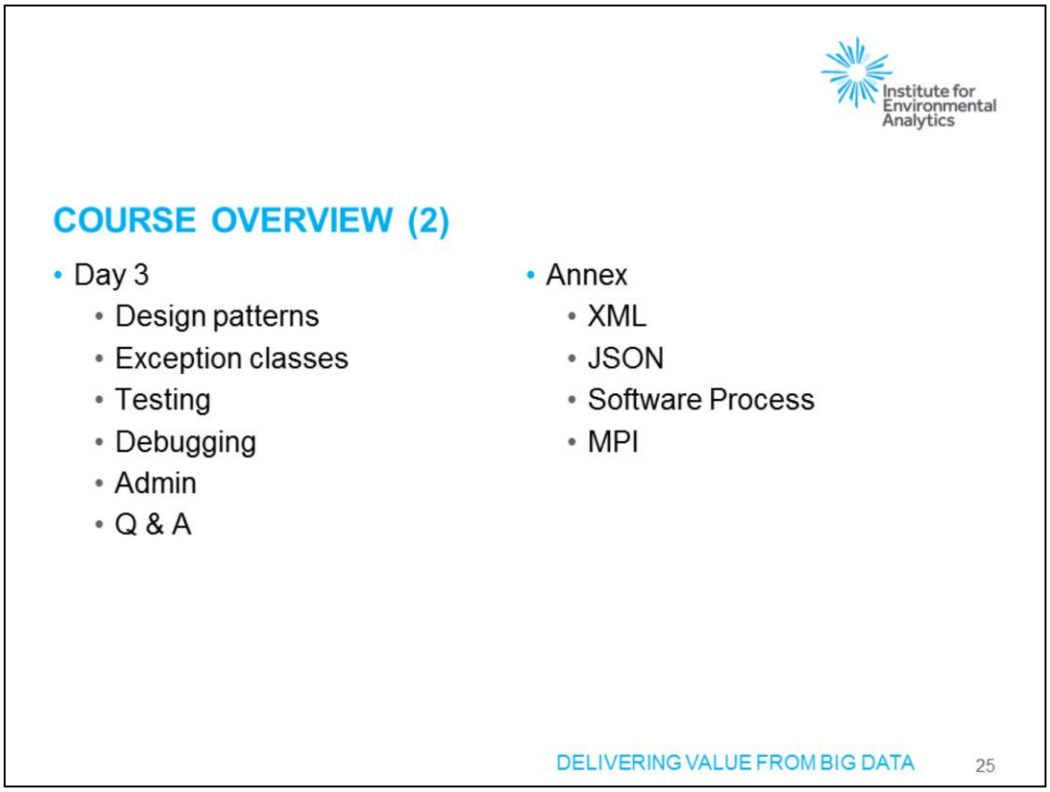
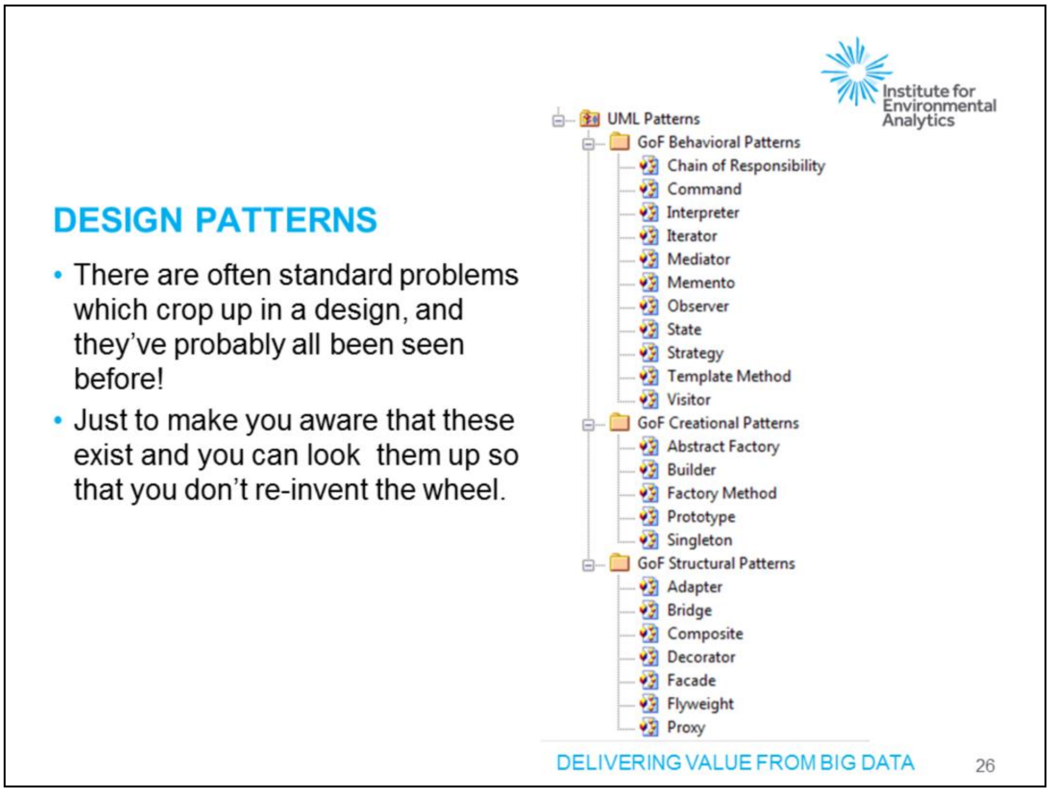


Institute for Environmental Analytics

COURSE OVERVIEW (2)

- Day 3
  - Design patterns
  - Exception classes
  - Testing
  - Debugging
  - Admin
  - Q & A
- Annex
  - XML
  - JSON
  - Software Process
  - MPI



DESIGN PATTERNS

- There are often standard problems which crop up in a design, and they've probably all been seen before!
- Just to make you aware that these exist and you can look them up so that you don't re-invent the wheel.

Institute for Environmental Analytics
UML Patterns
GoF Behavioral Patterns
Chain of Responsibility
Command
Interpreter
Iterator
Mediator
Memento
Observer
State
Strategy
Template Method
Visitor
GoF Creational Patterns
Abstract Factory
Builder
Factory Method
Prototype
Singleton
GoF Structural Patterns
Adapter
Bridge
Composite
Decorator
Facade
Flyweight
Proxy

DELIVERING VALUE FROM BIG DATA 26

Just to make you aware that you'll see a lot of references to design patterns, and also people calling various things 'anti-patterns'.

GoF refers to the 'Gang of Four' authors who wrote a very influential book: ***Design Patterns: Elements of Reusable Object-Oriented Software***
(http://en.wikipedia.org/wiki/Design_Patterns – this is well worth a read)

We'll introduce a few which may help address common problems:

Singleton

Factory

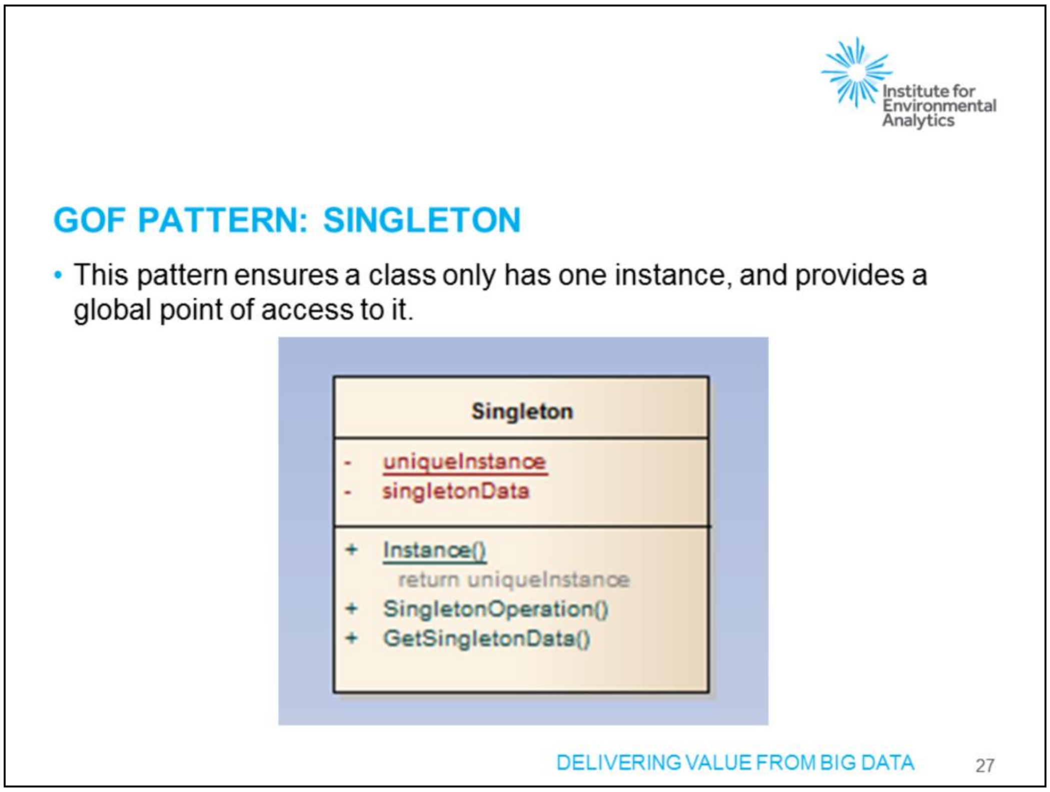
Visitor

Iterator

See this extremely useful quick guide to patterns, what they look like, when to use them and examples:

http://refcardz.dzone.com/refcardz/design-patterns

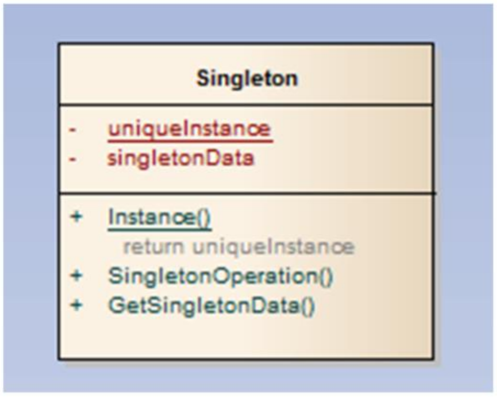
You can download the pdf.



Institute for Environmental Analytics

GOF PATTERN: SINGLETON

- This pattern ensures a class only has one instance, and provides a global point of access to it.



DELIVERING VALUE FROM BIG DATA 27

Singleton is known as a **creational** pattern - it's used to construct objects such that they can be decoupled from the implementing system. The definition of Singleton provided in the original Gang of Four book on Design Patterns states: *Ensure a class has only one instance and provide a global point of access to it.*

We just provide one point of access to create an instance of the Singleton class. The constructor is kept private, giving the getInstance( ) method the responsibility of providing access to the Singleton.

When you need to ensure there's one instance of an object, available to a number of other classes, you may want to use the Singleton pattern. Singletons are used a lot where you need to provide a registry, or something like a thread pool. Logging is also another popular use of Singletons, providing one single access point to an applications log file.

There are many downsides, and several authorities discourage the use of the singleton maintaining that it indicates poor design.

http://java.dzone.com/articles/design-patterns-singleton

This isn’t easy to do in Python!
http://stackoverflow.com/questions/6760685/creating-a-singleton-in-python

There are several ways of implementing a singleton, here’s another:

In Java:

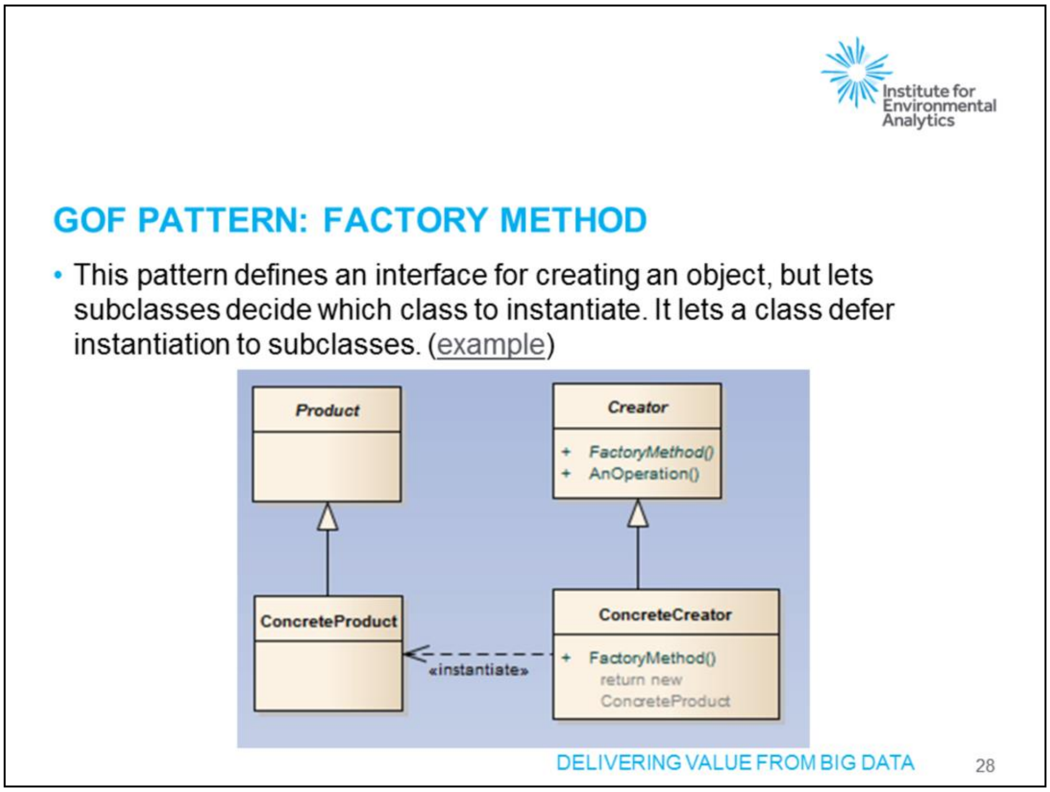
```
public class SingleObject {
        //create an object of SingleObject
        private static SingleObject instance = new SingleObject( );

        //make the constructor private so that this class cannot be
        //instantiated
        private SingleObject( ){}

        //Get the only object available
        public static SingleObject getInstance( ){
                return instance;
        }

        public void showMessage( ){ System.out.println("Hello World!");
        }
}
```

And then

```
public class SingletonPatternDemo {
        public static void main(String[ ] args) {
        //illegal construct
        //Compile Time Error: The constructor SingleObject( ) is not visible
        //SingleObject object = new SingleObject( );

        //Get the only object available
        SingleObject object = SingleObject.getInstance( );

        //show the message
```

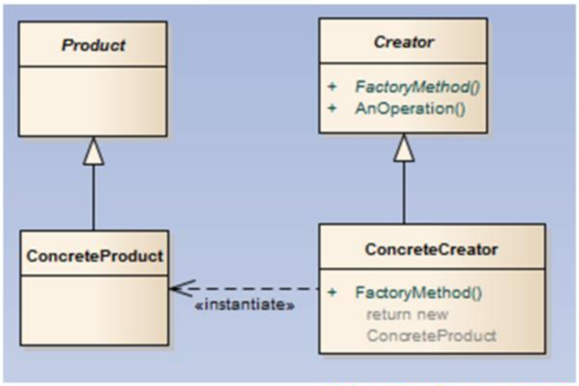
```
          object.showMessage( );
          }
}
```

(http://www.tutorialspoint.com/design_pattern/singleton_pattern.htm)



GOF PATTERN: FACTORY METHOD

- This pattern defines an interface for creating an object, but lets subclasses decide which class to instantiate. It lets a class defer instantiation to subclasses. (example)



DELIVERING VALUE FROM BIG DATA 28

Factory Method is known as a **creational** pattern - it's used to construct objects such that they can be decoupled from the implementing system. The definition of Factory Method provided in the original Gang of Four book on Design Patterns states: *Define an interface for creating an object, but let the subclasses decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses*

The **Creator** hides the creation and instantiation of the **Product** from the client. This is a benefit to the client as they are now insulated from any future changes - the Creator will look after all of their creation needs, allowing decoupling. Furthermore, once the Creator and the Product conform to an interface that the client knows, the client doesn't need to know about the concrete implementations of either. The factory method pattern really encourages coding to an interface in order to deal with future change.

Here the **Creator** provides an interface for the creation of objects, known as the factory method. All other methods in our abstract Creator are written only to operate on the **Product**s created in the **ConcreteCreator**. The **Creator** doesn't create the products - that work is done by its subclasses, such as **ConcreateCreator**.

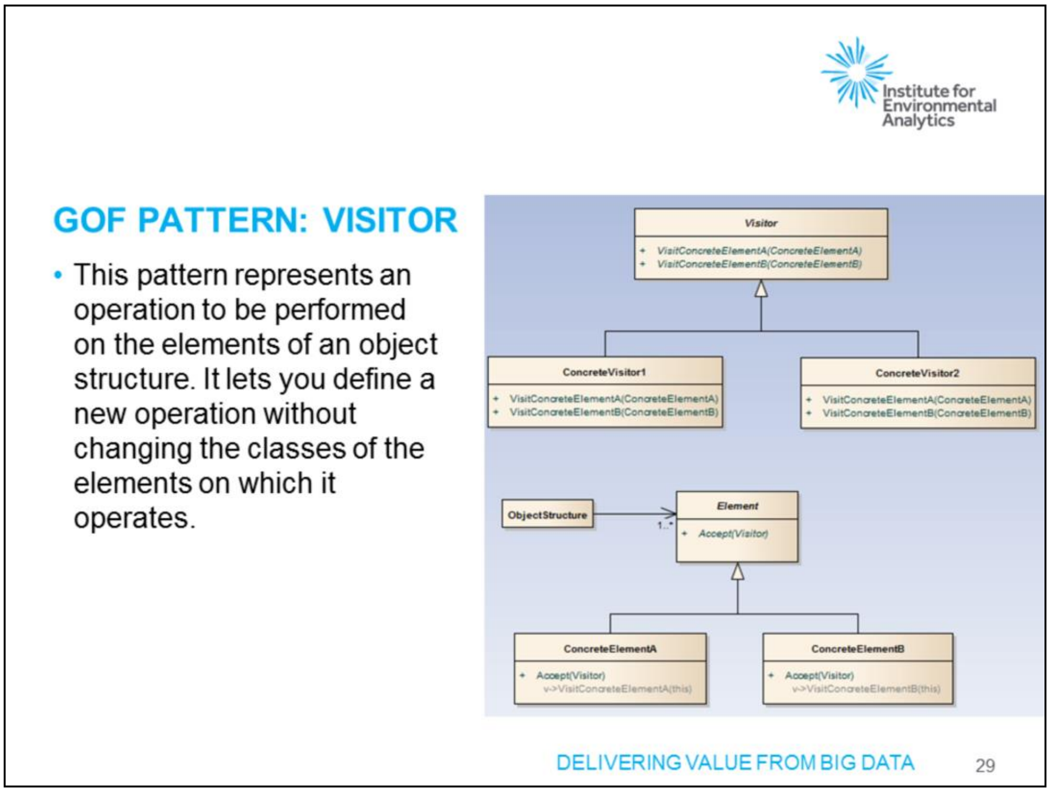
The idea behind the Factory Method pattern is that it allows for the case where a client doesn't know what concrete classes it will be required to create at runtime, but just wants to get a class that will do the job. The FactoryMethod builds on the concept of a simple Factory, but lets the subclasses decide which implementation of the concrete class to use. You'll see factories used in logging frameworks, and in a lot of scenarios where the client doesn't need to know about the concrete implementations. It's a good approach to encapsulation.

One thing not illustrated in the code example is the possibility to pass a paramater through to the creator in order to choose which type of concrete class to create. For example, we could have made FileLoggerCreator that could take a string parameter, allowing us to choose between XML and flat files.

http://java.dzone.com/articles/design-patterns-factory

----

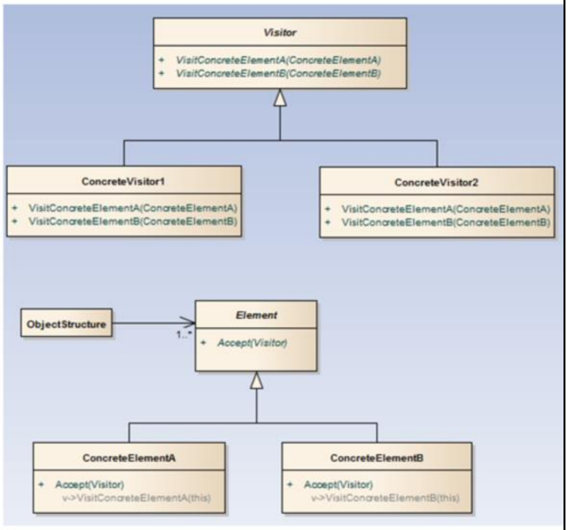
https://github.com/Reading-eScience-Centre/edal-java/blob/master/common/src/main/java/uk/ac/rdg/resc/edal/dataset/DatasetFactory.java



Institute for Environmental Analytics

GOF PATTERN: VISITOR

- This pattern represents an operation to be performed on the elements of an object structure. It lets you define a new operation without changing the classes of the elements on which it operates.


Visitor
+ VisitConcreteElementA(ConcreteElementA)
+ VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor1
+ VisitConcreteElementA(ConcreteElementA)
+ VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor2
+ VisitConcreteElementA(ConcreteElementA)
+ VisitConcreteElementB(ConcreteElementB)
ObjectStructure
1..*
Element
+ Accept(Visitor)
ConcreteElementA
+ Accept(Visitor)
v->VisitConcreteElementA(this)
ConcreteElementB
+ Accept(Visitor)
v->VisitConcreteElementB(this)

DELIVERING VALUE FROM BIG DATA 29

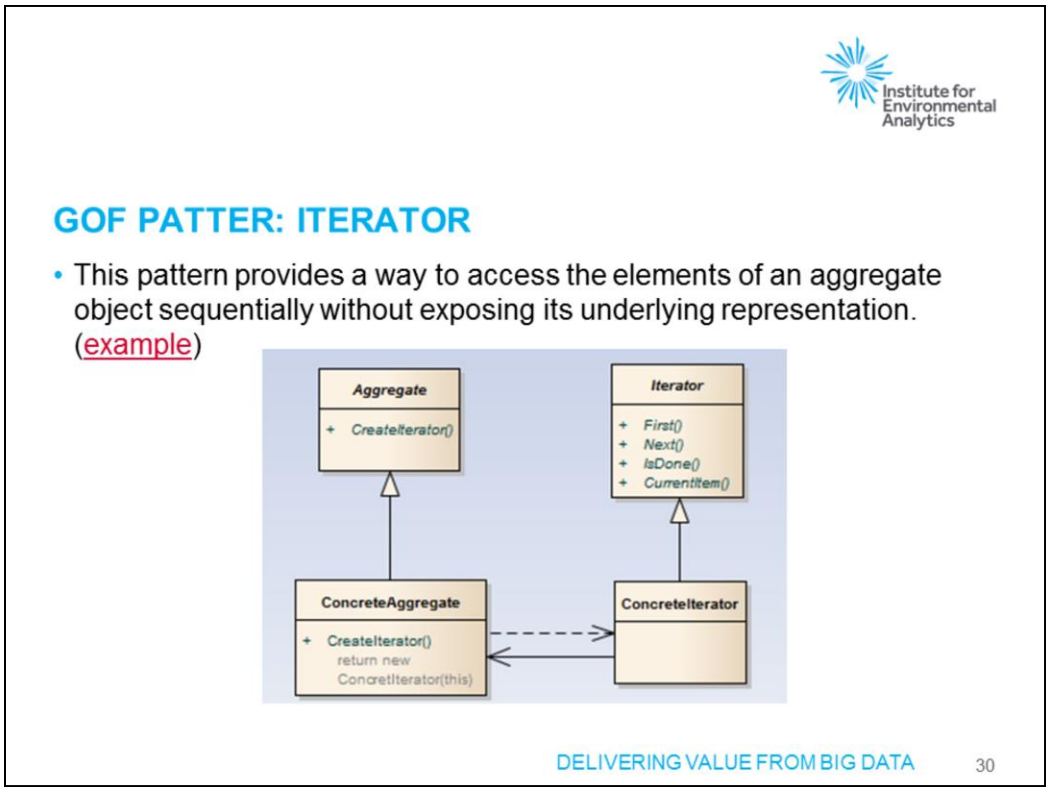
Shopping in the supermarket is a common example, where the shopping cart is your set of elements. When you get to the checkout, the cashier acts as a visitor, taking the disparate set of elements (your shopping), some with prices and others that need to be weighed, in order to provide you with a total.

The Visitor is known as a **behavioural** pattern, as it's used to manage algorithms, relationships and responsibilities between objects. The definition of Visitor provided in the original Gang of Four book on Design Patterns states: *Allows for one or more operation to be applied to a set of objects at runtime, decoupling the operations from the object structure.*

What the Visitor pattern actually does is create an external class that uses data in the other classes. If you need to perform operations across a disparate set of objects, Visitor might be the pattern for you. The GoF book says that the Visitor pattern can provide additional functionality to a class without changing it.

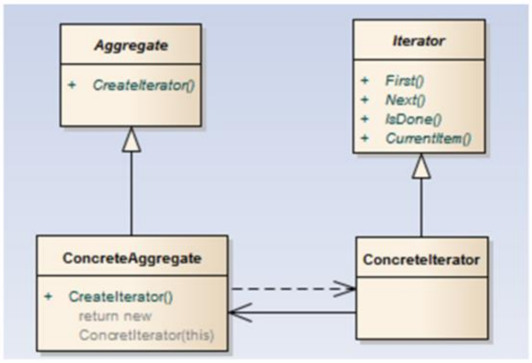
The core of this pattern is the **Visitor** interface. This interface defines a visit operation for each type of ConcreteElement in the object structure. Meanwhile, the **ConcreteVisitor** implements the operations defined in the Visitor interface. The concrete visitor will store local state, typically as it traverses the set of elements. The element interface simply defines an **accept** method to allow the visitor to run some action over that element - the **ConcreteElement** will implement this accept method.

http://java.dzone.com/articles/design-patterns-visitor



GOF PATTER: ITERATOR

- This pattern provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation. (example)



DELIVERING VALUE FROM BIG DATA 30

The Iterator pattern is known as a **behavioural** pattern, as it's used to manage algorithms, relationships and responsibilities between objects.. The definition of Iterator as provided in the original Gang of Four book on Design Patterns states: *Provides a way to access the elements of an aggregate object without exposing its underlying represenation.*

The **Aggregate** defines an interface for the creation of the Iterator object. The **ConcreteAggregate** implements this interface, and returns an instance of the ConcreteIterator. The **Iterator** defines the interface for access and traversal of the elements, and the **ConcreteIterator** implements this interface while keeping track of the current position in the traversal of the Aggregate.

Using this pattern, you can build on the standard concept of iteration to define special iterators that only return specific elements in the data set.

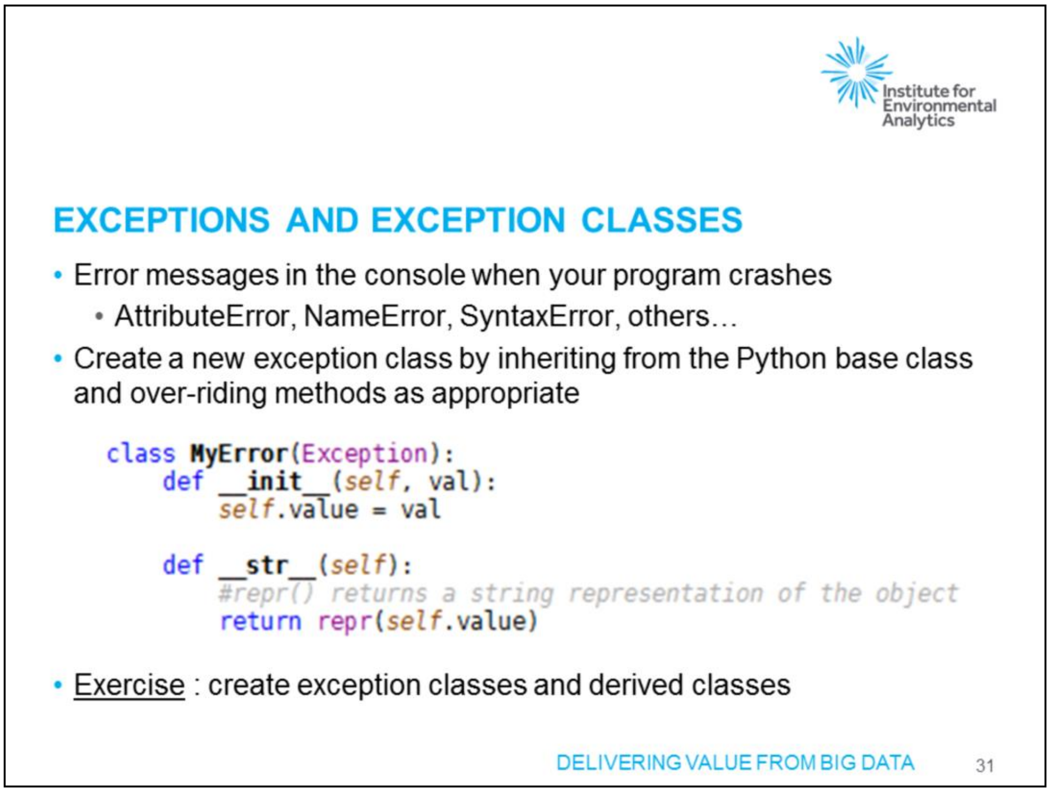
This pattern is useful when you need access to elements in a set without access to the entire representation. When you need a uniform traversal interface, and multiple traversals may happen across elements, iterator is a good choice.

It also makes you code much more reasonable, getting rid of the typical for loop syntax across sections of your codebase.

http://java.dzone.com/articles/design-patterns-iterator

----

https://github.com/Reading-eScience-Centre/edal-java/blob/master/common/src/main/java/uk/ac/rdg/resc/edal/util/Array1D.java


Institute for Environmental Analytics

EXCEPTIONS AND EXCEPTION CLASSES

- Error messages in the console when your program crashes
  - AttributeError, NameError, SyntaxError, others…
- Create a new exception class by inheriting from the Python base class and over-riding methods as appropriate

```
class MyError(Exception):
    def __init__(self, val):
        self.value = val

    def __str__(self):
        #repr() returns a string representation of the object
        return repr(self.value)
```

- Exercise : create exception classes and derived classes

DELIVERING VALUE FROM BIG DATA 31

Others include TypeError, IndentationError, IOError, KeyError, ZeroDivisionError and you've probably seen most of them!

https://docs.python.org/2/library/exceptions.html

How should we deal with them? The only reason they crash your program and appear on the console is because you're not catching and handling them. The mechanism to do this is pretty straightforward, but you have to make sure you are consistent. Not everyone thinks exceptions and their handling is a good idea, so read up on it and use it if you think it's worthwhile, often it's a case of appropriateness.

The code is

```
try:
        <your code>
except:
        <your code if any exception is thrown>
```

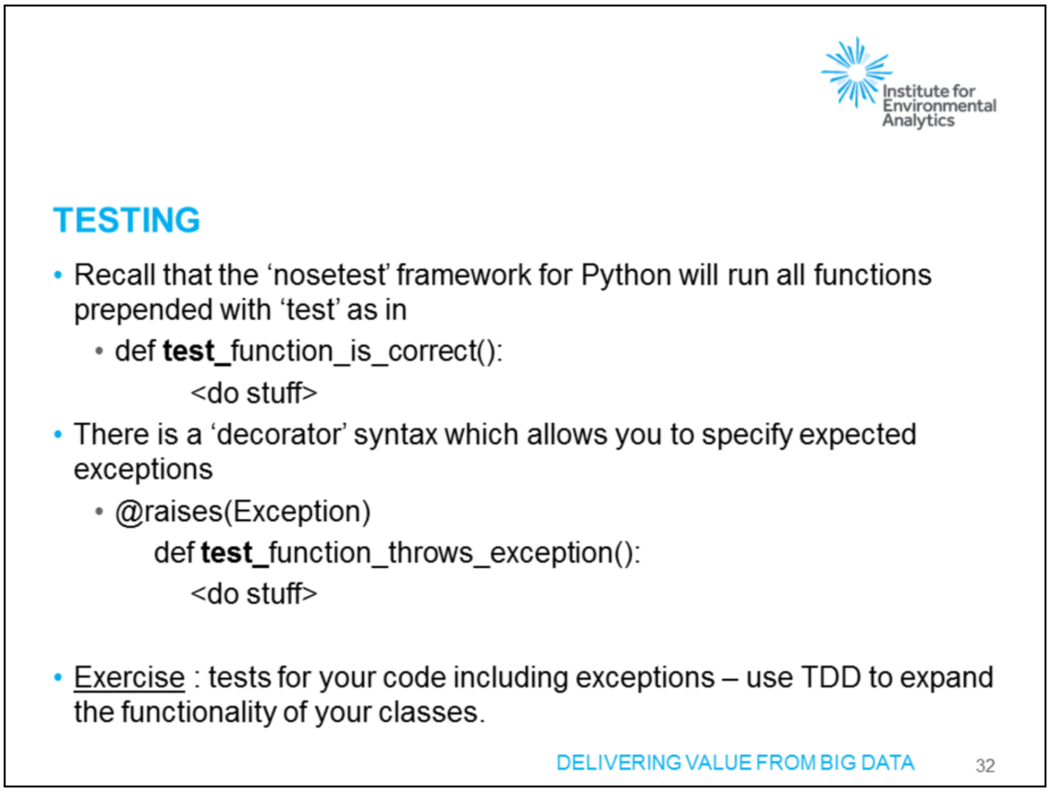
OR

```
except ValueError:
        <your code if a ValueError exception is thrown>
```

```
OR
        except ValueError as valErr:
                <as above, but you can interrogate 'valErr' for useful
information>
OR
        except (RuntimeError, TypeError, NameError):
                <your code to handle these three types of
exception>
OPTIONALLY
        else:
                <your code if the 'try' block succeeds and you don't
want to
                 catch more exceptions>
OPTIONALLY
        finally:
                <your code which *always* gets executed, even if
there's another
                 exception, or a break or a return statement>
```

You can use 'raise' either on its own, or with a specific exception to 'pass the buck' up the call stack.

https://docs.python.org/2/tutorial/errors.html



TESTING

- Recall that the ‘nosetest’ framework for Python will run all functions prepended with ‘test’ as in
  - def **test**_function_is_correct():

    <do stuff>
- There is a ‘decorator’ syntax which allows you to specify expected exceptions
  - @raises(Exception)

    def **test**_function_throws_exception():

    <do stuff>
- Exercise : tests for your code including exceptions – use TDD to expand the functionality of your classes.

We came across TDD in level 1, this time we’ll use it to drive the implementation of the classes we know we need. The Python ‘nosetests’ framework allows use of exceptions and checking that they get thrown when we think they do.

While we’re on the subject of ‘decorators’… this is a technique used in many languages whereby meta-data on a class or method or attribute may be specified.

See https://nose.readthedocs.org/en/latest/testing_tools.html for a list of those you can use in the nosetest framework. You may find the following particularly useful:

```
@with_setup(setup_func, teardown_func)
@istest(test_func)
@raises(exception_class)
```

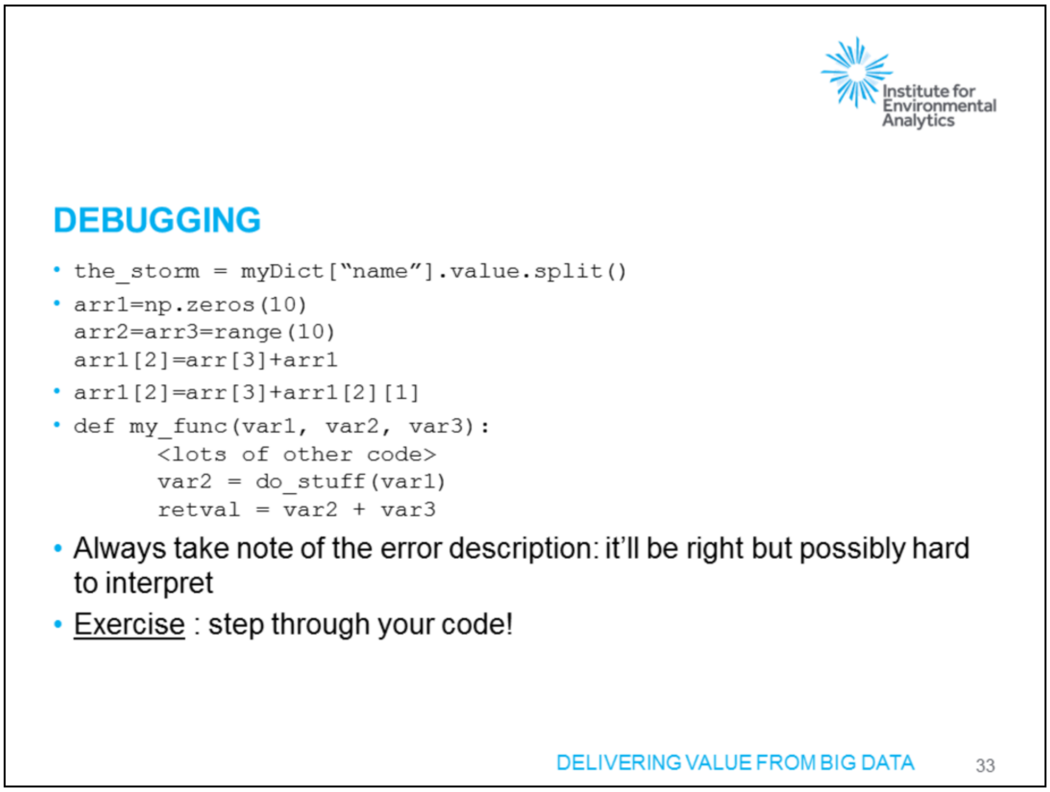
Because exceptions can be tricky, you may need to apply a little thought to the testing, the following examples are useful:

```
@raises(TypeError, ValueError)
def test_raises_type_error():
    raise TypeError("This test passes")
@raises(Exception)
def test_that_fails_by_passing( ):
    pass
```
And:
```
    from nose.tools import *
    l = [ ]
    d = dict( )

    @raises(Exception)
    def test_Exception1( ):
        '''this test should pass'''
        l.pop( )

    @raises(KeyError)
    def test_Exception2( ):
        '''this test should pass'''
        d[1]

    @raises(KeyError)
    def test_Exception3( ):
        '''this test should fail (IndexError raised but KeyError was
        expected)'''
        l.pop( )

    def test_Exception4( ):
        '''this test should fail with KeyError'''
        d[1]
```
And for testing a custom exception:
```
    def test_bad_token( ):
        sc = SomeClass('xxx', account_number)

        with assert_raises(MyCustomException) as e:
            sc.method_that_generates_exception( )
```

```
                assert_equal(e.exception.status, 403)
                assert_equal(e.exception.msg, 'Invalid User')
```

Where:

```
        class MyCustomException(Exception):
                def __init__(self, status, uri, msg=""):
                        self.uri = uri
                        self.status = status
                        self.msg = msg
                        super(Exception, self).__init__(msg)
```


Institute for Environmental Analytics

DEBUGGING

- `the_storm = myDict["name"].value.split()`

-
  ```
  arr1=np.zeros(10)
  arr2=arr3=range(10)
  arr1[2]=arr[3]+arr1
  ```
- `arr1[2]=arr[3]+arr1[2][1]`

-
  ```
  def my_func(var1, var2, var3):
          <lots of other code>
          var2 = do_stuff(var1)
          retval = var2 + var3
  ```
- Always take note of the error description: it'll be right but possibly hard to interpret
- Exercise : step through your code!

DELIVERING VALUE FROM BIG DATA 33

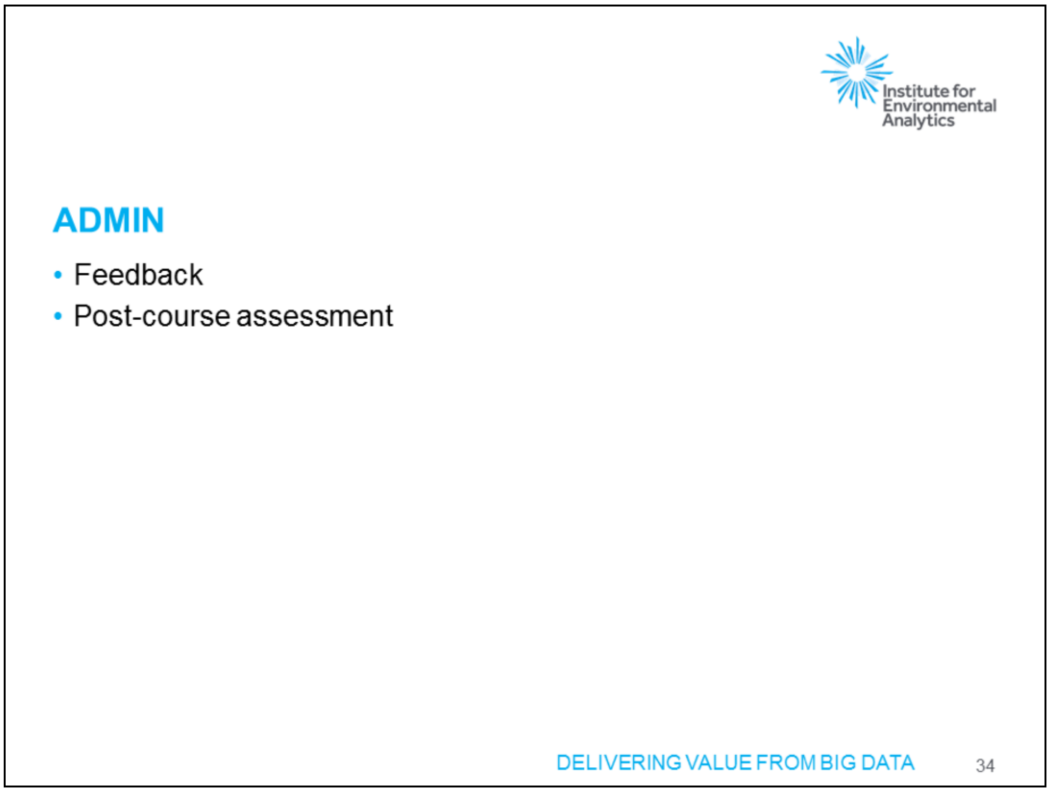
Using PyCharm

split( ) returns a list – need to dereference with a [0]

TypeError: unsupported operand type(s) for +: 'int' and 'list'

TypeError: 'int' object has no attribute '__getitem__'

Same variable name used… is this what you intend?



Institute for Environmental Analytics

ADMIN

- Feedback
- Post-course assessment

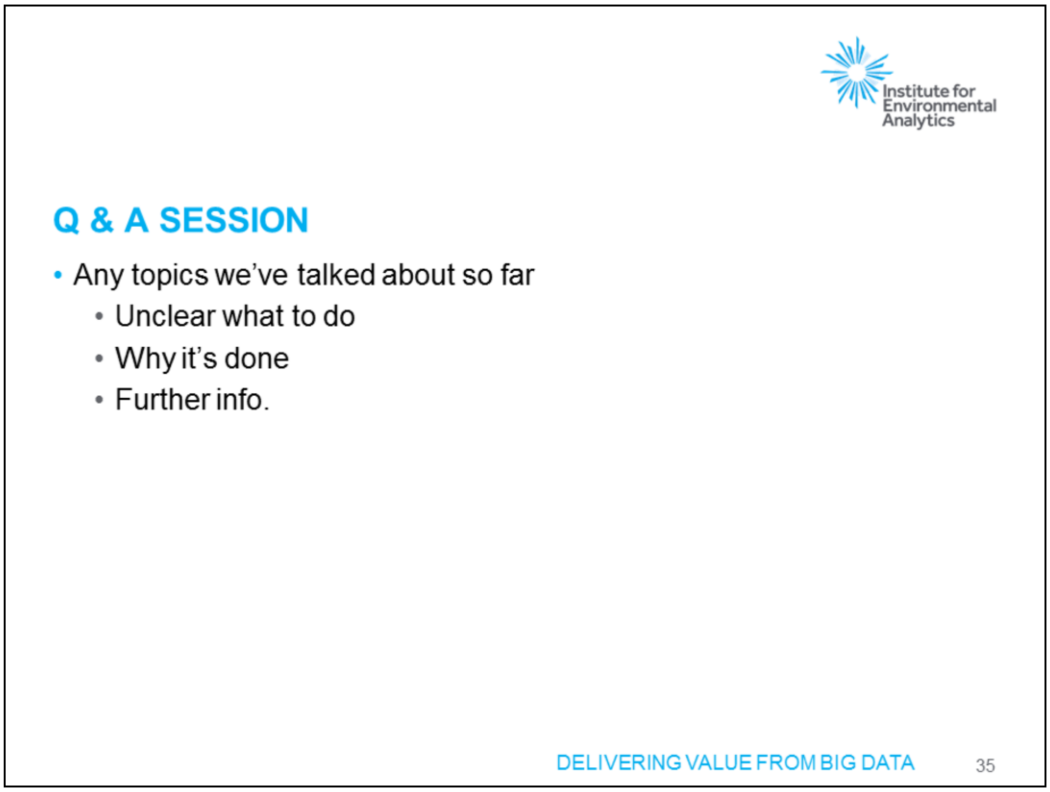
DELIVERING VALUE FROM BIG DATA 34

FEEDBACK

This is really important for us so that we can continue to make improvements. Please use the back of the sheet if you want to say anything not covered by the questions, or email us (resc@reading.ac.uk) or me (j.p.lewis@reading.ac.uk).

ASSESSMENT

This has the purpose of seeing how much you've picked up over the course, and seeing whether your approach to a software task has changed since the pre-course assessment (hopefully for the better!). There's no pass/fail ☺



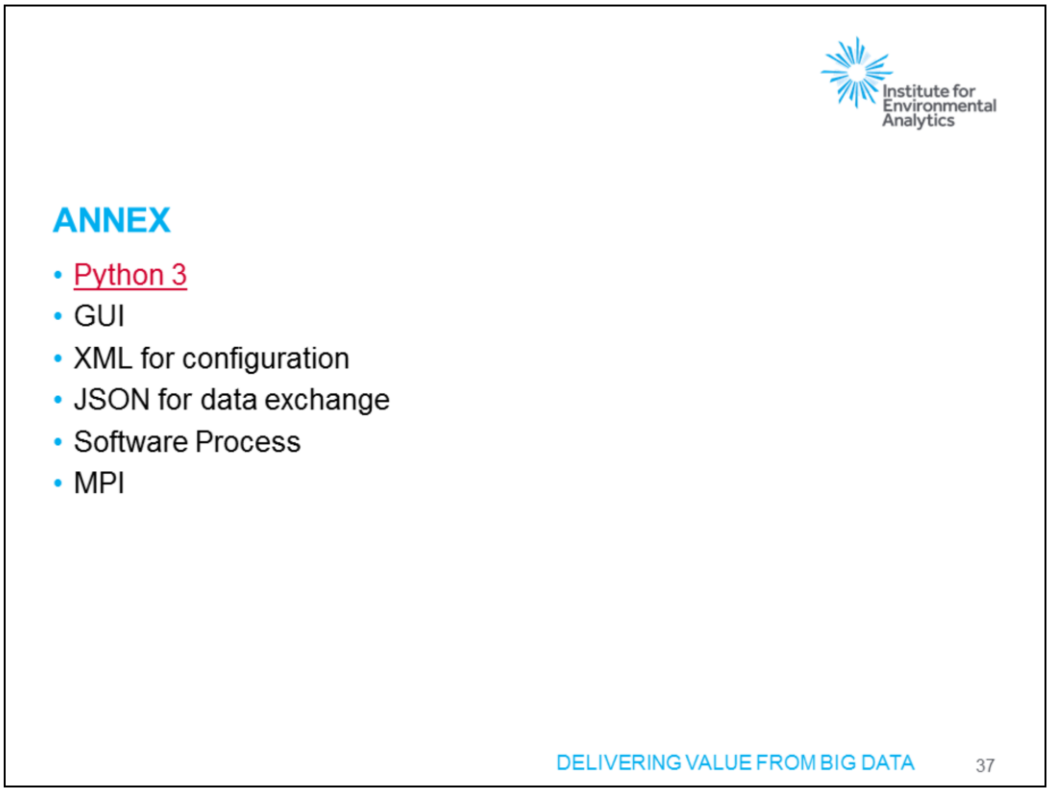
Institute for Environmental Analytics

Q & A SESSION

- Any topics we’ve talked about so far
  - Unclear what to do
  - Why it’s done
  - Further info.

DELIVERING VALUE FROM BIG DATA 35

Institute for Environmental Analytics


Institute for Environmental Analytics

ANNEX

- Python 3
- GUI
- XML for configuration
- JSON for data exchange
- Software Process
- MPI

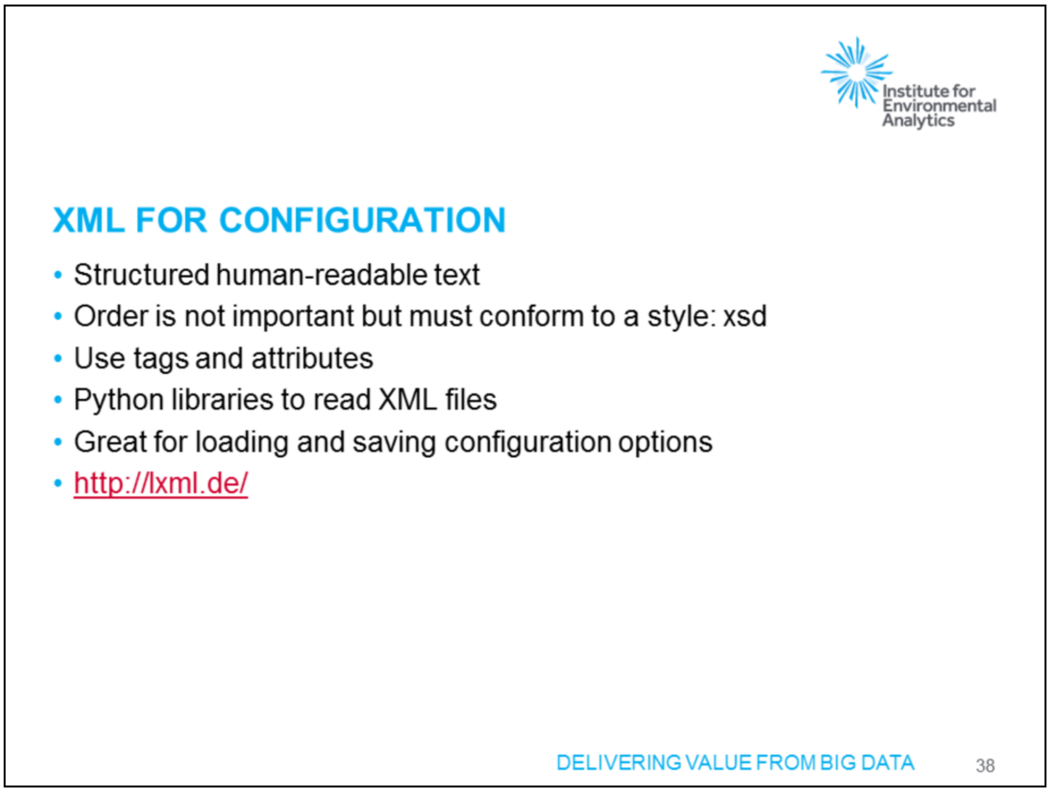
DELIVERING VALUE FROM BIG DATA 37

These are additional topics we don't have time to cover in our 3 days:

Python 3 doesn't appear to be a whole heap different from Python 2 for day-to-day use. There are fundamental alterations whereby code written for one is not compatible with the other. However, there are conversion routines you can run. This is a very good article on the main differences http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

The likelihood of anyone needing to develop a complex **GUI** is remote so we won't go into massive details on this. There are many many GUI packages to choose from in Python, probably too many: https://wiki.python.org/moin/GuiProgramming

If you start to do serious GUI work, you're probably better off switching to a language with more standardised support such as Java or C#. You'd then be more likely to find an IDE which will let you build your GUI graphically rather than having to code it from first principles. A good IDE will also make it easy to find the available callbacks for a particular UI control. Systems with a complex UI tend to be event driven with handlers for each of the control callbacks, and it is these which prod your code to do something. They are often designed using OO with handler classes, then task delegation to deeper classes. You may hear of the model-view-controller architectural pattern (http://en.wikipedia.org/wiki/Model-view-controller) which is a standard approach to GUI intense software using OO.



Institute for Environmental Analytics

XML FOR CONFIGURATION

- Structured human-readable text
- Order is not important but must conform to a style: xsd
- Use tags and attributes
- Python libraries to read XML files
- Great for loading and saving configuration options
- http://lxml.de/

DELIVERING VALUE FROM BIG DATA 38

http://infohost.nmt.edu/tcc/help/pubs/pylxml/web/index.html

http://www.diveintopython3.net/xml.html – good introduction to XML for those unfamiliar with it.

There's a lot of discussion about the 'best' way to use XML in Python, it's worth reading around if you want the full picture.

DOM, SAX, ElementTree and Xpath parsers for XML documents. They each have pros and cons and some have alternative meta-data representations of XML structure. Pick one!

We'll not use Xpath in the libxml2 module as it's more complex, but it's better if you want:

1. Compliance to the spec (http://www.w3.org/TR/xpath/)
2. Active development and a community participation
3. Speed. This is really a python wrapper around a C implementation.
4. Ubiquity. The libxml2 library is pervasive and thus well tested.

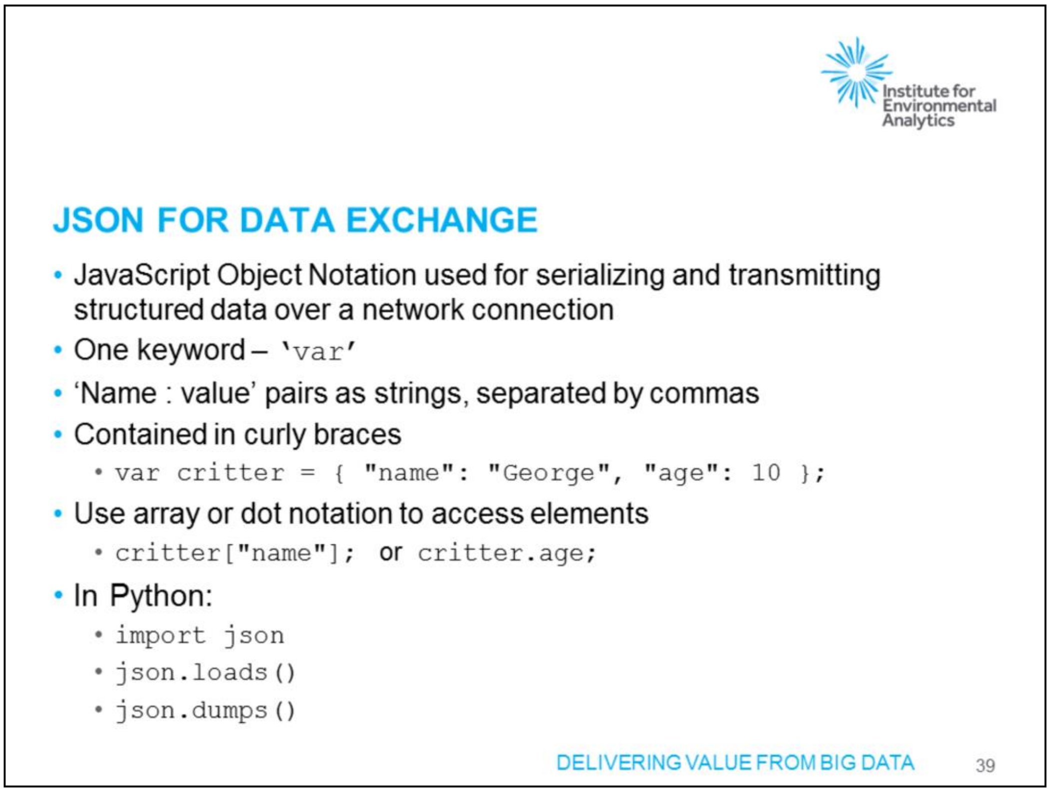
ElementTree is a much simpler XML library which would be sufficient to get you going with the principles, but we'll work with **LMXL** which purports to have the capabilities of libxml2 with the ease of use of ElementTree.

http://lxml.de/

And

http://lxml.de/tutorial.html which covers all the concepts on its first page!

Download from https://pypi.python.org/pypi/lxml


Institute for Environmental Analytics

JSON FOR DATA EXCHANGE

- JavaScript Object Notation used for serializing and transmitting structured data over a network connection
- One keyword – `'var'`
- 'Name : value' pairs as strings, separated by commas
- Contained in curly braces
  - `var critter = { "name": "George", "age": 10 };`
- Use array or dot notation to access elements
  - `critter["name"];` or `critter.age;`
- In Python:
  - `import json`
  - `json.loads()`
  - `json.dumps()`

DELIVERING VALUE FROM BIG DATA 39

JSON or JavaScript Object Notation is a lightweight is a text-based open standard designed for human-readable data interchange. The JSON format was originally specified by Douglas Crockford, and is described in RFC 4627. The official Internet media type for JSON is application/json. The JSON filename extension is **.json**.

Here's a more complex example:

var distance = { "Indianapolis" : { "Indianapolis": 0, "New York": 648, "Tokyo": 6476, "London": 4000 },

"New York" : { "Indianapolis": 648, "New York": 0, "Tokyo": 6760, "London": 3470 },

"Tokyo" : { "Indianapolis": 6476, "New York": 6760, "Tokyo": 0, "London": 5956 },

"London" : { "Indianapolis": 4000, "New York": 3470, "Tokyo": 5956, "London": 0 }, };

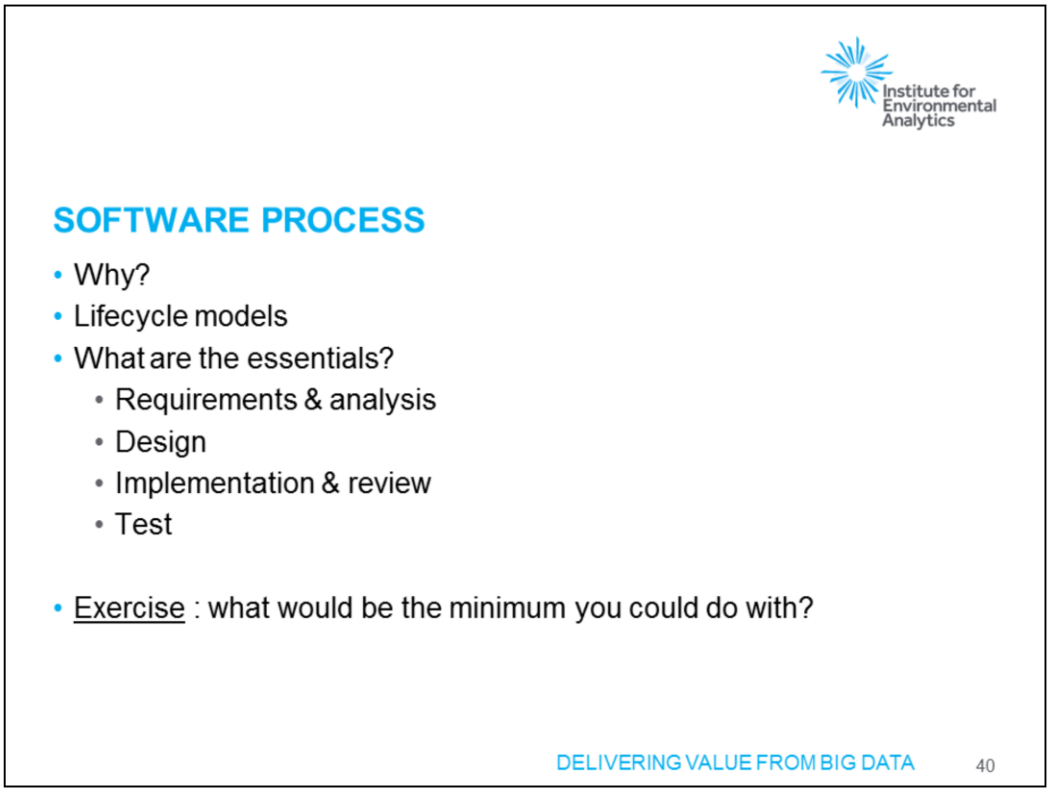
'distance' is an object and then the value for each key is also an object. It make it easy to serialize and deserialize into code.

JSON has a number of important advantages as a data format:

- **Self-documenting:** Even if you see the data structure on its own without any code around it, you can tell what it means.
- **The use of strings as indices makes the code more readable.**
- **JSON data can be stored and transported as text:** This turns out to have profound implications for web programming, especially in AJAX.
- **JSON can describe complex relationships:** The JSON format can be used to describe more complex relationships including complete databases.
- **Many languages support JSON format:** Many web languages now offer direct support for JSON. The most important of these is PHP, which is frequently used with JavaScript in AJAX applications.
- **JSON is more compact than XML:** Another data format called XML is frequently used to transmit complex data. However, JSON is more compact and less "wordy" than XML.
- **JavaScript can read JSON natively:** Some kinds of data need to be translated before they can be used. As soon as your JavaScript program has access to JSON data, it can be used directly.

http://www.dummies.com/how-to/content/how-to-use-javascript-object-notation-json-for-htm.html

https://docs.python.org/2/library/json.html


Institute for Environmental Analytics

SOFTWARE PROCESS

- Why?
- Lifecycle models
- What are the essentials?
  - Requirements & analysis
  - Design
  - Implementation & review
  - Test
- Exercise : what would be the minimum you could do with?

DELIVERING VALUE FROM BIG DATA 40

Why bother?

There are lots of reasons why most software is not done on-the-hoof…

Estimation

Quality & Maturity

Change control

Budget

Resources

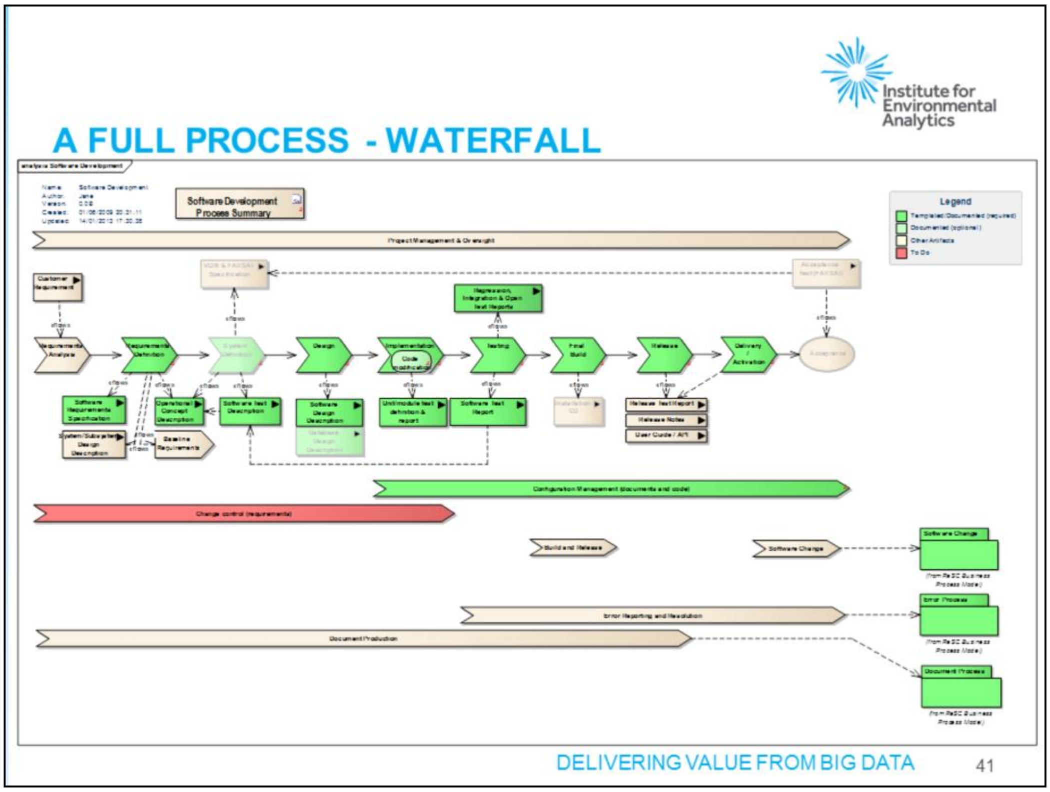
Delivery milestones

Customer acceptance

It's fine to have little or no process when you're hacking away at your own code which no one else will ever see, but usually software is developed in teams and everyone has to know the dependencies, what's expected of each component, the integration plan, the testing strategy and so on.

On big projects, there is a lot of documentation and the code itself is actually quite a small part of it. Safety or mission critical software almost drowns in paperwork, but for good reason.

The following slides will give you an idea of the complexity, and you may want to think a little more about some aspects, but I’m not suggesting for a moment that this is the way to go for research software!



A FULL PROCESS - WATERFALL

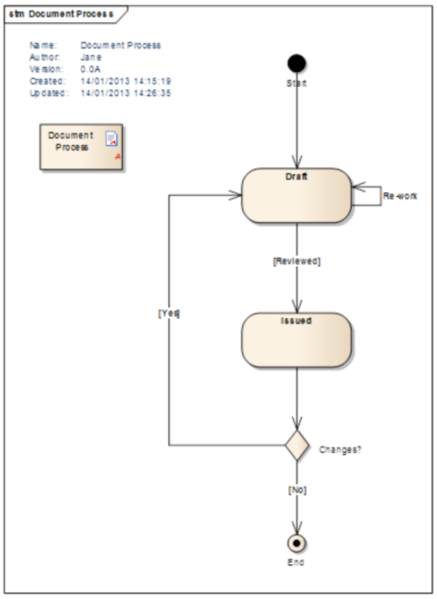
Institute for Environmental Analytics
analysis Software Development
Name: Software Development
Author: Jane
Version: 0.08
Created: 01/06/2009 20:31:11
Updated: 14/01/2013 17:30:38
Software Development Process Summary
Legend
Templated/Documented (required)
Documented (optional)
Other Artifacts
To Do
Project Management & Oversight
Customer Requirement
Requirements Analysis
Requirements Definition
System Definition
Design
Implementation
Code modification
Testing
Final Build
Release
Delivery / Activation
Acceptance
«flows»
Regression, Integration & Open Test Reports
Acceptance Test
Software Requirements Specification
Operational Concept Description
Software Test Description
Software Design Description
Database Design Description
Unit/module test definition & report
Software Test Report
Release Test Report
Release Notes
User Guide / API
System/Subsystem Design Description
Baseline Requirements
Configuration Management (documents and code)
Change control (requirements)
Build and Release
Software Change
Error Reporting and Resolution
Error Process
Document Production
Document Process
(from ReSC Business Process Model)

DELIVERING VALUE FROM BIG DATA 41

The software industry's nerd centre is the IEEE: Institute of Electrical and Electronic Engineers (https://www.ieee.org/index.html). It has many many standards and templates for how to organise and manage software projects. This diagram does use IEEE standard names for things and follows one of their process templates.

If you find yourself getting more involved in software later in your careers, it's worth a look, then at least you'll know some of the terms and be able to impress at interviews!

DOCUMENT CONTROL

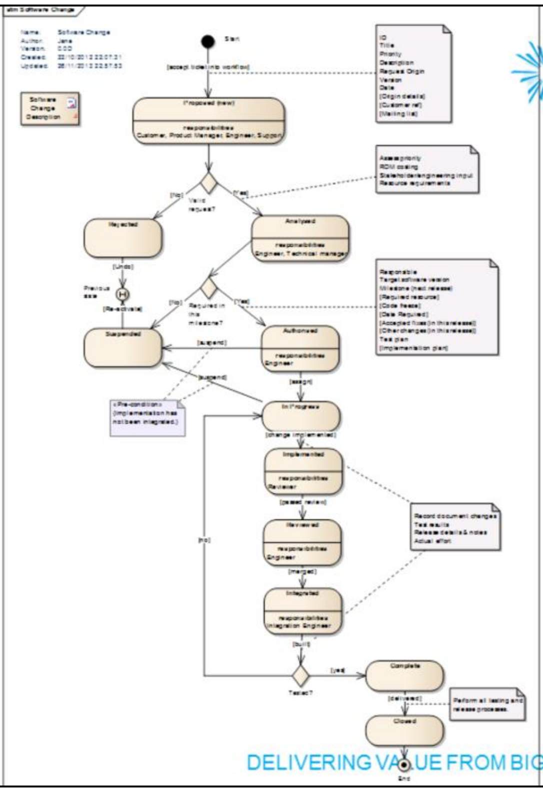


Institute for Environmental Analytics

Even documents can have a process!

Here's a very simple one which ensures everyone working on the same document uses a standard naming convention and it's clear when a document is in draft and when it's been reviewed, approved and issued. These considerations don't arise when you're working on your own, or do they?

SOFTWARE CHANGE



Institute for Environmental Analytics

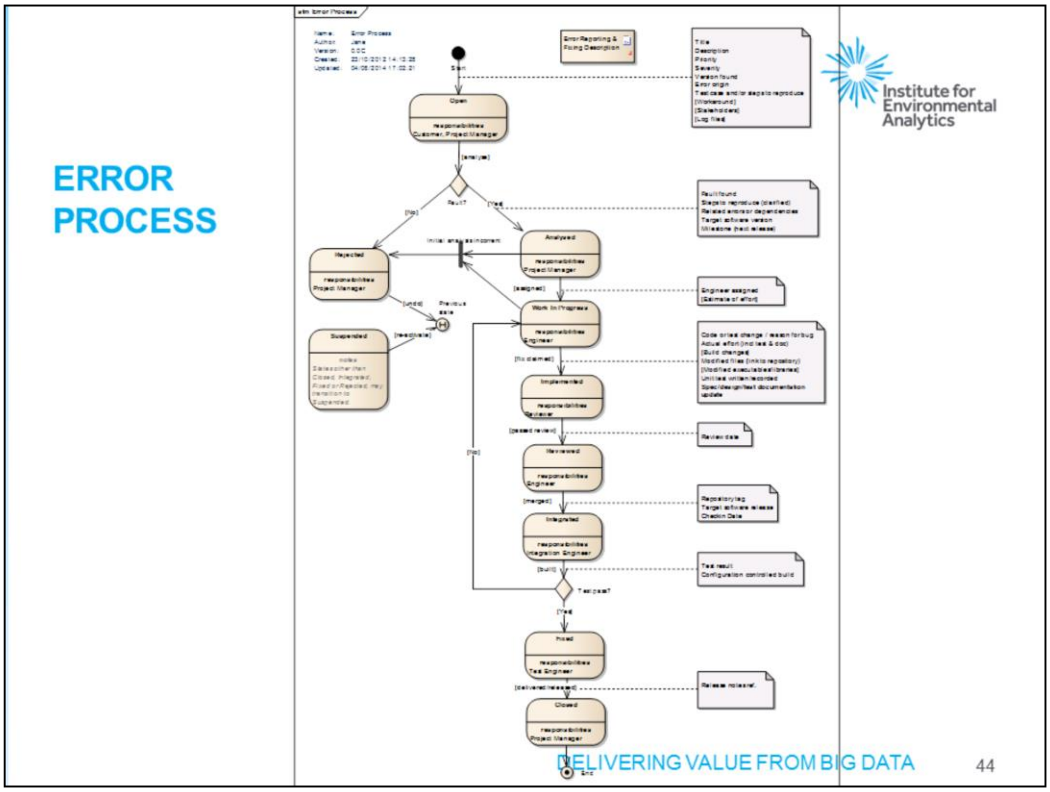
DELIVERING VALUE FROM BIG DATA

43

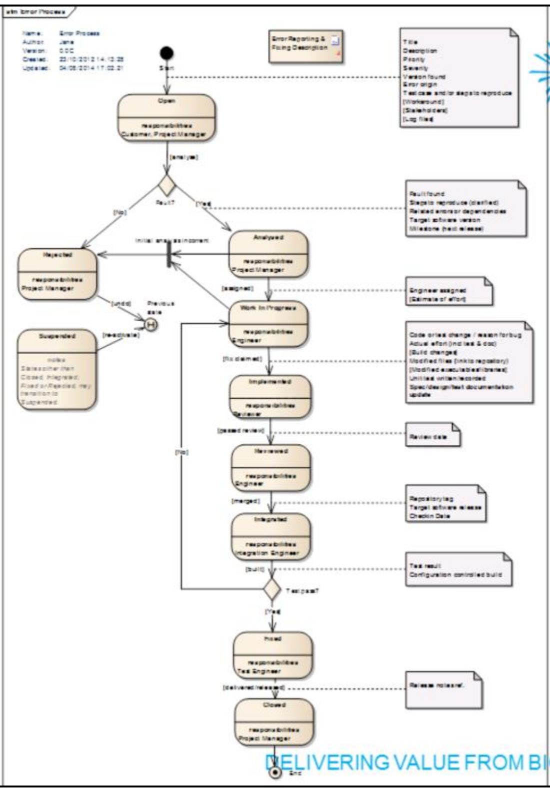
And what if you have a request to fulfil a requirement that didn't crop up to begin with? Or to extend the software to do something new? There's a way to handle that too.

This is pretty complex but the key points are:

1. A change should be thought about and its impact assessed before diving in, late big changes are the most disruptive.
2. You have to think about how it affects any design decisions, and how you are going to test it.
3. You must make sure you know how to check it hasn't broken anything else.
4. You should always do it on a new code branch and only accept it into the trunk when all tests have passed.



ERROR PROCESS



Institute for Environmental Analytics

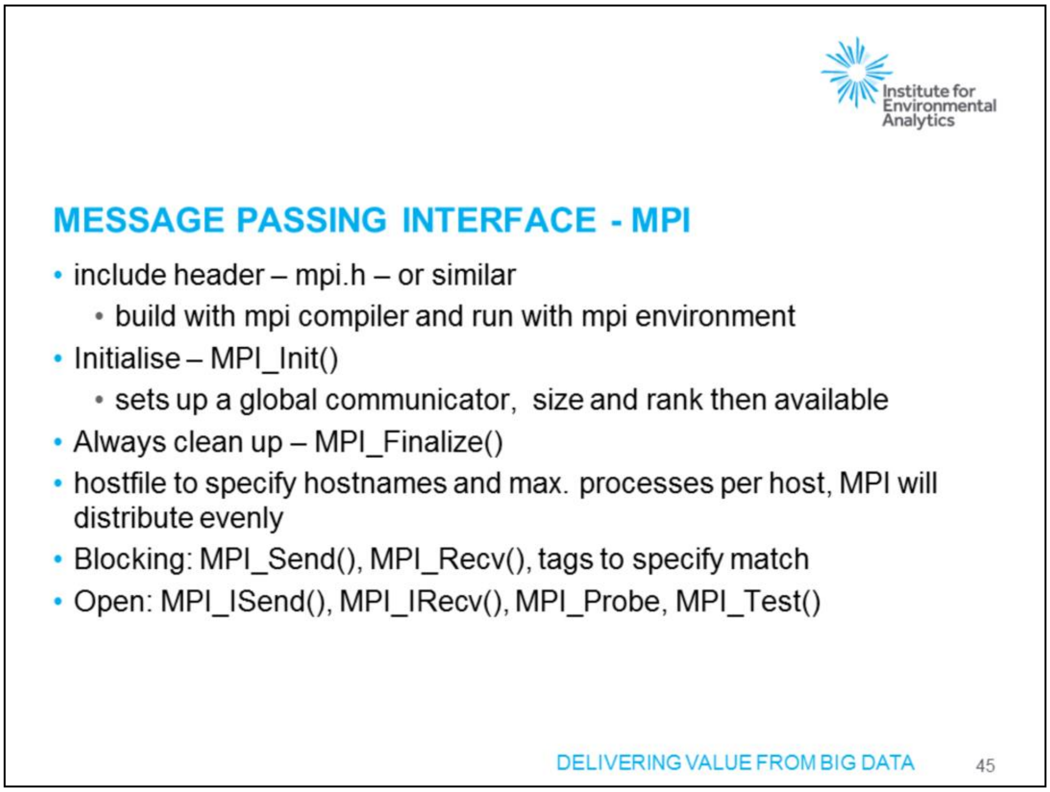
DELIVERING VALUE FROM BIG DATA 44

And here is an example of handling errors and their resolution. This looks a bit over the top too… and for most small projects it is of course.

However, there are key points again:

1. It must be analysed first…
    a. is it even a bug? Do you need to clarify what the software is meant to do so that the behaviour isn't classified as a bug?
    b. quick fix or big change? Will fixing it have ramifications through the software? How much testing will you need to do?
2. Test(s) must be written to check it
3. It should be double checked by a peer if possible, especially if it's a real nasty error.
4. You have to make a judgement on how much original testing will be invalidated by the change.

There is a free system for keeping an eye on requirements, errors and changes called Trac (http://trac.edgewall.org/). You can define processes and 'gates', collect information and allow users to update fields as they are required. If you would like some more information on this, please mail me j.p.lewis@reading.ac.uk.


Institute for Environmental Analytics

MESSAGE PASSING INTERFACE - MPI

- include header – mpi.h – or similar
  - build with mpi compiler and run with mpi environment
- Initialise – MPI_Init()
  - sets up a global communicator, size and rank then available
- Always clean up – MPI_Finalize()
- hostfile to specify hostnames and max. processes per host, MPI will distribute evenly
- Blocking: MPI_Send(), MPI_Recv(), tags to specify match
- Open: MPI_ISend(), MPI_IRecv(), MPI_Probe, MPI_Test()

DELIVERING VALUE FROM BIG DATA 45

MPI is a framework for carrying out parallel computing. It is NOT the same as forking a process or sharing memory between processes; instead, you create communication channels between instances of the same program running in parallel. They all have to be started in the MPI environment, and call MPI_Init( ) before doing anything else.

At this point, each and every process (i.e. duplicate of your program, or duplicates of several) has access to a global communication channel, knowledge of its own rank on that channel, and the total number of processes running in the MPI environment.

MPI_COMM_WORLD : reserved identifier for the initial communicator

MPI_Comm_rank( ) :returns unique identifier for the process

MPI_Comm_size( ) :tells you how many processes there are in the MPI environment

The best way to get to grips with the detail is to thoroughly read one of the many excellent tutorials: http://mpitutorial.com/beginner-mpi-tutorial/ is particularly helpful.

All sends and receives need to be matched, and you can use blocking messages when you know you'll be swapping information, non-blocking when you're not sure and may want a timeout coupled with a test.

There are other ways to distribute and collect information: MPI_Gather( ) and MPI_Scatter( ) for example. And you can block on everything finishing at a particular stage with MPI_Wait( ) and MPI_Barrier( ).