ECE40097 Special Topic-Software Testing and Debugging

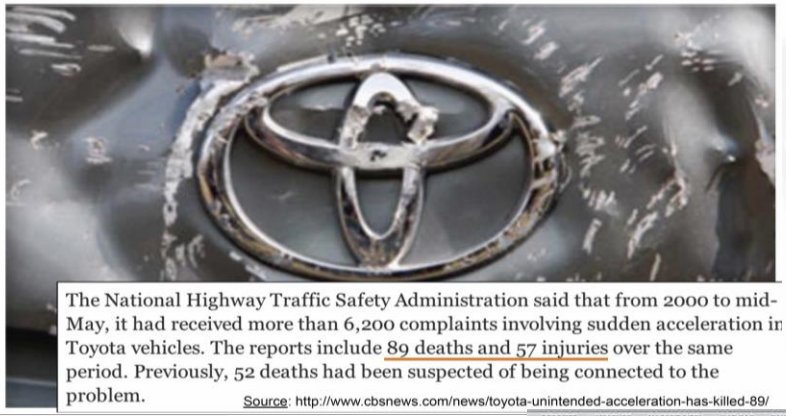
Why Software Testing Matters

Shin Hong

Poor SW Quality Threatens Society

By CBSNEWS / AP / May 25, 2010, 7:08 PM

Toyota "Unintended Acceleration" Has Killed 89

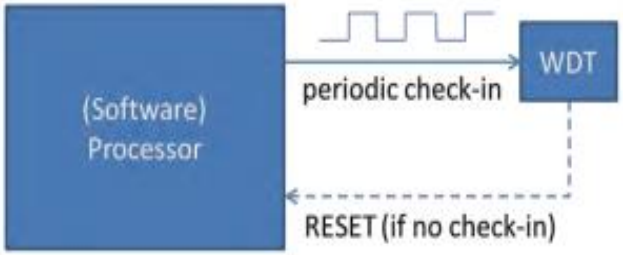


The National Highway Traffic Safety Administration said that from 2000 to mid-May, it had received more than 6,200 complaints involving sudden acceleration in Toyota vehicles. The reports include 89 deaths and 57 injuries over the same period. Previously, 52 deaths had been suspected of being connected to the problem.

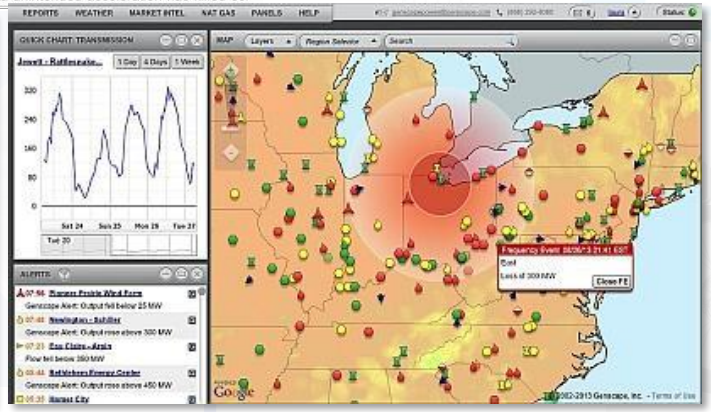
Source: http://www.cbsnews.com/news/toyota-unintended-acceleration-has-killed-89/

A "watchdog timer" is hardware to auto-reset software

- Healthy software should periodically "check-in" to prevent reset



With multiple tasks, health of all tasks must be checked



P. Koopman: A Case Study of Toyota Unintended Acceleration and Software Safety, 2014

M. Barr: Killer Apps-Embedded Software's Greatest Hit Jobs, 2014

Example of SQL injection

SQL Injection.

User-Id : srinivas

Password : mypassword

```
select * from Users where user_id= ' srinivas '
                      and password = ' mypassword '
```

User-Id : ` OR 1= 1; /*

Password : */--

```
select * from Users where user_id= '` OR 1 = 1; /* '
                      and password = ' */-- '
```

9lessons.blogspot.com

Damage of Software Failures in 2017

- The number of people influenced by software failures:
  3.7 Billions (c.f., the world population is 7.4 Billions in 2019)

- Estimated economic loss by software failure:
  1715 Billion USD (c.f., South Korea GDP is 1647 Billion in 2019)

- Amount of time required for resolving software failures:
  around 265 years world-wide

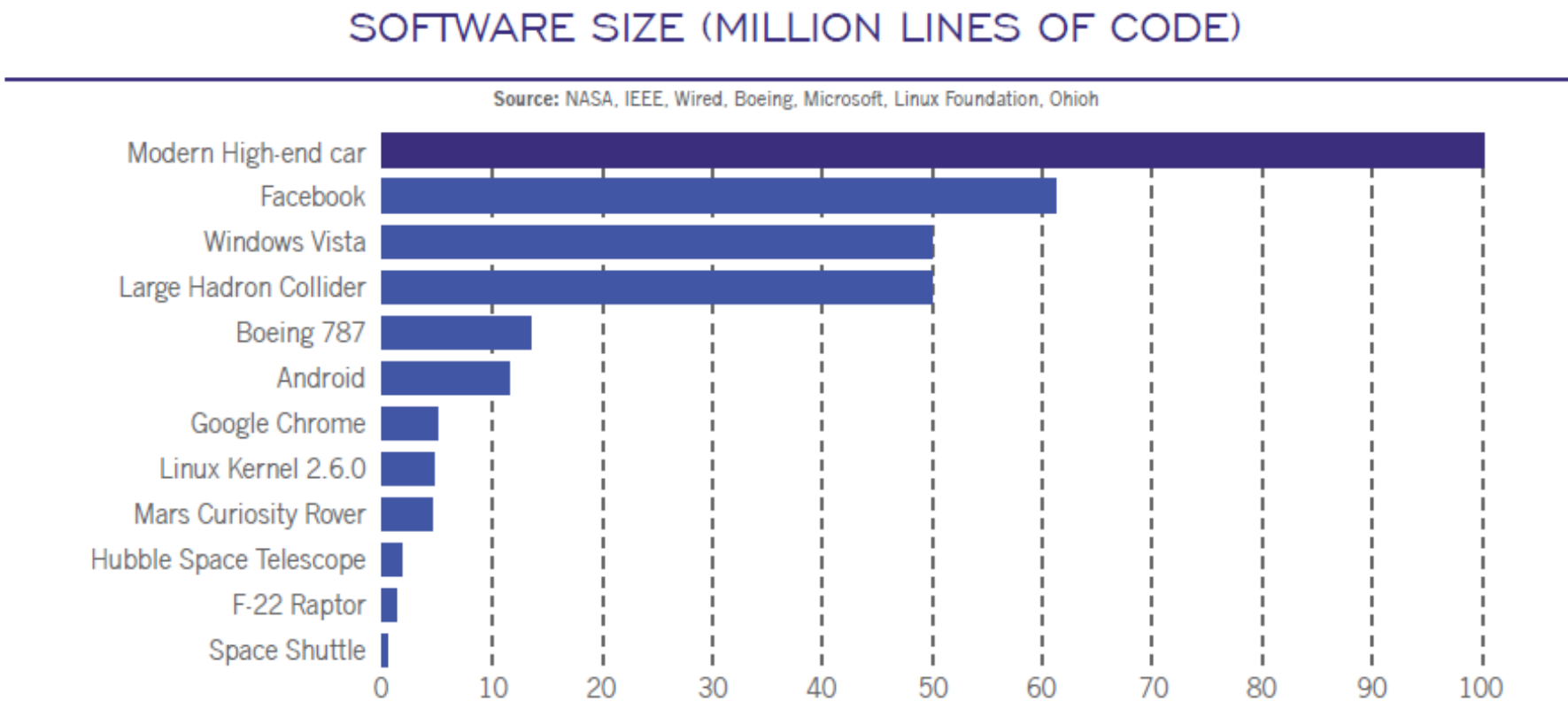
* Based on the data reported in *Software Fail Watch* (5th Ed.), Tricentis and *Allied Market Research*, 2019

Case: The Ariane 5 Explosion (1996)

```
L_M_BV_32 := TDB.T_ENTIER_32S ((1.0/C_M_LSB_BV) *
                                G_M_INFO_DERIVE(T_ALG.E_BV));
if L_M_BV_32 > 32767 then
   P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
   P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
   P_M_DERIVE(T_ALG.E_BV) := UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M
end if;

P_M_DERIVE(T_ALG.E_BH) := UC_16S_EN_16NS (TDB.T_ENTIER_16S
                                ((1.0/C_M_LSB_BH) *
                                G_M_INFO_DERIVE(T_ALG.E_BH)))
end LIRE_DERIVE;
```

https://www.viva64.com/en/b/0426/



http://blogs.blackberry.com/2016/12/ces-2017-holistic-security-for-the-software-defined-car/

Debugging Troubles SW Qualities

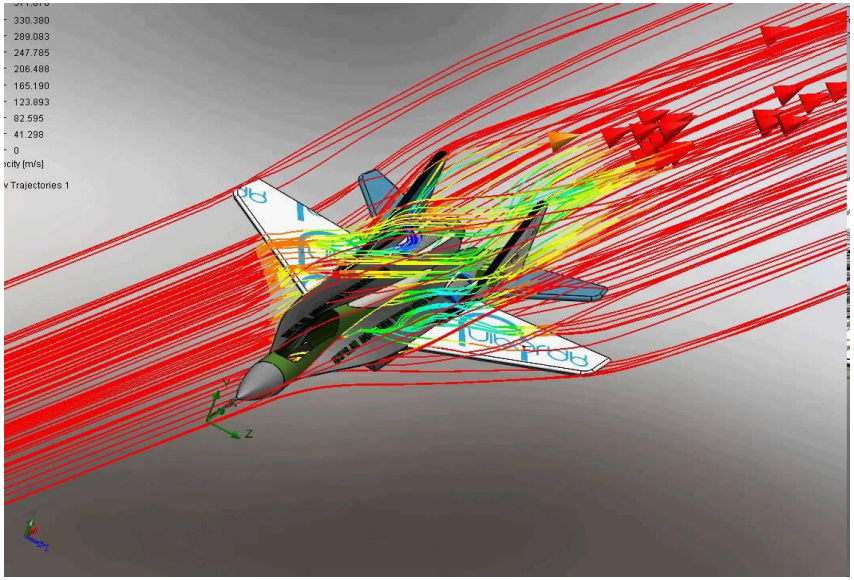
- Debugging (fixing bugs and making code work) consumes **50% of development time of developers** [Britton *et al.*, '13]
  - Developers spend 30% on their time coding
- Developers **run short of time for debugging** even for known issues
  - In one month after Firefox 3.5 released, 410000 crash reports on 750 different crashes were submitted [Kim *et al.*,, TSE'11]
- It take **long time for developers to resolve subtle bugs**
  - In Apache projects, it takes 140 to 500 days for 25% of bugs to be resolved after the bugs were first reported [Mockus *et al.*, TOSEM'02]

Why It's Difficult to Ensure SW Correctness

- Unique characteristics of software artifacts
  - **Complicateness**: no two parts are alike in SW
  - **Discontinuity**: behaviors are sensitive to subtle input conditions
  - **Determinism: failures are intended**
  - **No Intangible form**: no mechanical robustness
  - **Arbitral complexity:** SW confronts needs of human societies
  - **Mallability**: extremely difficult to revert faults and/or failures
- Inherent limitation of software engineering
  - **Undecidability** (a.k.a. the Halting problem)
  - **Lack of abstraction**: no model perfectly abstracts SW

Case: F-22 Raptor Autopilot (2007)

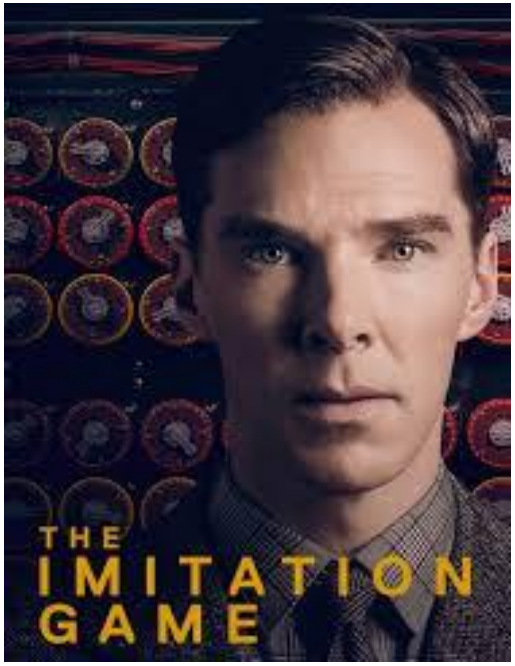
- F-22 autopilot system crached right after crossing International Date Line of Pactific due to SW bugs Feb 2007

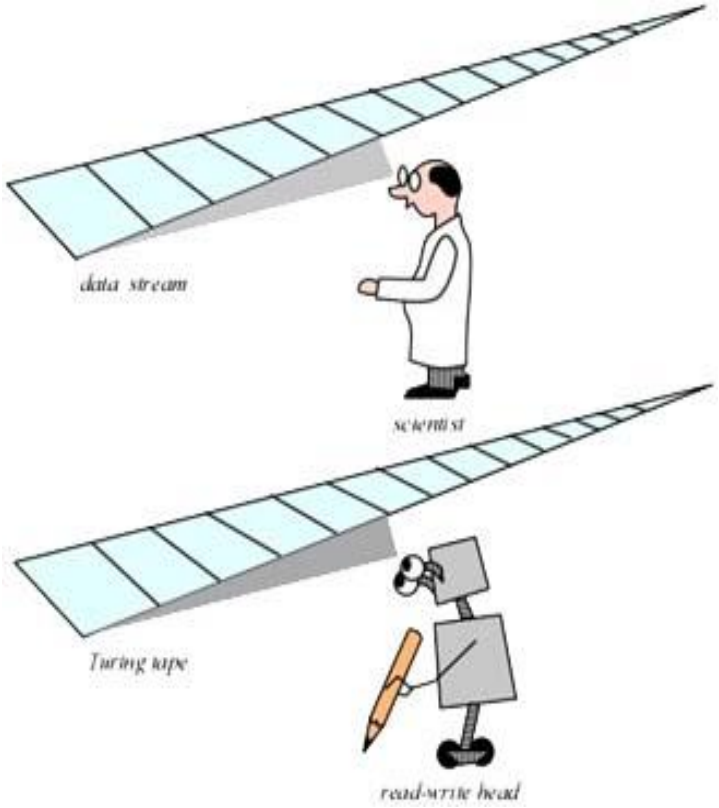


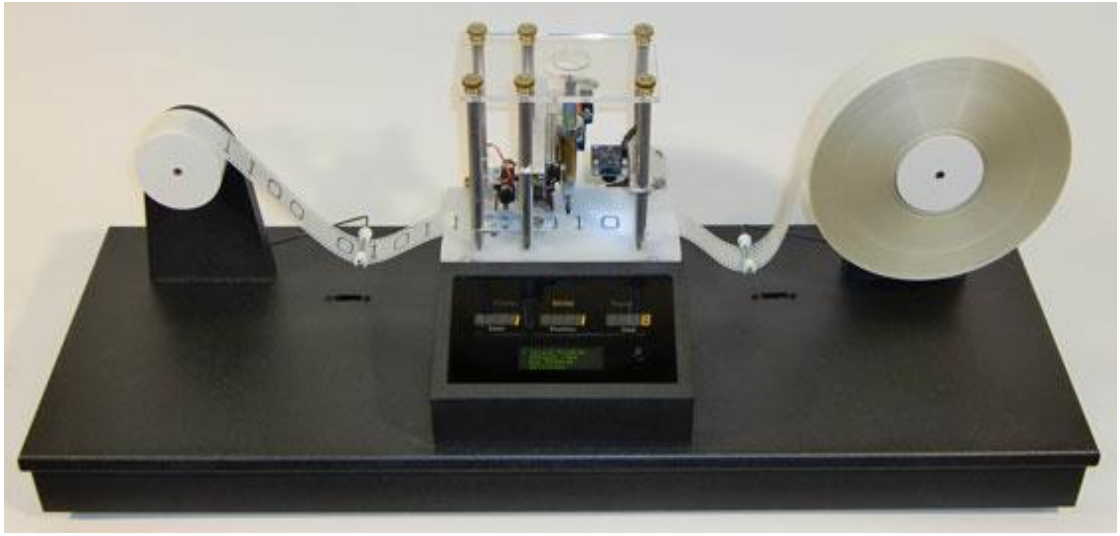
```
require_once('chorus/Utils.php');
require_once('chorus/Kestrel.php');
require_once('chorus/DataService.php');
require_once('chorus/Shard.php');
Database::set_defaults(
  array('user' => 'tumblr3', 'password' => 'm3MpH1C0Koh39AQD83TFhsBPlOM1R
        'database' => 'tumblr3', 'write_lock_tables' => '*',
        'extended_log' => (idate('G') == 17 && intval(idate('i')) == 56

if (__FILE__ == '/var/www/apps/tumblr/config/config.php' || __FILE__ == '/
  define('ENVIRONMENT', 'production');
  if (!defined('DEFAULT_DATABASE')) define('DEFAULT_DATABASE', 'primary');
  define('S3_BUCKET', 'data.tumblr.com');
  define('ENABLE_PANTHER', true);
  define('ENABLE_MEDIA_CDN', true);
  define('ASSETS_URL', (ENABLE_MEDIA_CDN && !(isset($_SERVER['HTTPS']) &&
  define('MEMCACHE_HOST', '10.252.0.68');
  define('MEMCACHE_VERSION_HOST', '10.252.0.67');
  define('VALIDATION_FAILURE_LOG', BASE_PATH . '/validate.log');
  define('REDIRECT_403_LOG', BASE_PATH . '/403.log');
  define('GOOGLE_API_KEY', (isset($_SERVER['HTTP_HOST']) && $_SERVER['HTTP
   'ABQIAAAAJlAd0HJn-kbPSqUsrS6CyhTpoeXstiwCMpsI5pU3slU-WDRPJxQts41ksQogKsy
   'ABQIAAAAJlAd0HJn-kbPSqUsrS6CyhTRJXjjauvD2gSXXVziOjeBJgmKOBTPW-8l5i5Pbk9
  Database::add('primary', array('host' => '192.168.200.142'));
  Database::add('db-tumblelogs', array('host' => '192.168.200.103'));
```

This slide is inspired by Prof. Shin Yoo's talk on software testing in Software Engineering Summer School 2018


THE
IMITATION
GAME


data stream
scientist
Turing tape
read-write head



Undecidability

- **Halting problem**: for a given arbitrary **program** and an arbitrary input, determine whether or not the program terminates within finite steps (i.e., halts) when it runs with the input.

- **Theorem**. There is no algorithm that solves the Halting problem.
  - there is no program that always returns the correct determination of whether a given arbitrary program terminates with a given arbitrary input, or not within a finite time.

- Proving the correctness of a given program may require a unique and create strategy specialized for the given program

Software Engineering Approaches to Solve SW Errors: Construct Verifiable Software Reliabily

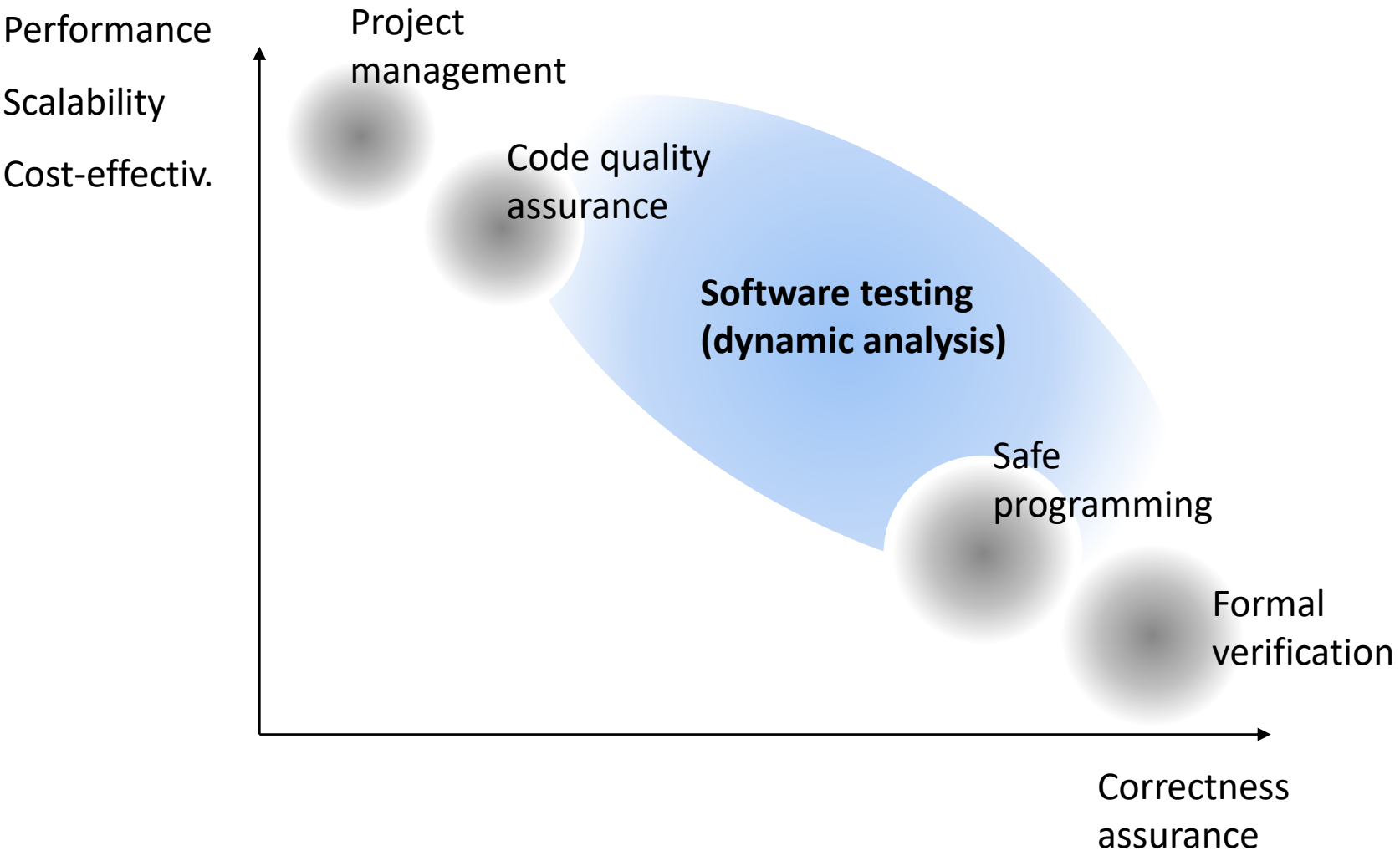
Project management

- Eliminate all potential risks at each step in SW dev. from beginning
- Methods
  - Software Process Model
    - risk assessment
    - software process quality assessment model
  - Requirement Engineering
  - Model-based SW Development
- Limitation
  - Extremely expensive
  - Cannot be responsive (at all)
  - Not generalizable to all SW dev.
  - Low code quality

Code (Product) Analysis

- Prove or disprove correctness by reasoning target program code/logic
- Methods
  - Program verification
    - static analysis
    - software model checking
  - Software testing
    - dynamic analysis
    - mutation testing
- Limitation
  - Accuracies (false alarms)
  - High computation cost

Product Management Approaches

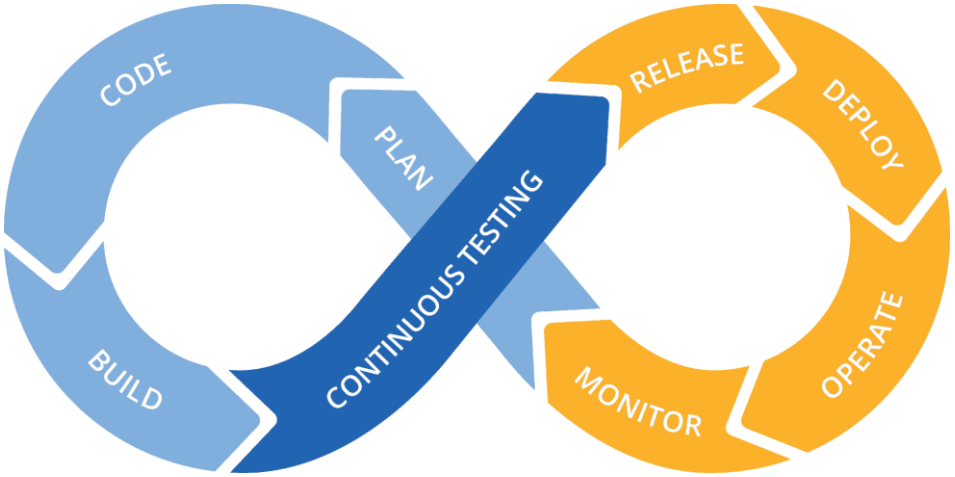


Software Testing

- Run a software product with various inputs to explore different program behaviors systematically
  - one of many verification methods
  - software testing is an essential step in software development
- Software tests are programmed for automated verification and works as immune system along project life cycle
- Many parts of software testing has become automated rapidly with the advances in program analyses, software design and machine learning techniques

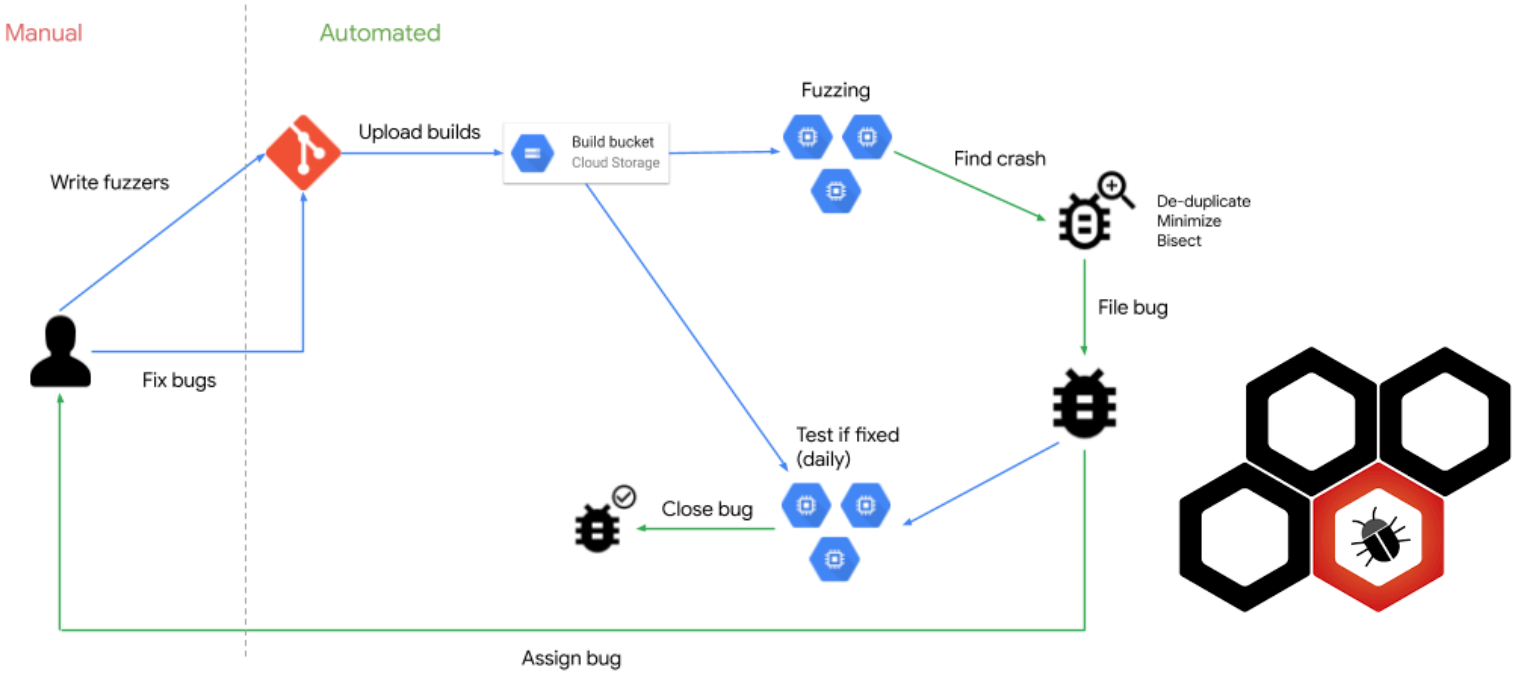
Tests As Executable Specification

- Test cases provide active and instant feedbacks to the developers and enables continuous improvement of software projects

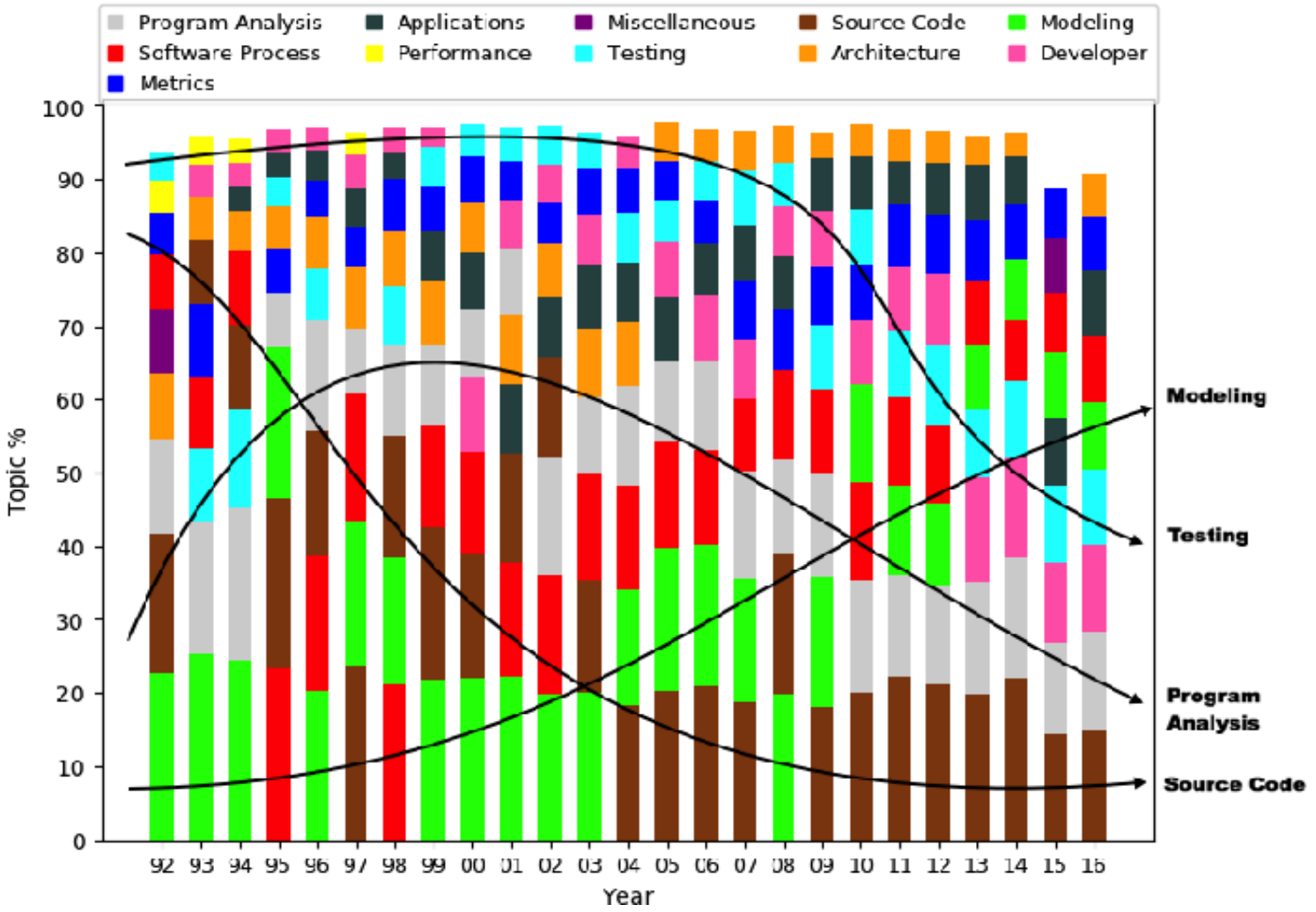


Testing for Detecting Security Issues

- The OSS-Fuzz project of Google is running 30,000 machines to test every commit of 340 open soure projects
  - found more than 30,000 bugs in last 5 years



Recent Trends in Software Engineering Research



G. Mathew et al., Finding Trends in Software Research, IEEE TSE, 2018

Topics and Class Schedule

Week	Topics
1	Course Introduction, Why SW Testing Matters, SW Bugs
2	SW Requirements, SW Testing Basics
3	Lab 1-Writing Test Driver, Input-domain modeling (blackbox testing)
4	**National Holiday** Structural Testing
5	Lab 2-Combinatorial Interaction Testing Logic Coverage
6	Data-flow Coverage Source-code Engineering with srcML,
7	Lab 3-MC/DC Coverage Quick Overview of Computation Theory,
8	Unit Testing Basics Lab 4-Turing Machine

Week	Topics
9	Lab 5-Unit Test Augmentation, Regression Testing Techniques
10	Mutation Testing, Lab 6-Test Case Prioritization
11	Symbolic Execution, Lab 7-Concolic Testing with CREST
12	Greybox Fuzzing
13	Lab 8-AFL & libFuzzer, Slicing, Delta debugging
14	Fault localization, Lab 9-Automated Debugging
15	Multithreaded Program Testing, Lab 10-Dynamic Data Race Detection
16	Guest lecture: TBD