Learning OOABL - The Human Factor

PROGRESS EXCHANGE 2013

DISCOVER. DEVELOP. DELIVER.

Timothy D. Kuehn

Senior OpenEdge Consultant

TDK Consulting Services Inc

Email: timk@tdkcs.ca or tim.kuehn@gmail.com
Ph: 519-576-8100
Skype: timothy.kuehn

About Tim and TDK Consulting Services

- Joined the Progress community version 8.2
- Primarily involved in functionality refresh of long-term Legacy Applications
- Taught ”Intro to OO” material at all PUG Challenge Americas conferences
- Presented on Multi Tenancy, ABL Performance Issues, and related topics
- Founding Chair of the PUG Challenge Americas Program Committee
- President of the Ontario PUG since 2007

Session Overview

- Why Learn OOABL?
- OO / Procedural Programming Models
- What's Involved in Learning OOABL
- "Things You Can Do Now" Code Samples
- Reading List
- Questions

Session Overview

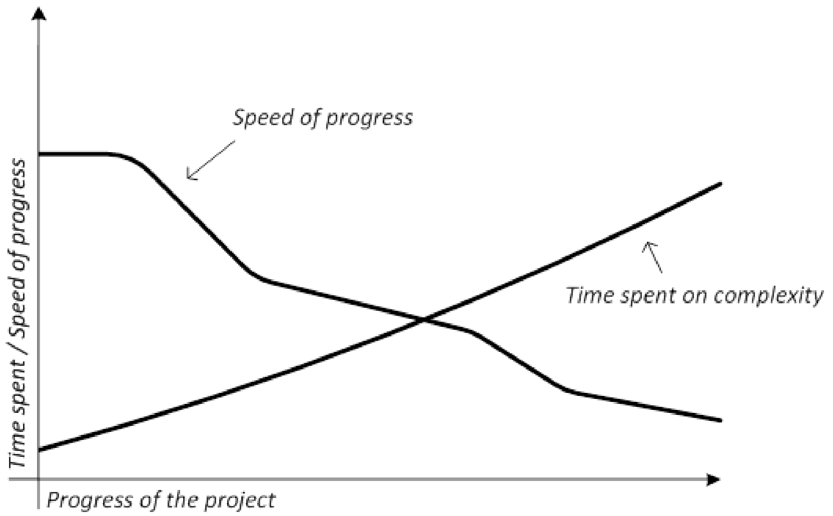
- <u>Why Learn OOABL?</u>
- OO / Procedural Programming Models
- What's Involved in Learning OOABL
- "Things You Can Do Now" Code Samples
- Reading List
- Questions

Why Learn OOABL?

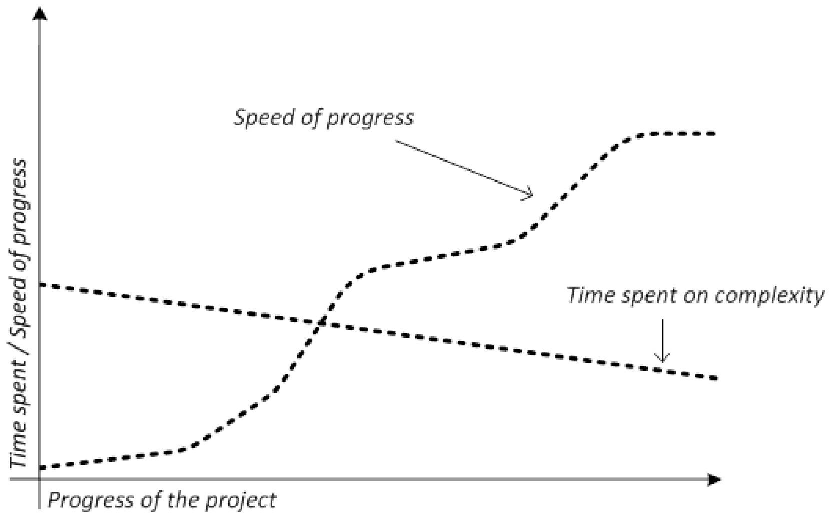
1. Helps manage complexity while increasing productivity
2. Improves code re-use, helps reduce “Copy and Paste” programming
3. Available literature on OO-related practices and technology
4. Schools train to the OO model, making it easier to find OO developers
5. Can “mix and match” OOABL strengths with Procedural strengths
6. PSC API’s, tools, etc will include more OOABL-oriented functionality going forward

Productivity Curves

Procedural Programming



Object Oriented Programming

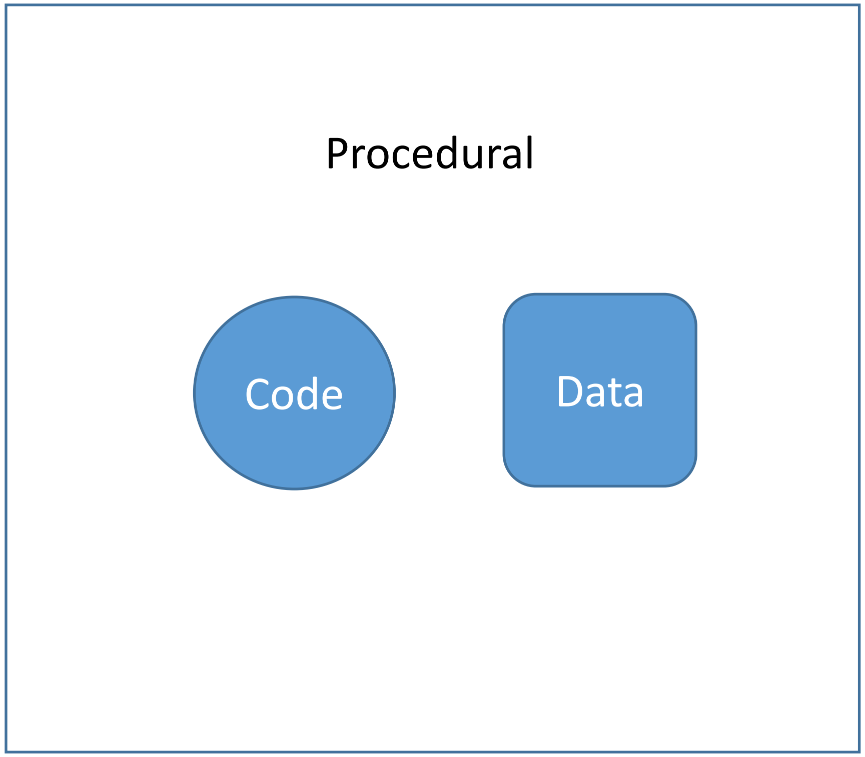


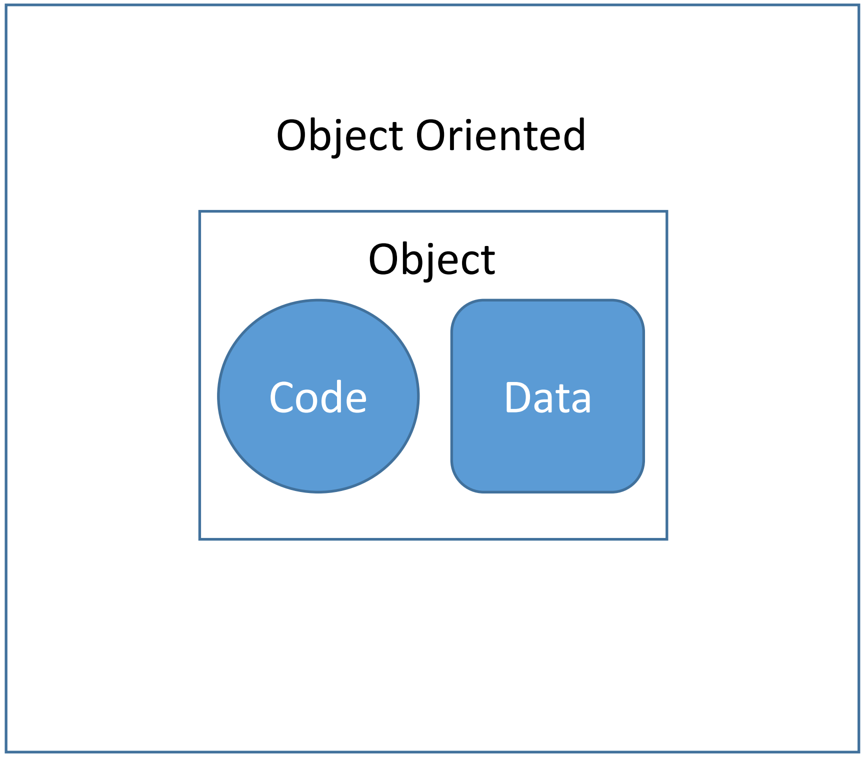
Refactoring and Design Patterns by Peter Kaptein
http://patterns.instantinterfaces.nl/current/Refactoring-and-Design-Patterns-OOP.html#OOP-PRDC

Session Overview

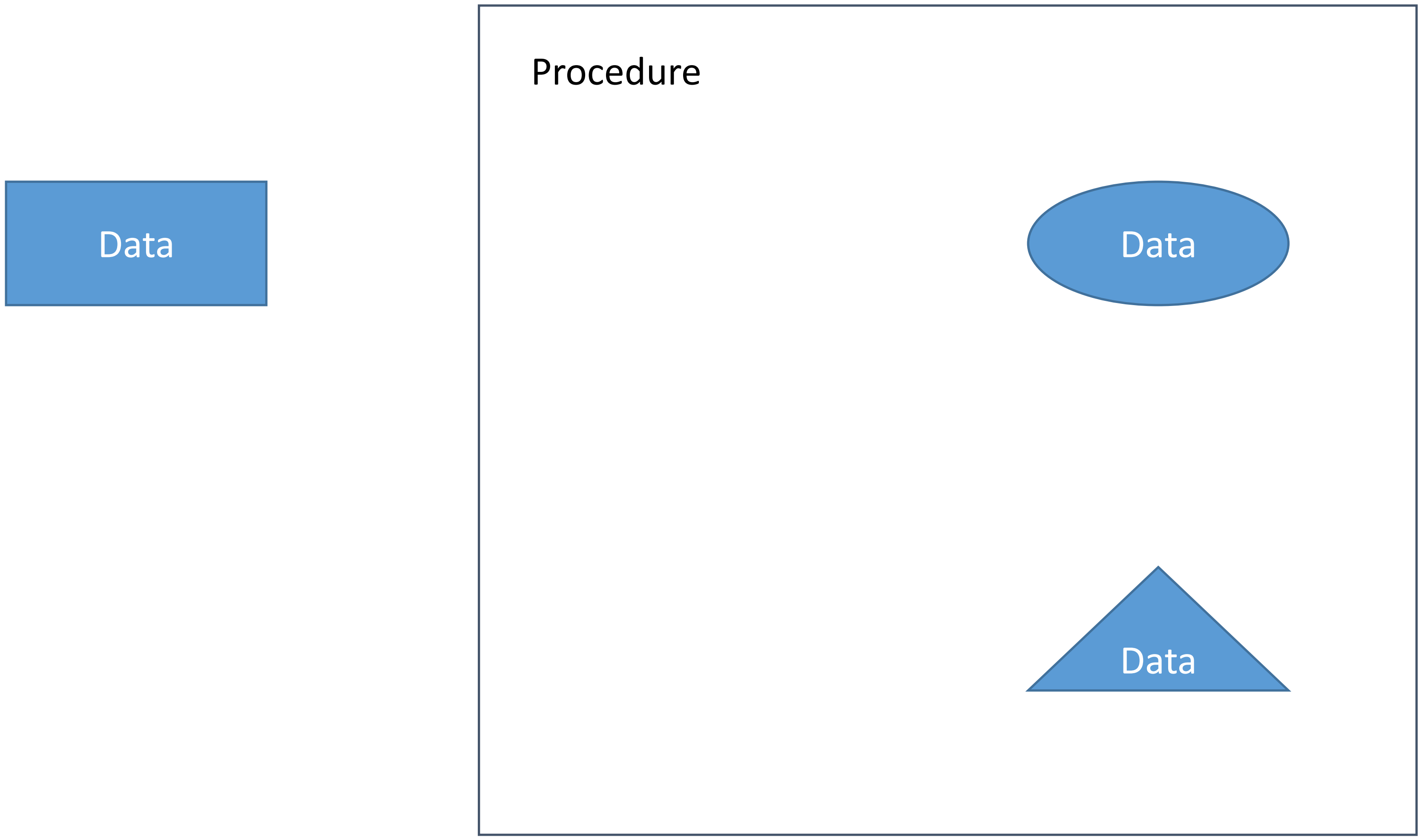
- Why Learn OOABL?
- <u>OO / Procedural Programming Models</u>
- What's Involved in Learning OOABL
- "Things You Can Do Now" Code Samples
- Reading List
- Questions

OOABL vs Procedural Concepts – OOABL Model

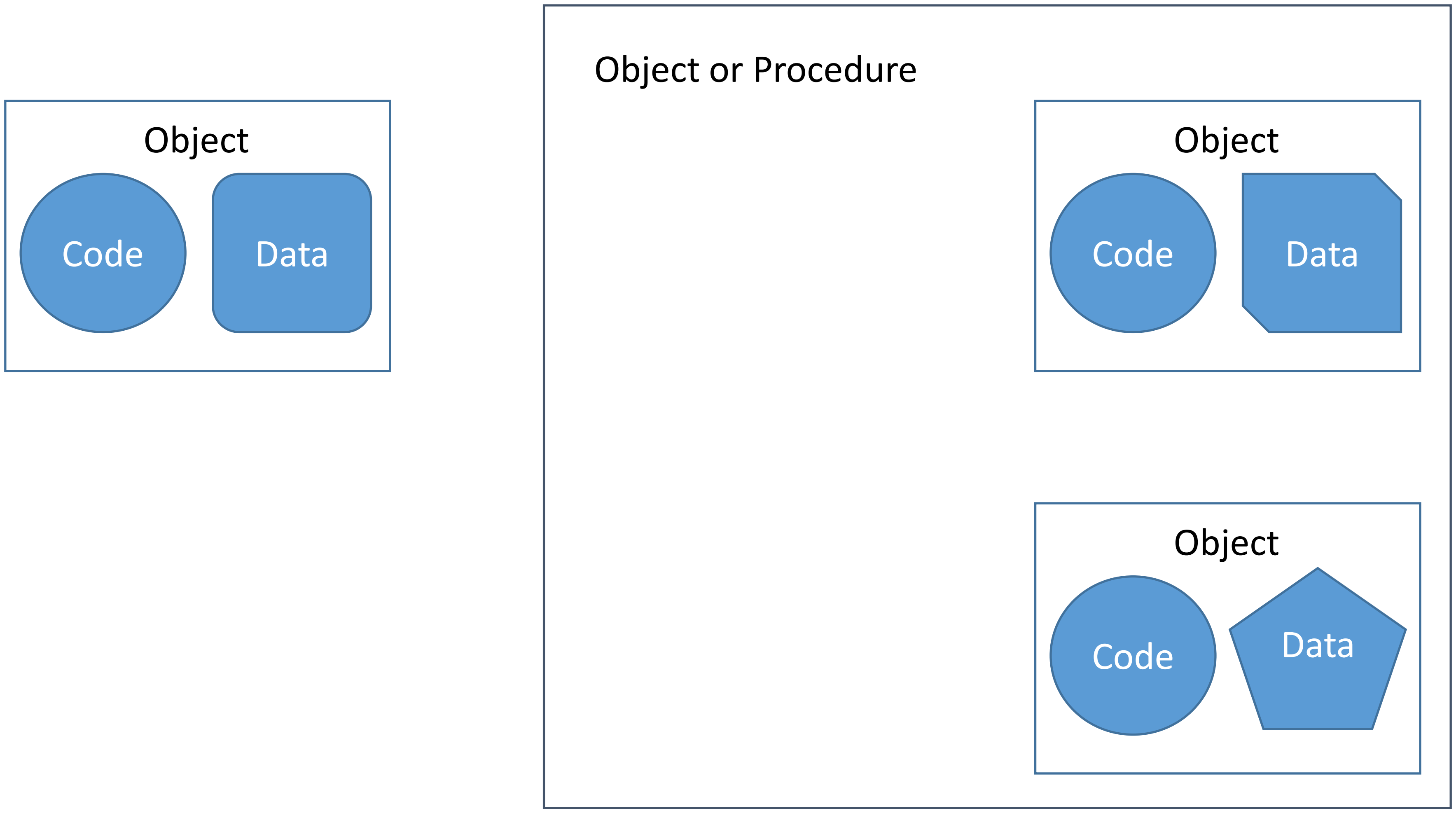




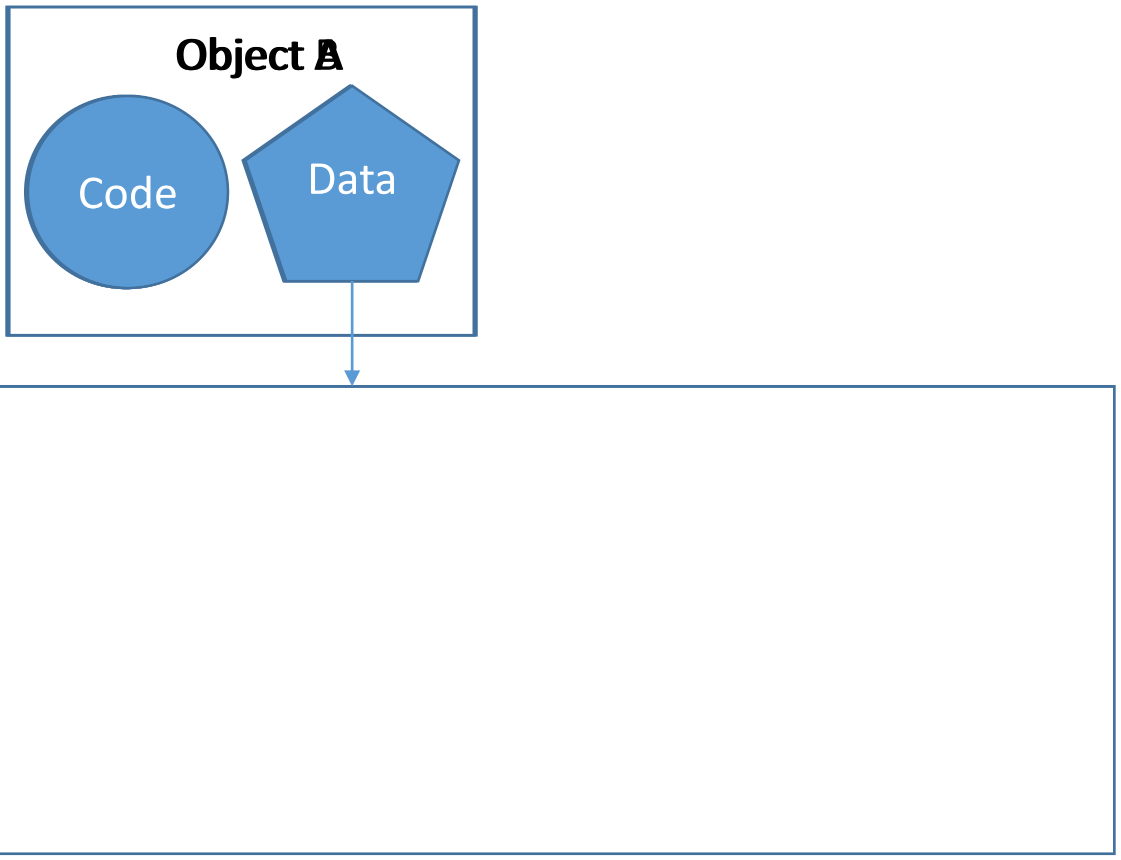
OOABL vs Procedural Concepts – Procedural Model



OOABL vs Procedural Concepts – OOABL Model

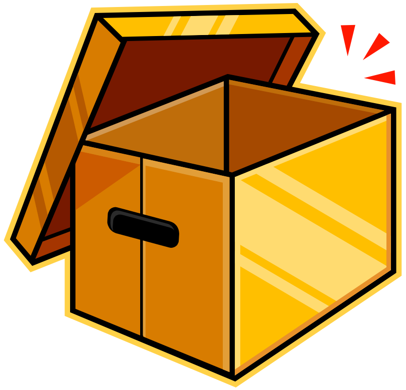


Time for a Paradigm Shift…



Adjusting Your Mind

Procedural Oriented
Data Structures


Object-Oriented
Data Structures

Session Overview

- Why Learn OOABL?
- OO / Procedural Programming Models
- <u>What's Involved in Learning OOABL</u>
- "Things You Can Do Now" Code Samples
- Reading List
- Questions

Learning OOABL: Technology - Language Elements and Concepts

OO Technology Language Elements:

- **Classes**
- **Properties**
- **Methods**
- **Data Access**
- Inheritance
- Static Members
- **Constructors**
- Destructors
- Interfaces
- Events

"The Tools"

OO Concepts:

- **Encapsulation**
- **Code re-use**
- **Granularity Control**
- **Patterns**
- Abstraction
- **Overloading**
- **Aggregation**
- Overriding
- Composition
- Delegation
- **Strong typing**
- Polymorphism

"How-To Instructions"

Learning OOABL: Concepts - Pattern Design and Use

Christopher Alexander says, "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice."[1]

Elements of a design pattern[2]:

- Name of the pattern
- Problem and it's context
- Generalized Solution
- Consequences – results / trade-offs of applying the pattern

1 Design Patterns – Elements of Reusable Software, page 2
2 Ibid – Page 3

Learning OOABL: Concepts - Patterns

Pattern Name	Examples
Singleton	SESSION handle
Parameter Object	ProDataSet
Model-View-Controller	User Interfaces

Learning OOABL: Changing Your Mind…

“The most difficult problem in teaching object-oriented programming is getting the learner to give up the global knowledge of control that is possible with procedural programs, and rely on local knowledge of objects to accomplish their tasks.“

A Laboratory for Teaching Object-Oriented Thinking
Kent Beck, Apple Computer Inc
Ward Cunningham, Wyatt Software Services Inc.
OOSPLA ’89 Conference proceedings

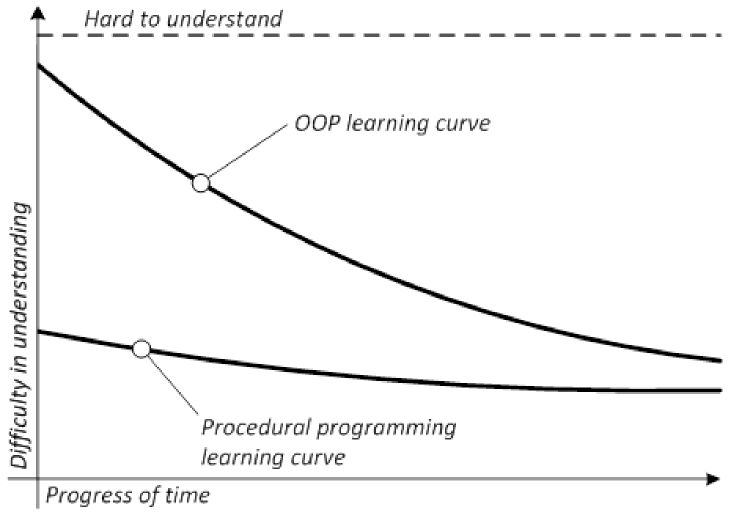
Programmers work out a model in their heads of how things work and have some trouble dislodging that model once they've tested it and come to believe in it.

Structured Programmers Learning Object-Oriented Programming - John Minor Ross and Huazhong Zhang
http://bulletin.sigchi.org/1997/october/papers/ross/

Expert structured programmers shifting to OO technology must be patient and should not expect to master OOP overnight.

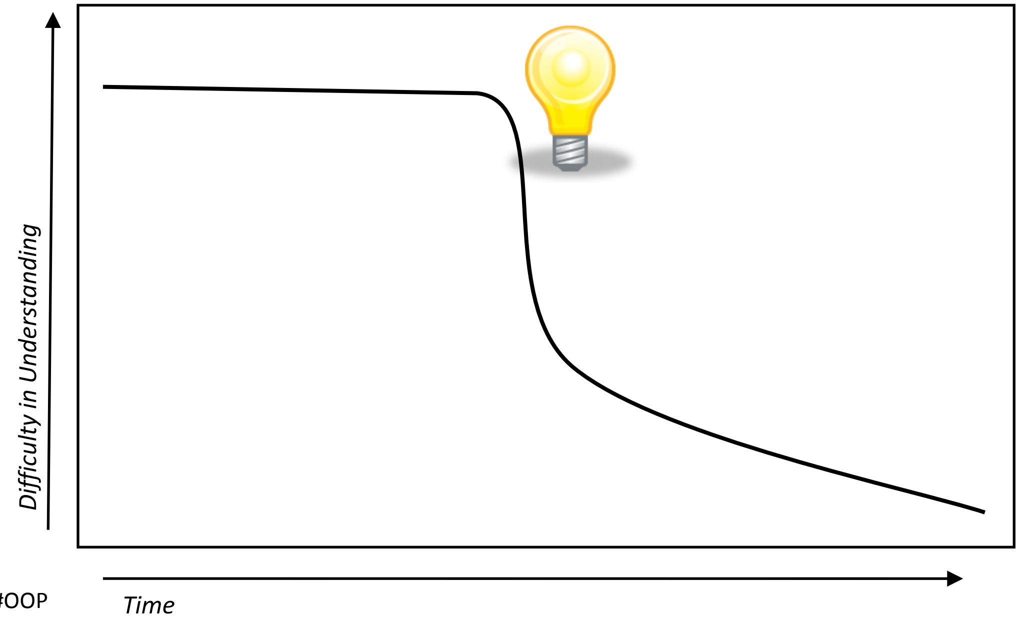
Structured Programmers Learning Object-Oriented Programming - John Minor Ross and Huazhong Zhang
http://bulletin.sigchi.org/1997/october/papers/ross/

Learning OOABL: The Learning Curve



Refactoring and Design Patterns by Peter Kaptein
http://patterns.instantinterfaces.nl/current/Refactoring-and-Design-Patterns-OOP.html#OOP

"Aha!" Learning Model



Learning OOABL: Mentorship Resources

There's "Knowing the OO Technology" and "Knowing How to Use OO Technology"

Q: "How do I know if I'm doing it right?"
A: OO Expert
A: OO Refactoring, Development, and Pattern Books
A: Compare with examples from the literature
A: Google
A: YouTube

Learning OOABL: The Gordian Knot of the OOABL Paradigm Shift

Q: If I'm happy with how things are going now, why should I learn OOABL?

A: Because a procedural developer can accomplish a lot using OOABL technology while developing proficiency in "thinking" OO

Session Overview

OO Sample Code – OO and TT Similarities

```
DEFINE TEMP-TABLE tt-name NO-UNDO
   FIELD ch1 AS CHARACTER
   FIELD i1    AS INTEGER.


CLASS ClassName:
DEFINE PROPERTY ch1 AS CHARACTER NO-UNDO GET. SET.
DEFINE PROPERTY i1    AS INTEGER      NO-UNDO GET. SET.

METHOD VOID            SetCh1(INPUT chValue   AS CHARACTER):
METHOD CHARACTER  GetCh1():

METHOD VOID            Seti1(INPUT iValue        AS INTEGER):
METHOD INTEGER        Geti1():

END CLASS.
```

OO Sample Code - BankLoan

```
DEFINE VARIABLE dePrincipal                AS DECIMAL  NO-UNDO.
DEFINE VARIABLE deInterestRate             AS DECIMAL  NO-UNDO.

DEFINE VARIABLE iAmortizationPeriod        AS INTEGER  NO-UNDO.
DEFINE VARIABLE deCompoundingPeriod        AS DECIMAL NO-UNDO.

RUN SomeProg.p( dePrincipal,
                deInterestRate,
                iAmortizationPeriod,
                iCompoundingPeriod).
```

OO Sample Code - BankLoan

```
CLASS BankLoan.LoanStructure:

DEFINE PUBLIC PROPERTY Principal                AS DECIMAL NO-UNDO GET. SET.
DEFINE PUBLIC PROPERTY InterestRate             AS DECIMAL NO-UNDO GET. SET.

DEFINE PUBLIC PROPERTY AmortizationPeriod    AS INTEGER  NO-UNDO GET. SET.
DEFINE PUBLIC PROPERTY CompoundingPeriod  AS DECIMAL NO-UNDO GET. SET.

END CLASS.
```

OO Sample Code - BankLoan

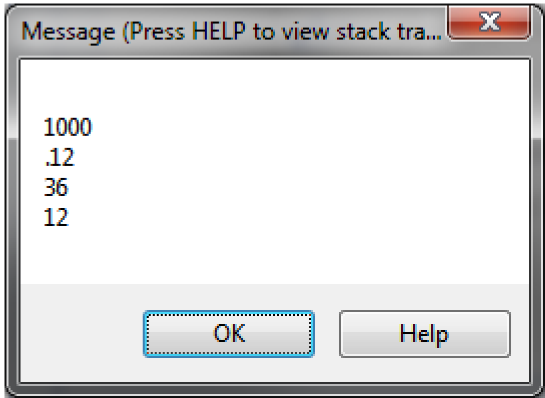
```
DEFINE VARIABLE oLoanStruct AS BankLoan.LoanStructure NO-UNDO.

oLoanStruct = NEW BankLoan.LoanStructure().

ASSIGN
  oLoanStruct:AmortizationPeriod   = 36
  oLoanStruct:CompoundingPeriod   = 12
  oLoanStruct:Principal           = 1000.00
  oLoanStruct:InterestRate        = 0.12
  .

RUN SomeCode.p(oLoanStruct).
```

OO Sample Code – Read Only Objects

```
CLASS BankLoan.LoanStructureValue:
DEFINE PUBLIC PROPERTY Principal             AS DECIMAL NO-UNDO GET. PRIVATE SET.
DEFINE PUBLIC PROPERTY InterestRate          AS DECIMAL NO-UNDO GET. PRIVATE SET.
DEFINE PUBLIC PROPERTY AmortizationPeriod    AS INTEGER  NO-UNDO GET. PRIVATE SET.
DEFINE PUBLIC PROPERTY CompoundingPeriod     AS DECIMAL NO-UNDO GET. PRIVATE SET.

CONSTRUCTOR LoanStructureValue(dePrinc AS DECIMAL, deIntRate AS DECIMAL,
                               iAmtPd  AS INTEGER, deCmpPd  AS DECIMAL):
ASSIGN
   THIS-OBJECT:Principal            = dePrinc
   THIS-OBJECT:InterestRate         = deIntRate
   THIS-OBJECT:AmortizationPeriod   = iAmtPd
   THIS-OBJECT:CompoundingPeriod    = deCmpPd
   .

END CONSTRUCTOR.
END CLASS.
```

OO Sample Code – Read Only Objects

```
DEFINE VARIABLE oLoanStructureValue AS BankLoan.LoanStructureValue NO-UNDO.

oLoanStructureValue = NEW BankLoan.LoanStructureValue(1000.0, 0.12, 36, 12).

MESSAGE oLoanStructureValue:Principal               SKIP
        oLoanStructureValue:InterestRate            SKIP
        oLoanStructureValue:AmortizationPeriod      SKIP
        oLoanStructureValue:CompoundingPeriod
  VIEW-AS ALERT-BOX.
```



```
1000
.12
36
12
```

OO Sample Code - Dates

```
DEFINE VARIABLE dtStart     AS DATE NO-UNDO.

/* Code for dtStart here… */

IF dtStart <> ? THEN
   DO: /* something */
   END.
```

OO Sample Code - Dates

```
CLASS Date.DateClass:
DEFINE PUBLIC PROPERTY DateValue    AS DATE       NO-UNDO GET. SET.

DEFINE PUBLIC PROPERTY IsValidDate  AS LOGICAL  NO-UNDO
                                     GET():      RETURN(THIS-OBJECT:DateValue <> ?).
                                     END GET.
                                     PRIVATE SET.
END CLASS.
```

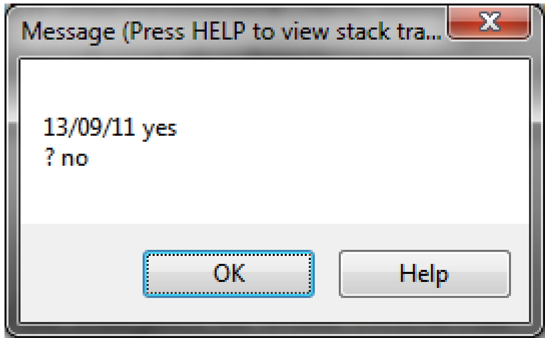
Learning OOABL – The Human Factor

Progress Exchange 2013

TDK Consulting Services Inc

OO Sample Code - Dates

```
DEFINE VARIABLE oStart  AS Date.DateClass NO-UNDO.
DEFINE VARIABLE oEnd    AS Date.DateClass NO-UNDO.

oStart  = NEW Date.DateClass().
oEnd    = NEW Date.DateClass().

oStart:DateValue    = TODAY.
oEnd:DateValue      = ?.

MESSAGE oStart:DateValue oStart:IsValidDate SKIP
            oEnd:DateValue   oEnd:IsValidDate
   VIEW-AS ALERT-BOX.
```



```
13/09/11 yes
? no
```

Using OO and Procedural Code – Extract Class

```
DEFINE INPUT PARAMETER iCustomer AS INTEGER NO-UNDO.

DEFINE TEMP-TABLE tt-event-log NO-UNDO
    FIELD event-number  AS INTEGER
    FIELD event-record    AS CHARACTER
    .
DEFINE VARIABLE iEventCount    AS INTEGER NO-UNDO.

RUN ipName(iCustomer).

RUN CaptureEvent("firstevent").
RUN CaptureEvent("secondevent").
RUN CaptureEvent("thirdevent").

PROCEDURE CaptureEvent:
DEFINE INPUT PARAMETER ch-event AS CHARACTER NO-UNDO.
END PROCEDURE.
```

This is the only code that really belongs here

Using OO and Procedural Code – Extract Class

```
CLASS  Example.EventLog:

DEFINE TEMP-TABLE tt-event-log NO-UNDO
   FIELD event-number  AS INTEGER
   FIELD event-record    AS CHARACTER.

DEFINE VARIABLE iEventCount    AS INTEGER NO-UNDO.

METHOD PUBLIC VOID CaptureEvent(ch-event AS CHARACTER):

END CLASS.
```

Using OO and Procedural Code – Extract Class

```
DEFINE INPUT PARAMETER iCustomer AS INTEGER NO-UNDO.

DEFINE VARIABLE oEventlog AS Example.EventLog    NO-UNDO.

oEventlog = NEW Example.EventLog().

RUN ipName(iCustomer).

oEventlog:CaptureEvent("firstevent").
oEventlog:CaptureEvent("secondevent").
oEventlog:CaptureEvent("thirdevent").
```

Using OO and Procedural Code – Extend Class

```
DEFINE INPUT PARAMETER iCustomer AS INTEGER NO-UNDO.

DEFINE TEMP-TABLE tt-event-log NO-UNDO
    FIELD event-number  AS INTEGER
    FIELD event-record   AS CHARACTER
    FIELD event-type      AS CHARACTER
    .

RUN ipName(iCustomer).

RUN CaptureEvent("slow",  "firstevent").
RUN CaptureEvent("",        "secondevent").
RUN CaptureEvent("",        "thirdevent").

PROCEDURE CaptureEvent:
DEFINE INPUT PARAMETER ch-event-type AS CHARACTER NO-UNDO.
DEFINE INPUT PARAMETER ch-event AS CHARACTER NO-UNDO.
```

This is the only code that really belongs here

Using OO and Procedural Code – Extend Class

```
CLASS  Example.EventLog:

DEFINE TEMP-TABLE tt-event-log NO-UNDO
    FIELD event-number  AS INTEGER
    FIELD event-record    AS CHARACTER
    FIELD event-type       AS CHARACTER.

DEFINE VARIABLE iEventCount    AS INTEGER NO-UNDO.

METHOD PUBLIC VOID CaptureEvent(ch-event         AS CHARACTER):
METHOD PUBLIC VOID CaptureEvent(ch-event-type AS CHARACTER,
                                ch-event         AS CHARACTER):

END CLASS.
```

Using OO and Procedural Code – Extend Class

```
DEFINE INPUT PARAMETER iCustomer AS INTEGER NO-UNDO.

DEFINE VARIABLE oEventlog AS Example.EventLog    NO-UNDO.

oEventlog = NEW Example.EventLog().

RUN ipName(iCustomer).

oEventlog:CaptureEvent("slow", "firstevent").
oEventlog:CaptureEvent("secondevent").
oEventlog:CaptureEvent("thirdevent").
```

Using OO and Procedural Code – Add Object Instance

```
DEFINE INPUT PARAMETER iCustomer AS INTEGER NO-UNDO.

DEFINE VARIABLE oOperatorEventLog      AS Example.EventLog   NO-UNDO.
DEFINE VARIABLE oTransactionEventLog   AS Example.EventLog   NO-UNDO.

oOperatorEventLog    = NEW Example.EventLog().
oTransactionEventLog = NEW Example.EventLog().

RUN ipName(iCustomer).

oTransactionEventLog:CaptureEvent("slow", "firstevent").

oOperatorEventLog:CaptureEvent("secondevent").
oOperatorEventLog:CaptureEvent("thirdevent").
```

Using OO and Procedural Code – Parameter Object

```
CLASS Example.ActivityParameter:

DEFINE PUBLIC PROPERTY CustomerID   AS INTEGER  NO-UNDO GET. SET.

DEFINE PUBLIC PROPERTY OperatorEventLog      AS Example.EventLog NO-UNDO GET. SET.
DEFINE PUBLIC PROPERTY TransactionEventLog  AS Example.EventLog NO-UNDO GET. SET.

CONSTRUCTOR ActivityParameter():
OperatorEventLog     = NEW Example.EventLog().
TransactionEventLog = NEW Example.EventLog().
END CONSTRUCTOR.

END CLASS.
```

Using OO and Procedural Code – Parameter Object

```
DEFINE VARIABLE oActivityParam AS Example.ActivityParameter NO-UNDO.

oActivityParam            = NEW Example.ActivityParameter().
oActivityParam:CustomerID = 1.

RUN ObjectActivityParameter.p(oActivityParam).
RUN ObjectActivityParameterReport.p(oActivityParam).
```

Using OO and Procedural Code – Parameter Object

```
DEFINE INPUT PARAMETER oActivityParm AS Example.ActivityParameter NO-UNDO.

RUN ipName(oActivityParm:CustomerID).

oActivityParm:TransactionEventLog:CaptureEvent("slow", "firstevent").

oActivityParm:OperatorEventLog:CaptureEvent("secondevent").
oActivityParm:OperatorEventLog:CaptureEvent("thirdevent").
```

Session Overview

- Why Learn OOABL?
- OO / Procedural Programming Models
- What's Involved in Learning OOABL
- "Things You Can Do Now" Code Samples
- Reading List
- Questions

OOABL Reading List

- PSC Education and OO Docs
- Head First – Object-Oriented Analysis and Design
- Head First – Design Patterns
- Implementation Patterns
- Patterns of Enterprise Application Architecture
- Refactoring to Patterns
- Service Design Patterns
- Design Patterns – Elements of Reusable OO Software
- Domain-Driven Design (Quickly)
- Domain-Driven Design
- Antipatterns
- Thinking In Java

Closing Thoughts

- OO is about managing complexity
- Transitioning from Procedural to OO will take time
- Immediate benefits can be realized during the transition

Session Overview

- Why Learn OOABL?
- OO / Procedural Programming Models
- What's Involved in Learning OOABL
- "Things You Can Do Now" Code Samples
- Reading List
- Questions

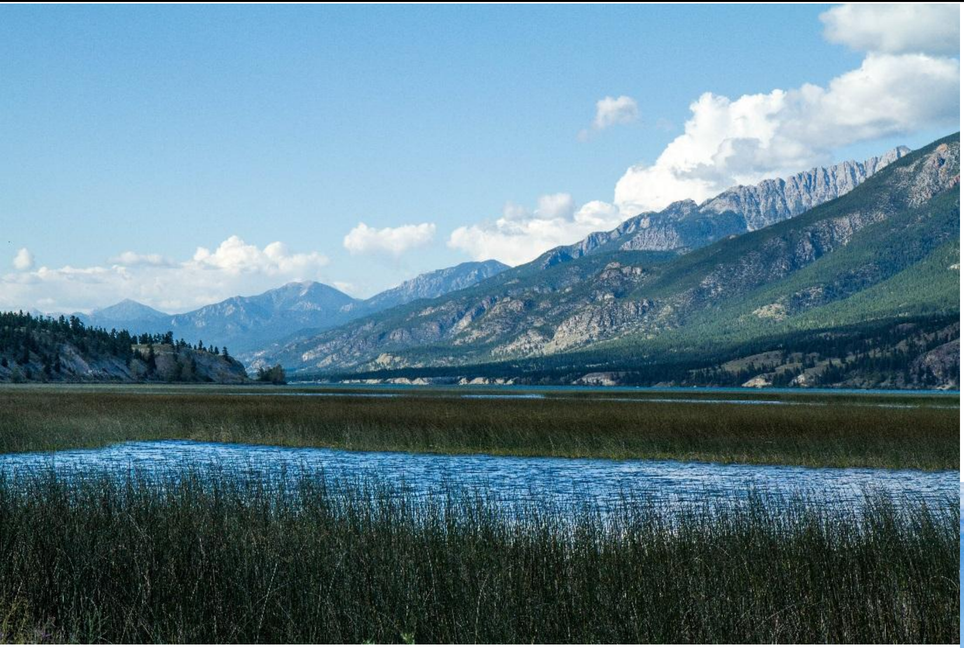
Questions?

Learning OOABL – The Human Factor

Progress Exchange 2013

TDK Consulting Services Inc

Thank You For Your Time and Attention!



Canal Flats – British Columbia