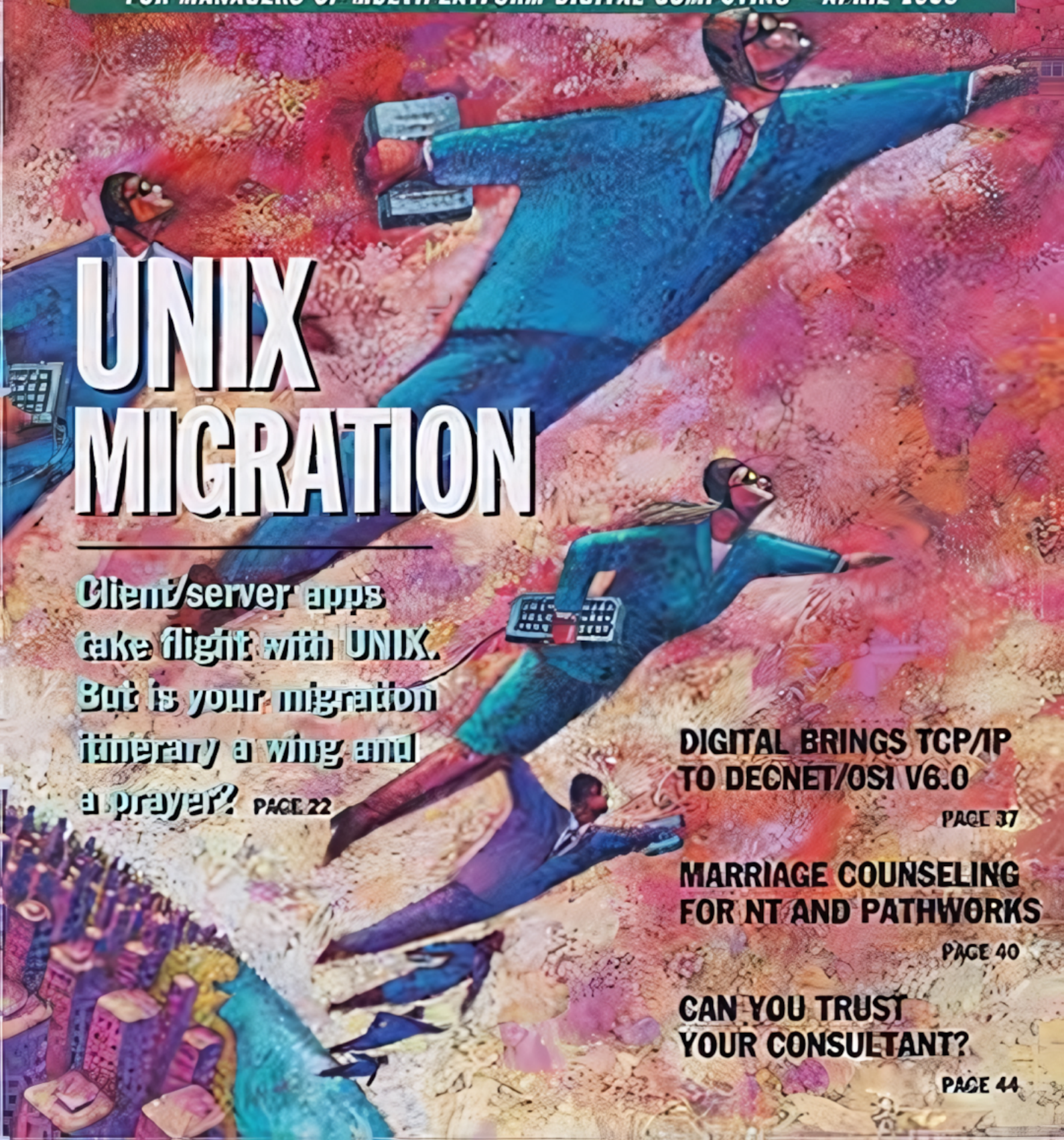
AN INDEPENDENT MAGAZINE FROM CARDINAL BUSINESS MEDIA

DEC Professional

FOR MANAGERS OF MULTIPLATFORM DIGITAL COMPUTING • APRIL 1995



UNIX MIGRATION

Client/server apps take flight with UNIX. But is your migration itinerary a wing and a prayer? PAGE 22

DIGITAL BRINGS TCP/IP TO DECNET/OSI V6.0 PAGE 37

MARRIAGE COUNSELING FOR NT AND PATHWORKS PAGE 40

CAN YOU TRUST YOUR CONSULTANT? PAGE 44

OpenVMS to UNIX:

Performing a Managed MIGRATION

For better or worse, a large number of OpenVMS system managers have been issued a corporate mandate to migrate computer operations to UNIX. Here's how to embark on a program that will efficiently migrate your code, personnel and overall IS resources to a UNIX operating environment.

ZALE TABAKMAN AND DAVID PIKCILINGIS

In companies across the globe, IS managers are looking toward the future with varying degrees of apprehension. In many cases their comfortable world has been disrupted by a mandate to move from their familiar operating system into the brave new world of UNIX.

For most OpenVMS managers the road from OpenVMS to UNIX does not look smooth. A gap exists, the void between the existing OpenVMS business solution and the new land of open systems. The managers' OpenVMS knowledge base seems useless for addressing the problems they could face in the coming months. Familiar commands no longer work. Text editors seem to have been designed to defeat the editing process rather than enable it. System management tools are nowhere to be found. Even HELP fails to provide information. It's no wonder IS managers don't sleep soundly.

But these fears do not have to be fully realized. You can implement specific strategies to help manage the transition from OpenVMS to UNIX. Several companies market tools and services that make the migration of both application code and end users cost-effective and timely. End-user migration tools can also aid in situations where OpenVMS will coexist with UNIX, enabling a consistent user interface (see "Compiling Your Migration Toolset," page 28).

Starting Out Right

Two primary reasons migration projects come in over budget or behind schedule are incorrect planning and insufficient risk management. Incorrect planning stems from using planning skills based in the software development paradigm instead of a paradigm developed specifically for migration projects. Insufficient risk management results from not incorporating the risk components directly into the migration methodology.

Planning — IS groups moving from OpenVMS into the open systems environment require new skill sets. While the goals of most activities are the same as those for any operating system, the specific actions you take are different. Be prepared for changes in the areas of system and network management, accessing the e-mail system and using the editor, for example.

Migration of critical corporate systems requires coordinating many diverse activities, including hardware resources, user documentation, training of the IS staff, and training of the trainers. You also need to address other activities such as beta testing and product release, which may not have been done for years.

Don't forget to plan for intangible issues as well. One striking effect of a migration project is the cultural change the IS staff experiences. OpenVMS and UNIX view the world in significantly different ways, and this can affect morale and productivity.

One item sometimes overlooked in the scramble to open systems is the impact on customers. Mandating a change in operating systems often has the same effect in an organization as mandating a change of religion would have in a community. You must develop a migration plan that is acceptable at all levels of the organization. In addition, you must consider both external and internal customers. Your customers are accustomed to using your software in an OpenVMS environment. Their first question will be, "Are you abandoning us?"

To make the transition as smooth as possible, clearly explain reasons that support the change. Nothing will cause more problems in a migration than an organization that does not fully support the effort. The same people who understand the applications and the process today are the people who can add the most value in porting the applications and migrating the users.

Risk Management — To effectively embark on a migration project, you must develop a risk-management plan. By identifying the problems and outlining strategies to deal with them, you can find solutions to all potential problems. If a demonstrated solution is not available, you may have to develop a worst-case scenario prototype to verify the solution.

There are essentially five stages to a UNIX migration. These stages correspond to five components of project risk management and should allow you to migrate effectively while maintaining and potentially enhancing your existing software and personnel investment. Key to migration is safeguarding your investment in those business-critical applications encompassing endless person-years of corporate knowledge and training while dealing with the proprietary parts of the applications.

A migration project that addresses the five risk-management components upfront will likely be delivered within budget and on schedule (see box, "5 Components of Migration Risk Management"). The methodology is based on five stages, which can run sequentially or in parallel. The purpose and goals of each stage remain the same on all projects; only the size and details change from project to project. The five migration stages are each linked to a component of migration risk (see box, "5 Stages of UNIX Migration"). Solid management and control of each stage can make migration a breeze.

5 Stages of UNIX Migration

1. **ASSESSMENT.** By understanding and detailing the project size and scope, you reduce the risk of the project's going over budget.
2. **PLANNING.** The planning stage provides a technical architecture for software identified in the assessment stage.
3. **PORTING.** During the porting stage, your UNIX system is created. The major effort of this stage is to obtain a working system that can be demonstrated.
4. **VALIDATION.** The validation stage can bring the most long-term value. The final deliverable is your OpenVMS solution on open systems with your users working with it.
5. **PRODUCTIZATION.** The final stage is productization. The productization stage converts an OpenVMS solution running on UNIX into a true open systems solution.

Assessment: Reducing Cost Risk

A complete project assessment is the first stage of any serious migration project. By understanding and detailing the project size and scope, you significantly reduce the risk of the project's going over budget. In the assessment stage, you must develop a detailed understanding of:

1. The existing corporate and computer environment.
2. How software is developed.
3. The technical environment in which the applications exist.
4. The target environment.

The migration process itself provides opportunities for the company to perform a fairly detailed situational analysis and should result in a clarification of its IS goals and direction. You can perform assessments in two steps, a quantitative analysis followed by a detailed risk analysis. No matter what format your assessment takes, at its completion it should account for every work item for the migration, and no unknowns can remain that could drive up the final project price.

Planning: Mitigating Schedule Risk

While the activities of a migration project are similar to those of a development project, the slant is different. First, you

must provide a technical architecture for all software identified in the assessment stage. For any items without demonstrated solutions, you must develop and verify small prototypes.

The second step is to outline specific activities to address the risks identified during the assessment stage. Address these risk items via avoidance, control, assumption or transfer.

Risk avoidance, used at the beginning of a project, removes the cause of the risk. Avoidance may involve descoping the project or changing some significant factor involved.

Risk control, performed during the migration project, involves intensive management intervention. It may include activities such as parallel migration subprojects or frequent management reviews.

Risk assumption involves going ahead as planned, accepting that there is the potential of not meeting the schedule or going over budget.

Risk transfer involves removing the source of the risk from the migration project. This could mean delaying the high-risk elements of the project, outsourcing parts of the project or hiring outside consultants. Although outside consultants may appear to add unnecessary cost to the project, their objectivity and experience joined with the knowledge of the in-house staff can result in substantial savings.

5 Components of Migration Risk Management

1. **Cost.** Will it be within budget?
2. **Schedule.** Will it be done on time?
3. **Technical.** Can it even be done?
4. **Operational.** Will it work?
5. **Support.** Can it be maintained?

The result of the planning stage should be a schedule with achievable, tangible and measurable milestones. The first major milestones will be configuration management and revision control and a reliable build of the application on the target environment. A rapidly moving migration project depends on the ability to have incremental builds on a daily basis.

Remember to plan for testing. Full system testing begins the day a successful build is available.

Porting: Managing Technical Risk

In the porting stage of the project, your UNIX system is created. The major effort of this stage is to obtain a working system that can be demonstrated. At this stage, you implement the solutions identified during the planning stage. The UNIX-based build becomes available; the emulation tools for DCL, FMS, system services, and so on, are integrated; and specific re-engineering tasks are performed.

Also at this stage, you must carefully watch configuration management and version control and the build mechanisms, because they can cause the project to fall behind schedule and be over budget.

This is a terrific time to train people on UNIX, because they can immediately get hands-on experience with an application they are already intimately familiar with. The new UNIX staff can build credibility and become integrated into the corporate culture. The OpenVMS

Migration In the Real World

Sector 7 was recently involved with a migration process that used the stages outlined in the accompanying article. The client had a legacy system involving some assembler, 2.5 million lines of VAX FORTRAN in 10,000 sources files, and 12,000 lines of DCL code. The application included more than 1,200 individual functions users could access. These functions were made available using about 170 images.

The entire application was designed around the strengths of the OpenVMS VAX architecture — for example, the use of the OpenVMS VAX parameter passing mechanism. Certain FORTRAN routines were designed to receive differing types of parameters. For example, you could pass to the same formal parameter an integer, a float or a string. For those not familiar with FORTRAN-77, this is probably considered illegal in conservative parts of the U.S.

The application accomplished this process by having the called subroutine receive a byte structure representing the OpenVMS parameter-passing structure and then decoding the fields. The application was built on RMS ISAM files accessed through the system services, and they needed to deliver their application in Oracle.

The client made two attempts to migrate the application. The first attempt used a software translator to convert the application into C. The target system included hardware with a different byte ordering than the VAX and required all data to be even byte-aligned. Needless to say, this attempt was not successful. After 10 months of the project the CEO went away for 4 days, and when he returned the project was 4 months behind schedule. At this point the company did a major reset on the project and spent weeks evaluating their options.

The result was a changeover of the majority of the project team and the suppliers of the software tools, and a completely new project plan. The new plan followed the methodology described in the accompanying article, and they delivered to the customer a reliable and fully tested system within 6 months.

While this may sound like a nightmare project, from my experience it is typical of most organizations attempting migrations. The wide availability of OpenVMS emulation software and the lack of experience combines to result in projects like this. Emulation tools significantly reduce the amount of rewriting necessary, but they can never replace good project management skills, experience and a solid plan. —*Z.T.*

staff members look to the UNIX staff for information and direction, while the UNIX staff look to the OpenVMS staff for application knowledge.

Validation: Controlling Operational Risk

The validation stage can bring the most long-term value. It should begin when the corporation has committed to the migration project. The final deliverable of this stage is your OpenVMS solution on open systems with your users working with it.

The major advantage of a migration project — the fact that the system is already working — is also potentially its largest problem. A working system provides practical opportunities. Specifically, a complete test suite can be available before the first line of code is migrated to the new system. The flip side is that the use of high-level languages and reliable emulation software hides some of the dangers of a migration.

There are a number of places errors can be introduced during migration:

1. Embedded application errors.
2. The compiler.
3. Hardware.
4. Fundamental differences between OpenVMS and UNIX.
5. Re-engineered code.
6. Emulation software.

However, when failures occur in migrated software, they fail big-time, and there are no new logic errors. The errors are almost always obvious and are quickly found and fixed.

You should employ automated testing with an emphasis on functional and regression testing. The final test set can act as your project acceptance criteria and provide a return on investment long after the migration project.

Productization: Minimizing Support Risk

The final stage is productization. Although it can be difficult to decide when productization is complete, its purpose is clear. The productization stage converts an OpenVMS solution running on UNIX into a true open systems solution. This represents activities that range from dealing with performance issues to adding a complete client/server GUI.

MIGRATION CAN BE THE solution for organizations looking for a low-risk introduction to open systems with a high payback. It is the type of project that is seemingly straightforward but actually quite complex. However, if you exercise proper care, migration projects can be delivered on time and within budget, and they can even be fun. If used appropriately by senior management, they can lead to a rejuvenated IS organization making better use of financial resources.

—Zale Tabakman is Director of Canadian Operations of Sector 7, Austin, Texas. David Pikcilingis is President of Boston Business Computing, Methuen, Mass.

Sector 7 - Migration Tools & Services

Sector 7 - Tools

VX/RMS	- OpenVMS RMS for UNIX
VX/RT	- Over 300 OpenVMS "$" services for UNIX
VX/BASIC	- OpenVMS BASIC transpiler for UNIX systems.
VX/SMG	- OpenVMS SMG for UNIX
VX/FMS	- OpenVMS FMS for UNIX
VX/DCL	- OpenVMS DCL for UNIX
VX/JSP	- OpenVMS QUEUE manager for UNIX
VX/FTP	- OpenVMS FORTRAN to ANSI FORTRAN
VX/CLD	- OpenVMS CLD compiler for UNIX
VX/CDD	- OpenVMS CDD for UNIX
VX/MSG	- OpenVMS MESSAGE compiler for UNIX
EDT+	- OpenVMS EDT for UNIX
VBACKUP	- OpenVMS BACKUP for UNIX
VMAIL	- OpenVMS MAIL interface for UNIX mail

Sector 7 - Services

OpenVMS MACRO to C conversion
OpenVMS RDB to ORACLE / SYBASE
OpenVMS FMS / TDMS / DEC FORMS conversion
OpenVMS COBOL to ANSI COBOL
OpenVMS ACMS conversion
OpenVMS DATATRIEVE conversion
OpenVMS DECNET to TCP/IP
DEC Message Queue conversion
Process Control & Real Time Migration
On-Site and Off-Site Migration Projects
CLIENT SERVER reengineering
Training: UNIX for OpenVMS Programmers
VT to GUI reengineering
Language Conversion (DIBOL, PL/I, PASCAL)
Migration Project Management
OpenVMS to WindowsNT® conversion