Lecture 3 & 4

Wednesday, 13 September 2023 13:17

an architectural style describes the components of a system, how they are connected and how they exchange data

an interface describes a shared boundary across which two or more separate components of a system exchange information

a connector is a mechanism that mediates communication, coordination or cooperation among components
allows for the flow of control and data between components
example of a physical connector: cable in a datacenter
but a connector may also be a protocol (conceptual connector)

dataflow styles
- batch sequential
- pipes & filters

layered styles

independent components
- client - server
- peer to peer
- SOA
- RESTful/ROA

event - based styles
- publish & subscribe

data - centered styles

other
- mobile agents

architectural styles can be chosen based on
- costs
- scalability
- performance
- reliability
- fault tolerance
- maintainability
- usability
- reusability

descriptions of styles consist of
- motivation
- vocabulary
- structure
- typical behavior
- rules
- weaknesses
- example(s)

layered style
separate concerns, limit dependencies, independent development, independent evolution
organize system in independent layers of abstraction → strict layering = only call to immediate previous layer
layer N calls upon layer N - 1; register callback for other way around
layering with bridge = calls to non-immediate previous layers are allowed
sidecar layering = has utility library, which can be called by all layers
layer N may only use lower layers (only N-1 in strict layering)
each layer adds additional complexity

a service is a contractually specified overall functionality (semantics) of an entity
extra-functional requirements of a service may place requirements on e.g. service quality

a service interface (API) consists of actions ('primitives') and responses that make the service available

a protocol is a formal set of rules that dictates how information exchange, as well as interaction between entities should take place
rules → includes finite state machine to keep track of what to do with responses (how to interpret responses)

a provided service is a service provided by a protocol ?
a carrier is realization of a protocol relies on another service and corresponding protocol

a binding is the set of rules that specify how a protocol is mapped onto a carrier

client - server style : sharing some localized resource
client, server, server discovery
dynamic structure : many - to - one; server may be distributed; server location possibly already determined at deployment time
client finds server access point through discovery; regular interaction as series of synchronous request - reply pairs
↓ session → stateful
servers are passive; clients are active; no connection among clients; limited state on the server per client
single server can be performance bottleneck; server can be single point of failure; stateful servers scale poorly
email, online banking

client - server architecture may improve security, as it is possible to place a filter between clients and server.

3 - layer logical organization (user interface, application, database)
(also: processing level)
2 - layer physical organization (client, server)
thin / fat clients → was an (exam) question last year

peer - to - peer style
sharing resources + content, cooperation in communities, symmetry in roles, increase concurrency, horizontal scalability
DHT (distributed hash table), peer, decentralized
dynamic community, overlay (structured/unstructured), on top of network technology
peer finds access point, joins community, provide & use services, collective results in new functionality
peer → symmetric in functionality / contribution, can discover / read / cooperate with all other peers ; superpeer → contributes extra resources
security, dealing with peers coming / going
BitTorrent

REST
portability; independent development & deployment; reduction of interaction complexity; reliability; scalability
proxy, gateway, user agent, origin server, cache, URI/URL
layered client - server style
resource state exchanged in text; nested request - reply; extensive caching
stateless; response labeled as cacheable; uniform interface; transparent layering; client functionality can be adapted by code-on-demand
SOAP may be used independently of HTTP, has better security support & deals better with transactions
S3, Twitter

publish/subscribe
decoupling data producer / consumer; send data when available + relevant; multiple producers / consumers
publisher → broker → subscriber
subscribers find brokers + subscribe to topic; publishers register with broker / are discovered; brokers forward to subscribers; broker can be omitted
subscription relates topic to subscriber; publisher can have multiple subscribers per topic; notification goes to all subscribers; broker decouples publishers / subscribers
large & fluctuating latency; not suitable for hard real-time constraints
RSS/Atom, stock tickers

cloud computing
virtualized resources
4 layers:

application	SaaS	actual applications
platform	PaaS	higher-level abstractions
infrastructure	IaaS	virtualization
hardware		

edge - server architecture : on-premises servers / on own network (not beyond internet 'boundary')

life cycle = series of stages product / system goes through from inception to decommissioning