Università della Svizzera italiana

Faculty of Informatics

Component Interfaces

Prof. Cesare Pautasso

http://www.pautasso.info

cesare.pautasso@usi.ch

@pautasso

Design Advice

- Keep it simple
  - Do One Thing and do it well
  - Do not surprise clients
- Keep it as small as possible but not smaller
  - When in doubt leave it out
  - You can always add more later
- Maximize information hiding
  - API First
  - Avoid leakage: implementation should not impact interface
- Names Matter
  - Avoid cryptic acronyms
  - Use names consistently
- Keep it consistent
  - Naming Conventions
  - Argument Ordering
  - Return values
  - Error Handling
- Follow the conventions of the underlying platform
- Document Everything
  - Classes, Methods, Parameters
  - Include Correct Usage Examples
  - Quality of Documentation critical for success
- Make it easy to learn and easy to use
  - without having to read too much documentation

Where to find APIs?

- Operating Systems
- Programming Language Runtimes
- Hardware Access
- User Interfaces
- Databases
- Web and Cloud Services

Operating Systems

- Win16, Win32 , Win64
- POSIX
- Cocoa (OS/X)
- android.os

Programming Languages

Standard libraries of any language runtime

- C `#include <stdlib.h>`
- Java Platform API (SE, EE)
- libstdc++
- Python Standard Library
- .NET Framework class library

Hardware Access

Abstractions to program any kind of hardware device

- Graphics (OpenGL , WebGL, OpenCL, CUDA)
- Network (NDIS)
- Printers (CUPS)
- Device Drivers (WDF, I/O Kit)

User Interfaces

Widgets, Gadgets, Controls

- Tcl/Tk, Qt, GTK+
- Java Swing, AWT, SWT
- Windows MFC, WPF
- JavaScript HTML5 APIs

Databases

Standardized database access

- JDBC
- ODBC
- PHP PDO

Web Services

Remote access through standardized protocols:

- SOAP/WSDL Services
- REST/Hypermedia Services
- JSON-RPC/HTTP Services

Examples: Google, Amazon WS, Facebook Graph, Twitter Firehose, Salesforce

Only one chance...

...to get the design right:

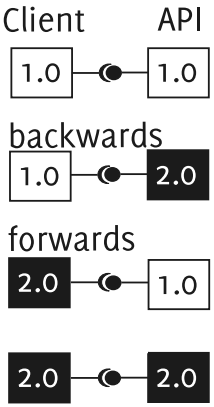
- Application Programming Interface
- Programming Language (and Standard Library)
- External Data Model:
  - File/Document Format
  - Database Schema
  - Wire/Message Representation Format
- Once the API becomes public, it is out of your hands and it will be very expensive to change!

API Evolution

- Once in the API, keep it there forever
  - Never add something you do not intend to keep forever
  - Easier to add than to remove
- Keep changes backwards and forwards compatible
- Make sure you explicit version all changes (including documentation) pointing out incompatibilities
- Publish 0.x versions of an API before freezing it to get early feedback from users
- Rename a component if its API has changed too much so you can start the redesign from scratch without breaking old clients

API Compatibility

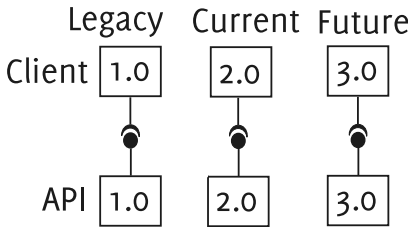
- Backwards compatibility:
  new version of API compatible with old client
- Forwards compatibility:
  old version of API compatible with new client



3 Versions Rule

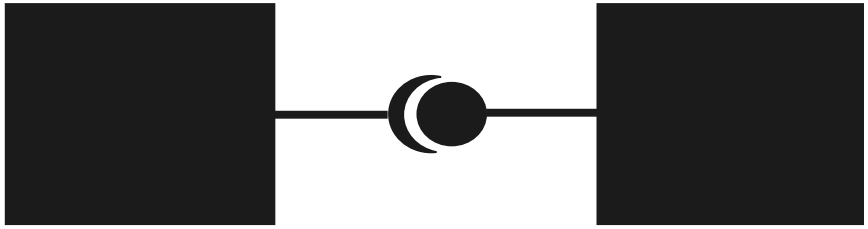
Up to 3 versions of an API exist at the same time:

- Legacy v1.0: migrate old clients
- Current v2.0: normal operation
- Future v3.0: develop new experimental clients



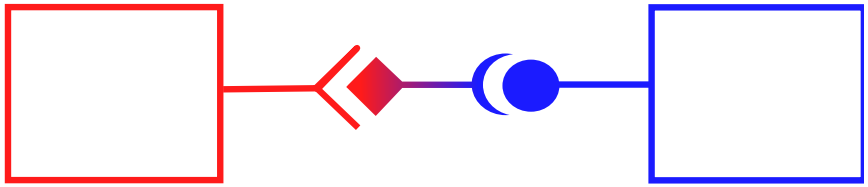
Before v4.0 can be added, v1.0 needs to be retired

Compatible Interfaces



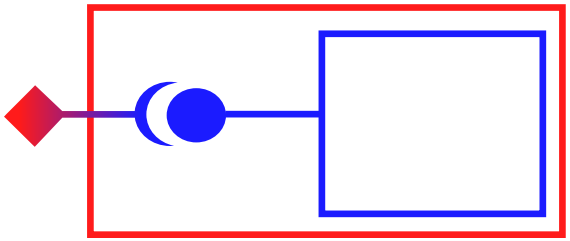
- To be connected, component interfaces need to match perfectly

Adapter



- Adapters help to connect mismatching interfaces
- Adapters deal with transforming the interaction provided by one to the interaction required by the other interface
- Warning: if data or functionality are missing the adaptation may be impossible

Wrapper



- The mismatching end of the adapter is hidden inside a wrapper component
- The adaptation logic is encapsulated within the wrapper and made reusable

Mismatch Example

```
id upload(user, image);
image download(id);
setTitle(id, title);
title getTitle(id);
ids[] list(user);
```

A

```
id upload(user, image, title);
{user, title, time} getImageMetaData(id);
image getImageData(id);
ids[] list();
```

B

Are Interfaces A and B equivalent?

References

- William Brown, Raphael Malveau, Hays McCormik III, Thomas Mowbray, Anti-Patterns, Refactoring Software, Architectures, and Projects in Crisis (http://sourcemaking.com/antipatterns) , Wiley, 1998
- Joshua Bloch, How to Design a Good API and Why it Matters, Google Tech Talk Slides (http://www.scribd.com/doc/33655/How-to-Design-a-Good-API-and-Why-it-Matters) Video (http://video.google.com/videoplay?docid=-3733345136856180693)
- Michi Henning, API: Design Matters, ACM Queue, Vol 5, No 4, May/June 2007
- Will Tracz, Confessions of a Used Program Salesman, Addison-Wesley, 1995
- Jaroslav Tulach, Practical API Design: Confessions of a Java Framework Architect, APress, 2008, ISBN 1-4302-0973-9