



Using AI to Build More Secure Software

DoD Air & Space Weapon Systems Software Summit '22

Dr. Mark Sherman
Technical Director, Cybersecurity Foundations
CERT Division

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2022 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM22-0136

The SEI Is a DoD R&D Federally Funded Research and Development Center



Established in 1984 at Carnegie Mellon University

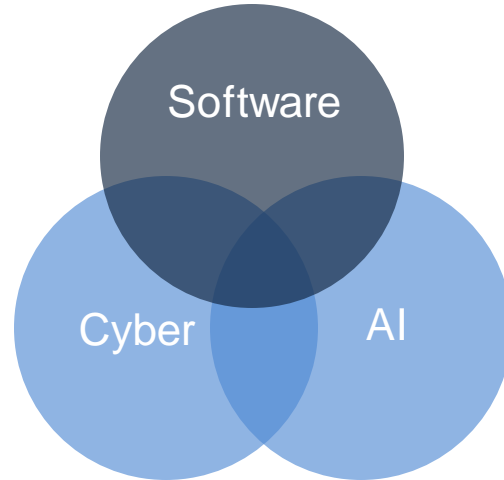
~650 employees (ft + pt), of whom about 70% are engaged in technical work

Initiated CERT cybersecurity program in 1988: the birthplace of cybersecurity

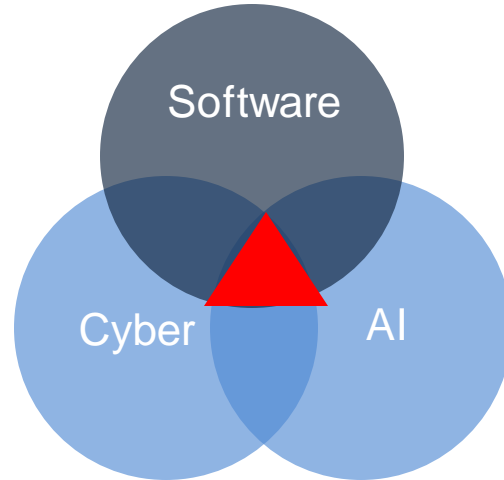
Offices in Pittsburgh and DC, with several locations near customer facilities

~\$140M in annual funding

SEI Strategic Framework



SEI Strategic Framework – Today's Focus



Software Cost and Vulnerability Threaten Military Capability



Finding and fixing software problems late in the acquisition lifecycle drives up cost and delays delivery

Latent cyber vulnerabilities and those exposed during operations or due to underlying dependencies put missions at risk

Statistically, a 10M LOC Weapons Platform written in C will be delivered with 280–1,400 exploitable vulnerabilities

F-15 Fighter Jet Hacked



Image: A U.S. Air Force F-15E Strike Eagle. (U.S. Air Force photo by Senior Airman Erin Trowe)

THE CYBERSECURITY 202: HACKERS JUST FOUND SERIOUS VULNERABILITIES IN A U.S. MILITARY FIGHTER JET

LAS VEGAS — In a Cosmopolitan hotel suite 16 stories above the Def Con cybersecurity conference this weekend, a team of highly vetted hackers tried to sabotage a vital flight system for a U.S. military fighter jet. And they succeeded.

It was the first time outside researchers were allowed physical access to the critical F-15 system to search for weaknesses. And after two long days, the seven hackers found a mother lode of vulnerabilities that — if exploited in real life — could have completely shut down the Trusted Aircraft Information Download Station, which collects reams of data from video cameras and sensors while the jet is in flight.

Will Roper, a top U.S. Air Force acquisitions executive, told the Washington Post: "there are millions of lines of code that are in all of our aircraft and if there's one of them that's flawed, then a country that can't build a fighter to shoot down that aircraft might take it out with just a few keystrokes."

Joseph Marks, Aug 14, 2019

Source: <https://www.washingtonpost.com/news/powerpost/paloma/the-cybersecurity-202/2019/08/14/the-cybersecurity-202-hackers-just-found-serious-vulnerabilities-in-a-u-s-military-fighter-jet/5d53111988e0fa79e5481f68/>

Satellites Hacked



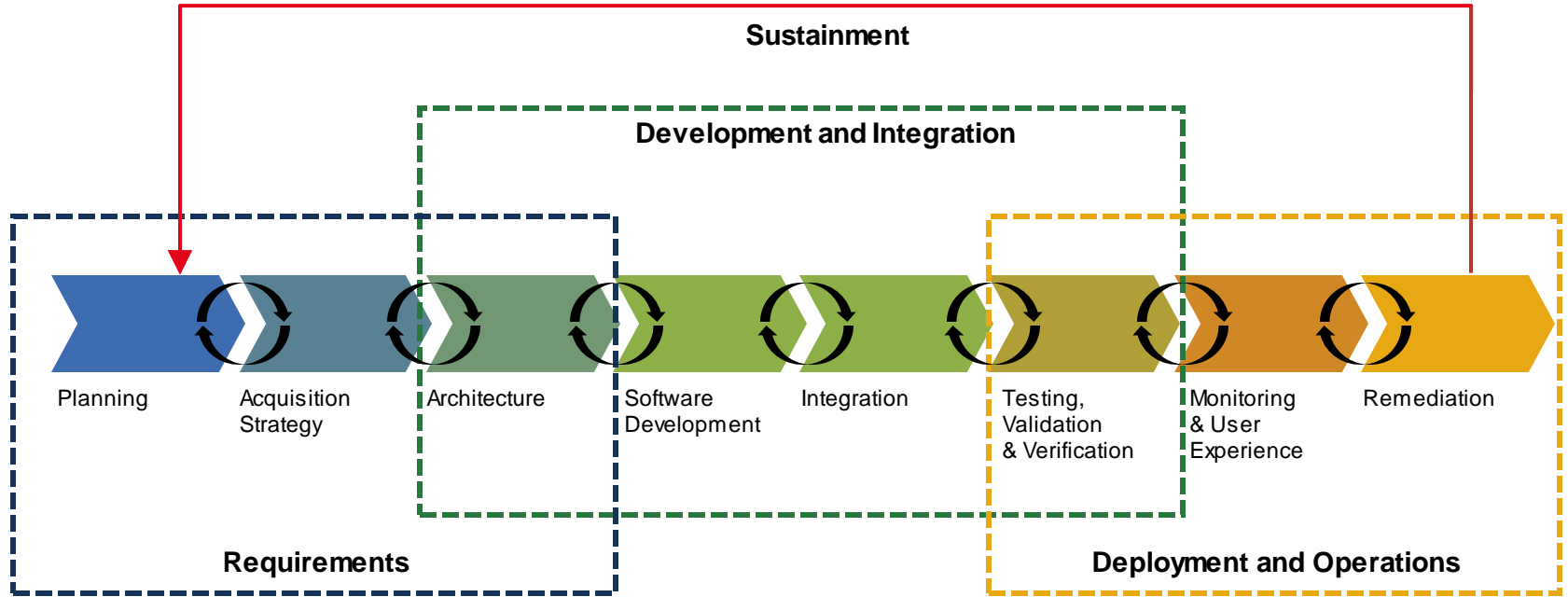
The Race to Hack a Satellite at DEF CON

By the end of the challenge, six teams had completed Challenge 5. ... Poland Can Into Space was the team chosen to have its Challenge 5 solution uploaded to a satellite, and it was successful in taking an image of the moon with a normally earth-facing satellite.

Curtis Franklin, Senior Analyst, Omdia, <https://www.darkreading.com/application-security/the-race-to-hack-a-satellite-at-def-con> , August 13, 2020

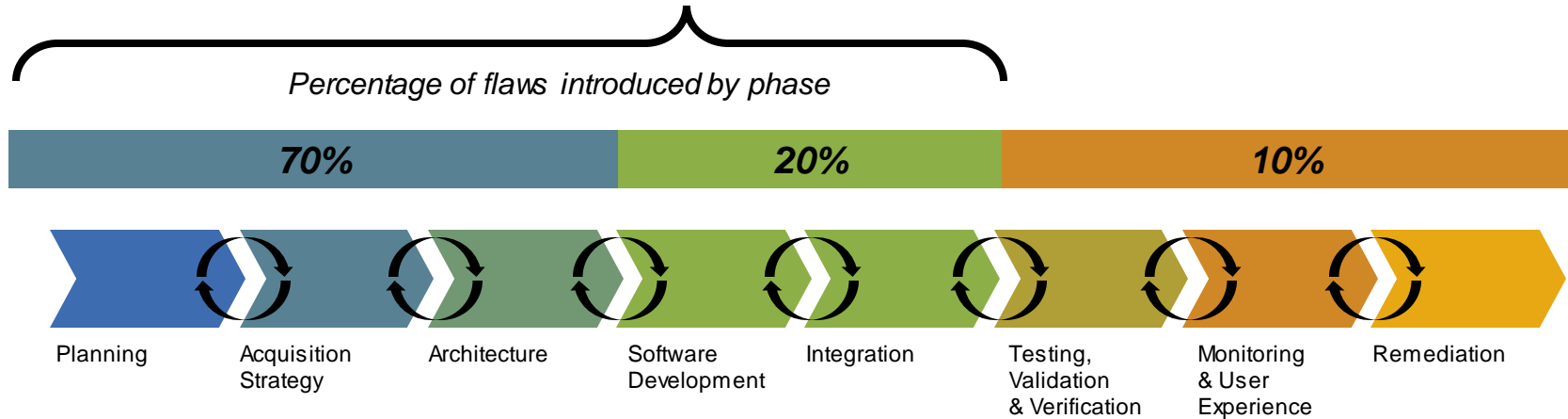
Image courtesy DEF CON Communications

Fixing Problems Late Drives Costs, Delays Deployment



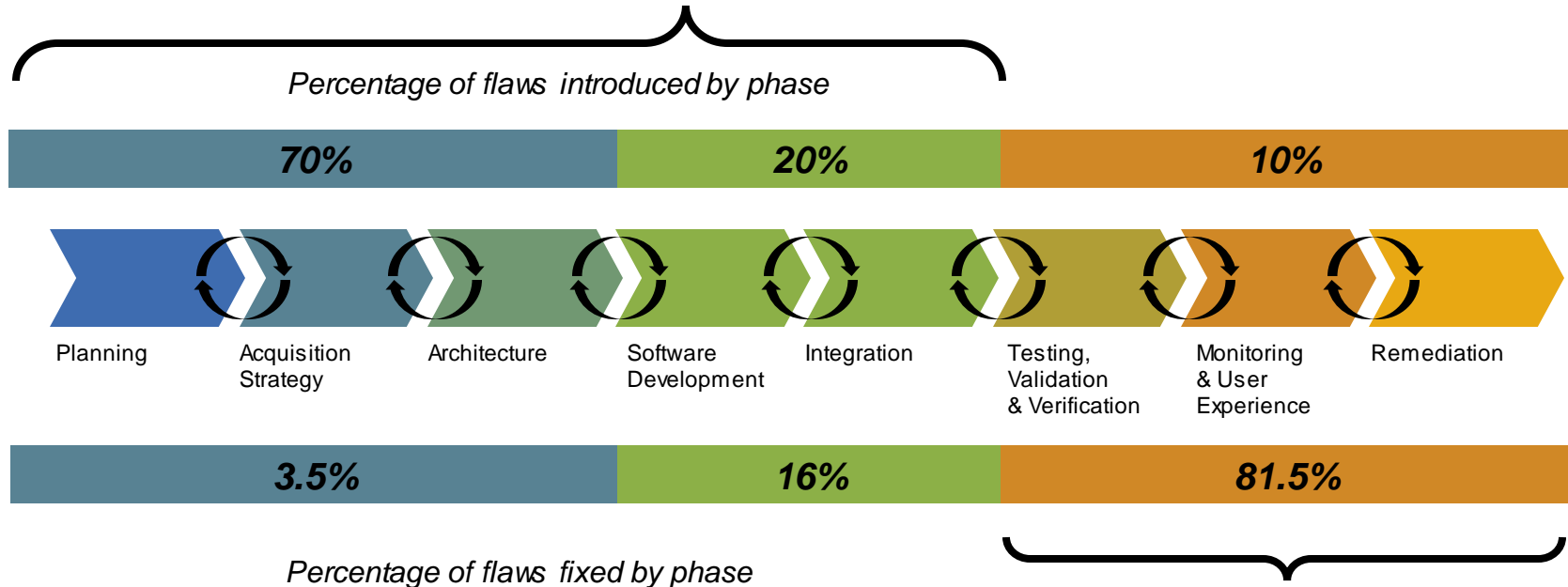
Fixing Problems Late Drives Costs, Delays Deployment

Software problems that drive costs
are introduced early in the lifecycle . . .



Fixing Problems Late Drives Costs, Delays Deployment

Software problems that drive costs
are introduced early in the lifecycle . . .



. . . But discovered late, increasing cost,
vulnerability, and schedule impact

AI in Automatic Programming – The Beginning

“The System will accept a concise formulation of a problem in terms of a mathematical notation and produce automatically a high speed program for the solution of the problem.”

AI in Automatic Programming – The Beginning

PRELIMINARY REPORT

Programming Research Group
Applied Science Division
International Business Machines Corporation

November 10, 1954

Specifications for
The IBM Mathematical FORMula TRANslating System,
FORTTRAN

“The IBM Mathematical Formula Translating System or briefly, FORTRAN, will comprise a large set of programs to enable the IBM 704 to accept a concise formulation of a problem in terms of a mathematical notation and to produce automatically a high speed 704 program for the solution of the problem.”

Source: J.W. Backus, H. Herrick and I. Ziller,
<https://archive.computerhistory.org/resources/text/Fortran/102679231.05.01.acc.pdf>

Using AI for Autocompletion

```
1 import os
2 import sys
3
4 # Count lines of code in the given directory, separated by file extension
5 def main(directory):
6     line_count = {}
7     for filename in os.listdir(directory):
8         _, ext = os.path.splitext(filename)
9         if ext not in line_count:
10             line_count[ext] = 0
11         for line in open(os.path.join(directory, filename)):
12             line_count[ext] += 1
13             line_count[ext] += 1
14             line_count[ext] += 1
15             line_count[ext] += 1
16             line_count[ext].append(
17                 line
18
19
```

© 2019 TabNine, See <https://tabnine.com/eula>

Safe, correct code could be written incrementally

- Rule based expert system
- Using n-grams for predictions
- Using deep learning to generate completion

Sources:

E. Schutte, Autocomplete from StackOverflow, 2016, <https://emilischutte.com/stackoverflow-autocomplete/>

(Jacob Jackson) TabNine, "Autocompletion with deep learning," July 18, 2019, <https://tabnine.com/blog/deep>

L. Tung, "New tool promises to turbo-charge coding in major programming languages," July 25, 2019, <https://www.zdnet.com/article/new-tool-promises-to-turbo-charge-coding-in-major-programming-languages/>

A. Hindle, E. T. Barr, Z. Su, M. Gabel, P. Devanbu, "On the Naturalness of Software," ICSE 2012, p. 837-847. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6227135>

Using AI for Code Generation



The image is a promotional banner for GitHub Copilot. At the top left, there are links for "Open Source" and "Product". The main headline reads "Introducing GitHub Copilot: your AI pair programmer". Below this, a sub-headline states "Today, we're launching a technical preview of GitHub Copilot, a new AI pair programmer that helps you write better code." A "Technical preview" badge is positioned above the second headline, "Your AI pair programmer". The central part of the banner features a screenshot of a code editor with a dark theme, showing a JavaScript function that fetches data from a URL. The code is as follows:

```
1 const fetchMAGAPictureOfTheDay = () => {
2   const fetch = fetch('https://api.nasa.gov/insanequery/astrotoday?lang=EN&KEY=');
3   method: 'GET',
4   headers: {
5     'Content-Type': 'application/json',
6   },
7 },
8
9 // ...
10 then(response => response.json())
11   .then(data => {
12     return data;
13   })
14 }
```

At the bottom of the banner is the GitHub Copilot logo, which consists of the GitHub octocat icon followed by the text "GitHub Copilot".

Nat Friedman, June 29, 2021, <https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/>

AI in Automatic Programming: Generating Coded Through Search – High Assurance SPIRAL

High Assurance Spiral In A Nutshell

Problem and main idea

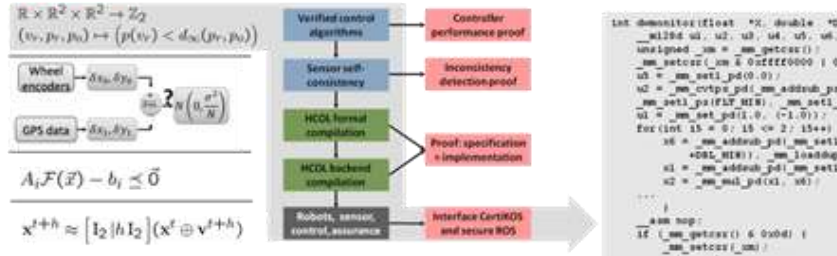
Co-synthesize high-quality code and proof for sensor-fusion based self-consistency algorithms



Results

- Four algorithms in HA Spiral formalized/in library dynamic window monitor, Z test for sensor mean, feasible state set test, ROS infrastructure math code
- HA Spiral Tool/GUI ready for beta testers soon
- End-to-end proof/code co-synthesis and deployment deployed on Landshark and ABCar Simulator
- Rule based backend compiler proof of concept Implemented in K framework, proofs in Isabelle

Approach

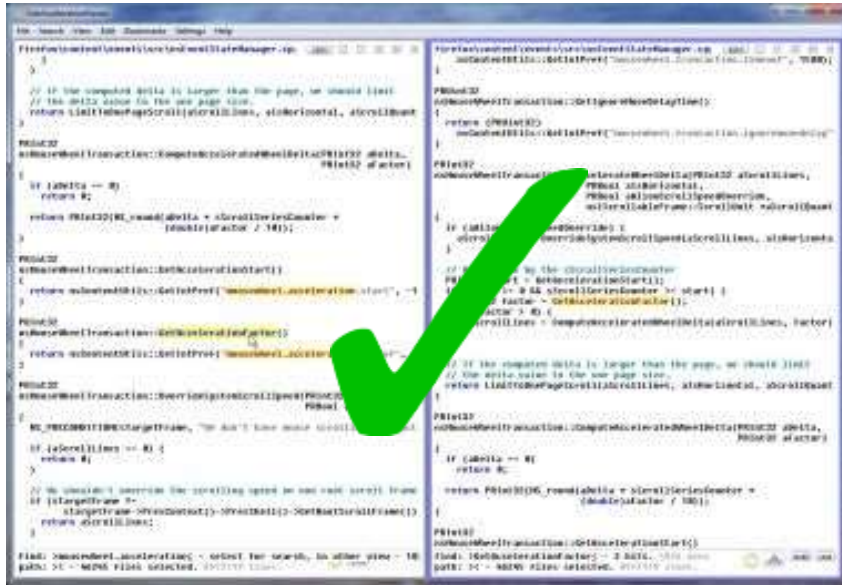


```
int deinitializerloat "X: double "0
    _m128d u1, u2, u3, u4, u5, u6
    unsigned _sm = _mm_getcsr();
    _mm_setcsr(_sm & 0xffff0000 | 0);
    u1 = _mm_set1_pd(0.0);
    u2 = _mm_cvtps_pd1._mm_addrub_pd1
    _mm_set1_pd(FLY_MIN); _mm_set1_
    u1 = _mm_set_pd(1.0, (-1.0));
    for(int i5 = 0; i5 <= 2; i5++) {
        u6 = _mm_addrub_pd1._mm_set1
            +DCL_MIN); _mm_loadup
            u1 = _mm_addrub_pd1._mm_set1;
            u2 = _mm_set_pd(x1, u6);
        ...
    }
    asm nop;
    if (_mm_getcsr() & 0x00d) {
        _mm_setcsr(_sm);
    }
}
```

“High Assurance SPIRAL aims to solve the last mile problem for the synthesis of high assurance implementations of controllers for vehicular systems that are executed in todays and future embedded and high performance embedded system processors.”

Sources: Franz Franchetti, José M. F. Moura, Manuela Veloso, Andre Platzer, Soumya Kar, David Padua, Jeremy Johnson, Mike Franusich, High Assurance Spiral: Scalable and Performance Portable Domain-Specific Control System Synthesis, <https://users.ece.cmu.edu/~franz/hacms.htm>; <http://www.spiral.net/>

Finding Programming Vulnerabilities – Source Code as Natural Language



Analyzing source code for insecure coding

- Supplements compiler-style checking
- Treats programs like natural language
- Transformer neural nets for dispersed errors

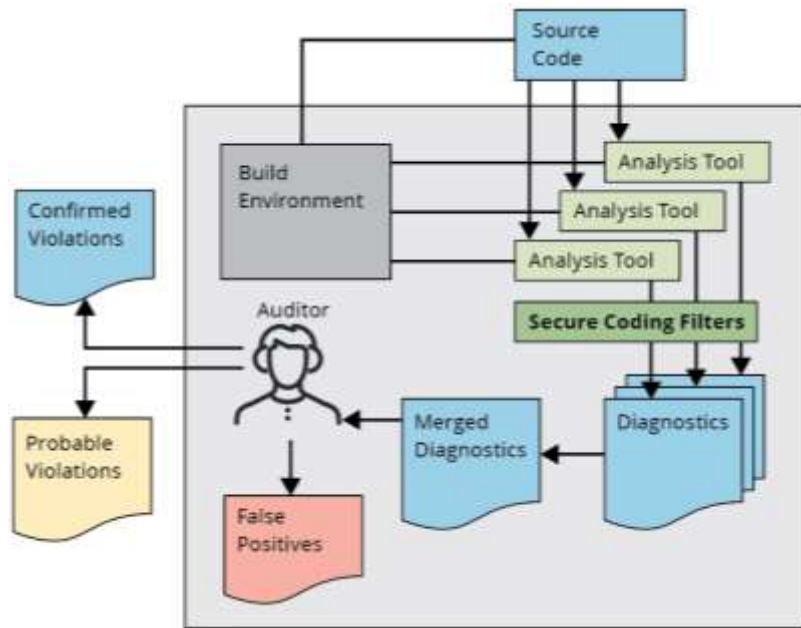
Sources: Carson D. Sestili, William S. Snavey, Nathan M. VanHoudnos, Towards securitydefect prediction with AI, Sep 12, 2018, <https://arxiv.org/abs/1808.09897>

Song Wang, Taiyue Liu, and Lin Tan. 2016. Automatically learning semantic features for defect prediction. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 297-308. DOI: <https://doi.org/10.1145/2884781.2884804>

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. code2vec: learning distributed representations of code. Proc. ACM Program. Lang. 3, POPL, Article 40 (January 2019), 29 pages. DOI: <https://doi.org/10.1145/3290353>

L. Buratti, S. Pujar, M. Bornea, S. McCarley, Y. Zheng, G. Rossiello, A. Morari, J. Laredo, V. Thost, Y. Zhuang, G. Domeniconi, "Exploring Software Naturalness through Neural Language Models," arXiv:2006.12641v2 [cs.CL] 24 Jun 2020, <https://arxiv.org/abs/2006.12641v2>

Combining Multiple Tools with AI to Find Source Code Flaws – SCALe

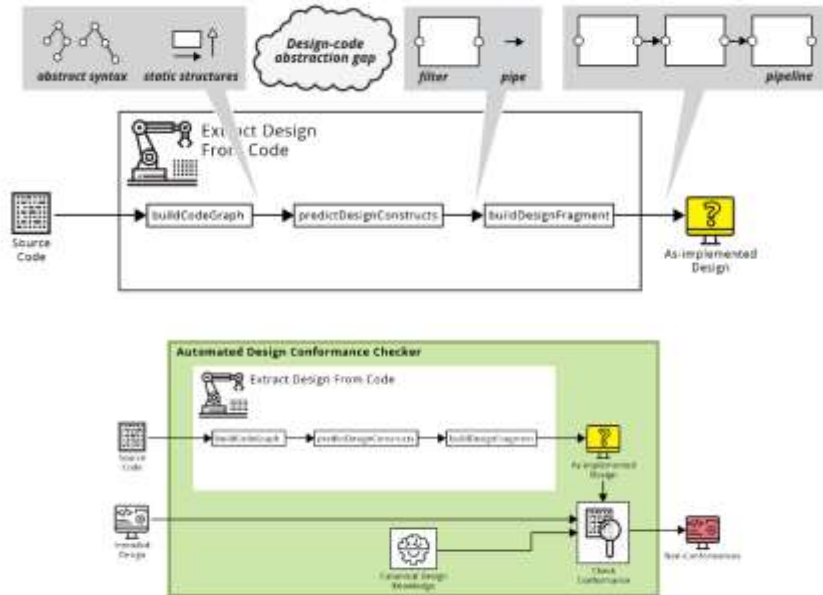


Using AI and machine learning to combine tool and environmental data

- Multiple static code analyzers
- Multiple environmental features
- Multiple classification techniques

Source: Lori Flynn, SCALe: A Tool for Managing Output from Static Analysis Tools, Sept 24, 2018, https://insights.sei.cmu.edu/sei_blog/2018/09/scale-a-tool-for-managing-output-from-static-code-analyzers.html ;
Lori Flynn, Automating Static Analysis Alert Handling with Machine Learning, MIT Lincoln Labs Cyber Security, Exploitation and Operations Workshop, June 19, 2018

Checking the design consistency with AI



Vulnerabilities can be introduced when design patterns are broken

- Introduced by technical debt
- Introduced by unwitting maintenance

Use AI on source code to close the design-code abstraction gap

- Track changes to ensure compliance and consistency using AI

Sources: J. Ivers, I. Ozkaya, R. Nord, Can AI Close the Design-Code Abstraction Gap? International Workshop on Software Engineering Intelligence, IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, <https://ieeexplore.ieee.org/document/8967435>

R. Nord, L. Pons, "Automated Design onformance during Continuous Integration," Nov 2021, https://resources.sei.cmu.edu/asset_files/Presentation/2021_017_001_741574.pdf

Using AI to Drive Test Inputs – Fuzzing



“Fuzzing:” Generating and testing random inputs

Original: Random or deterministic

Now: Use AI to guide generation of sample inputs

Sources: A. Householder, Announcing CERT Basic Fuzzing Framework Version 2.8, Oct. 5, 2016, <https://insights.sei.cmu.edu/cert/2016/10/announcing-cert-basic-fuzzing-framework-bff-28.html>

G. Yan, J. Lu, Z. Shu ; Y. Kucuk, “ExploitMeter: Combining Fuzzing with Machine Learning for Automated Evaluation of Software Exploitability,” 2017 IEEE Symposium on Privacy-Aware Computing (PAC), 1-4 Aug. 2017, <https://doi.org/10.1109/PAC.2017.10>

D. She, K. Pei, D. Epstein, J. Yang, B. Ray, S. Jana, “NEUZZ: Efficient Fuzzing with Neural Program Smoothing,” 40th IEEE Symposium on Security and Privacy, May 20--22, 2019, San Francisco, CA, USA, <https://arxiv.org/pdf/1807.05620.pdf>

Using AI to Improve Penetration Testing



Variety and combination of manual techniques can be executed by an AI system

- AI planning using an attack graph against attack surfaces
- Markov Decision Process (or Partially Observable Markov Decision Process) over application state
- Reinforcement learning

Sources:

K. Durkota and V. Lisy, "Computing Optimal Policies for Attack Graphs with Action Failures and Costs," Conference: Proceedings of the 7th Starting AI Researchers' Symposium (STAIRS), December 2013, https://www.researchgate.net/profile/Karel_Durkota/publication/273640839_Computing_Optimal_Policies_for_Attack_Graphs_with_Action_Failures_and_Costs

C. Sarraute, O. Buffet, and J. Hoffmann, "POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing," AAAI, 2012, <https://arxiv.org/pdf/1307.8182>

J. Schwartz, "Autonomous Penetration testing using Reinforcement Learning, Nov 16, 2018, <https://arxiv.org/ftp/arxiv/papers/1905/1905.05965.pdf>

Using AI for Automated Cyber Red Teaming



Red teaming simulates an attack across a system to evaluate the cybersecurity of a mission

Typically depends on having

- Cyber ontology of mission and system elements supporting mission
- Attack trees
- Mission priorities
- Planners: state-space planners, planning graph planners, and hierarchical task network-based planners

Sources:

S. Randhawa, et al, "Mission-Centric Automated Cyber Red Teaming," ARES 2018, Proceedings of the 13th International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany, https://www.researchgate.net/publication/327005899_Mission-Centric_Automated_Cyber_Red_Teaming

S. Upton, et al, "Breaking blue: Automated red teaming using evolvable simulations," Proceedings of Genetic and Evolutionary Computation Conference 2004, 2004, <http://gpbib.cs.ucl.ac.uk/gecco2004/WMSA015.pdf>

J. Yuen, "Automated Cyber Red Teaming," DTIC Document 2015, <https://apps.dtic.mil/docs/citations/ADA618584>

Automated Program Repair – DARPA Cyber Grand Challenge



“Mayhem” demonstrated automated cyber defense

- Detect attack on program
- Analyze changes to program
- Deploy updated software

Sources:

DARPA, “Mayhem” Declared Preliminary Winner of Historic Cyber Grand Challenge, Aug 4, 2016,
<https://www.darpa.mil/news-events/2016-08-04>

“Automated vulnerability detection and prevention,” <https://www.cylab.cmu.edu/research/carc/research.html>

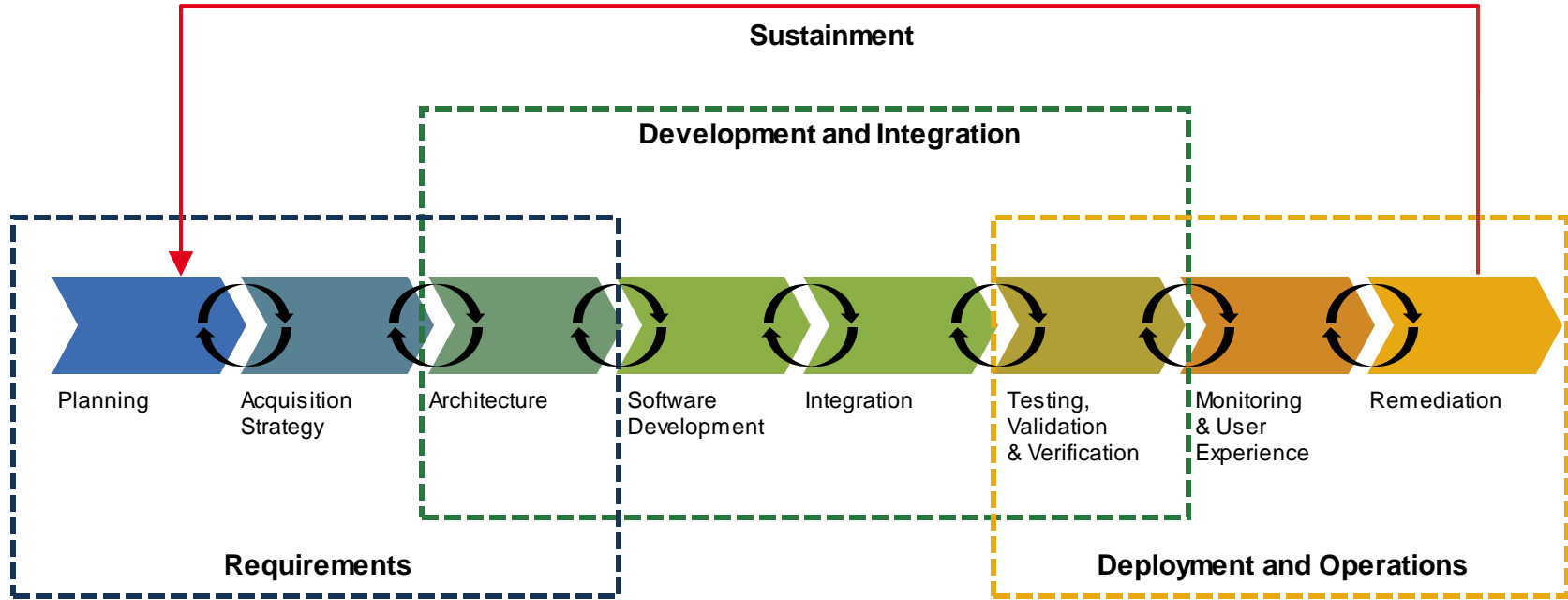
AI Supporting Judgement – IBM Watson to Improve Assurance



- Acquisition programs generate voluminous documentation
- Assurance is based on assembling and reviewing relevant evidence from documents
- Finding appropriate evidence or explanations can be challenging
- SEI proof of concept

Source: Mark, Sherman, Verifying Software Assurance with IBM's Watson, <https://www.youtube.com/watch?v=aW3497xhpyY>, Sep 11, 2017

Summary: Using AI to Produce Better Programs Across the Lifecycle



Contact Us



Carnegie Mellon University

Software Engineering Institute

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

412-268-5800

888-201-4479

info@sei.cmu.edu

www.sei.cmu.edu