Selected Idioms and Design Patterns

The Pimpl Idiom

Lesson Objectives

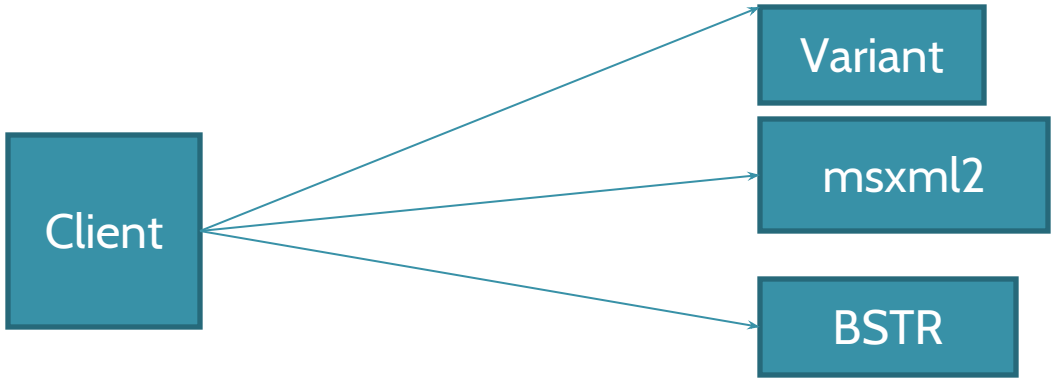
- You will understand and be able to use the PImpl idiom

The Pimpl Idiom

- Also called
  - Opaque Pointer
  - Cheshire Cat
  - Compiler Firewall idiom
- Pimpl is a way to hide implementation details of an interface from clients
  - The implementation may be changed without the need to recompile the modules using it
  - This is important for providing binary code compatibility through different versions of a shared library, for example

Evolution of design to PImpl

- Real life example
- First cut – clients use msxml2 directly
  - Issues – msxml2 uses COM interface
  - COM is ugly to use from C++
    - Resource management
    - Lots of BSTRs, VARIANTS
    - Hard to get right memory management
    - Error prone

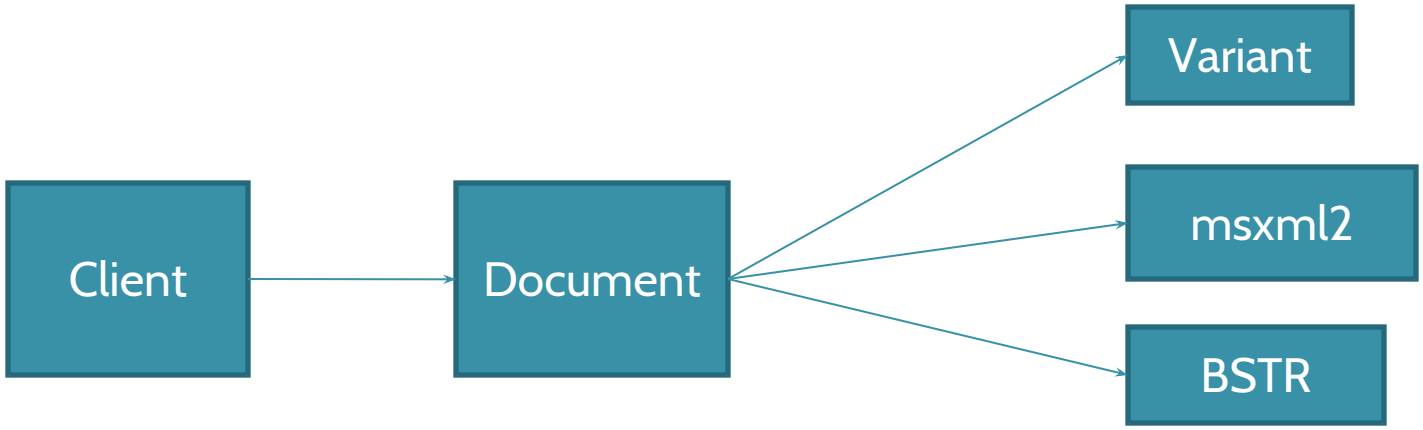


Problems with design

- COM complexity passed to all 50 or so clients in the system
- Half the lines of code in clients handling COM interface
  - Obscures the XML handling
- When we updated to msxml4 we had to change all clients
- When we found bug in memory management it had to be fixed in all 50 clients

First solution: Use wrapper

- Changed design to:

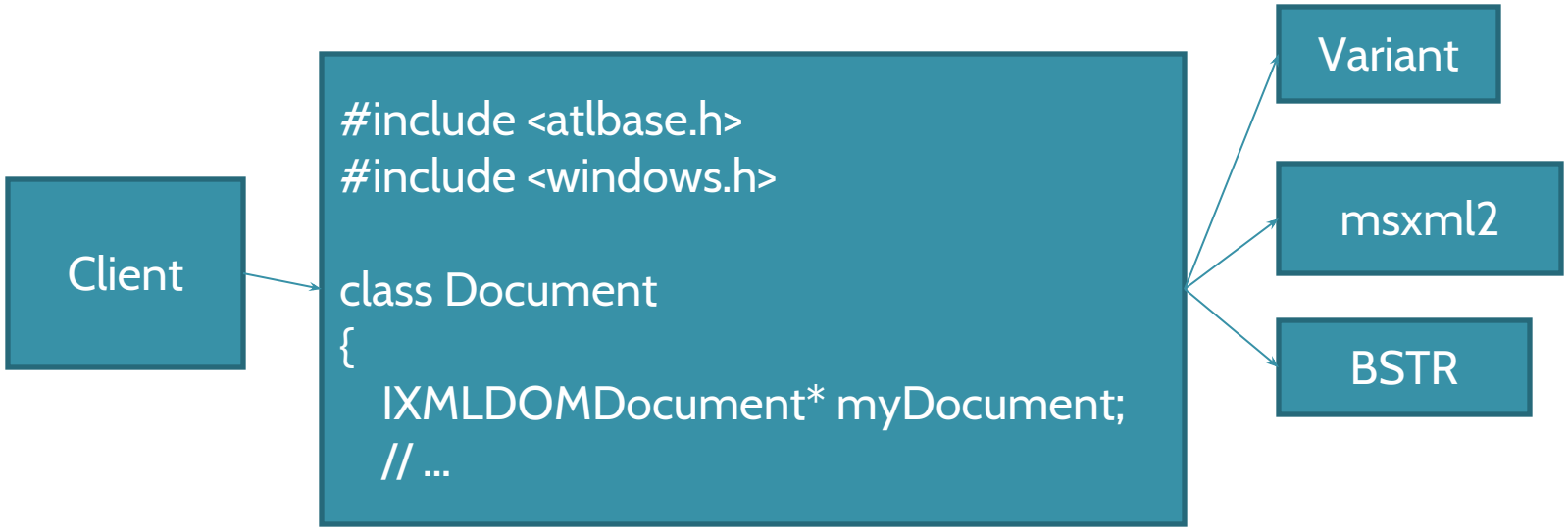


Benefits

- Lines of code reduced by about half in each client
- Client code is now readable
  - All the COM interface stuff is hidden from clients and encapsulated in new Document class
  - Only XML parsing stuff remains
  - Using C++ data types such as string

Design could be better yet

- When change made in Document, all clients must be recompiled before change can be tested
  - Takes about 3 hours on our system
- Why the need for recompilation?



Introducing PImpl

- Introduce a wrapper class that accesses private implementation via an opaque pointer
  - Rename Document as DocumentImpl
  - Create new class Document with same API as DocumentImpl
  - Document forwards all requests to its impl