - Dropbox
  - Google Drive
  - Skydrive
  - Amazon S3
- Cloud Databases

Application and Computation as a service

- SalesForce
- Web applications
- Middleware Services
- Basically anything…

Designed by Zombies

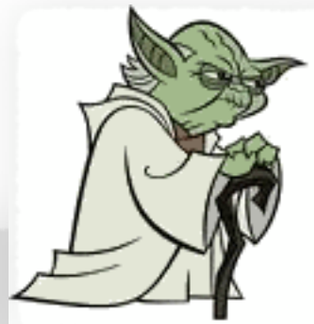
Horror, Terror, Suspense
Bad examples to avoid
Kids don’t do this at home

The XML Column

- Imagine a system for the storage and display of information of various types
- Users complain that the system is getting slower and slower
- Interestingly, every part of the system is getting slower regardless of the size of the data displayed
- The system is using a database… But there is only one table… odd… mhhh….
- This table has two columns: ID and XML
- Everything is stored in this XML column

Code Obfuscation

- The classic: Save your job and don't ever write understandable code

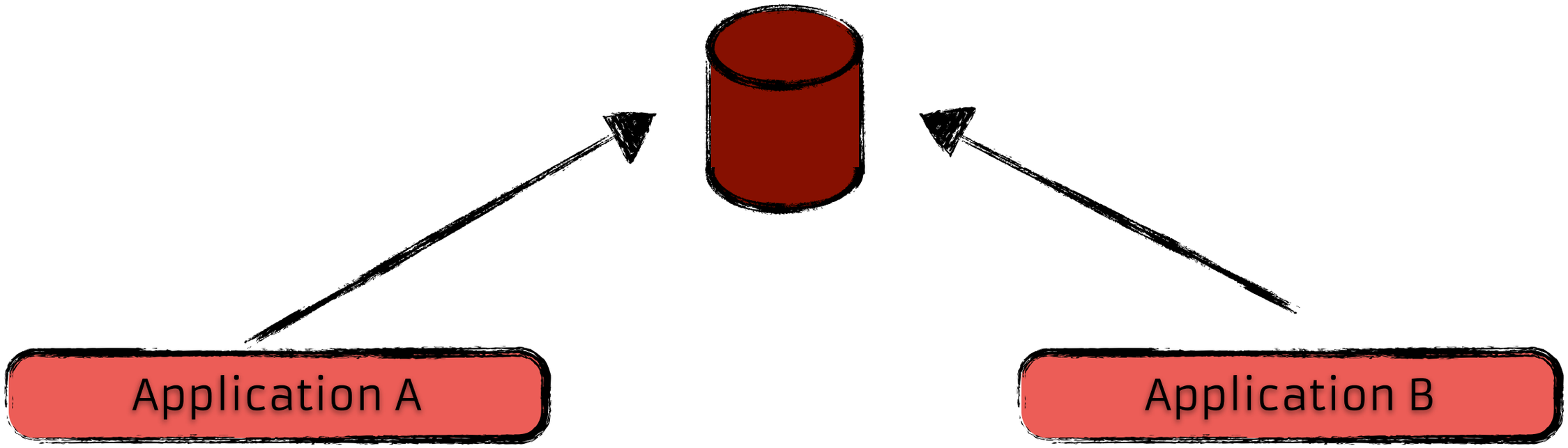


```
if (5 == count)
```

```
r = n / 2;
while ( abs( r
   r = 0.5 * ( r
}
System.out.prin
```

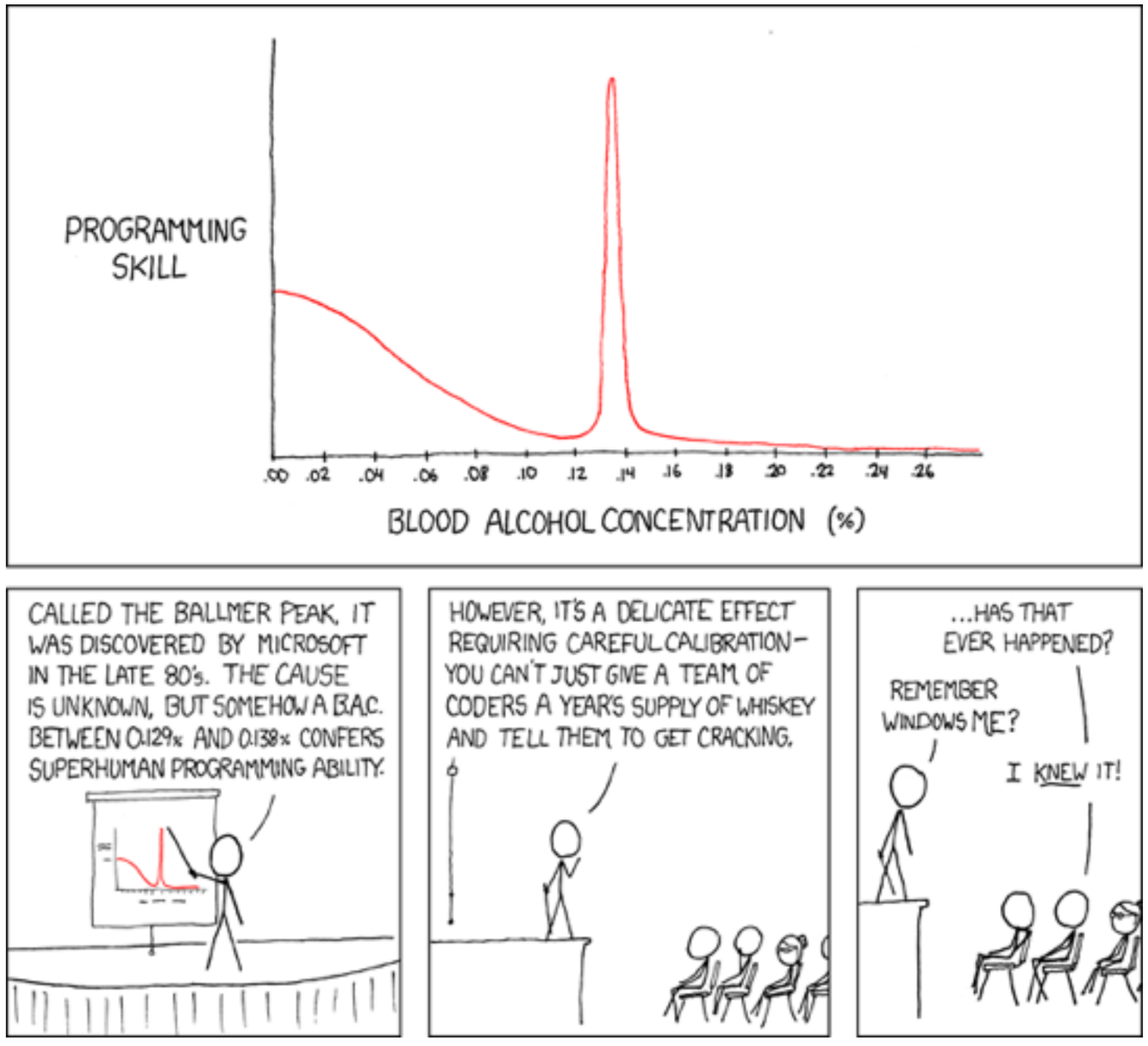
```
Pattern findGamesPattern = Pattern.With.Literal(@"<div")
    .WhiteSpace.Repeat.ZeroOrMore
    .Literal(@"class=""game""").WhiteSpace.Repeat.ZeroOrMore.Literal(
    .NamedGroup("gameId", Pattern.With.Digit.Repeat.OneOrMore)
    .Literal(@"-game""")
    .NamedGroup("content", Pattern.With.Anything.Repeat.Lazy.ZeroOrMo
    .Literal(@"<!--gameStatus")
    .WhiteSpace.Repeat.ZeroOrMore.Literal("=").WhiteSpace.Repeat.Zero
    .NamedGroup("gameState", Pattern.With.Digit.Repeat.OneOrMore)
    .Literal("-->");
```

Database-as-IPC



Alcohol fueled development

- A seriously bad idea



Pasta code (Spaghetti and Lasagna)

- Lots of pasta dishes in your code?
- Spaghetti code: Code that is so twisted and tangled that nobody will ever entangle it
- Lasagna code: Code with way to many layers…

```
10 i = 0
20 i = i + 1
30 PRINT i; " squ
40 IF i >= 10 THE
50 GOTO 20
60 PRINT "Prograr
70 END
```

Big Ball of Mud

- Something that lacks any kind of perceivable architecture

A Big Ball of Mud is a haphazardly structured, sprawling, sloppy, duct-tape-and-baling-wire, spaghetti-code jungle. These systems show unmistakable signs of unregulated growth, and repeated, expedient repair. Information is shared promiscuously among distant elements of the system, often to the point where nearly all the important information becomes global or duplicated. The overall structure of the system may never have been well defined. If it was, it may have eroded beyond recognition.

Brian Foote and Joseph Yoder, Big Ball of Mud. Fourth Conference on Patterns Languages of Programs (PLoP '97/EuroPLoP '97) Monticello, Illinois, September 1997

Lava flow

- The Lava Flow of obsolete technologies and forgotten extensions, leaves hardened globules of dead code in its wake
- Code that’s been around for a while
- Different developers working on it bringing their own ideas without touching parts that have already “hardened” - meaning it’s hard to understand

God object

- Also known as

The Monster Object

THE BLOB

- It knows to much and does to much
- God objects tend to accumulate stuff
- It is usually tightly coupled into the application

Cargo-cult programming

- Code is copied into a project that serves no purpose
- It is just there for ritual purposes
- Usually done by less experienced programmers in the lack of better knowledge
- It can be pretty small… Such as a cast to Object
- Or it can be large, like the usage of a library that is useless

Premature optimization

- One of the worst ideas ever
- Performance considerations alter the design of a program before even trying a simple solution
- Why is this evil?

Wrap up

- Network architectures
  - Client-server model
  - Peer-to-peer model
- Multi-tier architectures
  - 1-tier, 2-tier, 3-tier, n-tier architectures
- Integration architectures
  - Extract, Transform, Load
  - Enterprise Application Integration
  - Service-oriented Architectures
- Rich Internet and Cloud Architecture
- … and of course… lot’s of thing you should not do. :-)