

Traceability and Reproducibility in Integrated System Test Environments

Nancy S. Eickelmann

Motorola Labs
1303 East Algonquin Road
Schaumburg, IL 60196 USA
+1 847 538 0745

Nancy.Eickelmann@motorola.com

Allan L. Willey

Motorola Labs
1303 East Algonquin Road
Schaumburg, IL 60196 USA
+1 847 576 6343

Allan.Willey@motorola.com

ABSTRACT

The Motorola Automated Test Environment (MATE) provides a means of automating the test process and integrating tools to support required testing capabilities across the lifecycle. The MATE architecture is provided as a foundation to discuss the issues of test tool integration with COTS products that automate the full system test lifecycle. Specific tool integration issues include data, control, process, platforms and user-interface integration issues. The software architecture of MATE can facilitate or impede analysis of desired system properties and performance requirements. Architectural analysis is conducted to provide insight into the properties of the Motorola Automated Test Environment.

Keywords

Automated Test Environment (ATE), COTS Tool Integration, System Test.

1 INTRODUCTION

The Motorola Automated Test Environment initiative will enable Motorola to optimize the test process through standardization of tools; automation of manual processes; and integration of data, processes, and interfaces across a common test environment. Specific benefits of the MATE initiative relate to test management, automated test script generation and execution, UI platform independence, and improved quantification of release criteria. All of these factors provide the necessary infrastructure to support test traceability and reproducibility.

- **Test management** - Provides a central controller for the test environment, test artifacts and test data.
- **Automated test script generation and maintenance** - Provides the automatic generation of test scripts from a formal requirement representation. Maintenance of the test scripts as the requirements change is facilitated.

- **Automated test execution** - Alleviates staffing shortages at critical process bottlenecks.
- **Web front end** - Provides for access to the test tools and the data from anywhere on any platform. The web was seen as the optimal solution.
- **Reliability prediction** - Provides objective criteria to determine when a product can ship based on predicting remaining defects in a product after a test cycle.

Providing for continuous improvements in core processes often requires the insertion of tools and technologies to automate and standardize industrial practice. Understanding the strengths of tools and applying them appropriately in the context of the organizational maturity is essential.

Despite the critical importance of automated test environments in the development of high quality products, full lifecycle integration of tools is rarely achieved in practice. To understand what prevents full lifecycle integration of tools with respect to data, process, platforms, user-interfaces and control, Motorola is conducting concurrent, focused R&D initiatives, Motorola Automated Test Environment (MATE) and Core Process Redesign (CPR).

The Motorola Automated Test Environment (MATE) initiative provides technology to design, develop, and test high quality software and systems. The MATE initiative addresses a subset of the product development issues focusing on testing, which remains a very costly, and time-consuming phase of the product lifecycle.

The Core Process Redesign (CPR) initiative is identifying the necessary integration factors to insert an automated test environment that seamlessly inter-operates in the context of the overall product lifecycle. The CPR identifies decision criteria (decision gates), process input/output pairs, entry/exit criteria and common resource requirements and constraints.

The overriding objectives of MATE and CPR are to radically improve time-to-market and predictability in schedules, costs, and quality of products developed by:

- development of a common core process that enables consistent practices
- build a common platform that supports interoperability and reuse
- increase resource allocation flexibility to facilitate cost effectiveness

The MATE and CPR initiatives are complementary efforts that focus on standardization, automation, and integration of tools, data, and processes. A primary goal of the MATE initiative is to develop a common and automated test environment. With a common environment the entire corporation would be able to take advantage of optimization improvements made to the environment based on technology advancements external to Motorola and technology acquisition internal to Motorola. Therefore the MATE architecture must accommodate legacy tools, process changes and future needs met by technology insertion. As such the MATE architectural requirements are a negotiated construct among the identified stakeholders [8].

This paper examines software architectural constraints in relation to software and system test automation environments. The MATE architecture requires a mapping of multiple views of the environment including structure, functionality, process and data perspectives. An overview of the environment is represented by mapping the test functionality to hardware and software structures of the environment. An architectural representation provides a foundation for evaluating the impact of architectural and COTS choices on system test engineers and test managers. MATE integrates COTS tools such as DOORS, ClearQuest and ClearCase; internally developed test management support, TMS; and automation of the system test process.

2 BACKGROUND

Motorola Labs is the research department of Motorola Inc. There are groups around the world conducting research on the leading edge technologies and on advancements to current product offerings. A majority of the research conducted by the Labs is initiated by requests from

the product groups within the Communications Enterprise (CE) of Motorola. The CE is comprised of several large business units and accounts for more than half of Motorola's sales. The major businesses in the CE include the cellular handset and cellular infrastructure businesses, the paging device and paging infrastructure businesses, and public and private radio products with their associated infrastructures. All of these products, whether handheld or infrastructure, are computer-based. For the last two decades these products have evolved from a hardware component base to a computer platform with functional layers comprised of software specifically written to address the product requirements. The R&D organizations in Motorola have evolved accordingly so that at this time far more than half of the engineers developing new products worldwide in Motorola are software engineers.

The size of software components in our products vary according to their function. A simple one-way handheld pager will have no more than a few thousand lines of code. A cellular infrastructure product has over twenty million lines of code in its various components. A small core of the infrastructure products are concerned with "call processing," on the order of a few hundred thousand lines of code, and this core will be the most highly reliable part of the system. A similar core component of cellular handset and two-way radios will be key to their functionality, and will consist of tens of thousands of lines of highly optimized code. Numerous additional functions are a part of each product, for example billing tracking and customer authentication components of a cellular system. Some of these, such as authentication, are real-time and interact with the core call processing functions. Others, such as system maintenance functions, are less time-critical but often tend to require large functional blocks.

The rate of growth of the software component of all of these products reflects general telecommunication industry experience. Further, we expect this evolution to continue as future product directions seem to show an unabated demand for new products. For example, while the popularity of simple one-way paging devices is declining, two-way short-message-service paging

offerings are a growing market. These two-way devices are an order of magnitude more complex, and contain that much more embedded code.

Numerous other examples can be cited of increased product complexity and size, but underlying these significant changes is the fact that testing has remained a very costly, time-consuming phase of the product development life cycle [2,9,11,12]. As the market pressures increase to deliver more, better, and faster products, testing is being viewed increasingly as a bottleneck to success.

To bypass the testing bottleneck, Motorola has launched focused R&D initiatives to provide technology to design, develop, and test high quality software and systems. The MATE initiative addresses a subset of the issues. A concurrent corporate initiative of Core Process Redesign (CPR) addresses another view of the issues. The CPR initiative is instrumental in identifying the necessary integration factors to insert an automated test environment that seamlessly interoperates with the rest of the product lifecycle. The overriding objective of CPR is to radically improve time-to-market and predictability in schedules, costs, and quality of product development by:

- Development of a common process across the divisions.
- Identify points of synchronicity across divisions.
- Improve efficiency of processes.
- Identify best practice tools to create a common and global R&D environment
- Build a foundation to propel common platform design and reuse
- Increase resource allocation flexibility across the corporation.

The establishment of the Motorola automated test environment contributes to all of the above goals as well. The testing activities have tremendous effect on product time-to-market, cost and quality [3,4]. The above goals can also be applied to internal product deliveries from tool development groups. Establishment of common tools and processes eliminates the redundancy of the internal tool groups and allows for reallocation of personnel to focus on product delivery and not internal support. It also allows for innovative ideas

to be implemented in a common platform and taken advantage of by a multitude of users instead of restricting technological benefits to local groups. The MATE initiative will bring Motorola closer to achieving its goals. MATE focuses on standardization, automation and integration of tools, data and processes. To facilitate our discussion and provide common definitions for terms we introduce the automated test environment reference architecture used in our analysis.

3 AUTOMATED TEST ENVIRONMENT REFERENCE ARCHITECTURE

A reference architecture, the STEP model [5] provides a basis for the representation and formalization of the MATE architecture. The reference architecture segments the required functionality for the environment and captures the relationships among the functionalities in a diagrammatic format.

The domain is partitioned into six canonical functions: test execution, test development, test failure analysis, test measurement, test management, and test planning. Each of these functions is defined to provide consistency in the use of terms.

- **Test Execution** includes the execution of the instrumented source code and recording of execution traces. Test artifacts recorded include test output results, test execution traces, and test status.
- **Test Development** includes the specification and implementation of a test configuration. This results in a test suite, the input related test artifacts, and documentation. Specific artifacts developed include test oracles, test cases and scripts, and test adequacy criteria.
- **Test Failure Analysis** includes behavior verification and documentation and analysis of test execution pass/fail statistics. Specific artifacts include pass/fail state and test failure reports.
- **Test Measurement** includes test coverage measurement and analysis. Source code is typically instrumented to collect execution traces. Resulting test artifacts include test coverage measures and test failure measures.
- **Test Management** includes support for test artifact persistence, artifact relations persistence, and test execution state preservation. Test process automation requires a repository for test artifacts. A passive repository such as a file serves the basic need of storage. However, an active repository is needed to support relations among test artifacts and provide for their persistence.

- **Test Planning** includes the development of a master test plan, the features of the system to be tested, and detailed test plans. Included in this function are risk assessment issues, organizational training needs, required and available resources, comprehensive test strategy, resource and staffing requirements, roles and responsibility allocations, and overall schedule.”

The STEP model, shown in Figure 1, stratifies test functionalities from the apex of the pyramid to its base in a corresponding progression of the test process lifecycle as described in [9]. The test process evolution is aligned with the arrow to the right of the pyramid and segments test functionality according to the test lifecycle focus. Each segment represented in the pyramid includes the functionalities of previous periods as you descend from the apex to the base.

The top section of the pyramid represents the function of test execution. Test execution is clearly required by any test process. The test process focus of debugging includes only test execution.

The second segment of the pyramid, from the top, is divided into two scalene triangles.

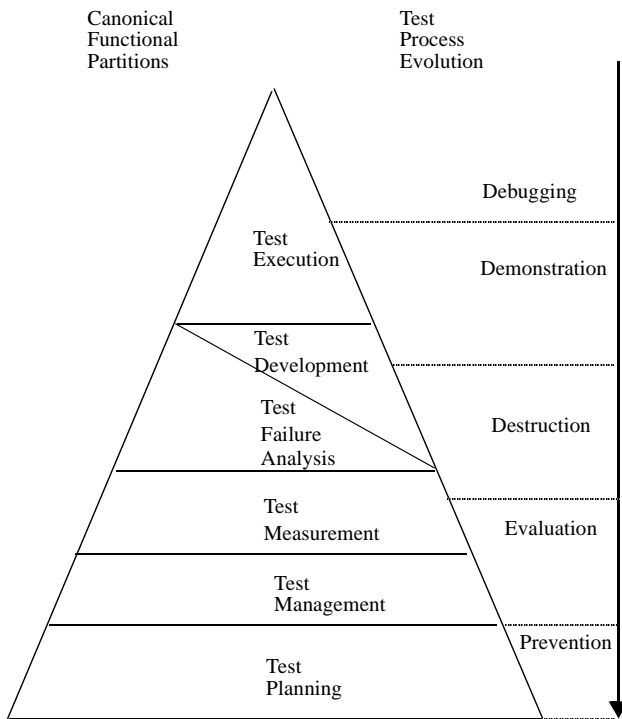


Figure 1. STEP model reference architecture [5].

The smaller scalene triangle represents test development. The larger scalene triangle represents test failure analysis. The relative positions

and sizes have semantic significance

Test development played a more significant role to the overall test process when focused on demonstration and destruction due to the manual intensive nature of test development. Test development methods have not significantly changed, although they have improved in reliability and reproducibility with automation. Thus, their role in test process diminishes in significance as you automate the test development process.

Test failure analysis is less important when performed manually, as interactive checking by humans adds little benefit for test behavior verification. The methods that can be applied to test failure analysis have increased in their level of sophistication, making test failure analysis more significant to the overall test process. One of the most significant advances is specification-based test oracles [5]. This is a key difference in test process focus.

Test measurement is represented by the third segment in the pyramid. Test measurement is required to support an evaluation focus for test, which represents a full lifecycle approach. A significant change in the test process focus is that testing is applied in parallel to development, not merely at the end of development. Test measurement also enables evaluating and improving the test process.

Approaching the base of the pyramid, the fourth segment represents test management, which is essential to the test process due to the sheer volume of information that is created and must be stored, retrieved, and reused. Test management is critical for test process reproducibility.

The base, or foundation, of the pyramid is test planning. Test planning is the essential component of prevention focused test efforts. Test planning introduces the test process before requirements, so that rather than being an afterthought, testing is pre planned and occurs concurrently with development. The STEP model provides the core functionality and process focus to describe the MATE architecture. The architectural description requires integration of multiple views of the test environment including structure, functionality, process

and data.

In the next section, the MATE architecture is described and a detailed discussion of test automation and tool integration issues is undertaken. Software architectural analysis is used to determine if a specific structural decomposition and functional allocation to system structures supports or impedes achieving desired properties for MATE. Changes to an ATE such as enhancements to system functionality, improvements to performance (space and time), and reuse of components, data representation and changes to processing algorithms are all sensitive to system architectural constraints [8].

4 MATE ARCHITECTURAL VIEWS

There are several architectural views in the literature including: Map View (mapping of functions and components), Static View (structure diagram), Resource View (mapping of software onto hardware), Dynamic View (operational diagrams).

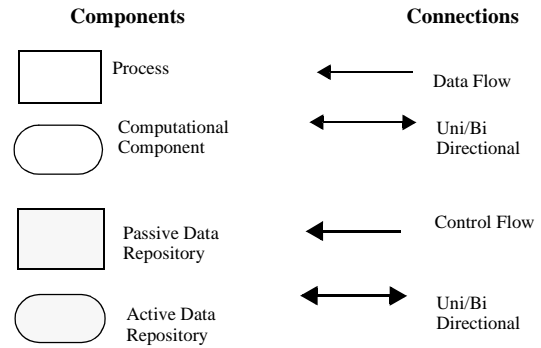


Figure 2. SAAM graphical architectural notation [10].

We provide a diagrammatic representation of the static, map, and resource architectural views for MATE. The static and map views are combined by using the SAAM notation. The resource view uses a generic representation.

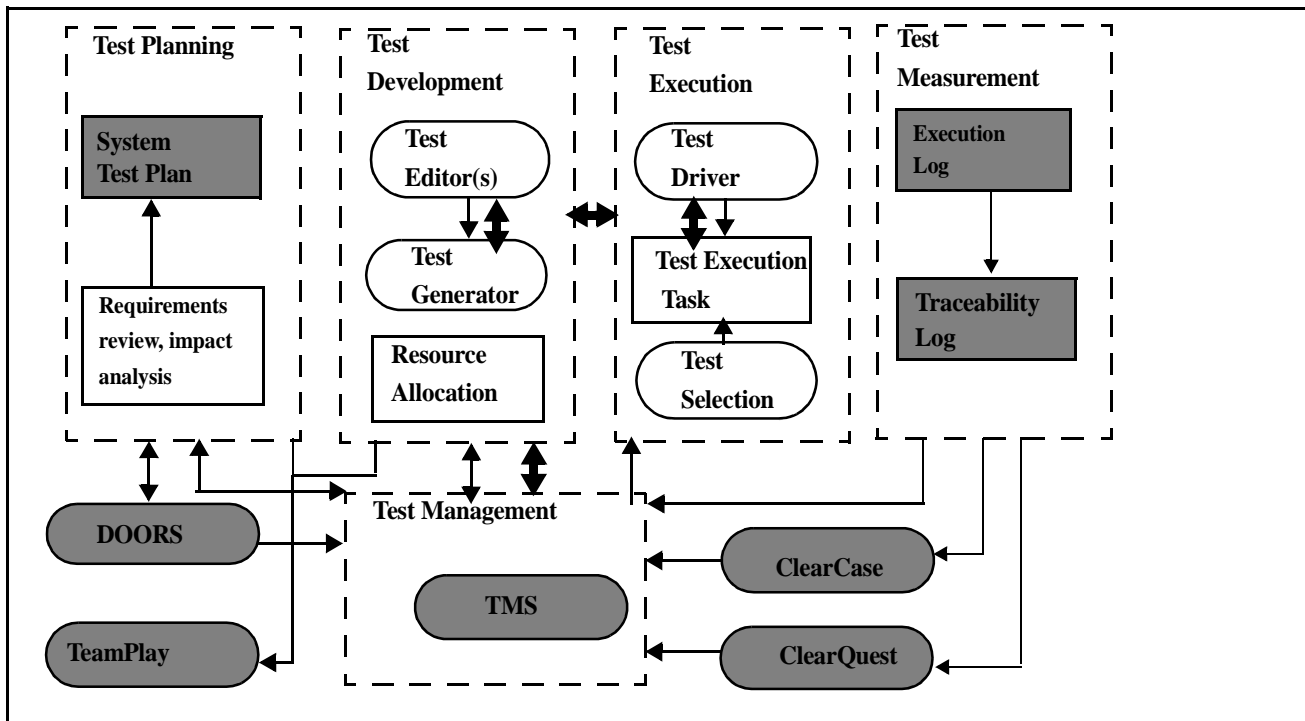


Figure 3. Combined Static and Map views.

4.1 MATE Architectural Static View and Map View

The Software Architectural Analysis Method (SAAM) provides a concise notation that includes a static view and a map view [10]. The static view represents the structure as a decomposition of the system components and their interconnections. The map view groups the components according to their high level functionality. The canonical functions for MATE were defined in the reference architecture.

The SAAM graphical notation is shown in Figure 2. In this notation there are four types of components: a process (unit with an independent thread of control); a computational component (a procedure or module); a passive repository (a file); and an active repository (database). There are two types of connectors: control flow and data flow, either of which may be unidirectional or bidirectional.

The allocation of domain functionality to the software completes the graphical representation of the ATE, see Figure 3. The allocation provides the mapping of a system's intended functionality to the concrete interpretation in the implementation. The diagram segments the canonical functions for MATE using a broken line. The COTS active repositories are not included in the domain functionality as they interact through the TMS management system. The five functionalities inclusive in MATE, test execution, test development, test measurement and test management are discussed below.

Test Execution includes the execution of the system in the target environment testbed, and recording of execution traces. Test artifacts recorded include test output results, test execution traces, and test status. The test execution environment must provide a bridge to interface with legacy solutions that must be migrated in an evolutionary versus revolutionary approach. Testing in a distributed context with heterogeneous platforms was considered desirable for flexibility and reusability. Another factor of significance was language independence and open interfaces for interoperability.

Test Development includes the specification and implementation of a test configuration. This results in a test suite, the input related test artifacts, and documentation. Specific artifacts developed include test oracles, test cases and scripts, and test adequacy criteria. Test development solutions should support multi-platform and multi-site access to data while providing adequate response time and usability in the user interface. Automated test case generation solutions focused on SDL and MSC model specifications that support auto test case generation. Significant process and data integration issues arose that were addressed by interfacing ClearCase test data with TMS and ClearQuest CM and defect data with TMS. Resource allocations for system test are documented in TeamPlay and provided on-line to the test developers.

Test Measurement includes documentation and analysis of test execution pass/fail statistics. Specific artifacts include pass/fail state, test failure reports and test traceability reports. MATE's support for test measurement must be addressed across all the functionality and interface with legacy data integration issues for test and development groups. A distributed collection and repository tool provides for heterogeneity with transparency of access issues through web-enabled tools.

Test Management includes support for test artifact persistence, artifact relations persistence, and test execution state preservation. Test process automation requires a repository for test artifacts. A passive repository such as a file serves the basic need of storage. However, an active repository is needed to support relations among test artifacts and provide for their persistence. Test management should provide a state-of-the-art solution with web-based access and interface. TMS was available on UNIX platforms and was being migrated to an NT platform as well. Since it was Java based, TMS would easily adopt to new platforms as Motorola test groups use them. Prototypes of TMS had been operational on SUN Solaris, HPUNIX, Windows NT, Windows 98, MacOS and Linux. TMS has been designed to take advantage of the multi-site features of Oracle and a software configuration management tool by using both as underlying technology to the tool. The TMS API will also allow for a quick

replacement of either technology as improved technologies become available. A significant finding through the MATE effort was that terminology was not used uniformly throughout the organization. A living glossary was implemented continues to track and resolve discrepancies in word usage and understanding.

Test Planning includes the development of a master test plan, the features of the system to be tested, and detailed test plans. Included in this function are risk assessment issues, organizational training needs, required and available resources, comprehensive test strategy, resource and staffing requirements, roles and responsibility allocations, and overall schedule. Test planning requires inputs from the DOORS repository and TMS test management function. Test planning also requires project management tool support which is provided by TeamPlay. Test planning is not well supported by most commercially available tools as it requires integration with data repositories, process issues, and tool interfaces.

4.2 Resource View

A common representation for architecture is the resource view. This view of the MATE architecture is diagrammed in generic terms such as provided in Figure 4.

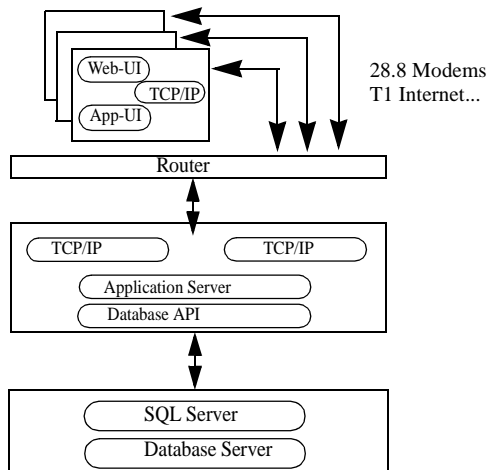


Figure 4. MATE Architecture Resource View.

The resource view focuses on essential communications links such as routers, modems, internet and intranet connections and the architectural topology, client/server. Analysis of this view lends

itself to performance analysis, queuing models and throughput simulations, see Table 1. This view is also used to evaluate system availability constructs using a composite quantification of the system hardware failure/time interval and the corresponding software failure/time interval. Mean time to repair is estimated for hardware and corresponding software providing an incidence frequency with an estimate in hours to restore the system. The declaration of system hardware and software computational units also provides for an affordability analysis. The initial investment costs, updates, and maintenance costs can be obtained from the vendors of choice. Given the payback period, and the minimum time required for return on investment, a Net Present Value (NPV) comparison can provide an objective valuation of hardware and software configurations.

Table 1: Performance Analysis

| Scenarios | Comments |
|--|--|
| Network latency from client to server and from server to client in ms. | Latency time provided by network specification |
| Server latency consists of dequeuing time (Cdq=ms) and task computation time (Cfnc=ms) | Latency time provided in server specification |
| Distribution time of periodic updates minimum | 8@10sec;8@20sec;8@40sec;8@80sec |
| Distribution time of periodic updates maximum | 48@10sec;24@20sec;12@40sec;8@80sec |

5 CORE PROCESS DESCRIPTION

The CPR identifies decision criteria (decision gates), process input/output pairs, entry/exit criteria and common resource requirements and constraints. The CPR initiative provides a high level common process with fixed decision gates. However, it also provides for insertion of customized process segments for insertion into organizations with legacy tools and technologies that impose unique process constraints. The software development process and software and system test groups in Motorola follow various lifecycle models that comply with internal

standards. An external standard that provides a high level process description for test is the IEEE Std. 1012-1998, Standard for Software Verification and Validation. This current standard is an update of the IEEE Std. 1012-1986. The updated standard is comprehensive in its scope and details full lifecycle activities and cross-references according to compatibility with other process standards. The substantial benefit of a well-defined process with objective decision criteria is realized when it is instantiated in an automated test environment.

6 TECHNOLOGY TRANSITION PLAN

The MATE transition effort is analyzed in the context of the 3 vector space of the SEI Technology Transition Conceptual Framework [6], see Figure 5. The 3 vector space represents the relationships among an increasing magnitude of technological change; the required effort to adapt or learn new skills, procedures, structures, strategies, or culture; and time required to adapt

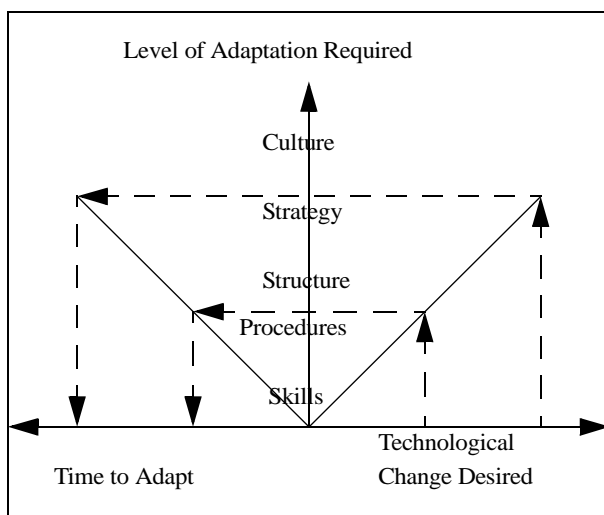


Figure 5. Dimensions of technology transition adapted from [CMU/SEI-93-TR-31]

and institutionalize the changes. This structure supports evaluating the “size” of a change based on the level of adaptation required. We will reference this figure in our subsequent discussion of MATE. The MATE effort as described using the SEI’s technology life cycle conceptual framework depicts an effort towards a common test environment that is comprised of technologies in

various phases of maturity. The majority of MATE technologies will require new skills and procedures as a foundation for their introduction. These changes are seen as requiring typically a time horizon measured in months.

The time required to achieve a technology transition is partially dependent on an additional classification of the technology life cycle. The technology life cycle includes 3 distinct phases, R&D, new product development, and adoption and implementation. Tools and technologies of MATE span the full technology life cycle and therefore may be described as short term to long term change efforts. The insertion of MATE represents new technologies for the system test organizations, however the technologies include both recent R&D developments of Motorola Labs and mature commercial tools to be integrated into the common environment.

We outline the steps of each of the 3 technology life cycle phases as described in Figure 6. The focus of phase 1, R&D of the technology life cycle, is primarily the technology itself. Inclusive in the R&D phase of the life cycle;

- concept formulation
- development and extension
- enhancement and exploration (internal)
- enhancement and exploration (external)
- early popularization.

The aspects relevant to MATE in phase 2, the new product development phase includes;

- generating new product ideas
- screening those ideas
- testing product concepts

Adoption and implementation, phase 3 of the technology life cycle includes multiple steps relevant to the MATE transition effort;

- needs assessment
- selection of candidate products
- evaluation of candidate products
- introduction of product to user groups, management, stakeholders
- gathering feedback from target groups
- implementation planning and execution

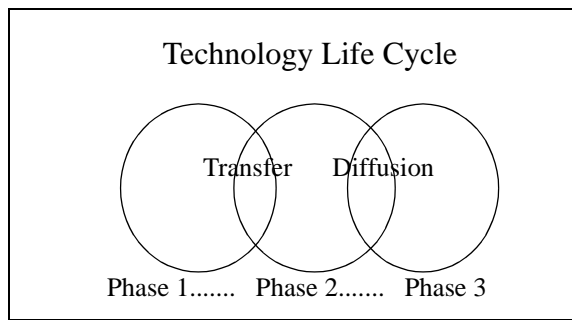


Figure 6. Technology life cycle phases adapted from [CMU/SEI-93-TR-31].

We will use the steps outlined for the 3 phases of the technology life cycle in describing the MATE effort and will reference the tools and technology maturity with regard to this structure as appropriate.

Test execution was comprised of primarily mature technologies in phase 3 of the technology lifecycle, yet aspects of the total solution for full automation and integration might represent less mature technologies. Test case generation technologies were seen as being in the transfer phase, or phase 2, of the technology lifecycle and experiencing the early stages of transfer. The technology lifecycle maturity of the test management technologies are typically in the third phase of the lifecycle. Test measurement technologies were primarily in phase 3 of the technology lifecycle and represented the diffusion of measurement technologies.

The MATE effort as described using the SEI's technology lifecycle conceptual framework and technology lifecycle maturity evaluation depicts an effort towards a common test environment that is comprised of technologies in various phases of maturity. The majority of MATE technologies are in phase 3 of the technology lifecycle and are well-supported for successful transition.

REFERENCES

1. Brown, Alan W., Earl, Anthony N. and McDermid, John A., Software Engineering Environments: Automated

- Support for Software Engineering, McGraw-Hill, 1992.
2. Eickelmann, Nancy S., "Emerging Test Technologies: Shifting Tester Requirements," In the Proceedings of the 17th International Conference on Testing Computer Software, June 12-16, 2000.
3. Eickelmann, Nancy S., "Measuring and Evaluating the Software Test Process." European Software Measurement Conference, FESMA '98, Antwerp, Belgium, May 6-8, 1998.
4. Eickelmann, Nancy S. and Richardson, Debra J., "Leveraging the Cost of Software Testing with Measurable Process Improvement," In the Proceedings of the Computing in Engineering Conference, ETCE-ASME '97, Houston, Texas, January 28-30, 1997.
5. Eickelmann, Nancy S. and Richardson, Debra J. "An Evaluation of Software Test Environment Architectures" in proceeding of the Eighteenth International Conference of Software Engineering, 1996.
6. Fowler, Priscilla and Levine, Linda "A Conceptual Framework for Software Technology Transition", Software Engineering Institute Technical Report CMU/SEI-93-TR-031, December 1993.
7. D. Garlan, G. Kaiser, and D. Notkin. "Using tool abstraction to compose systems." IEEE Computer, vol. 25, June 1992.
8. D. Garlan and M. Shaw. "An introduction to software architecture.". Advances in Software Engineering and Knowledge Engineering, Volume I, World Scientific Publishing Co., 1993.
9. D. Gelperin and B. Hetzel. "The growth of software testing." Communications of the ACM, 31(6):687-695, June 1988.
10. R. Kazman, L. Bass, G. Abowd, and M. Webb. "SAAM: A method for analyzing the properties of software architectures." In Proceedings of the Sixteenth International Conference on Software Engineering, 81-90, Sorrento, Italy, May 21, 1994.
11. G. J. Myers. The art of software testing. New York, John Wiley and Sons, 1978.
12. E. Miller. Mechanizing software testing. TOCG Meeting, Westlake Village, California, April 15, 1986.
13. Guideline for Lifecycle Validation, Verification, and Testing of Computer Software. National Bureau of Standards Report NBS FIPS 101. Washington, D.C., 1983.

