Using scripting languages in products *can accelerate change !*

SASG meeting 7-feb-2012

Tom Hoogenboom, ASML

Here are some statements to start off discussions

- For some reason script languages have always been popular
    - except with ‘real’ programmers

- Today I present some statements claiming that use of scripting languages, compared to conventional languages, can accelerate change
    - with only marginal disadvantages...

- Please consider why you agree / disagree with these statements → → discussion . . .

Summary

- Historical perspective
- There are many popular ones
- Scripting languages are not for serious programming
- Scripting languages can accelerate change

Summary

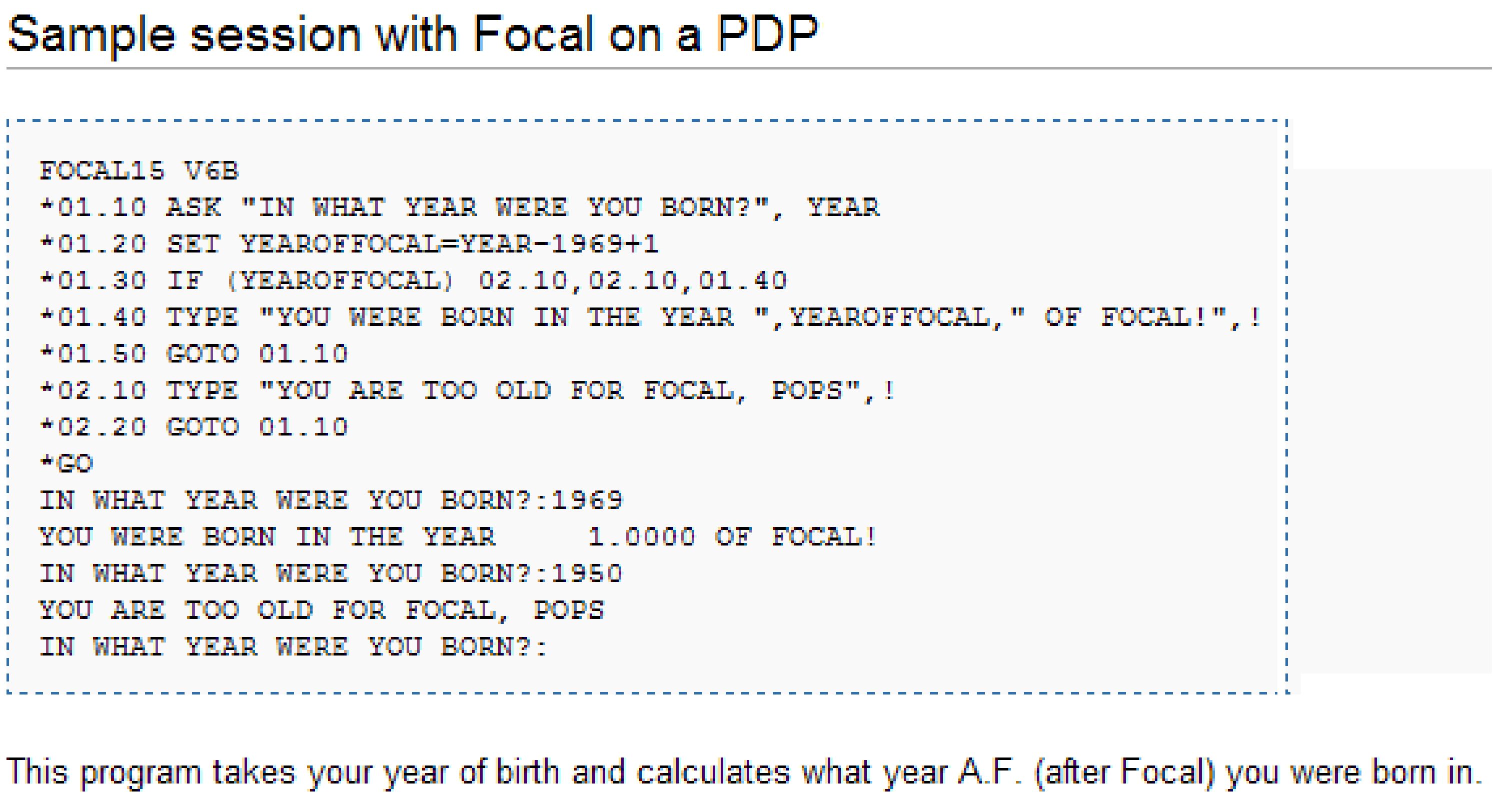
- Historical perspective
- There are many popular ones
- Scripting languages are not for serious programming
- Scripting languages can accelerate change

Scripting languages are as old as the computer

Scripting languages are as old as the computer

- From my memory

  1973: FOCAL on PDP-8



Sample session with Focal on a PDP

```
FOCAL15 V6B
*01.10 ASK "IN WHAT YEAR WERE YOU BORN?", YEAR
*01.20 SET YEAROFFOCAL=YEAR-1969+1
*01.30 IF (YEAROFFOCAL) 02.10,02.10,01.40
*01.40 TYPE "YOU WERE BORN IN THE YEAR ",YEAROFFOCAL," OF FOCAL!",!
*01.50 GOTO 01.10
*02.10 TYPE "YOU ARE TOO OLD FOR FOCAL, POPS",!
*02.20 GOTO 01.10
*GO
IN WHAT YEAR WERE YOU BORN?:1969
YOU WERE BORN IN THE YEAR     1.0000 OF FOCAL!
IN WHAT YEAR WERE YOU BORN?:1950
YOU ARE TOO OLD FOR FOCAL, POPS
IN WHAT YEAR WERE YOU BORN?:
```

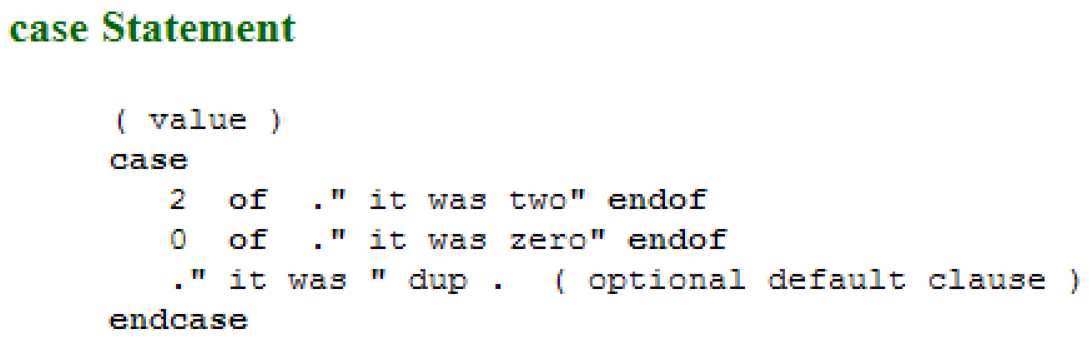
This program takes your year of birth and calculates what year A.F. (after Focal) you were born in.

Another early example:
1975: FORTH, on ‘any computer’

- 1975: FORTH on any computer
  Example: 6809 structured assembler / disassembler:

```
. . .
\ Structured assembler constructs.
: IF >R A; R> C, >MARK ;
: THEN A; >RESOLVE ;
: ELSE A; $20 C, >MARK SWAP >RESOLVE ;
: BEGIN A; <MARK ;
: UNTIL >R A; R> C, <RESOLVE ;
: WHILE >R A; R> C, >MARK ;
: REPEAT A; $20 C, SWAP <RESOLVE >RESOLVE ;
: AGAIN $20 UNTIL ;
. . .
```

- example, today: Sun Sparc console command interpreter/OLPC



case Statement

```
( value )
case
   2  of  ." it was two" endof
   0  of  ." it was zero" endof
   ." it was " dup .  ( optional default clause )
endcase
```

Extreme scripting: MasterMind in APL

(A Programming Language (?))

```
mmind←{                     ⍝ Mastermind or "cows and bulls".
     '*'∨.≠(⍴⍵)↑⎕←'*+'/⍨⍵{(⍺+.=⍵),⍺{+/⊃⌊/+/¨(⊂∪⍺)∘.=¨(⍺≠⍵)∘/¨⍺ ⍵}⍵}⍞:∇ ⍵
}
```

Summary

- Historical perspective:
  scripting languages have always been popular
- There are many popular ones
- Scripting languages are not for serious programming
- Scripting languages can accelerate change

Have you heard of all of these?

AppleScript	Game Maker Language (GML)	**R**
AWK	**Groovy**	REBOL
Bash	ICI	Revolution
BeanShell	Io	REXX
Candle	JASS	**Ruby**
Ch (Embeddable C/C++ interpreter)	**Javascript**	**sed**
CLIST	Join Java	S-Lang
CMS EXEC	**Lua**	**Smalltalk**
ColdFusion	MAXScript	**Tcl**
DCL	MEL	Tea
ECMAScript	Mondrian	TorqueScript
EXEC 2	Mythryl	**Unix Shells**
Falcon	**Perl**	VBScript
Fancy	**PHP (for Web servers)**	Winbatch
Frink	Pikt	Windows PowerShell
F-Script	**Python**	**Matlab**

most:

- perform well
- have reasonable run-time support
- can be deployed in an embedded system (or are specifically designed for that purpose)

Summary

- Historical perspective:
  scripting languages have always been popular

- There are many popular ones
  so they must be useful

- ➡ Scripting languages are not for serious programming

- Scripting languages can accelerate change

Scripting is not for serious programming

- Typical applications:
  - Command Line Interpreter (low level / no GUI)
  - GUI activity logging & playback
  - Testing and debugging
    - SW Oscilloscope
    - Insert SW test points
  - SW not worth coding
    - Test SW, factory only SW, R&D SW
  - Customization
    - by customer at the expense of customer support
    - by customer support at the expense of development

Summary

- Historical perspective:
  scripting languages have always been popular

- There are many popular ones
  so they must be useful

- Scripting languages are not for serious programming
  but its applications are serious enough

- Scripting languages can accelerate change

But scripting languages can beat Brooks*

- **Brooks claims**
    - A *product* (more useful than a program costing x€):
        - can be run, tested, repaired by anyone
        - usable in many environments on many sets of data.
        - must be tested
        - needs documentation
    - Brooks estimates a **cost increase** to 3x€.
    - To be a component in a *programming system* (collection of interacting programs like an OS):
        - input and output must conform in syntax, semantics to defined interfaces
        - must operate within resource budget
        - must be tested with other components to check integration (very expensive since interactions grows exponentially in n).
    - Brooks estimates that this too costs **3x**.

So same functionality, cost increases to 9x€

*) Frederick P. Brooks, Jr, the mythical man-month, ISBN 0-201-00650-2, Addison-Wesley, 1975

Scripting languages can reduce factors ‘x’ and ‘3’ in 3x

- Perhaps 3 → 2.5,
  x’ → x*0.72
  - total gain 9 → 4.5 or 50% cost reduction
- How?
  - Easier/quicker integration and testing
    - No lengthy compiles and builds
    - any configuration will run
    - scripting (develoipment/debugging) can run in parallel to production environment with minimal disturbance
  - No need for 100% defined interfaces
    - excess parameters are ignored
  - Tolerant to simple failures
    - interpreter keeps running at all times
  - Easier to add people to the project
    - real programmers are hard to find
  - Fewer people involved
    - less complex communication
  - Allows for experiments
    - the best solution can persist
- So less cost, faster delivery of functionality

Statements (True/False):

Scripting languages can accelerate change*!*

- Use scripting languages whenever you can
  - interface at the highest possible level (‘magnification’) or hide lower levels (‘lens actuator x’)
- Use coding standards to avoid known pitfalls
  - works for C/C#/Java..., so why not for Perl/Python/...?
- Plan to throw one away (you will anyhow...)
  - code in Perl/Python/...
  - refactor once in Perl/Python/...
  - only if result is OK and business case is solid then cast in C/C#/Java... stone
- Customers/Customer support can
  - can debug themselves
  - can make repairs or workarounds themselves
  - can insert better solutions that developers can
    - product becomes more fun to use
    - no more need to wait for patches from vendor

Final Summary

- Historical perspective:
  scripting languages have always been popular

- There are many popular ones
  so they must be useful

- Scripting languages are not for serious programming
  but its applications are serious enough

- Scripting languages can accelerate change
  and beat Brooks's mythical man-month

Next steps

- Please consider why you agree / disagree with these statements → → discussion...
  - *Scripting languages have serious applications*
  - *Scripting languages have few disadvantages*
  - *Scripting languages can accelerate change and beat Brooks's mythical man-month*