

# Using OS Isolation Layers for Two-Step Migration

Janardhanan PS

Systems Technology & Software Division

Hewlett Packard Company

[Contact: [janardh@india.hp.com](mailto:janardh@india.hp.com)]

# Agenda

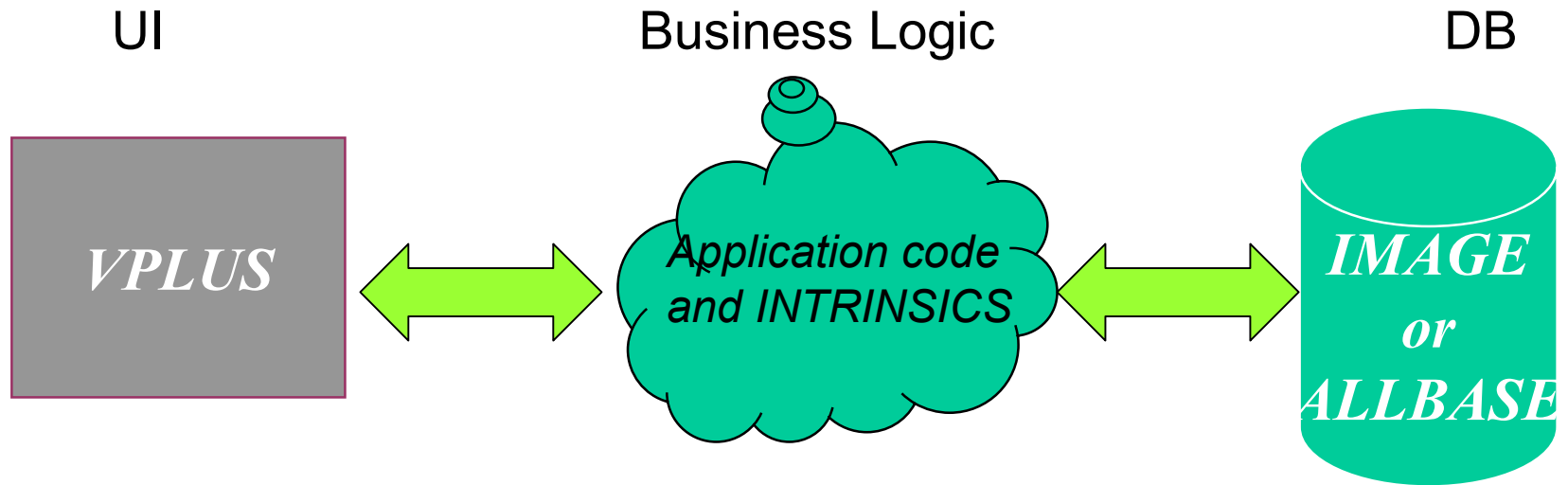
- Background
- Migration Approaches
- Two Step Application Migration
- Benefits
- Q&A

# Scenario

Moving an application from MPE/iX to Unix poses many challenges:

- Complexity of changes
- Time taken to switch
- Resources & schedule
- User & support staff training
- Performance
- Maintenance

# Typical Application Profile



- Written in COBOL using MPE/iX Intrinsics
- OLTP / job oriented
- Uses CI scripts and UDCs
- ISV tools/utilities/4GLs

# Different Migration approaches

- Business Logic Migration
- Application re-engineering
- Application Migration using middle layers
- First by middle layer, followed by re-engineering

# Business Logic Migration

## Re-development of application on target platform

### Pros

- Scalable
- Adaptable.
- Make use of new technologies.
- Sustainable

### Cons

- Huge investments in resources
- May cause end-user frustration

# Migration by re-engineering

## Porting of application to target platform

### Pros

- Application architecture is preserved
- Features native to target platform are utilized
- Easy to maintain & enhance

### Cons

- **Application undergoes drastic changes**
- Requires domain expertise in MPE and target platforms
- Needs simultaneous focus on functionality and performance
- Extensive code changes based on application logic
- Difficult to troubleshoot
- User re-training may be required
- Large costs & schedules

# Migration by middle layers

## Migration by tools that emulate MPE features

### Pros

- Minimum source code changes
- Full functionality at low cost
- Easy interfacing with the target environment
- Quick migration
- Off-the-shelf tools available
- Look & Feel retained

### Cons

- Extra layer between Application and native OS
- Can cause performance degradation
- Under-utilization of RDBMS features
- No perfect replacement for all intrinsics
- Hard to maintain/enhance
- May be expensive to operate

# Migration in steps

Breaks down migration task into 2 evolutionary stages

## Step 1

- Installation of MPE emulation environment /libraries
- Porting of data files
- Testing for **functionality**

## Step 2

- Adapting to native OS in managed pace
- Application re-engineering during adaptation
- Testing for **performance**

# Migration by middle layer

Step 1

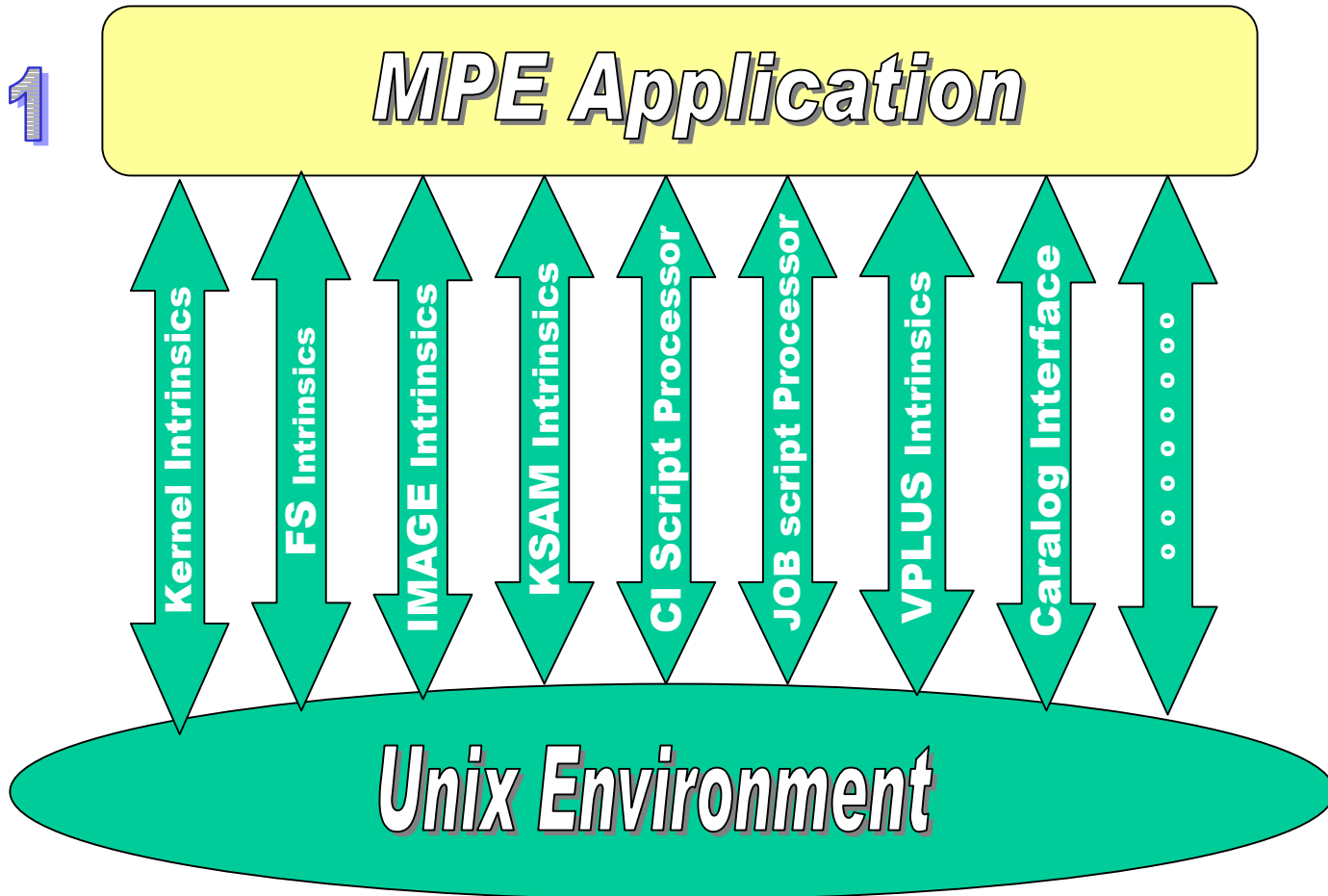
*MPE Application*

*MPE Emulation layer*

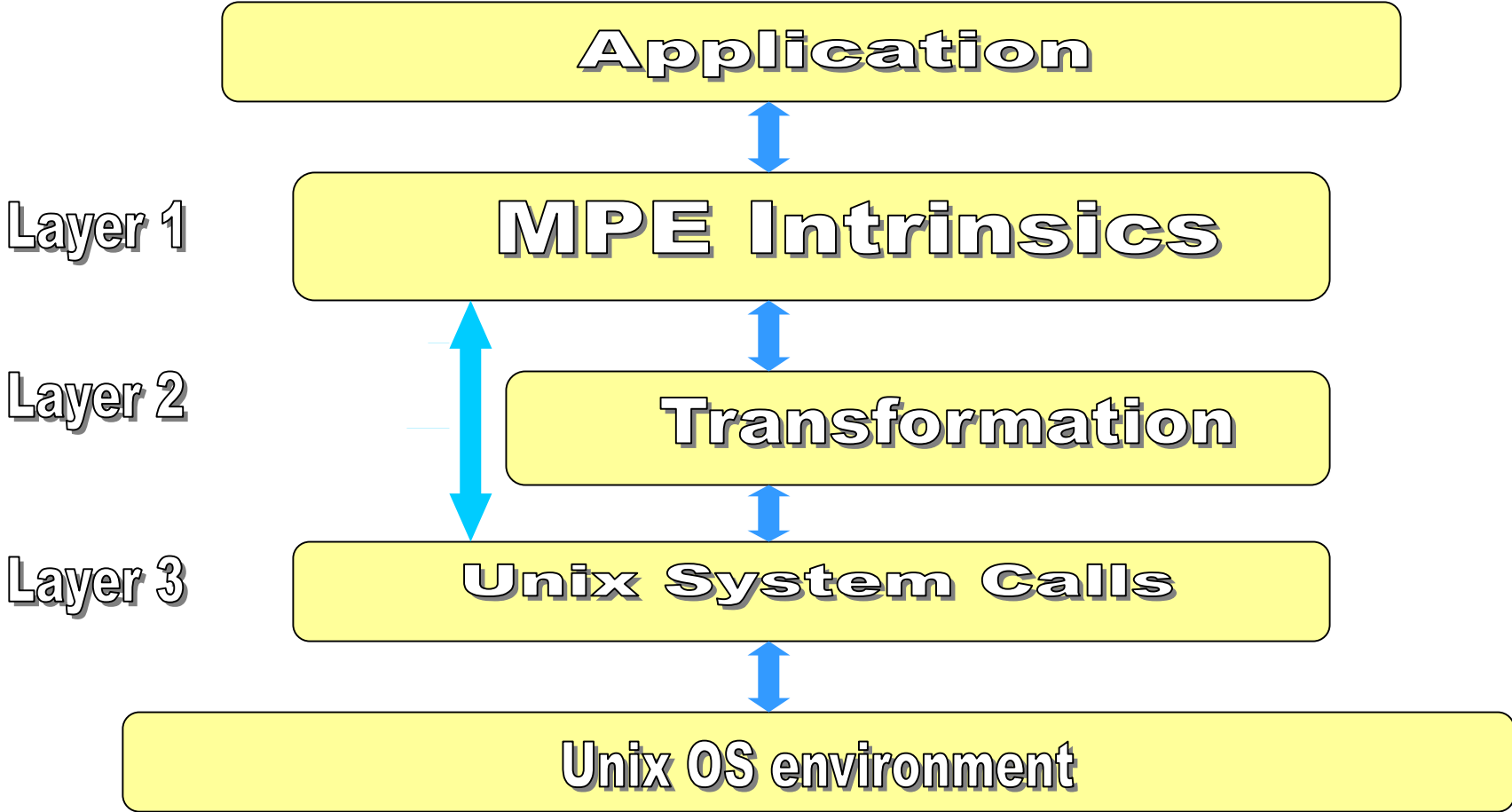
*Unix Environment*

# Inside the emulation layer

Step 1



# Emulation Layers



# Migrating MPE Files

- Features unique to MPE
  - Types: Message files, circular files, relative I/O
  - Domains: NEW, TEMP, PERM
  - User labels
  - CAPS: Multi-level access control and lock-words
  - KSAM Files
- Emulation layer need to support many of these features

# MPE View of Unix Files

- Store MPE file attributes in a separate Label table
- Allow Global Data Pointer Sharing
- File Type Managers
- MPE file system security / lock-words
- CI to give MPE view of files

# Migrating Data

- ASCII and Binary Data files

**Variable record length files need to be converted**

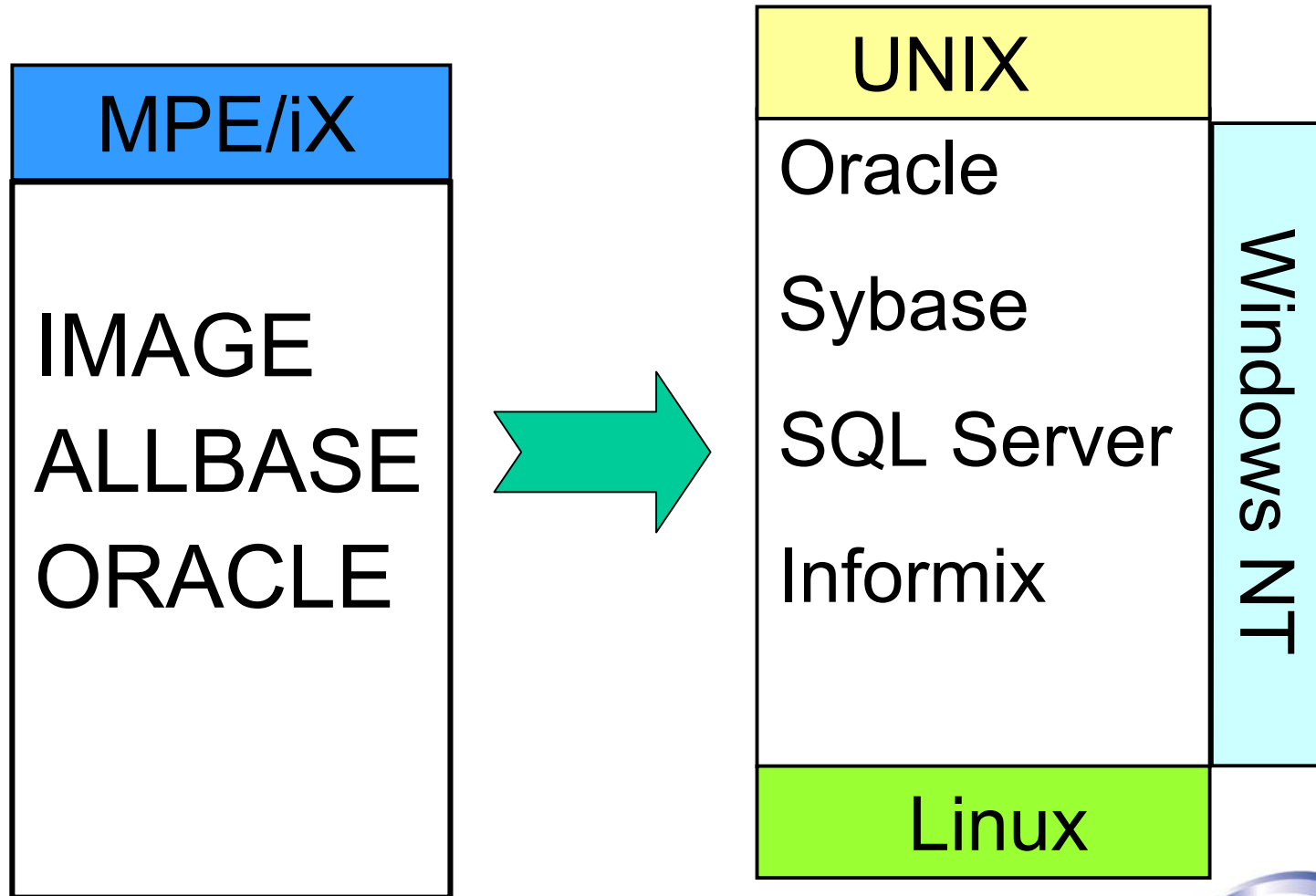
- Database Migration

**Data type compatibility**

**Indexes, Scans**

**UNLOAD/LOAD of data**

# Database Migration options



# HP to MF Cobol Code Translation

## Directly compatible

- Divisions
- Sections
- Paragraph
- ALL symbolic characters
- All Special Character words
- PICTURE Character Strings
- Comment entries
- REDEFINES
- SYNC Clause
- USAGE Clause

## With code Tweaking

- Literals
- Parameter passing to Intrinsics
- Condition code
- Random access files
- Relative files
- Index files
- FD and SD Clauses
- EXTERNAL Clause

# Kernel and FS Intrinsic

Kernel Intrinsic

- Emulate MPE intrinsic
- Support MPE file types & modes
- Support for MPE file attributes

FS Intrinsic

- Wrapper routines to follow COBOL calling conventions
- Return CCODE values in MPE style

# DBMS Support



- Wrapper routines for Image intrinsic to Oracle calls
- Form dynamic SQL statements from parameters
- Return data & status codes exactly as Image would
- Implement unsupported features of Image on Oracle

# KSAM support



- Provide KSAM Intrinsic wrappers
- Maintain user labels in emulated label table
- Support Record access in chronological order
- Enable Native language usage

# MPE Shell on Unix

- Emulate MPE Session and JOB environment
- Support all commonly used CI commands
- Support UDCs and File equations
- Give an MPE view of the Unix files to users
- Allow MPE security features



# Message Catalogs



- Port MPE catalog files as ASCII
- Convert to Unix catalog files
- Access through Unix catalog interface
- Make use of Unix language localization

# UI Migration



## Emulate VPLUS

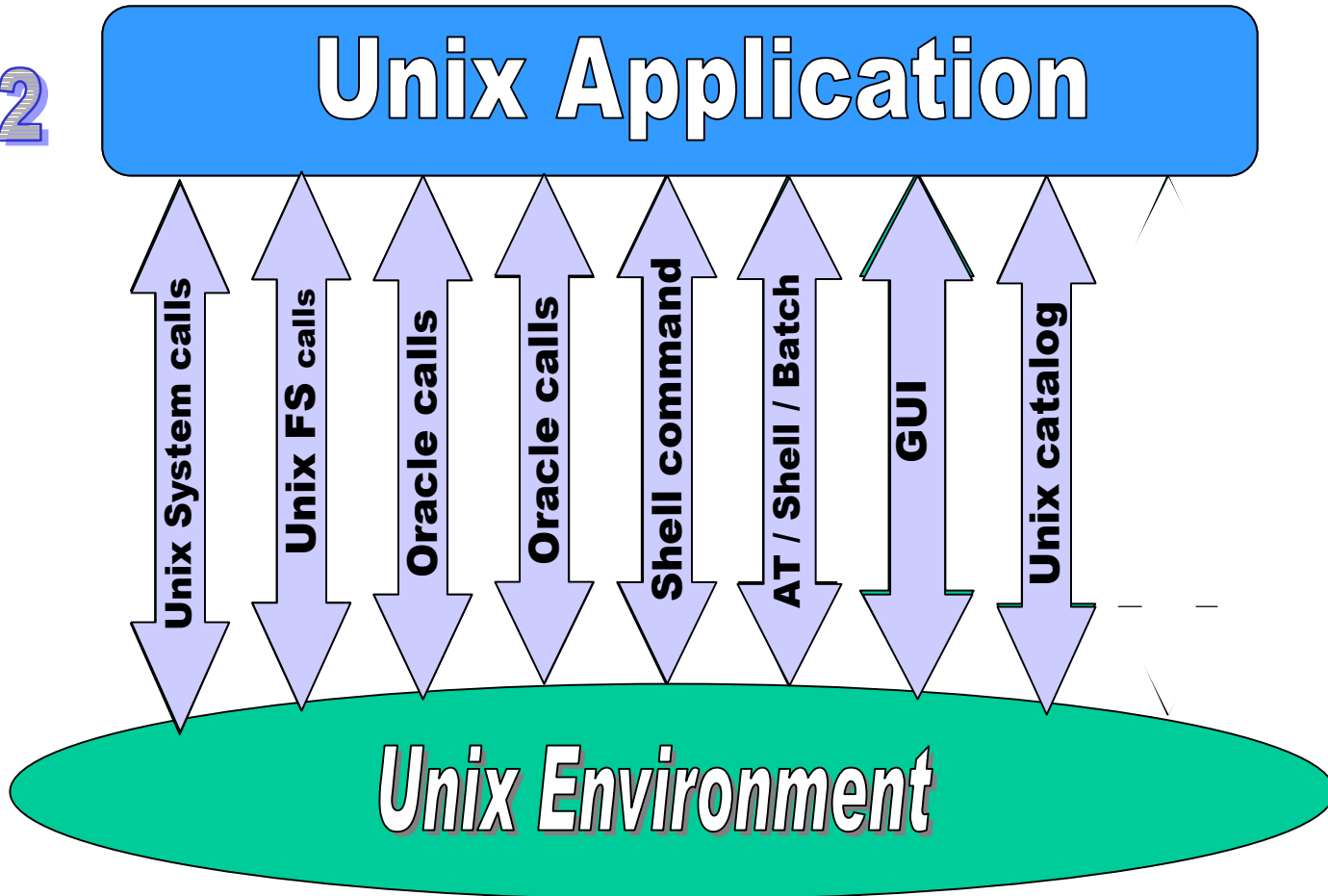
- Allow usage of VPLUS forms files as such on UNIX
- Allow editing of forms files in Unix
- Support all VPLUS intrinsics
- GUI based on MOTIF, Block mode or Curses

## PC Based GUIs

- VPLUS to VB Conversion tools

# Middle Layer Elimination

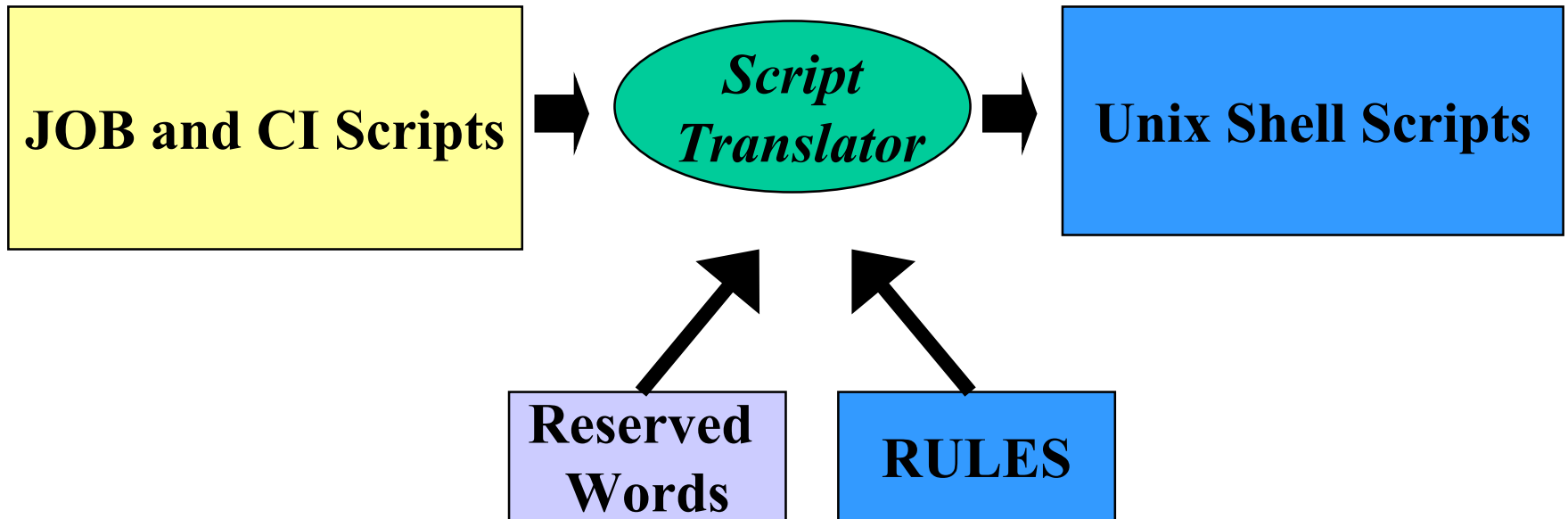
Step 2



# Major tasks in step-II

- Dependency elimination on emulation layer
- MPE intrinsics to Unix system calls
- JOB / CI script translation to Unix shell commands
- Database normalization and application revamping
- IMAGE intrinsics to Oracle calls
- KSAM intrinsics to Oracle calls

# Porting JOB and CI scripts



# Benefits

- Combining the best of two approaches
- Quick migration to target platform
- Minimum User Training
- Dependency on Emulation layer is eliminated in stages
- Co-existence of emulation layer and adapted modules
- Option of staggered IT spending
- Gradual ramp up for support staff on target platform
- Re-engineering can be done with minimum resources
- Cost effective

Q & A

# Happy Migration