****SIMULATION IN HOST-TARGET TESTING****

INTRODUCTION:

This article describes the importance of simulation in "host-target testing". And accordingly leads to a theoretical strategy that can be followed for the testing of host-target developments.

WHAT IS IT?

Before getting into "host-target testing" first we should concentrate on the words "host" and "target". The "host" is the environment in which the application is developed and "target" is the environment where the application finally executes. Take the example of mobile phone games. Are they developed in the mobile? No, they are developed in the work stations. The environment in which they are developed and in which they run are completely different. This kind of development is traditionally being followed for the embedded system softwares. There the application is developed in a powerful multi-user environment and ultimately runs in the embedded microprocessor environment.

This kind of development is called "host-target development" and the associated testing is called "host-target testing" or "cross testing". Almost every application development follows cross testing to some extent, depending upon the difference between the target and the host environment. This difference ranges from minor differences in configuration to major differences like in embedded system.

Cross testing has become important these days because it is very rare that the application is developed in the same environment in which it is meant to run.

ISSUES INVOLVED IN HOST-TARGET TESTING

Technically all testing should be carried out in the environment where the application has to finally execute or in the target environment. But there are issues that influence the testing to be carried out in the target system.

<A>Feasibility issues:

- *The target for which the application is being developed may not yet be available.*
- *The target environment may be less convenient to work with than the host. Because a target environment may not have a keyboard, debugger or even a processor.*
- *A bottleneck may arise for many developers competing for the target environment to test their work. This may hamper the development process.*

<B>Financial issues:

- *Usually the target and its associated tools like in-circuit emulator costs many times more than that of the host.*

This leads us to the conclusion that most of the testing should be carried out in the host side. But it has also got some potential problems.

- *The application may have interface routines that require direct access to the hardware or software of the target.*
- *The host and target may have potential differences in areas like word length, significance of bits, data structure etc.*

Therefore in the design phase, interface routines should be made isolated, leaving the rest of the software target independent (Fig:1). Also by simulating target in the host environment, target dependency in testing can further be reduced.

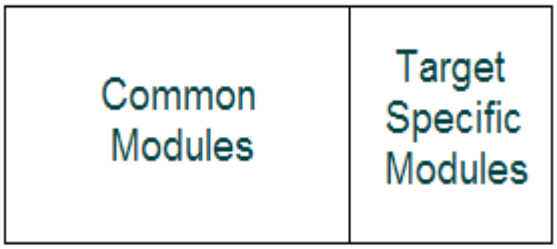

Fig: 1(good software architecture)

SIMULATION BASICS:

Even with careful design as shown in Fig: 1 the target dependency is only reduced to a limit. It depends on the amount of coupling between the application and the target. So, plenty of testing is still left, that has to be done in the target environment. Therefore we have no alternative but to create the "Target Environment" in the host itself. This is the idea behind "simulation".

Simulation means creating target environment (that includes the target OS, target h/w etc) in the host, so that the application cannot distinguish between the actual target and the simulated one. It is important to take care that simulated target environment is not tightly coupled with the host environment.

The first benefit of simulation is that it saves target resources. Also by simulation, testing done is "non-destructive". Because here we are not actually testing with the real target h/w. So, real damages are rare.

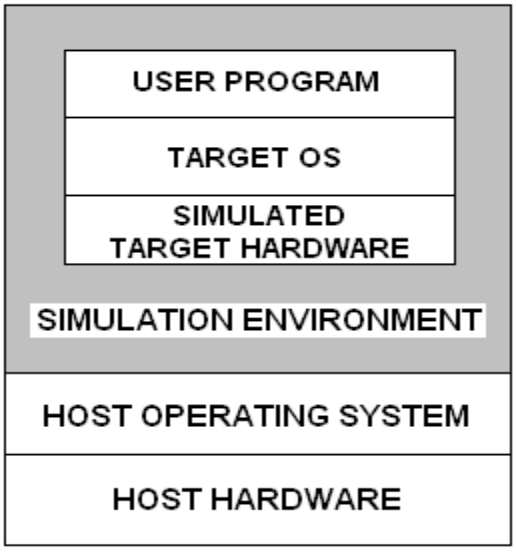

Fig:2(simulation model)

DECIDING A STRATEGY

<A> Unit testing:

All modules that are not target dependent can be tested in the host. And also target dependent modules can be tested in the host by simulation. It is always easier and faster to carry out unit testing in the host.

After the modules are unit tested in the host, a confirmation test can be carried out in the target to ensure that results do not vary due to the differences in the environments.

The confirmation test at this level finds out any unknown or unanticipated differences between the host and target, which is a useful piece of information for the future.

<B> Integration testing:

In integration testing, unit tested modules are tested for inter-module communication .Without simulation ,integration testing can be carried out in the host environment,by replacing target dependent modules with test stubs. But this becomes almost impossible in case of embedded systems as the coupling between the target environment and the modules is very tight. So only low level integration testing can be performed without simulation.

But with simulation,most of the integration testing can be carried out in the host. The application can not distinguish between the actual target and simulated one. So the results produced almost explain the behavior of the application in the target.

As in case of unit testing a few confirmation tests are carried out in the target .The confirmation tests at this level indicate differences between target and host in areas like memory allocation-deallocation etc.

<C> System and acceptance testing:

For system testing, tests can be developed and executed in the host and then ported to the target for confirmation testing. The tests may require some modification before porting to the target because of the difference between the host and the target. Usually less number of people are involved in the system testing. So it is possible to perform the system testing directly in the target.

Finally acceptance testing should be carried out in the target environment. Because acceptance of a system should be tested in the actual environment, not in a simulated one.

CONCLUSION:

No strategy in software is permanent. They will keep changing from application to application. Same with "host-target testing". Amount of simulation "feasible" and amount of simulation "desired" are two variables to be considered before deciding in favor of simulation.

Prepared by: Smruti Ranjan Kar.
Mindfire Solutions.