Software Quality

&

Testing

Steve Hostettler

Software Modeling and Verification Group

University of Geneva

inspired by the ISTQB foundation syllabus 2011

Test Levels

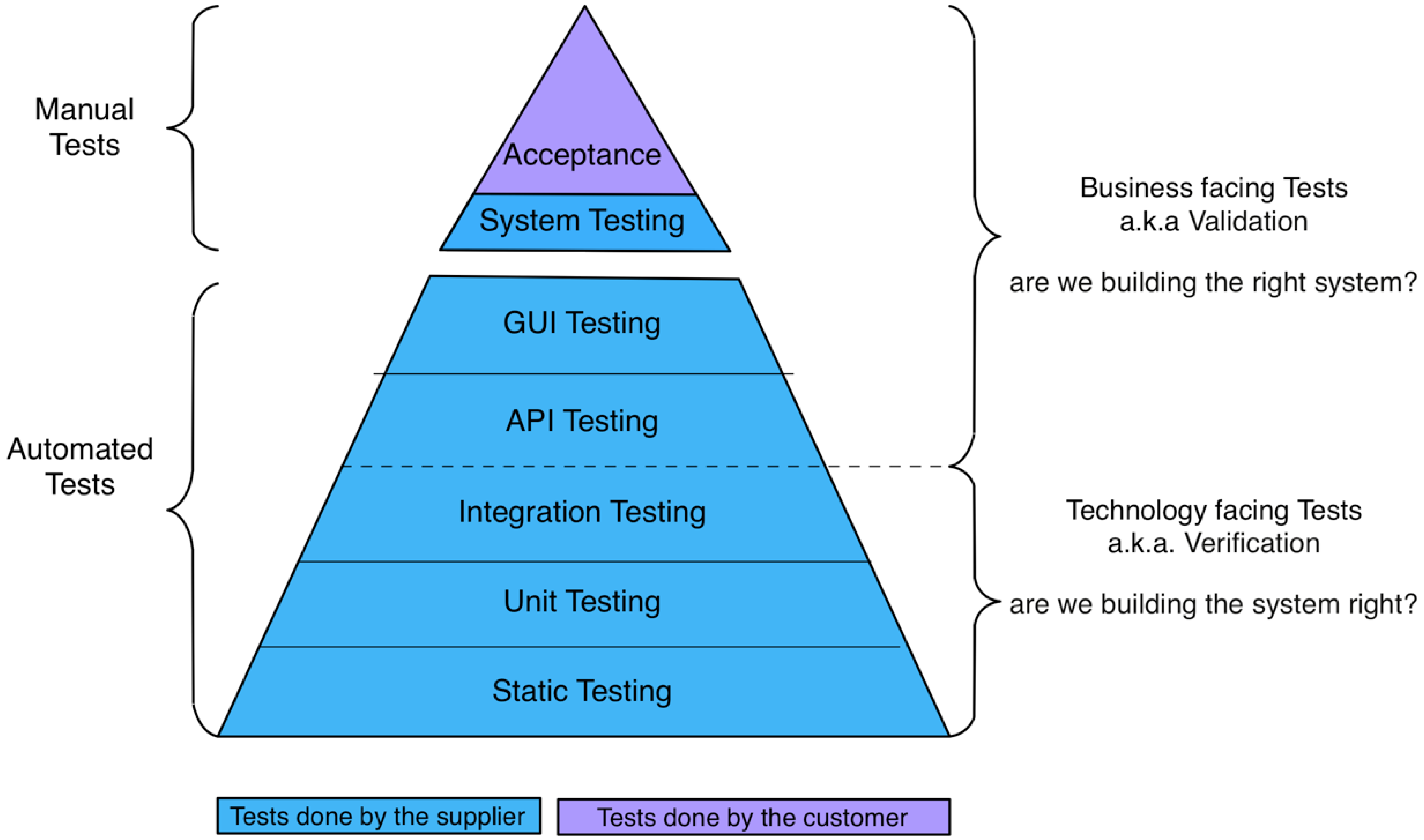
Components Testing

Integration Testing

System Testing

Acceptance Testing

Test Levels


Manual Tests
Automated Tests
Acceptance
System Testing
GUI Testing
API Testing
Integration Testing
Unit Testing
Static Testing
Business facing Tests
a.k.a Validation
are we building the right system?
Technology facing Tests
a.k.a. Verification
are we building the system right?
Tests done by the supplier
Tests done by the customer

Component Testing

a.k.a. Unit Testing, Module Testing

Test basis: Component requirements, detailed design, code

Test objects: Components (classes, package, programs, database scripts, ...)

Component Testing

In isolation

Uses mock, stubs, fakes

Tests functional and non-functional requirements

By programmers (low management overhead)

Integration Testing

Test basis: architecture and design, use cases, workflows

Test objects: Set of Components, subsystems, infrastructure, System configuration, integration to external components

Integration Testing

Some mock may remain

After component testing

Integrating to much thing may impair traceability

System Testing

Test basis: System requirements, Use cases, risk analysis

Test objects: System as a whole, documentation, help, configuration

System Testing

No more mocks
After integration testing

The test environment should as close as possible to the production environment

Usually done by a different team (no developer)

User acceptance Testing

Test basis: user requirements, system requirements, use cases, business processes, risk analysis

Test objects: user procedures, forms, reports, configuration

User acceptance Testing

No more mocks
After system testing

goal: establish confidence to make an informed decision about go/no go

Done by the end-user

Test types

Functional	Structure/Architecture
Non-Functional	Change-related

Test techniques

White box

Black box

Test techniques

Static

- Review/Audit
- Static analysis

Dynamic

- White box
- Black box
- Code coverage

Static Testing

review/auditing

Catch difficult and hidden bugs
but is time-consuming

Requires experienced developers

Static Testing

static analysis

Automatic inspection of the code

No execution / Symbolic execution

Catch deviation from standards and well-known bugs, e.g.
if (t = true) instead of if (t == true)

Dynamic Testing

white box

Building test cases knowing the structure of the software (statements, decisions, exceptions)

100% decision coverage => 100% statement coverage

Dynamic Testing

black box

Building test cases knowing the specification

Equivalence partitioning

Boundary values

Decision tables

Dynamic Testing

code coverage

Compute the % of code exercised by the tests on different basis:

Structure (class, package)

Statements

Branch