

# Module OOP11

## Methodology Comparison

©B. Henderson-Sellers, 2000-2008

OOP11.1

- Some of the groundbreaking first and second generation methods in OO (Booch, OMT, RDD, C/Y, OOSE, BON, Shlaer/Mellor)
- Other third generation approaches
  - RUP
  - Agile (a.k.a. Lightweight): XP, Crystal
  - Others: Catalysis

©B. Henderson-Sellers, 2000-2008

OOP11.2

## **"Booch" methodology**

**Chief Scientist with Rational**

**Booch (1991), Object-Oriented  
Design with Applications,  
Addison-Wesley**

**(1994), second edition  
Object-Oriented Analysis and  
Design with Applications**

**Developed from Ada background**

**Now oriented to C++ although  
language-independent**

*©B. Henderson-Sellers, 2000-2008*

*00P11.3*

**The following relationships  
are supported:**

**1) Containing**

Aggregation, is-composed-of.  
Weak semantics.

**2) Association**

Bidirectional

**3) "Uses-a"**

Client-server relationship -- refinement of  
an association i.e. one class uses another.

**4) Metaclass**

Relationship between a class  
and the metaclass

**5) Instantiate**

Creates-a

*©B. Henderson-Sellers, 2000-2008*

*00P11.4*

## 6) Inheritance

Single  
Multiple -- to a lesser extent  
e.g. name clash problems  
Repeated -- permitted but  
not really supported

### Also supported:

#### Visibility

Visibility of classes within  
categories

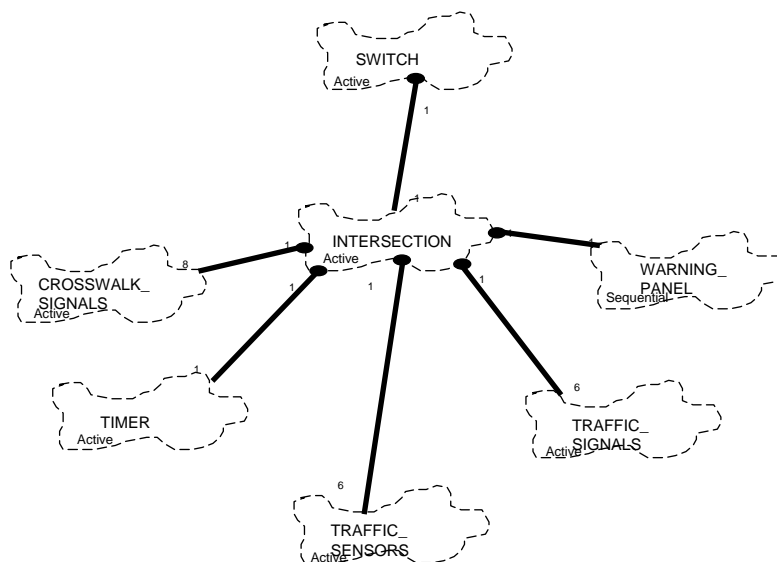
#### Cardinality

Relationship cardinalities

#### Communication mechanisms

Simple message passing;  
synchronous; balking;  
timeout; asynchronous

## Class Diagram



# Object Modeling Technique (OMT)

Developed at GE Advanced Concepts  
Center (later part of Martin  
Marietta)

Rumbaugh *et al.* (1991) -- Object-  
Oriented Modeling and Design,  
Prentice Hall  
(2nd edition never prepared)

Evolutionary approach (carefully  
considered extension of ER  
modelling)

Language independent

©B. Henderson-Sellers, 2000-2008

00P11.7

**The following relationships  
are supported:**

- 1) **Aggregation**  
Well-defined semantics (but now seen to be inappropriate)
- 2) **Association**  
Similar to relationships in ER.  
Ternary associations supported.  
Link attributes and qualifiers.
- 3) **Generalization**  
Single  
Multiple -- to a lesser extent  
Repeated -- permitted but  
not really supported  
Overlapping and disjoint
- 4) **Instantiate**  
(in DFDs)

©B. Henderson-Sellers, 2000-2008

00P11.8

**Model includes associations as an equal to classes making the model very similar to an ER approach. This makes the approach somewhat less pure in OO terms.**

**Based on an Extended ER model plus dynamic representation**

**Quite complex/expressive**

**Language-independent**

**Data-oriented**

## **Representation**

**The OMT method is expressed using the following 3 models:**

### **A) Graphical Representation**

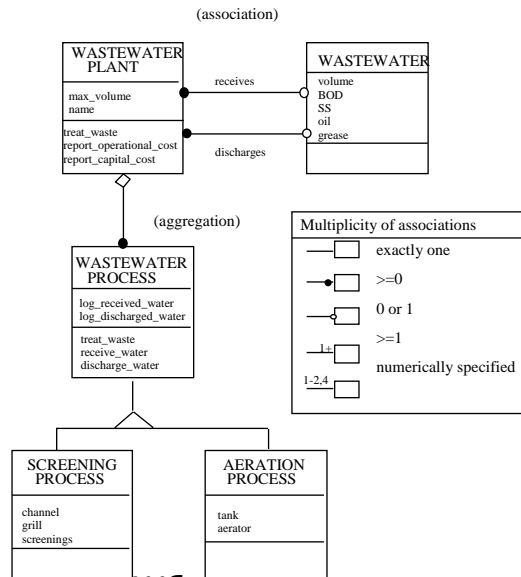
Object Model (structure)  
Dynamic Model (behaviour)  
Event Trace  
State Diagram  
Functional Model (behaviour)  
-- much criticized by  
Hayes and Coleman (1991)

### **B) Textual Representation**

Class Specification  
Scenarios

## Rumbaugh et al. (1991) Notation

### 1) Object Model



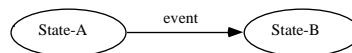
©B. Henderson-Sellers, 2000-2008

00P11.11

## Rumbaugh et al. (1991) notation - continued

### 2) Dynamic Model

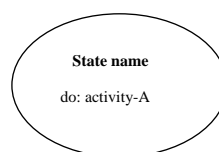
Event causing transition between states



Initial and final states



Activities whilst in a state

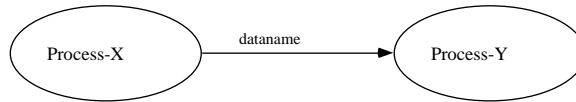


©B. Henderson-Sellers, 2000-2008

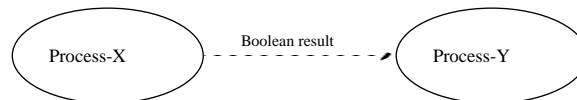
00P11.12

### 3) Functional Model

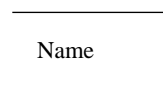
Data flow between processes



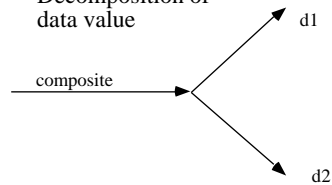
Control flow



Data store



Decomposition of data value



©B. Henderson-Sellers, 2000-2008

00P11.13

## **Responsibility Driven Design (RDD)**

**Developed at Tectronix**

**Wirfs-Brock et al. (1990) --  
Designing Object-Oriented  
Software, Prentice Hall**

**Smalltalk-oriented**

**One of the earliest approaches**

©B. Henderson-Sellers, 2000-2008

00P11.14

**The following relationships  
are supported:**

**1) Collaboration**

Similar to "uses" --  
dynamic relationship

**2) Inheritance**

Single

Multiple -- to a lesser extent  
e.g. name clash problems

Repeated -- permitted but  
not really supported

**The following communication  
mechanism is supported**

Message Connection:

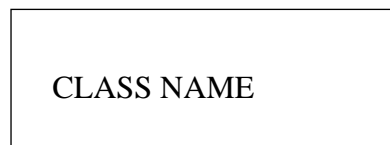
Simple message passing

©B. Henderson-Sellers, 2000-2008

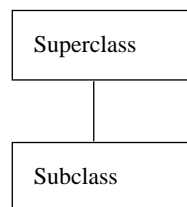
00P11.15

**Wirfs-Brock et al. (1990) Notation**

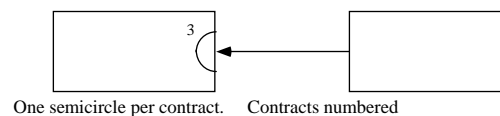
Class Icon



Inheritance



Collaborations



©B. Henderson-Sellers, 2000-2008

00P11.16

## **Coad and Yourdon**

**Peter Coad and Ed Yourdon -- independent consultants**

**Coad and Yourdon (1990/1) --  
Object-Oriented Analysis, Prentice Hall**

**Coad and Yourdon (1991) --  
Object-Oriented Design, Prentice Hall**

**Coad and Nicola (1993) --  
Object-Oriented Programming, Prentice Hall**

**Coad et al. (1995) -- Object Models: Strategies, Patterns  
and Applications**

**Information engineering background**

**Language independent**

**Aims for simplicity**

*©B. Henderson-Sellers, 2000-2008*

*00P11.17*

### **The following relationships are supported:**

**1) Whole-part**

Aggregation, is-composed-of.  
Relatively weak semantics.

**2) Instance connection**

Association, is-related-to.  
Structural relationship  
between objects.

**3) Gen-Spec -- Inheritance**

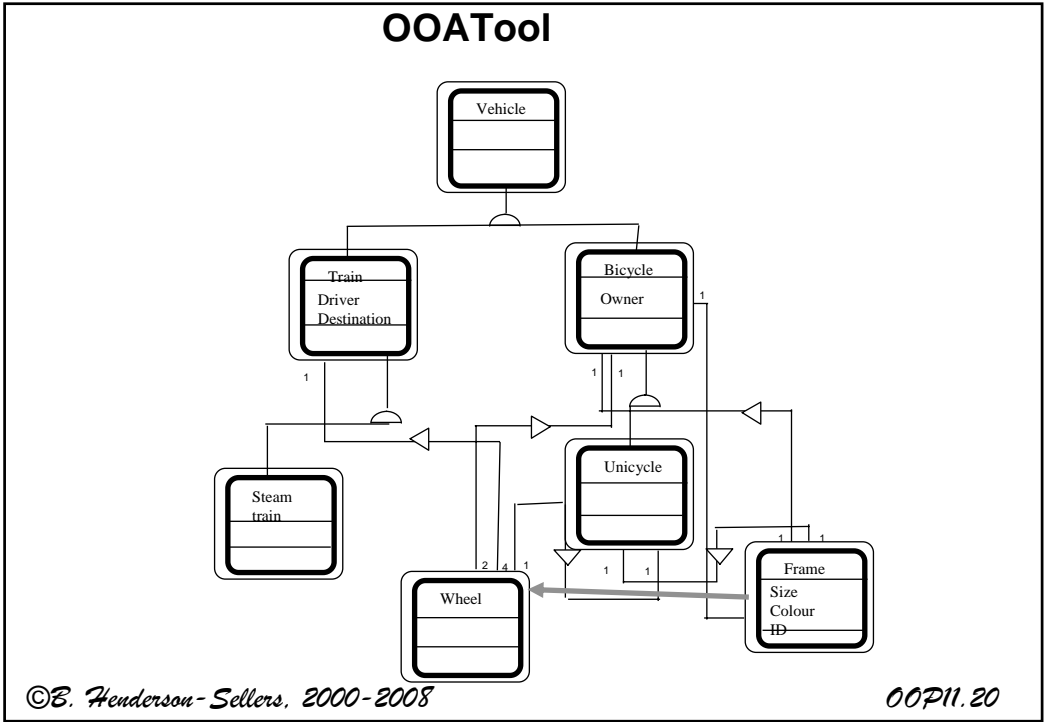
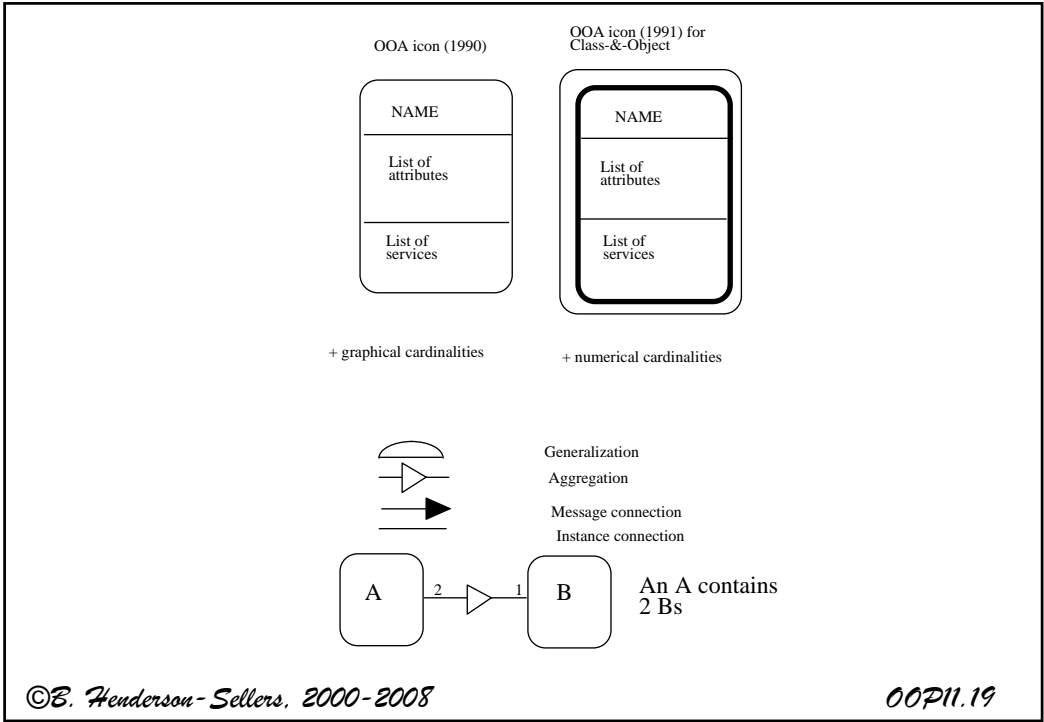
Single  
Multiple -- to a lesser extent  
e.g. name clash problems  
Repeated -- permitted but  
not really supported.

**4) Cardinality**

Relationship cardinalities

*©B. Henderson-Sellers, 2000-2008*

*00P11.18*



## **OOSE/Objectory**

**OOSE (Ivar Jacobson et al., 1992)**

**Objectory Level 3 proprietary methodology from Objective Systems SF AB (Sweden)**

**Long history, but only recently public domain (OOSE)**

**Developed for telecommunications industry, initially in Sweden**

**Introduced "use cases" to OO world**

**The following relationships are supported:**

**1) Consists of**

Aggregation

**2) Acquaintance = "Uses-a"**

Unidirectional association.

**3) Communication**

Sends a stimulus to another object

## 4) Inheritance

### Also supported:

Subsystems

Attribute and attribute  
types treated differently

### Cardinality

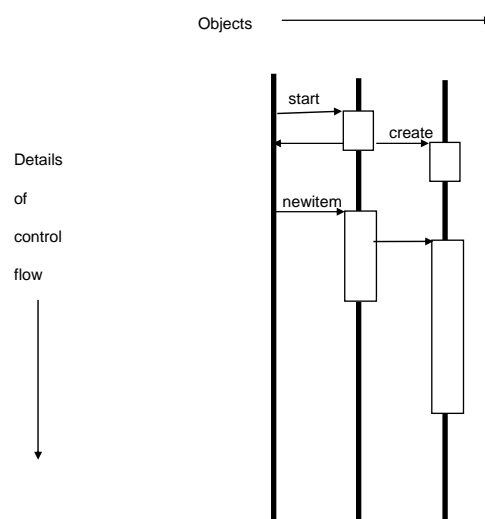
### Communication mechanisms

Synchronous; asynchronous  
events

©B. Henderson-Sellers, 2000-2008

00P11.23

## Interaction Diagram



©B. Henderson-Sellers, 2000-2008

00P11.24

## Shlaer and Mellor

Realtime focus.

The major translational (cf. elaborational)  
method

Links to BridgePoint tool

## BON (Walden and Nerson)

Main focus is seamlessness and  
reversibility

Static and dynamic diagrams similar  
and no state machine

Contains formal grammar for assertions

Links to Eiffel

## **Other 1st/2nd generation methodologies /notations**

SOMA (Graham)  
Fusion (Coleman et al.)  
ADM3/4 (Firesmith)  
Berard  
OBA  
Martin and Odell  
OOSD/StP (Wasserman et al.)  
OOram

## **Current (3rd generation) OO/CBD processes**

- RUP
- Catalysis
- Agile methods (e.g. XP, Crystal)

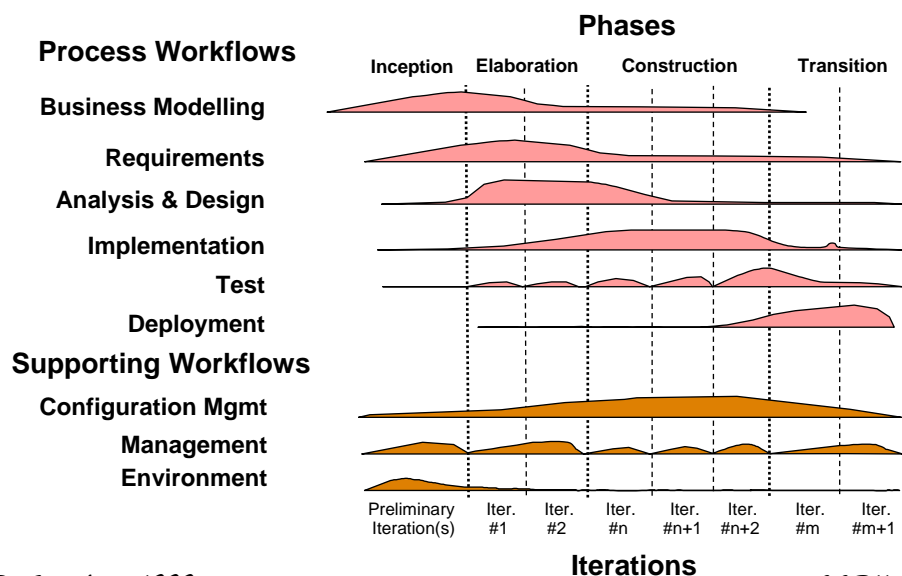
## a) RUP

- RUP was initially intended primarily to be a one-size-fits-all with minimal tailoring. Tailorability (which is being enhanced) starts from a full and comprehensive process, then modifies mostly by subtraction and modification.
- RUP strongly advocates a use-case driven approach

©B. Henderson-Sellers, 2000-2008

00P11.29

## Workflows in RUP (typical diagram)

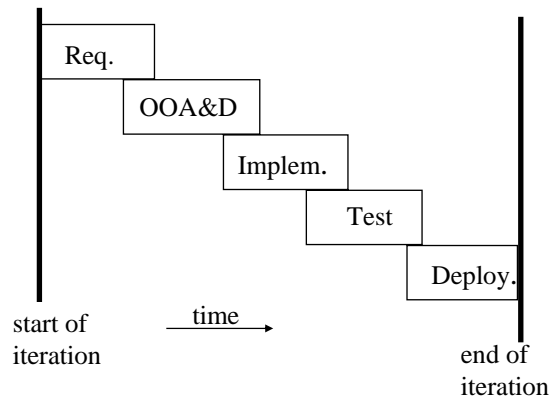


© P. Kruchten, 1999

00P11.30

- Each “iteration”/increment emphasizes the production of working code
- Each Iteration Workflow is a mini-waterfall

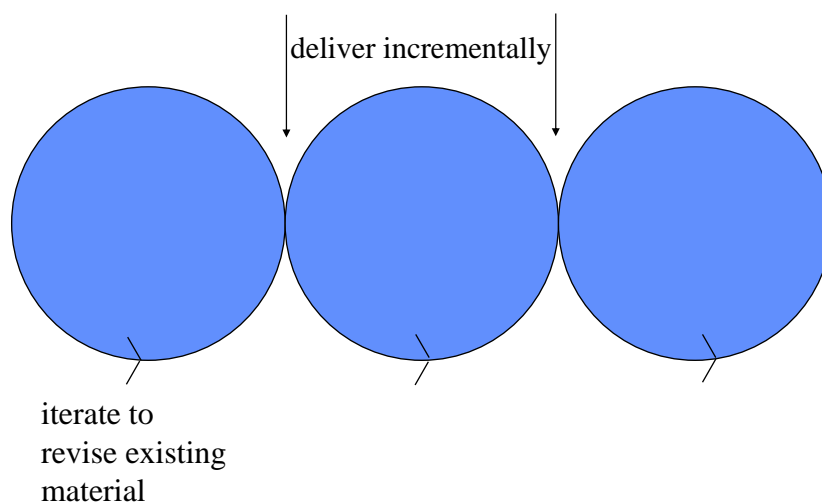
e.g. Elaboration iteration



©B. Henderson-Sellers, 2000-2008

00P11.31

*The word iteration in RUP actually means both iteration and increment*

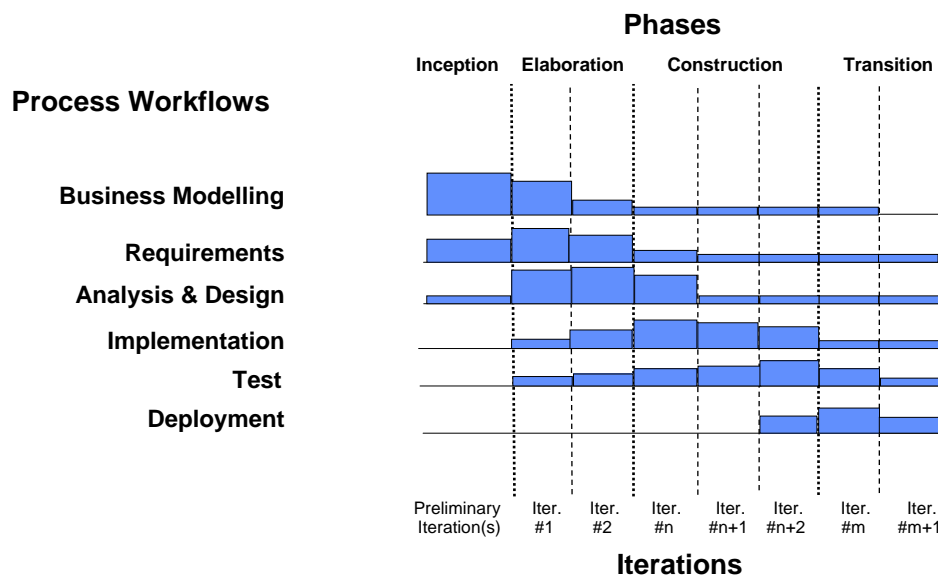


©B. Henderson-Sellers, 2000-2008

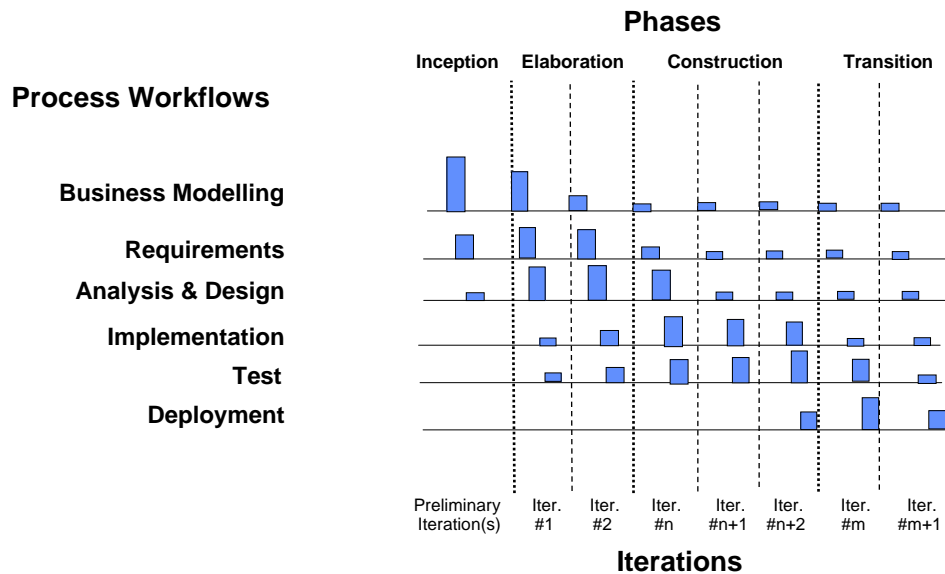
00P11.32

- A problem with the “hump” curve is that time flows both horizontally and vertically.
- In reality, the curves are discrete not continuous.

### The unit of time here is one “iteration”



## Adding the embedded waterfall



©B. Henderson-Sellers, 2000-2008

00P11.35

## c) Comparison

- OPEN is in the public domain; RUP is a proprietary product
- OPEN is highly flexible; RUP less so, especially at the large scale (at least before the OMG (2001) metamodel was incorporated) - tailoring is supported in both at the more detailed scale

©B. Henderson-Sellers, 2000-2008

00P11.36

## Further comparison

- RUP emphasizes use case driven; OPEN also supports well responsibility-driven, document-driven and many others
- RUP's and OPEN's models of the process lifecycle are different
  - RUP is linear: a waterfall embedded within a waterfall; OPEN embodies an IIP contract-driven lifecycle and can support protocols for parallel activities

## Some other differences

- No deontic scores in RUP
- RUP elaborational. Transitional not supported. OPEN neutral on this.
- OPEN uses standard PM terminology; RUP uses arguably neologistic terms

# Catalysis

- Originally an application of rigorous methods to OMG
- Focus on collaborations; component interface definitions; high integrity design - precise abstract specification and unambiguous traceability
- But *not* full lifecycle; really design and development

- Objects + actions (events, tasks, jobs, changes of state, interactions etc.) - as peers
- Postconditions help to add precision
- Three underpinning principles
  - abstraction
  - precision
  - pluggable parts

## **Process in Catalysis:**

- Nonlinear, iterative and parallel
- Rigour, QA and testing are continuous
- Emphasis on architecture
- Flexible (many routes through process)
- Uses documented process patterns

## **Agile (or lightweight) methods**

have recently been created since many felt heavyweight methods have too much overhead (resource-hungry)

They are

- more code-oriented
- adaptive rather than predictive
- people-oriented

## Adaptive processes useful for

- uncertain or volatile requirements
- responsible and motivated developers
- customer who understands & gets involved

## cf. Predictive processes useful for

- team size > 50
- fixed price/fixed scope contracts

## Examples are

- XP (eXtreme Programming)
- Crystal

Also

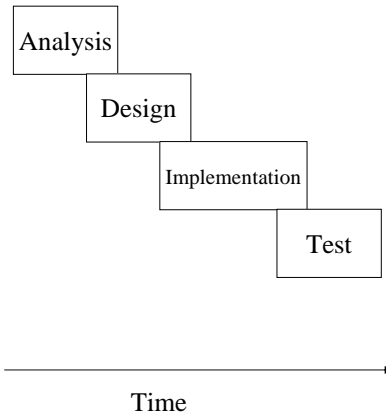
- Adaptive Systems (Highsmith), SCRUM, FDD (Feature Driven Design/Coad)

## XP's characteristics

- Smalltalk community origins
- 4 values: communication, feedback, simplicity, courage
- Emphasis on testing
- Customer involvement (as team member)
- Focus on current iteration with no design for anticipated future needs (design as you go)

- Evolutionary design process, reliant on lots of refactoring
- Pair programming
- Iterations of 1-3 weeks
- 40 hour weeks
- Code focus, eschews UML diagrams
- Flexibility requirement yet process relatively inflexible - challenging to adhere to

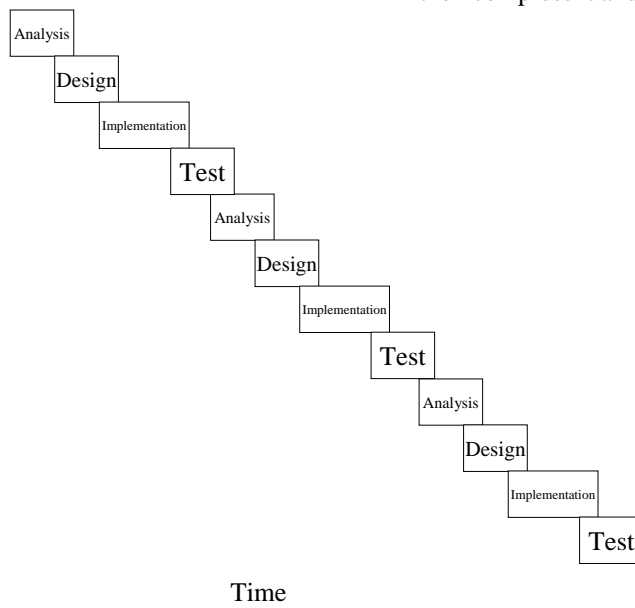
The idea behind XP's lifecycle is to take the waterfall and ...



©B. Henderson-Sellers, 2000-2008

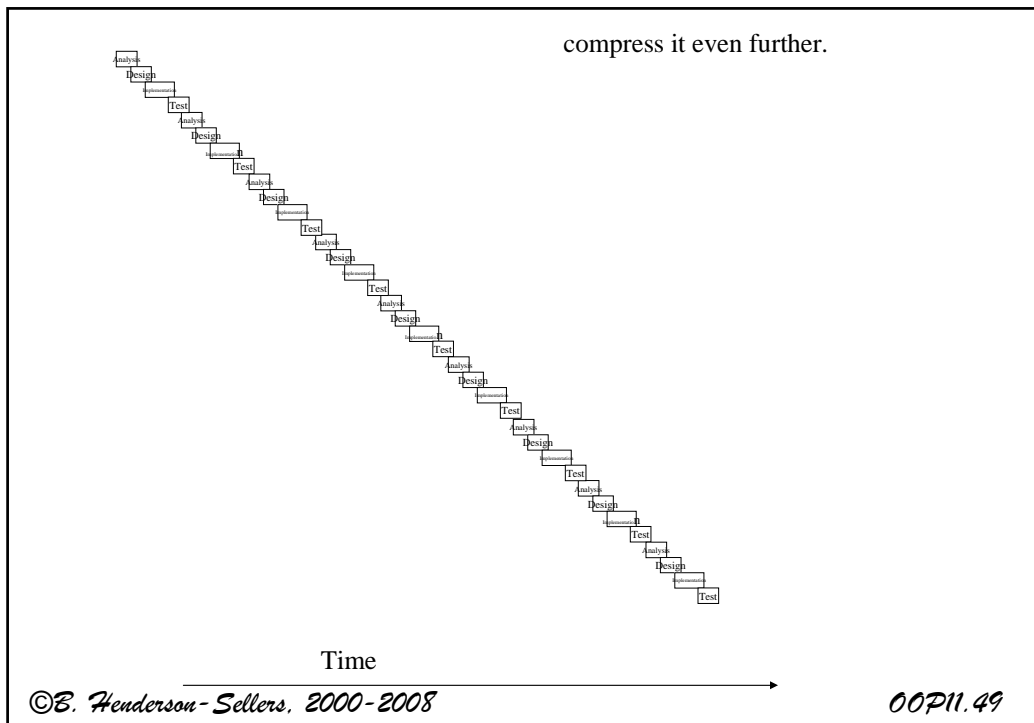
00P11.47

then compress it and then ...



©B. Henderson-Sellers, 2000-2008

00P11.48



## Crystal characteristics

- A family of methodologies
- People focussed
- Influenced by lot of experience
- 2 axes: number of people on project + consequence of errors
- The *least* disciplined process that could still succeed

- In contrast to XP's highly disciplined process, Crystal acknowledges people find it hard to follow a disciplined process
- Argued to be easier to follow than XP, although less productive

## **Adaptive Systems approaches (esp. Highsmith)**

- advice on why adaptive development important

### **SCRUM**

- 30 day iterations (“sprints”)
- daily “scrum” (15 minute team meetings of managers and developers)
- focus on iterative planning and tracking

## **FDD**

- Short iterations, each delivering tangible functionality
- Iterations of 2 weeks
- 2 kinds of developers: class owners and chief programmers

- Five processes
  - develop overall model
  - build features list
  - plan by feature(within each iteration)
  - Design by feature
  - Build by feature

## Summary

- Some of the groundbreaking first and second generation methods in OO (Booch, OMT, RDD, C/Y, OOSE, BON)
- Other third generation approaches
  - RUP
  - Lightweight/Agile: XP, Crystal
  - Others: Catalysis

## Source of material

- Fowler, M., The new methodology, [www.martinfowler.com/articles/newMethodology.html](http://www.martinfowler.com/articles/newMethodology.html)
- Beck, K., 2000, *Extreme Programming Explained*, Addison-Wesley
- Henderson-Sellers, B. and Edwards, J.M., 1994, *BOOKTWO of Object-Oriented Knowledge. The Working Object*, Prentice Hall