Conclusion

Service-Oriented Architecture

Jeremy Gibbons

Contents

1 What have we learned?

- SOA is an approach for large systems
- toleration of heterogeneity and imperfection
- enabling scalability under modification
- quadratic growth leads to bottlenecks at hubs
- basically, it's all about *loose coupling*: minimizing dependencies

1.1 Forms of loose coupling

	tight	*loose*
physical	point-to-point	mediated
communication	synchronous	asynchronous
data model	complex types	simple types
type system	strong, static	loose, dynamic
interaction	object navigation	data-centric
process logic	central	distributed
binding	static	dynamic
platform	strong dependence	independence
transactions	2PC	compensation
deployment	simultaneous	gradual
versioning	explicit	implicit

1.2 Asynchronous communication

- concurrent execution
- non-determinism, race conditions
- interrupt, re-entrance
- correlation of response with request
- recollection of original context

1.3 Heterogeneous datatypes

- hoping for harmonization is the kiss of death
- too difficult to reach consensus: “analysis paralysis”
- union of all extensions gets unwieldy
- instead, accept mappings between different representations

1.4 Mediators

- point-to-point: sender specifies receiver's address
- what if receiver moves? overloads? breaks?
- intermediary isolates sender and receiver
- either query then send (broker, name server)
- . . . or send and forward

1.5 Loose type-checking

- strong static typing catches many errors early
- but then interface changes propagate
- moreover, more components need to know about the types (service repository, validating ESB, . . .)
- generic data structures more adaptable
- eg dictionary of key–value pairs rather than parameter list
- similarly documents rather than object hierarchies
- similarly dynamic rather than static binding

2 The two-edged sword

- loose coupling sounds good
- but it nearly always incurs a cost

  > Any problem in computer science can be solved with another layer of indirection. But that usually will create another problem.
  >
  > (David Wheeler)

- in other words, sometimes

  > Worse is better.
  >
  > (Richard Gabriel, `http://www.jwz.org/doc/worse-is-better.html`)

Index

Contents

Service-Oriented Architecture

Monday	Tuesday	Wednesday	Thursday	Friday
Introduction Components	REST	Composition	Architecture	Engineering
coffee	coffee	coffee	coffee	coffee
Components	REST	Composition	Architecture	Conclusion
lunch	lunch	lunch	lunch	lunch
Web Services	Qualities	Objects	Semantic Web	
tea	tea	tea	tea	
Web Services	Qualities	Objects	Semantic Web	