Table 1: A feature-based classification of architectural styles

Style	Constituent parts		Control issues			Data issues				Control/data interaction		Type of reasoning
	Components	Connectors	Topology	Synchronicity	Binding time	Topology	Continuity	Mode	Binding time	Isomorphic shapes	Flow directions	
Data flow styles: Styles dominated by motion of data through the system, with no "upstream" content control by recipient												
Batch sequential [Be90]	stand-alone programs	batch data	linear	seq	r	linear	spor hvol	passed, shared	r	yes	same	Functional composition
Dataflow network [B+88]	transducers	data stream	arb	asynch	i, r	arb	cont lvol or hvol	passed	i, r	yes	same	
•Sub-styles	*See Section 4.1*											
Closed loop control [Sh95]	embedded process, function	continuous refresh	fixed	asynch	w	fixed cyclic[1]	cont lvol	passed, shared	w	no	n/a	
Call-and-return styles: Styles dominated by order of computation, usually with single thread of control												
Main program/subroutines [Pa72, Bo86]	procedures, data	procedure calls	hier	seq	w, c	arb	spor lvol	passed, shared	w, c, r	no	n/a	Hierarchy (local reasoning)
Information hiding systems [Pa72]	managers	procedure calls	arb	seq	w, c, r	arb	spor lvol	passed	w, c, r	yes	same	
•Abstract data types [Sh81]	managers	static procedure calls	arb	seq	w, c	arb	spor lvol	passed	w, c, r	yes	same	
•Classical[2] objects [Bo86]	managers (objects)	dynamic procedure calls	arb	seq	w, c, r	arb	spor lvol	passed	w, c, r	yes	same	
•Naive[3] client/server	programs	procedure calls or RPCs	star	synch	w, c, r	star	spor lvol	passed	w, c, r	yes	opposite	
Interacting process styles: Styles dominated by communication patterns among independent, usually concurrent, processes												
Communicating processes [An91, Pa85]	processes	message protocols	arb	Any but seq	w, c, r	arb	spor lvol	any	w, c, r	possibly	if isomorphic, either	Nondeterminism
•Lightweight processes	lightweight processes	threads, (shared data[4])	arb	ls/par, synch	w, c	arb	spor (th), cont (da)	passed, shared	w, c	no	n/a	
•Distributed objects	managers	remote proc calls	arb	ls/par, synch	w, c, r	arb	spor lvol	passed	w, c, r	no	n/a	
•Process-based naive client/server[3]	processes	request/reply messages	star	synch	w, c, r	star	spor lvol	passed	w, c, r	yes	opposite	
•Other sub-styles	*See Section 4.2*											
Event systems [Ba86b, G+92, Ge89, HN86, He69, KP88, Re90]	processes	implicit invocation	arb	asynch, opp	i, r	arb	spor lvol	bdcast	i, r	no	n/a	

Table 1: A feature-based classification of architectural styles

Style	Constituent parts		Control issues			Data issues				Control/data interaction		Type of reasoning
	Components	Connectors	Topology	Synchronicity	Binding time	Topology	Continuity	Mode	Binding time	Isomorphic shapes	Flow directions	
Data-centered repository styles: Styles dominated by a complex central data store, manipulated by independent computations												Data integrity
Transactional database [Be90, Sp87]	memory, computations	trans. streams (queries)	star	asynch, opp	w	star	spor lvol	shared, passed	w	possibly	if isomorphic, opposite	ACID[5] properties
•Client/server	managers, computations	transaction opns with history[3]	star	asynch.	w, c, r	star	spor lvol	passed	w, c, r	yes	opposite	
Blackboard [Ni86]	memory, computations	direct access	star	asynch, opp	w	star	spor lvol	shared, mcast	w	no	n/a	convergence
Modern compiler [SG96]	memory, computations	procedure call	star	seq	w	star	spor lvol	shared	w	no	n/a	invariants on parse tree
Data-sharing styles: Styles dominated by direct sharing of data among components												Representation
Compound document	editable documents	shared representation	n/a	n/a	n/a	hier	cont	shared	r	n/a	n/a	
Hypertext	documents	internal refs.				arb	cont	shared	w, c, r			
Fortran common, Jovial Compool	data structures	sharing (aliasing)				arb	cont	shared	w, c			
Lightweight processes[4]	*See interacting processes style group. This style hybridizes processes and shared data, with emphasis on process*											
Hierarchical styles: Styles dominated by reduced coupling, with resulting partition of the system into subsystems with limited interaction												Levels of service
Layered [Fr85, LS79]	various	various	hier	any	any	hier	spor lvol, cont	any	w, c, i, r	often	same or opp	
•Interpreter (Virtual machine) [HR85]	memory, state machine	direct data access	fixed hier	seq	w, c	hier	cont	shared	w, c	no	n/a	

Key to column entries	
Topology	hier (hierarchical), arb (arbitrary), star, linear (one-way), fixed (determined by style)
Synchronicity	seq (sequential, one thread of control), ls/par (lockstep parallel), synch (synchronous), asynch (asynchronous), opp (opportunistic)
Binding time	w (write-time--that is, in source code), c (compile-time), i (invocation-time), r (run-time)
Continuity	spor (sporadic), cont (continuous), hvol (high-volume), lvol (low-volume)
Mode	shared, passed, bdcast (broadcast), mcast (multicast), ci/co (copy-in/copy-out)

Notes:

1. Closed loop control establishes a controlling relation between an embedded process and a control function that responds to perturbations.
2. By "classical object" we mean objects as they originally emerged: non-concurrent, interacting via procedure-like methods. Objects are now often defined much more broadly, especially in their types of interactions.
3. True client/server systems maintain context that captures the current state of an ongoing series of actions. "Client/server" is sometimes used to describe systems that ignore this requirement and simply use components that call and define procedures or send request/reply messages amonge processes. We call the latter "naive client/server systems."
4. Lightweight processes may take advantage of the shared name space; they become a hybrid of communicating processes and shared data.
5. The ACID properties are atomicity, consistency, isolation, and durability.