Longhorn

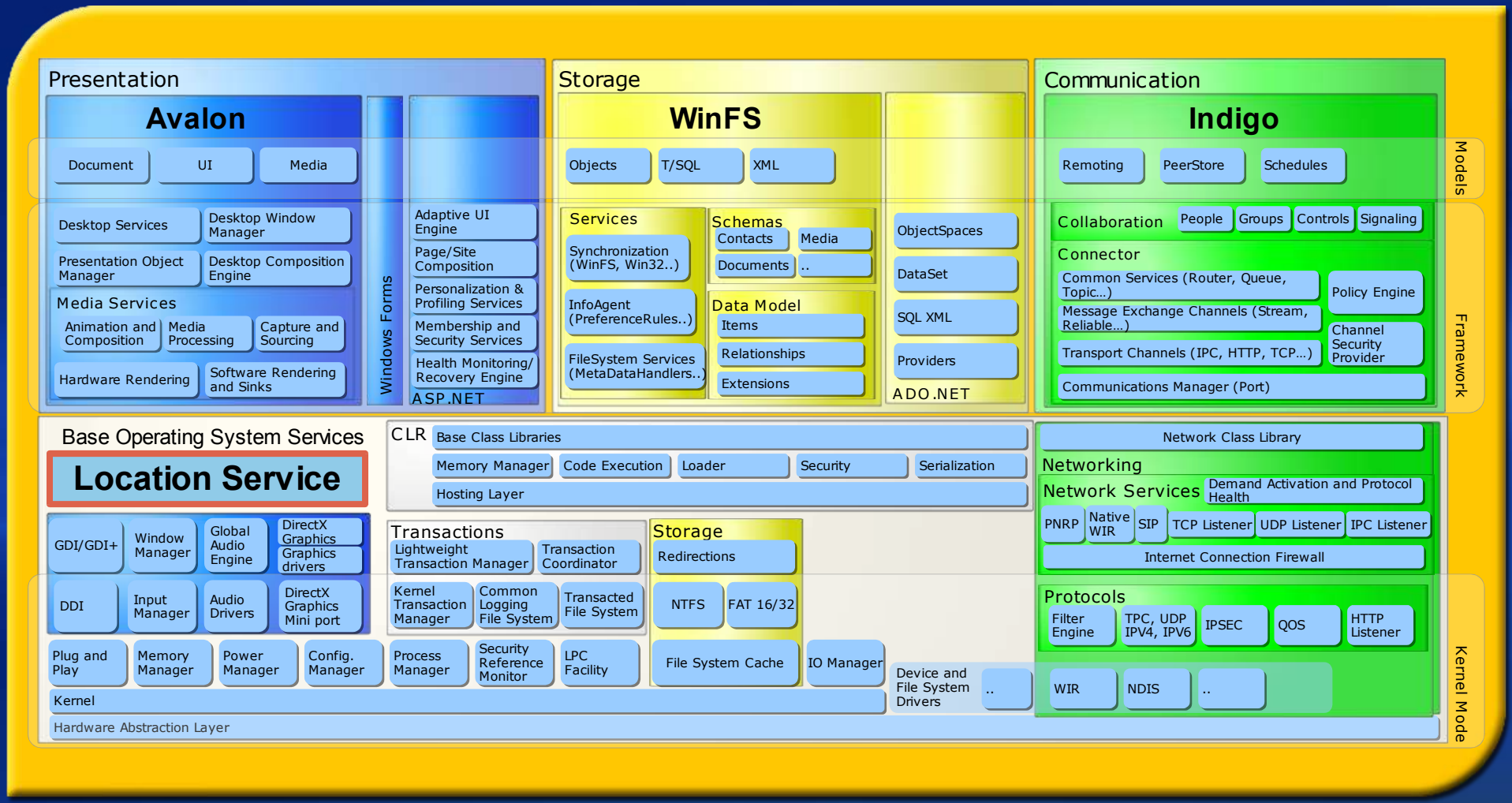
Brief overview

Matthew J. Dovey
Oxford University Computing Service

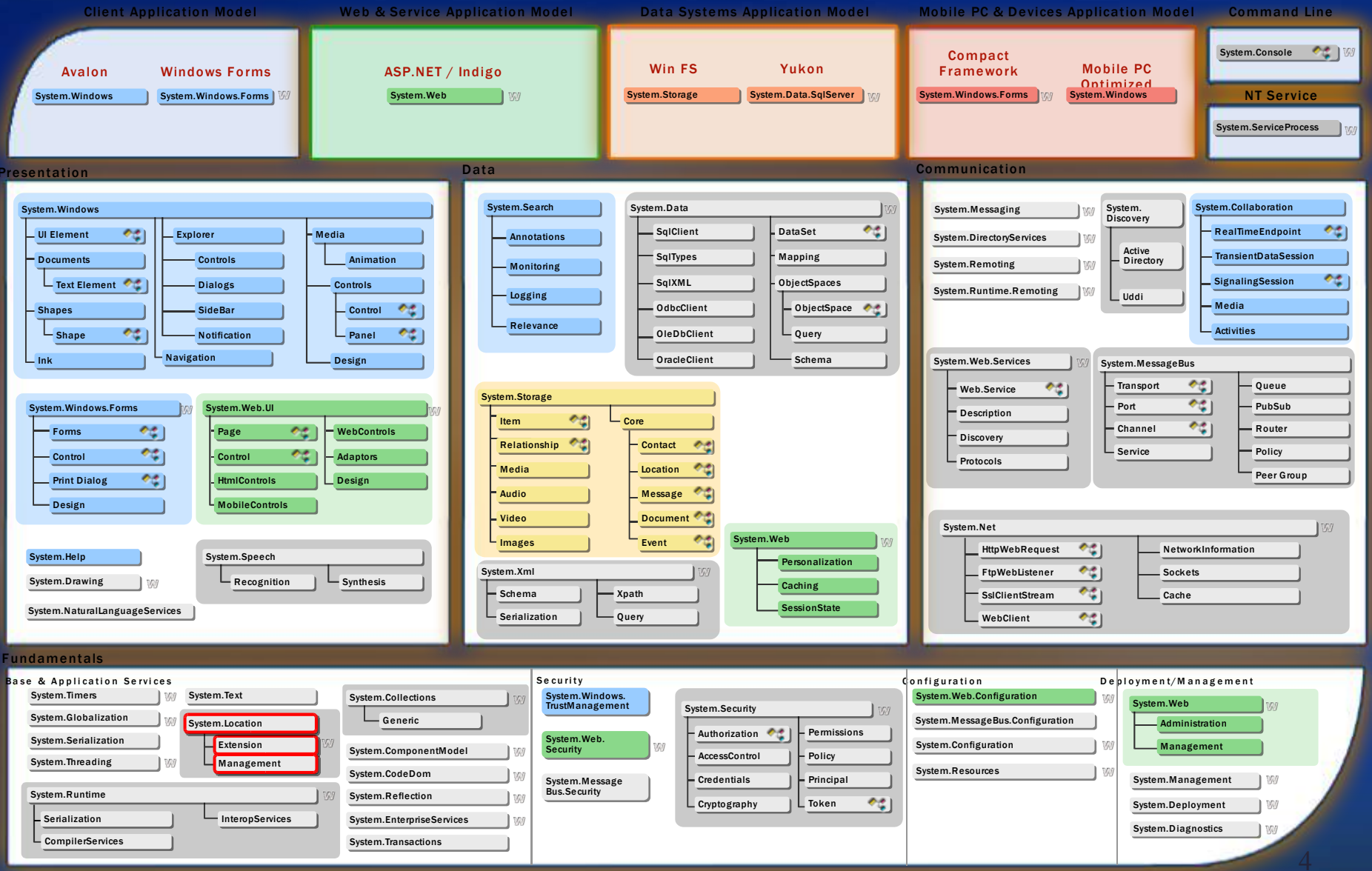
Longhorn

- Codename for the next major version of Windows
- Major release (although most technologies have been seen before)
- Currently in alpha technical previews
- Due for release 2005-2006? (when ready!)
- Interim updates
  - e.g. Windows XP Service Pack 2
  - Windows 2003 Server “SE”

Longhorn Architecture



WinFX

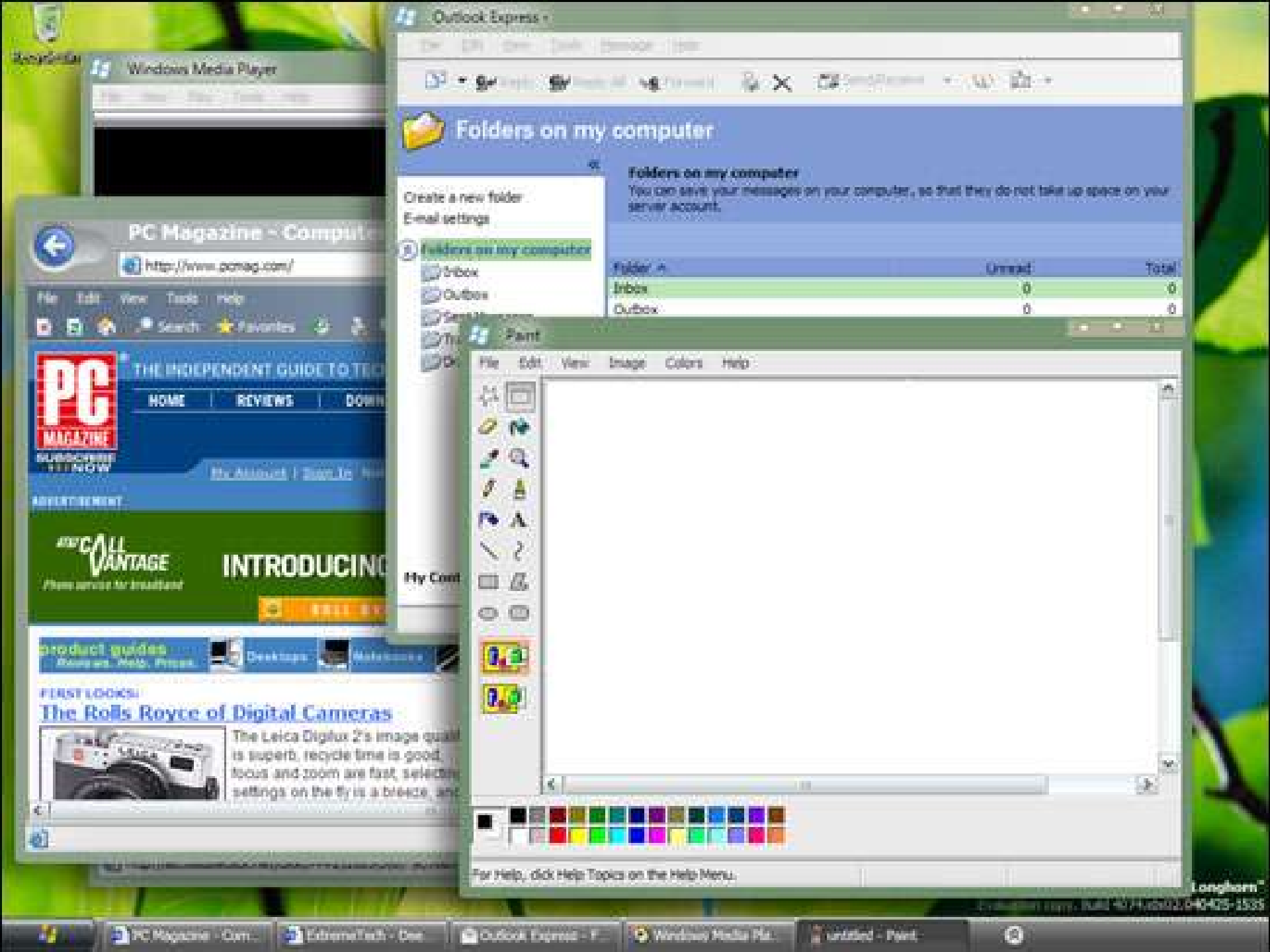

Client Application Model
Avalon
Windows Forms
System.Windows
System.Windows.Forms
Web & Service Application Model
ASP.NET / Indigo
System.Web
Data Systems Application Model
Win FS
Yukon
System.Storage
System.Data.SqlServer
Mobile PC & Devices Application Model
Compact Framework
Mobile PC Optimized
System.Windows.Forms
System.Windows
Command Line
System.Console
NT Service
System.ServiceProcess
Presentation
System.Windows
UI Element
Documents
Text Element
Shapes
Shape
Ink
Explorer
Controls
Dialogs
SideBar
Notification
Navigation
Media
Animation
Controls
Control
Panel
Design
System.Windows.Forms
Forms
Control
Print Dialog
Design
System.Web.UI
Page
Control
HtmlControls
MobileControls
WebControls
Adaptors
Design
System.Help
System.Drawing
System.NaturalLanguageServices
System.Speech
Recognition
Synthesis
Data
System.Search
Annotations
Monitoring
Logging
Relevance
System.Data
SqlClient
SqlTypes
SqlXML
OdbcClient
OleDbClient
OracleClient
DataSet
Mapping
ObjectSpaces
ObjectSpace
Query
Schema
System.Storage
Item
Relationship
Media
Audio
Video
Images
Core
Contact
Location
Message
Document
Event
System.Xml
Schema
Serialization
Xpath
Query
System.Web
Personalization
Caching
SessionState
Communication
System.Messaging
System.DirectoryServices
System.Remoting
System.Runtime.Remoting
System.Discovery
Active Directory
Uddi
System.Collaboration
RealTimeEndpoint
TransientDataSession
SignalingSession
Media
Activities
System.Web.Services
Web.Service
Description
Discovery
Protocols
System.MessageBus
Transport
Port
Channel
Service
Queue
PubSub
Router
Policy
Peer Group
System.Net
HttpWebRequest
FtpWebListener
SslClientStream
WebClient
NetworkInformation
Sockets
Cache
Fundamentals
Base & Application Services
System.Timers
System.Globalization
System.Serialization
System.Threading
System.Text
System.Location
Extension
Management
System.Runtime
Serialization
CompilerServices
InteropServices
System.Collections
Generic
System.ComponentModel
System.CodeDom
System.Reflection
System.EnterpriseServices
System.Transactions
Security
System.Windows.TrustManagement
System.Web.Security
System.Message Bus.Security
System.Security
Authorization
AccessControl
Credentials
Cryptography
Permissions
Policy
Principal
Token
Configuration
System.Web.Configuration
System.MessageBus.Configuration
System.Configuration
System.Resources
Deployment/Management
System.Web
Administration
Management
System.Management
System.Deployment
System.Diagnostics

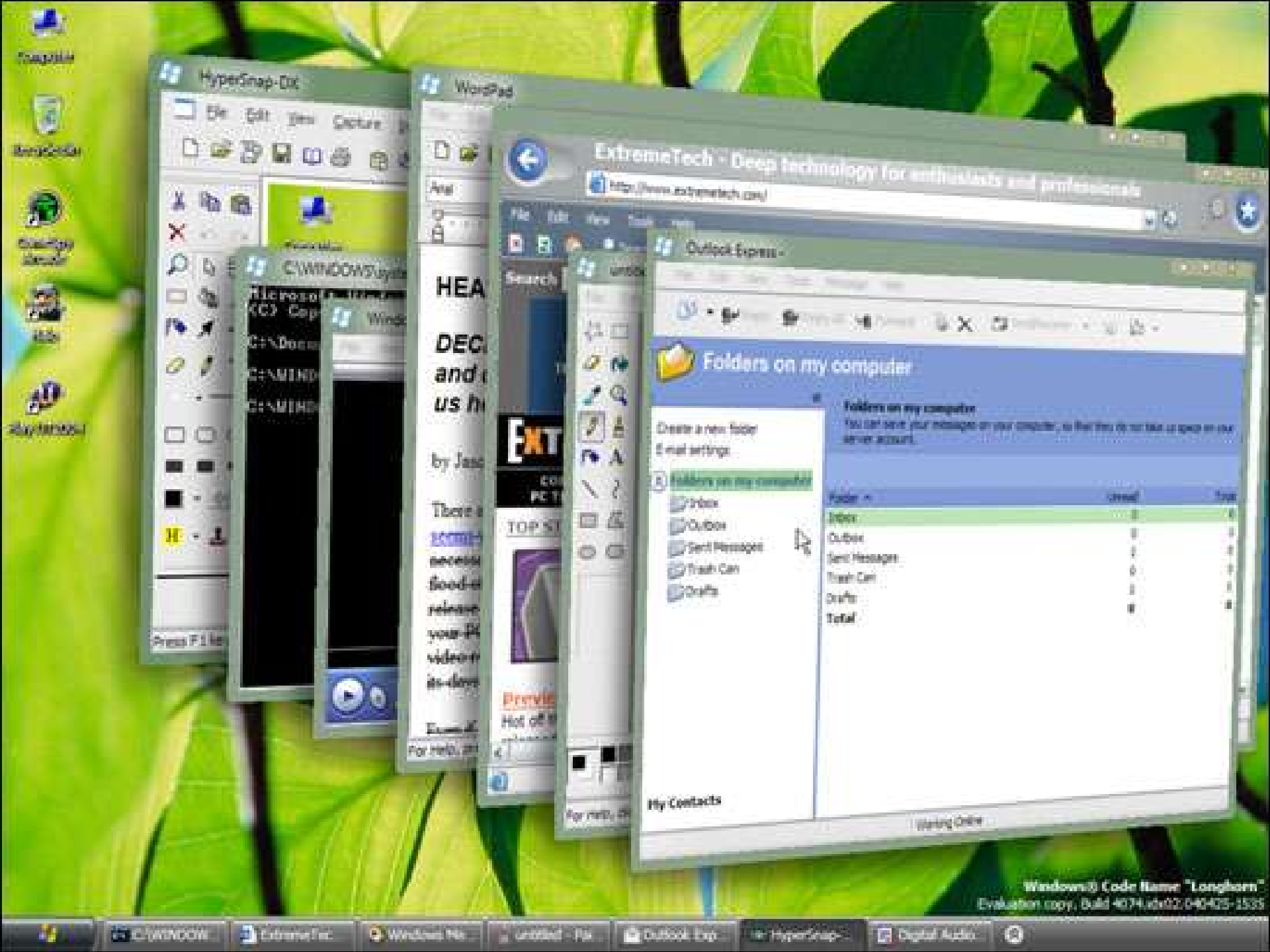
Longhorn

Aero

Matthew J. Dovey
Oxford University Computing Service


Windows Media Player
File View Play Tools Help
Ready
Windows® Code Name "Longhorn"


Windows Media Player
Outlook Express
Folders on my computer
Create a new folder
E-mail settings
Folders on my computer
Inbox
Outbox
Folder
Unread
Total
Inbox 0 0
Outbox 0 0
PC Magazine - Comput
http://www.pcmag.com/
File Edit View Tools Help
Search Favorites
PC MAGAZINE
THE INDEPENDENT GUIDE TO TEC
HOME
REVIEWS
DOWN
INTRODUCING
product guides
Desktops
FIRST LOOKS:
The Rolls Royce of Digital Cameras
Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.
PC Magazine - Com...
ExtremeTech - Dee...
Outlook Express - F...
Windows Media Pla...
untitled - Paint


HyperSnap-DX
WordPad
ExtremeTech - Deep technology for enthusiasts and professionals
Outlook Express
Folders on my computer
Windows® Code Name "Longhorn"

Longhorn

Avalon

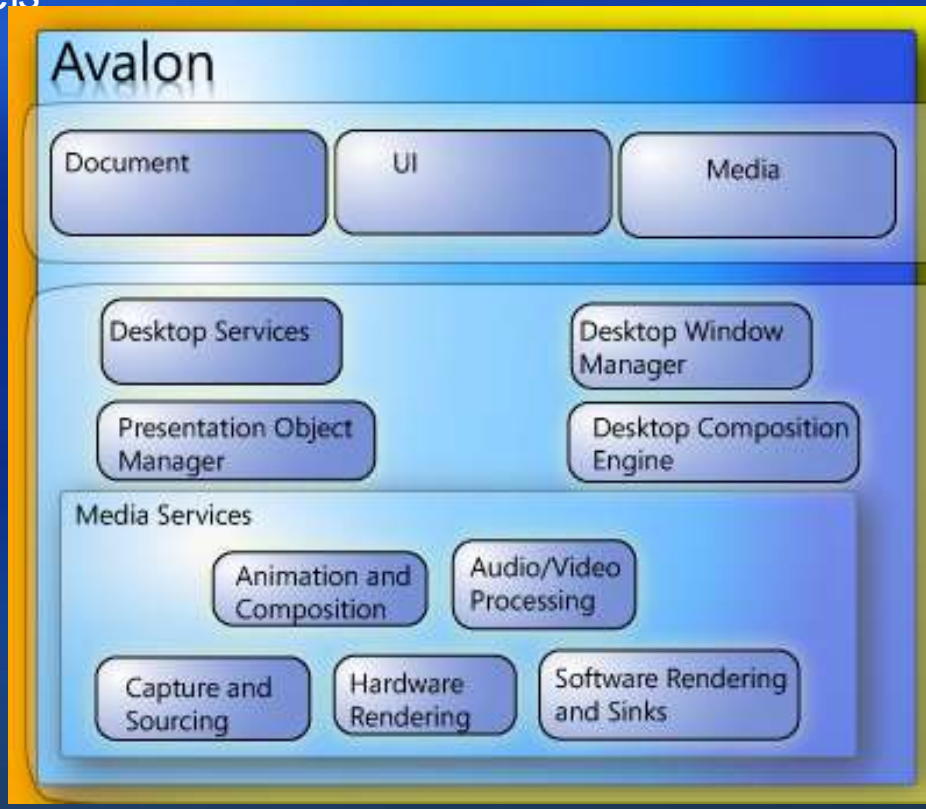
Matthew J. Dovey
Oxford University Computing Service

The Avalon Approach

- Unified approach to UI, Documents, and Media
  - Integration as part of development and experience
- Integrated, vector-based composition engine
  - Utilizing the power of the PC throughout the graphics stack
- Declarative programming
  - Bringing designers directly into application development

Integration – The Guiding Vision

- Avalon - the integrated platform for UI, Media, and Documents
  - UI, Media, and Documents share the benefits of a new stack built from the bottom up
  - Anchored on the .NET Framework and Direct3D
  - Parallel procedural and declarative models
- **UI**
  - Flexible component architecture
  - Layout services
  - Two-way transformable data binding
- **Media**
  - Graphics
  - Audio, video, animation
- **Documents**
  - Fixed, flow, and adaptive layouts
  - Pagination/printing
  - Rights management



Developer Experience
Best of Web, Best of Windows

Bringing together the advantages from both worlds

- Web
  - seamless deployment, update, and administration
  - flowable layout
  - progressive download and rendering
  - declarative model (text-based markup)
- Windows
  - unrestricted functionality
  - integration with Windows desktop
  - good offline support
  - scalability/performance
  - broad developer language and tools support

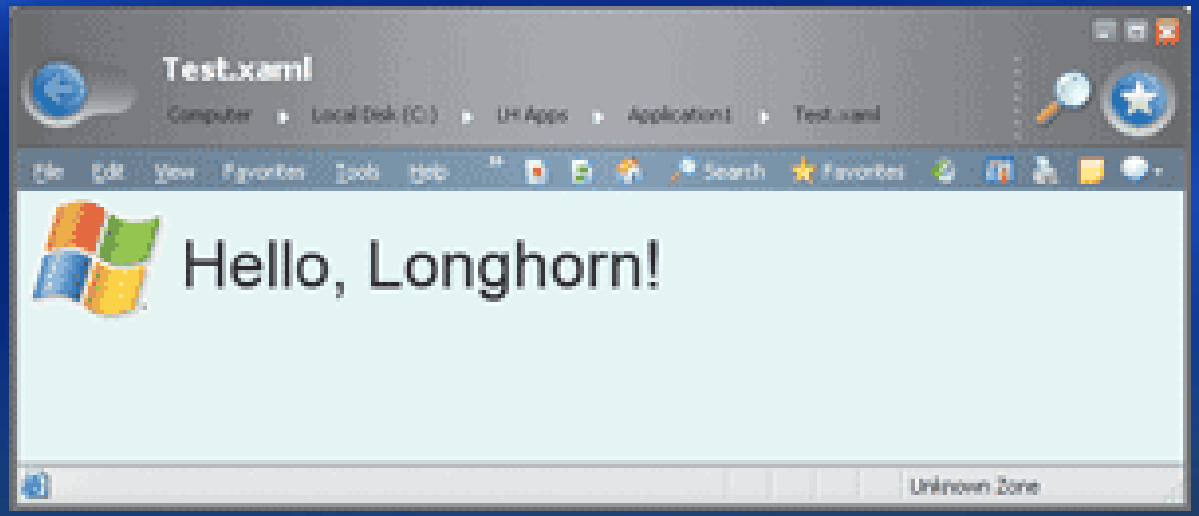
Developer Experience
Declarative Programming

- Extensible Application Markup – codenamed XAML
  - One-to-one correspondence with object model
  - Key role in enabling interoperation between UI authoring tools and developer tools

- Fundamental XAML concepts:
  - Markup can contain code
  - Markup can be compiled for execution
  - Markup and procedural code are peers in functionality and performance

XAML

```
<Canvas xmlns="http://schemas.microsoft.com/2003/xaml"
    Background="LightCyan" Width="100%" Height="100%">
  <Image Source="lh.bmp" Canvas.Left="5" Canvas.Top="5" />
  <Text Canvas.Left="90" Canvas.Top="20" FontSize="36">Hello,
    Longhorn! </Text>
</Canvas>
```

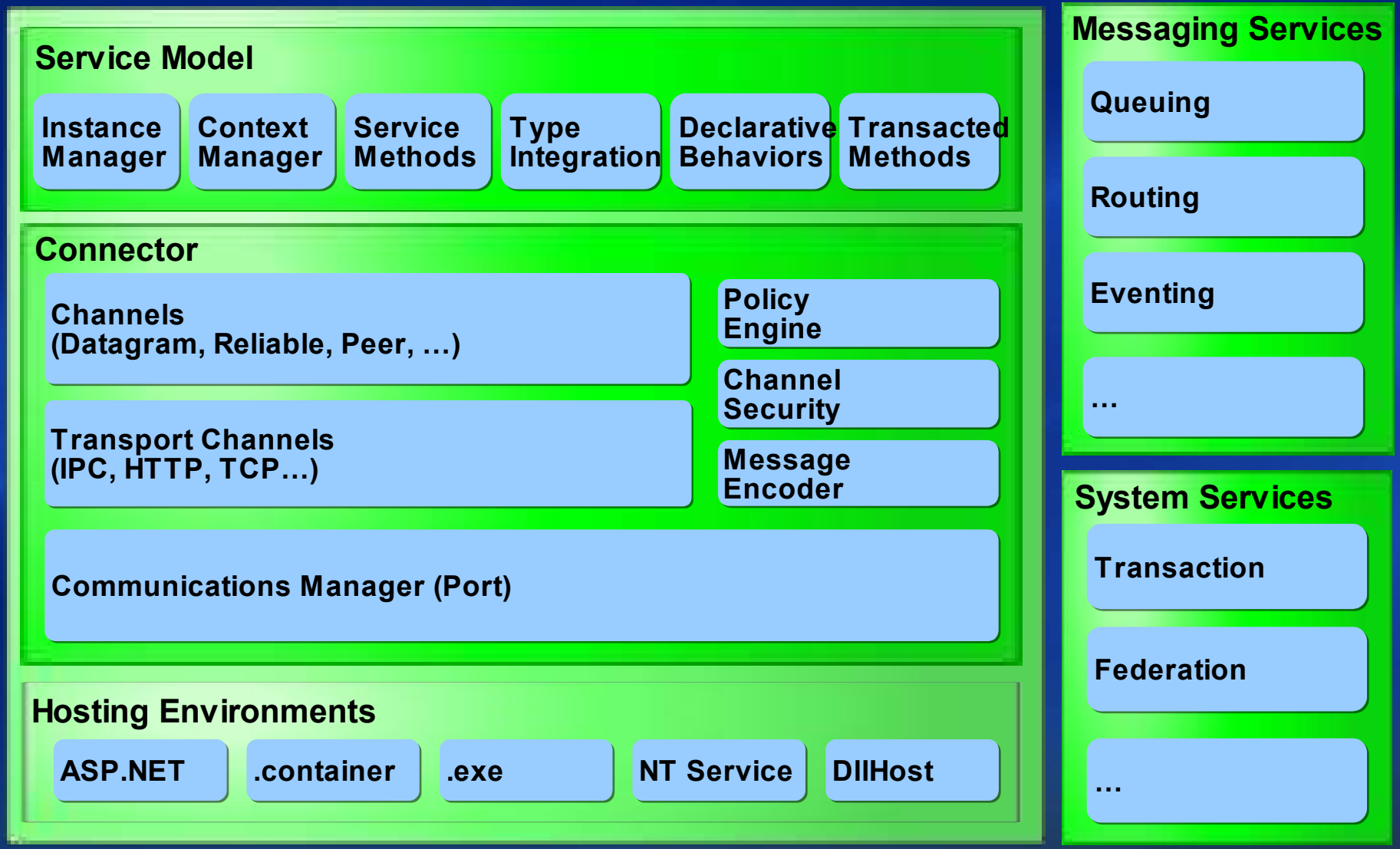


Longhorn

Indigo

Matthew J. Dovey
Oxford University Computing Service

Indigo Architecture



“Indigo” Security Goals

- Provide message-based security leveraging Web Service Security standards
- Provide simple, constrained, out-of-box security solutions that meet most application security requirements
- Provide adequate flexibility for customizing security solutions
- Provide extensibility for authentication, authorization, token types, security providers

Turn-Key Deployment
Configuration and Profiles

- Define security profiles which indicate how security requirements are to be satisfied
- Developer or deployer may define their own security profiles
- Common security profiles are predefined in machine.config
- A scope of messages are bound to a security profile

Longhorn

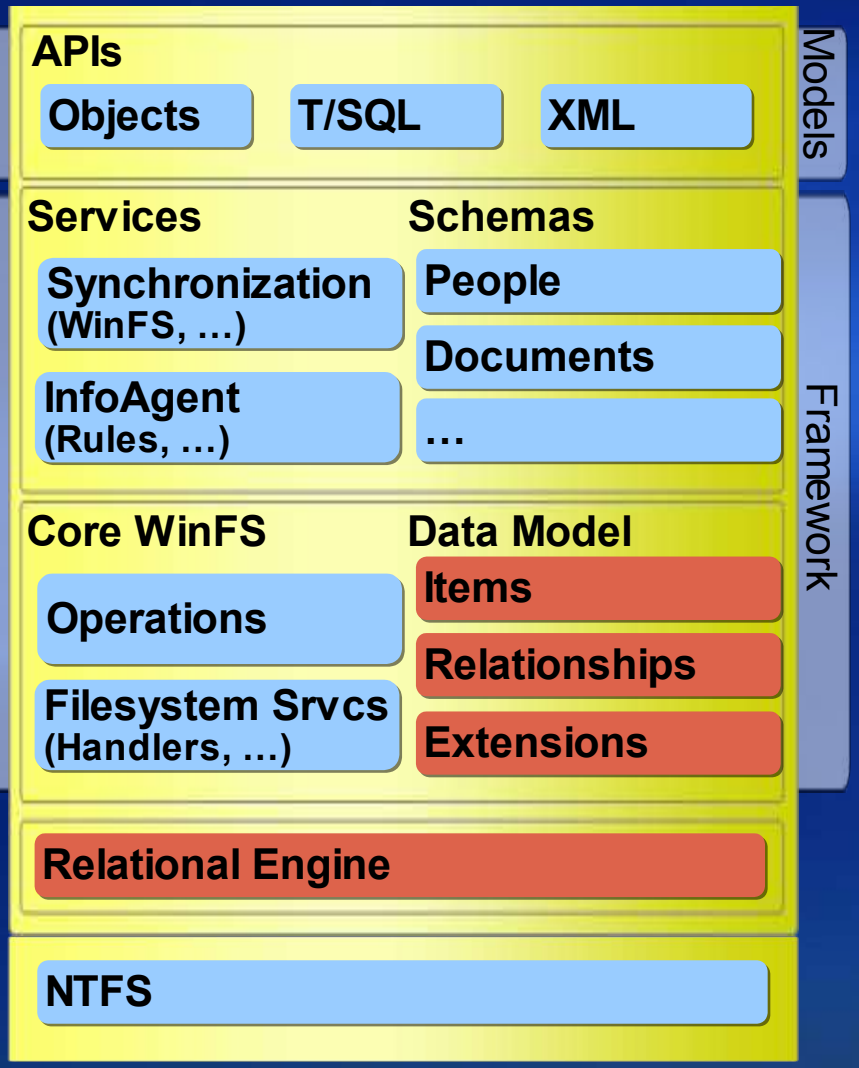
WinFS

Matthew J. Dovey
Oxford University Computing Service

WinFS Is

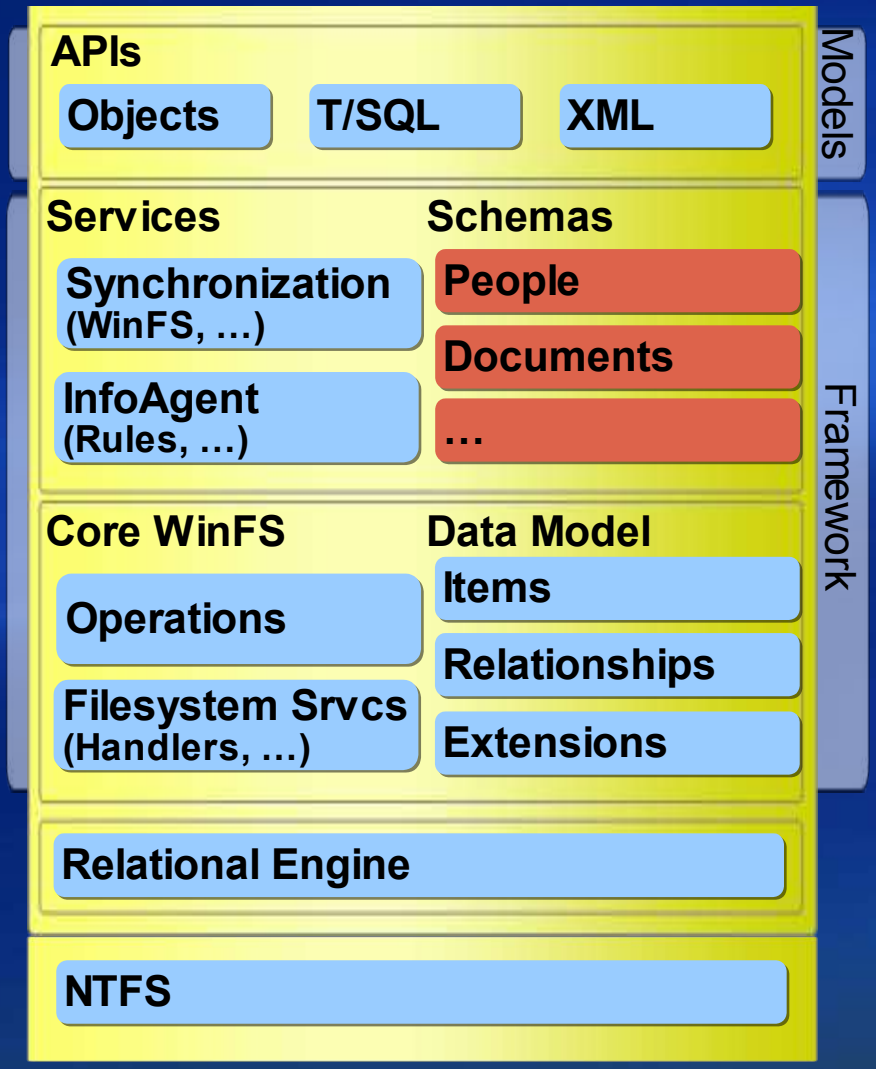
- All end-user data lives in Longhorn
- New user experience in Longhorn Shell
- A trustworthy place to store data
- Data model built on relational database technology
- Filesystem capabilities built on NTFS
- Everyday Information - domain-specific schemas
- Services that make data active

WinFS Data Model



- Items
  - The new atomic unit of data
    - Items have subsumed Files
    - Copy, put in Folders, etc.
  - A group of simple and complex types that represent data
    - Defined in a schema, arranged in types
  - Structured, Semi-Structured, and, Opaque
  - Persisted
- Relationships
  - Explicitly relate Items together
    - E.g.; Author binds Document to Contact
  - Schema can model complex items
  - Containment, reference, embedding, categories, etc.
- Extensions
  - Provide ability to add new data to existing Item types

WinFS Schemas



- Windows Everyday Information
  - Documents, Messages, Annotations, Notes
  - Media, Audio, Video, Images
  - Events, Appointments, Locations, UserTask
- Windows System
  - SystemTasks, Config, Programs
  - Explorer, Help, Security
- New Schemas
  - Developers can define own data shape
  - Comprised of
    - Scalars
    - Complex Types
    - XML
    - Binary/Filestream

Example

WinFS Schema

```
<ItemType Name="Person"
  BaseType="Core.Contact" ... >
   <Property Name="PersonalNames"
     Type="MultiSet"
     MultiSetOfType="FullName"
     Nullable="true">
   <Property Name="AddressLine"
     Type="WinFS.String" Nullable="true">
   <RelationshipType Name="Employment"
     BaseType="WinFS.Relationship"
     AllowsHolding="true"
     AllowsEmbedding="false"
     AllowsReference="true">
   <Property Name="IrisScan"
     Type="WinFS.FileStream" …/>
</ItemType>
```

Table View of Person

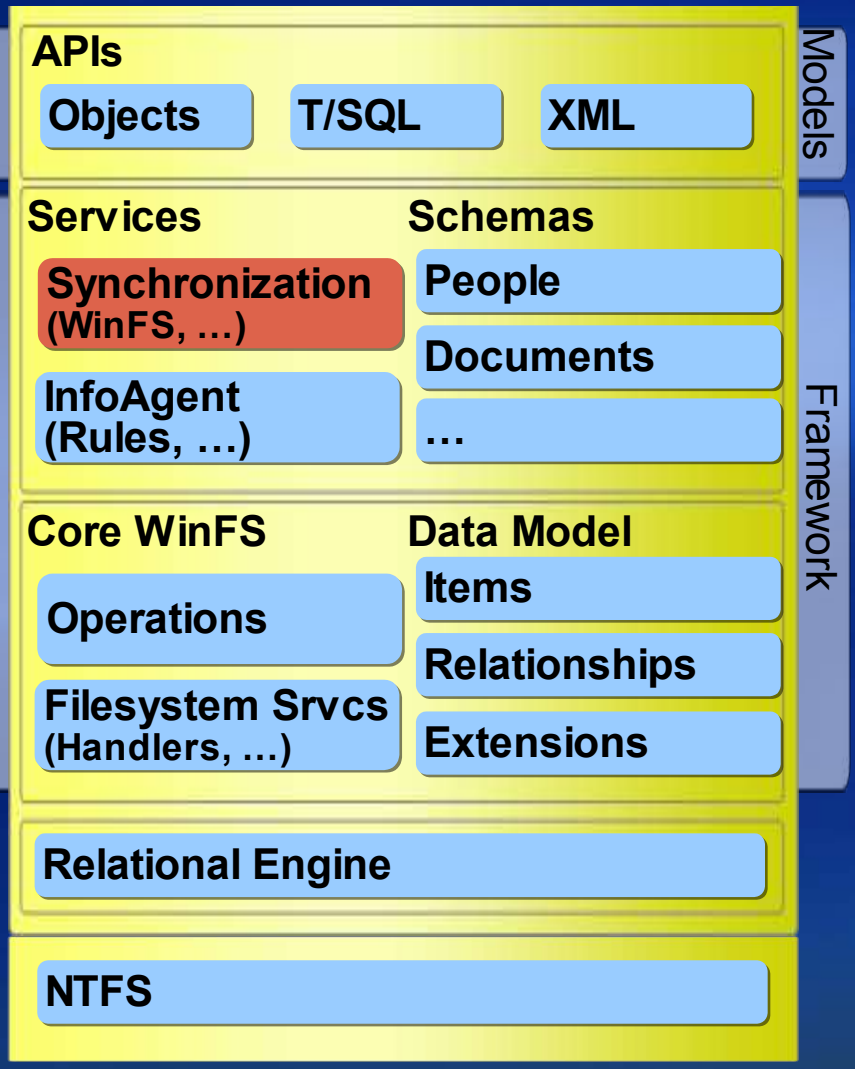
ItemId	Name	Addresses	IrisScan
	FirstName LastName	Street City State Zip Street City State Zip Street City State Zip	

NTFS stream

Longhorn And Filesystems

- Files can live solely in an NTFS volume
  - Available for boot
    - E.g., C:\Windows is in NTFS
  - Volume can be mounted on down level machine
    - E.g., Firewire drive on both XP and Longhorn
- Items can live solely in WinFS
  - File-backed Items
    - Accessible through standard Win32 APIs
    - Metadata Handlers get data in and out of file streams
  - User data moved into WinFS
    - I.e., C:\Documents and Settings
  - Has Import/Export utilities

WinFS Services
Synchronization

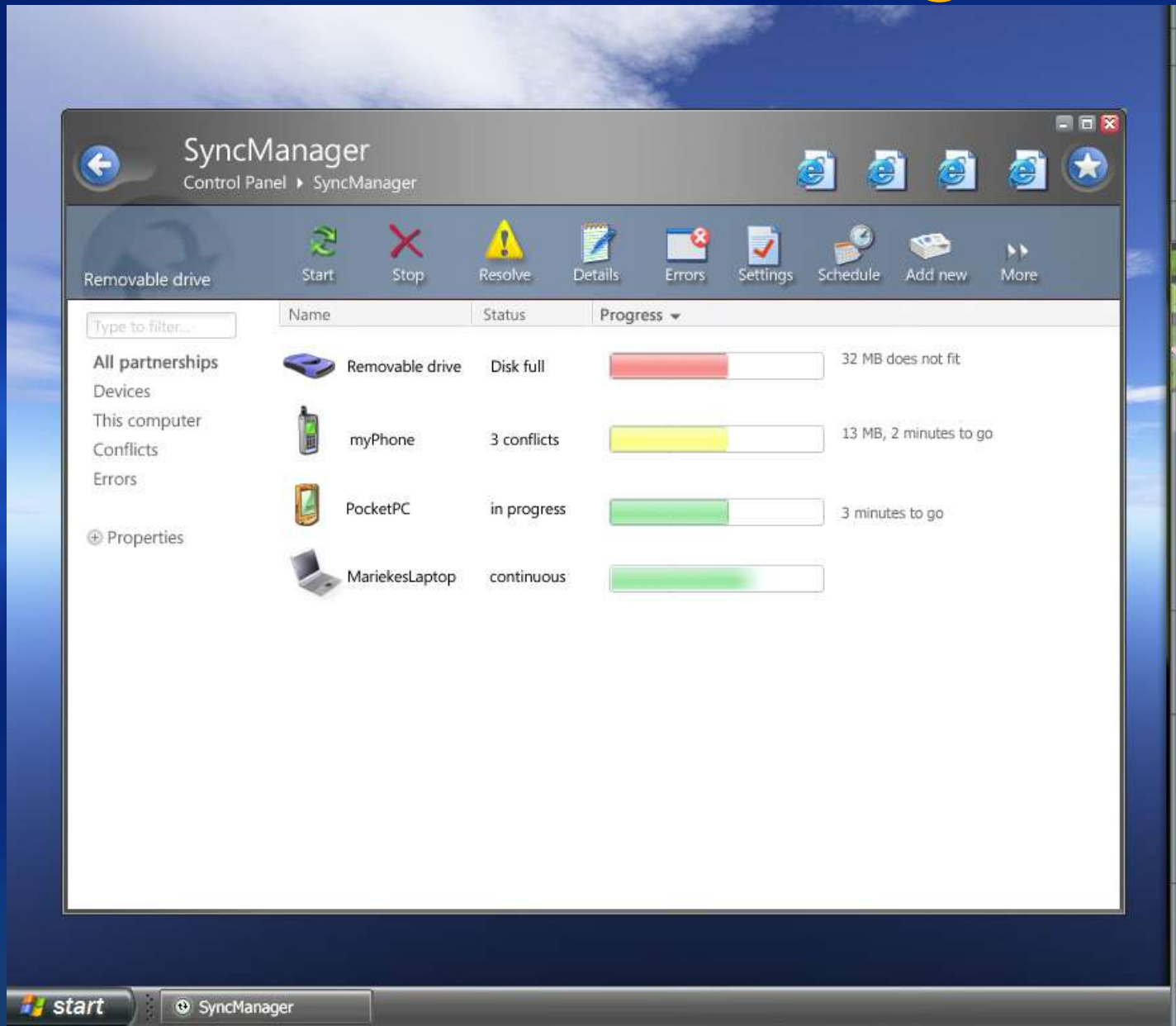


- Synchronize one WinFS with another
  - Keep My Contacts and My Files in sync across my home machines
  - Peer to Peer sharing
- Synchronize WinFS with other data sources
  - Keep My Contacts in sync with online email contacts, enterprise CRM, etc.

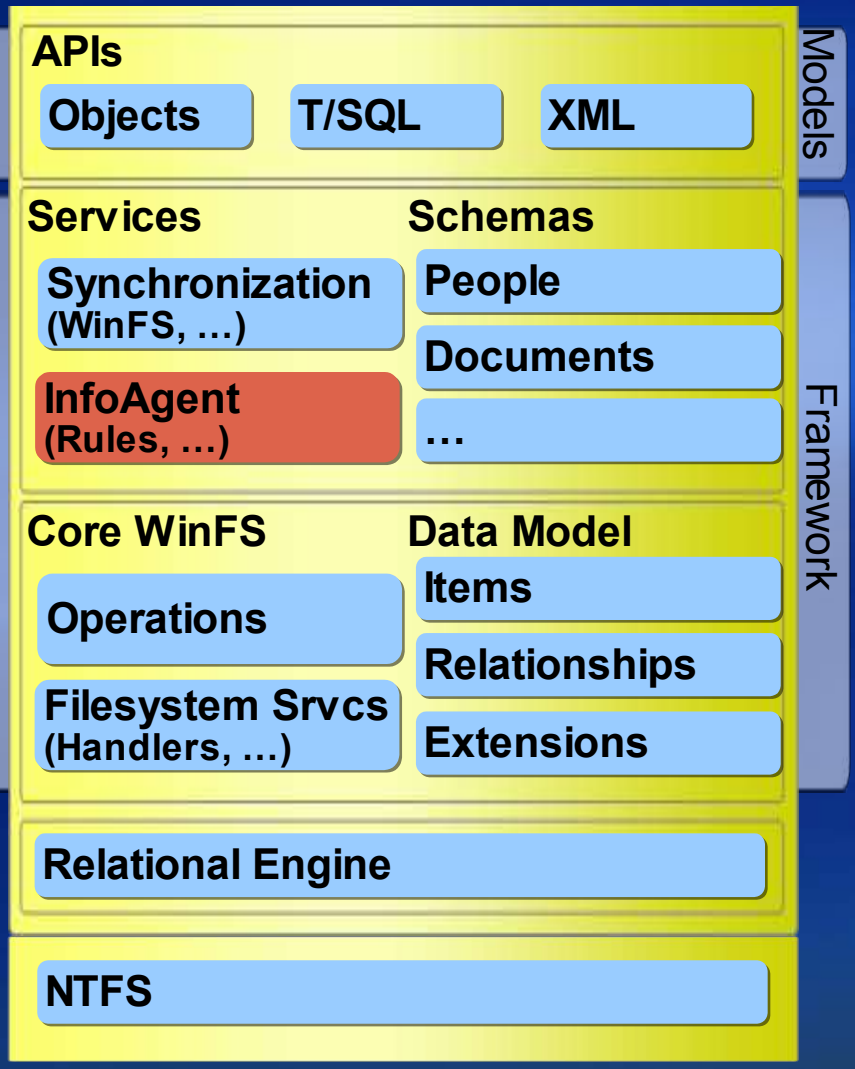
Synchronization Overview

- Approach
  - Multi-master replication
    - Replicas make changes independently
  - Net-change synchronization
    - Looking at cumulative changes, not logs
  - A set of common services for all data sources and all schemas
    - Change tracking, change enumeration, conflict handling, etc.
- Extending
  - Schema design
    - Granularity of change units is declared in the WinFS schemas
  - Custom conflict resolution handlers
    - Extend the system conflict policies with code
  - Synchronization Adaptors
    - Outside datasources for one way or bidirectional synchronization

Synchronization Manager



WinFS Services
InfoAgent



- Users want to control how their PCs behave
  - It's called a personal computer after all
  - Every aspect of the system can be personalized
- InfoAgent enables rich, flexible customization
  - "When I receive a high priority email from a customer, show me a popup message if I'm at my desk, otherwise forward it to my cell phone"
  - "When I download new photos from my camera,

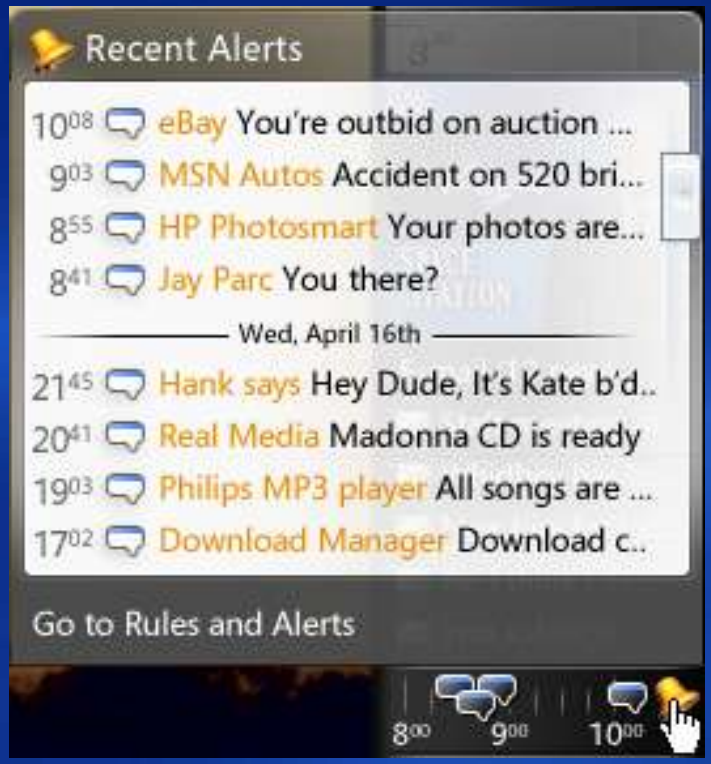
Notifications And InfoAgent

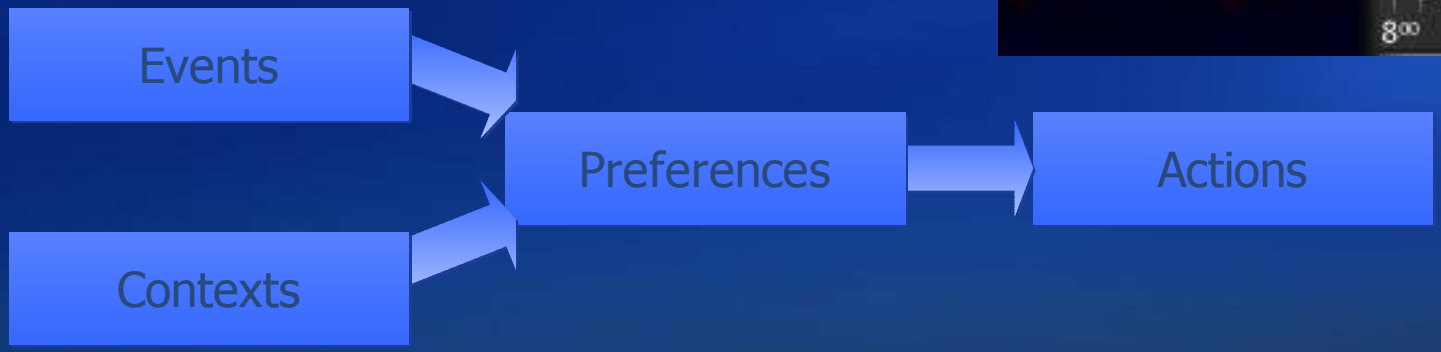
‘Active Data’ – Subscribe to WinFS changes

- Item change subscriptions
- Item Domain containment/query subscriptions

InfoAgent Integration

- Inclusive set of events, contexts, and actions
- Preferences stored as WinFS items
- Unified management of notification rules





Longhorn

Microsoft Shell

Matthew J. Dovey
Oxford University Computing Service

Microsoft Shell

Problem

- Weak cmd shell
  - Weak language
  - spotty coverage
- GUI focus
  - Hard to automate
- SDK Focus
  - Programmers

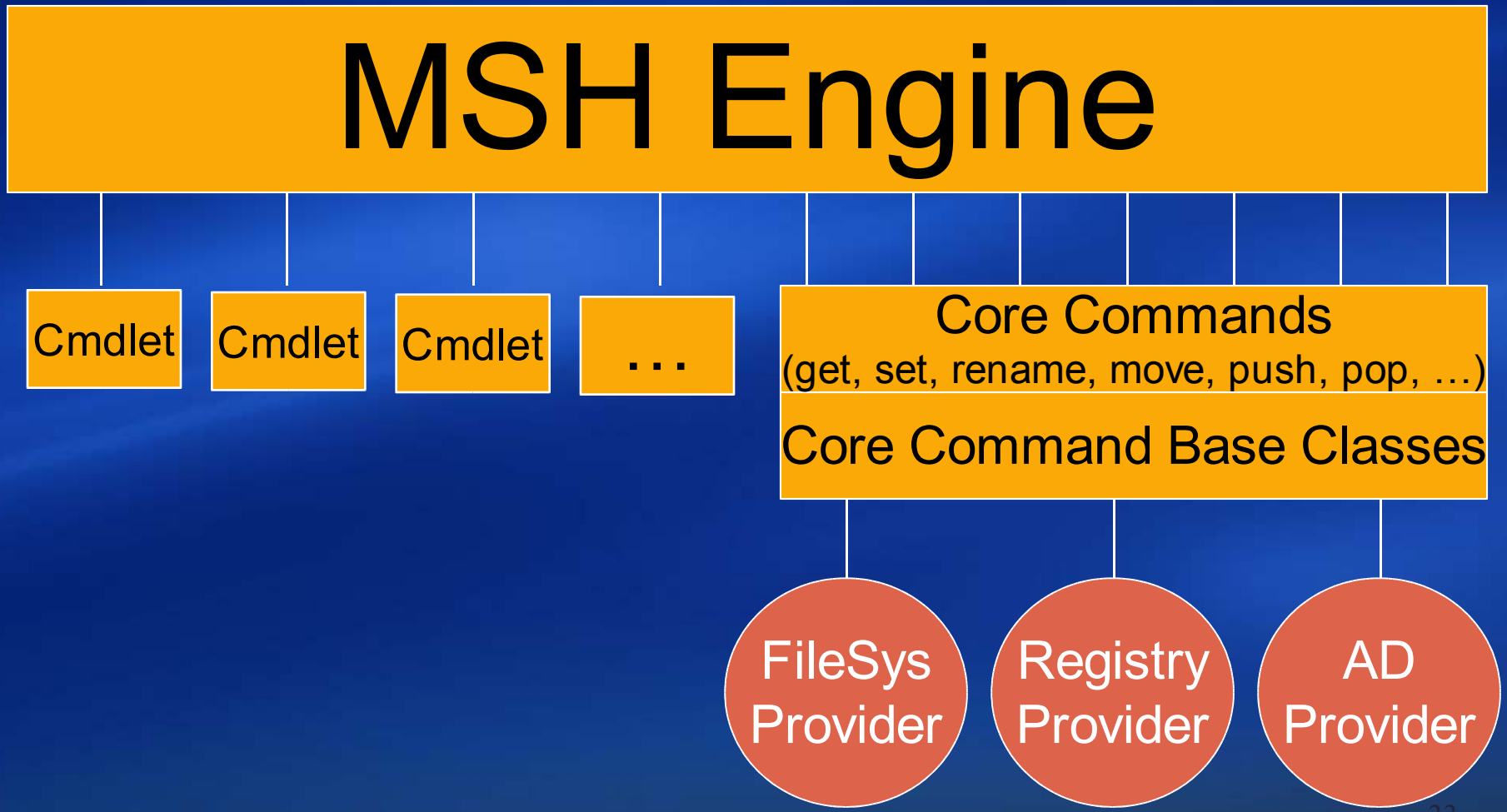
Solution: MSH

- Foundation for task-based management
- Focused on power users and admins
- Provides:
  - Interactive shell
  - Cmdlets
  - Utilities
  - Scripting language
  - Remote scripting

Core Concepts

- Command line scripting language
  - Best of sh/ksh, Perl/Ruby, DCL/CL
- Commands are classes (Cmdlets)
- Hosting model

How It Works


MSH Engine
Cmdlet
Cmdlet
Cmdlet
…
Core Commands
(get, set, rename, move, push, pop, …)
Core Command Base Classes
FileSys Provider
Registry Provider
AD Provider

Parameters And Confirmation

```
[CommandDeclaration("stop", "ps")]
public class StopPs: Cmdlet
{
 [ParsingMandatoryParameter]
 [ParsingPromptString("Name of the process")]
  public string ProcessName;
  public override void ProcessRecord()
  {
        Process [ ]ps;
       ps = Process.GetProcessesByName(ProcessName);
       foreach (Process p in ps)
       {
            if (ConfirmProcessing(p.ProcessName))
            {
                 p.Kill();
            }
       }
  }
}
```

Navigation Provider

```
[ProviderDeclaration("REG", "Registry",
ProviderCapabilityFlags.None)]
public class RegistryProvider :  NavigationCmdletBase
{
    protected override void GetItem(string path)
    {
        RegistryKey key = GetRegkeyForPath(path, false);

        if (key == null)
        {  WriteErrorObject(path,
            new ArgumentException("does not exist"));
        }
        WriteObject(key);
    }
    ....
}
```

Longhorn

Application Automation

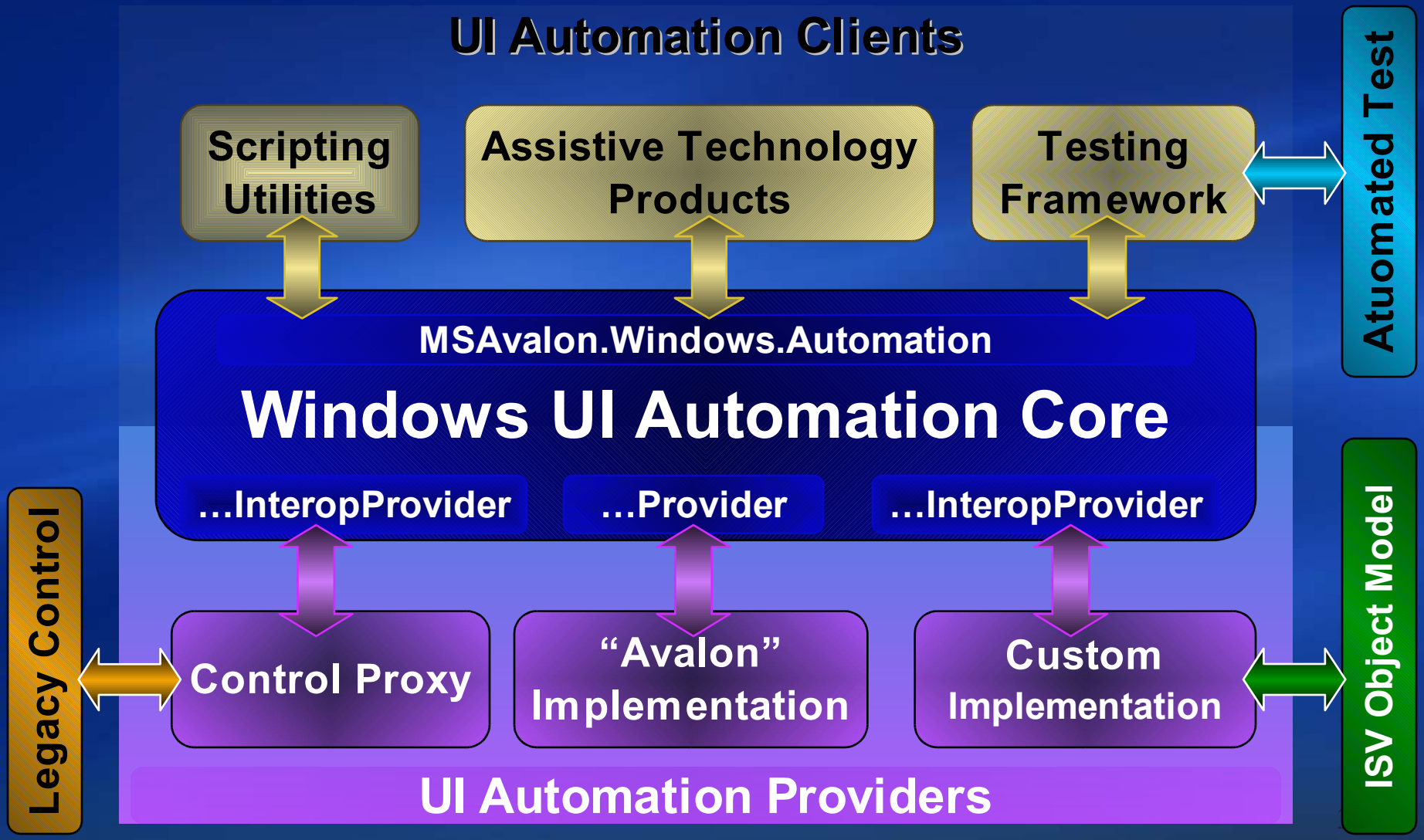
Matthew J. Dovey
Oxford University Computing Service

"UI Automation" defined

– ***code that programmatically drives another application's UI to yield a desired result***

- Gather information about the UI
  - Dynamically discover UI structure
  - Extract property information
  - Receive event notifications when UI changes
  - Query an element for its behavior
- Interact with UI elements
  - Click a button, scroll a list, move a window, etc.
  - Inject keystrokes and mouse input

Longhorn Model: UI Automation



Windows UI Automation

- Automation framework built into Longhorn
  - Platform-level support for automating all UI elements
    - Avalon, WinForms, Win32, Visual Basic, etc.
  - Exposes a consistent object model for all controls
    - 3rd party controls easily integrate into model
  - Security Model – client must be trusted
  - Locale, machine, and resolution independent
- Creates new opportunities for innovation in:
  - Automated UI Testing
  - Assistive Technology Products
  - Command-and-Control Utilities

UI Automation Overview

- Logical Tree – structure of the UI
  - Stitches all UI trees into one coherent structure
  - Eliminates unnecessary elements
  - Resembles the structure perceived by an end user
- Properties – important UI information
  - Name, Bounding Rectangle, Persistent ID, etc.
- Events – notification of UI changes
  - Window creation, change in focus or selection, etc
- Control Patterns – control behavior
  - Scroll, Selection, Window, ExpandCollapse, etc.
- Input – simple mouse and keyboard input

UI Automation Control Patterns

- Mutually exclusive classes of control behavior
- Control developers (providers) expose these patterns for new or existing controls
- Automation developers (clients) use patterns to
  - Discover what functionality a control offers
  - Gather pattern-specific property information
  - Receive pattern-specific events
  - Programmatically manipulate the control
- Examples:
  - Button: Invoke
  - ListBox: Scroll, Selection
  - ComboBox: Scroll, Selection, ExpandCollapse

Security Model

- No default automation permissions
- UI Automation functionality is protected according to the following permissions:
  - Read – Navigate tree, get properties, receive events
  - Write – Call control pattern methods
  - Input – Call methods in the Input class
- Access to Rights Managed requires additional permissions

Longhorn

Deployment

Matthew J. Dovey
Oxford University Computing Service

ClickOnce Vision

*Bring the **ease** & **reliability** of web application deployment to client applications.*

The Best of the Client & Web

	Web	Click Once	MSI Client
Reach	Y		
No Touch Deployment	Y	**Y**	
Low System Impact	Y	**Y**	
Install/Run Per-User	Y	**Y**	
Rich / Interactive		**Y**	Y
Offline		**Y**	Y
Windows Shell Integration		**Y**	Y
Per-Machine/Shared Components			Y
Unrestricted Install			Y

Install Goals

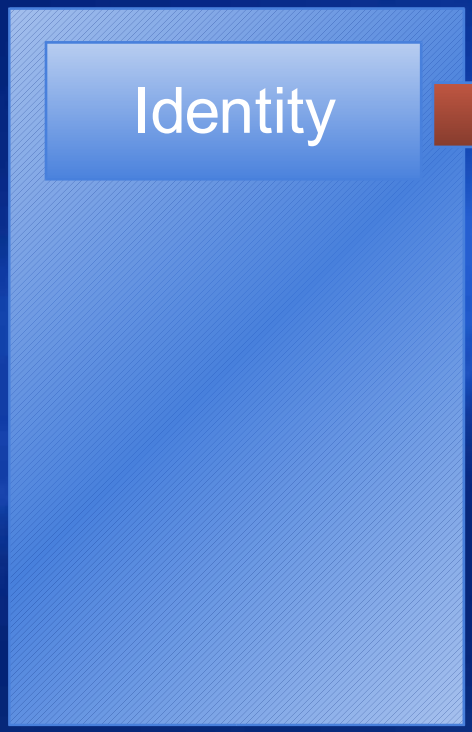
- Reduce install fragility
- Allow what's low impact
  - Ex. App file copy, start menu integration, etc…
  - Can always undo what was installed
- Disallow what's not low impact
  - Apps never run with admin rights (LUA)
  - Driver registration, COM objects, etc..
  - Custom actions; large source of install uncertainty
- Expand the definition of "low impact"
  - Requires OS Changes. Starts with Longhorn

Declarative Install

- Application Manifest
  - Describes the application
  - Ex.. What assemblies constitute the app
  - Authored by the developer
- Deployment Manifest
  - Describes the application deployment
  - Ex.. What version clients should run
  - Authored by the administrator

Deployment Manifest

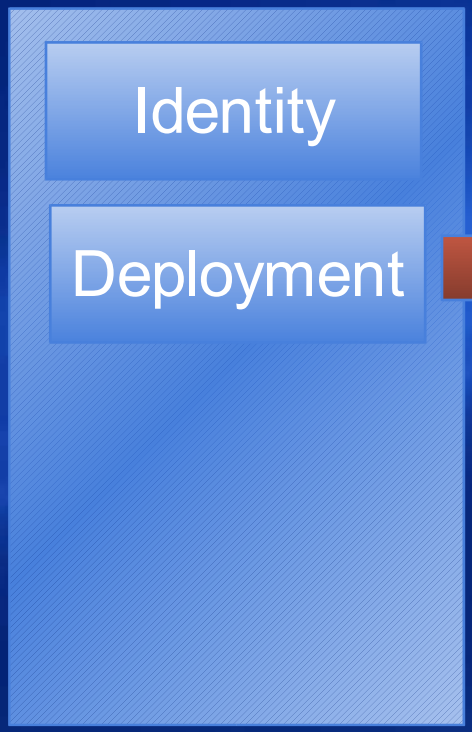
MyApp.Deploy




```
<assemblyIdentity
    name="TaskVision.deploy"
    version="1.0.0.0"
    publicKeyToken="…"
    processorArchitecture="x86"
    asmv2:culture="en-US" />

 <description
     asmv2:publisher="Microsoft"
     asmv2:product="TaskVision">
  </description>
```

Deployment Manifest

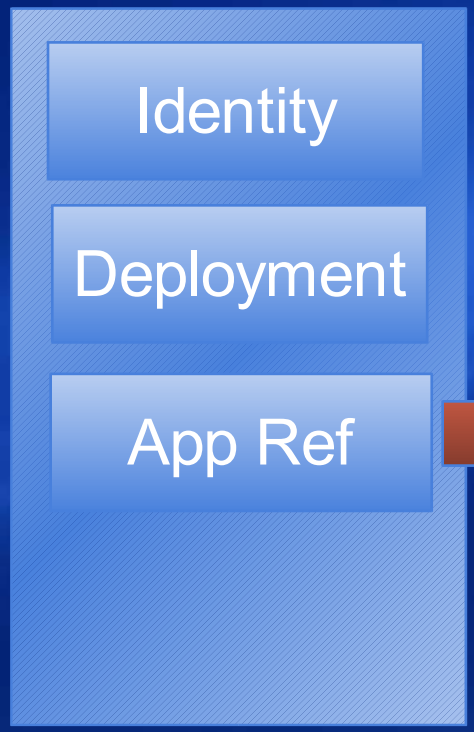
MyApp.Deploy




```
<deployment isRequiredUpdate="false" >

  <install shellVisible="true" />

  <subscription>
    <update>
      <beforeApplicationStartup />
      <periodic>
        <minElapsedTimeAllowed
                time="0" unit="hours" />
      </periodic>
    </update>
  </subscription>

</deployment>
```

Deployment Manifest

MyApp.Deploy




```
<dependency>

  <dependentAssembly>
    <assemblyIdentity
        name="TaskVision.manifest"
        version="1.0.0.0"
        publicKeyToken="…"
        processorArchitecture="x86"
        asmv2:culture="en-US" />
  </dependentAssembly>

  <asmv2:installFrom
      codebase="1.0.0.0/TV.manifest" />

</dependency>
```

Deployment Manifest

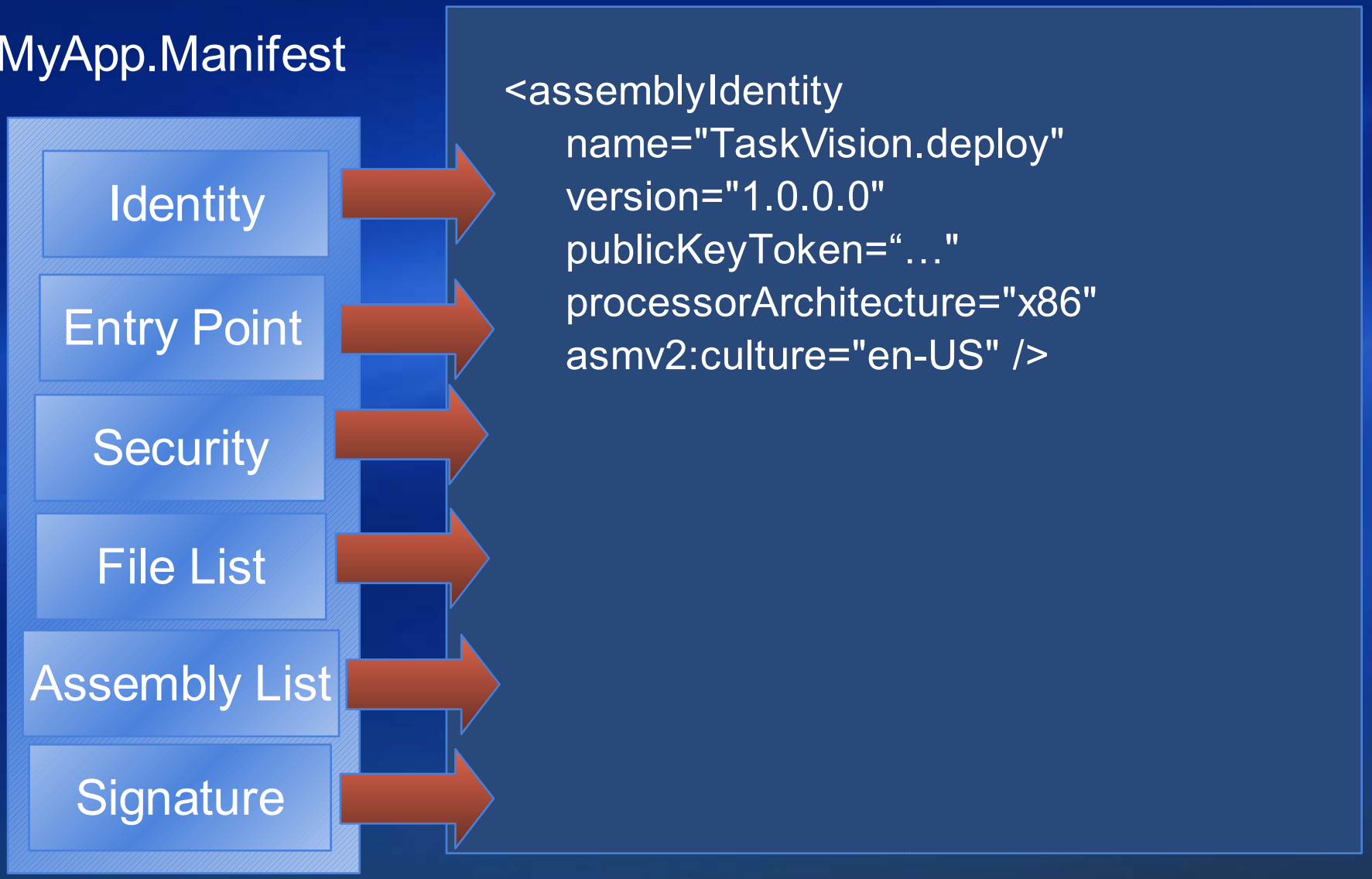
MyApp.Deploy

- Identity
- Deployment
- App Ref
- Signature

```
<Signature >
  <SignedInfo>
    <Reference URI="">
      <DigestMethod Algorithm="http://…" />
      <DigestValue>2xKk…</DigestValue>
    </Reference>
</SignedInfo>

  <SignatureValue>vNTBod96H7k…</SignatureValue>

  <KeyInfo>
    <KeyValue>
      <RSAKeyValue>
        <Modulus>+Wnh5RN9…</Modulus>
        <Exponent>AQAB</Exponent>
      </RSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

Application Manifest

MyApp.Manifest



Deployment Options

- ‘Installed’ Applications
  - From Web, UNC or CD
  - Start Menu, Add/Remove Programs
  - Varied update options
- ‘Launched' Applications
  - App launches but doesn’t “install”
  - No Start Menu, Add/Remove Programs
  - Always update on launch

Update Options

- On App Startup
  - If found, ask user to update app
- After App Startup
  - If found, ask user to update on next run
- Programmatic
  - Integrate update experience into app
- Required
  - Update can specify minimum version required
- **Background Updates**
  - **Updates drizzle in silently – like Windows Updates**
  - **“Longhorn” only**

Secure Updates

- Only the original deployer can update
  - No auto-deployment of viruses
- Manifests are signed
  - XMLDSIG
  - Deployer key needed to publish updates

“Longhorn Web” Apps

- Integrated with Browser
  - Install UI built into browser
  - Best possible user experience
  - Leverages Avalon app/navigation model
  - No shell presence (ex. Start Menu shortcut)
  - Runs in semi-trust
- Progressive Install
  - App automatically installs as it’s used
  - File level install

When Should I Use The Windows Installer (MSI) ?

- ClickOnce is the solution for new self-contained applications
  - Low System Impact
  - No Touch Deployment
  - Install / Run Per-User
  - Rich Interactive applications
- Use Windows Installer if you need to
  - Install Shared Resources
  - Install Win32 Applications
  - Perform custom actions during installation

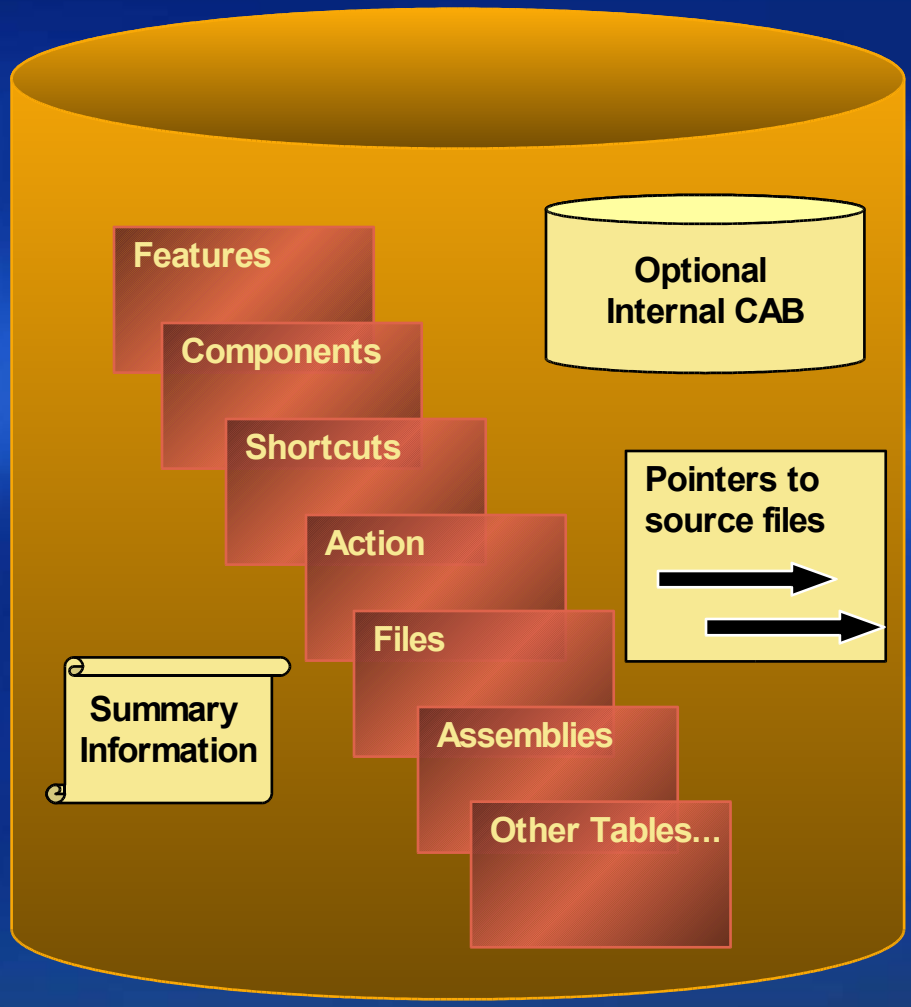
ClickOnce And Windows Installer (MSI)

	Click Once	MSI Client
No Touch Deployment	Y	
Low System Impact	Y	Y*
Install/Run Per-User	Y	Y
Rich / Interactive	Y	Y
Offline	Y	Y
Windows Shell Integration	Y	Y
Per-Machine/Shared Components		Y
Unrestricted Install		Y

* MSI applications can be authored for “low system impact”

Windows Installer Basics .MSI

- MSI database
  - Populated by setup developer
  - .MSI file extension
  - One per product
  - Described in relational tables
- Products have
  - Features
  - Components
  - Installable resources
  - Entry points



Windows Installer Basics .MSP

- MSP is a Windows Installer patch package
- Patches make changes to the configuration information database and resources (files, registry)
- Patch package (MSP) contains
  - Summary Information Stream
  - Transforms
  - Cabinet file

Windows Installer v4.0
MSI 40

- Longhorn extensions
  - MSI will support new Longhorn shell extension manifest
- No-Reboot support for setup / updates
  - MSI detects processes holding files in use
  - Sends notification to processes
  - Design your applications to save state, shutdown and resume

Windows Installer v4.0 Image Based Setup

- Longhorn uses a new Image Based Setup model
  - Minimizes number of images
  - Deployment of Windows + Applications is faster
  - Images can be maintained, serviced &modified offline/online
- MSI applications can be deployed with Images
  - FASTOEM property is used by major OEMs to speed up factory floor setup
  - Files copied with the OS image
  - Installation and configuration are done on first boot

Longhorn

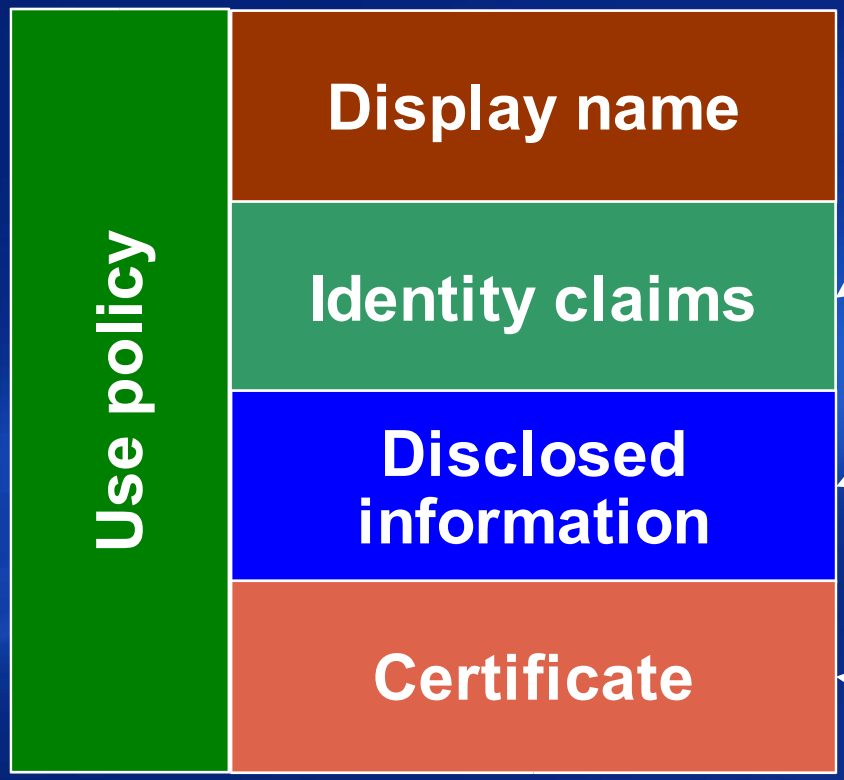
Identity

Matthew J. Dovey
Oxford University Computing Service

The Identity System

- Ubiquitous store, development platform for applications that consume identity
  - Built on “WinFS” storage subsystem (CLI201)
  - Schema for unified representation of identity
  - API with specialized types, methods for principals
- Provides *recognition* between principals
  - Bootstrap and manage recognition between people, computers, groups, organizations
  - Extends Windows security services, can be used by existing applications
- Principals can be serialized, exchanged using document we call an *”Information Card”*

What is an Information Card?



Unique identifier(s)

- For a person: email address
- For organization: web site

Data I choose to disclose

- Home address
- Phone number

Public key certificate

- Local account: self-signed
- Domain account: signed by CA in Active Directory

- Exchangeable identity statement allowing verification of signature

How Are Information Cards Used?

- Information Cards are used to manage *secure digital relationships* with people and organizations
- When an Information Card is imported, it becomes a contact in the contact explorer
  - Can be recognized using Windows security services (SSPI)
  - Can be granted access to shared spaces
- Will seek broad adoption of Information Card, encourage others to implement

Person · Details

My Computer > Person Details > Lisa Jacobson > View Credentials

Lisa Jacobson

- Save as contact
- Start a conversation
- Email this person

contact information

Name **Lisa Jacobson**
Phone **(206)555-1234** home
Phone **(206)555-5678** work
Email Address **lisa@fabrikam.com**
IM Address **ljacobson@hotmail.com**

work information

personal information

credentials

Name **Lisa Jacobson**
Windows Identity Recognition Number **732 AB-S4-127**

notes and keywords

- **Details**
- Recent communication (2)
- Related files and links (1)
- CRM plug-in

other places:

- All contacts
- Boeing

Jeff Henshaw
Chicago

People
- Lisa Jacobson
- Stephanie Con
- Mark Faeber
- Amy Rusko
- Jose Lugo
- Karen Berge
- John Arthur

Calendar

Inbox
- Lab results: Le
- Friday consult
- Current statisti
- Application re

Task Alerts
- Roger Lengel x
- Review Richard
- Practice mainte

Schedule Alerts
- 11:15 Appointi
- Rescheduled:

Online

U2 - One more voice

start 11:31 am

Identity-Based Host Firewall

- Only people you recognize and to whom granted access can make inbound connections to your computer
- Other callers see IPSEC negotiation port, nothing else
- Greatly reduces exposed attack surface of a Windows computer on a network

Authentication Versus Authorization

- Accepting an Information Card does not grant a contact access to the computer
  - Recognition only – clear separation of authentication, authorization
  - A contact must have no implicit access
- To revoke someone's access to computer
  - Remove from access policies on resources
  - Optionally, delete contact object, no longer recognize that person
- E.g.
  - Person to Person - WinFS Sync with "Castle"s
  - Person to Organisation
  - Organisation to Organisation

Tracking Disclosed Information

- Identity system tracks Information Card disclosure
  - To whom Information Cards were sent
  - What information was sent
- If information changes, can selectively or automatically send updates
  - Updates signed thus known to be from you, can process automatically at destination
  - For example: your mailing address changes – automatically update magazine subscriptions

Roaming

- Within home: “Castle” replicates data
- Within organization
  - Credentials, data stored in Active Directory
  - Download to Identity System on clients
- To arbitrary other computers
  - Identity system data can be backed up, encrypted, and stored in vault in “cloud”
- Can also use combination smartcard storage “dongle” for any of the above

Identity Loss and Recovery

- What happens if your computer dies?
  - If a “Castle”, data is on other computer(s)
  - Or, restore from system backup
- Mechanisms used for roaming can also apply to recovery
  - Upload from smart dongle
  - Download from vault in cloud or from Active Directory

Identity Theft

- What if computer, smart dongle is stolen?
  - Send signed revocation message to people you have sent an Information Card
  - If backup in cloud vault, service could send revocation for you, like canceling credit card
  - Bootstrap replacement identity using disclosure information from backup
- How know if identity has been stolen?
  - How discover this today? For example, by checking credit card statement
  - May need similar mechanisms online

Longhorn

Trustworthiness and Security

Matthew J. Dovey
Oxford University Computing Service

Trustworthy Commitment

- Microsoft Cultural Shift
  - Thousands of hours spent in security reviews on .NET Framework to date
  - Foundstone, @Stake security reviews
- “Hardening” the .NET Framework
- Making Security Easier for Customers
  - Prescriptive Architectural Guidance
  - Feature changes in .NET Framework

SECSYM: Security Symposium
ARC340: CLR Under the Covers: .Net Framework Application Security

Right Privilege At The Right Time

- User accounts (Only two account types)
  - Normal users runs with least-privileged
  - Admin users runs with least-privileged
    - Admin applications need privilege elevation
    - Only trusted applications get to run with elevated privilege

Trust Application Execution Overview

Trust Evaluation Process

- Code validation is a human decision
  - Authenticode signed manifests
    - Certificate in the store
  - Domain administrators signed
    - Deployment manifest
  - Local administrators blessed
    - All machine have a signing key
- Default behavior changed by policy

Security: The Sandbox (SEE)

- Apps run in secure sandbox by default
  - Similar to IE javascript
  - Ensures applications are safe to run
- Increased sandbox size
  - “Longhorn” > “Whidbey” > .NET V1.1
- VS helps author for the sandbox
  - Debug in Zone
  - PermissionCalc
  - Security Exception helper

Security: Sandbox Restrictions

- Some apps need more permission
  - Un-managed code access
    - Export to Excel or any MS Office integration
  - Un-restricted file access
  - Un-restricted network access

Security: Policy Deployment

- Application level policy
  - “Trust this app”
  - “App” defined by it’s app manifest
  - Baked into core CLR security
- Trust Licenses
  - License issued by admin, deployed with app
  - License indicates admin says app is trusted
  - Requires only one-time (ever) client touch
    - To configure trusted license issuer

TrustManager

- Decides if app needs additional trust
  - Requested permissions beyond default
  - No previous trusted version
  - No admin policy
- Display user prompt if necessary
- ITrustManagerConfig
  - Control when / how prompting happens

User Consent

- Admins should make trust decisions, but…
  - Not always possible
  - Home users are their own admin
- Users make trust decisions all the time
  - Putting a CD in their computer
  - Installing software
  - Submitting a Credit card to a web page

User Consent Design

- App request permissions needed
  - Requests specified in app manifest
  - VS helps identify needed permissions
- Prompt is simple & binary
  - Happens at install / 1st launch
  - Combined Install & Trust Prompt
- User prompted if:
  - App needs permissions above the sandbox
  - Admin has configured to allow prompting

Code Access Security (CAS)

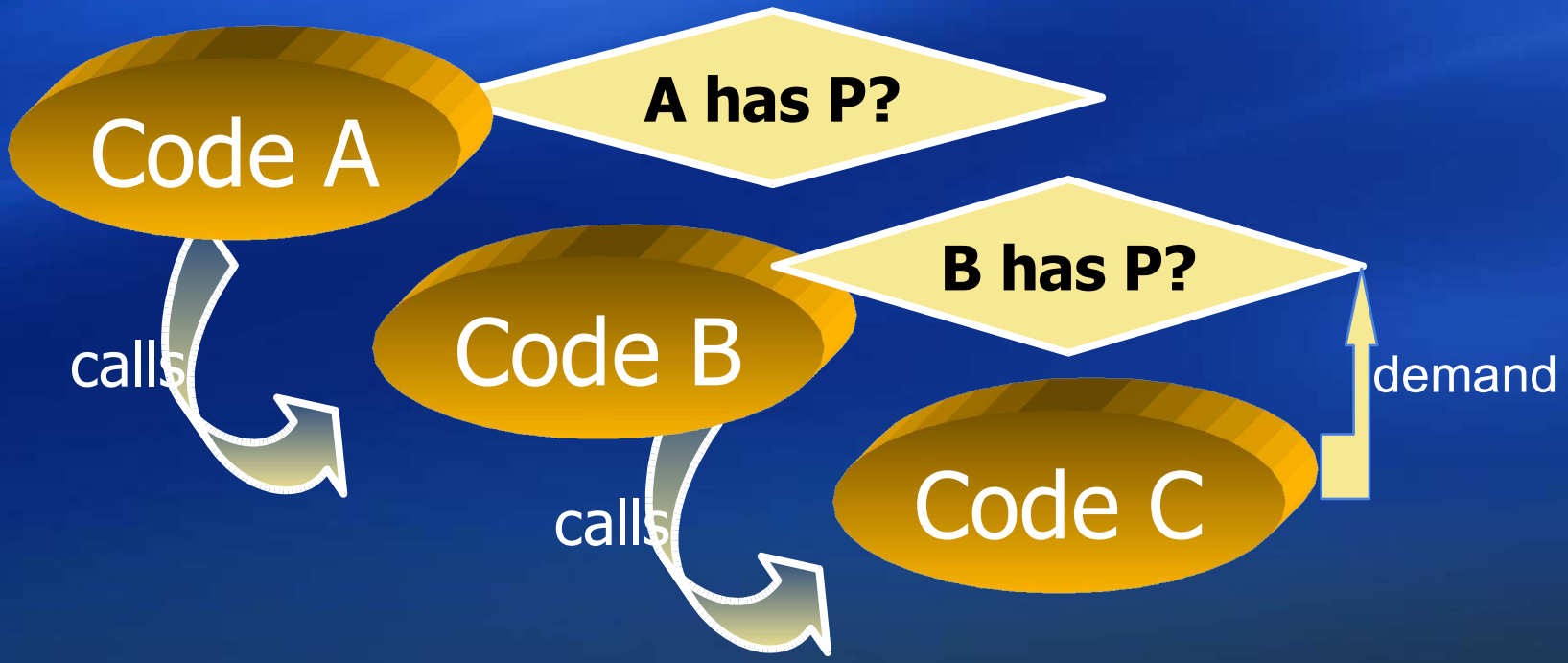
- Based on trust of *code*
  - Recognizes that trusted users (e.g. admin) run less trusted code (e.g. browsing the web)
  - System intersects rights of code with rights of user = 2 levels of defense
- Key features
  - Evidence (location, signature, etc.) is combined with policy to grant permissions to code
  - Protected operations require permissions
  - All callers must have permissions so bad code cannot “trick” good code and be exploited

CAS: How It Works

- Managed code *verifiably* robust
  - No buffer overruns! No unsafe casting!
  - Only well-defined interactions (no ptrs)
  - Components can protect their interfaces
- Trusted libraries as security gate keepers
  - Before doing a protected operations, library *demands* permission of its callers
  - Stack walk – all callers must have permission to proceed; otherwise exception prevents it
  - When demand succeeds the library can override (Assert) and do the operation safely

Code Access Security (CAS)

- Demand must be satisfied by all callers
  - Ensures all code in causal chain is authorized
  - Cannot exploit other code with more privilege



What Is The Secure Execution Environment?

- A new platform for secure applications
- Code written to the SEE is inherently more secure because only safe operations are possible within it
- Security restrictions are enforced by CLR
- Permission Elevation is possible in a declarative and predictable way, and there is a user experience.

- The SEE is simply a default grant set of Code Access Security permissions

Why Code To The SEE?

- Deploy without Trust Dialogs!
- Reduce test surface
- You know that your code cannot harm users machine
- Reduce TCO
  - Business: admin doesn’t have to worry about what the code might do.
  - Home: SEE app cannot harm your machine

Why The SEE Is Safe

- SEE applications
  - All code has only limited safe permissions
  - Can only use SEE-approved trusted libraries
- Security principles
  - Code can further restrict self to least privilege
  - Application isolation
  - Library code is limited to a known safe set

Limited User Account(LUA)

Protected Admin (PA)

Application Impact Management

LUA Problem Statement

- Running with elevated privilege leads to disasters
  - One reason why viruses can cause damaged is because too many people run with full privilege
  - Wash Post even is telling us to run without privilege
  - Every Admin tells us they want to limit users, but…
- Most people demand to run as admin because:
  - Rich web experience, dependant on ActiveX installation, currently requires admin privilege
  - “If we don’t run as admin, stuff breaks”
  - Testing is really easy when everyone’s an admin!
  - Everything works including malicious code!
- Customers want tools and help
  - “Please help us to get applications that run with Least Privilege”
  - Win98 & XP users are admin, so apps are built for admin
  - This is the vicious circle that we must break

LUA – The Good And The Bad

- Long term: we will greatly improve the TCO and “Secure by Deployment” story with Limited User
- LUA apps have no legitimate reason to ask for admin privilege
- Good LUA apps do not try to change system or domain state – they work on XP today as LUA
- Bad LUA apps (the majority) inadvertently change system state
- Short term: some LUA apps will not be fixable by Application Impact Management
  - The target is to have only 20% of apps in this category
  - The expected behavior is that these apps will fail for Longhorn

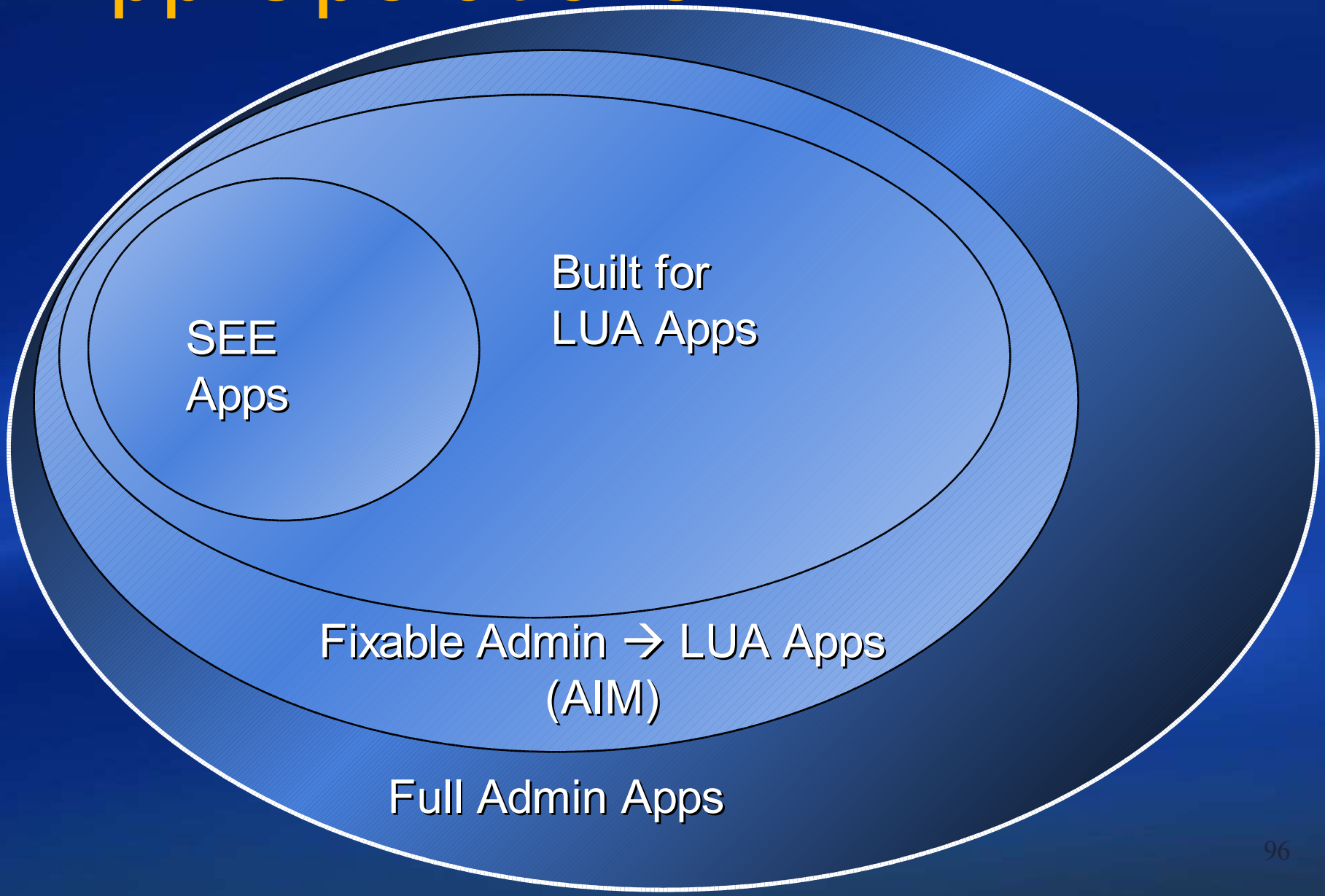
Three Customers For LUA

- Fully locked down corporations
  - Lots of research shows that the enterprise admin wants this feature
  - Reduce security threats
  - Reduce number of apps loaded
  - Reduce TCO
- Admins that need a safe place to run apps
  - Should have the least privilege needed by app
- At Home where the admin wants to increase security
  - Parental controls, so that the child uses only age-appropriate apps
  - User self lockdown to protect PC from security problems

LUA In Longhorn

- All applications will have a manifest listing the application parts
  - Enabling Windows to provide a safe environment for the application to run.
  - All applications will undergo a Trust Evaluation
- Contain applications to limit potential damage
- Create Compartments where code can run
  - Least-privileged User Account (LUA)
    - Most apps can run with user privileges in user space
    - Apps run in LUA space by default in LH
  - Admin Privilege (Protected Admin)
    - Only trusted applications will run with admin privilege in admin space
    - Admins will not enable PA if LUA is not useful

App Operations



Code Validation Process

- All code validation is a human decision
  - Publishers can get signed app manifest (need to be in cert store)
  - Domain admins can sign deployment manifest (enterprise store)
  - Local admins can “bless” apps
  - By policy user can decide to change default behavior
- All local validation decisions are preserved in App Context
- Code Integrity is assured by checking every .EXE and .DLL for validity
- Application trust is assured at Runtime

Application Impact Management And LUA/PA

- All system impact changes are logged for potential rollback on uninstall
- LUA & Admin apps will have their impactful registry writes monitored as well
- Apps are given their own view of certain files & regkeys

User Experience Goals

- Longhorn is Secure by Default yet the system is as flexible and easy to use as Windows XP
- Users know when they are about to do something potentially unsafe and are able to make an informed decision
  - Longhorn always gives strong Security recommendations
  - Users can undo damaging changes
- Users feel confident they can install or run any program without compromising their data or their PCs
  - They feel that, compared to previous versions of Windows, Longhorn is much safer.
  - They trust Longhorn more than any other OS
- Users do not need to learn any major new concepts or procedures to be protected

Other Big Changes

- Winlogon is being rewritten for Longhorn
  - Addressing reliability issues - too many unnecessary processes in Winlogon
  - Addressing performance issues - too many unnecessary components loaded in Winlogon
- Winlogon in Longhorn will no longer support replaceable GINAs, new mechanisms provide existing functionality
  - New, simpler Credential Provider model
  - Eventing mechanism
  - Stacking/chaining

Longhorn

Next Generation Secure Computing Base

Matthew J. Dovey
Oxford University Computing Service

Next Generation Secure Computing Base Defined

- Microsoft's Next-Generation Secure Computing Base (NGSCB) is a new security technology for the Microsoft Windows platform
  - Uses both hardware and software to protect data
  - Offers new kinds of security and privacy protections in an interconnected world

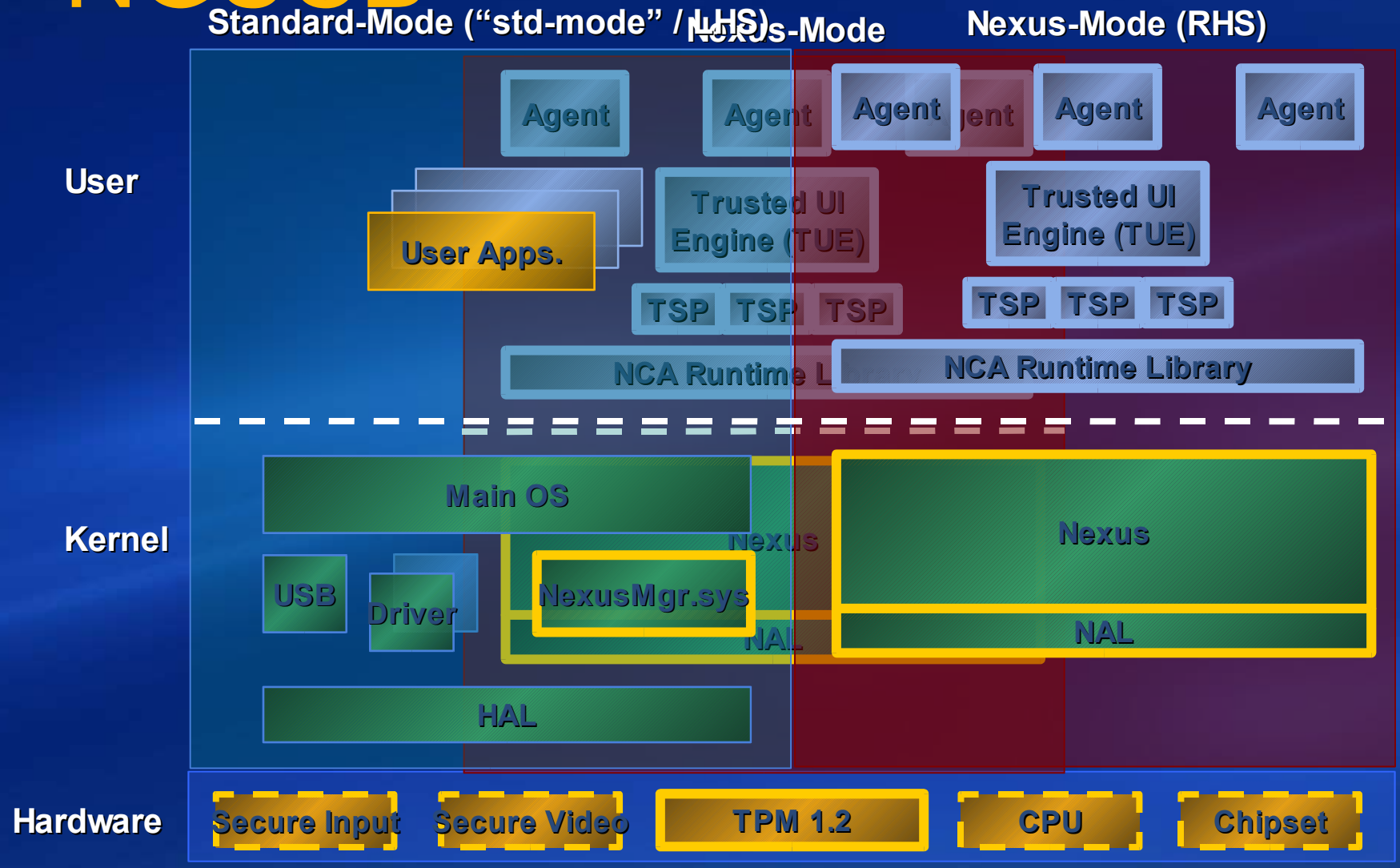
Threats Mitigated in V1

- Tampering with Data
  - Strong process isolation prevents rogue applications from changing our data or code while it is running
  - Sealed storage verifies the integrity of data when unsealing it
- Information Disclosure
  - Sealed storage prevents rogue applications from getting at your encrypted data
- Repudiation
  - Attestation enables you to verify that you are dealing with an application and machine configuration you trust
- Spoofing Identity
  - Secure path enables you to be sure that you're dealing with the real user, not an application spoofing the user

Version 1 Details

- Fully aligned with Longhorn
  - Ships as part of Longhorn
  - Betas and other releases in synch with and delivered with Longhorn's
- Focused on enterprise applications
- Example opportunities:
  - Document signing
  - Secure IM
  - Internal applications for viewing secure data
  - Secure email plug-in
- Hardware based on
  - Trusted Computer Group (https://www.trustedcomputinggroup.org/home)
  - Memory protection (AMD and Intel Prescott CPUs)

NGSCB


Standard-Mode ("std-mode" / LHS)
Nexus-Mode
Nexus-Mode (RHS)
User
Kernel
Hardware
Agent
Agent
Agent
Agent
Agent
User Apps.
Trusted UI Engine (TUE)
Trusted UI Engine (TUE)
TSP
TSP
TSP
TSP
TSP
TSP
NCA Runtime L
NCA Runtime Library
Main OS
Nexus
Nexus
USB
Driver
NexusMgr.sys
NAL
NAL
HAL
Secure Input
Secure Video
TPM 1.2
CPU
Chipset

Nexus Mode Environment

- Basic Operating System Functions
  - Process and Thread Loader/Manager
  - Memory Manager
  - I/O Manager
  - Security Reference Monitor
  - Interrupt handling/Hardware abstraction
- But not a complete Operating System
  - No File System
  - No Networking
  - No Kernel Mode/Privileged Device Drivers
  - No Direct X
  - No Scheduling
  - No…
- Kernel mode has no pluggables
  - All of the kernel loaded at boot and in the PCR

NGSCB Features

- All NGSCB-enabled application capabilities build off of four key features
  - Strong process isolation
  - Sealed storage
  - Secure path
  - Attestation
- The first three are needed to protect against malicious code
- Attestation breaks new ground in distributed computing
  - “Subjects” (software, machines, services) can be securely authenticated
  - This is separate from user authentication

Summary

- NGSCB ships as part of Longhorn
- NGSCB is a combination of
  - New hardware which creates a secure environment for…
  - …A new kernel, called the Nexus, which…
  - …Will run agents in a secure memory partition, and which…
  - …Will provide these agents with security services so that they can…
  - …Provide users with trustworthy computing
- Remember that:
  - When the Nexus is turned off, literally everything runs just like before
  - When the Nexus is on, the LHS runs very close to everything that ever ran
  - The Nexus makes no claims about what runs on the LHS
  - The hardware should run any Nexus, and give full function to any Nexus (with, at most, an admin step by the user)
  - The Nexus will run any software the user tells it to

Longhorn

Questions

Matthew J. Dovey
Oxford University Computing Service

Sources

- Longhorn Development Centre
  - http://msdn.microsoft.com/longhorn/
- Trusted Computer Group
  - https://www.trustedcomputinggroup.org/home