Testing a Software Port

Mindfire Solutions
www.mindfiresolutions.com

August 29, 2001

Abstract:

While justifying the need of a separate testing strategy for ported application, this paper discusses testing methodology for porting projects. Portions of this document talk about the stage-wise testing approach and then elaborate each one to create a substantial and comprehensive process. Best practices gained from practical experiences; make this paper a good guide for testing software port

Software Migration: Growing old with Mindfire

This paper distillates some of our best practices in testing, from our experience in various porting projects till date. It is intended to serve as a guide to planning and conducting a successful test for porting projects.

Why is a testing port different?

A common assumption in porting projects is – Developing and testing a ported application is simpler than the regular projects as the specifications are clear, UI and other visual areas are available to be copied, architecture is already thought of in the original application, functionality requirements are pre-defined, test cases may be reused and the list of reasons will go on. But in practice, it is not so simple. It's true that a lot of technical work can be avoided, but at the same time, many decisional factors pose stronger problems. Here is a list of the common conflicts we have experienced and these are quite possible to occur if you are testing a porting project.

The confusion in testing ports arises out of unclear "desired behavior".

Known bugs: Many a times, a known bug in the original platform is easier to fix in the target platform as a part of the porting effort. It's difficult to decide whether or not to go for a fix. The results will definitely make the application inconsistent over both the platforms. At the same time, leaving a repairable bug in the system would not look correct.

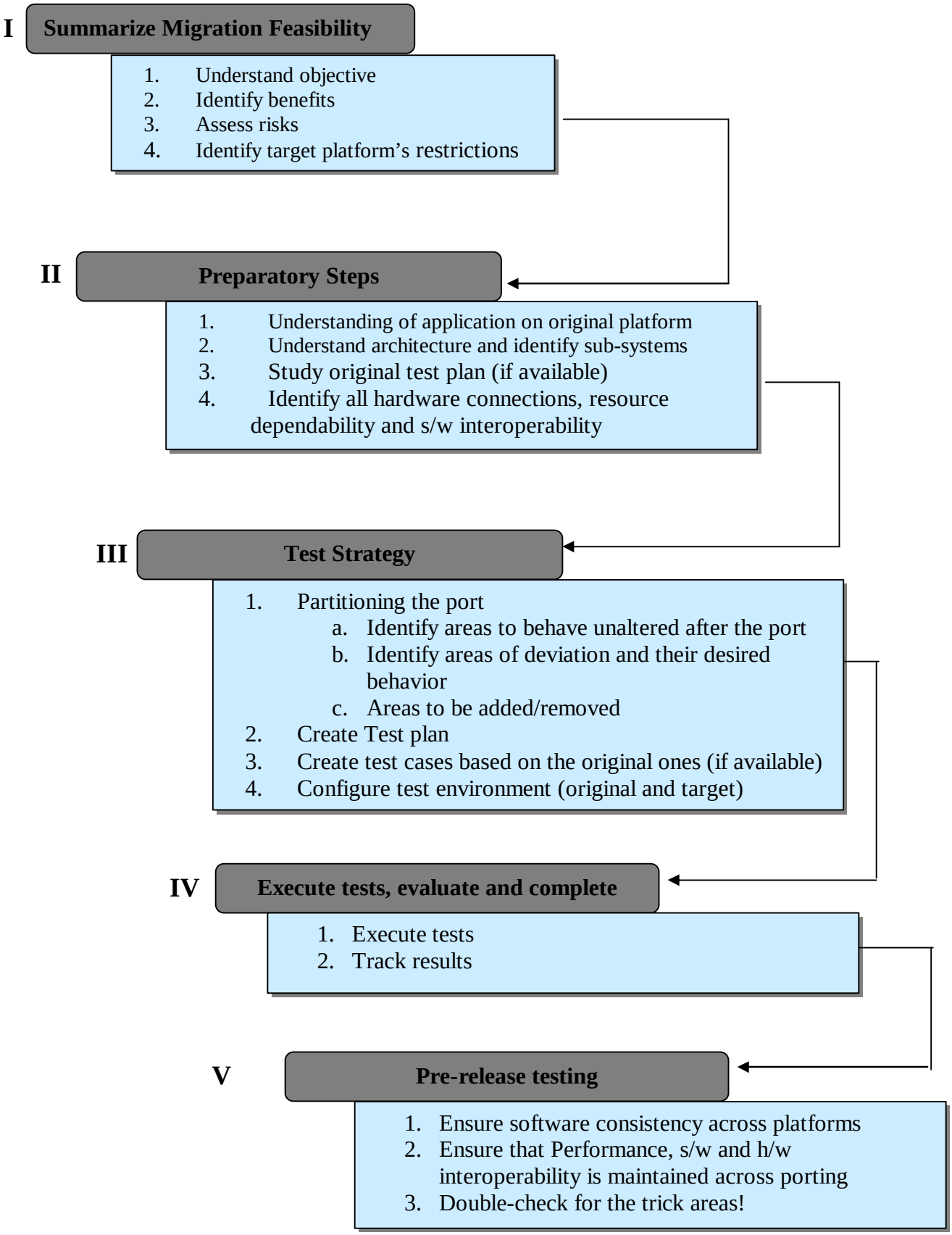
User Interface guidelines on different platforms: Testing will compare the actual behavior against the desired. But at times, it's difficult to identify what could be a desired behavior when it's driven by many other parameters. For example, in Windows buttons will appear as rectangular but the same controls will appear as round-edged rectangles on Mac. Hence it's difficult to predict whether the desired behavior should be having the same look for the controls or to go by the conventions of the target platform. In order to restore consistency, one has to use custom buttons for Mac, but this will create an application that does not maintain the Mac's look-and-feel.

Features handled by the OS: Many elements in the application are directly handled by the OS and hence behave differently across platforms. E.g. on Windows, every open window has application specific menu bar attached to it. But on Mac, a single menu is displayed for all open applications and gets dynamically updated according to the window in focus.

Phases for Testing a Port

Testing a port is just like testing any other software project, with a few extra things to do. The following figure outlines the phases we follow, at Mindfire Solutions.

Figure 1: **Phases for Testing a Port**



Summarize Migration Feasibility

Migration feasibility assessment is not a part of the testing activities. The client, management and technical group perform feasibility studies before the project is accepted. But involvement of testers in this stage has proven beneficial for us.

Our feasibility studies involve a lot of interaction with the client and taking their input in terms of business objectives, target user base, support resources for the new migration and compatibility with the existing infrastructure. Our porting analyst will anticipate the issues and assess risks before going ahead with the development work. In general, the most potential stumbling blocks – structural, functional and restrictions on the target platform – are identified ahead of time making it easy for our clients, developers and testers to resolve critical issues.

The QA team at Mindfire not only helps the analyst (primarily a senior programmer) identify and analyze risks, but also prepare to resolve the quality-oriented issues that may crop up during testing the application with the anticipated restrictions. Involvement of the QA from this initial stage of feasibility assessment prepares the test team for the following issues.

I. Understand the migration objective so that testing can be designed in the similar perspective. Whether the porting is intended to create a market presence or acquiring more user base etc. may prove vital while planning for testing.

II. Review the computing infrastructure and design the test lab accordingly. Mock setups are done in order to ensure that the migrated application works well with the existing infrastructure.

III. Study the user base in order to simulate different platform, hardware/software/ network combinations to meet compatibility requirements.

IV. Assess support requirement for the migrated application. This is important in order to test adequacy of system documentation, on-line help and trouble-shooting issues.

V. Identify the target platform's restrictions and variations. Few behaviors will be unavoidable because of the platform's specific design and structure. But care must be taken to preserve system integrity. Any deviations will be anticipated and the QA team specifies acceptable alternatives for the same.

VI. Think about interoperability issues. Key to a successful deployment of the migrated application is to ensure its ability to demonstrate that those who work on the old platform as well as the new platform can coexist in the network and exchange common sharable data. It is also important to assess the difficulty of migrating existing data to the new platform.

VII. Most important of all is to assess performance; stability and reliability of the application on the target platform should meet the existing benchmark.

Preparatory Steps

- ***Understanding Initial Application***

The purpose of understanding the initial application is:

- Identify basic/core functionality
- Familiarize with user interface and operation
- Identify external dependencies
- Detailed functionality
- Foundation to develop user session scenarios

To achieve these objectives, the testers must have a thorough knowledge of the original application functionality.

- Run the application on original platform till the level of understanding required is gained
- Review Help and other documentation
- Note down broad level functionalities
- Make informal notes about likely application port-incompatibility
- Run complete sessions as user

- ***Understand architecture and identify sub-systems***

Go through the architecture design document for the existing platform. Find from the development team if there are any architecture modifications required for porting.

The major sub-systems of the program need to be identified and the way they interact with each other should be understood. Also identify major areas in the application to be tested. E.g. for a typical desktop application, the major areas to be tested would look like the following.

- App-specific functional sub-systems
- User interface
  - Windows and frames
  - Buttons/labels and other controls
  - Complex UI elements
  - Graphics
- File/database store
- External software/hardware dependencies
- Help
- Performance
- Stability

- ***Study initial test plan (if available)***

If test cases for the original application are available, testers should read the same in order to get inputs for designing test cases for the target platform. Some of these test cases may be reused as it is. Some will need modification. There may be needs for designing fresh test cases for some features and areas as the old ones just won't fit the requirement. Testers also need to know the known bugs in the existing platform.

Testers don't write test cases during this stage. Actual test cases are prepared in the next stage. All they need to do is developing a good understanding of the available material in order to prepare the list of --

- Re-usable test cases
- Tests for modification
- Tests to be omitted
- Tests to be re-written

Test Strategy

Before starting with the initial preparations for testing, the scope of testing needs to be finalized. Deciding the scope depends on various factors like resources available, time allocated for testing and customer's primary requirements. The following points are considered while finalizing the scope.

- Should testing just ensure that the applications could be installed and launched properly? Or should it check further for specific functionality and performance?
- Should testing extend to thorough investigation of any problems? Or does it suffice just to make note of them and move on?
- While checking dependencies and interoperability features, do we test all the applications running on the original platform? Or should we focus only on the top priority applications?

Identifying the scope will help the test team focus in the correct direction and tracking progress.

- ***Partitioning the Port***

Partitioning the port will primarily mean breaking down the broad areas in the application to smaller features and organizing them under 4 major sections such as -- what will be dropped, what will be added, what will remain same and what will change

What will be dropped: This section would list all the features in the original application that should be removed after porting (for various reasons, most commonly absence of support on the target platform).

What will be added: List out the extra features to be implemented in the target platform. This may be due to improving the program by adding few small but new features. Also alternative features as a solution (work-around) to impossible issues are included in here.

What will remain same: Identify the features/components to behave unaltered after the port. All the features that should look and function like in the original application should be listed here.

What will change: Identify areas of deviation and project their alternative behavior.

Sometimes it's a conscious decision to remove un-necessary and inferior features in the existing application. Similarly few features may be added to improve the application. Also few issues seem to be difficult, impossible and defect triggers because of the target platform's restrictions, for which alternative behavior should be determined on the target platform. All features should be recorded in a checklist under the 4 sections mentioned above.

The following figure displays a sample partitioning exercise (in the form of a simple checklist) for a specific project.

Figure 2: **Partition Checklist for a sample port from Windows to Mac**

Partition Check-List							
Sub-systems		**Individual Areas/Features**	**Drop**	**Add**	**Same**	**Change**	**Comments**
User Interface	External to the Application	Application Menu				•	In Mac a common menu bar is shared across applications and changes dynamically as user switches from one to another.
		"About"				•	According to Mac conventions, it should appear under Apple menu.
		Multiple Document Interface (MDI)	•				Is not supported on Mac.
	Internal to the Application	Graphical buttons for Application toolbar		•			Client's request
		Encyclopedia View			•		
Functionality	Lab File Creation	Specifying species color and icons			•		
		Macros	•				Not required from this version onwards.
	Test Mode						

- ***Create Test Plan***

Test plan creation follows the same steps and techniques as in normal testing. However, specific input from the previous steps drives the plan.

- ***Create Test Cases***

Functional test cases should be designed based on the prior activities of understanding the original application and partitioning areas. Testers are clear by now as to what functionalities should they check for and what to omit. Addition of new functionality means reworking the code, which makes it a defect prone area. Test cases should be designed in order to focus more on code areas being altered. The following types of test cases may be considered for development.

- Functional test cases
- Regression test cases
- User-interface test cases – Testing the UI design and consistency, considering User Interface guidelines of the platform ported to
- "Scenario-based" test cases (with pre-defined parameters and values)
- "User session" test cases – These will be simulation of user activities across a complete session. These will generally include all user activities from the time application is started till it's closed.

Some applications may have the original test cases available. Those can serve as an input to creating the tests for target platform. The reusable test cases are picked up. Any known bug list should be taken into records.

- ***Create a "Trick Area" list***

A "Trick Area" list identifies areas that may look like weak links and bug-prone. Testers may anticipate such areas based on prior experience of similar porting projects. A good understanding of both platforms also helps in isolating areas that may cause cross-platform problems. Test execution for such areas should be rigorous.

Another way to find trick areas is to interact with the developers or sit in their meetings and pick up issues that they find difficult, confusing or hazardous. We have found the following method suitable for catching these pitfalls:

- Sit in on developer meetings/discussions
- Take notes on issues/fights/stumbling blocks
- If the team discusses anything for more than 10 minutes, investigate for specific issues
- Anything across 2 meetings, definite red flag
- The fundamental premise is: if they're finding it difficult, they'll probably stumble on it

- ***Configure Test Environment***

A test environment would mean installing the original platform as well as the target platform. All hardware and software dependencies should also be configured. In order to simulate wide variety of possibilities, a good combination of the above is required. Ideally the testers should have the application running on the original and target platform nearby, preferably on the same desk. This helps in quick referencing and increases test efficiency.

The following issues in the test environments are a must-check.

- Configure servers and client machines.
- Configure the original and target operating systems.
- Identify all hardware dependencies (external peripherals) like printer, scanner, storage media, drives etc
- List all primary software dependencies (External software that the application may use at some point of time e.g. a mail client or a word processor)
- List the secondary software dependencies (External software that may be found on the user's machine but not shared with the original application. It's recommended to pick up some frequently used software)
- Keep clean machines
- Databases or other data sources should be configured.

Execute tests, evaluate and complete

While testing is in progress, one needs to keep a tight rein on activities in the lab and the execution of the testing plan. There are several strategies that will serve to keep the test plan on track.

- ***Execute Tests***
  - Check that test cases are executed according to the test plan and schedule.
  - Perform regression, stability and performance testing.
  - Check that consistency of look and behavior of the application is preserved while porting.
  - Stress on test cases built for Trick Areas, as they posses more possibility of finding defects.

- ***Track Results***
  - Coordinate the efforts of the testers with the stated objectives, ensuring that they test all necessary components to satisfy the testing criteria.
  - Decide early about a systematic bug management procedure and implement the same. Checks should be conducted to ensure that developers and testers adhere to it for entering open bugs as well as fix and close repaired ones. You may use a bug-tracking tool or a simple spread sheet for this purpose. The motive should be to decide upon a system and implement it.

**** At mindfire, we use our in-house bug management system, which takes care of bug tracking, bug distribution and reporting system.*

- Coordinate and standardize documentation of the test results. Bugs should be entered in a definite format with adequate information about the bug behavior and reproducing methods.
- Decide early about tester-developer communication protocol.
- Compile reports while testing being conducted. Reports help building awareness among testers and developers about the progress of the application.

Pre-release Testing

This is a high-stress and exhausting phase of testing. Issues that are critical to porting projects are crosschecked in this phase. This includes a last minute consistency checking, performance on the target platform and software/hardware interoperability. The "Trick-Area" list prepared before should be reviewed again.

Conclusion

Testing a ported application needs a paradigm shift from conventional testing strategies in order to adequately address the radically new needs represented during a porting endeavor. The new testing process should be geared towards handling critical issues with more up-front design, performance testing and ongoing integrity testing in order to reveal migration bottlenecks. This paper aims to help test teams intuit the possible porting criticalities and plan to test more efficiently.

Mindfire Solutions is an offshore software services company in India. Mindfire possesses expertise in multiple platforms, and has built a strong track record of delivery. Mindfire passionately believes in the power of porting and its many advantages for software product companies.

We have developed specific QA/testing techniques to test ports efficiently and make porting a risk-free exercise. We offer specialized QA/testing services for your porting projects. For porting projects that our development team executes, the same high level of Quality Assurance is mandatory.

If you want to explore the potential of porting and our QA/testing services, please drop us an email at info@mindfiresolutions.com. We will be glad to help you.

To know more about Mindfire Solutions, please visit us on www.mindfiresolutions.com

---