دانشگاه آزاد اسلامی واحد تبریز

نام درس: آزمون نرم افزار

بخش: معیارها در آزمون نرم افزار
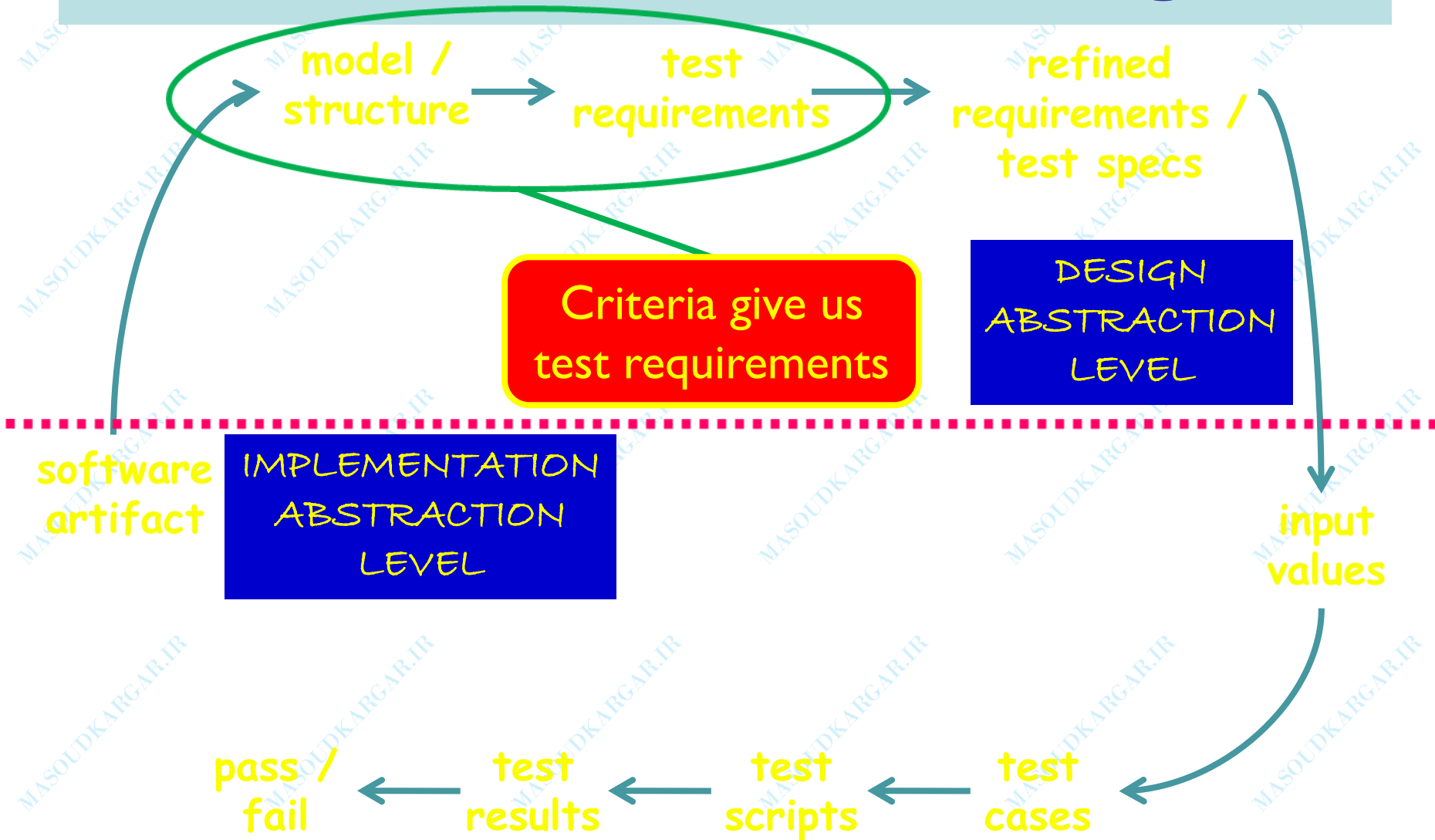
نام استاد: دکتر مسعود کارگر

Changing Notions of Testing

- Old view focused on testing at each software development phase as being very different from other phases

  _Unit, module, integration, system …

- New view is in terms of structures and criteria

  _input space, graphs, logical expressions, syntax

- Test design is largely the same at each phase

  _Creating the model is different

  _Choosing values and automating the tests is different

Model-Driven Test Design

model / structure → test requirements → refined requirements / test specs

Criteria give us test requirements

DESIGN ABSTRACTION LEVEL

software artifact

IMPLEMENTATION ABSTRACTION LEVEL

input values

pass / fail ← test results ← test scripts ← test cases ← input values

درس : آزمون نرم افزار   استاد : دکترمسعودکارگر   دانشگاه آزاداسلامی واحد تبریز

New : Test Coverage Criteria

A tester's job is simple :

Define a model of the software, then find ways to cover it

- Test Requirements : A specific element of a software artifact that a test case must satisfy or cover

- Coverage Criterion : A rule or collection of rules that impose test requirements on a test set

Testing researchers have defined dozens of criteria, but they are all really just a few criteria on four types of structures …

Source of Structures

- These structures can be extracted from lots of software artifacts

  _Graphs can be extracted from UML use cases, finite state machines, source code, …

  _Logical expressions can be extracted from decisions in program source, guards on transitions, conditionals in use cases, …

- This is not the same as "model-based testing," which derives tests from a model that describes some aspects of the system under test

  _The model usually describes part of the behavior

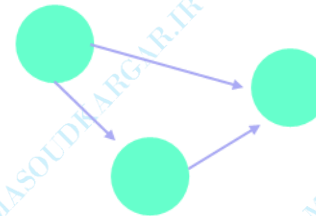 _The source is explicitly not considered a model

Criteria Based on Structures

Structures : Four ways to model software

1. Input Domain Characterization (sets)

```
A: {0, 1, >1}
B: {600, 700, 800}
C: {swe, cs, isa, infs}
```

2. Graphs

3. Logical Expressions

```
(not X or not Y) and A and B
```

4. Syntactic Structures (grammars)

```
if (x > y)
    z = x - y;
else
    z = 2 * x;
```

Example : Jelly Bean Coverage

Flavors :

1. Lemon
2. Pistachio
3. Cantaloupe
4. Pear
5. Tangerine
6. Apricot

Colors :

1. Yellow (Lemon, Apricot)
2. Green (Pistachio)
3. Orange (Cantaloupe, Tangerine)
4. White (Pear)

- Possible coverage criteria :

  1. Taste one jelly bean of each flavor

  Deciding if yellow jelly bean is Lemon or Apricot is a controllability problem

  2. Taste one jelly bean of each color

Coverage

Given a set of test requirements TR for coverage criterion C, a test set T satisfies C coverage if and only if for every test requirement tr in TR, there is at least one test t in T such that t satisfies tr

- Infeasible test requirements : test requirements that cannot be satisfied
  - _No test case values exist that meet the test requirements
  - _Example: Dead code
  - _Detection of infeasible test requirements is formally undecidable for most test criteria
- Thus, 100% coverage is impossible in practice

More Jelly Beans

T1 = { three Lemons, one Pistachio, two Cantaloupes, one Pear, one Tangerine, four Apricots }

- Does test set T1 satisfy the flavor criterion ?

T2 = { One Lemon, two Pistachios, one Pear, three Tangerines }

- Does test set T2 satisfy the flavor criterion ?

- Does test set T2 satisfy the color criterion ?

Coverage Level

The ratio of the number of test requirements satisfied by T to the size of TR

- T2 on the previous slide satisfies 4 of 6 test requirements

Two Ways to Use Test Criteria

1. **Directly generate** test values **to satisfy** the criterion
   - _Often assumed by the research community
   - _Most obvious way to use criteria
   - _Very hard without automated tools

2. Generate test values **externally** and **measure** against the criterion
   - _Usually favored by industry
   - _Sometimes misleading
   - _If tests do not reach 100% coverage, what does that mean?

> Test criteria are sometimes called metrics

Generators and Recognizers

- Generator : A procedure that automatically generates values to satisfy a criterion

- Recognizer : A procedure that decides whether a given set of test values satisfies a criterion

- Both problems are provably undecidable for most criteria

- It is possible to recognize whether test cases satisfy a criterion far more often than it is possible to generate tests that satisfy the criterion

- Coverage analysis tools are quite plentiful

Comparing Criteria with Subsumption (5.2)

- Criteria Subsumption : A test criterion C1 subsumes C2 if and only if every set of test cases that satisfies criterion C1 also satisfies C2

- Must be true for every set of test cases

- Examples :

  _The flavor criterion on jelly beans subsumes the color criterion … if we taste every flavor we taste one of every color

  _If a test set has covered every branch in a program (satisfied the branch criterion), then the test set is guaranteed to also have covered every statement

Advantages of Criteria-Based Test Design (5.3)

- Criteria maximize the "bang for the buck"

    _Fewer tests that are more effective at finding faults

- Comprehensive test set with minimal overlap

- Traceability from software artifacts to tests

    _The "why" for each test is answered

    _Built-in support for regression testing

- A "stopping rule" for testing—advance knowledge of how many tests are needed

- Natural to automate

Characteristics of a Good Coverage Criterion

1. It should be fairly easy to compute test requirements automatically

2. It should be efficient to generate test values

3. The resulting tests should reveal as many faults as possible

- Subsumption is only a rough approximation of fault revealing capability
- Researchers still need to gives us more data on how to compare coverage criteria

Test Coverage Criteria

- Traditional software testing is expensive and labor-intensive
- Formal coverage criteria are used to decide which test inputs to use
- More likely that the tester will find problems
- Greater assurance that the software is of high quality and reliability
- A goal or stopping rule for testing
- Criteria makes testing more efficient and effective

How do we start applying these ideas in practice?

How to Improve Testing ?

- Testers need more and better software tools
- Testers need to adopt practices and techniques that lead to more efficient and effective testing
  - _More education
  - _Different management organizational strategies
- Testing & QA teams need more technical expertise
  - _Developer expertise has been increasing dramatically
- Testing & QA teams need to specialize more
  - _This same trend happened for development in the 1990s

Four Roadblocks to Adoption

1. Lack of test education

Microsoft and Google say half their engineers are testers, programmers test half the time

Number of UG CS programs in US that require testing ? **0**

Number of MS CS programs in US that require testing ? **0**

Number of UG testing classes in the US ? **~50**

2. Necessity to change process

Adoption of many test techniques and tools require changes in development process
This is expensive for most software companies

3. Usability of tools

Many testing tools require the user to know the underlying theory to use them
Do we need to know how an internal combustion engine works to drive ?
Do we need to understand parsing and code generation to use a compiler ?

4. Weak and ineffective tools

Most test tools don't do much – but most users do not realize they could be better
Few tools solve the key technical problem – generating test values automatically

Needs From Researchers

1. Isolate : Invent processes and techniques that isolate the theory from most test practitioners
2. Disguise : Discover engineering techniques, standards and frameworks that disguise the theory
3. Embed : Theoretical ideas in tools
4. Experiment : Demonstrate economic value of criteria-based testing and ATDG (ROI)

   _Which criteria should be used and when ?

   _When does the extra effort pay off ?
5. Integrate high-end testing with development

Needs From Educators

1. Disguise theory from engineers in classes

2. Omit theory when it is not needed

3. Restructure curricula to teach more than test design and theory
   _Test automation
   _Test evaluation
   _Human-based testing
   _Test-driven development

Changes in Practice

1. **Reorganize** test and QA teams to make effective use of individual abilities

   _One math-head can support many testers

2. **Retrain** test and QA teams

   _Use a process like MDTD

   _Learn more testing concepts

3. **Encourage** researchers to embed and isolate

   _We are very responsive to research grants

4. **Get involved** in curricular design efforts through industrial advisory boards

Criteria Summary

- Many companies still use "monkey testing"
  - A human sits at the keyboard, wiggles the mouse and bangs the keyboard
  - No automation
  - Minimal training required
- Some companies automate human-designed tests
- But companies that use both automation and criteria-based testing
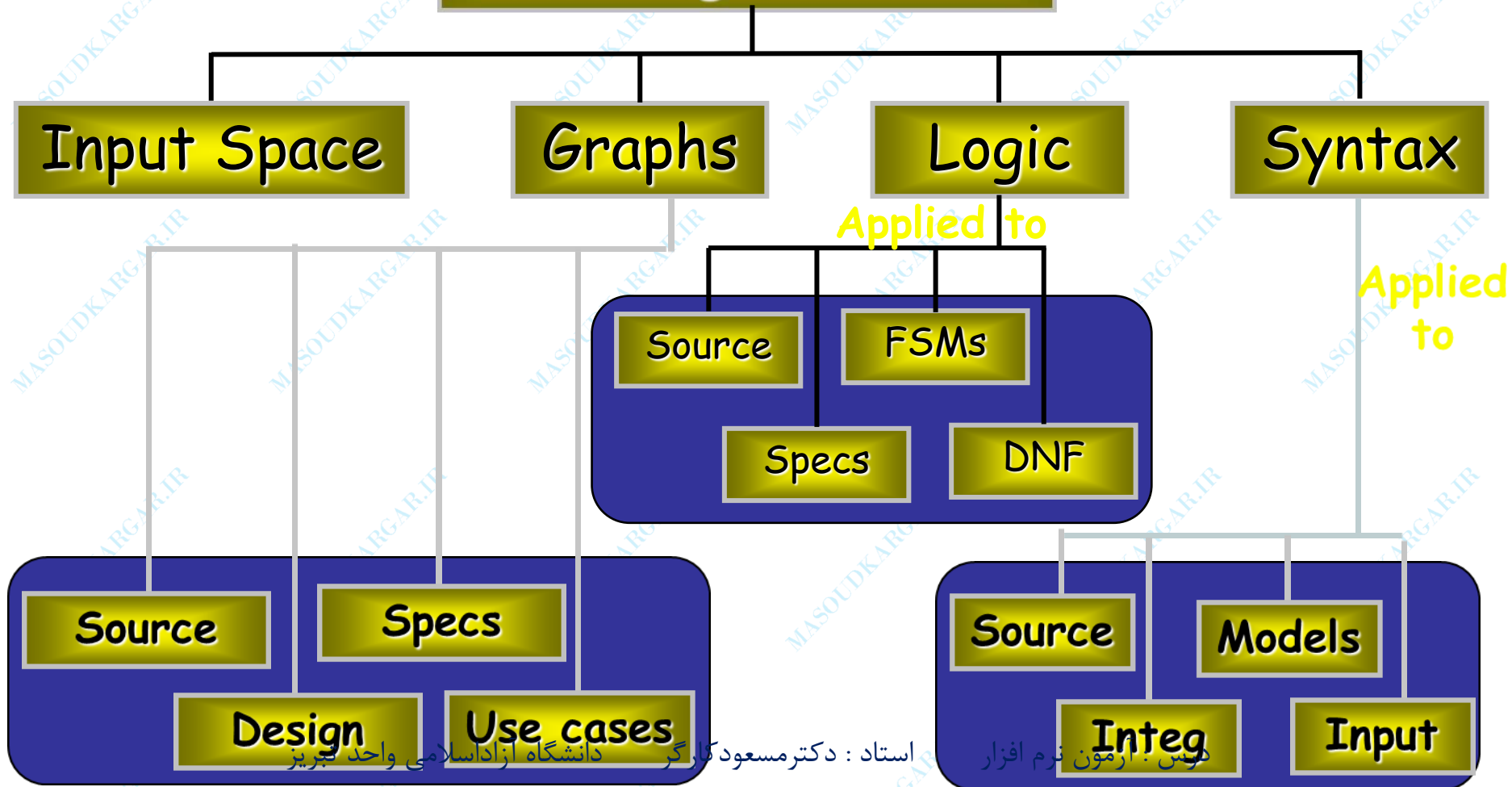
Save money

Find more faults

Build better software

Structures for Criteria-Based Testing

Four Structures for Modeling Software

- **Input Space**
- **Graphs**
- **Logic**
- **Syntax**

Applied to

Applied to

Logic:
- Source
- FSMs
- Specs
- DNF

Input Space / Graphs:
- Source
- Specs
- Design
- Use cases

Syntax:
- Source
- Models
- Integ
- Input

استاد : دکترمسعودکارگر   دانشگاه ازاداسلامی واحد تبریز   درس : آزمون نرم افزار

Summary of Part 1's New Ideas

1. Why do we test – to reduce the risk of using software

  _Faults, failures, the RIPR model

  _Test process maturity levels – level 4 is a mental discipline that improves the quality of the software

2. Model-Driven Test Design

  _Four types of test activities – test design, automation, execution and evaluation

3. Test Automation

  _Testability, observability and controllability, test automation frameworks

4. Test Driven Development

5. Criteria-based test design

  _Four structures – test requirements and criteria

Earlier and better testing empowers test managers