Legacy Modernization to SOA using Compass/VB

Case Study

Avi Yaeli, Netta Aizenbud, Jonathan Bnayahu, Nurit Dor, Alex Akilov, Sara Porat
IBM Research Labs in Haifa
Jenny Choy, IGS

Oct. 27, 2005

Agenda

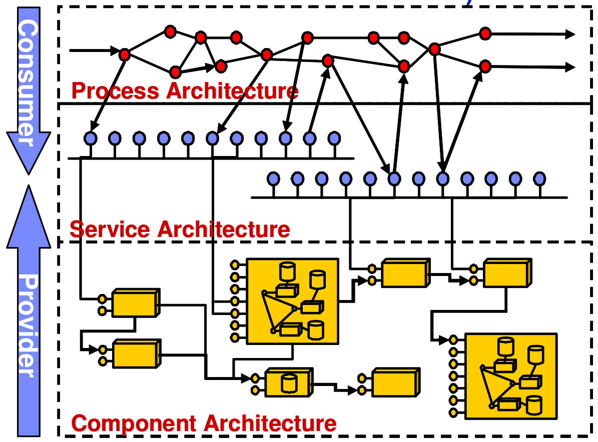
- Service Oriented Architecture (SOA) Overview
- Legacy Modernization to SOA
- The LNAB Engagement
- Summary and Conclusions

What is Service Oriented Architecture (SOA)?

- Service Oriented Architecture is an approach that considers all business systems, applications and functions as resources (i.e. services)
- A "service oriented architecture" is not one component or technology, it is an approach to designing an enterprise architecture consisting of multiple infrastructure and application components
  - It does not mandate building everything from scratch
  - It does not mean that it will be more expensive to implement.
    - Development costs will be lower over time due to enforcement of a new level of reuse (process, interfaces and components)
  - It increases flexibility due to loosely coupling of resources (services), allowing plug and play of underlying technologies and individual components (with multiple vendor and best of breed solutions)
  - It supports faster time to market

What is Service Oriented Architecture (SOA)? – cont.

- These resources (services) can be:
  - A function that needs to be used by more than one system
  - An entire application
  - A particular product service (e.g. a scoring service)
  - A particular common utility (e.g. assembling a document, printing a document)
  - An external web application (e.g. a credit check transaction to the credit bureau)
  - A host transaction (e.g. IMS) via web services



- Web Services are not a single technology but a set of capabilities defined by open standards that can be used to construct architectures or applications. *Web Services is one instance of an implementation of a SOA*

Legacy Modernization to SOA

- Motivation
  - Increased maintenance cost of legacy systems
  - Legacy applications are usually reliable, efficient and optimized
    - But are often monolithic, intertwined, complex and inflexible
  - Domain expert knowledge of the system is lost with staff turnover
    - New legacy skills are hard to find
  - Replacing a legacy systems is a huge effort and often not feasible
    - High costs involved
    - The requirements are often not available or not updated
      - The business rules are often buried in the code
  - IT managers often need to find ways to leverage their existing investments and make incremental changes that bring immediate value to the business

Legacy Modernization to SOA – cont.

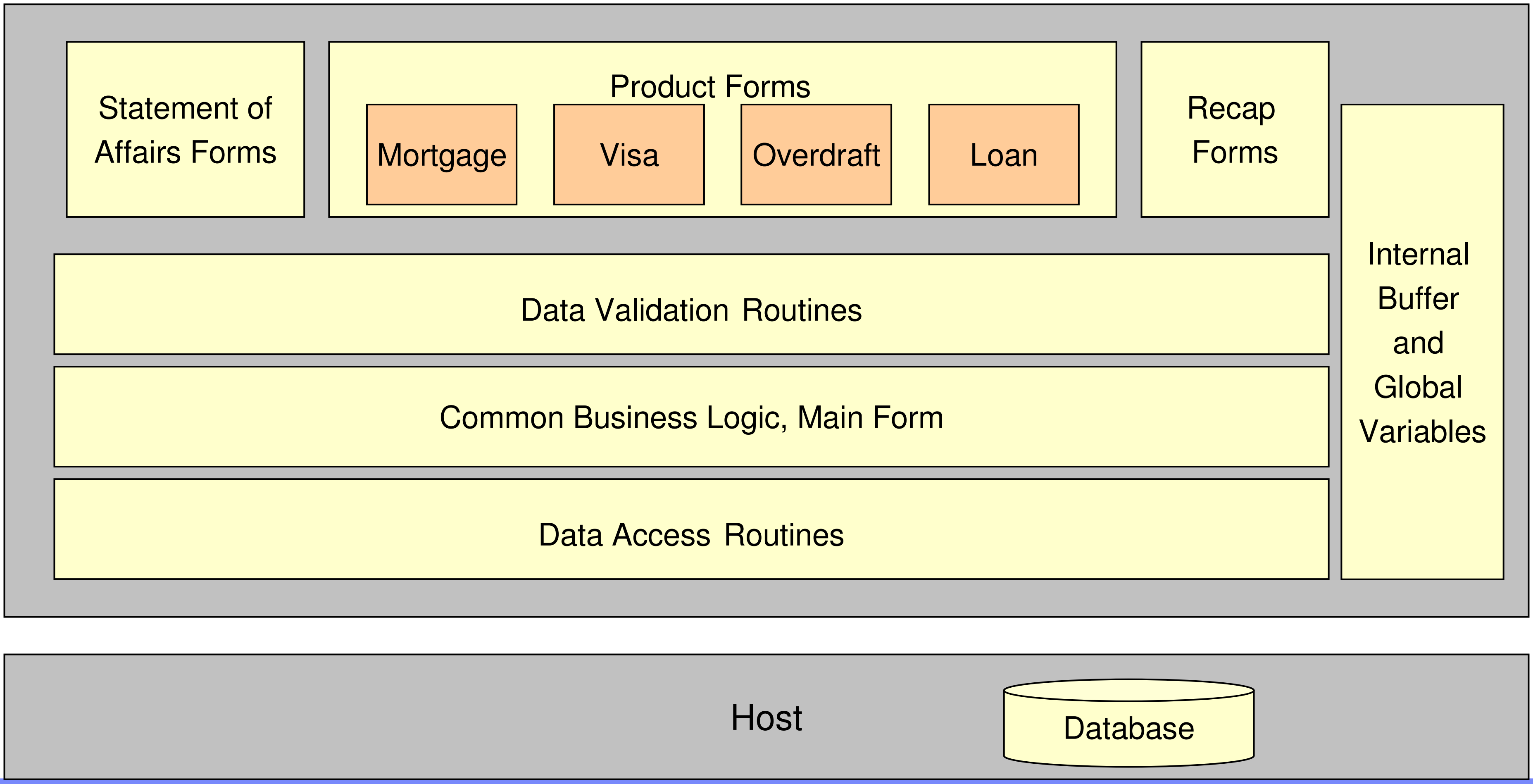
- The Benefits of Moving to SOA
  - Reuse legacy applications and data in new services, renewing their value
  - Access these assets in real time to support business intelligence and customer service initiatives
  - Flexibility and interoperability with internal and external services
  - Control budget through isolation and incremental changes
  - Enable programmers with today's skills to work more effectively with yesterdays' systems

Gartner - "a better route, they say, is exposing the business processes in legacy applications, keeping the core of the application intact. That approach allows developers to continue to work with whatever language they're familiar with."

Overview of the LNAB Engagement

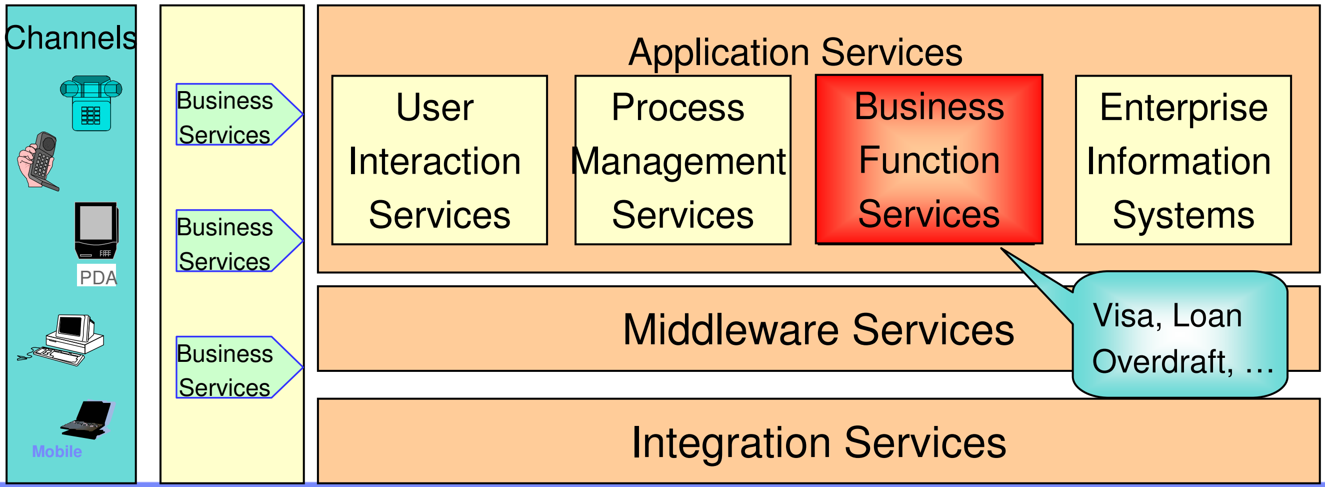
- In 2004 IBM Global Services began a legacy modernization project in a large, North-American bank
  - Bank aimed at renovating its core systems
  - One of the applications that need to be migrated is a VB6 based monolithic application that began as a simple Visa application but evolved over a course of 5 years to a monstrous credit application system
  - Wanted to move to SOA
  - Incremental change, spanning several years

The Legacy Architecture

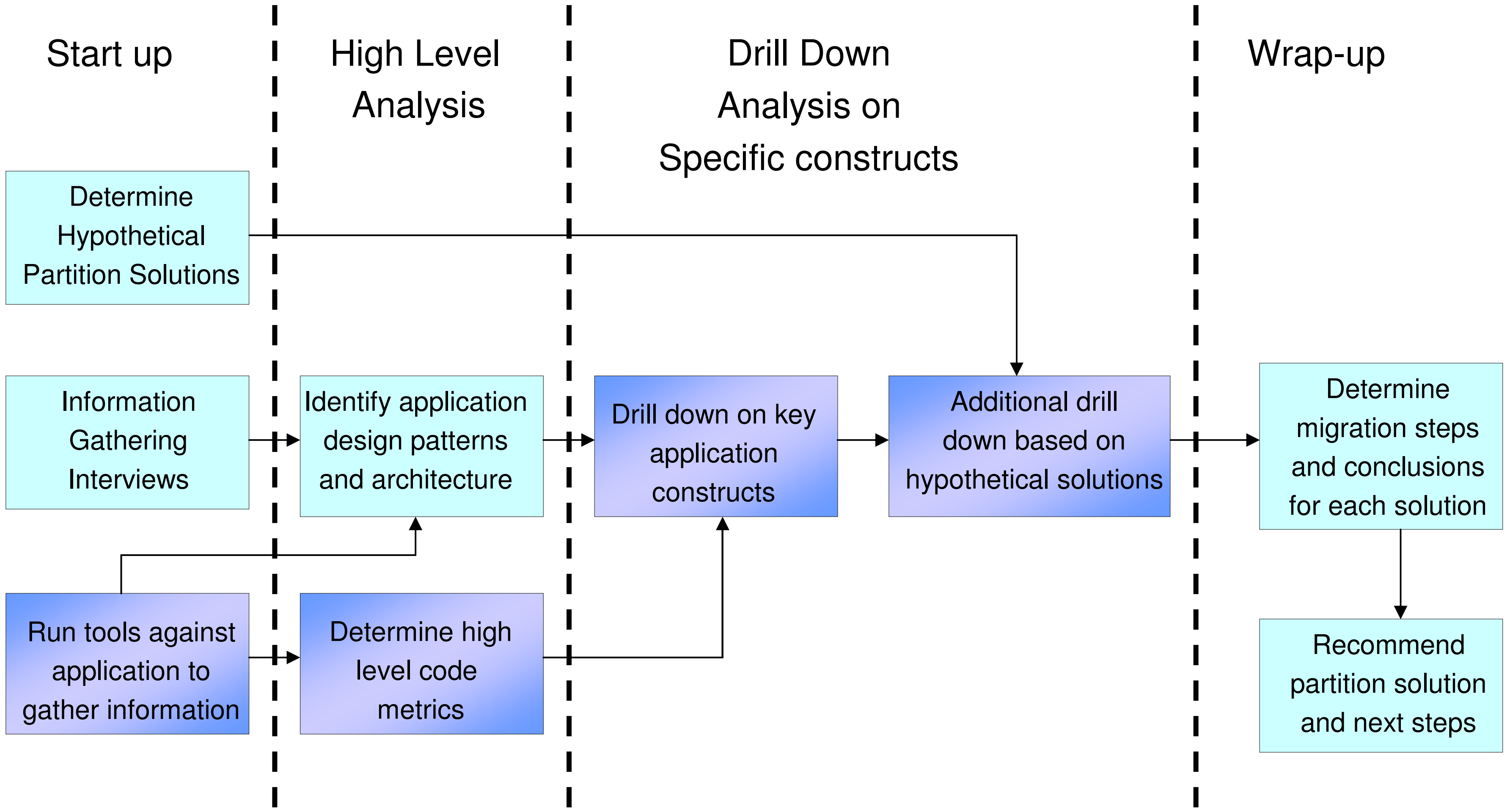


The Target Architecture

- The target vision is a service oriented architecture, enabling channel and product neutrality, common and reusable components, and shared infrastructure
  - Back end uses IBM WebSphere technologies
  - Front end in .NET (web application)
- The conceptual architecture was defined - how to **incrementally** migrate to the new architecture



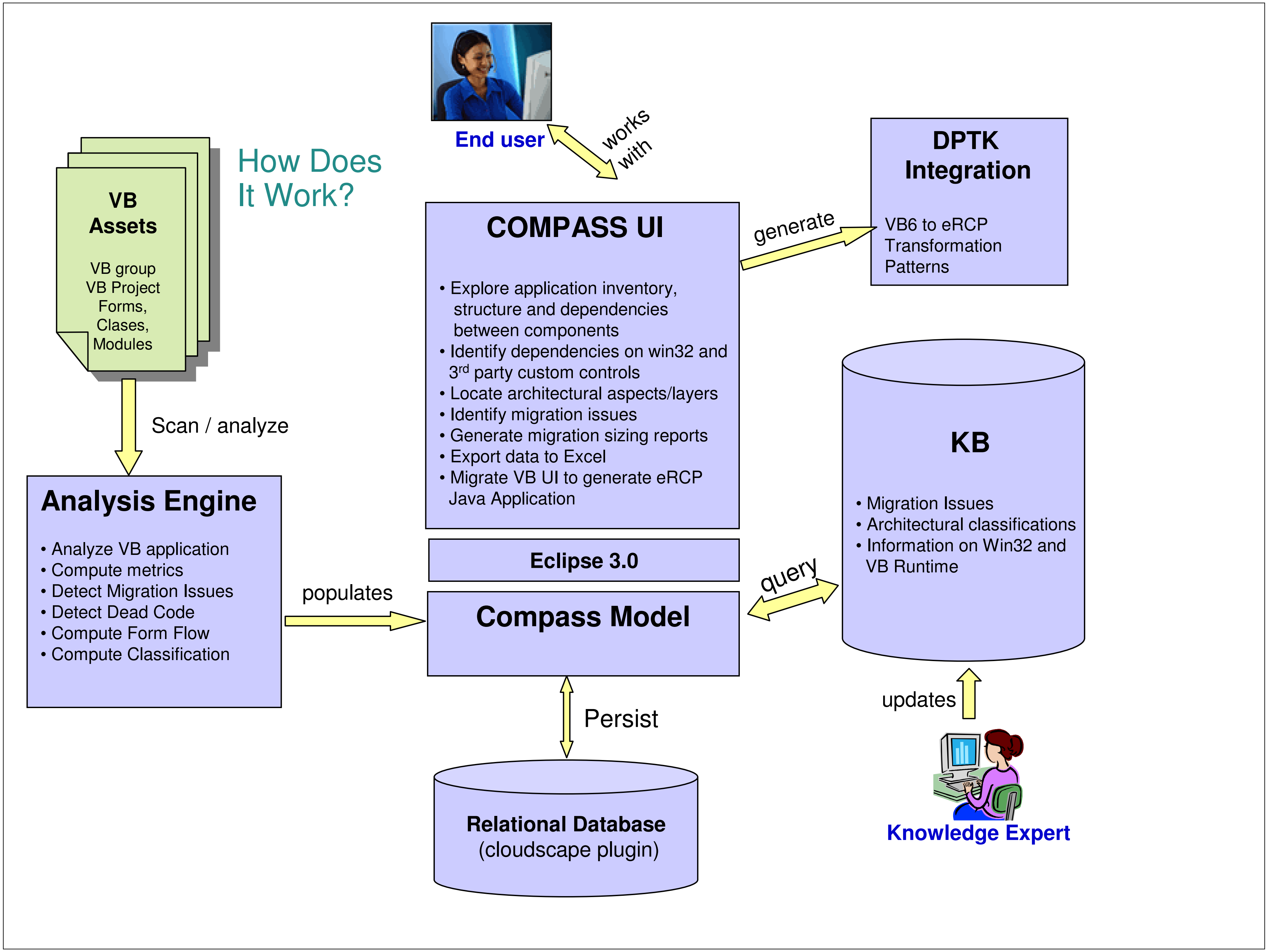
The Assessment Approach


Start up
High Level Analysis
Drill Down Analysis on Specific constructs
Wrap-up
Determine Hypothetical Partition Solutions
Information Gathering Interviews
Identify application design patterns and architecture
Drill down on key application constructs
Additional drill down based on hypothetical solutions
Determine migration steps and conclusions for each solution
Run tools against application to gather information
Determine high level code metrics
Recommend partition solution and next steps

Tooling Support – Compass/VB

- Compass - **Co**de **M**igration **P**lanning and **Ass**essment Workbench
  - Developed as a prototype during 2004
  - Expected to be available publicly through IBM alphaWorks later this year
- Features
  - Understand application inventory, structure, and relationships between components
  - Understand the architecture of the application, layers, flows, and interactions between components
  - Identify obstacles and migration issues
  - Generate assessment reports
  - Help make decisions on how to migrate the application: which parts are translatable, which parts should be rewritten, etc.
- Compass/VB
  - Support for Visual Basic understanding and migration on top of Compass architecture

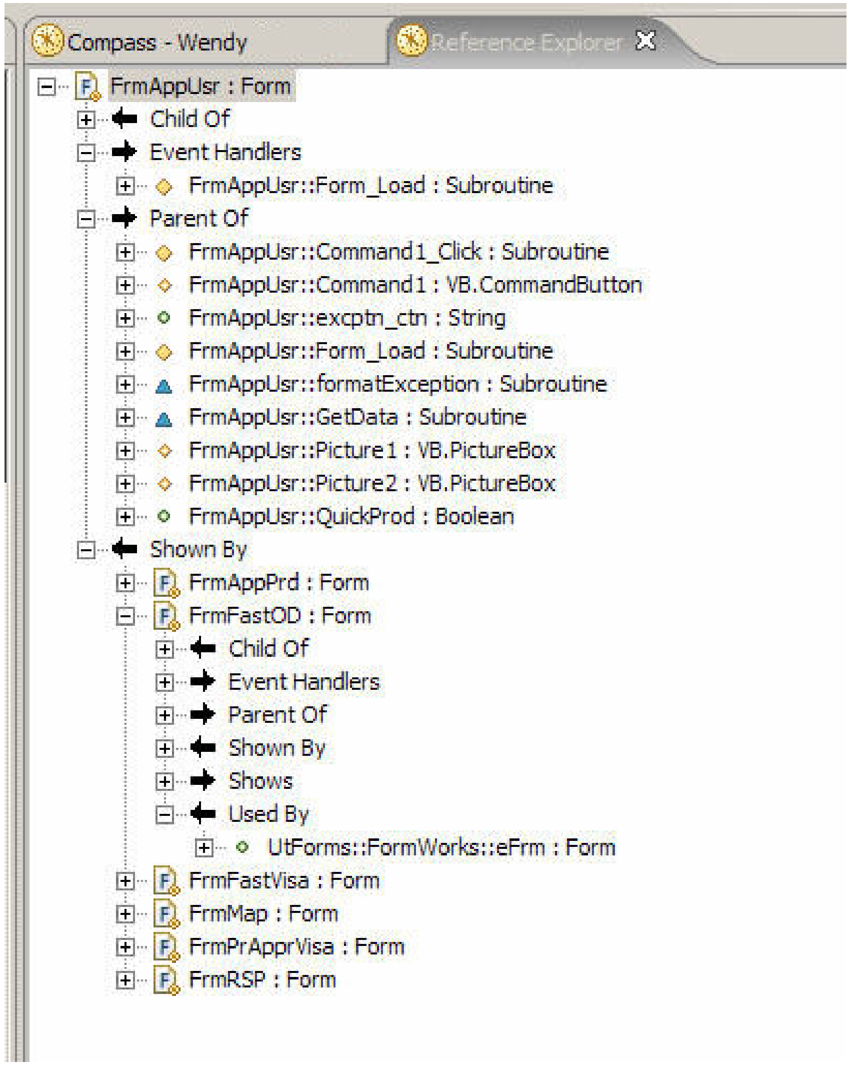
How Does It Work?



End user
works with
VB Assets
VB group
VB Project
Forms,
Clases,
Modules
Scan / analyze
Analysis Engine
• Analyze VB application
• Compute metrics
• Detect Migration Issues
• Detect Dead Code
• Compute Form Flow
• Compute Classification
populates
COMPASS UI
• Explore application inventory, structure and dependencies between components
• Identify dependencies on win32 and 3rd party custom controls
• Locate architectural aspects/layers
• Identify migration issues
• Generate migration sizing reports
• Export data to Excel
• Migrate VB UI to generate eRCP Java Application
Eclipse 3.0
Compass Model
Persist
Relational Database
(cloudscape plugin)
generate
DPTK Integration
VB6 to eRCP Transformation Patterns
KB
• Migration Issues
• Architectural classifications
• Information on Win32 and VB Runtime
query
updates
Knowledge Expert

Compass/VB Reference Explorer

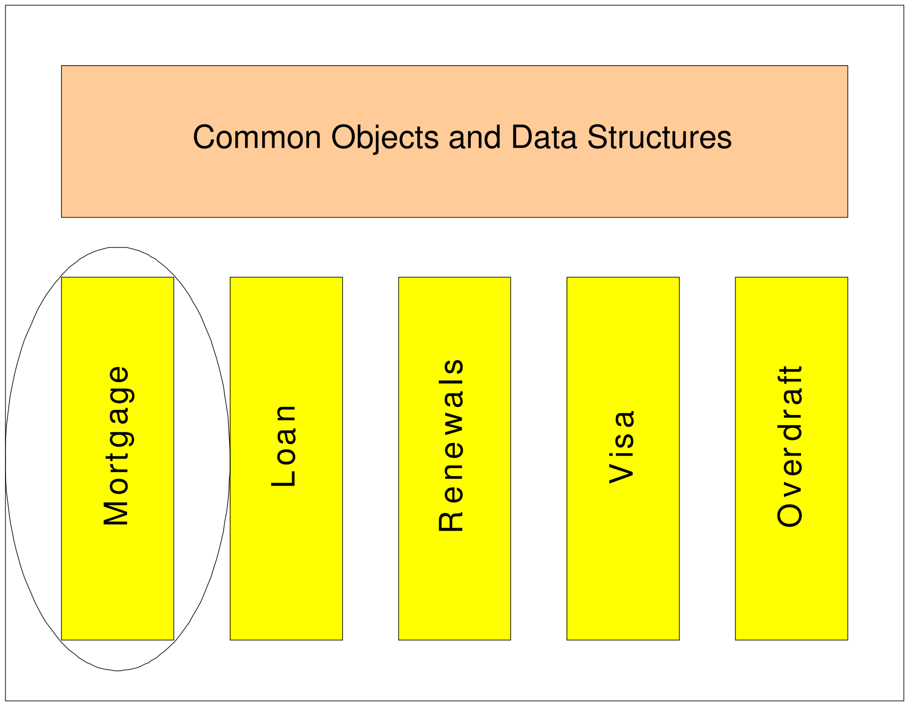
- Tree-based view of all relationships in the model
  - Parent / Child
  - Uses / Used-by
  - Shows / Shown by
- Logical elements
  - Forms
  - Subroutines
  - Controls
  - Variables
  - Types



```
Compass - Wendy    Reference Explorer
FrmAppUsr : Form
  Child Of
  Event Handlers
    FrmAppUsr::Form_Load : Subroutine
  Parent Of
    FrmAppUsr::Command1_Click : Subroutine
    FrmAppUsr::Command1 : VB.CommandButton
    FrmAppUsr::excptn_ctn : String
    FrmAppUsr::Form_Load : Subroutine
    FrmAppUsr::formatException : Subroutine
    FrmAppUsr::GetData : Subroutine
    FrmAppUsr::Picture1 : VB.PictureBox
    FrmAppUsr::Picture2 : VB.PictureBox
    FrmAppUsr::QuickProd : Boolean
  Shown By
    FrmAppPrd : Form
    FrmFastOD : Form
      Child Of
      Event Handlers
      Parent Of
      Shown By
      Shows
      Used By
        UtForms::FormWorks::eFrm : Form
    FrmFastVisa : Form
    FrmMap : Form
    FrmPrApprVisa : Form
    FrmRSP : Form
```

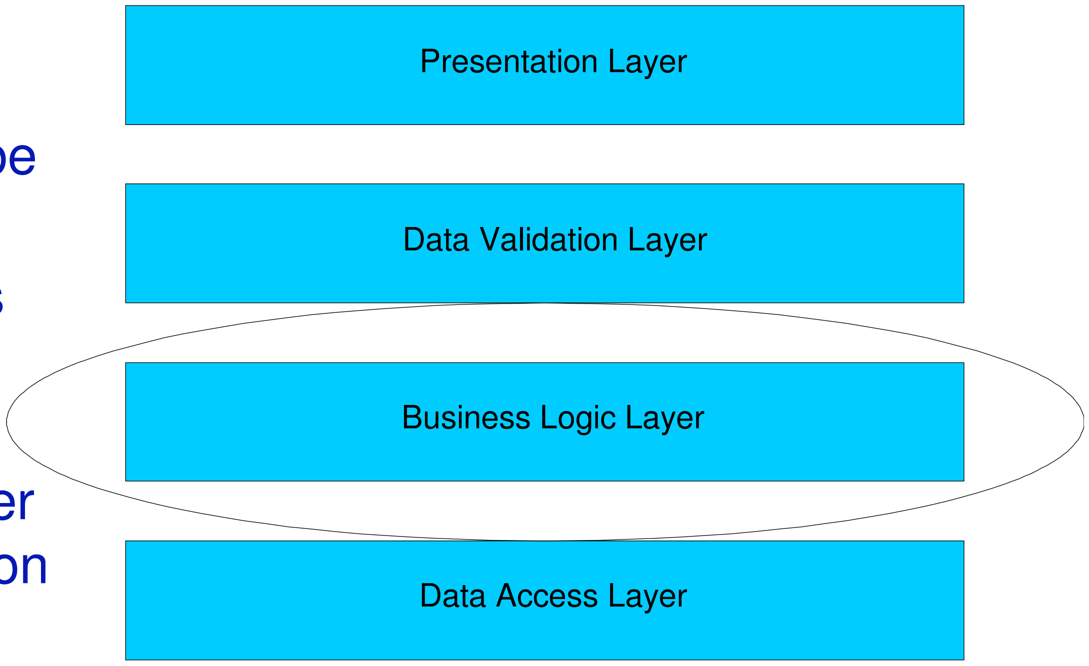
Three Hypothetical Partition Strategies

- Partition by Product
  - Move an existing product out and migrate that to a new front end or application
- Steps:
  1. Identify shared code objects and data structures
  2. Replicate or Partition shared code objects
  3. Create a separate instance of the application to support one of the products



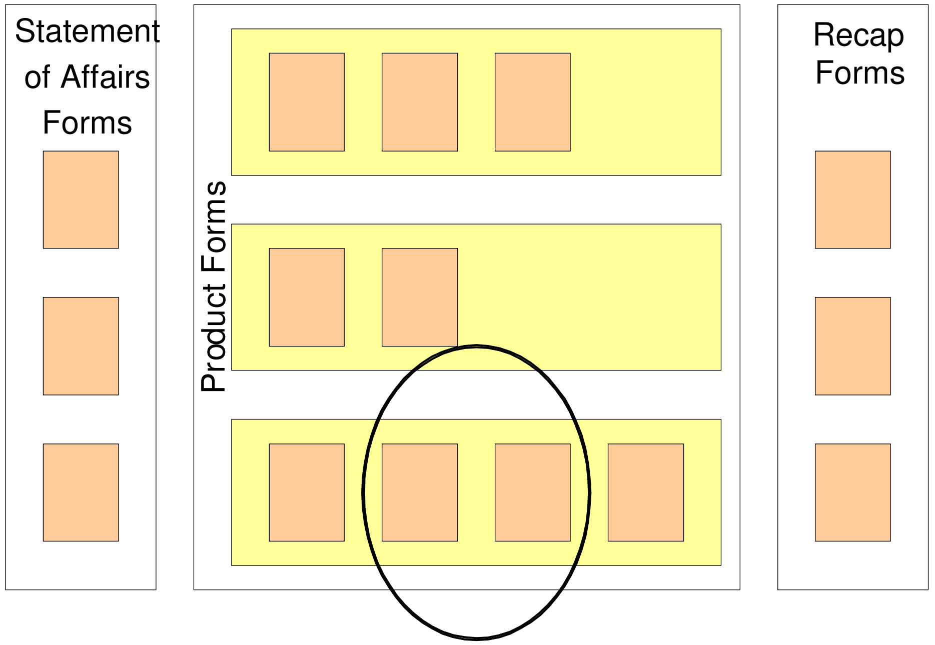
Three Hypothetical Partition Strategies

- Partition by Application Layer
  - Implement application layers in a new technology
- Steps:
  1. Find out what layer can be isolated
  2. Find its integration points with the rest of the application
  3. Build a new strategic layer and change the application to integrate with the new layer



Three Hypothetical Partition Strategies

- Partition by Business Process
  - Move an existing set of forms or business functions out and migrate to the new front end
- Steps:
  1. Identify business processes
  2. Remove or reduce dependencies between processes
  3. Develop new front end for the processes that were moved out
  4. Develop data synchronization between the application and the new front end

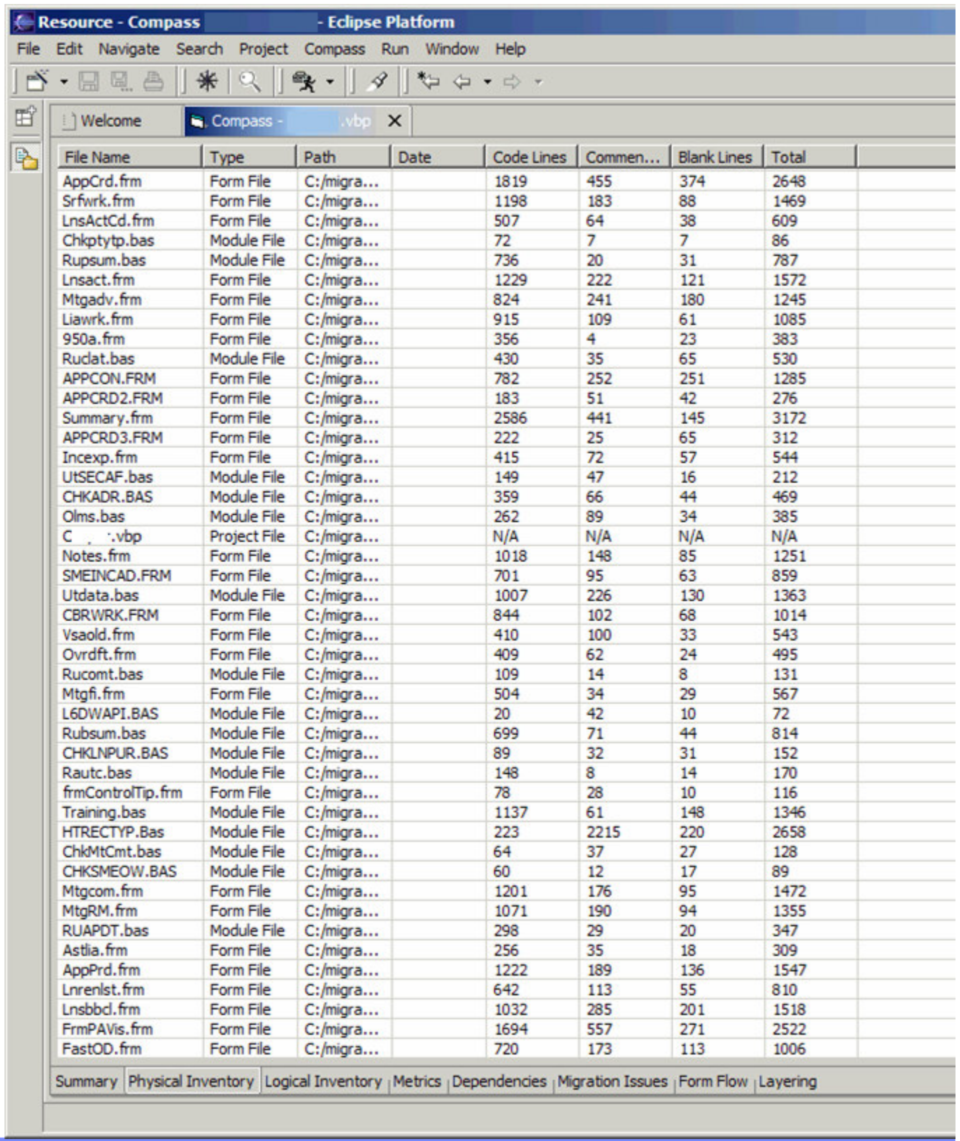


Start Up - Information Gathered through Interviews

- The application is huge
- Over time, the implementation strayed from the original design
  - Application has evolved over the course of several years
  - Original code used design patterns, later additions did not
  - Developers had to bypass the infrastructures in some cases in order to implement new functionality

Start Up - Inventory Information

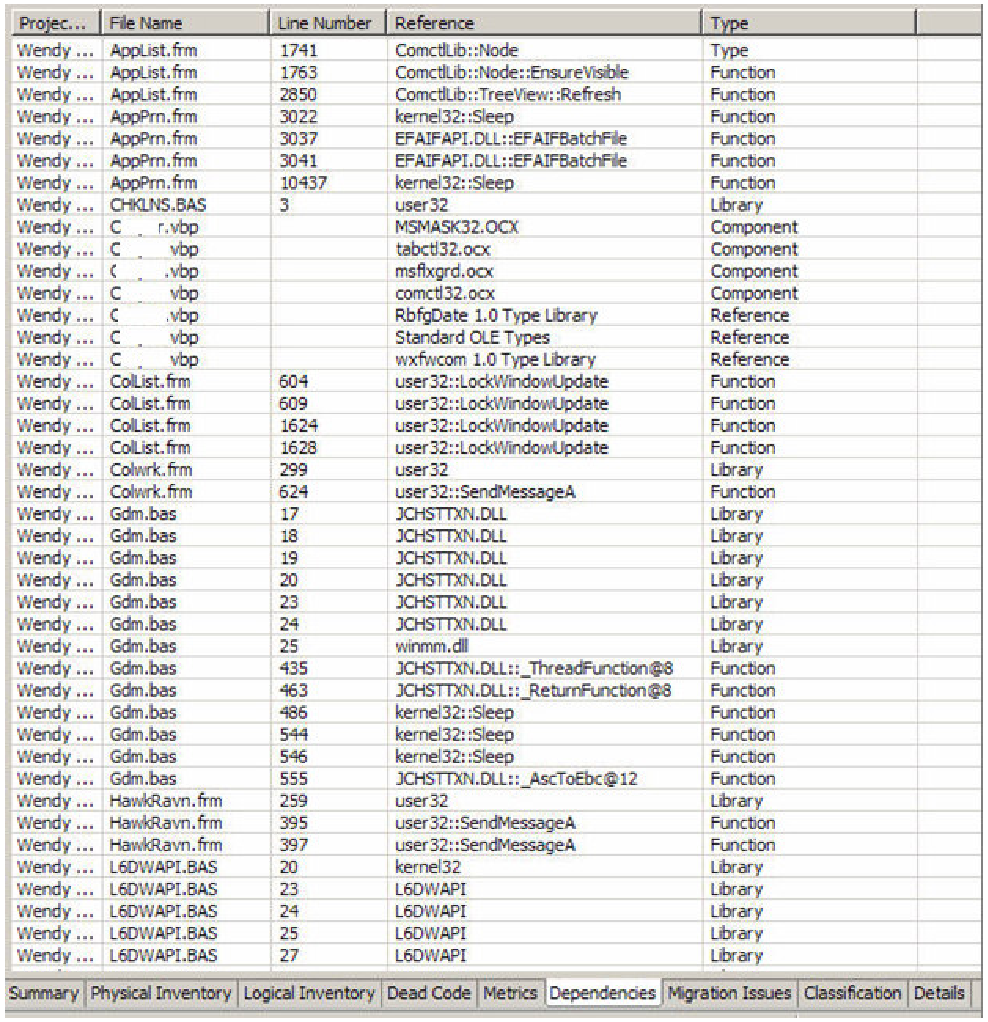
- Physical
  - Number of files (of each type)
  - File sizes
  - Lines of code
- Logical
  - Number of forms
  - Number of controls
  - Number of methods
  - Number of event handlers



File Name	Type	Path	Date	Code Lines	Commen...	Blank Lines	Total
AppCrd.frm	Form File	C:/migra...		1819	455	374	2648
Srfwrk.frm	Form File	C:/migra...		1198	183	88	1469
LnsActCd.frm	Form File	C:/migra...		507	64	38	609
Chkptytp.bas	Module File	C:/migra...		72	7	7	86
Rupsum.bas	Module File	C:/migra...		736	20	31	787
Lnsact.frm	Form File	C:/migra...		1229	222	121	1572
Mtgadv.frm	Form File	C:/migra...		824	241	180	1245
Liawrk.frm	Form File	C:/migra...		915	109	61	1085
950a.frm	Form File	C:/migra...		356	4	23	383
Ruclat.bas	Module File	C:/migra...		430	35	65	530
APPCON.FRM	Form File	C:/migra...		782	252	251	1285
APPCRD2.FRM	Form File	C:/migra...		183	51	42	276
Summary.frm	Form File	C:/migra...		2586	441	145	3172
APPCRD3.FRM	Form File	C:/migra...		222	25	65	312
Incexp.frm	Form File	C:/migra...		415	72	57	544
UtSECAF.bas	Module File	C:/migra...		149	47	16	212
CHKADR.BAS	Module File	C:/migra...		359	66	44	469
Olms.bas	Module File	C:/migra...		262	89	34	385
C .vbp	Project File	C:/migra...		N/A	N/A	N/A	N/A
Notes.frm	Form File	C:/migra...		1018	148	85	1251
SMEINCAD.FRM	Form File	C:/migra...		701	95	63	859
Utdata.bas	Module File	C:/migra...		1007	226	130	1363
CBRWRK.FRM	Form File	C:/migra...		844	102	68	1014
Vsaold.frm	Form File	C:/migra...		410	100	33	543
Ovrdft.frm	Form File	C:/migra...		409	62	24	495
Rucomt.bas	Module File	C:/migra...		109	14	8	131
Mtgfi.frm	Form File	C:/migra...		504	34	29	567
L6DWAPI.BAS	Module File	C:/migra...		20	42	10	72
Rubsum.bas	Module File	C:/migra...		699	71	44	814
CHKLNPUR.BAS	Module File	C:/migra...		89	32	31	152
Rautc.bas	Module File	C:/migra...		148	8	14	170
frmControlTip.frm	Form File	C:/migra...		78	28	10	116
Training.bas	Module File	C:/migra...		1137	61	148	1346
HTRECTYP.Bas	Module File	C:/migra...		223	2215	220	2658
ChkMtCmt.bas	Module File	C:/migra...		64	37	27	128
CHKSMEOW.BAS	Module File	C:/migra...		60	12	17	89
Mtgcom.frm	Form File	C:/migra...		1201	176	95	1472
MtgRM.frm	Form File	C:/migra...		1071	190	94	1355
RUAPDT.bas	Module File	C:/migra...		298	29	20	347
Astlia.frm	Form File	C:/migra...		256	35	18	309
AppPrd.frm	Form File	C:/migra...		1222	189	136	1547
Lnrenlst.frm	Form File	C:/migra...		642	113	55	810
Lnsbbcl.frm	Form File	C:/migra...		1032	285	201	1518
FrmPAVis.frm	Form File	C:/migra...		1694	557	271	2522
FastOD.frm	Form File	C:/migra...		720	173	113	1006

Start Up - Dependencies and Classification

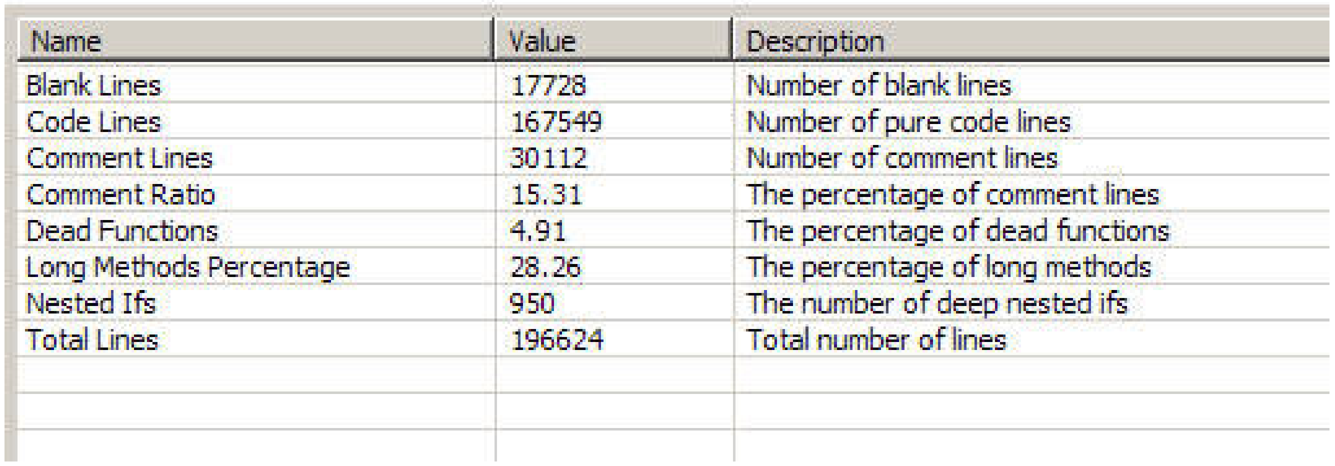
- Dependencies
  - Calls to the host system
  - Dependencies on the platform
    - Windows API calls
  - Usage of VB external libraries
- Classification – categorize code elements according to their functionality
  - UI
  - Database
  - OLE
  - I/O



Projec...	File Name	Line Number	Reference	Type
Wendy ...	AppList.frm	1741	ComctlLib::Node	Type
Wendy ...	AppList.frm	1763	ComctlLib::Node::EnsureVisible	Function
Wendy ...	AppList.frm	2850	ComctlLib::TreeView::Refresh	Function
Wendy ...	AppPrn.frm	3022	kernel32::Sleep	Function
Wendy ...	AppPrn.frm	3037	EFAIFAPI.DLL::EFAIFBatchFile	Function
Wendy ...	AppPrn.frm	3041	EFAIFAPI.DLL::EFAIFBatchFile	Function
Wendy ...	AppPrn.frm	10437	kernel32::Sleep	Function
Wendy ...	CHKLNS.BAS	3	user32	Library
Wendy ...	C r.vbp		MSMASK32.OCX	Component
Wendy ...	C vbp		tabctl32.ocx	Component
Wendy ...	C .vbp		msflxgrd.ocx	Component
Wendy ...	C vbp		comctl32.ocx	Component
Wendy ...	C .vbp		RbfgDate 1.0 Type Library	Reference
Wendy ...	C vbp		Standard OLE Types	Reference
Wendy ...	C vbp		wxfwcom 1.0 Type Library	Reference
Wendy ...	ColList.frm	604	user32::LockWindowUpdate	Function
Wendy ...	ColList.frm	609	user32::LockWindowUpdate	Function
Wendy ...	ColList.frm	1624	user32::LockWindowUpdate	Function
Wendy ...	ColList.frm	1628	user32::LockWindowUpdate	Function
Wendy ...	Colwrk.frm	299	user32	Library
Wendy ...	Colwrk.frm	624	user32::SendMessageA	Function
Wendy ...	Gdm.bas	17	JCHSTTXN.DLL	Library
Wendy ...	Gdm.bas	18	JCHSTTXN.DLL	Library
Wendy ...	Gdm.bas	19	JCHSTTXN.DLL	Library
Wendy ...	Gdm.bas	20	JCHSTTXN.DLL	Library
Wendy ...	Gdm.bas	23	JCHSTTXN.DLL	Library
Wendy ...	Gdm.bas	24	JCHSTTXN.DLL	Library
Wendy ...	Gdm.bas	25	winmm.dll	Library
Wendy ...	Gdm.bas	435	JCHSTTXN.DLL::_ThreadFunction@8	Function
Wendy ...	Gdm.bas	463	JCHSTTXN.DLL::_ReturnFunction@8	Function
Wendy ...	Gdm.bas	486	kernel32::Sleep	Function
Wendy ...	Gdm.bas	544	kernel32::Sleep	Function
Wendy ...	Gdm.bas	546	kernel32::Sleep	Function
Wendy ...	Gdm.bas	555	JCHSTTXN.DLL::_AscToEbc@12	Function
Wendy ...	HawkRavn.frm	259	user32	Library
Wendy ...	HawkRavn.frm	395	user32::SendMessageA	Function
Wendy ...	HawkRavn.frm	397	user32::SendMessageA	Function
Wendy ...	L6DWAPI.BAS	20	kernel32	Library
Wendy ...	L6DWAPI.BAS	23	L6DWAPI	Library
Wendy ...	L6DWAPI.BAS	24	L6DWAPI	Library
Wendy ...	L6DWAPI.BAS	25	L6DWAPI	Library
Wendy ...	L6DWAPI.BAS	27	L6DWAPI	Library

High Level Analysis - Code Complexity

- Metrics
  - Comment Ratio
  - Long Method %
  - Nested Conditionals
- Inspect for various levels: group of applications, application, class, form



Name	Value	Description
Blank Lines	17728	Number of blank lines
Code Lines	167549	Number of pure code lines
Comment Lines	30112	Number of comment lines
Comment Ratio	15.31	The percentage of comment lines
Dead Functions	4.91	The percentage of dead functions
Long Methods Percentage	28.26	The percentage of long methods
Nested Ifs	950	The number of deep nested ifs
Total Lines	196624	Total number of lines

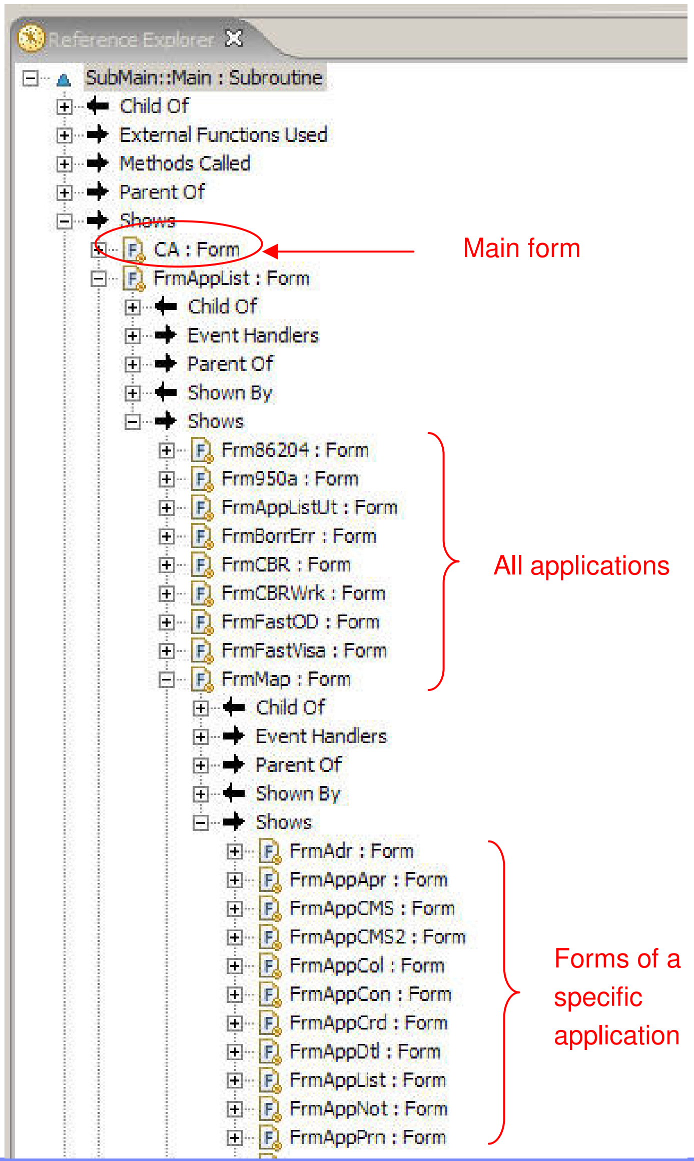
Drill Down Analysis

- Calling chains
- Buffer data analysis
- Coding practices

- Tooling support for the first two

Drill Down Analysis
Calling Chains: Form Flow

- Main form flow of the application
  - Starts at the SubMain::Main subroutine

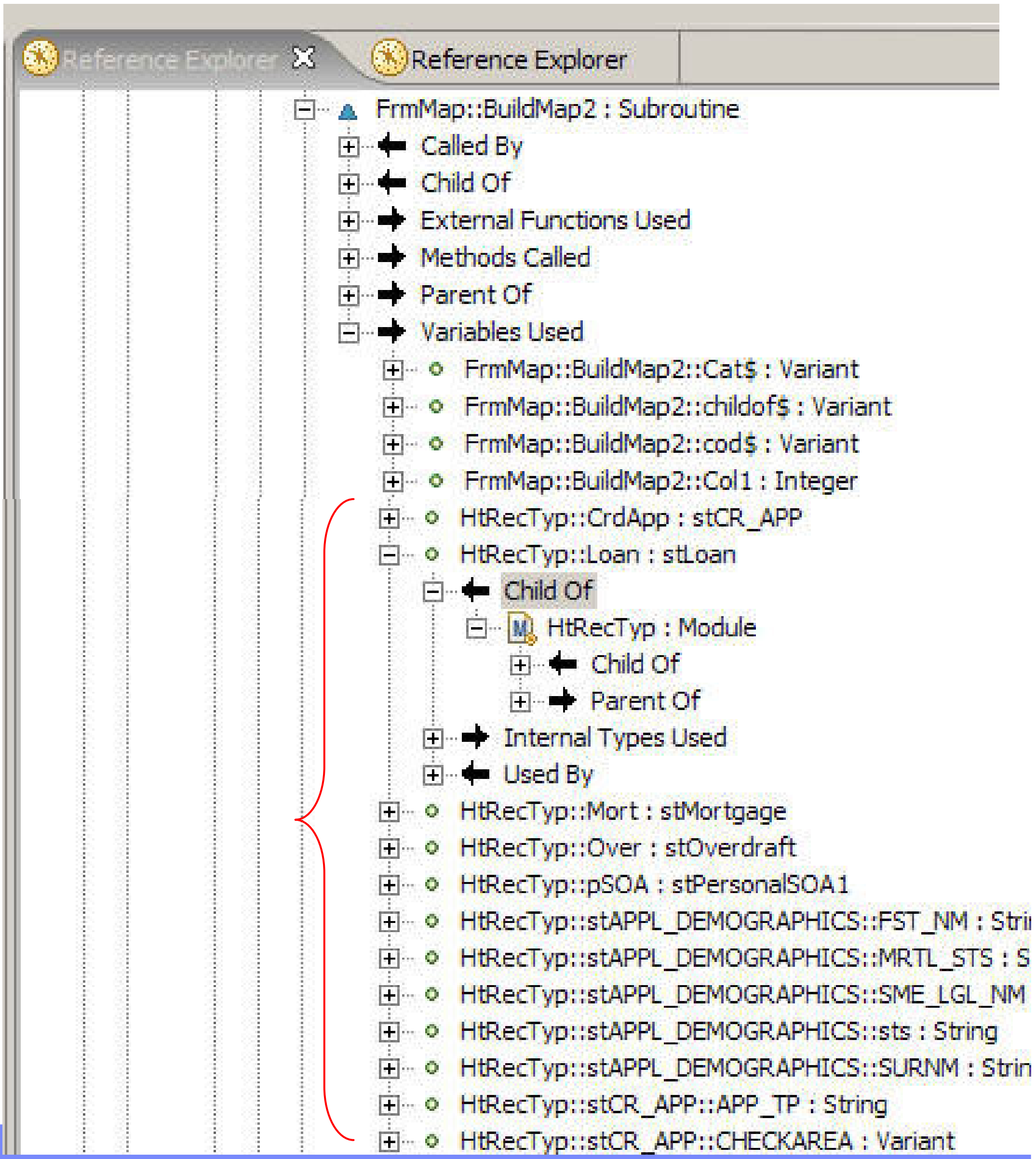

Reference Explorer

```
SubMain::Main : Subroutine
  Child Of
  External Functions Used
  Methods Called
  Parent Of
  Shows
    CA : Form
    FrmAppList : Form
      Child Of
      Event Handlers
      Parent Of
      Shown By
      Shows
        Frm86204 : Form
        Frm950a : Form
        FrmAppListUt : Form
        FrmBorrErr : Form
        FrmCBR : Form
        FrmCBRWrk : Form
        FrmFastOD : Form
        FrmFastVisa : Form
        FrmMap : Form
          Child Of
          Event Handlers
          Parent Of
          Shown By
          Shows
            FrmAdr : Form
            FrmAppApr : Form
            FrmAppCMS : Form
            FrmAppCMS2 : Form
            FrmAppCol : Form
            FrmAppCon : Form
            FrmAppCrd : Form
            FrmAppDtl : Form
            FrmAppList : Form
            FrmAppNot : Form
            FrmAppPrn : Form
```

Drill Down Analysis
Buffer Data Analysis

- Identify the most commonly used buffer elements throughout the application and their usage

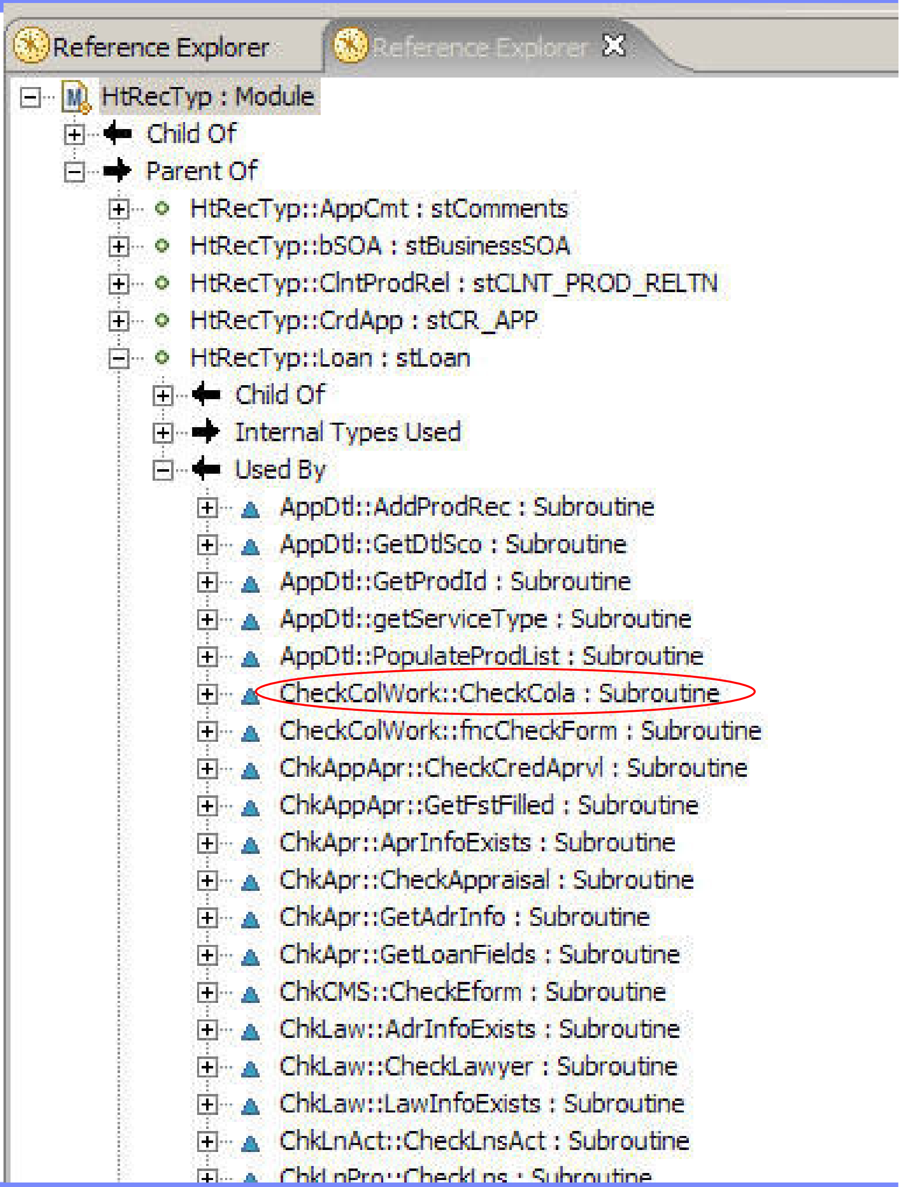
- Find usage of global variables that are defined in the HtRecTyp module


Reference Explorer
Reference Explorer

```
FrmMap::BuildMap2 : Subroutine
  Called By
  Child Of
  External Functions Used
  Methods Called
  Parent Of
  Variables Used
    FrmMap::BuildMap2::Cat$ : Variant
    FrmMap::BuildMap2::childof$ : Variant
    FrmMap::BuildMap2::cod$ : Variant
    FrmMap::BuildMap2::Col1 : Integer
    HtRecTyp::CrdApp : stCR_APP
    HtRecTyp::Loan : stLoan
      Child Of
        HtRecTyp : Module
          Child Of
          Parent Of
      Internal Types Used
      Used By
    HtRecTyp::Mort : stMortgage
    HtRecTyp::Over : stOverdraft
    HtRecTyp::pSOA : stPersonalSOA1
    HtRecTyp::stAPPL_DEMOGRAPHICS::FST_NM : Stri
    HtRecTyp::stAPPL_DEMOGRAPHICS::MRTL_STS : S
    HtRecTyp::stAPPL_DEMOGRAPHICS::SME_LGL_NM
    HtRecTyp::stAPPL_DEMOGRAPHICS::sts : String
    HtRecTyp::stAPPL_DEMOGRAPHICS::SURNM : Strin
    HtRecTyp::stCR_APP::APP_TP : String
    HtRecTyp::stCR_APP::CHECKAREA : Variant
```

Drill Down Analysis
Buffer Data Analysis

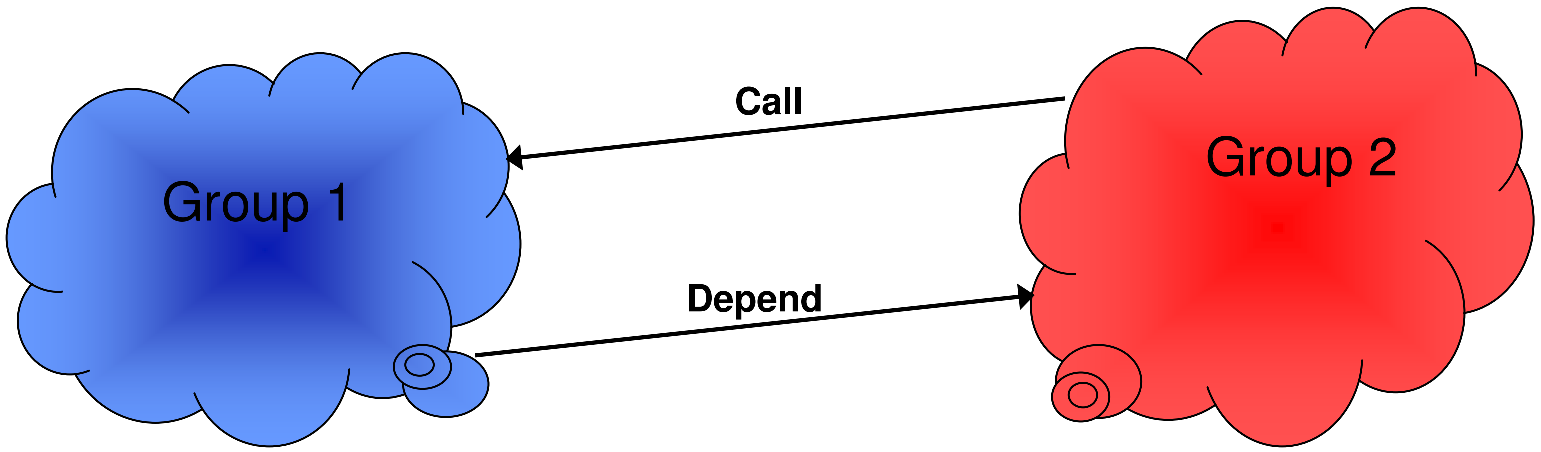
- Drilling down, we get a list of all subroutines using any of the global variables
- Open the using subroutines in a separate window to better inspect the usage


Reference Explorer | Reference Explorer

```
HtRecTyp : Module
  Child Of
  Parent Of
    HtRecTyp::AppCmt : stComments
    HtRecTyp::bSOA : stBusinessSOA
    HtRecTyp::ClntProdRel : stCLNT_PROD_RELTN
    HtRecTyp::CrdApp : stCR_APP
    HtRecTyp::Loan : stLoan
      Child Of
      Internal Types Used
      Used By
        AppDtl::AddProdRec : Subroutine
        AppDtl::GetDtlSco : Subroutine
        AppDtl::GetProdId : Subroutine
        AppDtl::getServiceType : Subroutine
        AppDtl::PopulateProdList : Subroutine
        CheckColWork::CheckCola : Subroutine
        CheckColWork::fncCheckForm : Subroutine
        ChkAppApr::CheckCredAprvl : Subroutine
        ChkAppApr::GetFstFilled : Subroutine
        ChkApr::AprInfoExists : Subroutine
        ChkApr::CheckAppraisal : Subroutine
        ChkApr::GetAdrInfo : Subroutine
        ChkApr::GetLoanFields : Subroutine
        ChkCMS::CheckEform : Subroutine
        ChkLaw::AdrInfoExists : Subroutine
        ChkLaw::CheckLawyer : Subroutine
        ChkLaw::LawInfoExists : Subroutine
        ChkLnAct::CheckLnsAct : Subroutine
        ChkLnPro::CheckLns : Subroutine
```

Additional Drill Down based on Hypothetical Solutions

- Group several elements together
  - Manually
  - Automatically, based on advanced analyses
    - Layers
    - Resource usage
- Aggregate the references between the groups



Wrap Up - Main conclusions gathered from the assessment phases

- There are many code objects with high complexity, which contain the most business logic
  - Shared amongst forms and products
- Forms generally follow a standard design pattern, with typically a low number of links between them
- Examples for violation of MVC approach
  - Forms calling host directly
  - Forms calling a routine in another form
- The data access layer appears to be a discrete set of objects that can easily be partitioned

Wrap Up

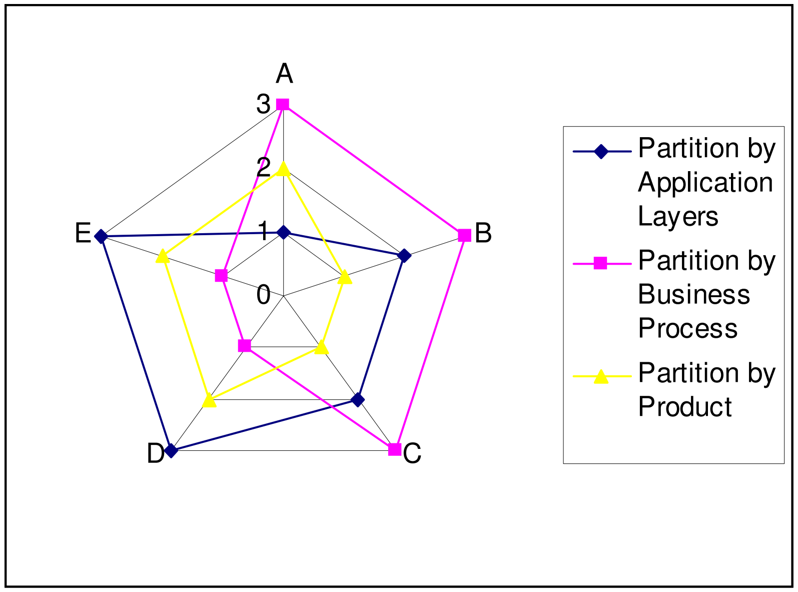
- In light of the analysis, each partition strategy was evaluated against a number of factors
  - Intrusiveness to the application
    - Indication of complexity and risk of changing the existing code
  - Amount of throwaway code
    - Indication of the effort and cost associated with writing code for the interim stages, that will be removed later on
  - Maintainability of the interim solution
    - Indication of effort, complexity and risk of maintaining the interim application, and possibly multiple versions of them in parallel
  - Alignment with the overall transformation strategy
  - Ability to factor the transformation into smaller, incremental steps

Wrap Up

Evaluation Principle	Partition by Layers	Partition by Process	Partition by Product
A. Intrusiveness	1	3	2
B. Throwaway code	2	3	1
C. Interim maintainability	2	3	1
D. Alignment with strategy	3	1	2
E. Ability to factor	3	1	2

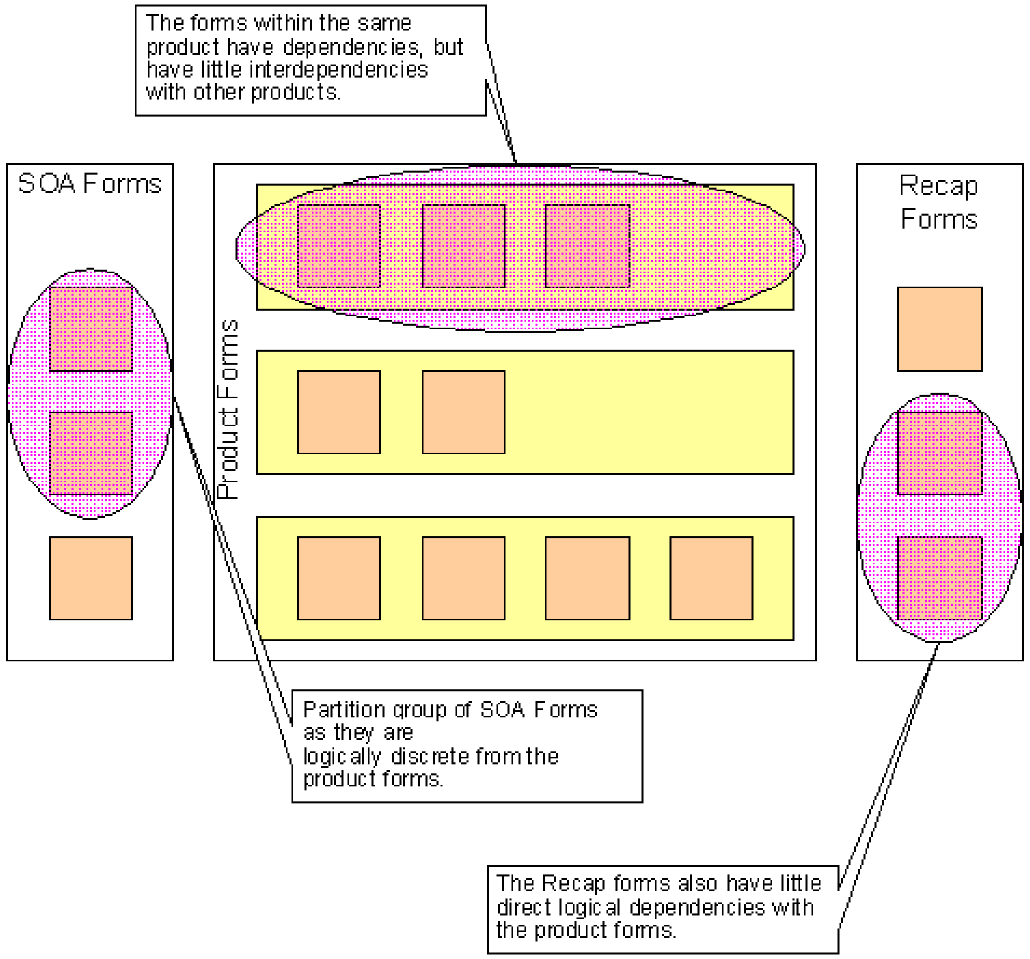
Wrap Up

- There is no one particular clear winner
  - Partition by Application Layer - does not align to the transformation strategy and will reap little early return on investment (ROI)
  - Partition by Product - partially aligns to the transformation strategy but will reap little early ROI, and is moderately complex to implement
  - Partition by Business Process - aligns closely to the transformation strategy and will likely reap early ROI, but is complex to implement



Wrap Up - Assessment Results

- A “hybrid” approach was selected
  - A combination of “by Product” and “by Business Process”


The forms within the same product have dependencies, but have little interdependencies with other products.
SOA Forms
Product Forms
Recap Forms
Partition group of SOA Forms as they are logically discrete from the product forms.
The Recap forms also have little direct logical dependencies with the product forms.

Summary and Conclusions

- Tooling support is helpful in such engagements
  - Shorten the assessment phase
  - Identify migration obstacles early
  - Reduce risk, more confidentiality
  - Validate migration plans
  - Identify services/components

- Still human intervention is required to apply the tool

- Future directions
  - Support for grouping, both manual and automatic
  - Identify patterns: architectural, behavioral
  - Help in business rule extraction