Moving VistA Forward – a Modest Proposal

K.S. Bhaskar
Development Director, FIS
ks.bhaskar@fisglobal.com
+1 (610) 578-4265

Major Take-Away

- Horizontal vs. Vertical Re-engineering

Challenges Faced By A MUMPS Application

- Expert friendly
  - Dated coding style
  - Arcane rules, sometimes violated, exceptions not well documented
  - Human interaction not always well separated from business logic
    - User interface technology changes faster than business logic
- Other management concerns
  - Ability to staff
    - New programmers not always willing to invest time & energy in learning
- Relentless promotion by big-brand RDBMS vendors
  - Long-term viability of M technology questioned

FIS

Challenges Faced By A MUMPS Application

- Expert friendly
  - Dated coding style
  - Arcane rules, sometimes violated, exceptions not well documented
  - Human interaction not always well separated from business logic
    - User interface technology changes faster than business logic
- Other management concerns
  - Ability to staff
    - New programmers not always willing to invest time & energy in learning
- Relentless promotion by big-brand RDBMS vendors
  - Long-term viability of M technology questioned

And I'm not talking about VistA!!!

FIS

FIS Profile in the mid 1990s

- Similar concerns to those that apply to VistA today
  - Terminal based primary user interface – mostly screens & menus
  - Thick client written in PowerBuilder for most common users
    - No thin client interface & web was just taking off
  - Thousands of routines of hand-coded M
    - Mostly well written – but not always
    - Well-defined schema – by-passed occasionally
    - Without transaction fences, application Consistency not assured after a crash

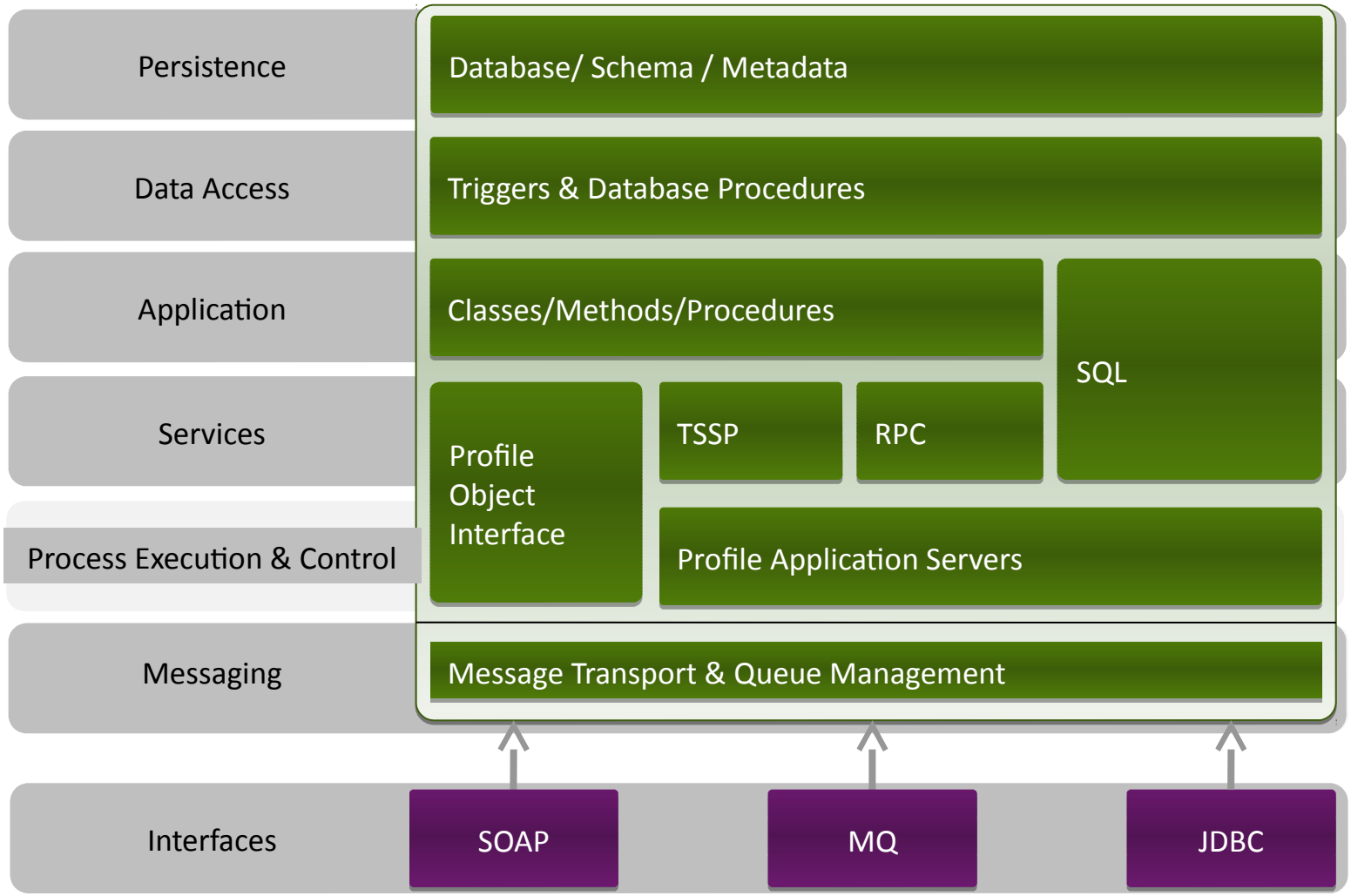
```
N COUNTER
I '$G(DATE) S DATE=$S($G(TJD):$D(TJD),$G(^CUVAR(2)):1)
I '$G(NBD) S NBD=1
I $G(CAL)="" S CAL="IBS"
S COUNTER=NBD
F  D  Q:'COUNTER
.    S DATE=DATE-1
.    I $$BD(DATE,CAL) S COUNTER=COUNTER-1
Q DATE
```

FIS Profile today

- Runs three largest real-time core banking systems in the world
  - (If you know of any bigger, please do let me know)
- Stateless server processes receive and respond to messages
  - Service requests are TCP/IP messages ("transfer $100 from checking to savings"), independent of UI technology
    - New UI technologies can be added with little to no change to servers
  - Each request processed as an ACID transaction
    - Database recovered after crash guaranteed to be application Consistent
    - Maintaining Consistency with Logical Multi-Site application deployments for business continuity are easier if all updates are wrapped in transactions
    - M Locks no longer needed
    - (Transactions work so well, they are even used for "batch" processing)
  - SQL/JDBC access to application
  - Deployable on big-brand RDBMS
    - No technical obstacle preventing Profile from running on Java or other languages (will only require retargeting code generator of PSL compiler)

FIS

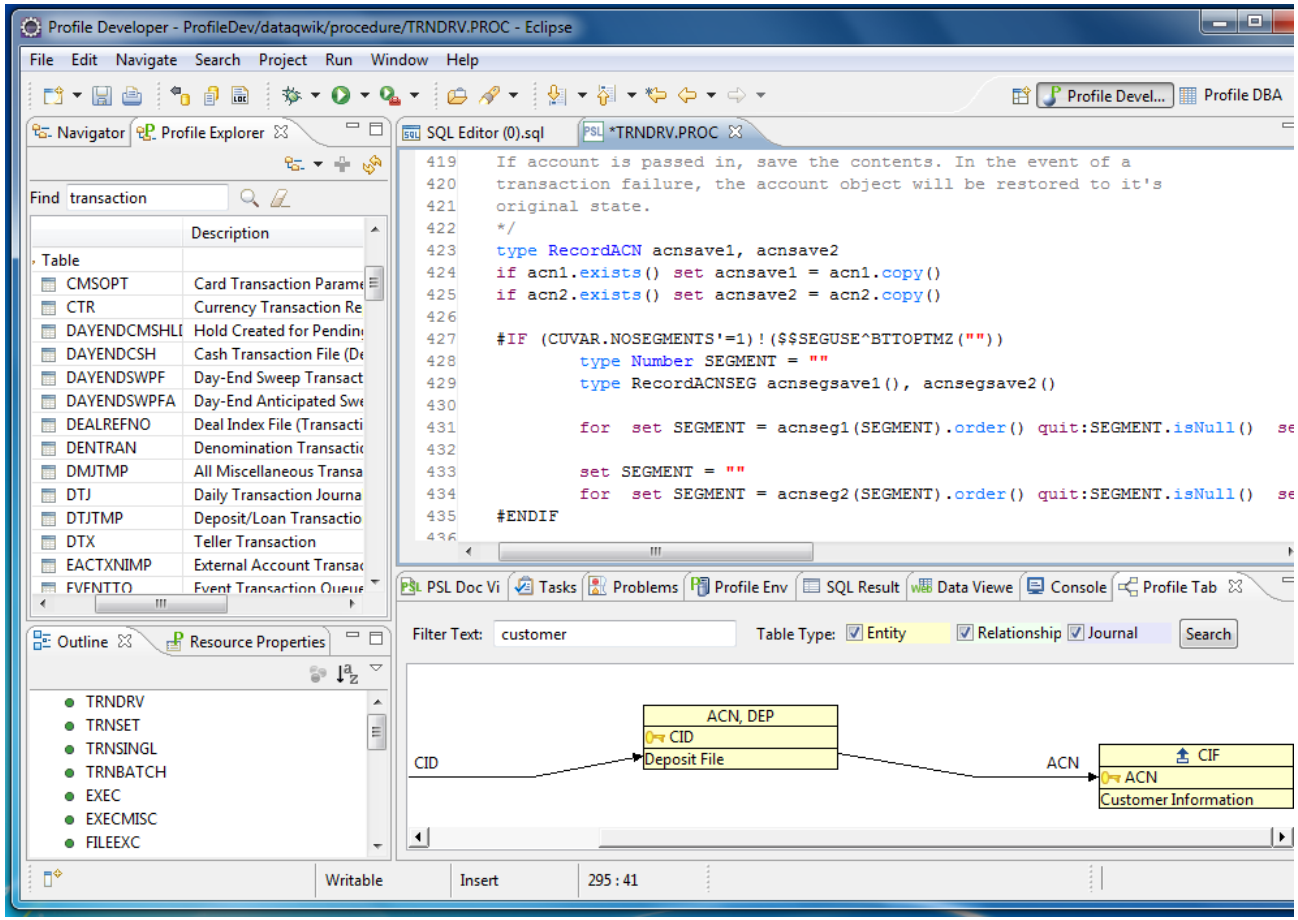
Profile Architecture



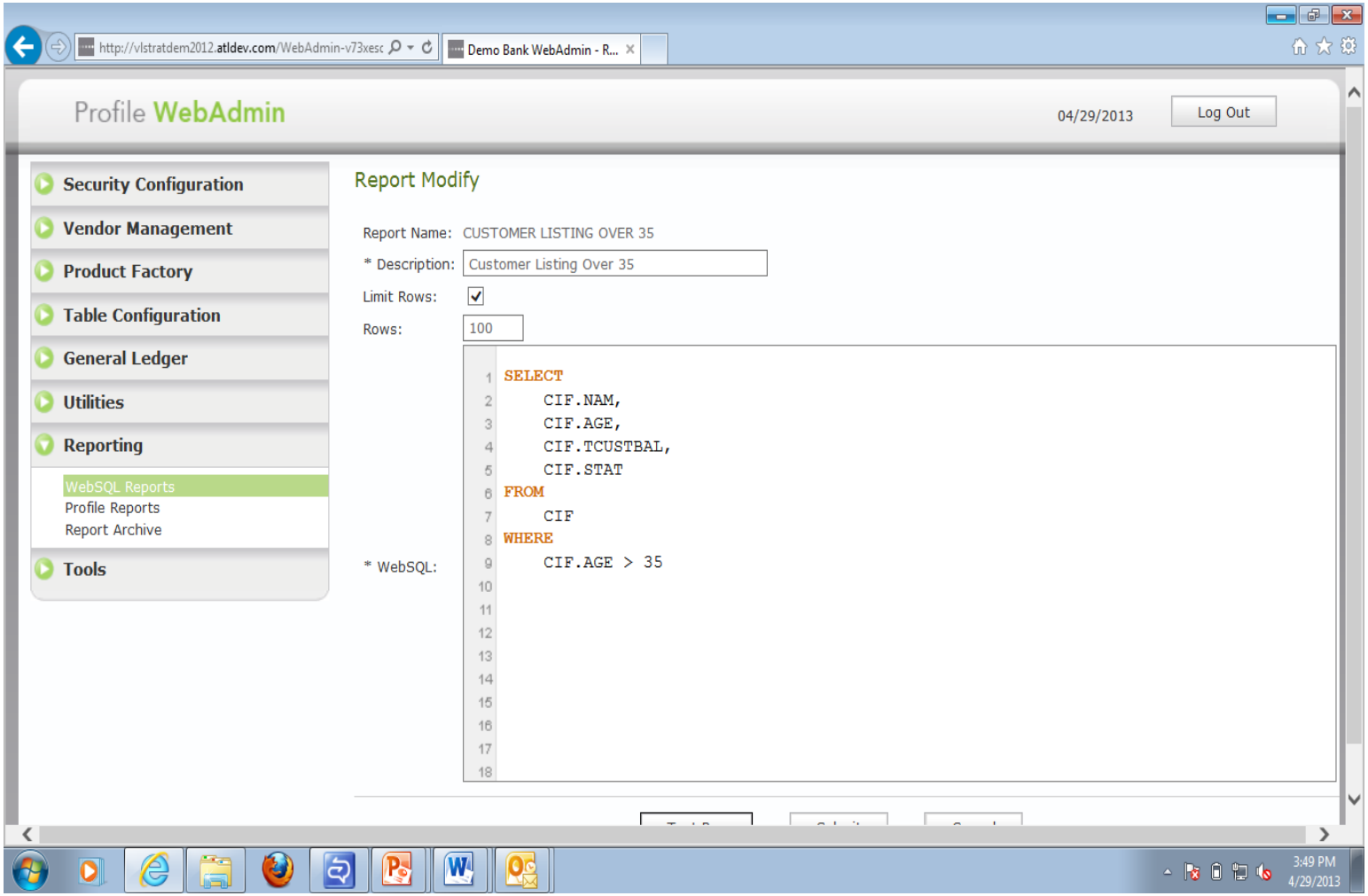
Profile Scripting Language (PSL)

```
type String COUNTER                        // Type checking done by compiler

if 'DATE.get() set DATE = %SystemDate  // Object.method notation enforces schema
if 'NBD.get() set NBD = 1              // and allows compiler to generate code
if CAL.get() = "" set CAL = "IBS"      // for multiple targets

set COUNTER=NBD

for  do { quit:'COUNTER                // Block structure and flexible whitespacing
    set DATE = DATE-1                  // can make code easier to read & maintain
    if $$BD(DATE,CAL) set COUNTER = COUNTER-1
    }

return DATE                            // DNA shared with M
```

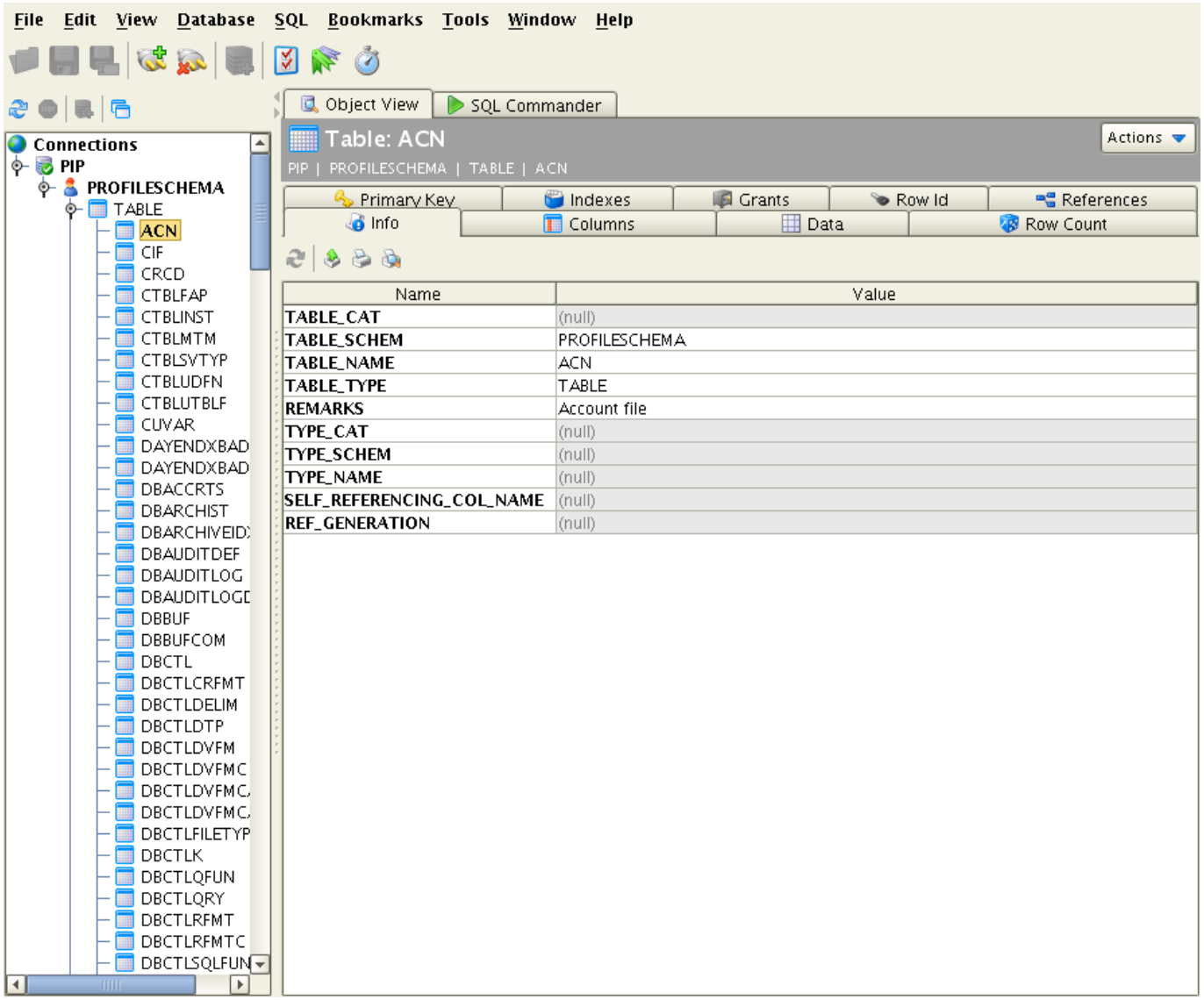
Profile IDE


Profile Developer - ProfileDev/dataqwik/procedure/TRNDRV.PROC - Eclipse
File Edit Navigate Search Project Run Window Help
Profile Devel... Profile DBA
Navigator Profile Explorer
SQL Editor (0).sql *TRNDRV.PROC
Find transaction
Description
Table
CMSOPT Card Transaction Parame
CTR Currency Transaction Re
DAYENDCMSHLI Hold Created for Pendin
DAYENDCSH Cash Transaction File (De
DAYENDSWPF Day-End Sweep Transact
DAYENDSWPFA Day-End Anticipated Swe
DEALREFNO Deal Index File (Transacti
DENTRAN Denomination Transactio
DMJTMP All Miscellaneous Transa
DTJ Daily Transaction Journa
DTJTMP Deposit/Loan Transactio
DTX Teller Transaction
EACTXNIMP External Account Transac
EVENTTO Event Transaction Queue
419 If account is passed in, save the contents. In the event of a
420 transaction failure, the account object will be restored to it's
421 original state.
422 */
423 type RecordACN acnsave1, acnsave2
424 if acn1.exists() set acnsave1 = acn1.copy()
425 if acn2.exists() set acnsave2 = acn2.copy()
426
427 #IF (CUVAR.NOSEGMENTS'=1)!($$SEGUSE^BTTOPTMZ(""))
428 type Number SEGMENT = ""
429 type RecordACNSEG acnsegsave1(), acnsegsave2()
430
431 for set SEGMENT = acnseg1(SEGMENT).order() quit:SEGMENT.isNull()
432
433 set SEGMENT = ""
434 for set SEGMENT = acnseg2(SEGMENT).order() quit:SEGMENT.isNull()
435 #ENDIF
PSL Doc Vi Tasks Problems Profile Env SQL Result Data Viewe Console Profile Tab
Filter Text: customer
Table Type: Entity Relationship Journal
Search
Outline Resource Properties
TRNDRV
TRNSET
TRNSINGL
TRNBATCH
EXEC
EXECMISC
FILEEXC
ACN, DEP
CID
Deposit File
CID
ACN
CIF
ACN
Customer Information
Writable
Insert
295 : 41

Profile Ad Hoc SQL Reporting



Access from DBVisualizer



How did Profile get here from there?

- No magic… just innumerable cycles of recoding / rearchitecting

What does this imply for VistA?

- Revolution through evolution
  - Changes over a half dozen major versions over as many years
  - Pick your pithy saying…
    - Incremental changes are safer than big bangs
    - Many small steps can make a big journey
    - Patience is not the same as paralysis
    - Eat the elephant one bite at a time
    - Spread the cost over several budget years
    - No “none of it is done till it's all done” malady
    - Confidence in code comes from using it
    - You can wait forever for the perfect solution
    - The longest journey is the one that is never begun

What does this imply for VistA?

- Revolution through evolution
- Avoid encapsulation
  - Encapsulation creates logic that resists change
    - No knowledgeable programmers, changes perceived as high risk
  - Unchanging code leads to ossification and fossilization
  - And in turn, premature application obsolescence

FIS

What does this imply for VistA?

- Revolution through evolution
- Avoid encapsulation
- Choose a language and compile it into M
  - Choice of language not important as long as it's "good enough"
    - We created PSL (now free / open source software – http://fis-pip.com)
    - JavaScript, PHP, Python, and Ruby will all probably work
    - Java and .Net have fans and detractors
    - Proprietary languages don't match free / open source ethos of VistA
  - Choose a well defined subset
    - Every language has its idiosyncrasies and this is not a good time to explore them
  - Importance of compiling into M cannot be emphasized enough
    - Granularity of change is the line, not the module
    - New code and old code can share local variables and other context
    - Code dynamically generated from templates in the database continues to work
    - Facilitates some conversion automation – see a pattern and convert it everywhere
  - Enforce coding discipline with new code, e.g., direct global accesses
    - Mandatory for application deployment on non-M databases

FIS

What does this imply for VistA?

- Revolution through evolution
- Avoid encapsulation
- Choose a language and compile it into M
- Re-architect as you go
  - Some changes may be "lumpy" - requiring all of the application to do something the new way, e.g., generating code for an RDBMS target required zero direct M global access and 100% access through schema layer
  - Other changes are not, e.g., use of transaction processing

FIS

What does this imply for VistA?

- Revolution through evolution
- Avoid encapsulation
- Choose a language and compile it into M
- Re-architect as you go
- Integration beats Balkanization
  - It's tempting and sometimes expedient to replace pieces with "best of breed" applications
  - Breaking off pieces of core business logic converts an integrated patient/customer-centric architecture into a stovepipe architecture – see http://sourcemaking.com/antipatterns/software-architecture-antipatterns
  - That said, sometimes slicing off functionality is the right thing to do, e.g., UI

FIS

What does this imply for VistA?

- Revolution through evolution
- Avoid encapsulation
- Choose a language and compile it into M
- Re-architect as you go
- Integration beats Balkanization
- The value of M technology is as an execution engine
  - Leverage M for what it does well – compact & efficient M code scales up to the needs of the largest enterprises
    - "Indistinguishable from line noise" doesn't matter for code generated by a compiler
  - M does not need to be the next "cool" programming language
  - Establishing value is the key to ongoing investment
    - Compiling PSL to run Profile on a big-brand RDBMS and generating Java from PSL established the importance of GT.M in a way that no amount of white papers or slide decks ever could have done

FIS

What does this imply for VistA?

- Revolution through evolution
- Avoid encapsulation
- Choose a language and compile it into M
- Re-architect as you go
- Integration beats Balkanization
- The value of M technology is as an execution engine

Horizontal vs. Vertical Re-engineering!!!

FIS

Links

- FIS GT.M: http://fis-gtm.com
- FIS Profile: http://fis-profile.com
- FIS PIP: http://fis-pip.com
- Anti-patterns: http://sourcemaking.com/antipatterns/software-architecture-antipatterns
- K.S. Bhaskar / ks.bhaskar@fisglobal.com / +1 (610) 578-4265