IBM Software Group

Enterprise Application Transformation

There is GOLD in Legacy …

ON DEMAND BUSINESS

Fall 2006

Agenda

- The Challenge: Legacy Applications
- Application Transformation and Modernization Process
- Enterprise Generation Language Overview & Demo
- Specific Transformations Solutions
- Success
- Resources
- Summary
- Q & A

Legacy Application

A computer system or application program which continues to be used because of the cost of replacing or redesigning it and often despite its poor competitiveness or compatibility with modern equivalents. The implication is that the system is large, monolithic and difficult to modify.

Source: The Free On-line Dictionary of Computing 1993 - 2005 Denis Howe

What are legacy systems?

- Applications that [redacted] business process change
- Databases that [redacted] data access
- Platform technologies that [redacted] business growth
- Languages that [redacted] the ability to enhance, enable and extend
- Proprietary tools that [redacted] integration

Why do we (still) have them?

- The Creature Features
  - Familiarity
  - Functionality
  - Stability
- The Business Edge
  - “Ahead of the pack” functionality
  - Hundreds of man-years of enhancements
  - Satisfied customers, vendors and employees
- Fear, Uncertainty and Doubt (#1)

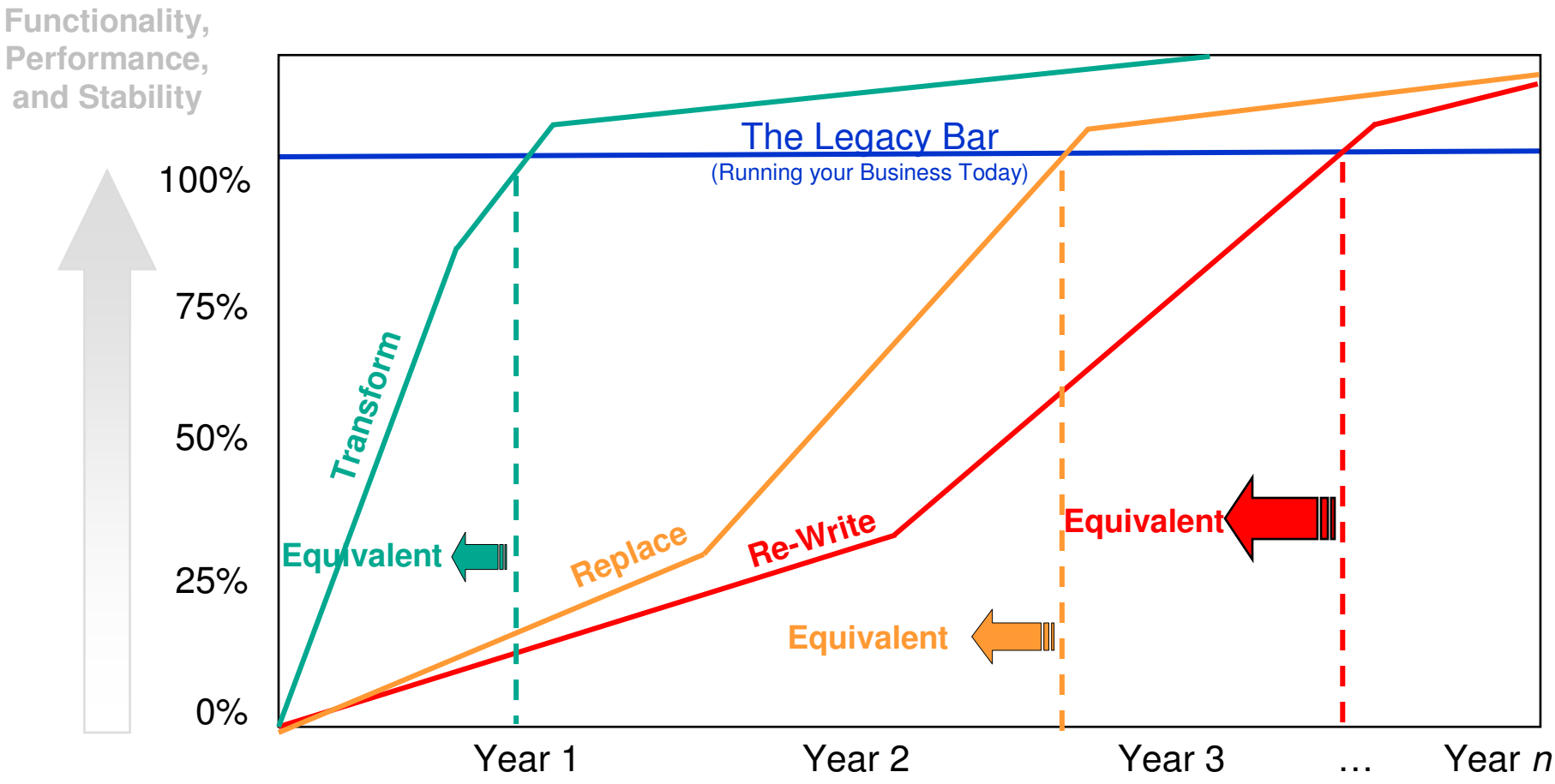
Legacy Systems…

- Are inhibitors
- Are a part of almost every organization
- Are not “broken”
- Are often discussed… and seldom unseated
- Are invaluable to our businesses today
- Can be modernized to a “non-Legacy” status
- Are “Gold” …..

What is application transformation?

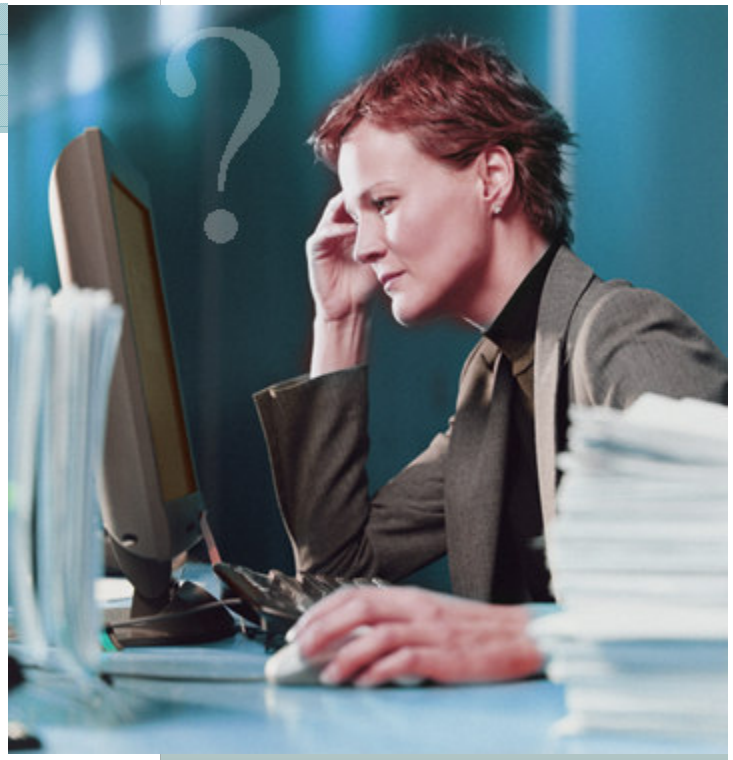
- Modernization of a Legacy Application System
- The elimination of one or more "inhibitors"
  - Applications that inhibit business process change
  - Databases that inhibit data access
  - Platform technologies that inhibit business growth
  - Languages that inhibit the ability to enhance, enable and extend
  - Proprietary tools that inhibit integration
- The move to "re-classify" and extend a legacy application system

The Options: Rip & Replace, Re-Write, Transform & Leverage



Source: ATERAS

Aging Enterprise Applications

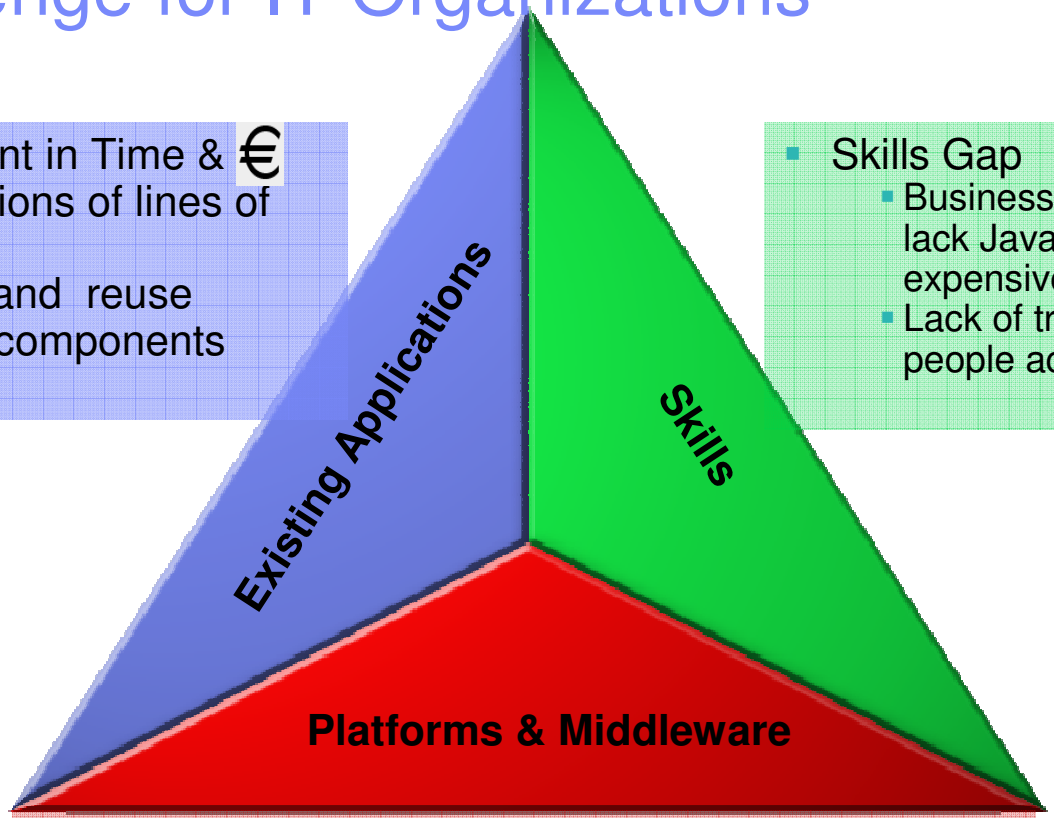


Cost: legacy application maintenance, separate tools for each platform, retaining and attracting talent

- Major investment in legacy applications (time, money, effort)
- This is tested, proven, business-critical code – it runs the business
- Advantageous to reuse (harvest) as much as possible
- Advantageous to leverage existing IT staff/skills

The Challenge for IT Organizations

- Huge Investment in Time & €
- Thousands/Millions of lines of code
- Must leverage and reuse proven, tested components



- Skills Gap
  - Business Oriented Developers lack Java and OO skills - very expensive to attain, very risky
  - Lack of transferability of people across projects

- Ability to support and target the broadest variety of platforms and middleware
- e.g. Mergers/acquisitions, right/down/up size, QoS requirements, strategic redirections, SOA, On-Demand etc.

Who drives the decision?

- Upper Management
- Technical Teams
- Customers
- Vendors
- Users

- **Fear**
- **Uncertainty**
- **Doubt**

What drives the decision?

- Cost
- Time
- Teams

- **Fear**
- **Uncertainty**
- **Doubt**

Risks? What Risks?

- Too much
  - Money
  - Time
- Too many
  - People
  - Problems
- Too little expertise
- Failure

The Benefits

- Eliminate one or more “inhibitors”
- Increase your ability to…
  - Enhance
  - Enable
  - Extend
- Reduce your costs
- Increase development performance

- Ensure your technology future

Applications that inhibit business process change
Databases that inhibit data access
Platform technologies that inhibit business growth
Languages that inhibit the ability to enhance, enable and extend
Proprietary tools that inhibit integration

Measure Tangible Benefits

- Cost reductions
- Development time reductions
- Number of Issues / Issue resolution time reductions
- Application interface improvements
- New technology for existing teams (with negligible learning curve)
- SOA

Set a Transformation /Modernization Strategy

1. Define your Goals
2. Identify the Sponsor
3. Engage all Stakeholders
4. Define what you have Today
5. Define your Target Wish List
6. Measure the Benefits
7. Assess the Risk
8. Mitigate the Risk
9. Select the Solution
10. Define the Project
11. Execute
12. Assess & Reiterate (as appropriate)

Define your Goals

- Same functionality, new technology?
- New functionality, new technology?
- Lower cost of maintenance?
- Ease maintenance and enhancement efforts?
- Increase (developer) productivity?
- Reduce operating costs?
- Increase available resource pool?
  - Break down the “silos” of development

Make your Goals SMART

Specific

Measurable

Attainable

Realistic

Tangible

Identify the Sponsor(s)

- The leader
- The team builder
- The visionary
- The ‘champion’
- The Project Manager(s)

Agenda

- The Challenge: Legacy Applications
- Application Transformation and Modernization Process
- IBM Rational Approach
- Enterprise Generation Language Overview & Demo
- Specific Transformations Solutions
- Success
- Resources
- Summary
- Q & A

Application Transformation

Phase I

DNA – Discovery aNd Analysis

Phase II

Transformation & Education

Phase III

Deployment & Implementation

Define the Project

- Phase I
  - DNA - Assessment and Scoping
    - Inventory
    - Measure
    - Analyze
- Phase II
  - Education and Training
  - Application Conversion and Installation of:
    - Software
    - Database
    - Other software
- Phase III
  - Implementation & Deployment of application(s)
  - Data migration
  - Testing

<<< the longest window of time

- Post-implementation Support

*iterative

Define what you have Today

- Assessment Process
  - Operating Systems and Platforms
  - DNA
    - Application Languages
    - Database Management Systems
  - What is used? What is not?
    - Utilities (backups, archives, integration, replication)
    - Purchased software (reporting, scheduling, mrp, erp)
    - Inter-relationships, inter-dependencies, integration

DNA (Discovery aNd Analysis) Automated Assessment…

- Phase I
  - Must be
    - Structured
    - Repeatable / Iterative
    - Fast
    - Inclusive
  - Result
    - Detail reporting on your application system(s)
    - Provides an ‘Eyes Wide Open’ view
    - Helps size and scope

Automated Application Conversion Process

- Phase II
  - Must Be
    - Structured
    - Customizable
    - Fast (day or weeks NOT months or years)
    - Flexible
    - Repeatable/Iterative
  - Result
    - 95+% automated conversion
    - Remediation process /documentation as required

Technical Validation

- Resulting code, architecture, design
  - Easy to maintain? Understandable?
  - Multi-tiered? SOA oriented?
  - Is it functionally equivalent?
  - Does it require proprietary middleware, licenses?
  - Will it meet your needs?
  - What changes can you make during a conversion? After conversion?
  - Remediation?

Project Management

- Review vendor project management
  - Is the process managed? Or “over the wall”?
  - How are issues handled?
  - What access do you have to vendor expertise?
  - Communication
    - How often do you meet with the vendor teams?
    - How is status measured? Reported?
    - What are the escalation procedures?
- Review your own project management

Knowledge Transfer/Training

- Review the knowledge-building process
  - Is the converted software self-documented?
  - When are your teams educated?
  - How are your teams educated? PoC?
  - Is the training specific to your project?
  - How long will the vendor support you?

What Happens Where?

- Off-site (Phase I & II)
  - DNA (Analysis) of source code
  - Data normalization and relational schema generation (as required)
  - Conversion of source code to EGL (iterative)
  - Unit test of converted application
  - Rerun conversion to handle any remediations discovered (iterative)
- On-Site (Phase III)
  - POC/Education /Training
  - Establish EGL development environment
  - Delivery of new EGL Projects and DBMS schemas (i.e. DB2)
  - Create new DMBS (e.g. DB2) and Tables
  - Load new DBMS Tables (e.g. DB2)
  - Data conversion and migration to new DBMS (i.e. DB2)
  - Systems Testing and user acceptance of EGL application (iterative)

Sample Transformation Schedule

(*actual time frames will vary according to system scope/size)

Phase	Task and Requirements	Owner	Duration*
Pre DNA (Analysis)	▪System documentation ▪Identify data & data structures ▪Test environment (remote access will be required)	Rational EGL EcoSystem team, IBM Services and IBM BPs	3 weeks
Training	▪EGL Training and PoC	Rational EGL Ecosystem team , IBM BP & customer	2 - 3 weeks
Migration Phase I **DNA (Analysis)**	▪Select 4GL Application(s) ▪Perform DNA to determine scope ▪Provide Analysis reports ▪Identify Tool modification (if required)	IBM BPs , IBM Services, Rational EGL Ecosystem team & customer	3 - 6 weeks +
Migration Phase II	▪Actual 4GL Migration to EGL ▪DB2 normalization & load ▪Post Migration Assessment ▪Systems Testing ▪User Acceptance	IBM Services, IBM BPs & customer	3 months +
Migration Phase III	▪Iterative – select next 4GL System transformation candidate ▪Repeat Phases I & II steps	IBM Services, IBM BP & customer	variable

Assess the Risk

- What if the project…
  - Takes too long?
  - Costs too much?
  - Is too difficult?
  - Fails?
- What if the result...
  - Is a poor performing system?
  - Can't adapt to our business?
  - Is a hard-to-maintain application?
- What happens if we *don't* do this project?

Mitigate the Risk

- Find solutions that
  - Use Automated Tools to provide a 95%+/- code conversion rate
  - Address data - new data structure and data normalization – as well as source code
  - Provide an easy-to-maintain architecture/language (the new SDP)
- Request
  - Demo of the new Software Development Platform (SDP)
  - Proof of Concept (POC) of the SDP
  - DNA discussions

Recap: Customer Benefits of Application Transformation

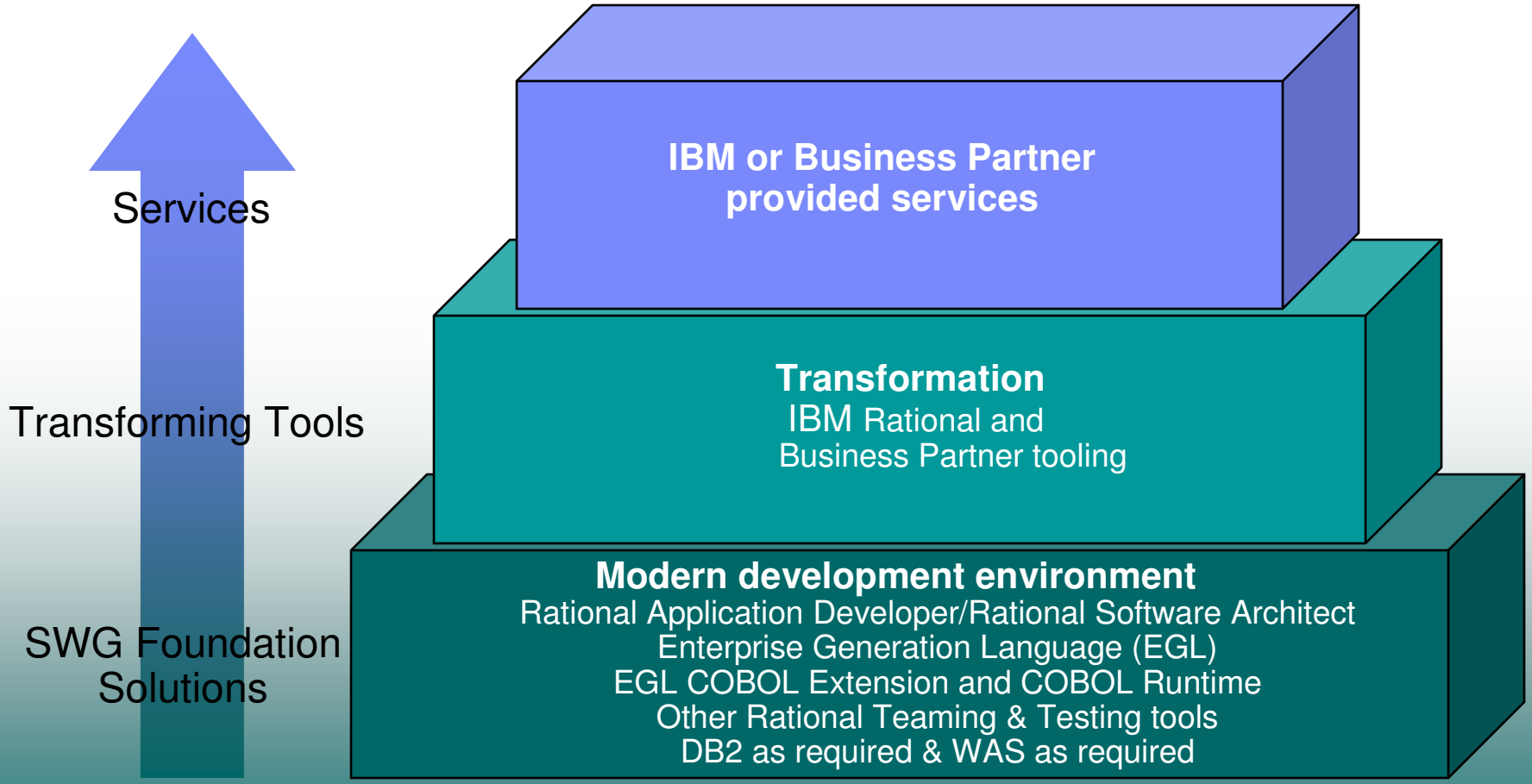
- Reduces the **Expenditure** (time and cost) with eliminating older technologies/platforms
  - **Eliminate** or greatly reduce 4GL software vendor maintenance costs
- **Lowers the RISK** based on **Reuse** of proven functionality migrated from existing core business systems
- **Enables Redeployment** of proven, stable functionality to SOA, model driven or other new architectures
  - Exploit new application architecture – Web Services, Java/J2EE
  - Implement browser based apps with Web / Rich Client deployment
- **Leverage existing development IT staff**
  - 3GL/4GL developers may not be able to migrate to Java/J2EE/OO programming paradigm - too expensive to re-train, too risky
- Move to a **cost-effective, cross-platform** application development strategy which embraces zSeries and iSeries

Eliminate Inhibitors

Agenda

- The Challenge: Legacy Applications
- Application Transformation and Modernization Process
- IBM Rational Approach
- Enterprise Generation Language Overview & Demo
- Specific Transformations Solutions
- Success
- Resources
- Summary
- Q & A

IBM Solution for Application Transformation



IDE: EGL & Rational/WebSphere Development Tools

Rational COBOL Generation for zSeries

Rational Software Architect

- UML 2.0 modeling
- Freeform diagrams
- Visual Modeling
- Apply and Author Patterns
- Asset Browser
- Traceability analysis

Rational Application Developer **

- EJB development
- UML diagrams
- Relational Database tools
- Page & site visual composition
- JSP, Servlet, XML, & Web services tools
- Portlet and Portal tools
- Rich Java client tools
- Integrated Test Environment
- Application testing tools
- Java performance tools
- Rapid simplified development
- JSF/SDO, Wizards, Struts Visual Builder

EGL Java/J2EE Generation

WebSphere Developer for zSeries

- Eclipse based tools for traditional zSeries application development (COBOL, PL1, CICS, IMS)
- EGL COBOL Generation for zSeries

WebSphere Development Studio Client for iSeries – Advanced Edition

- Eclipse based tools for traditional iSeries application development (RPG, COBOL, C, C++)
- EGL COBOL Generation for iSeries

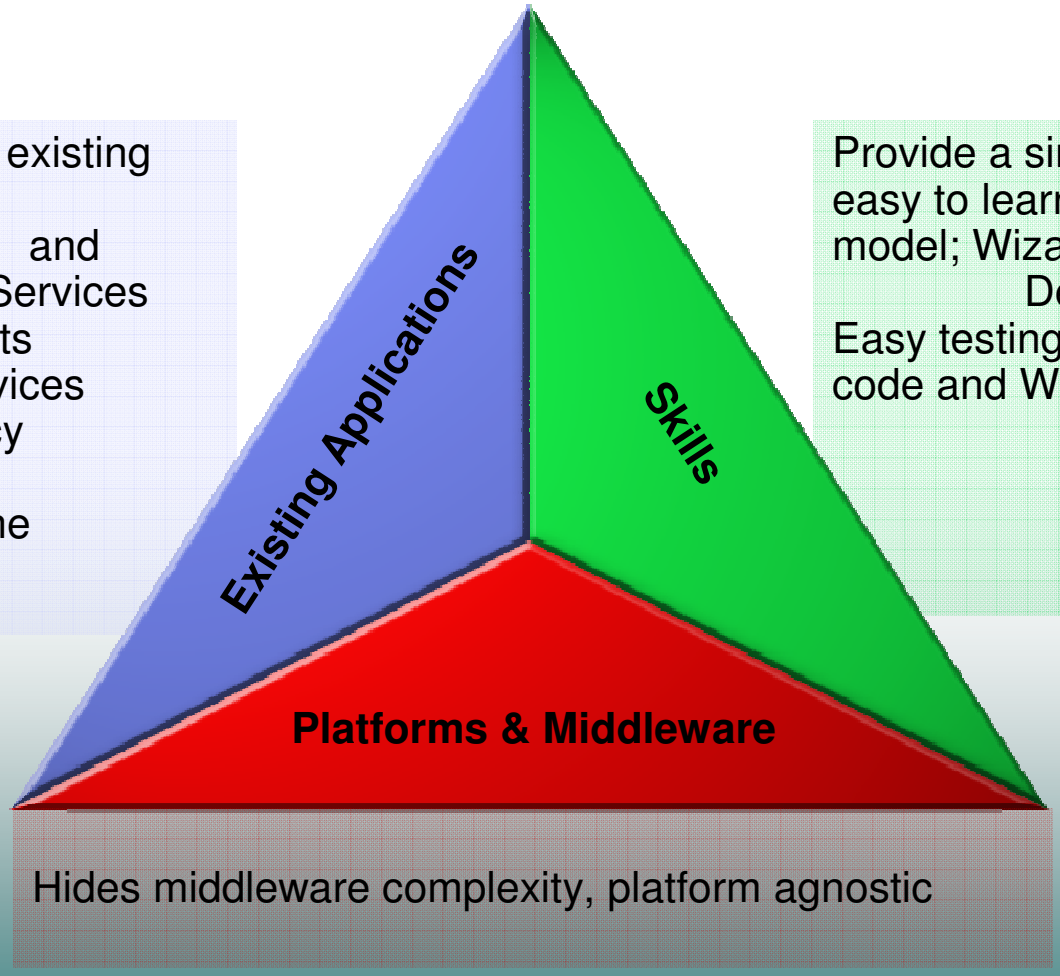
Workbench

IBM's commercially supported version of the Eclipse Workbench

**** RAD chosen as best IDE (Evans Data Corp. March 2006 http://www.evansdata.com/n2/index.shtml***

Enterprise Generation Language is a Critical Component

Breath new life into existing applications
Create Web, EGL, and COBOL Services
Call assets from Services
Migrate key legacy assets
Concentrate on the business Logic



Provide a simple, intuitive easy to learn programming model; Wizard Driven Development
Easy testing of generated code and WSDL

Reasons to use (transform to) EGL

- Bridge the gap between legacy environments and new technology
  - A painless path to SOA and Java / J2EE adoption
- Automated transformations to EGL
  - VA Gen, Informix 4GL, **Natural**, CA Cool:Gen, CA Ideal, CA Cool:Enterprise, HPS/Seer, Progress, RPG, SYNON …
- Current IT staff easily come up to speed on EGL, leveraging their business domain knowledge
  - Improve governance through better resource allocation - portability of developers across projects
  - Lower development and training costs
- Develop applications once and deploy anywhere
- Allow IT to be more responsive to new business needs
- Future-proof development from technology changes
- Eliminate and/or greatly reduce 4GL maintenance costs