Software Quality Engineering

01. Introduction — Part A

Achiya Elyasaf

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

הנדסת מערכות תוכנה ומידע
Software and Information Systems Engineering

לימודים בזמן מלחמה

הסמסטר הזה אינו ככל הסמסטרים. אנחנו כולנו במצב של אי וודאות לאומית ואקדמית. חלקכם מגויסים למילואים. לחלקכם בני/בנות זוג במילואים. חלקכם מפונים ורובכם, כך אני מניח, מכירים פצועים, חללים או חטופים, במעגל ראשון או שני. המצב מאתגר ואנחנו (כחברי סגל, מורים, בני אדם) ננסה לעשות את ההתאמות הנדרשות על מנת להקל ולו במעט על הסמסטר. באתר מופיע כעת הסילבוס המעודכן ובו מספר הבהרות. יתכנו התאמות נוספות בהתאם לנסיבות ולהרכב הנרשמים (מגויסים, מפונים, וכו').

בינתיים, תוכלו לפנות בכל בעיה ושאלה ואנסה לענות במידת האפשר.

לימודים בזמן מלחמה

איך אתם יכולים לעזור?

תמנו אחראי:

- סיכומים
- הודעות ועדכונים בין המרצה למילואימניקים
- הקלטות - לבדוק בתחילת השיעור שהשיעור המשודר בזום / בהיברידי מוקלט היטב, ושרואים מה שצריך (לוח/מצגת/מרצה).

שיהיו ימים שקטים.

Today’s Agenda

- Course administrations
- Course agenda
- What is software quality?
- Motivation — Research
- Motivation — The software crisis
- Basic Terminology

Course administrations

Lecturer: Dr. Achiya Elyasaf ד"ר אחיה אליסף

Building 224, room 96

achiya@bgu.ac.il

Office hours: Mondays at 12:00

TA:

- Bruno Machado
- Keren Gorelik

Contact **me** regarding:

- Any problem with the course/staff
- Incompatibility between the TAs and me
- Problems in assignments
- etc.

Course agenda — may change

Week	Lecture	Practical Session	Assignment
1	Introduction 1		
2	Introduction 2	Unit testing — intro	Unit testing (2 weeks, 5%)
3	Unit testing	Unit testing — isolation	
4	Control flow & Symbolic execution	Control flow	Control flow (1 week, 5%)
5	Integration testing	Symbolic execution	Symbolic execution (1 week, 5%)
6	Functional testing	Selenium in JUnit	Functional testing (1 week, 5%)
7	Model-based testing	Cucumber	Model-based testing (3 weeks, 10%)
8	System tests	Provengo	
9	Data flow	Data flow	
10	Domain testing	Domain testing	
11	AI-Based SQE	Rehearsal	

Recommended References

- Software Testing and Quality Assurance /Naik, Tripathy
- Test Driven Development / Kent Beck
- Metrics and Models in Software Quality Engineering/ Kan

What is Software Quality?

How to measure quality?

- Cars: quality = brands (Mercedes vs. Peugeot)
- Home theater: quality = number of features

⇒ Quality = expensive & complex products

?

How to measure quality?

- Cars: quality = brands (Mercedes vs. Peugeot)
- Home theater: quality = number of features

⇒ Quality = expensive & complex products

- Can we measure quality using testing? Yes, but not enough…

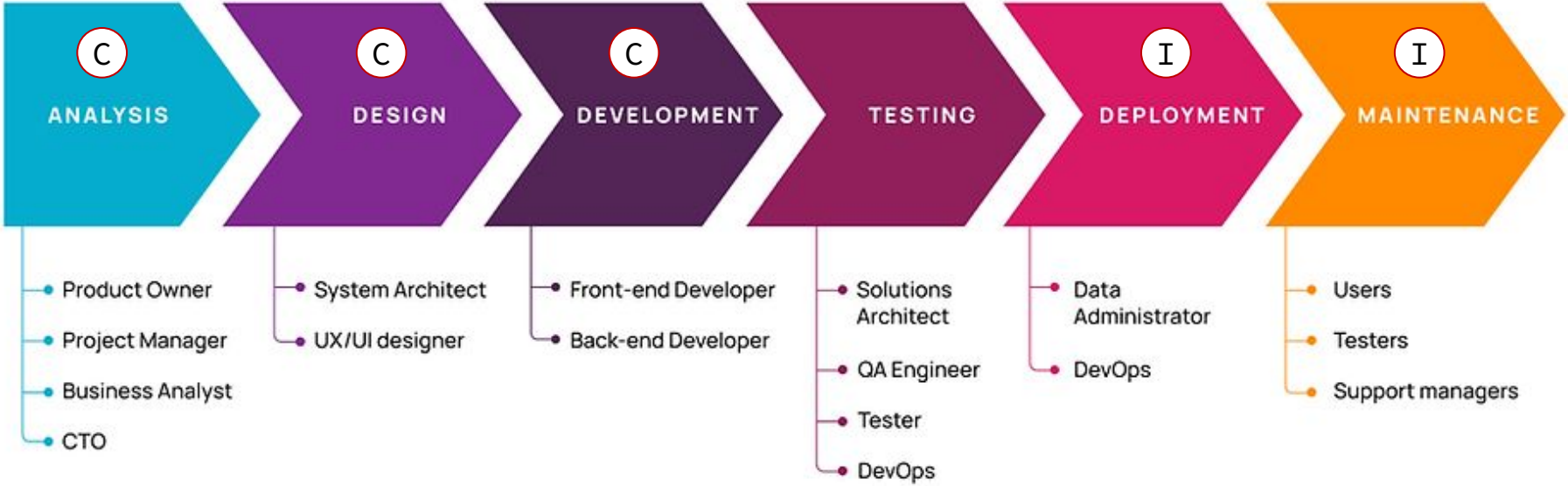
Testing $\subsetneq$ Software Quality

6 Phases of the Software Development Life Cycle

For each phase, we need methodologies for:

- Producing quality outcomes
- Measuring the resulted outcomes

6 Phases of the Software Development Life Cycle



For each phase, we need methodologies for:

- Producing quality outcomes
- Measuring the resulted outcomes

C Covered in other courses/degrees

I Covered in industry

6 Phases of the Software Development Life Cycle

Development methodologies and practices: C

- Agile, waterfall, etc.
- Design patterns
- *Code review*
- Pair programming
- Version control system (git)
- *Test-driven development*
- *Code synthesis*
- *AI-aided programming*
- …

Testing/verification methodologies and practices:

- *Testing phases.*
- *Coverage criteria*
- *Code analysis*
- *AI-based testing*
- *Model-based testing*
- *Formal methods*

Course topics in *italics*

Key Players in Maintaining Quality

- Quality Assurance (QA)
  - **Create and enforce development standards**
  - These standards aim at improving the dev. process

    … and to prevent bugs
- Quality Tester (QT)
  - **Define tests** scripts/specifications and run them
  - **Find bugs**
  - Includes the developers as well (shift left, unit testing)

Short motivation

- The software crisis
- Money
- Research (open questions in software quality)

Software Quality Assurance

Why do we need it?

The software crisis

11 of the most costly software errors in history · Raygun Blog

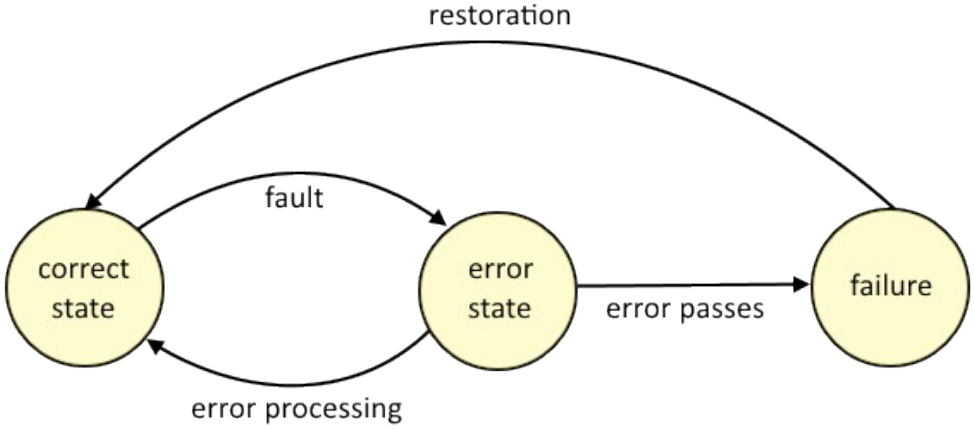
https://raygun.com/blog/costly-software-errors-history/

Some definitions

- Failure: when the external behavior of a system does not match the system specification, a failure occurs

- Error: a state of the system. When the system is in error state and no action is taken to correct it, it will lead to a failure

- Fault/defect: the cause of an error



Fault is the path to the dark side.

Fault leads to error. Error leads to Failure. Failure leads to suffering."

Mark II Computer (1947)

\ Site
  \ Harvard University

\ Symptom (Failure)
  \ The computer suddenly stopped working

\ Bug (Fault)
  \ A moth stuck between a set of relay contacts

\ Remedy
  \ Remove the moth and clean the contacts

\ Cost of Failure
  \ Undocumented

Patriot Missile Defense System (1991)

- Symptom (Failure)
  - The rocket failed to intercept an Iraqi Scud missile.
- Bug (Fault)
  - Inaccurate calculation of the time since boot, due to computer arithmetic errors.
- Cost of failure
  - 28 American soldiers died.
- Cause? (specification, design, code, or testing)

Disney's Lion King Game (1994-1995)

\ Symptom (Failure)

  \ The software did not work on many customers' computers

\ Bug (Fault)

  \ The software was incompatible with the most common operating systems on the market

\ Remedy

  \ Replace the incompatible version with the compatible one

\ Cost of Failure

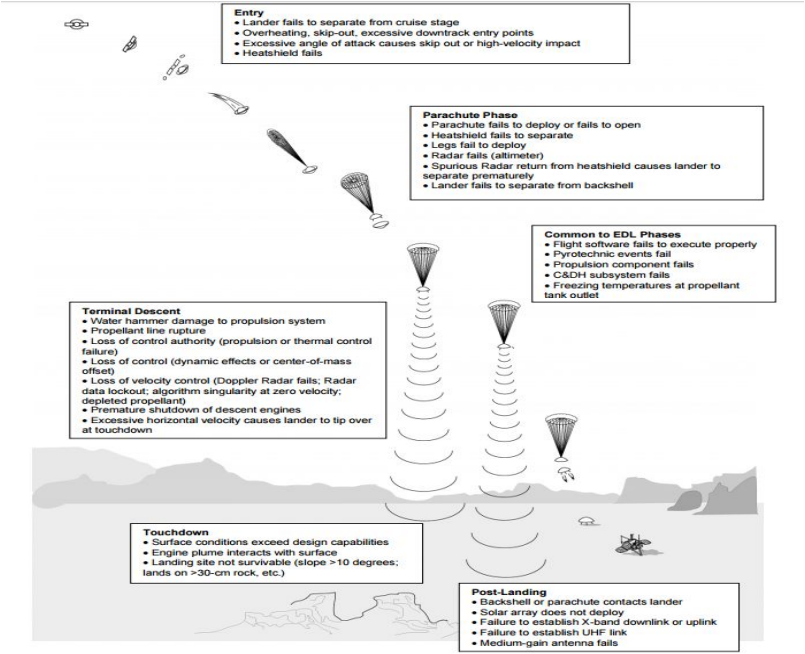
  \ Unsatisfied (mostly crying) customers

\ Cause? (specification, design, code, or testing)

Ariane-5 (1996)

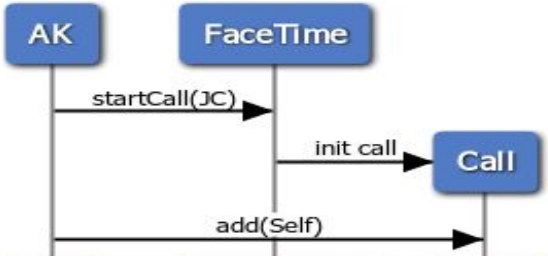
- Symptom (Failure)
  - Self-destructed 40 sec. after takeoff, because the angle of attack exceeded 20 degrees
- Bug (Fault)
  - Erroneous data transmitted by the inertial reference system
  - The data was meaningful only before lift-off
- Forensic Conclusion
  - Careless reuse of the Ariane-4 software in the Ariane-5 project
- Cost of Failure
  - Over $500 million
- Cause? (specification, design, code, or testing)

NASA Mars Polar Lander (1995)



- Symptom (Failure)
  - Probe crashed on Mars when landing
- Bug (Fault)
  - Engine turned off about 1,800 meters from surface due to unexpected setting of a single data bit. WHY?
  - Triggered by the mechanical vibration of the probe's three
- Forensic Conclusion
  - The legs fold-down procedure and the landing process were tested by different teams
- Cost of Failure
  - $165 million
- Cause? (specification, design, code, or testing)

Appel's FaceTime Bug (2019)



- Symptom (Failure)
  - FaceTime can be used for eavesdropping
- Bug (Fault)
  - Starting a FaceTime Video call and whilst the call is dialling, add your own phone number to the call.
  - A group FaceTime call including yourself and the audio of the person you originally called is started, even if they haven't accepted the call yet.
- Forensic Conclusion
  - ?
- Cost of Failure
  - ?

Detected by Grant Thompson, a 14-year-old in Arizona

The Y2K Bug 00

\ Symptoms

  \ All 2000 dates shown as 1900 dates

  \ Several senior citizens invited to Kindergarten around 1999 ☺️

\ Bug (Fault)

  \ Year stored in 2-digit format

\ Remedy

  \ Update / replace legacy programs

\ Cost of Fixing the Bug

  \ $US 300 billion

\ Cost of Failure

  \ No significant level of computer failure took place when the clocks rolled over into 2000

\ Cause? (specification, design, code, or testing)

Therapy Planning Software (2000) (National Cancer Institute, Panama City)

\ Symptom (Failure)

- \ In a series of accidents, therapy planning software miscalculated the proper dosage of radiation for patients undergoing radiation therapy

\ Bug (Fault)

- \ The software was designed to allow technicians to draw on a computer screen the placement of four metal shields called "blocks" designed to protect healthy tissue from the radiation
- \ The doctors discovered that they could trick the software by drawing all five blocks as a single large block with a hole in the middle
- \ The software recommended twice the necessary exposure when a hole was drawn in certain direction

\ Cost of Failure

- \ At least eight patients died, while another 20 receive overdoses likely to cause significant health problems

\ Court's Decision

- \ The physicians, who were legally required to double-check the computer's calculations by hand, were indicted for second-degree murder

\ Cause? (specification, design, code, testing, or maintenance)

Daylight False Alarm USA, October 2007

\ Symptom (Failure)

- \ Some PCs, cellphones, PDAs, and electronic clocks switched back to the “winter time” on Sunday, October 28, one week before the official date

\ Bug (Fault)

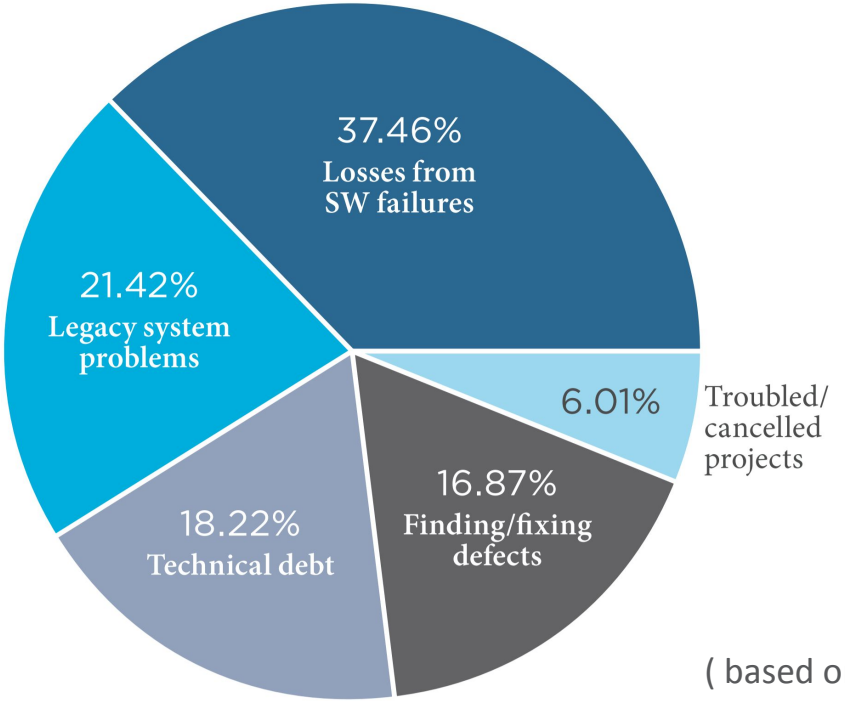
- \ Daylight saving used to end on the last weekend in October
- \ The US Congress changed it to the first weekend in November in an effort to save power
- \ Various devices were still programmed to the old date

\ Cost of Failure

- \ Thousands of people calling information lines just to ask what time it is
- \ High-tech parking meters in Baltimore expired early
- \ Some people missed their morning meetings on Sunday

\ Cause? (specification, design, code, testing, or maintenance)

The cost of poor-quality software in the US in 2018 is approximately $



(based on CISQ: Consortium for IT Software Quality)

Why is it so hard?

The testing oxymoron

Software bug: an unanticipated behavior.

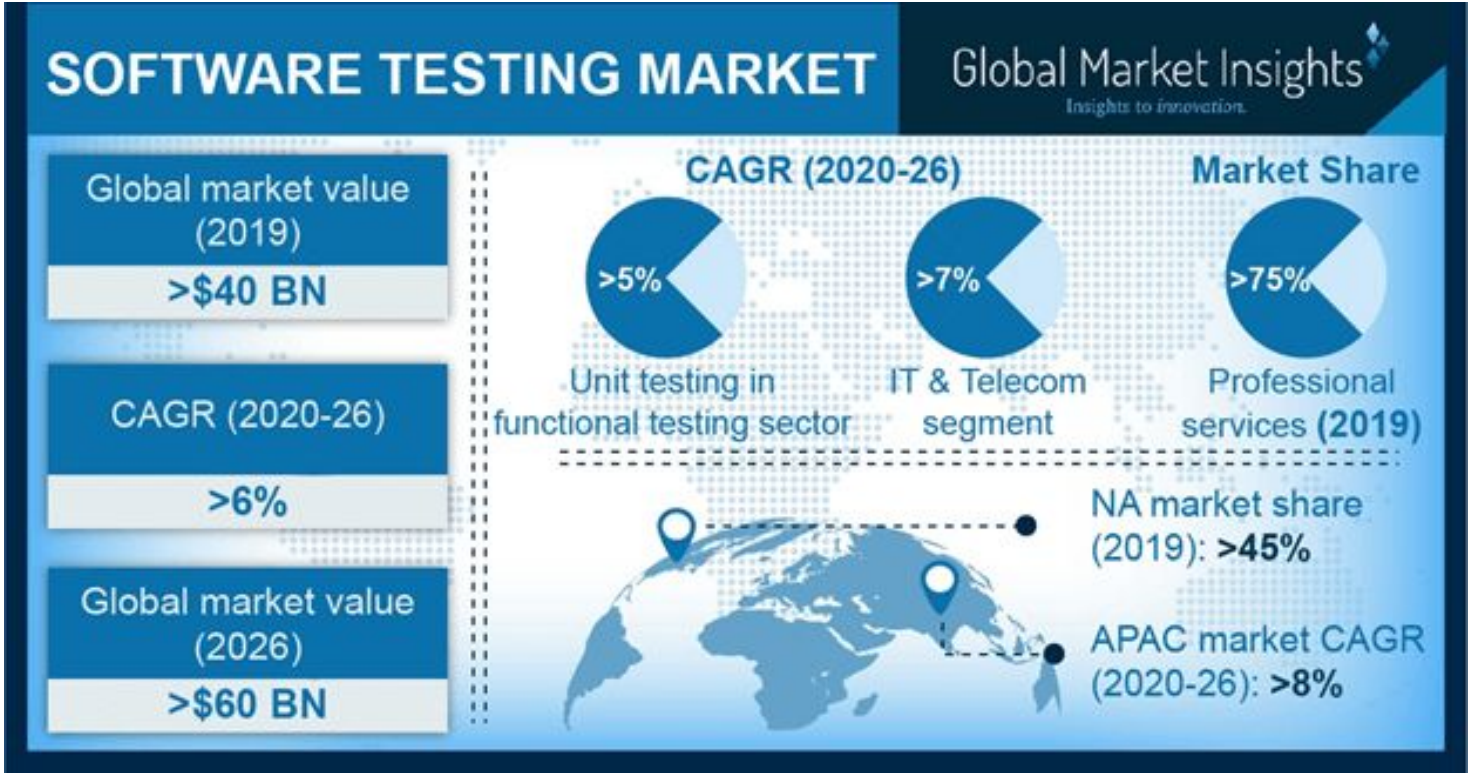
Test: a behavior that we suspect that may be problematic

Fault is the path to the dark side.

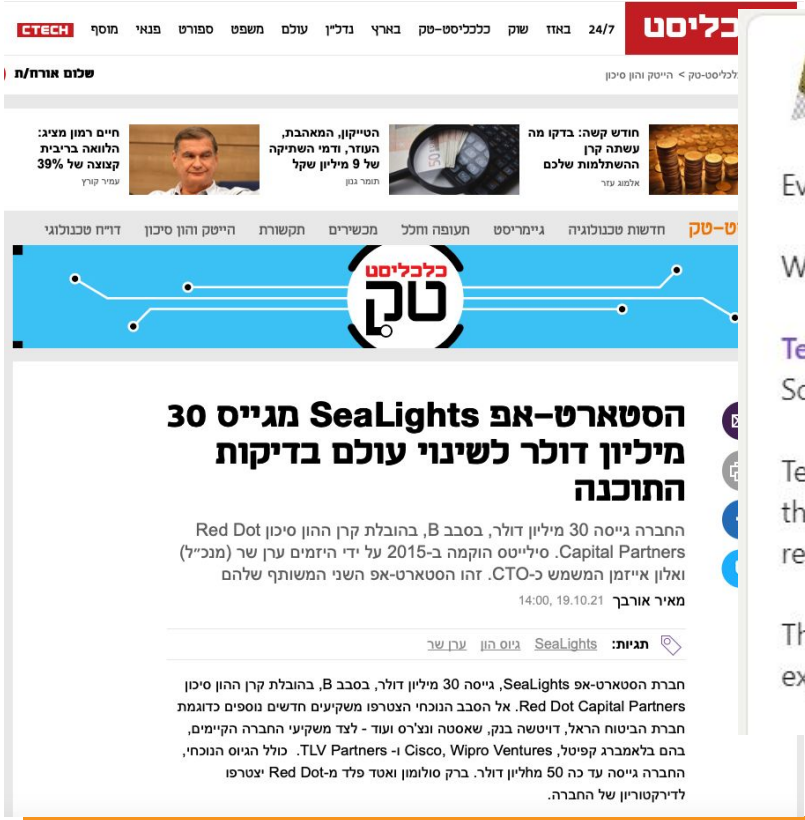
Fault leads to error. Error leads to Failure. Failure leads to suffering."

As a Consequence…..

Software Testing Market (December 2019)



Software Testing Startups



CTECH מוסף פנאי ספורט משפט עולם נדל"ן בארץ כלכליסט-טק שוק באזז 24/7 כלכליסט

שלום אורח/ת

כלכליסט-טק > הייטק והון סיכון

חיים רמון מציג: הלוואה בריבית קצוצה של 39%
עמיר קורץ

הטייקון, המאהבת, העוזר, ודמי השתיקה של 9 מיליון שקל
תומר גנון

חודש קשה: בדקו מה עשתה קרן ההשתלמות שלכם
אלמוג עזר

דו"ח טכנולוגי | הייטק והון סיכון | תקשורת | מכשירים | תעופה וחלל | גיימריסט | חדשות טכנולוגיה | כלכליסט-טק

כלכליסט טק

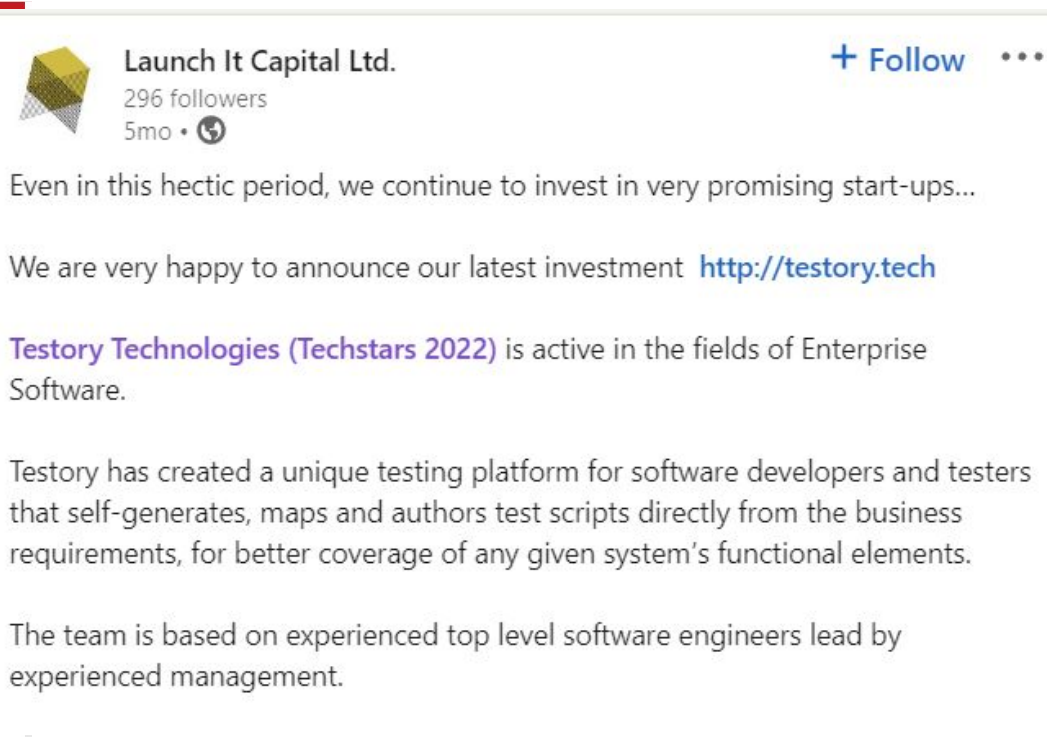
הסטארט-אפ SeaLights מגייס 30 מיליון דולר לשינוי עולם בדיקות התוכנה

החברה גייסה 30 מיליון דולר, בסבב B, בהובלת קרן ההון סיכון Red Dot Capital Partners. סילייטס הוקמה ב-2015 על ידי היזמים ערן שר (מנכ"ל) ואלון אייזמן המשמש כ-CTO. זהו הסטארט-אפ השני המשותף שלהם

מאיר אורבך 14:00, 19.10.21

תגיות: SeaLights | גיוס הון | ערן שר

חברת הסטארט-אפ SeaLights, גייסה 30 מיליון דולר, בסבב B, בהובלת קרן ההון סיכון Red Dot Capital Partners. אל הסבב הנוכחי הצטרפו משקיעים חדשים נוספים כדוגמת חברת הביטוח הראל, דויטשה בנק, שאסטה ונצ'רס ועוד - לצד משקיעי החברה הקיימים, בהם בלאמברג קפיטל, Cisco, Wipro Ventures ו- TLV Partners. כולל הגיוס הנוכחי, החברה גייסה עד כה 50 מליון דולר. ברק סולומון ואטד פלד מ-Red Dot יצטרפו לדירקטוריון של החברה.



Launch It Capital Ltd.
296 followers
5mo •

+ Follow

Even in this hectic period, we continue to invest in very promising start-ups...

We are very happy to announce our latest investment http://testory.tech

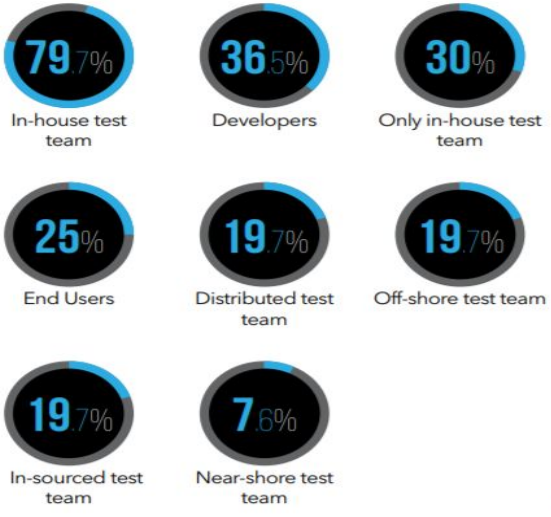
Testory Technologies (Techstars 2022) is active in the fields of Enterprise Software.

Testory has created a unique testing platform for software developers and testers that self-generates, maps and authors test scripts directly from the business requirements, for better coverage of any given system's functional elements.

The team is based on experienced top level software engineers lead by experienced management.

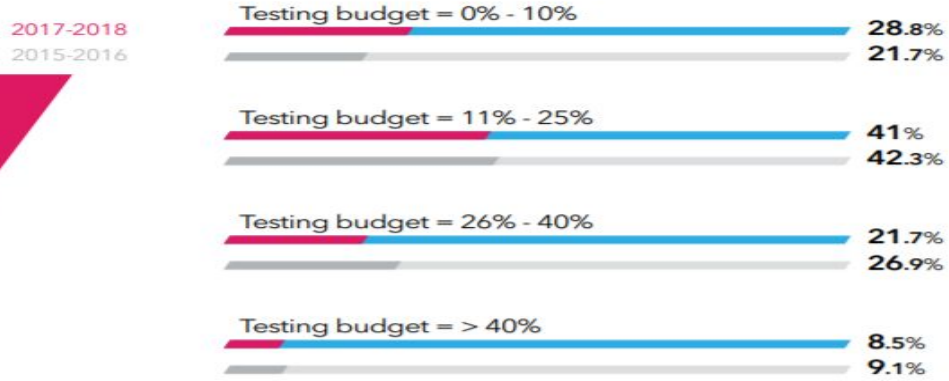
WHO IS RESPONSIBLE FOR SOFTWARE TESTING IN YOUR COMPANY?

* Selecting multiple choices were available



Source: ISTQB

WHAT PERCENT OF A TYPICAL IT/ R&D PROJECT BUDGET IS ALLOCATED TO SOFTWARE TESTING?



Source: ISTQB

WHAT IS YOUR EXPECTATION FOR YOUR ORGANIZATION'S SOFTWARE TESTING BUDGET IN THE NEXT 12 MONTHS?

Source: ISTQB

WHAT ARE THE MAIN OBJECTIVES OF YOUR TESTING ACTIVITIES?

As in the previous report, the main objective of respondents testing activities is "To detect bugs (88.1%). Next top three most popular answers are "To show the system is working properly" (68.6%), "To gain confidence" (55%) and "To evaluate requirements" (51.2%).

* Selecting multiple choices were available

Source: ISTQB

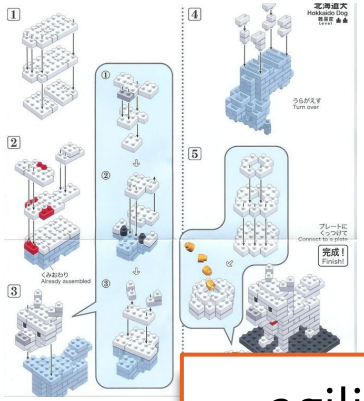
QA Research

SE & AI Lab

- **Funding:** ISTRC, IBM, Provengo
- **Collaborations:** Provengo, BIU, HUJI, Weizmann Institute, IBM
- **Research:**
  - Behavioral Programming, Evolutionary Algorithm, Model-Based Testing, Context Awareness, Explainable AI, Robotics, Automatic Code Generation, and more.
- **Team:** https://achiya.elyasaf.net/lab

Behavioral Programming (BP)

Specification



App. Agnostic Execution engine

Behavior

agility, alignment to requirements, anti scenarios

1. David Harel, Assaf Marron, Gera Weiss. "Behavioral Programming." Communications of the ACM 55.7 (2012): 90-100.

2. Achiya Elyasaf. "Context-Oriented Behavioral Programming." Information and Software Technology 133 (2021): 106504.

High-level Programming

Programming nowadays

Future programming

- 20% - what the system may/must/must-not do.
- 80% - optimizing the possibilities derived from the 20%.

Automatic Code Generation

Text
(requirements / docs / code)

Natural Language Processing

Code

OpenAI Codex

GitHub Copilot

DeepMind

AlphaCode

Code

Genetic Improvement

Better code

GISMOE

FINCH

EvoSuite

ELM (OpenAI 6/2022)

GenPROG

Execution traces

Genetic Generation

Code

GPBP

CBGP (GECCO 22): similar motivation, but addresses algorithmic problems — not full system evolution but small sub-components

Guiding & Guarding Learning

Tom Yaacov

Features augmentation:

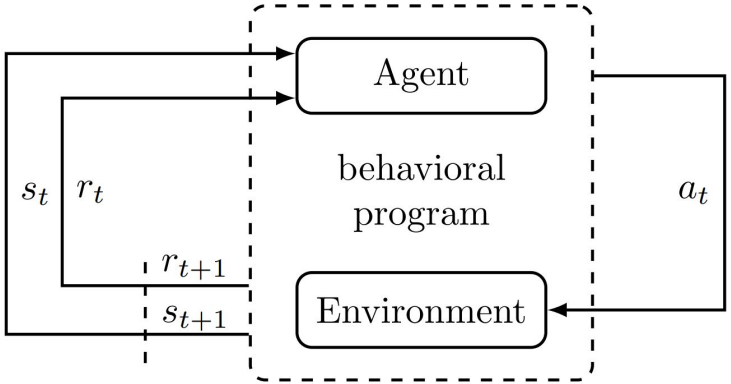
- Automatic labeling and grouping of sparse & non-representative data
- Addresses the curse of dimensionality

Reward-function augmentation:

- Agile adaptation
- Context awareness
- Robustness to edge cases

Override scenarios:

- Stateful, comprehensible, powerful and expressive
- Increase safety
- Correctness can be proved



Parametric modeling of human thinking

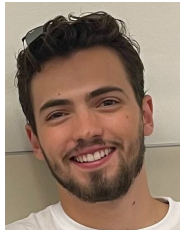
Can I explain human actions in domain's language?

> "player gave taking a free pawn in favor of taking a bishop in three moves"

Approach:

1. Define strategy b-threads (playbook)
2. Train a BP player using chess.com (features = strategies' statuses)
3. Explain human players' actions by tracing the strategies' values

86% accuracy

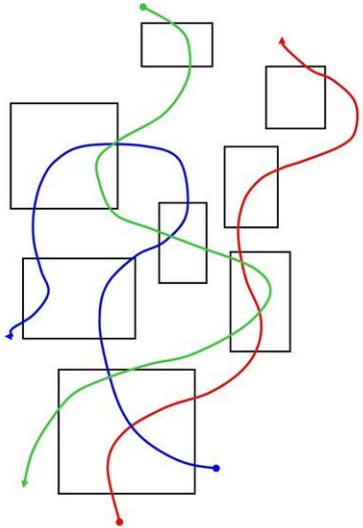


Benny Skidanov

Designless Programming

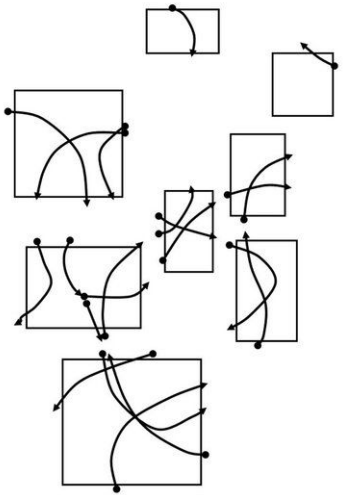
ChatGPT/Github Copilot

We cannot programm what must not happen



(a) Inter-object scenarios

Requirements



(b) Intra-object specifications

Programming (OOP)

Designless Programming

Ron Ziskind

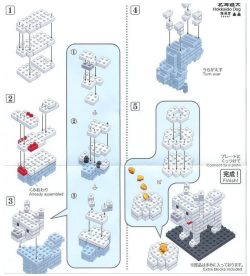
Eliad Shem-Tov

+

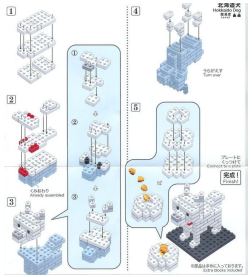
ChatGPT/LLM

=

BP Specification



BP Specification



+

BP Execution engine

=

Behavior

SE & AI 4 SQ (software quality)

Is my research good? Applicable?

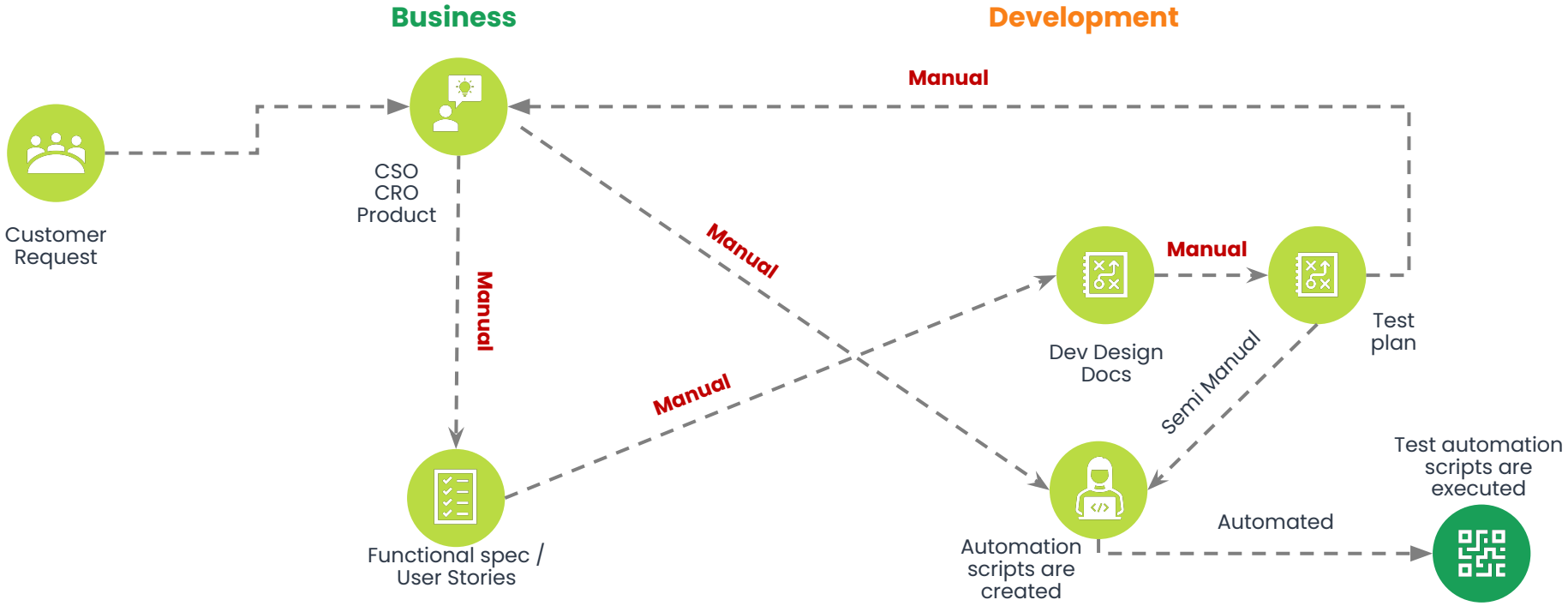
How to validate SE research?

SE & AI 4 SQ (software quality)

Deep-science solutions

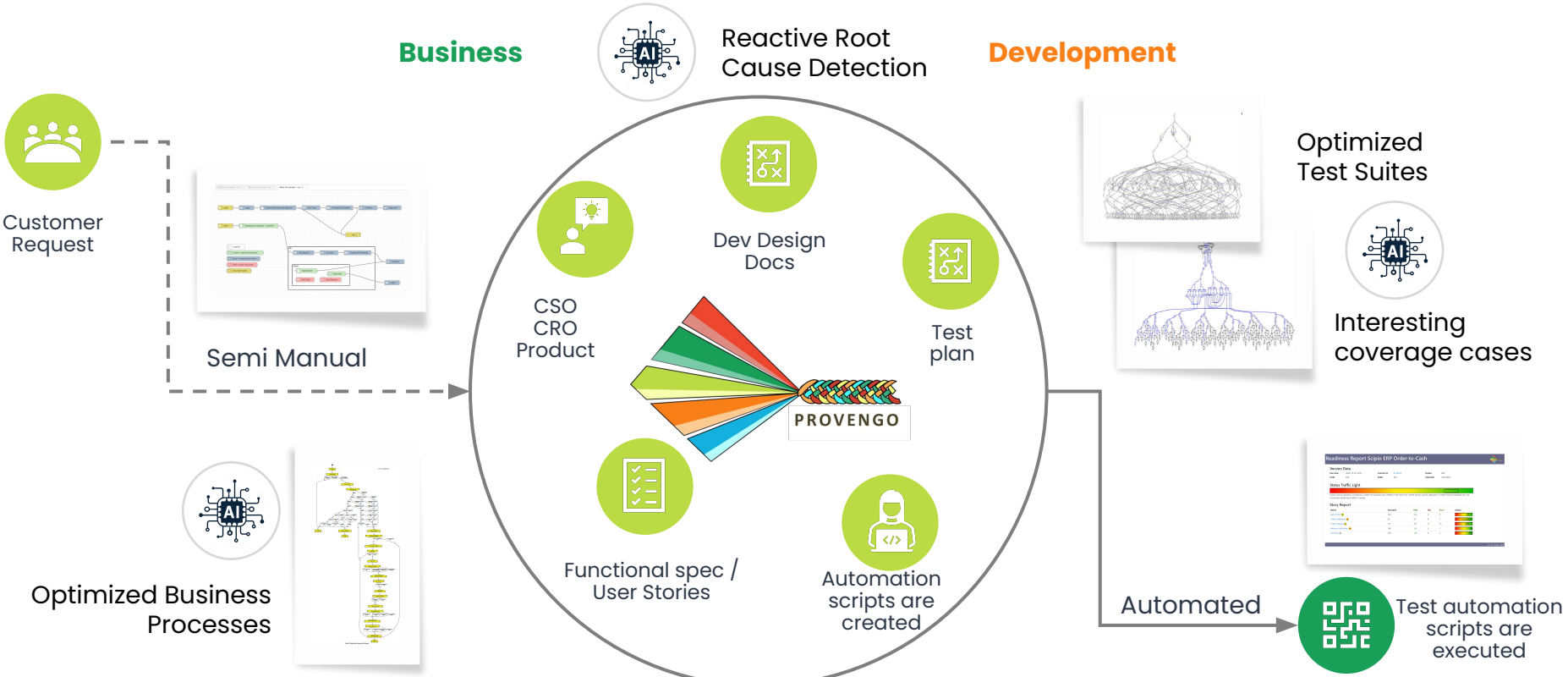
Money (scholarships), use cases, research validity, exposure, production-ready research tools

SE & AI 4 SQ (software quality)



Manual steps are costly and prone to errors

SE & AI 4 SQ (software quality)



What is Software Quality?

How to measure quality?

- Cars: quality = brands (Mercedes vs. Peugeot)
- Home theater: quality = number of features

=> Quality = expensive & complex products

?

How to measure quality?

As engineers, we must define quality in measurable ways:

- Conformance to requirements:
  - Requirements must be precisely defined
  - Any stray from the requirements = malfunction
- Fitness for use:
  - Take into account also user expectations
  - Different client & use cases = different expectations
  - Quality parameters: Design quality; How we met the requirements and expectations

How to measure quality?

- No single answer – means different things to different people
- Context dependent
  - Transcendental View: can be recognized but hard to define
  - User view: fitness for purpose. Satisfy user needs and expectations
  - Manufacturing view: meeting specifications
  - Product view: quality is tied to the product internal qualities
  - Value-based view: quality is related to the amount of people willing to pay

Transcendental View

- Can be recognized but hard to define
- Quality is something ideal, which is complex to be precisely defined
- A good-quality object stands out, and it is easily noticeable!
- Due to its philosophical nature, no effort is made to express it using concrete measures

User view

- The extent to which a product meets user needs and expectations
- This view is highly personalized in nature
- Due to its personalized nature, a product is considered good if it satisfies a large number of customers
- As quality engineers, it is useful to identify what product attributes users consider to be important
- Examples of *subjective* elements are *usability*, *reliability* and *efficiency*

Manufacturing view

\ Quality is seen as meeting all requirements

\ The concept of **process** plays a key role in this view

\ An efficient process works "right the first time" so that development and maintenance costs are reduced

\ When the process is not well defined, bad-quality products are manufactured in a consistent manner

Product view

\ Approach: if a product is manufactured with good internal properties, then it will have good external qualities

\ Example: high degree of modularity, which is an internal property, makes a software testable and maintainable

\ The current quality level of a product indicates the presence or absence of measurable product properties

\ And so, the quality can be assessed in an objective manner

Value-based view

\ how much a customer is willing to pay for a certain level of quality?

\ In practice, quality is meaningless if it does not make cost effective

\ So this view represents a trade-off between cost and quality

Verification and validation

Verification

- determines whether the product (spec, code, manual) satisfies the requirements
- confirms that one is building the *product correctly*
- includes static analysis techniques, such as inspection, walkthrough and reviews

Validation

- checks that a product meets its intended use
- confirms that one is building the *correct product*
- includes running the system in its real environment and using a variety of tests

Test levels (1)

- Four testing levels:
  - Unit
  - Integration
  - System (functional & non-functional)
  - Regression
  - Acceptance — customer

  Unit, Integration, System, Regression: manufacturer (or developer)
- First three are performed by the manufacturer (or developer) functions
- Acceptance is performed by the customer

Test levels (2)

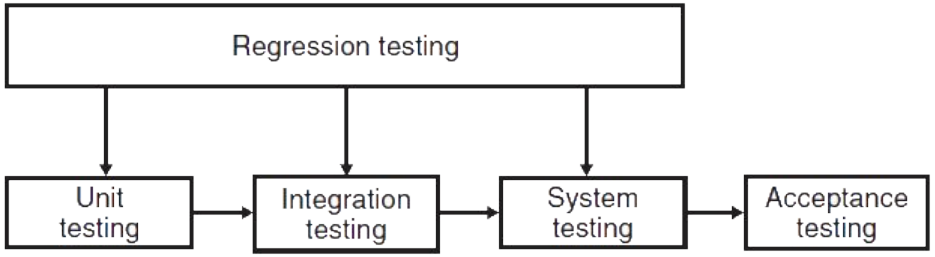
- Unit
  - Performed by the programmers
  - Tests individual software units like functions and classes in isolation
- Integration
  - Performed by programmers and test engineers
  - Tests aggregated modules that have been unit-tested
  - It verifies the interfaces between modules

Test levels (3)

- System
  - Performed by programmers and test engineers
  - Includes the following tests: functionality, security, robustness, load, stability, stress, performance and reliability
  - Includes:
    - Creating a test plan
    - Designing a test suite
    - Preparing test environment/s

Test levels (4)

- Regression
  - Ensure that nothing is broken after software modifications
  - Tests for new faults in the portion that was NOT modified
  - It is not a distinct level of testing but rather a sub-phase of unit, integration and system testing



Test levels (5)

- Acceptance
  - Performed by the customer after software delivery
  - The objective: measure the quality of the product (and not searching for defects)