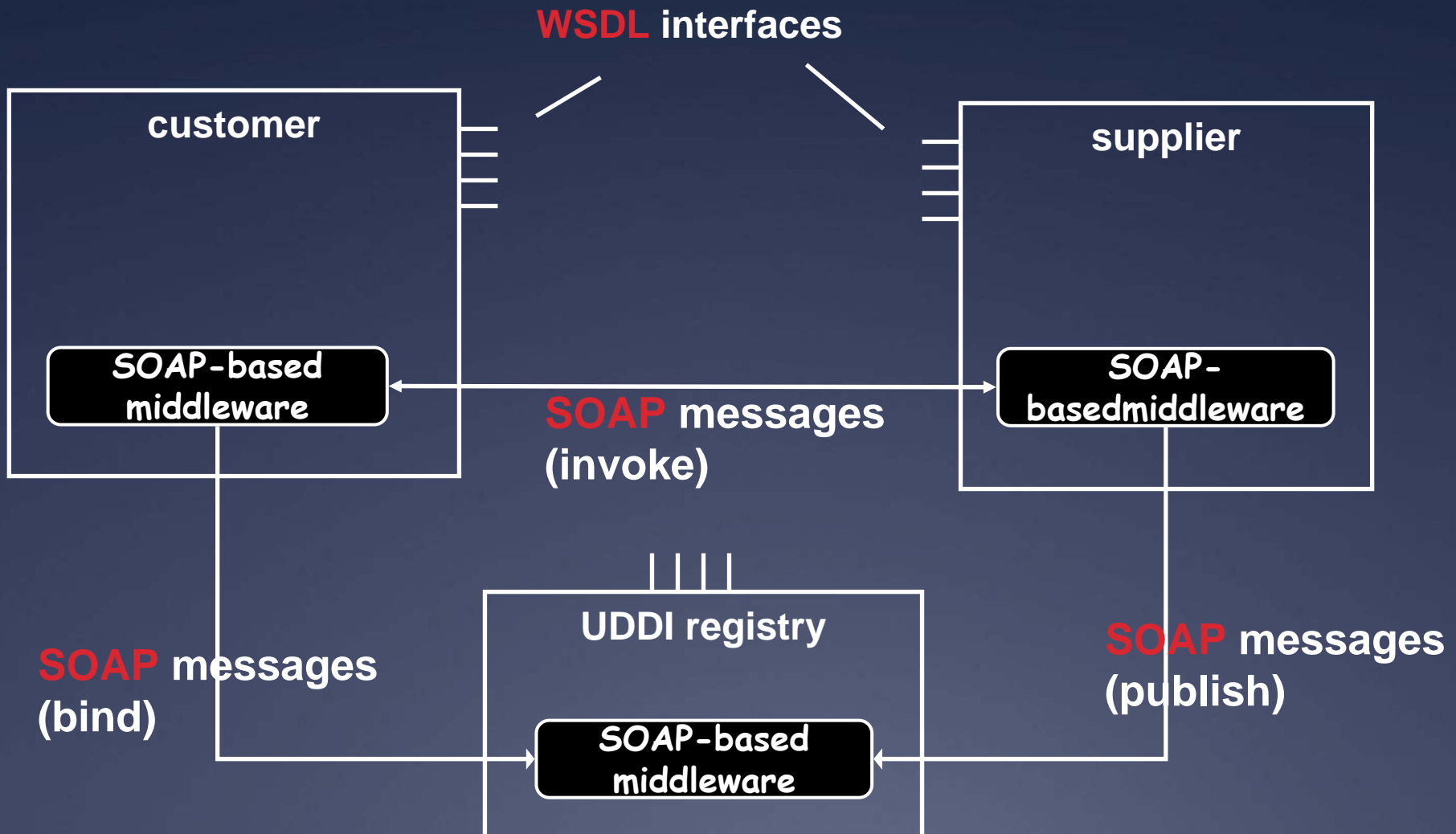


Web Languages

Part II – Advanced Topics

The Journey till now...



- * Do we need all this?
- * Is this all we need?



Part II

- * Approach to designing service-oriented systems
- * Properties of the interaction
 - * Secure
 - * Reliable
 - * Routing
 - * ...
- * Policies
- * Coordination
 - * Business Protocols
 - * Transactions
 - * Service Bus
- * Composition and Mashups

Part II - Project

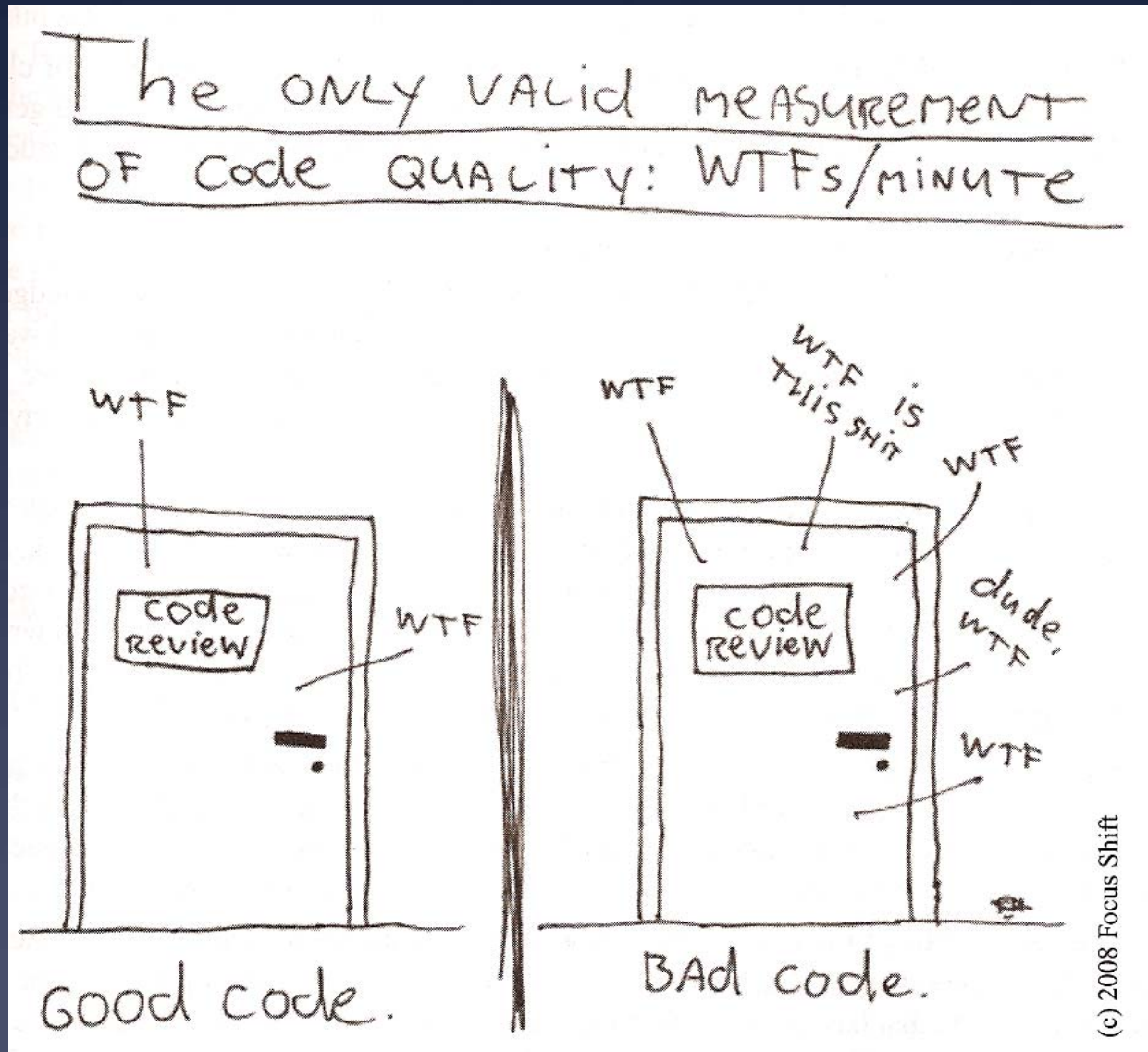
- * Develop a useful service combining building blocks
- * Publish them as web services and make them available as Web pages
- * Adopt interface-first approach
- * BPEL

A note on interfaces

The art of KISSing



Quality of code (and of interfaces)



“good” code (and interfaces)

- * Despite deadlines, take time to write good code.
- * Good code does not help only for doing maintenance next year, it will help you doing “maintenance” in a couple of hours.
 - * LeBlanc’s law: later equals never
 - * Robert Martin: “Take the time to go fast”
 - * Poor code will slow you down even before the deadline

Characteristics of good code

- * **Stroustrup**: elegant, straightforward (make it hard for bugs to hide)
- * **Thomas**: can be read and enhanced by somebody else. Includes testing. Minimal dependencies, Minimal API
- * **Featers**: looks like written by someone who cares
- * **Booch**: *reads like well-written prose*

(from "Clean Code", by Robert Martin. Read also "effective java", by Joshua Bloch)

Google Big Table

9 Lessons

In t
and
and

One lesson we learned is that large distributed sys-

The most important lesson we learned is the value of simple designs. Given both the size of our system (about 100,000 lines of non-test code), as well as the fact that code evolves over time in unexpected ways, we have found that code and design clarity are of immense help in code maintenance and debugging. One exam-

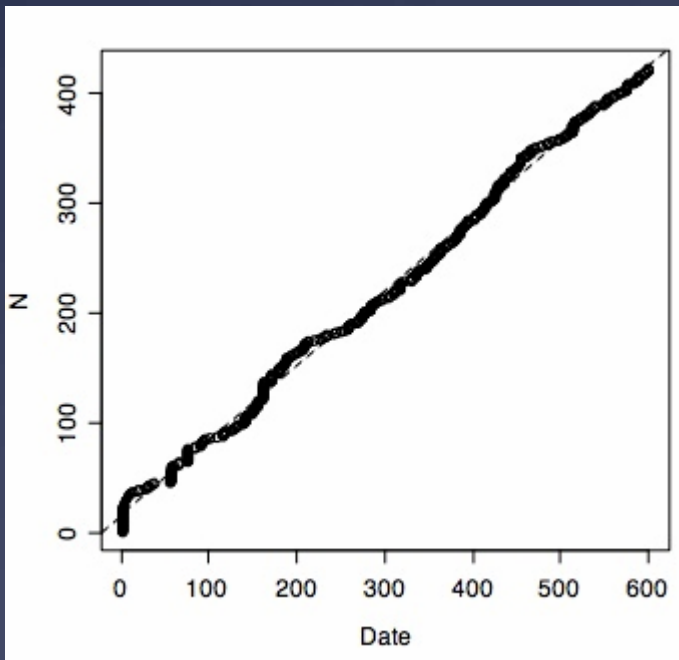
What about Web services interfaces

- * The same, but with MORE emphasis on simplicity
- * SERVICE DESCRIPTION AND INTERFACES SPECS ARE WRITTEN FOR OTHERS TO READ. THINK ABOUT THE READER
- * If the specs are not clear, it's your fault
- * If the specs are not clear, your service does not get used
- * REMEMBER THE HOW TO

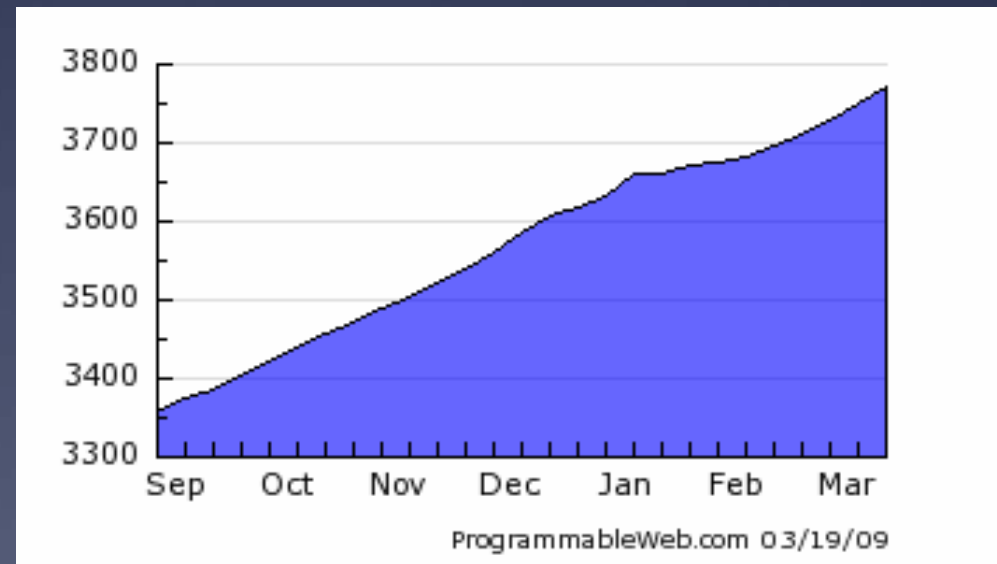
Dynamics of the ecosystem

- * Constant growth since programmableweb.com went online (over 600 days) [by Michael Weiss, Carleton University]

Number of APIs

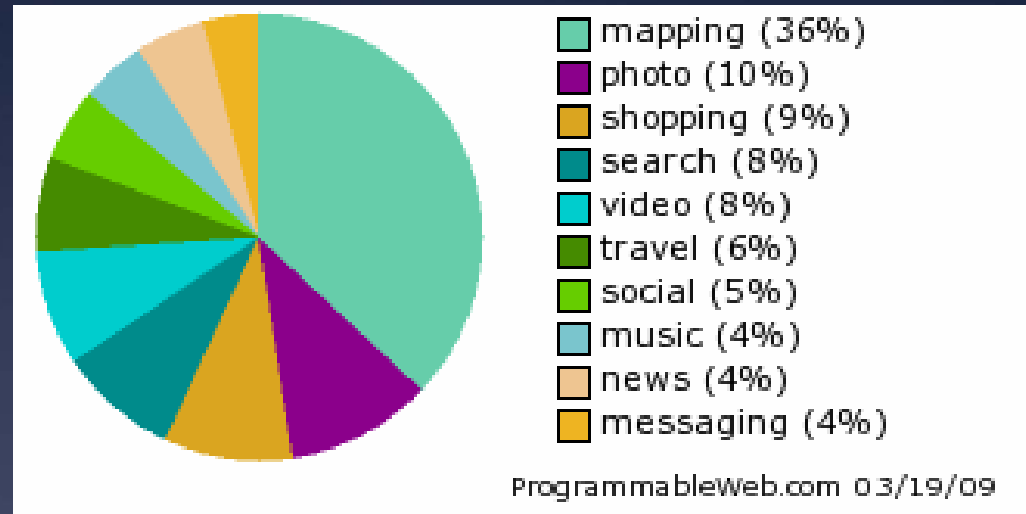


Number of mashups

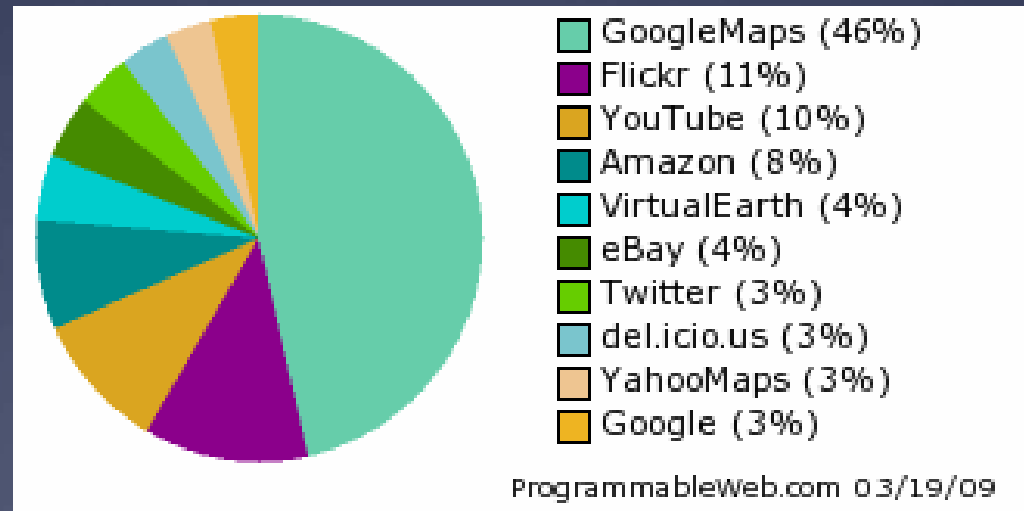


Some figures (programmableweb.com)

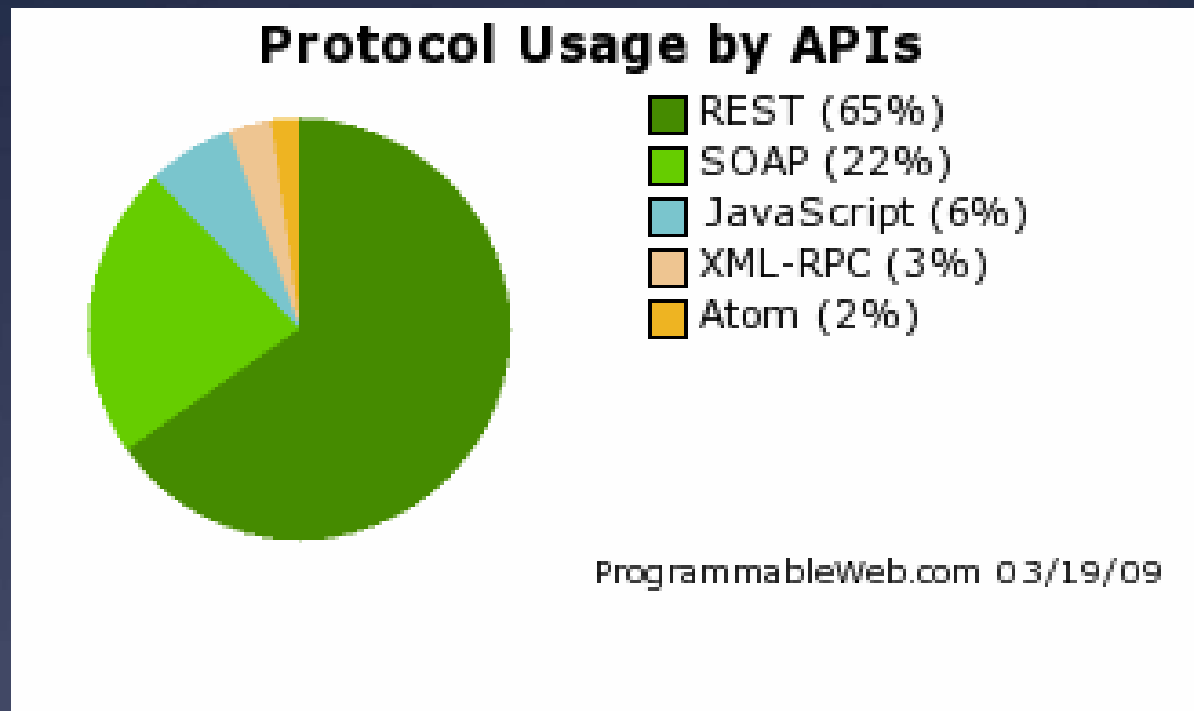
- Most popular categories of **mashups**



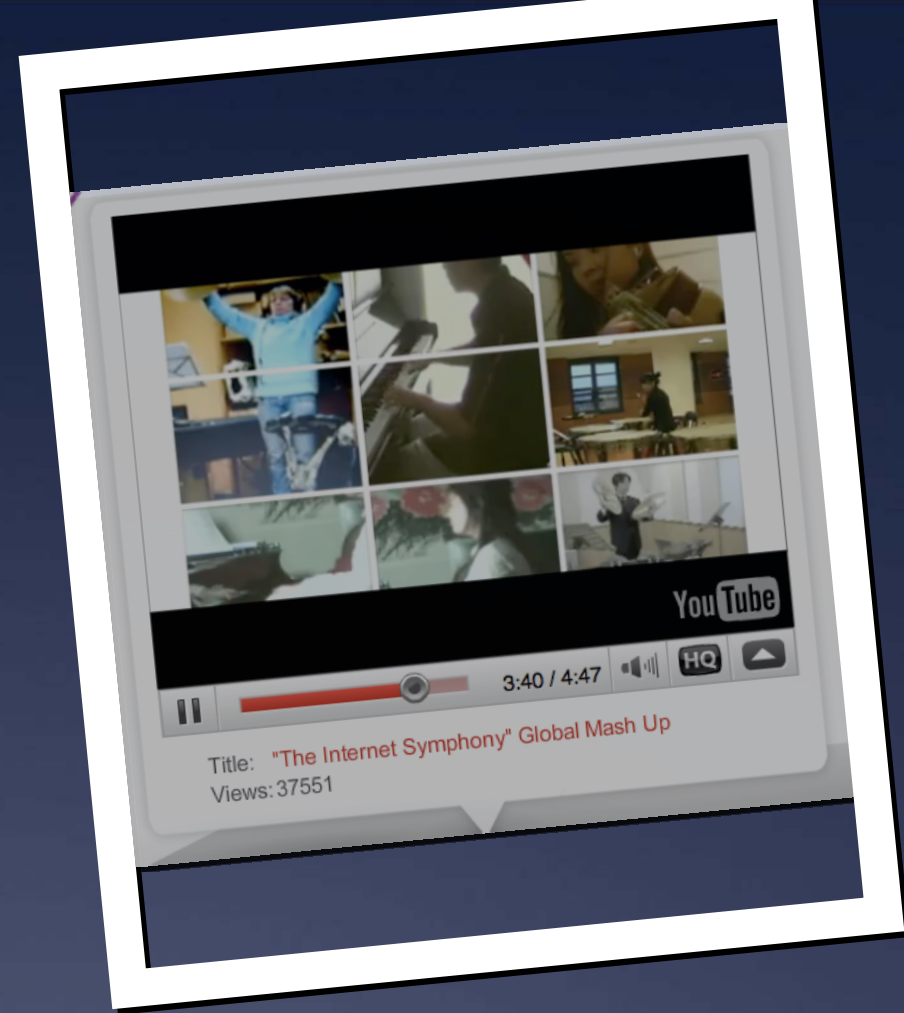
- * Most popular web **APIs**



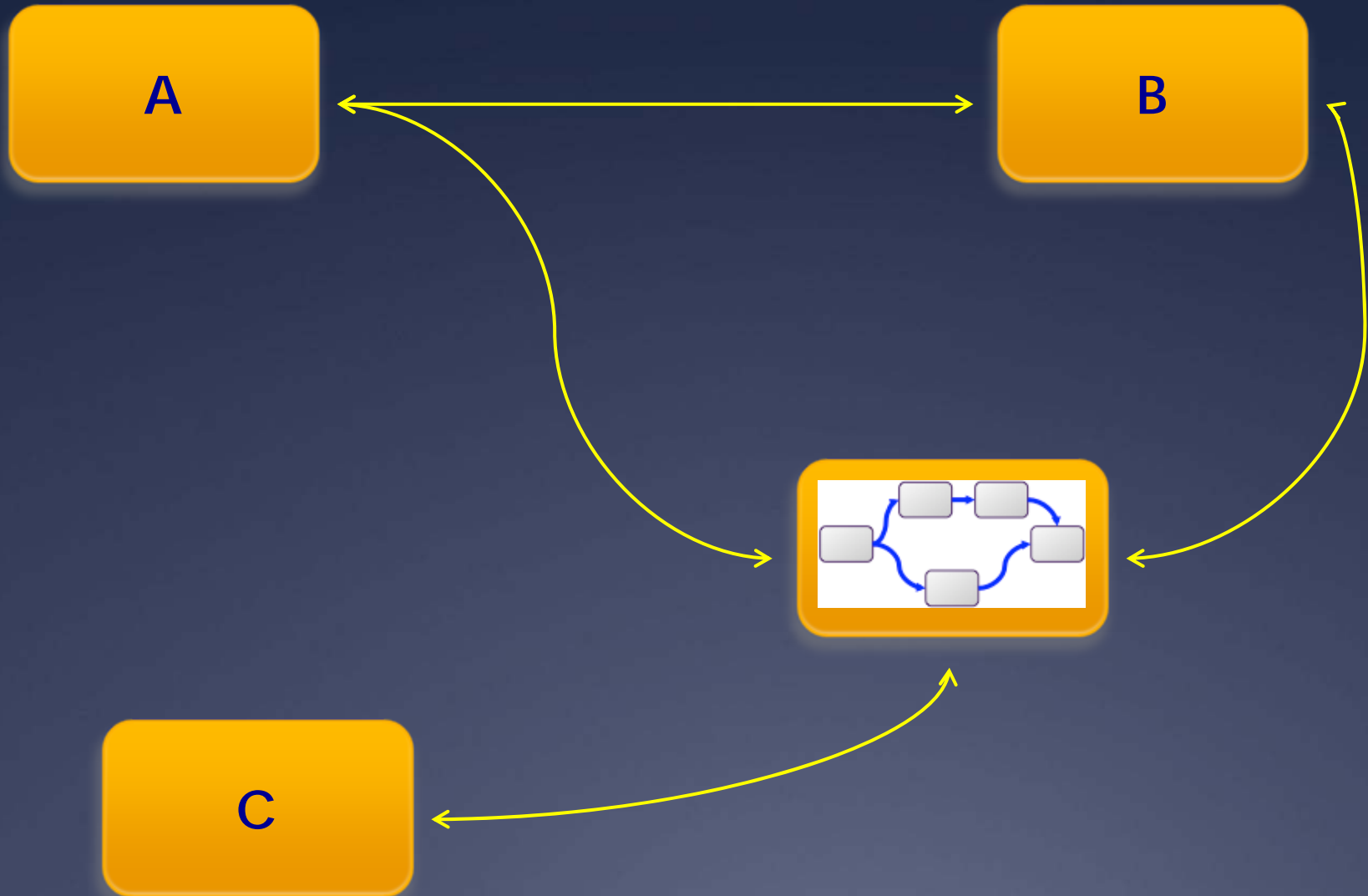
“Protocol” usage by APIs



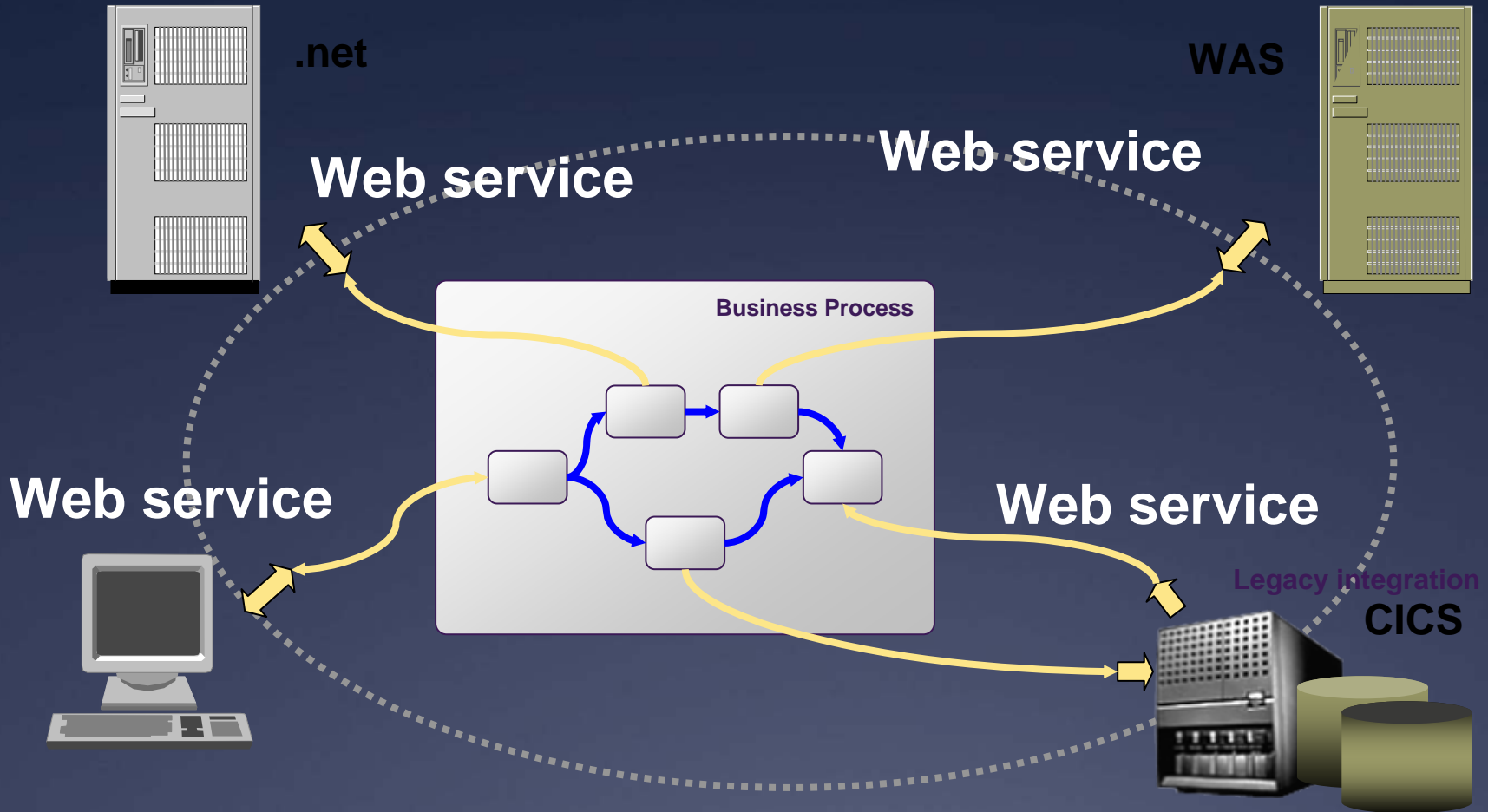
Composition and Mashups



Choreography vs Composition



Service composition



WS-BPEL defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. The interaction with each partner occurs through Web Service interfaces.

(Enterprise) Application Integration

