Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design

Paper Presentation [http://dret.net/netdret/publications#pau09a] at WWW2009 [http://www2009.org/] (Madrid, Spain)

Cesare Pautasso (**Faculty of Informatics**, **University of Lugano**)
Erik Wilde (**School of Information**, **UC Berkeley**)

April 22, 2009

Contents

Abstract (2)

Loose coupling is often quoted as a desirable property of systems architectures. One of the main goals of building systems using Web technologies is to achieve loose coupling. However, given the lack of a widely accepted definition of this term, it becomes hard to use coupling as a criterion to evaluate alternative Web technology choices, as all options may exhibit, and claim to provide, some kind of “loose” coupling effects. This paper presents a systematic study of the degree of coupling found in service-oriented systems based on a multi-faceted approach. Thanks to the metric introduced in this paper, coupling is no longer a one-dimensional concept with loose coupling found somewhere in between tight coupling and no coupling. The paper shows how the metric can be applied to real-world examples in order to support and improve the design process of service-oriented systems.

Authors/Presenters

Cesare Pautasso (4)

- Computer Science at Politecnico di Milano, Italy [http://www.polimi.it]
- Ph.D. at ETH Zürich [http://www.ethz.ch/index_EN] (2004)
- Post-Doc at ETH Zürich [http://www.iks.inf.ethz.ch/]
  - Software: JOpera: Process Support for more than Web services [http://www.jopera.org]
- Researcher at IBM Zurich Research Lab [http://www.zurich.ibm.com] (2007)
- Assistant Professor at the Faculty of Informatics [http://www.inf.unisi.ch/] (since September 2007)
- Representations: Web [http://www.pautasso.info]

Erik Wilde (5)

- Computer Science at Technical University of Berlin (TUB) [http://www.tu-berlin.de/eng/] (88-91)
- Ph.D. at ETH Zürich [http://www.ethz.ch/index_EN] (92-97)
- Post-Doc at ICSI, Berkeley [http://www.icsi.berkeley.edu/] (97/98)
  - book on "Technical Foundations of the World Wide Web [http://dret.net/netdret/publications#wil98]"
- Various activities back in Switzerland (98-06)
  - teaching at ETH Zürich [http://www.ethz.ch/index_EN] and FHNW [http://www.fhnw.ch/]
  - working as independent consultant (training, courses, consulting)
  - research focus on Web architecture and XML technologies
- Professor at the School of Information [http://ischool.berkeley.edu/] (since Fall 2006)
  - technical director of the Information and Service Design (ISD) program [http://isd.ischool.berkeley.edu/]
- Representations: Web [http://dret.net/netdret/]; twitter [http://twitter.com/dret]; blog [http://dret.typepad.com/]

Coupling in Information Systems

Web Service Quotes (7)

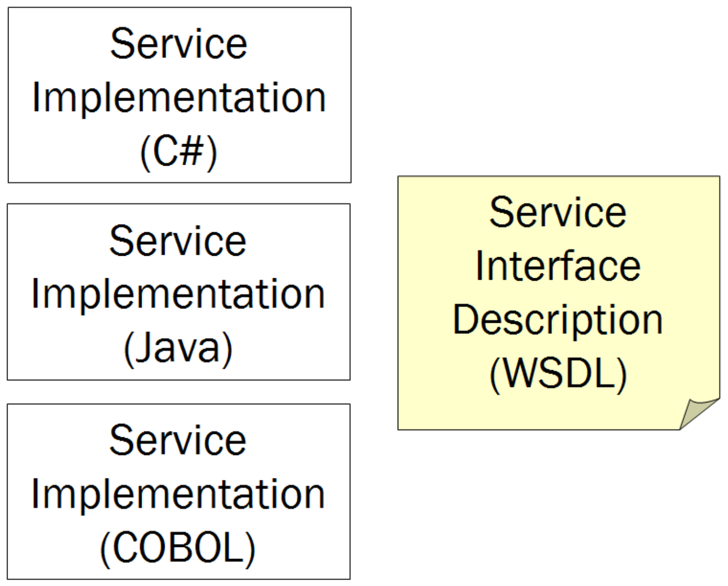
The notion of designing services to be loosely coupled is the most important, the most far reaching, and the least understood service characteristic.

"Understanding SOA with Web Services", Eric Newcomer and Greg Lomow, Addison-Wesley, 2004 [http://www.informit.com/store/product.aspx?isbn=0321180860]

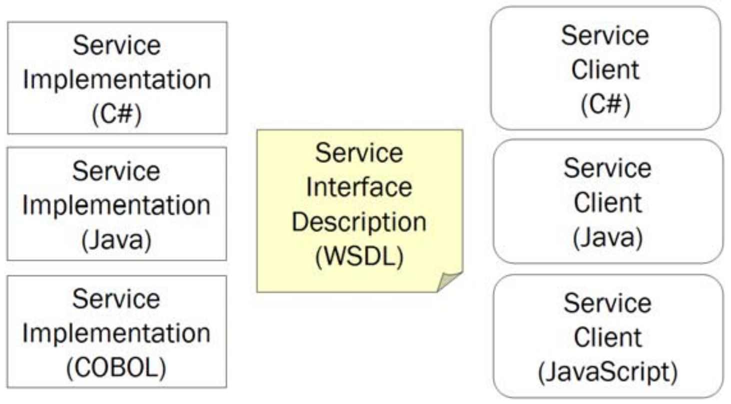
Loose Coupling is the secret sauce of Web services.

"Service Orient or Be Doomed!", Jason Bloomberg and Ronald Schmelzer, Wiley, 2006 [http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471768588.html]

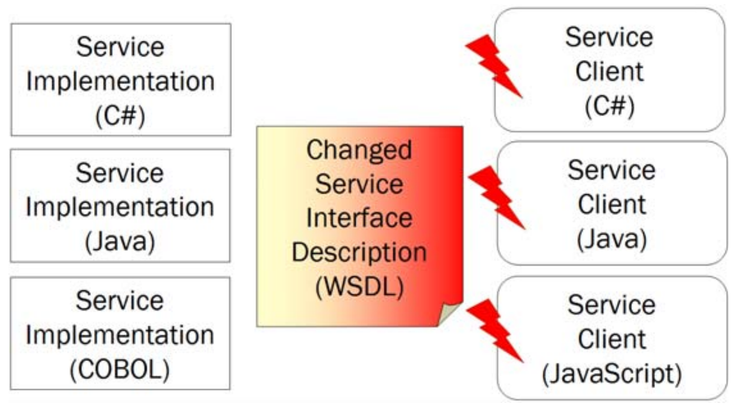
Is WSDL Loosely Coupled? (8)


Service Implementation (C#)
Service Implementation (Java)
Service Implementation (COBOL)
Service Interface Description (WSDL)

Is WSDL Loosely Coupled? Yes! (9)


Service Implementation (C#)
Service Implementation (Java)
Service Implementation (COBOL)
Service Interface Description (WSDL)
Service Client (C#)
Service Client (Java)
Service Client (JavaScript)

Is WSDL Loosely Coupled? No! (10)


Service Implementation (C#)
Service Implementation (Java)
Service Implementation (COBOL)
Changed Service Interface Description (WSDL)
Service Client (C#)
Service Client (Java)
Service Client (JavaScript)

Multi-Faceted Coupling (11)

- Is interface description like WSDL “loosely coupled”?
  - it is because it allows language-independent services to be used
  - it is not because generated code breaks when the interface changes
- “Coupling” depends on perspectives and goals
  - there is more than one perspective to look at SOA scenarios
  - there is more than one goal to achieve for each of these perspectives
- Facets [Facets (1)] allow to use various perspectives
  - they are not “dimensions” because they are not completely independent
  - it is Future Work [Future Work (1)] to better figure out the dependencies

Facets

Coupling Facets (13)

- Your system is “loosely coupled”? What do you mean by that?
- Your system is “tightly coupled”? What do you mean by that?
- Coupling is more than just a binary switch
- Apple's *Push Service* for the iPhone SDK
  - pushing notifications is tightly coupled with Apple's infrastructure
  - handling notifications is loosely coupled and works like a regular Web app
- iPhone map apps can now be built on an API
  - apps are tightly coupled with the iPhone (web-based apps are loosely coupled)
  - platform is tightly coupled with the device (high switching costs)
  - data flow can be loosely coupled (for example with a RESTful service)

12 Facet Summary (14)

Facet	Tight Coupling	Loose Coupling
Discovery **[Discovery (1)]**	Registration	Referral
Identification **[Identification (1)]**	Context-Based	Global
Binding **[Binding (1)]**	Early	Late
Platform **[Platform (1)]**	Dependent	Independent
Interaction **[Interaction (1)]**	Synchronous	Asynchronous
Interface Orientation **[Interface Orientation (1)]**	Horizontal	Vertical
Model **[Model (1)]**	Shared Model	Self-Describing Messages
Granularity **[Granularity (1)]**	Fine	Coarse
State **[State (1)]**	Shared, Stateful	Stateless
Evolution **[Evolution (1)]**	Breaking	Compatible
Generated Code **[Generated Code (1)]**	Static	None/Dynamic
Conversation **[Conversation (1)]**	Explicit	Reflective

Discovery (15)

- Centralized Registration is tightly coupled
  - services register with a centralized UDDI registry
  - clients query the registry to discover new services
- Decentralized Referral is loosely coupled
  - clients learn about new services by getting “a link” from other clients
  - services may refer clients to other services

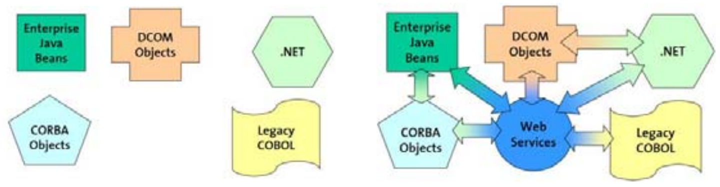
Identification (16)

- Naming is one of the core tasks in information systems
  - without naming, it is impossible to reliably identify anything
- Naming can be done centrally or independently
  - centralized authorities often have a limited (non-global) scope
  - federated naming has to insure that names and namespaces do not clash
- Tightly coupled naming binds name to a non-global scope
  - names leaving that scope have to be *contextualized* to remain usable
- Loosely coupled naming uses globally unique names
  - names can be safely used without additional scoping/context rules

Binding (17)

- *Early binding* makes it hard to change binding decisions
- *Late binding* allows more dynamic approach (but may be expensive)
- Binding happens on various levels
  - DNS names have to be resolved to IP addresses (*CDNs* exploit this fact)
  - service interfaces could be resolved to service providers
- *Discovery* can be a *compile-time* or *run-time* process
  - discovering interface descriptions compiles interfaces into code [Generated Code (1)] (tightly coupled)
  - using uniform interfaces makes it easier to switch services (loose coupling)
  - binding to *expected resource representations* also increases coupling
- Dynamically switching compile-time bindings is very hard
  - in theory possible if the services implement the same technical model
  - in practice hard to do reliably, rarely done, and tightly coupled

Platform (18)

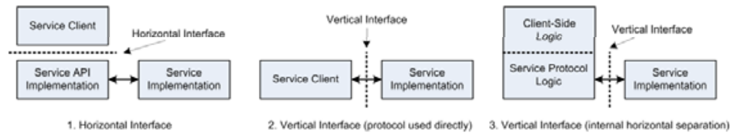


- Tightly coupled: platform/language dependent
- Loosely coupled: platform/language independent

Interaction (19)

- Do two services need to be available at the same time in order to successfully interact?
  - synchronous → tight coupling
  - asynchronous → loose coupling

Interface Orientation (20)



- *APIs* hide distribution and heterogeneity
  - they also hide the actual protocol required to interact
  - apps are tightly coupled with the specific API
- *Protocols* expose the rules for formats for communications
  - they require clients to directly interact with services
  - the protocol can be implemented by any service client
  - there may/should be some internal separation in the client
- APIs are convenient, but protocols are more important

Model (21)

> Die Bedeutung eines Wortes ist sein Gebrauch in der Sprache. (The meaning of a word is its use in language.)

Ludwig Wittgenstein [http://en.wikipedia.org/wiki/Ludwig_Wittgenstein], "Philosophical Investigations [http://en.wikipedia.org/wiki/Philosophical_Investigations]", 1953

- Shared models often lead to 1:1 serializations of objects
- Shared representations require each peer to map from model to representation
  - internally, they can use any model they like
- "Model sharing" should be limited as much as possible
  - shared "deep models" of data → tight coupling
  - representations as "surface models" → loose coupling

Granularity (22)

- Trade-off between:
  - complexity/reusability of the service interface and
  - number of interactions required to service a client
- fine-grained → tight coupling
  - many functions, often derived from an implicitly shared model [Model (1)]
- coarse-grained → loose coupling
  - few functions that focus on exchanging agreed-upon representations [Model (1)]

State (23)

- *Shared State* requires expensive session tracking
  - tight coupling because services need to keep track of clients
- *Stateless Interactions* keep state either on the client or in resources
  - loose coupling because services can treat interactions in isolation

Evolution (24)

- Services should support *versioning*
- *Compatibility* can come in two different flavors
  1. *backward compatibility*: an older client can work with an newer service
  2. *forward compatibility*: a newer client can work with an older service
- Coupling can be associated with various compatibility issues
  - tightly coupled scenarios require synchronized updates
  - loosely coupled scenarios allow independent versioning

Generated Code (25)

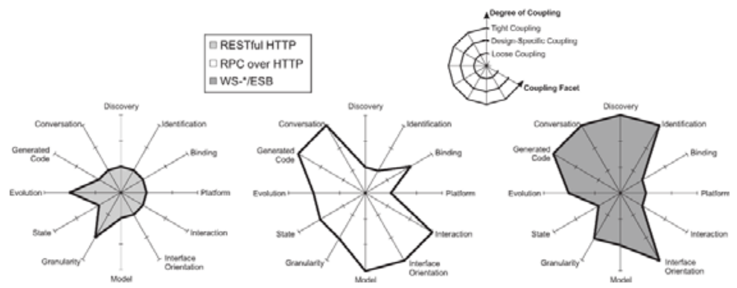
- *Code generation* takes interface descriptions and generates stubs
  - it is based on compile-time binding to interface descriptions [Binding (1)]
  - it creates a horizontal interface [Interface Orientation (1)] on top of the generated code
- Code can be generated/hardcoded for various aspects of service handling
  - *interaction code* handles access functions and handling workflows
  - *resource handling* processes representations in RESTful scenarios
- *Extension mechanisms* can be used to create less coupling
  - browsers can be extended for handling new media types

Conversation (26)

- Services may provide functionality that requires clients to exchange a set of messages in a certain order
  - design-time, explicit conversation description → tight coupling
  - run-time, reflective discovery of the conversation → loose coupling

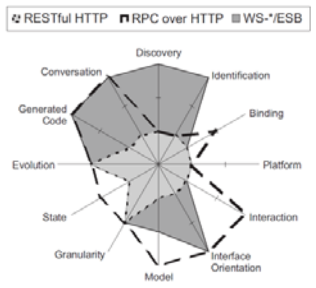
Examples

RESTful HTTP vs. RPC over HTTP vs. WS-*/ESB (28)



Coupling Comparison Summary (29)

Coupling	REST	RPC	WS-*/ESB
Loose	10	3	4
Tight	0	4	5
Design-Specific	2	5	3



Conclusions

The Web and Web Services (31)

- The Web is only paralleled by the international phone system
  - it looked like a step back from sophisticated GUIs and CORBA
  - "dumbing down" GUIs creates powerful network effects
  - enterprise IT experiences a crisis caused by weight and rigidity
- *Web Services* were invented to use the Web
  - they were designed and built based on a non-Web architectural style
  - in their WS-* flavor, they use the Web as transport layer
  - as an alternative, *RESTful Web services* are built like the Web
- RESTful Web services are not "the best solution"
  - but they may be good enough and a good fit for "loose coupling"
- How to decide when to use the Web and when to use something else?

Future Work (32)

- Find *more facets* and how they affect coupling
- Be more disciplined in exploring *facet dependencies*
- Come up with a *better visualization* of facets

Summary (33)

- Many service-related decisions can be regarded as *integration* vs. *cooperation*
- *Distribution* and *Federation* are part of many scenarios
- Information system architectures have different goals
  - implementing *one integrated logical system* with the goal of hiding distribution/heterogeneity
  - enabling *cooperation among communicating peers* with little control of the individual peers
- Architectures and architectural styles are driven by requirements
  - there is no such thing as the “best architectural style”
  - there is no such thing as the “best architecture”
  - architectural styles often are implicit by background or tradition