Tools and Techniques for Software Productivity Improvement

Mr. C. P. Tantry
Assistant General Manager

Mr. Sujeet Gore
Assistant General Manager

System Integration Division
Tata Infotech Limited
Mumbai 400096
INDIA

1. Abstract

This paper on "Tools and Techniques for Software Productivity Improvement" demonstrates the use of Tools and Techniques to achieve productivity improvement during the project lifecycle.

The project, on which the paper is based, involved the migration of a large proprietary mainframe-based legacy application to an open J2EE architecture comprising industry standard products. The migration was achieved with little or no knowledge of the application functionality.

The paper describes the methodology adopted and provides metrics on improvement in productivity, achieved by continually improving tools and techniques used in the project.

The paper also discusses the lessons learnt from the usage of tools and techniques. The learning experience could be of use to those considering migration of legacy application to new technologies.

It also details the points considered during estimation and describes how to factor productivity improvements during the project. This helps in better resource planning by having people allocated to the project based on the variation of effort required during the project.

In conclusion, effective tool management plays a significant role in the execution of such migration projects. Identification of opportunities for increasing the automation level by enhancing tools should be planned in the project lifecycle.

2. Introduction

Legacy systems are typically large software systems that run on proprietary platforms and contain the core business logic of an organization. CIOs are often faced with the challenges of maintaining such systems in the face of changing business needs. These challenges are compounded by the fact that these systems are often written in obsolete languages. As time goes by, available tools and resources (people, hardware, software and support) become scarce, the risks become more acute, while the cost of migration increases exponentially.

Their sheer size and ingrained business knowledge prohibit re-development from scratch. There have been a number of attempts to provide a solution to this problem but to no avail. From this perspective, migration aids, which help automate part of the transition effort, increasingly assume significance.

When source and target systems are based on similar programming frameworks, for example: procedural or object-oriented, a tool suite that addresses all aspects of migration can decrease the migration effort significantly when compared to manual estimates. The use of tools also significantly reduces the risks associated with timelines, cost and functionality replication.

2.1 The Customer

The customer was a renowned university in the USA, founded in 1909, and one of the oldest and biggest institutions in the USA. It has 10 sister varsities spread across the country. It has a total enrolment of more than 9,000 undergraduate and graduate students.

The University migrated their existing Unisys legacy system, a mission critical application used by more than 600 users, to a Web based client/server architecture on an open platform, based on Intel chips, Windows operating systems, and Oracle 9i relational database.

2.2 Options

Before deciding to migrate to a new technology, the three options evaluated by the university are mentioned in the table.

Options →	**Upgrade**	**Replace**	**Migrate**
Involves	Retain existing environment and upgrade to newer versions of hardware and software.	PeopleSoft (Recommended by a Big 5 consulting firm)	**Move to OPEN architecture** **(Rewrite/Mig-rate decision)**
Cost	Hardware/software upgrade ~ $2 million (Over 4 years) Annual maintenance ~ $500K per year	New Hardware ~ $2 million PeopleSoft Implementation ~ $5 million (Over 4 years)	**New Hardware** **~ $2 million** **Migration** **~ $1 million** **(Over 2 years)**
Total Cost (4 Years)	$4 Million	$7 million	**$3 million**
Timeframes	3-6 months staggered	2 years	**Less than 1 year**
Key Risks	Retaining proprietary platform	Need for BPR to be in line with PeopleSoft Plus Two failed past attempts	**Performance of the migrated system**

The factors that prompted the University to go in for the Migrate option included

- Timeframe for expiry of the mainframe lease
- Benefit in retaining the mature business logic
- Comparative cost/timeframe/risk

- Coexistence of migrated COBOL and Java
- System maintenance is easier and minimal training is required

Additional benefits included

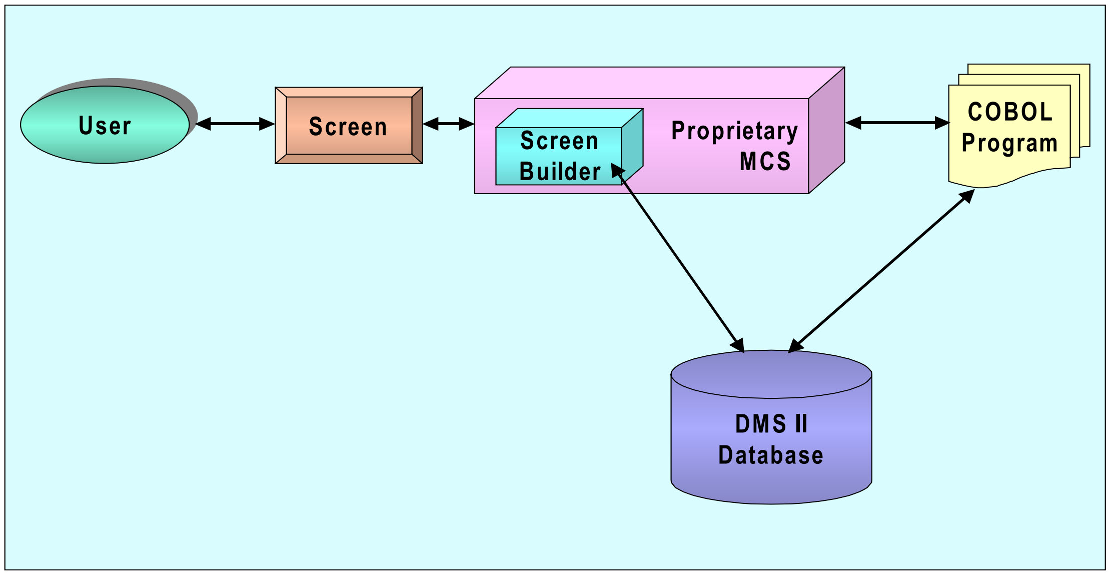
- Migrated application is web-enabled
- Operational cost saving
- Gradual rewrite of COBOL to Java is possible at a pace chosen by the University.

2.3 Legacy Application

The University's legacy application, written in COBOL 74 using the DMSII database, was hosted on a Unisys ClearPath A-Series machine (Model 4601-21). The machine had a dual processor with a memory of 144 MB (24 Mega Words). The operating system was the MCP Release SSR 48.1. The application used a customized ALGOL MCS controller program that handled transaction routing and passing of messages between the user and the system, with more than 600 users using the application.

The application had both online and batch COBOL programs. Message handling was done by MCS and the batch COBOL programs were executed via WFLs (Work Flow Languages).

Existing Architecture:

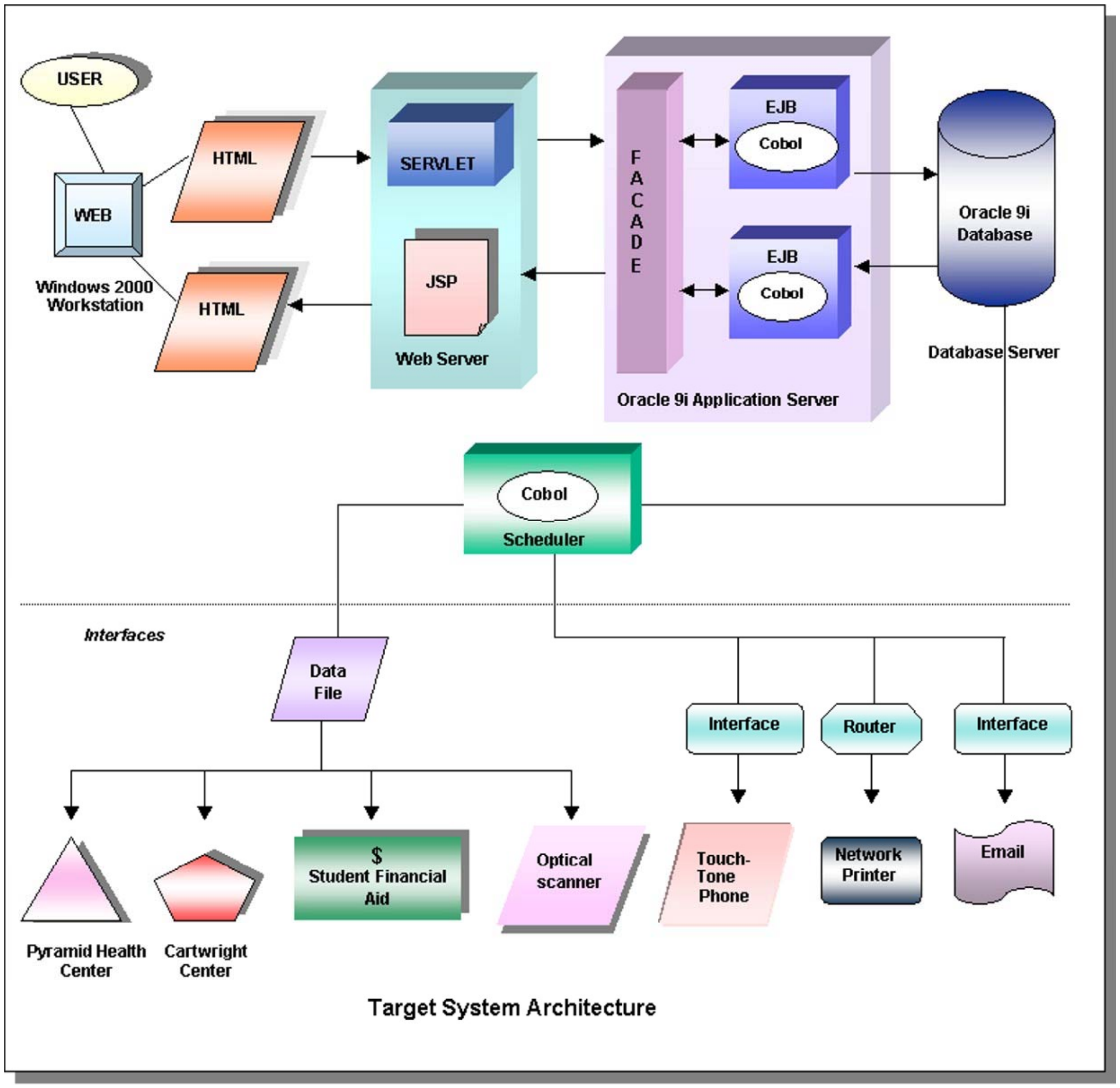


The University had been using this application for more than 20 years. During this period, the application was customized and enhanced continually.

The table below gives an idea of the size of the application.

Type	Number	LOC
COBOL Programs	1,042	810,655
ALGOL Programs	23	16,292
Screens	615	
WFLs	957	16,498
Database Tables	141	
Database Indexes	380	

2.4 Target Environment



Database Configuration

Database Cluster: (2) Compaq DL-580 rack mounted servers

Hardware: 3 Pentium III Xeon 900 MHz Processors

Software: Windows 2000 Advanced Server OS w/ Service Pack 3
Oracle 9i database Enterprise Ed. Release 2 (9.2.0.1.0)
Oracle 9i Rapid Application Cluster and Cluster Guard Release 3.3.1

Application Server Configuration

Hardware: 2 Pentium III Xeon 900 MHz Processors

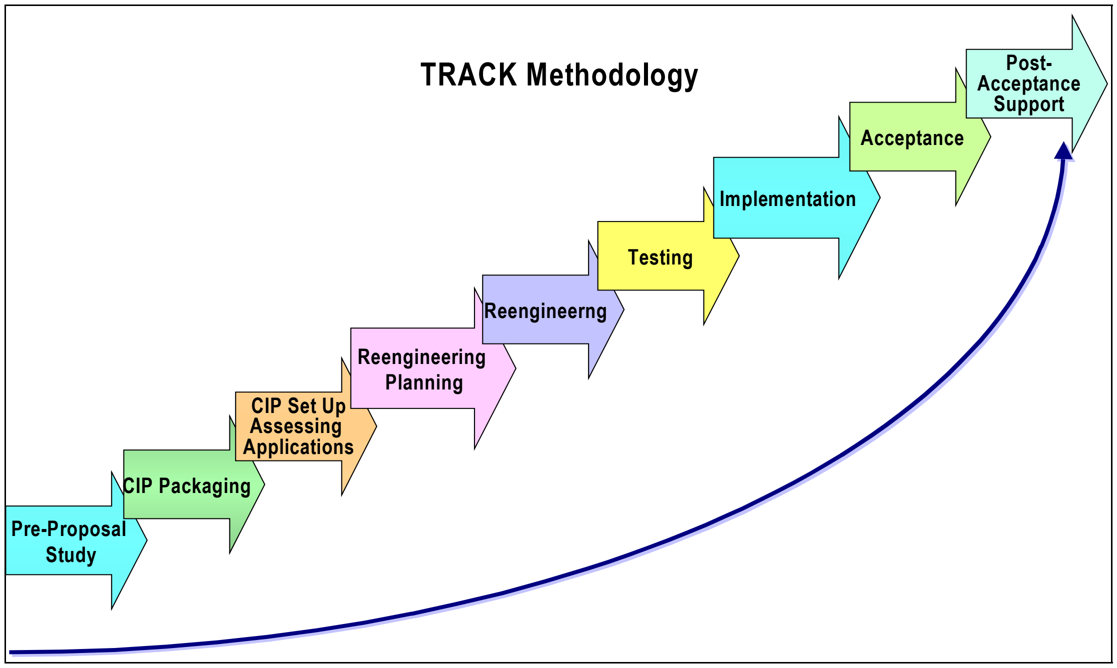
Software: Windows 2000 Advanced Server OS w/ Service Pack 3
Oracle 9i Application Server Release 2 (9.2)

User Interface

Microsoft Internet Explorer 5.5 or higher

3. Methodology

TRACK (Tata Infotech Reengineering and Conversion Kit) Methodology was used for the University migration project. Using this methodology, the migration project was accomplished through a series of well-defined phases, with each phase achieving a clearly defined and measurable result towards the end goal.



3.1 Phase I – Pre-proposal Study

In this phase, an assessment of the customer's system is made with a view to estimating the scope of work. It involves analyzing the genealogy of applications, that is, environments used in the past, tools used, and methods used for any type of migration as well as the people in the organization that support the application.

Tools Used: iTrack

3.2 Phase II – CIP Packaging

In this phase, the inventory of applications to be migrated is collected, based on the scope of work. The Customer Input Package (CIP) is then formed with the source code of applications, related documentation, copybooks, etc.

Tools Used : iTrack
PacketCheck
TokenSearch

3.3 Phase III – CIP Package Setup and Assessment of Applications

In this phase, the CIP package received from the customer is set up, the impacted programs and their lines of code are identified using tools. A study of the existing system design is made to find out if any additional modules need to be developed.

Tools Used : iTrack
PacketCheck
TokenSearch

3.4 Phase IV – Reengineering Planning

Based on the assessment, the strategy for migration is revalidated, and alternative approaches with their advantages and limitations are analyzed in this phase. The Reengineering Plan document, prepared in this stage, describes the assessment observations, standards of migration, change control procedures, etc.

Tools Used : iTrack
TokenSearch

3.5 Phase V – Reengineering

The Reengineering Plan Document forms the basis for executing the migration in a planned manner. Activities like building of the architecture, code migration, screen generation, database setup and data migration etc are done in this phase.

Tools Used : iTrack
ASCOT
HTMLGEN
WFLCON
IOGEN
DATAGEN
CodeWrap
CobolChecker

3.6 Phase VI – Testing

In the Testing phase, test plans prepared are focused on replicating the functionality and are executed in a formal, systematic manner.

Tools Used : iTrack
Rational Test Suite

3.7 Phase VII – Implementation

In the Implementation phase, the target environment is set up and the converted application is installed. This involves setting up the database, runtime environments, and various system parameters.

Tools Used : iTrack

3.8 Phase VIII – Acceptance

Using the Acceptance Test Plan, the customer tests the system in the Acceptance phase. A parallel run of the old and new systems is done. At the end of successful testing, the customer accepts the system and delivers an acceptance letter.

Tools Used : iTrack

3.9 Phase IX – Post Acceptance Support Phase

During this phase, system support is provided to resolve any problems that might arise due to the migration.

Tools Used : iTrack

4. Tools and Techniques

4.1 Role of Tools and Techniques

Tools and Techniques play a pivotal role in automating the activities of a project, and result in obvious advantages. The main advantage is the reduction in effort required to complete a task. An activity, which would have taken much longer to complete manually, is finished in a very short time with the aid of tools and techniques. This leads to increased productivity, as shown in the graph in the Section on Metrics.

Since automation requires fewer manual changes in a program, the risk of errors cropping up due to manual changes reduces considerably. The quality of the migration is improves, with hardly any errors found during testing.

Since the use of tools reduces the effort and timeframe for tasks within a project, tool usage should be considered while estimating the project effort.

4.2 Management of Tools and Techniques

It is imperative that tasks during a project lifecycle are continually monitored for repetitiveness. Repetitive tasks need to be identified and attempts be made to automate them. Thus, projects should plan for enhancing tools.

A project could begin with a requirement to automate certain tasks. Based on whether the baseline version of the tool caters to every aspect of the project need, certain tools may require customisation or enhancement to cater to specific requirements. During the course of the project, more activities may be identified for automation; hence, tools must be enhanced continually to meet these needs.

Enhancing a tool, however, can present a few issues like rerunning the tool on programs that have already been converted manually. If a certain additional change is identified midway during a project, the tool may be enhanced to cater to it. The enhanced tool will work well for programs yet to be converted, but for programs that have already been migrated, one needs to be careful in using the enhanced version of the tool.

Such, and other, issues usually crop up during a project and need to be aptly handled. Depending on the specific case, it may be prudent to create a separate, small utility for the additional need instead of enhancing the main tool. This way separate execution is possible. Tools can have the facility of selectively migrating the different syntax present in the source code.

Management of tools, thus, should constitute an integral part of the learnings during a project life cycle. Milestones should be set in the project plan, where learnings are collected and analyzed to evaluate possible enhancement of tools.

4.3 Third Party v/s In-house Tools

Management often has to decide on a Build v/s Buy option for tools. No one tool, whether developed in-house or third party can meet all the needs of such Migration projects.

During the course of the project, learnings identify repetitive tasks that can be automated. In contrast to third party tools, in-house tools can be enhanced and customized to meet the specific needs of the project. In-house developed tools do not need to consider issues like licensing, cost, vendor dependence for support etc.

On the other hand, third party tools are readily available and save the effort required to build the tool.

The pros and cons of both need to be carefully evaluated before finalizing on the tools suite used for the migration.

Projects therefore need to use a combination of third party and in-house tools.

4.4 Testing Techniques

The execution of a migration project does not follow the same methodology as a normal development project. In a standard development project, test cases are developed based on the understanding of the application’s functionality. In contrast, a migration project does not require the understanding of the complete functionality of the application. Thus, there is a heavy dependency on the client in preparing the test cases.

The emphasis on testing in a migration project is to ensure that the functionality is retained during migration. Test cases are developed by running through the business transactions on the source environment and capturing the results. After the migration, the transactions are executed in the target environment, and the results are compared and matched. This is Replication testing.

Tools provide a clear advantage during the testing phase of a migration project. Once the migration of a particular syntax is automated and thoroughly tested on the sample code, one needs to worry less about using it across all occurrences of the same syntax in the entire source code. Hence, there are substantial savings in the testing effort with this approach.

4.5 Tool Suite

The table lists some of the important tools used during the Migration of the University's mission-critical applications.

Tool	Description
iTrack	Project management tool used to assist in tracking of various types of items such as defects, issues etc.
TOKENSEARCH	A tool used for impact analysis.
PacketCheck	A tool used for inventory analysis.
HTMLGEN	A tool to convert character-based screens to HTML screens.
DATAGEN	A tool to migrate data from a proprietary database to a relational database.
ASCOT	A code generator to handle platform related changes.
IOGEN	A tool to generate Open Database access routines (relational database).
CobolChecker	A tool to review code migration.
CodeWrap	Tool to convert a COBOL program to Object COBOL program for J2EE environment.
WFLCON	A tool to convert proprietary job control/workflow programs to BAT files.
Rational Test Suite	A tool for performance testing.

iTrack

Area: Project Management

iTrack is an item monitoring and tracking tool for tracking various items such as defects, queries, issues, etc. It enables a project workflow to be established. Data maintained in the iTrack system helps in the collection of metrics. Based on this, at various predefined review checkpoints in the project schedule, steps can be taken to reduce risks and to improve processes.

The system, thus, provides full item traceability that improves productivity and reduces the time spent on obtaining clarifications, etc. Weekly trends for item types such as queries, issues and defects can be monitored, allowing for initiating action if outstanding items go beyond the project defined control limits.

TOKENSEARCH

Area: Impact Analysis

TOKENSEARCH is used to find occurrences of various tokens across the inventory. The tool is especially useful in the analysis phase when the inventory is scanned for complexity. It helps in determining the extent of changes and aids in effort estimation of the migration.

The tool also gives the occurrence and the count of token specified, which helps in deciding the types of enhancements required on the tool.

Packetcheck

Area: Inventory Analysis

Packetcheck scans the complete inventory and reports the count of various components of the inventory, like types of programs (source, copybooks, etc.), language of programs (COBOL74, COBOL85, and ALGOL), missing inventory entities (missing sources and copybooks). It generates a comprehensive report detailing the counts and Lines of Code (LOC) of all types of components identified.

By aiding in analyzing the inventory and determining its complexity, the Packetcheck tool helps in effort estimation.

HTMLGEN

Area: Screen Migration

HTMLGEN is used to convert character-based screens to HTML screens. The tool reads the character screens and interprets the control characters used to build the screen. It also assigns attributes to a field such as Read-Only, Justify, etc.

Based on these characters, the tool builds HTML screens with the same layout and screen/field attributes.

DATAGEN

Area: Database and Data Migration

Migration involves extracting data from a proprietary database. For the Migration project, data was migrated from the Unisys mainframe (DMSII database) and loaded into the Oracle database. The legacy application had around 140 tables and 380 indexes. In addition, more indexes, views and sequences were created in accordance with the database migration strategy.

It was a formidable task and required a lot of effort if it were done manually. The manual effort would have increased the risk of incorrect schema and data migration that would have been difficult to identify.

This issue was overcome by using DATAGEN to generate the database schema. It created the complete schema by converting the proprietary schema into an Oracle database compatible schema, and by generating additional entities for the migration strategy.

The tool also dumped data from each table into a tab separated flat file so that it could easily be loaded into an Oracle database. For large tables, it dumped the table into multiple smaller files so that the ftp of the data files could be done faster and in parallel; this reduced the elapsed time for data migration.

Following adequate testing of the tool on a few representative tables, data migration of the remaining tables was completed within a short span without any errors.

ASCOT

Area: COBOL Code Migration

Migration of the COBOL code from Unisys COBOL to MFCOBOL constituted the bulk of the project activity. The application had around 1 million lines of COBOL code. Since the complete logic for the business processes was inside COBOL programs, it was essential that the COBOL code be migrated with minimal errors and be tested thoroughly. Manual migration of the code would have resulted in a longer timeframe and more migration errors.

The COBOL code migration was a repetitive task for which rules and strategies were documented. ASCOT was used to automate these standard code migrations. Use of ASCOT ensured that the errors related to incorrect migration were reduced to a minimal. Automation also meant that the timeline required to convert a program was reduced and the available time was used to test the complex business logic.

IOGEN

Area: Generating Database Access Routines

Owing to the change in the database environment, from hierarchical to relational, the proprietary database access logic had to be converted to work with relational databases. Yet, the challenge was to ensure that the application received the data in a manner similar to the old environment so that the core business logic was not affected.

In accordance with the migration strategy, database routines were extracted into a separate copybook to increase modularity and reuse. The analysis of the application revealed that there were around 1500 such different variances of database access across the complete set of tables.

IOGEN was used to generate database routines from the tables in the database. It used a template to generate routines for all standard syntax of database access.

CobolChecker

Area: Code Migration

This tool scanned the complete program, and based on the standards identified for code migration, flagged off lines in which incorrect code migration was done or set standards were not followed.

CodeWrap

Area: Code Migration

The migration methodology involved COBOL programs to be wrapped into object COBOL using the wrapper wizard provided by MicroFocus IDE (Integrated Development Environment). Although the wrapping of individual programs took about couple of minutes, looking at the inventory size and the number of iterations each program needed, wrapping would have been a significant effort in person months without the tool. The tool drastically reduced the cumulative effort required for wrapping as well as eliminating the element of human error.

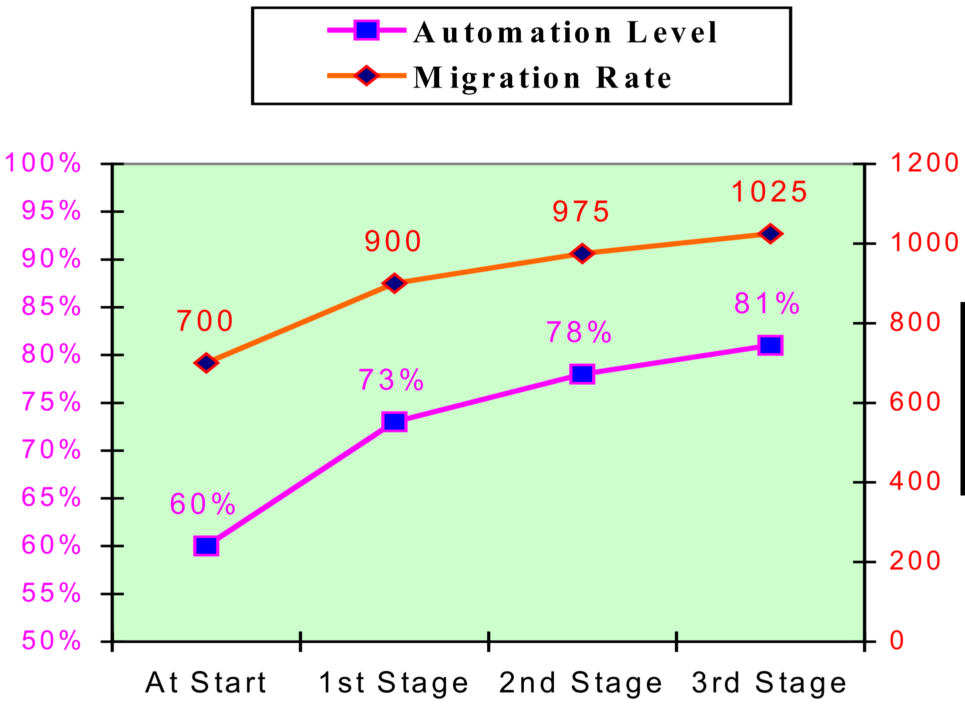
4.6 Metrics

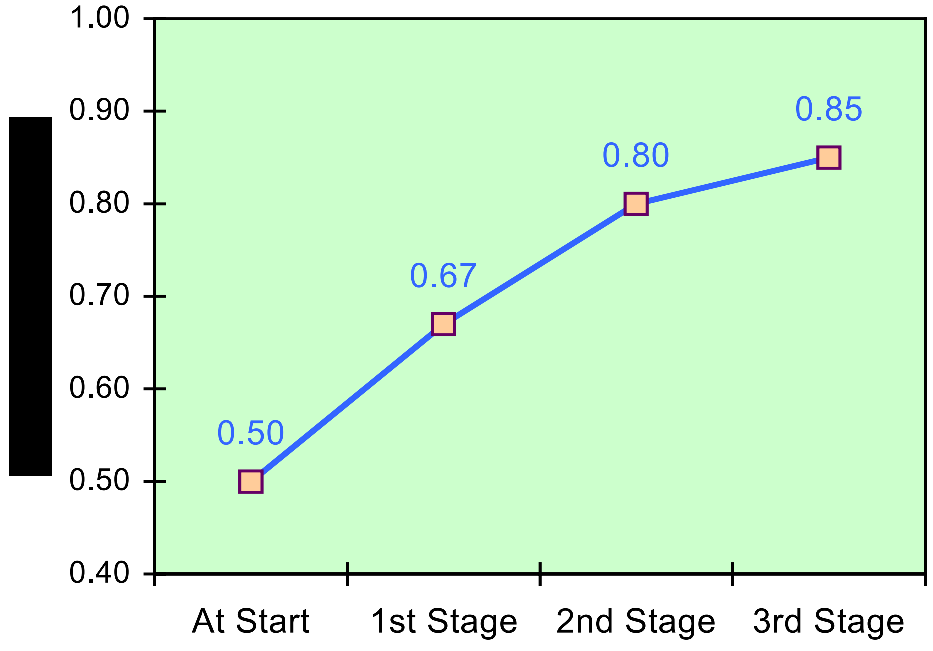
Metrics Used for Migration Projects

Metric	Formula
Automation level	$\frac{\text{LOC changed by a tool}}{\text{Total LOC changed}}$
Rate of migration	$\frac{\text{LOC migrated}}{\text{Day}}$
Screen migration Rate	$\frac{\text{Number of screens migrated}}{\text{Day}}$

Improving Metrics With the Use of Tools

Metrics	At Start	Stage 1	Stage 2	Stage 3
Automation Level	60%	73%	78%	81%
Rate of migration	700	900	975	1025
Screen migration rate	0.5	0.67	0.80	0.90





Note: The plotting in the graphs represents typical improvements.

5. Challenges

No doubt, tools and techniques ease and speed up the migration effort, but the migration process has its share of challenges. Some of these challenges are:

Data Migration: During data migration, tools have to take into account the native collating sequence of the source and target machines. Migration of signed computational fields from EBCDIC to ASCII is a difficult task because of the difference of encoding in these two collating sequences.

Schema Migration: In migration projects, most of the programs are not rewritten but are only migrated to be compatible with the target environment. The database layout in terms of the schema cannot be altered without corresponding modifications to the programs. Hence, it is recommended that the database schema on the target platform be **not** normalized. If done, considerable effort will be spent in distinguishing migration errors from that due to database normalization.

Screen Generation: While automating screen generation provides saving in terms of effort, dynamic screens provide a challenge to automation. Since the layout of these screens depends upon the data at runtime, they cannot be built using a tool.

Code Migration: A cost benefit analysis of the task to be automated needs to be done to decide whether it is worth automating it. Some of the syntax on the source platform does not have an equivalent on the target platform. In such cases, workarounds need to be arrived at to come up with the best possible solution to migrate the source code.

6. Inferences and Conclusions

The use of such Tools and Techniques reduces the amount of tedious and error-prone manual changes for a migration project.

As the automation levels of tools increase during the course of the project, productivity too increases. This increase in productivity should be accounted for in the effort estimation of subsequent modules. This would result in decreased timelines and help in planning future resources requirements.

The estimation model too should consider the increased productivity due to tools. This will help in arriving at a more realistic effort and timelines for projects.

Effective tool management plays a significant role in the execution of such migration projects. Identification of opportunities for increasing the automation level by enhancing tools should be planned in the project lifecycle.

Bibliography:

1. Bennett, K., "Legacy Systems: Coping with Success", IEEE Software, 12(1), pp. 19-23, January 1994.

2. L. A. Belady & M. M. Lehman., "A model of large program development", IBM Systems Journal, 15:225-252, 1976.

3. Ruhl, M., and M. Gunn (1991), "Software Reengineering: A Case Study and Lessons Learned," NIST Special Publication 500-193, Washington, DC, September 1991.

4. Jacob S. Dhinakar and Dr. Madhuchhanda Das, '"Automated Migration : A Phase of Re-Engineering", International Workshop on Program Comprehension (IWPC), 2003.

5.Bergey, John; O'Brien, Liam; Smith, Dennis. An Application of an Iterative Approach to DoD Software Migration Planning (CMU/SEI-2002-TN-027).

6. O'Brien, Liam; Smith, Dennis. MAP and OAR Methods: Techniques for Developing Core Assets for Software Product Lines from Existing Assets