

Smalltalk & Time to Market

Peter William Lount
Active Information Corporation

ActiveInfo.ca Smalltalk.org

peter@smalltalk.org

Project Outcome Frame

- What is the vision and purpose of a project, product or service? Why do it?

Outcome Frame

- State the outcome in positive terms.
- What are the criteria for success?
- Must be maintainable by self or the team.
- Must be S.M.A.R.T.
 - Specific, Measurable, Achievable, Realistic and Timed.
- Ecological. Must work in the real world.

Actually getting to Market?

- A chief concern is actually getting your project, product or service to market.
- How can we improve the odds that the technological and human factors can be solved within real project time frames?

Time to Market

- How long will it takes to realize your project?
- Who is the project, product or service aimed at?
- How well does the project, product or service meet the needs of it's target audience?
- How can we improve the time to market of software systems using Smalltalk?

What are the advantages of Smalltalk that enable a rapid time to market?

- The Nature of the Smalltalk Language itself.
- Untyped Language Syntax and System.
- The Integrated Development Environment.
- Excellent state of the art versions of Smalltalk from many different vendors.
- Excellent class libraries.

What development methodologies can be used to improve the time to market?

- Object Oriented Analysis & Design Methodologies
 - Extreme Programming (XP)
 - Agile
 - Many different methodologies.
- Value and Results Focus (VRF) Process
 - A process that humans use to focus on human interaction in teams working on a project.
 - Includes a set of easily learnable skills.

The Nature of the Smalltalk Language

- Smalltalk maps very well to a subset of simplified human languages.
- Smalltalk is simple: a world of objects communicating with messages.
- Designed for humans yet can satisfy hard core “rubex cube” coders.
- Smalltalk variables are Untyped.
- Smalltalk is extensible with Blocks.

The Nature of the Smalltalk Language

- Smalltalk maps very well to a subset of simplified human languages.
- In particular Smalltalk maps well to simplified business English writing style as described by Tom McKeown in his book “Powerful Business Writing” (ISBN 0-07-551382-X).

Similarity of Smalltalk to Simplified Human Language

- “The power English sentence presents one main thought, uses the active voice, uses an action verb and uses specific words.”

Specific Subject	Specific Action Verb	Specific Object
Tom	bounce	the red ball.
aPerson	bounce:	theRedBall.

Similarity of Smalltalk to Simplified Human Language

- Why does this similarity matter?
- It can simplify the translation process between the “business requirements” of a project and the “software reality” of a project.
- It enables the possibility of creating implementations that are relevant and correct.
- If used appropriately it can help improve the communication between the business people and the technology people on a project.

Similarity of Smalltalk to Simplified Human Language

- How can we take advantage of the similarity of Smalltalk and Simplified Business English?
- By using Simplified Business English Style as the common language among the project stake holders and the project implementers.
- The idea is to use Simplified Business English to increase the understanding between all team members of a project, and to push this clarity forward into the actual analysis, design implementation and testing phases of a project.

The Nature of the Smalltalk Language

- Designed for humans yet can satisfy hard core “rubex cube” coders.
 - The idea is to make programming less complex, easier to grasp, easier to use.
 - This means a simpler syntax is better.
- Smalltalk variables are Untyped for Humans.
 - Since typed variables are more complex to use it’s better to not have them in a computer language IF you goal is to simplify for humans.

The Nature of the Smalltalk Language

“In English, word order is crucial.

Phrases can be arranged and rearranged like putty,

turning nouns into verbs,

verbs into nouns, nouns into adjectives.

One can, as the *Encyclopedia Britannica* observes,

plan a table or table a plan,

book a place or place a book,

lift a thumb or thumb a lift.”

- U.S. News & World Report

Literate Programming

“I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by

**considering programs to be
works of literature.**

Hence, ... "Literate Programming.”

-Donald Knuth.

"Literate Programming (1984)" in

Literate Programming. CSLI, 1992, pg. 99.

Also, <http://www.LiterateProgramming.com>.

Literate Programming

“Let us change our traditional attitude to the construction of programs:

Instead of imagining that our main task is to instruct a computer what to do,

let us **concentrate** rather **on explaining to human beings what we want a computer to do.**”

- Donald Knuth

Literate Programming

“The practitioner of literate programming can be regarded as an essayist, whose **main concern is with exposition and excellence of style.** ...

He or she **strives for a program that is comprehensible** because its concepts have been introduced **in an order that is best for human understanding**, using a mixture of formal and informal methods that reinforce each other.”

- Donald Knuth

Literate Smalltalk Programming

- Keep Smalltalk sentences simple by using variables for all or most sub phrases.
- Group Smalltalk sentences that are related into “power paragraphs” using “vertical white space” (blank lines) as a visual separator, or even better, break “power paragraphs” into separate methods if possible.
- Reduce “rubex cube coding” to a minimum or eliminate it altogether when possible.
- Extend the language using blocks when this increases clarity or improves correctness.

Literate Smalltalk Programming

- Choose variable names that make sense and that distinguish clearly between method parameters, method local variables and object instance variables.
- Name object classes with clear names that make sense.

Smalltalk Blocks

- Smalltalk is extensible with Blocks.
 - Blocks are one of the most powerful aspects of Smalltalk.
 - Blocks are like phrases in human languages, they express a sub idea that results in a noun.
 - Enables “complex flow control” or “thought” structures to be “captured” for reuse and to ensure “correctness” when reused.

Smalltalk Block Extension Example 1

```
anObject isNil ifTrue: [  
    anObject := Object new  
] ifFalse: [  
    “Object exists...”  
].
```

The above uses the original “isNil ifTrue:” method of testing whether an object is the nil object or not.

This leads to cryptic coding style that can slow down the human comprehension of a program.

```
anObject ifNil: [  
    anObject := Object new  
] ifNotNil: [  
    “Object exists...”  
].
```

The example on the right uses the newer “ifNil:” method of testing whether an object is the nil object or not.

This leads to literate programming style that tends to increase human comprehension due to the explicitness of “ifNil:”.

Smalltalk Block Extension Example 2

```
anList
```

```
doFirst: [:aFirstElement |  
    aFirstElement someMessage  
] doMiddle: [:anElement |  
    anElement someOtherMessage  
] doLast: [:aLastElement |  
    aLastElement someMessageForTheLastElement  
].
```

The above example of an extension to the Smalltalk collection classes enables having separate blocks of code execute for the elements in the collection based upon their position in the collection. The user doesn't have to do the tests for these positions since that is taken care of in the doFirst:doMiddle:doLast: method that uses blocks to extend the Smalltalk language.

Smalltalk Block Extension Example 3

anList

```
doIfNone: ["do this when the list is empty"]
doIfOne: [:aSingleElement | "do this to the only element"]
doFirst: [:aFirstElement | "do this to the first element"]
doMiddle: [:aMiddleElement | "do this to the middle elements"]
doLast: [:aLastElement | "do this to the last element"]
doBefore: [:aFirstElement | "do this before the first element"]
doBetween: [:aPreviousElement :aNNextElement |
    "do this between the two elements"]
doAfter: [:aLastElement | "do this before the last element"]
doOnError: [:theException | "take care of any errors"]
```

This full work horse method from the collection extension encodes many of the sometimes needed variations on basic looping when one has to deal with separate elements of a list different ways.

Smalltalk Blocks – Two Return Values

- Some methods need to return two (or more) objects and that don't want to encapsulate them into a new class can pass in a block (or more) with the code to execute on these return values.
 - This works when there isn't much code too execute after.
 - Similar to an “after method”.
 - A bit too much of a “rubex cube” solution for my tastes since it requires more rather than less comprehension.
 - Also may not be generic enough since another user may want the two (or more) objects encapsulated!
 - Your mileage may vary.

Objects instead of Smalltalk Blocks

- Imagine custom objects that implement the block “value:” protocols and act as if they are a block.
- Inspired by the **ease of multi-column sorting** in Excel.
- Why can't I sort as easily, or even easier than in Excel?
- The Sort Critter (ok, **AIMSSortCriteria**) uses the neat trick of polymorphism to implement the BlockContext “value:value:” protocol.
 - In some implementations a couple of other methods are needed usually two or three.
- The SortCriteria can be used in place of a sort block in a SortedCollection.

SortCriteria instead of a Sort Block

- Along with the SortCriteria you also have SortCriteriaColumn.
- The **SortCriteriaColumn** encodes the sorting options (method aspect selector and ascending or descending sort) for N multiple columns where N can vary from 1 to any practical integer of columns that you care to sort.
- Somewhat efficient since it only needs to compare multiple columns when the prior columns were equal.
- Unfortunately it does use “perform” but you can code custom sort column sub classes; or,
- It also supports dictionary “at:” protocol instead of “perform”.
- The take home point: faster development and good performance.

SortCriteria Example

```
| aSortedList aSortCriteria |
```

```
aSortCriteria := SortCriteria new.
```

```
aSortCriteria addColumnName: #code ascendingFlag: true.
```

```
aSortCriteria addColumnName: #size ascendingFlag: false.
```

```
aSortCriteria addColumnName: #length ascendingFlag: false.
```

```
aSortedList := SortedCollection sortBlock: aSortCriteria.
```

“That’s it! Now simply fill your sorted list with your objects.

Also works with existing sorted collections, just send

```
aSortedCollection sortBlock: aSortCriteria
```

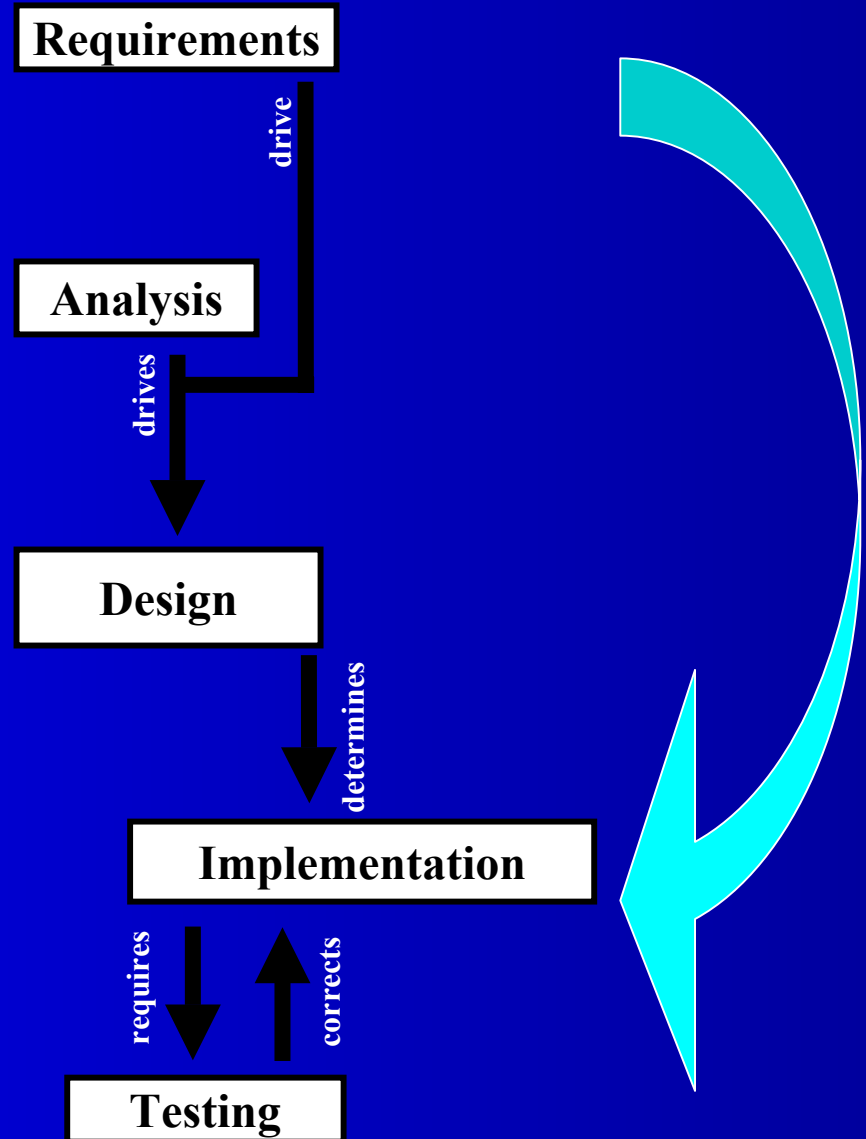
to the sorted collection instance.”

Sort Criteria Benefits

- **Dynamic** by changing the sort criteria columns **on the fly**.
- Note that the **SortedCollection** required **no changes** what so ever. Zero. Zippo. None. ;--)
- **Simple, effective and powerful** way to **save time coding** multiple column sorts.
- **Reduces errors** due to **correct code** testing once. No more writing the multi-column test algorithm.
- **First extension** to Smalltalk's sorting system since 1980.
- Small, with less than 12k including a large test class.
- **Open source**. Smalltalk.org/components/SortCriteria.html
- **Vendors please include it** with your BASE Smalltalk release.
- Nice example of the **power of polymorphism and blocks**.

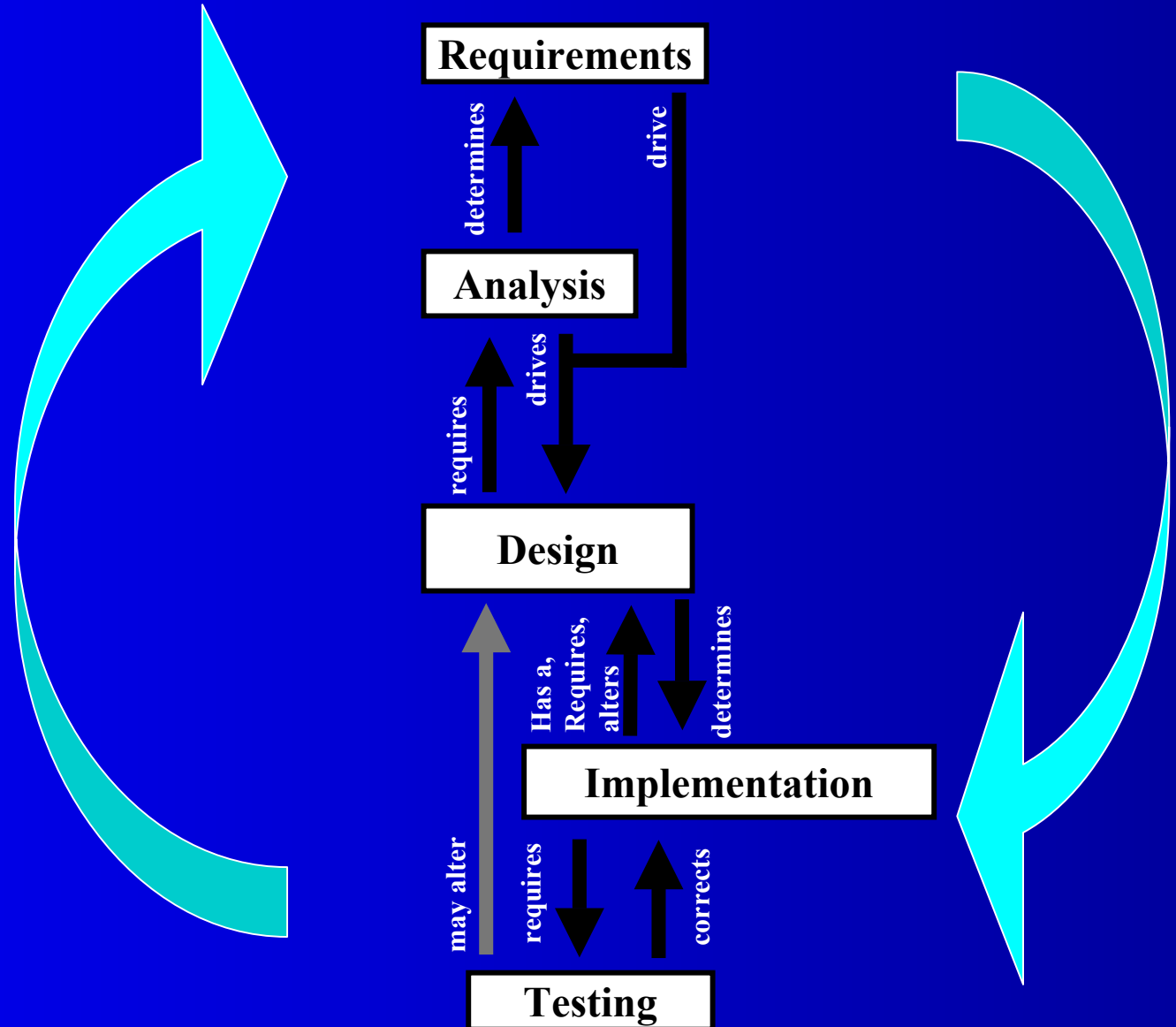
Traditional Development Process

Active Information Corporation



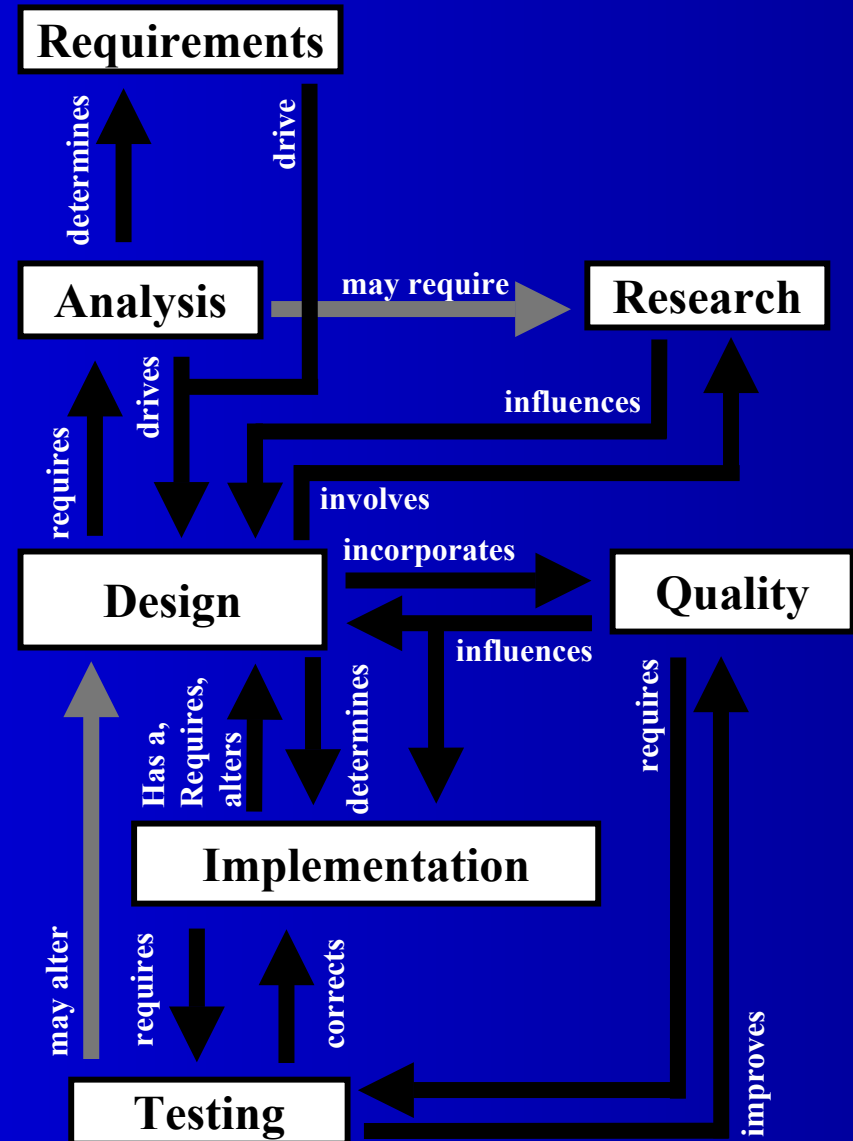
Improved Development Process

Active Information Corporation



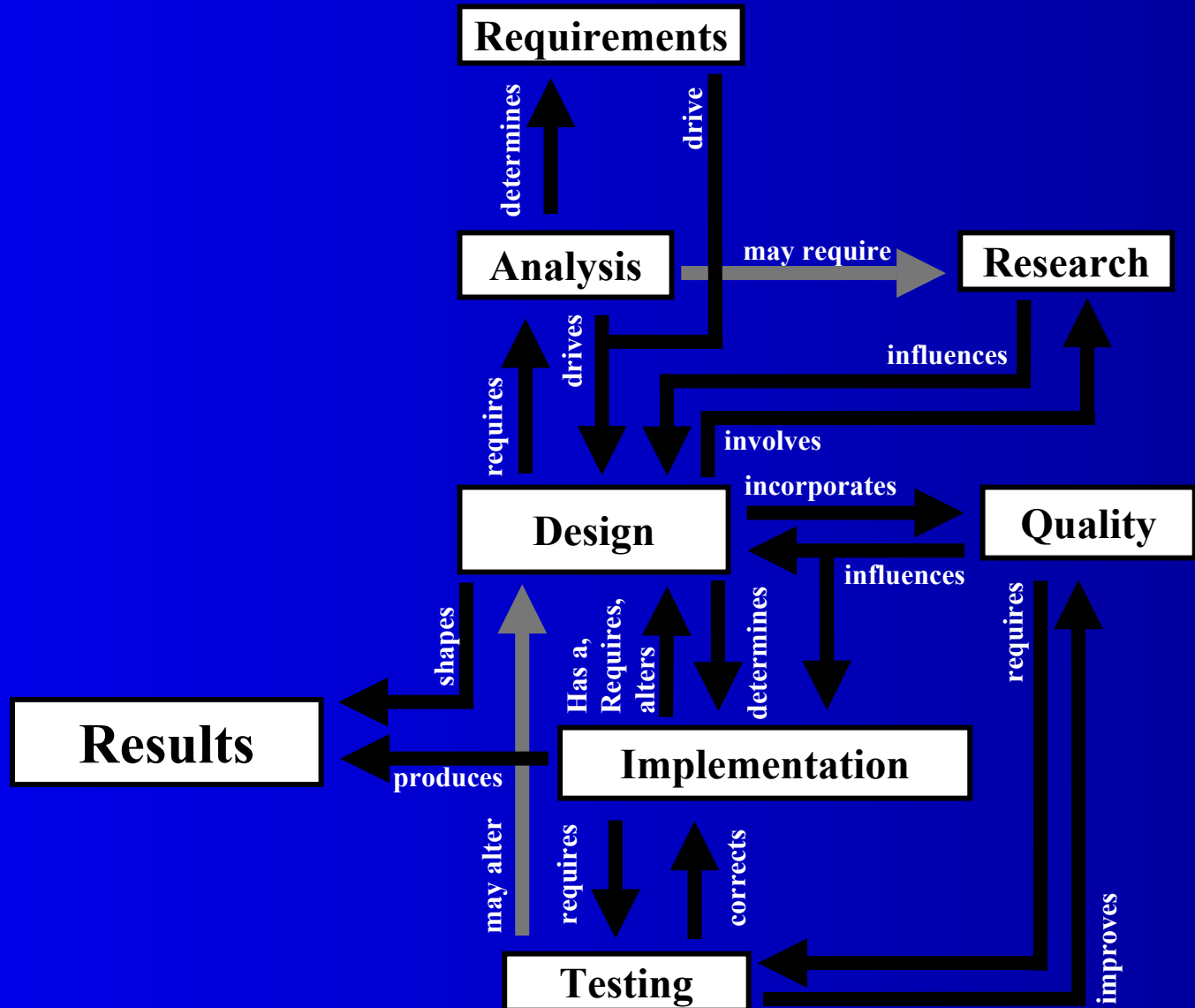
Extended Development Process

Active Information Corporation



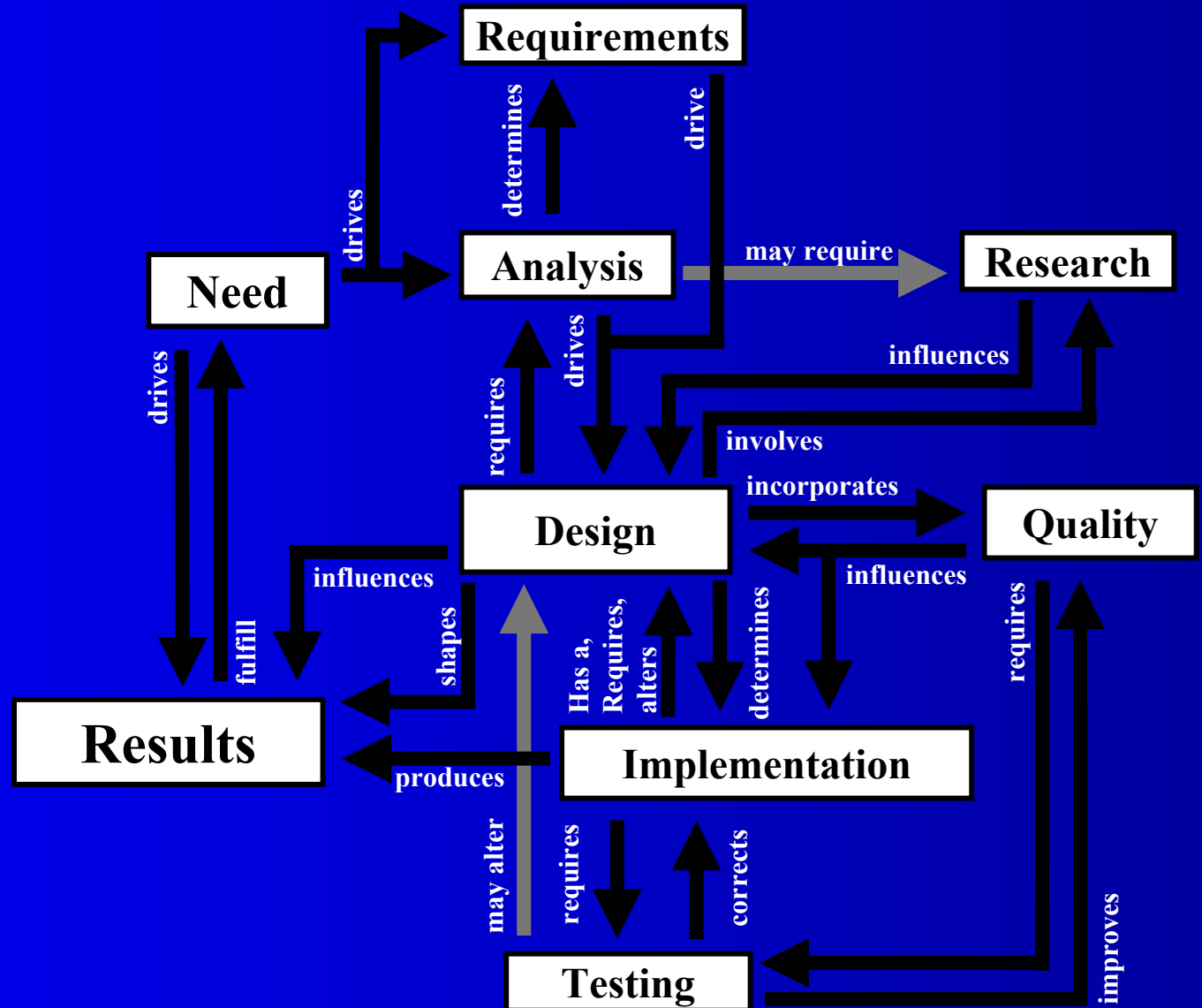
Result Focused Process

Active Information Corporation



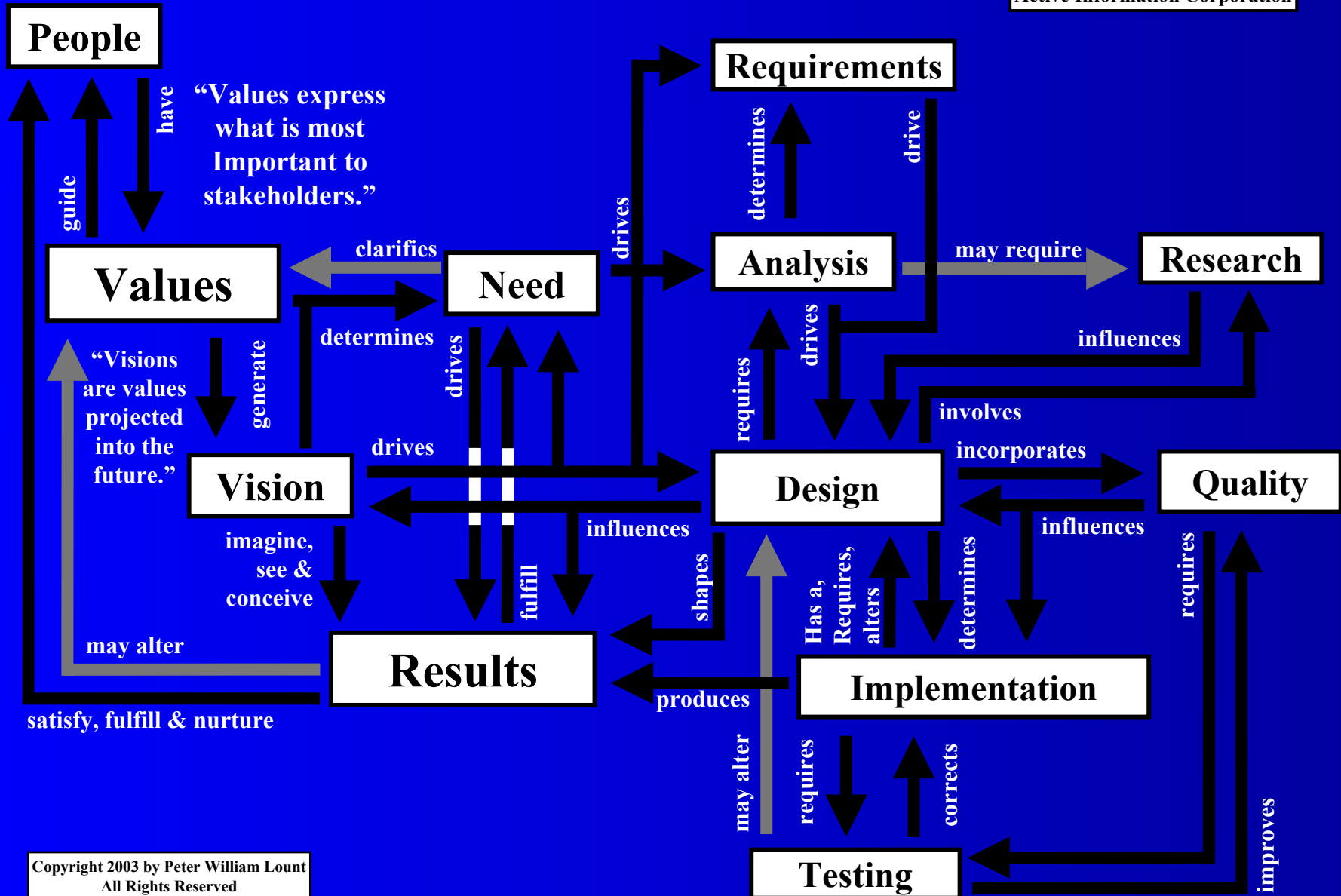
Need & Result Process

Active Information Corporation



Value & Results Focus (VRF) Process

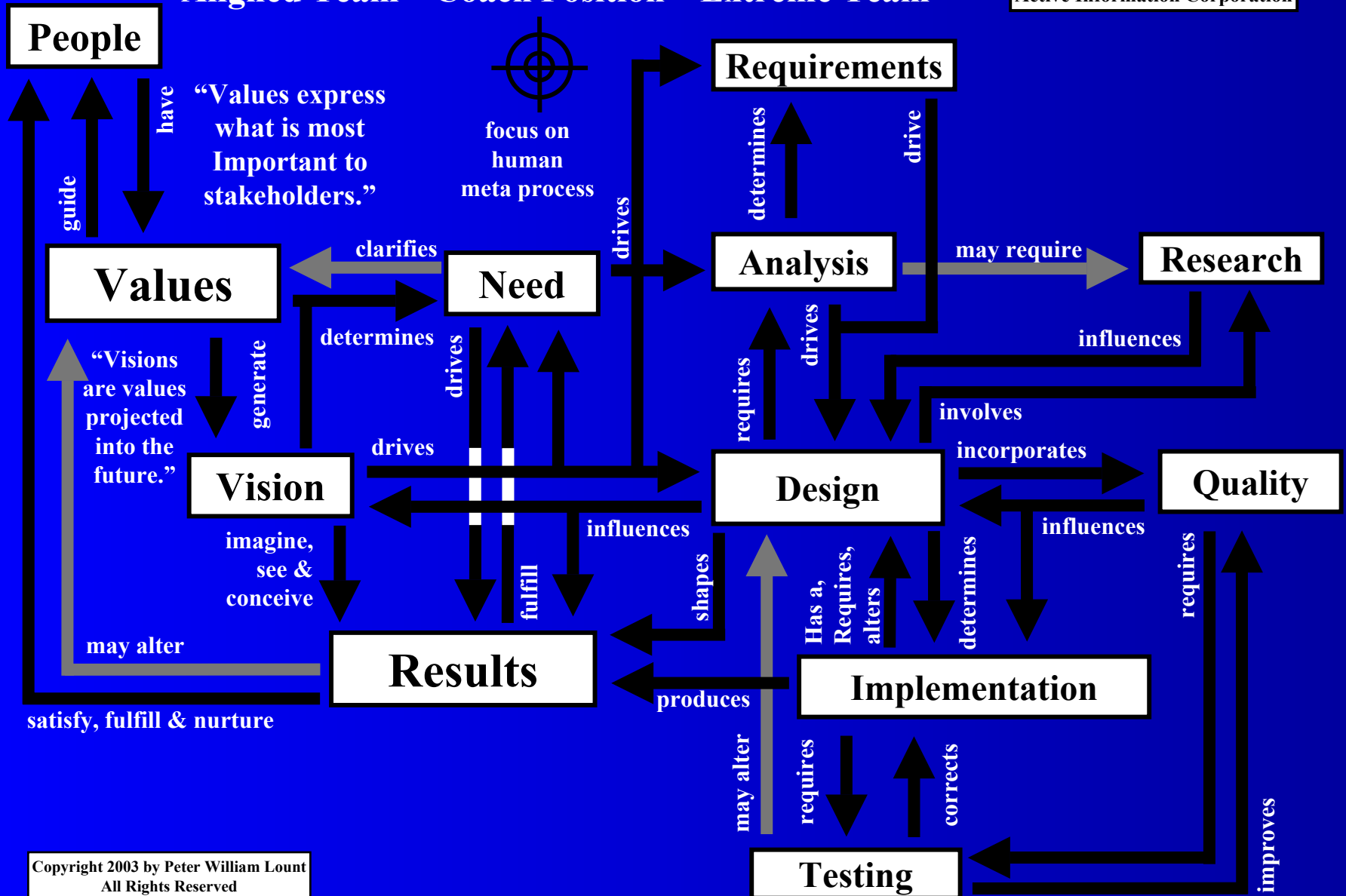
Active Information Corporation

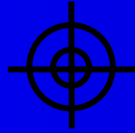


Extreme Value & Results Focus (E-VRF) Process

Aligned Team + Coach Position = Extreme Team

Active Information Corporation





**focus on
human
meta process**

Coach Position

- While in Coach Position (CP) you are observing and evaluating your, and your teams, development process.
- Offers a “forum” for communicating about the meta process at all levels in the project’s organization.
- Tends to flatten the team organization for communication purposes.
- The process helps ensure that all team members are on the same page when needed, and lets them excel on their own as needed.
- In Coach Position you may choose to alter the current direction of the development process to fine tune the results.
- Coach Position is a “judgment free” neutral observer position.
- There are many tools available while in CP.

Coach Position and Extreme Programming

- Some of the key reasons Extreme Programming (XP) works is:
 - that it's easier to enter Coach Position when there is another person, with the same goals and purpose, present.
 - You catch each others mistakes.
 - You leverage each others skills.
 - Two heads are often better than one.
- The Extreme Value & Results Focus (E-VRF) Process can be viewed as an extension of XP for the whole team.
- The E-VRF Process has many tools that facilitate team communication and focus on relevant actions.

Coach Position

Solution Focused Tools

- Outcome Frame
- Logical Levels
- Failing Forward To Success
- As if Frame
- The Miracle Question
- Value Based Self Image
- Positive & Powerful Feedback Frame
- Relevancy Frame
- Four Tones
- Present State – Desired State
- Presuppositions
- Backtracking
- (Un)Conscious & (In)Competence – Matrix
- Important – Urgent Matrix
- Coaching Contract
- Committed Conversations
- Time Line
- State Line – Emotions
- Questions
- Scaling
- 100+ More tools...

Conclusion

- Human language and Smalltalk are very similar in some important ways. Lets leverage that.
- Smalltalk has powerful blocks.
- Literate programming is effective.
- Human process matters in projects of all kinds.
- A focus on the interaction process between people on a project team makes a big difference.
- The flow of the Extreme Value and Results (EVR) Process is what we already do.
- Enhance that with Coaching Skills and see the productivity improvements occur.

Thank you

Peter William Lount, Consultant & Coach

Active Information Corporation

Vancouver, B.C., Canada

Peter@ActiveInfo.ca

Peter@Smalltalk.org

Smalltalk.org

- Please contribute to the web site.
- New dynamic Smalltalk driven version in the works.
- Please support our activities by advertising your company or consulting services.
 - You get `YourName@Smalltalk.org` plus more