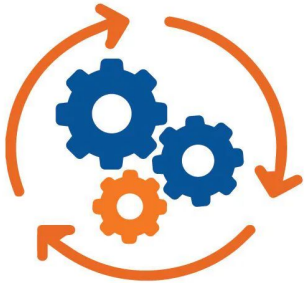
Chapter 1

The Birth and Evolution of Automated Testing



Lecturer: Nguyen Thanh Quan (MSc)
Email: tg_nguyenthanhquan_cntt@tdtu.edu.vn

Content

1. Automation Testing
2. Background on Software Testing
3. The Automated Test Life-Cycle Methodology (ATLM)
4. ATLM’s Role in the Software Testing Universe
5. Software Testing Careers

Introduction

An effective test program:

- Incorporates the automation of software testing
- Involves a mini-development life cycle of its own

Automated testing amounts to a development effort involves:

- Strategy and goal planning
- Test requirement definition
- Analysis
- Design
- Development
- Execution
- Evaluation activities

Automation Testing

Challenges (1)

REQ1: “We need the new software application sooner than that”

REQ2: “I need those new product features now.”

Challenges (2)

Software managers and developers deal with

- Ever-shrinking schedules
- Minimal resources

90% of developers have missed ship dates.

67% of developers have missing deadline routines

91% forced to remove key functionality late in the development

Getting a product to market **as early as possible** may mean the difference between product **survival and product death** - company **survival and death**

Missions

Reduce cost

Minimize the development schedule

Require incremental software builds

Do more with less

“The management and performance of test activities, to include the **development and execution of test scripts** so as to **verify test requirements, using an automated test tool**.”

When to use automation testing

The automation of test activities provides its greatest value in instances where:

- Test scripts are repeated
- Test script subroutines are created and then invoked repeatedly by a number of test scripts

Reusable scripts may be run a great number of times, offers a significant payback.

Benefits (1)

- The performance of integration test using an automated test tool for subsequent incremental software builds provides great value.

- Given the continual changes and additions to requirements and software, automated software test serves as an important control mechanism to ensure accuracy and stability of the software through each build.

Benefits (2)

Regression tests seek to verify that the functions provided by a modified system or software product perform as specified and that no unintended change has occurred in the operation of the system or product.

➔ **Automated testing allows for regression testing to be executed in a more efficient manner.**

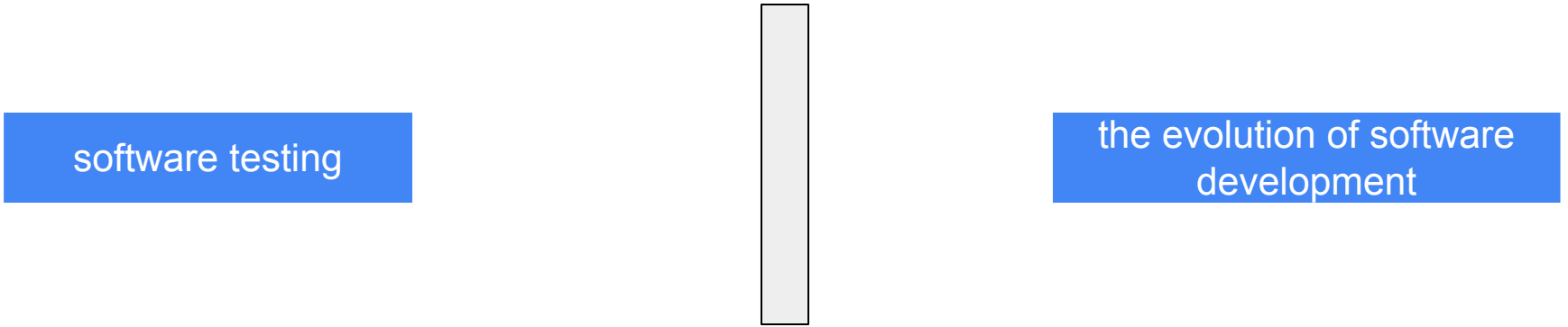
Categories

Databases, and involve both front-end and back-end processing

- Functional requirement testing
- Server performance testing
- User interface testing
- Unit testing
- Integration testing
- Program module complexity analysis
- Program code coverage testing
- System load performance testing
- Boundary testing
- Security testing
- Memory leak testing
- …

Background on Software Testing

Software Testing and Software Development



The history of software testing mirrors the evolution of software development itself.

Former tests

The era of software development focused on large-scale scientific and Defense Department programs coupled with corporate database systems developed on mainframe or minicomputer platforms.

- Test scenarios were written down on **paper**
- Tests targeted **control flow paths, computations of complex algorithms, and data manipulation.**
- Testing was initiated at the **end** of the **project schedule**, **performed** by **personnel** who were available at the time.

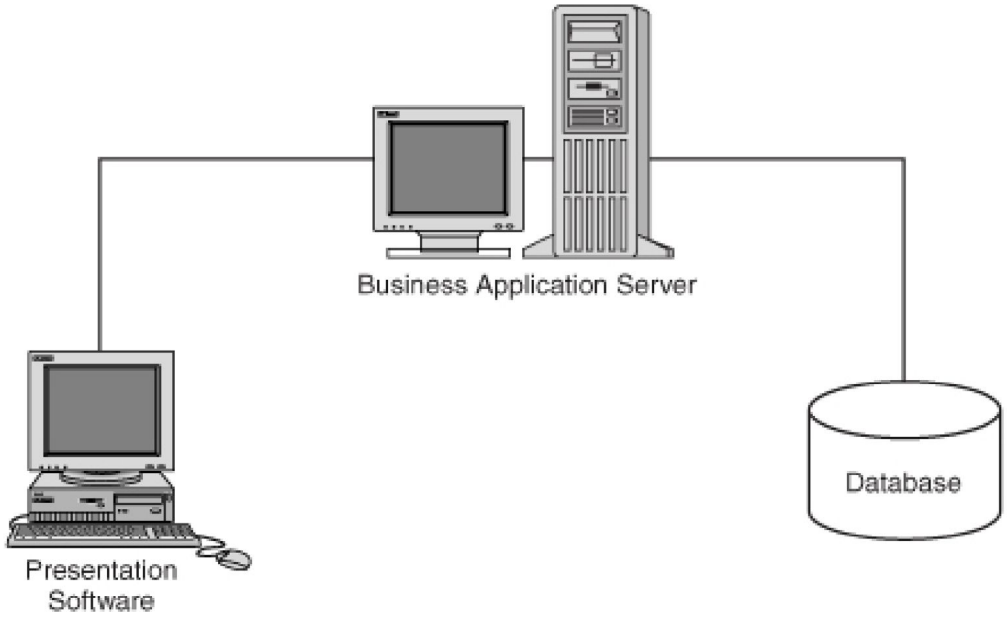
Testing development

The era of personal computers

- the explosive growth of **commercial software development.**
- **on-line systems** replacing batch-mode operation.
- the test effort for online systems **required a different approach** to test design.

Graphical User Interface (GUI)

The terms of Graphical User Interface (GUI)/Client - Server



Testing with GUI (1)

The terms of Graphical User Interface (GUI)/Client - Server

- Introduces new **complexity** into the test effort.
- The test engineer is **no longer exercising a single**, closed application operating on a single system

The client-server architecture involves three separate components:

- **Server**
- **Client**
- **Network**

➔ Testing is concerned with the **performance of the server and the network**, as well as the **overall system performance and functionality** across the three components.

Testing with GUI (2)

The terms of Graphical User Interface (GUI)

- The nature of **GUI screens presents some additional challenges.**
- The **GUI replaces character-based applications.**
- **GUIs involve the presentation of information** in user screen windows.
- **Event-driven environmen**t (screen size change, image's position etc.)

➔ GUI application development became **attractive to test applications**.

➔ **Personnel** performing the test were **required to be more familiar** with the application

➔ **Personnel** need to have **more specific skill requirements** relevant to the platforms and network

The Automated Test Life-Cycle Methodology (ATLM)

Early involvement benefits

- To achieve these benefits, test activity and test planning must be **initiated early** in the project.
- **Test engineers** need to be **included** during **business analysis.**
- **Requirements** activities and be involved in **analysis** and **design** review activities.

➔ **Early involvement** allows the test team to **gain understanding** of the customer needs.
➔ Not only supports **effective test design.**
➔ Also provides **early detection of errors.**
➔ **Prevents migration of errors** from requirement specification to design.
➔ **Reduces cost, minimizes rework, and saves time.**

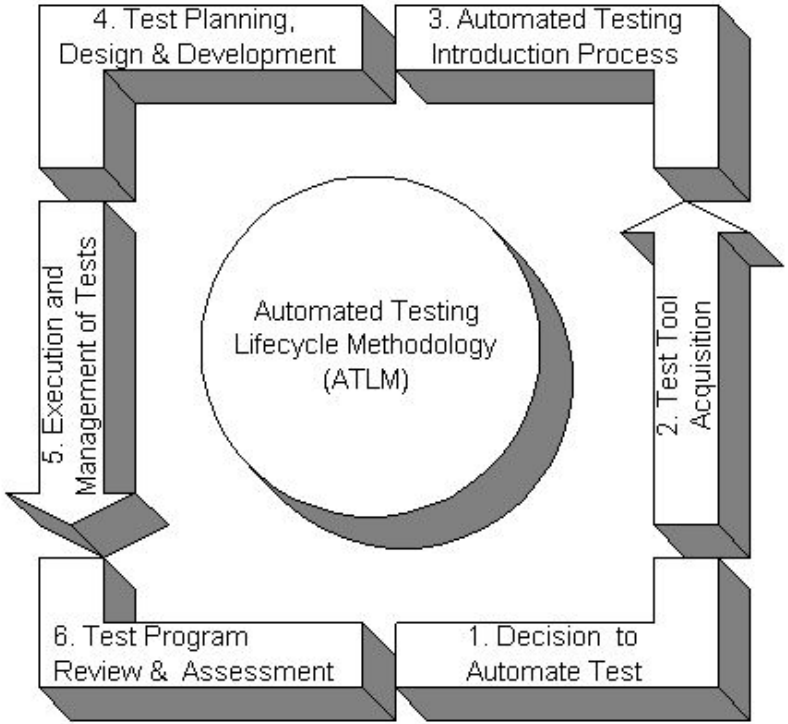
Error removal cost

Phase	Cost
Definition	$1
High-Level Design	$2
Low-Level Design	$5
Code	$10
Unit Test	$15
Integration Test	$22
System Test	$50
Post-Delivery	$100+

Error removal cost multiplies over system development life cycle

ATLM in the software development

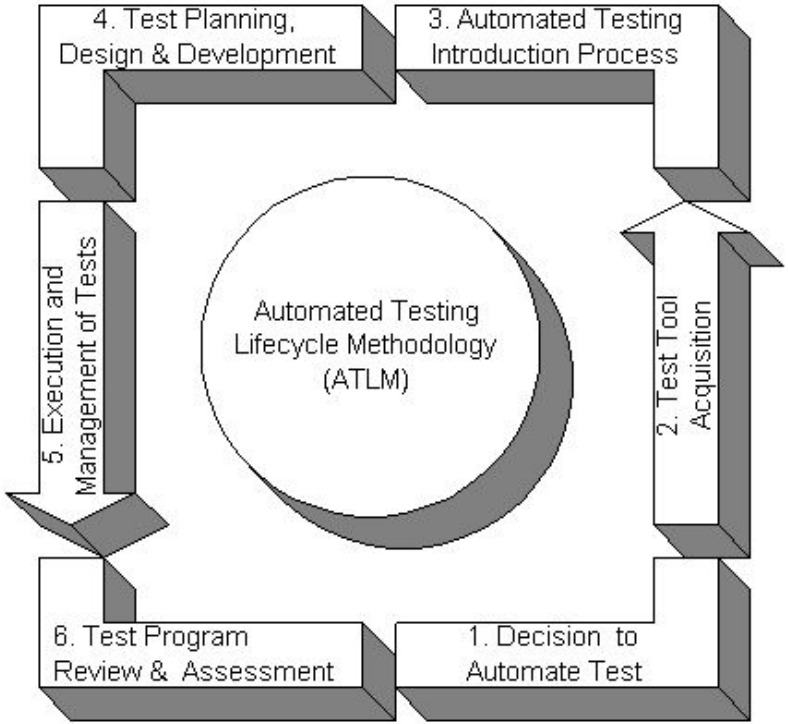
The ATLM approach mirrors the benefits of **modern rapid application development** efforts, where such efforts **engage** the user **early** on throughout **analysis**, **design**, and **development** of each software version, which is built in an incremental fashion.



Benefits of ATLM

The auxiliary benefit of using a test methodology like ATLM

- **parallels** the development life cycle is the development of a close working relationship between software developers and the test engineers
- f**osters greater cooperation** and makes possible better results during unit, integration, and system testing.



Testing critera

The functional specifications must be evaluated, at a minimum, using the criteria:

- **Completeness**: Evaluate the extent to which the requirement is thoroughly defined.
- **Consistency**: Ensure that each requirement does not contradict other requirements.
- **Feasibility**: Evaluate the extent to which a requirement can actually be implemented with the available technology, hardware specifications, project budget and schedule, and project personnel skill levels.
- **Testability**. Evaluate the extent to which a test method can prove that a requirement has been successfully implemented.

Test Strategy (1)

- Test strategies should be **determined** d**uring the functional specification/requirements phase**.
- **Automated tools** can help produce functional requirements that are testable, **minimize** the **effort** and **cost** of testing.
- With test automation in mind, the product design and coding standards can **provide** the **proper environment** to get the most out of the test tool.

Test Strategy (2)

ATLM supports:

- test efforts involving automated test tools, incorporates a multistage process
- the detailed and interrelated activities that are required to decide whether to employ an automated testing tool.
- the development and management of test data and the test environment
- describing a way to develop test documentation so as to account for problem reports.

Test Mistakes

The ATLM represents a structured approach which helps steer the test team away from several common test program mistakes:

- Implementing the use of an automated test tool without a testing process in place, which results in an ad hoc, nonrepeatable, nonmeasurable test program.
- Implementing a test design without following any design standards.
- Attempting to automate 100% of test requirements, when the tools being applied do not support automation of all tests required.
- Using the wrong tool.
- Initiating test tool implementation too late in the application development life cycle.
- Involving test engineers too late in the application development life cycle.

Automated Testing Stages

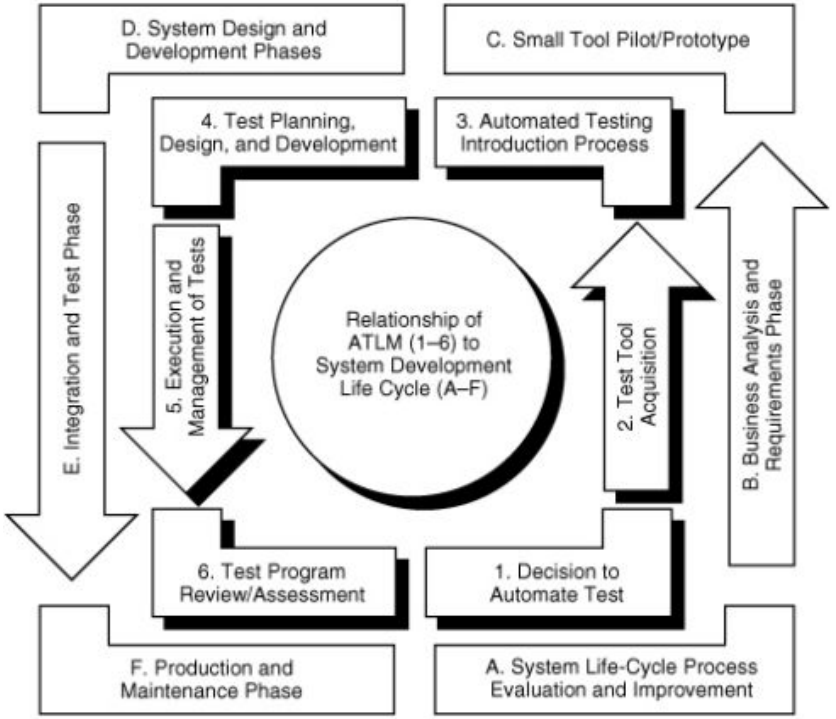
Decision to Automate	
1.1 Automated Test Expectations	Chapter 2
1.2 Benefits of Automated Test	Chapter 2
1.3 Acquiring Management Support	Chapter 2
Test Tool Acquisition	
2.1 Review System Engineering Environment	Chapter 3
2.2 Review Tools Available on the Market	Chapter 3
2.3 Tool Research and Evaluation	Chapter 3
2.4 Tool Purchase	Chapter 3
Introduction of Automated Testing	
3.1 Test Process Analysis	Chapter 4
3.2 Test Tool Consideration	Chapter 4

Test Planning, Design, and Development	
4.1 Test Plan Documentation	Chapter 6
4.2 Test Requirements Analysis	Chapter 7
4.3 Test Design	Chapter 7
4.4 Test Development	Chapter 8
Execution and Management of Automated Test	
5.1 Automated Test Execution	Chapter 9
5.2 Testbed Baseline	Chapter 8
5.3 Defect Tracking	Chapter 9
5.4 Test Progress Tracking	Chapter 9
5.5 Test Metrics	Chapter 9
Process Evaluation and Improvement	
6.1 Post-Release: Test Process Improvement	Chapter 10

ATLM’s Role in the Software Testing Universe

SDLF vs ATLM Relationship

For **maximum** test program benefit, the ATLM approach needs to be pursued in **parallel** with the system life cycle.



System Development Life Cycle - ATLM Relationship

Test Maturity Model (TMM)

- TMM was developed by the Illinois Institute of Technology and augmented by Automated Software Testing Maturity.
- Test teams that implement the ATLM will make progress toward levels 4 and 5 of TMM
- TMM contains a set of maturity levels.
- TMM promotes greater professionalism in software testing, similar to the intention of the Capability Maturity Model (CMM) for software, which was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University.

Correlation Between the CMM and TMM

- The TMM was developed as a complement to the CMM.
- Testing process is a subset of the overall software development process.
- Any organization that wishes to improve its testing process throughout implementation of the TMM (and ATLM) should first commit to improving its overall software development process by applying the CMM guidelines.
- A good software test program cannot stand alone.
- A good software test program must be an integral part of the software development process.

Testing Maturity vs AST Maturity Levels 1-5 (1)

TMM Level 1	Automated Software Testing Level 1
Initial. Testing is a chaotic process; it is ill defined and not distinguished from debugging. Tests are developed in an ad hoc way after coding is complete. Testing and debugging are interleaved to get the bugs out of the software. The objective of testing is to show that the software works [9]. Software products are released without quality assurance. Resources, tools, and properly trained staff are lacking. This type of organization would be at level 1 of the CMM developed by the Software Engineering Institute, There are no maturity goals at this level.	This level of automated software testing is referred to as "accidental automation." At the first level, automated testing is not done at all or only on an ad hoc basis. An automated test tool might be used on an experimental basis. With a capture/ playback tool, automated test scripts are recorded and played back with only tool-generated scripts being used. Scripts are not modified for reusability or maintainability. No automated script design or development standards are followed. Resulting scripts are not reusable and difficult to maintain and must be recreated with each software build. This type of automation can actually increase testing costs by 125% or more—for example, 150% of manual test costs with each test cycle (see "Case Study: Value of Test Automation Measurement" in Chapter 2).

Testing Maturity vs AST Maturity Levels 1-5 (2.1)

TMM Level 2 (1)	Automated Software Testing Level 2 (1)
Phase Definition. Testing is separated from debugging and is defined as a phase that follows coding. Although it is a planned activity, test planning at level 2 may occur after coding for reasons related to the immaturity of the test process, For example, at level 2 there is the perception that all testing is execution-based and dependent on the code, and therefore it should be planned only when the code is complete.	“At this level, testing is becoming a planned activity. This implies a commitment to the completion of testing activities. A project planning tool will aid the project manager in defining test activities, allocating time, money, resources, and personnel to the testing process” [11]. This level of automated software testing is referred to as “incidental automation.” At the second level, automated test scripts are modified but no documented standards or repeatability exists.

Testing Maturity vs AST Maturity Levels 1-5 (2.2)

TMM Level (2)	Automated Software Testing Level 2 (2)
The primary goal of testing at this level of maturity is to show that the software meets its specifications [10]. Basic testing techniques and methods are in place. Many quality problems at this TMM level occur because test planning takes place late in the software life cycle. In addition, defects propagate into the code from the requirements and design phases, as no review programs address this important issue. Post-code, execution-based testing is still considered the primary testing activity.	The types of tools used at this level can include project planning tools, capture / playback tools, simulators and emulators, syntax and semantic analyzers, and debugging tools. The introduction of automated test tools to a new project is not planned and a process is not being followed. Test design or development standards do not exist. Test schedules or test requirements are not taken into consideration or are not reliable when contemplating the use of an automated test tool. As with level 1, this type of test automation does not provide much return on investment and can actually increase the testing effort.

Testing Maturity vs AST Maturity Levels 1-5 (3.1)

TMM Level 3 (1)	Automated Software Testing Level 3 (1)
Integration. Testing is no longer a phase that follows coding; rather, it is integrated into the entire software life cycle, Organizations can build on the test planning skills they have acquired at level 2. Unlike level 2 (planning for testing at TMM), level 3 begins at the requirements phase and continues throughout the life cycle supported by a version of the V model [12].	This level of test maturity is referred to as “intentional automation.” At the third level, automated testing becomes both well defined and well managed. The test requirements and the test scripts themselves proceed logically from the software requirement specifications and design documents.

Testing Maturity vs AST Maturity Levels 1-5 (3.2)

TMM Level 3 (2)	Automated Software Testing Level 3 (2)
Test objectives are established with respect to the requirements based on user and client needs and are used for test-case design and success criteria, A test organization exists, and testing is recognized as a professional activity. A technical training organization pursues a testing focus. Basic tools support key testing activities. Although organizations at this level begin to realize the important role of reviews in quality control, no formal review program has been established, and reviews do not yet take place across the life cycle. A test measurement program has not yet been established to qualify process and product attributes.	Automated test scripts are created based on test design and development standards, yet the test team does not review automated test procedures, Automated tests become more reusable and maintainable. At this level of automated testing, the return on investment is starting to pay off and a break-even point can already be achieved by the second-regression test cycle (see “Case Study: Value of Test Automation Measurement” in Chapter 2). The types of tools used at this level include requirement management tools, project planning tools, capture//playback tools, simulators and emulators, syntax and semantic analyzers, and debugging tools.

Testing Maturity vs AST Maturity Levels 1-5 (4)

TMM Level 4	Automated Software Testing Level 4
Management and Measurement. Testing is a measured and quantified process. Reviews at all phases of the development process are now recognized as testing and quality-control activities. Software products are tested for quality attributes such as reliability, usability, and maintainability, Test cases from all projects are collected and recorded in a test case database to test case reuse and regression testing. Defects are logged and given a severity level, Deficiencies in the test process now often follow from the lack of a defect prevention philosophy and the porosity of automated support for the collection, analysis, and dissemination of test-related metrics.	This level of test maturity, referred to as “advanced automation,” can be achieved when adopting many aspects of the ATLM described in this book. This testing maturity level represents a practiced and perfected version of level 3 with one major addition post-release defect tracking. Defects are captured and sent directly back through the fix, test creation, and regression test processes, The software test team is now an integral part of product development, and test engineers and application developers work together to build a product that will meet test requirements. Any software bugs are caught early, when they are much less expensive to fix. In addition to the tools mentioned at the previous testing levels, defect and change tracking tools, test procedure generation tools, and code review tools are used at this level.

Testing Maturity vs AST Maturity Levels 1-5 (5)

TMM Level 5	AST Level 5
Optimization, Defect Prevention, and Quality Control. Because of the infrastructure provided by the attainment of maturity goals at levels 1 through 4 of the TMM, the testing process is now said to be defined and managed and its cost and effectiveness can be monitored, At level 5, mechanisms fine-tune and continuously improve testing. Defect prevention and quality control are practiced, The testing process is driven by statistical sampling and measurements of confidence levels, trustworthiness, and reliability. An established procedure exists for the selection and evaluation of testing tools. Automated tools totally support the running and rerunning of test cases, providing support for test-case design, maintenance of test-related items, defect collection and analysis, and the collection, analysis, and application of test-related metrics.	When incorporating the guidelines of the ATLM described in this book and using the applicable tools in an efficient manner, a TMM level 5 maturity can be achieved. Tools used at this highest level include the ones mentioned within previous levels plus test data generation tools and metrics collection tools, such as complexity and size measurement tools, coverage and frequency analyzers, and statistical tools for defect analysis and defect prevention, (All tools described at the various levels are discussed in detail in Chapter 3, and tool examples are provided in Appendix B.}

Test Automation Development (1)

- Modularity
- Shared function libraries
- Use of variables
- Parameter passing
- Conditional branching
- Loops
- Arrays
- Subroutines

➔ Not only the software developer, but also the software test engineer

Test Automation Development (2)

- The development of an automated test can be viewed as a **mini-development** life cycle.
- Automated test development requires **careful design and planning**.
- BASIC, C, or C++, Python, etc code can be **modified** and **reused** to serve as automated test scripts for other applications
- Similar to software development, the software test engineer has a set of (test) **requirements**, **blueprint** (design), a **product** (test procedures/scripts)
- The use of **variables**, **looping** constructs, **API** call/use etc.
- The software developer and the software test engineer therefore have **similar missions**.
- Test scripts must be **reusable**.

Test Automation Development (3)

- **Maintainability** is as important to the test engineering product.
- The test team's test scripts need to be structured in ways that can support **global changes** because products under development will change.
- Test engineers must therefore focus their **test activity** on the portion of the application that exercises the majority of **system requirements**.

Test Automation Development (4)

- Software tests require that software test engineers begin to use **analytical skills** and **perform** more of the same type of design and development tasks that software developers carry out.
- Test engineers need to **follow the lead of test team** personnel who have more experience with the test tool or who have significantly more programming experience.
- In GUI-based client-server system environments, test team personnel need to be **more like a system engineer** as well as **a software developer** (knowledge and understanding of LANs, OS, Databases etc.
- In the future, the **skills** required for both the **software developer** and the **software test engineer** will continue to **converge**.

Test Effort

"We have only one week left before we are supposed to initiate testing. Should we use an automated test tool?"

- Building an infrastructure of automated test scripts, for the purpose of executing testing in **quick fashion**, requires considerable investment in test planning and preparation.
- The test team needs to be **cautious** when it encounters statements to this effect.
- Ad hoc implementation of automated testing can actually increase testing costs by **125%** or even **150%** of manual test costs with each test cycle.

Test Effort

"We have only one week left before we are supposed to initiate testing. Should we use an automated test tool?"

- In fact, during the **first use** of a particular automated test tool by a test team, **no or very little savings may be realized**.
- The use of automated tools may **increase the scope and breadth** of the test effort.
- The use of automated testing introduces a new level of **complexity** (especially in case of inexperience).
- On a new project, it is important to **listen** for the level of **expectations** communicated by the **project manager or other project personnel**.

Software Testing Careers

Expectation

"I like to perform a variety of different work, learn a lot of things, and touch a lot of different products."

"I also want to exercise my programming and database skills, but I don't want to be off in my own little world, isolated from others, doing nothing but hammering out code."

Reasons to choose automated test careers

The **evolution of automated test capabilities** has given birth to many new career opportunities for software engineers.

Many software engineers are choosing careers in the automated test arena for **two reasons:**

- **The different kinds of tasks involved.**
- **The variety of applications for which they are introduced.**

➔ Automated testing knowledge, understanding and skills are highly required.

Requirements for test engineers

"How do I know whether I would make a good test engineer?"

- Be able to **make things fail**.
- Require a developer's **mentality** to develop **work-around solutions**.
- Need to be **structured**, attentive to **detail**, and **organized**, given the **complexities** of automated testing.
- Possess a **creative and planning-ahead type of mindset**.
- Be both **assertive** and **poised** when working through **trouble reports** and **issues with developers**.
- Should have a broad range of **technical skills.**

Test Career Progression (1)

Career Progression	Description
Junior Test Engineer	An entry-level position for an individual with a computer science degree or an individual with some manual testing experience. Develops test scripts and begins to become familiar with the test life cycle and testing techniques.
Test Engineer/Programmer Analyst	A test engineer or programmer who has one to two years of experience. Programs automated test scripts and provides early leadership in test programming. Further develops skills in programming languages, operating systems, networks, and databases.

Test Career Progression (2)

Career Progression	Description
Senior Test Engineer/ Programmer Analyst	A test engineer or programmer who has three to four years of experience. Helps to develop or maintain testing or programming standards and processes, leads peer reviews, and acts as a mentor to other, more junior test engineers or programmers. Continues to develop skills in programming, languages, operating systems, networks, and databases.
Team Lead	A test engineer or programmer who has four to six years of experience, Responsible for the supervision of one to three test engineers or programmers. Has some scheduling and effort size /cost estimation responsibilities, Technical skills become more focused.

Test Career Progression (3)

Career Progression	Description
Test/ Programming Lead	A test engineer or programmer who has six to ten years of experience. Responsible for the supervision of four to eight personnel. Responsible for scheduling, size/cost estimation, and delivery of product within schedule and budget targets. Responsible for developing the technical approach to the project. Provides some customer support and presentations. Develops technical expertise in a few particular areas.
Test/QA/Development (Project) Manager	Ten-plus years of experience. Responsible for the eight or more personnel performing on one or more projects. Full development life cycle responsibility within this area (test/QA/development). Provides some customer interaction and many presentations. Has cost, schedule, planning, and staffing responsibility.

Test Career Progression (4)

Career Progression	Description
Program Manager	Fifteen-plus years of development and support (test/QA) activity experience, Responsible for personnel performing on several projects and full development life cycle responsibility. Assumes project direction and profit/loss responsibility.

Summary (1)

- The incremental release of software, together with the development of GUI-based client-server or multitier applications, introduces new complexity to the test effort.
- Organizations want to test their software adequately, but within a minimum schedule.
- Rapid application development approach calls for repeated cycles of coding and testing.
- The automation of test activities provides its greatest value in instances where test scripts are repeated or where test script subroutines are created, and then invoked repeatedly by a number of test scripts.

Summary (2)

- The use of automated test tools to support the test process is proving to be beneficial in terms of product quality and minimizing project schedule and effort.
- The Automated Testing Life-cycle Methodology (ATLM) includes six components and represents a structured approach with which to implement and execute testing.
- Early life-cycle involvement by test engineers in requirements and design review (as emphasized by the ATLM) bolsters test engineers' understanding of business needs, increases requirements testability, and supports effective test design and development - a critically important activity when utilizing an automated test tool.

Summary (3)

- Test support for finding errors early in the development process provides the most significant reduction in project cost.
- Personnel who perform testing must be more familiar with the application under test and must have specific skills relevant to the platforms and network involved as well as the automated test tools being used.
- Automated software testing, using automated test tools, is a development activity that includes programming responsibilities similar to those of the application-under-test software developer.

Summary (4)

- Building an infrastructure of automated test scripts for the purpose of executing testing in quick fashion requires a considerable investment in test planning and preparation.
- An effective test program, incorporating the automation of software testing, involves a development life cycle of its own.
- Software test engineering can provide for an interesting and challenging work assignment and career, and the marketplace is in high demand for test engineering skills.

References

1. CenterLine Software, Inc. Survey. 1996. CenterLine is a software testing tool and automation company in Cambridge, Massachusetts.
2. http://www.standishgroup.com/chaos.html.
3. Littlewood, B. *How Good Are Software Reliability Predictions? Software Reliability Achievement and Assessment*. Oxford: Blackwell Scientific Publications, 1987.
4. Burnstein, I., Suwanassart, T., Carlson, C.R. *Developing a Testing Maturity Model,* Part II. Chicago: Illinois Institute of Technology, 1996.
5. Ibid.
6. Paulk, M., Weber, C., Curtis, B., Chrissis, M. *The Capability Maturity Model Guideline for Improving the Software Process.* Reading, MA: Addison-Wesley, 1995.
7. Paulk, M., Curtis, B., Chrissis, M., Weber, C. "Capability Maturity Model, Version 1.1." *IEEE Software* July 1993: 18–27.
8. Paulk, M., et al. "Key Practices of the Capability Maturity Model, Version 1.1." Technical Report CMS/SEI-93-TR-25. Pittsburgh, PA: Software Engineering Institute, 1993.
9. Gelperin, D., Hetzel, B. "The Growth of Software Testing." CACM 1998;31:687–695
10. Daich, G., Price, G., Ragland, B., Dawood, M. "Software Test Technologies Report." STSC, Hill Air Force Base, Utah, August 1994.
11. Maglitta, M. "Quality Assurance Assures Lucrative Contracts." *Contract Professional* Oct./Sept. 1997.
12. Bernstein, L. Pacific Northwest Software Quality Conference (1997). *TTN On-Line Edition Newsletter* December 1997.