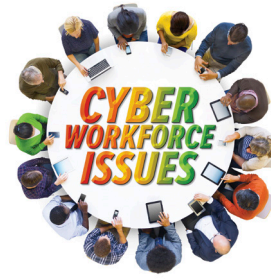


CROSSTALK

March / April 2016 **The Journal of Defense Software Engineering** Vol. 29 No. 2



Cover Design by
Kent Bingham

Departments

- 3 From the Sponsor
- 36 Upcoming Events
- 38 BackTalk

Cyber Workforce Issues

4 Addressing Narrowing Cyber Workforce Gaps with Intrusion Detection and Response Automation

Cybersecurity threats continue to expand and the need to increase our cyber workforce across the public and private sectors is exceeding our current capacity to educate and train cybersecurity professionals.

by Dr. Peter Fonash and Dr. Thomas Longstaff

10 People-driven Process-enabled Software Development: A 21st Century Imperative

In the 21st century, software will continue to grow in a sociotechnical ecosystem comprising customers, end users, developers, maintainers, testers, and other stakeholders.

by Azad M. Madni

14 Developer Training: Recognizing the Problems and Closing the Gaps

Problems with the way we have historically trained developers, and continue to do so, gets in the way of learning to do secure development.

by Mike Lyman

18 Building and Operating a Software Security Awareness and Role-Based Training Program

Today's application software developers are encouraged (and rewarded) by developing lots of code quickly but are usually not provided the adequate time and skills to build it with security and quality as top of mind requirements.

by Mark S. Merkow

24 Strategic Human Resource Management of Government Defense R&D Organizations

Reformulating the sustainable growth rate metric to account for R&D experience within government R&D organizations.

by Kadir Alpaslan Demir, Ph.D.

31 PixelCAPTCHA: A Unicode Based CAPTCHA Scheme

A new visual CAPTCHA scheme leverages the 64K Unicode code points from the Basic Multilingual Plane (plane 0) to construct CAPTCHAs that can be solved with 2 to 4 mouse clicks.

by Gursev Singh Kalra

CROSSTALK

NAVAIR Jeff Schwab

DHS Peter Fonash

309 SMXG Karl Rogers

76 SMXG Mike Jennings

Publisher Justin T. Hill

Article Coordinator Heather Giacalone

Managing Director David Erickson

Technical Program Lead Thayne M. Hill

Managing Editor Brandon Ellis

Associate Editor Colin Kelly

Senior Art Director Kevin Kiernan

Art Director Mary Harper

Phone 801-777-9828

E-mail Crosstalk.Articles@hill.af.mil

Crosstalk Online www.crosstalkonline.org

CROSSTALK, The Journal of Defense Software Engineering is co-sponsored by the U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Ogden-ALC 309 SMXG and Tinker-ALC 76 SMXG. DHS co-sponsor: Office of Cybersecurity and Communications in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of **CROSSTALK** providing both editorial oversight and technical review of the journal. **CROSSTALK's** mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

Subscriptions: Visit www.crosstalkonline.org/subscribe to receive an e-mail notification when each new issue is published online or to subscribe to an RSS notification feed.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the **CROSSTALK** editorial board prior to publication. Please follow the Author Guidelines, available at www.crosstalkonline.org/submission-guidelines. **CROSSTALK** does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the authors and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with **CROSSTALK**.

Trademarks and Endorsements: **CROSSTALK** is an authorized publication for members of the DoD. Contents of **CROSSTALK** are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CROSSTALK Online Services:

For questions or concerns about crosstalkonline.org web content or functionality contact the **CROSSTALK** webmaster at 801-417-3000 or webmaster@luminpublishing.com.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

CROSSTALK is published six times a year by the U.S. Air Force STSC in concert with Lumin Publishing luminpublishing.com. ISSN 2160-1577 (print); ISSN 2160-1593 (online)

CROSSTALK would like to thank DHS for sponsoring this issue.



Cybersecurity is a risk management issue, not just a technology issue. Regrettably, for many years, cybersecurity risk decisions were delegated to low levels, ignored, or not even imagined by senior leaders as well as those who conceive, engineer, and sustain software-intensive systems. Delivering products often focused on the availability and functionality of a product rather than balancing those attributes with data integrity and confidentiality. Some producers accepted risk for those who used their products without incorporating cybersecurity best practices into product design. As a consequence, those with malicious intent (commonly referred to as “bad actors”) increasingly leverage product weaknesses to gain unauthorized access to information, presenting threats to privacy, public safety, protection of intellectual property, and national security. The threats now are many and present real and difficult challenges for those charged to protect vital information.

Security and resiliency must be tightly integrated throughout the conception, design, implementation, deployment, maintenance, and operation of our cyber assets. Ultimately, it falls on the ability and attention of cyber workers across a broad range of specialties to better manage risk by identifying and mitigating vulnerabilities at every stage of a product's life. Unfortunately, the average cyber worker is increasingly over-matched when confronted by an ever-increasing demand to master more and more skills, languages, configurations, etc. You, our readers, can help those cyber workers better manage cyber risk by “baking-in” security at every stage of a system's lifecycle. By doing so, you can deny attackers opportunities to disrupt or corrupt vital services. The Department of Homeland Security sponsors this issue of CrossTalk to feature new ideas that promise to better empower the cyber workforce.

As you read through this issue, please ask yourself a couple of questions. For instance, are we fully utilizing the talent of our cyber analysts, or can we offload some of this work to automated cyber tools? Are we developing and delivering easy-to-use tools where security is a default setting rather than an option? Is process-driven software development maximizing the value of the people in the process, or can it be improved? How can training programs better recognize and address common gaps in the developer knowledge base? Are attempts to increase the size of the workforce sustainable? Analysis by experts in the field in the following articles shed light on some of these topics.

In today's environment marked by a vexing scarcity of cyber talent, it is clear that we cannot afford to waste the potential of our cyber human capital. In fact, to keep pace with emerging challenges, we must do our utmost to expand its potential. No single, easy solution has emerged to remedy our cyber challenges, and finding today's solutions will require plenty of smart, innovative people, new ideas and approaches to solving problems and, ultimately, a lot of hard work. It is important to strive to ensure that within our cyber workforce, nothing is wasted.

Greg Touhill
Brigadier General, USAF (ret)
Deputy Assistant Secretary
Office of Cybersecurity and Communications
U.S. Department of Homeland Security

Narrowing Cyber Workforce Gaps with Intrusion Detection and Response Automation

Dr. Peter Fonash, Department of Homeland Security
Dr. Thomas Longstaff, Applied Physics Laboratory

Abstract. Cybersecurity threats continue to expand and the need to increase our cyber workforce across the public and private sectors is exceeding our current capacity to educate and train cybersecurity professionals. In other industries, automation has effectively increased workforce productivity, leading to higher performance, quality, and affordability across the global economy. Within the cybersecurity industry, high-level automated responses to intrusion, where the shortage of cyber professionals is most notable, have not yet been widely adopted. While there are several barriers that have prevented automation from flourishing in the cyber realm, an investigation has shown that automation could provide an effective force multiplier to help the cyber workforce cope with an escalating barrage of increasingly efficient attacks.

Introduction

The gap between the capabilities of our adversaries and our defenders continues to increase across the cyber ecosystem.¹ While some of the growing number of intrusions have been addressed² through software assurance, training, security processes, procedures, and risk reduction, the productivity of the cyber workforce that manages our networks and in particular those that respond to attacks and intrusions remains inadequate.

To illustrate the growing gap between the effectiveness of the attackers and defenders, Figure 1 compares the percent of breaches where the time to compromise a system was days or less to the time it took to discover a compromise in days or less. This gap will only increase as the number and capabilities of our cyber workforce fail to keep pace with the increase in capabilities of the adversaries.

Some current proposals to address this increasing gap include closing wage differences between the public and private sector, investing in additional cyber educational resources, focusing on education in science, technology, engineering, and mathematics, and defining and improving the career paths for cyber professionals. Unfortunately, these efforts do not address the fundamental problem that manual processes alone will not be able to close this growing gap in cybersecurity. Historically, we have addressed this gap through security controls (e.g. National Institute of Standards and Technology Special Publication 800-53 series) and through risk reduction (e.g. the Framework

for Improving Critical Infrastructure Cybersecurity) [1]. While implementing standard security best practices can be effective in preventing attacks [2], they do not enable timely response to automated attacks that bypass the security controls. This article discusses a recent investigation that has explored capability improvements for handling intrusions by increasing automation and interoperability, in order to better understand the potential benefits to the cyber workforce.

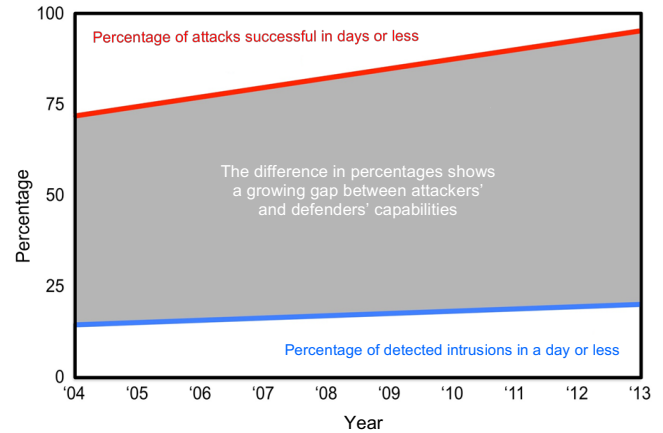


Figure 1: Indicates Attacker versus defender efficiency capability. Attackers successfully improved their time to compromise a systems so less than 75 percent of compromises in 2004 were completed in days or less to around 90 percent in 2013, but the efficiency capability to detect system compromises within days only improved from approximately 13 to 20 percent over the same time period. Source: adapted from Verizon's 2014 Data Breach Investigations Report (www.verizonenterprise.com/DBIR).

Critical Cyber Workforce Need

Published reports, articles, and blogs, described in this section, discuss the critical need for a larger and more varied cyber workforce. Not only are there not enough people to fill existing cybersecurity positions, but the number of information technology systems and networks continue to grow at an exponential rate. As the number of devices and networks continues to grow, this increase will result in a greater need for people to manage and protect those systems and networks. Further, the adoption of an increasingly diverse array of technology products encourages organizations to hire a greater variety of specialized experts to secure these new products.

Given the increase in both the volume and variety of the demand for cyber talent, our current approach to grow the cyber workforce isn't working fast enough to protect the rapidly growing cyber ecosystem. As figure 2 illustrates, the evolution of the Internet of Things is also expected to drastically expand the number of devices and the types of technologies within the ecosystem. The challenge today is to develop cyber technologies that enable a finite number of cybersecurity professionals to protect tens of billions of devices and to ensure that there is a breadth of expertise within the cybersecurity workforce that will be able to defend a rapidly diversifying set of systems, applications, and devices within the global cyber ecosystem.

The number of cybersecurity professionals is already insufficient, and the problem is getting worse. According to the Council on Cybersecurity:

The number of entrants into the IT workforce has not kept up with demand, leaving a significant gap in capacity to adequately protect these networks from attack. At the same time, the lack of clarity and consistency in job profiles, competency models, skills assessment and workforce management contribute to a sub-optimal deployment of these scarce resources [3].

One telling statistic from a recent Ernst and Young report on skilled resources shows that 53% of surveyed organizations cite lack of skilled resources as a major cybersecurity challenge [4]. While the growth of the cyber workforce has been falling behind, the frequency of detected cyber attacks is picking up. According to a February 2015 Government Accountability Office report, there were 67,168 cyber incidents reported in 2014 that negatively affected federal systems, up 1,121 percent from 2006 [5].

Improvements in intrusion detection and increased reporting have contributed to the increase in the number of reported incidents, but these figures clearly indicate that there is a rapidly increasing number of incidents to be handled by the cyber workforce. Judging by the growing number of detected intrusions, we would need to grow the cyber workforce at an unreasonable rate in terms of both cost and training to keep pace with our cyber adversaries.

The current approach of protection and risk reduction, as reflected in the NIST Cybersecurity Framework [6] and NIST SP-800-53 [7] controls, has provided a defensive framework and processes for organizations to use in structuring personnel, technology solutions, and information systems. The framework and controls do seek to allow for the maturity of individual security tools, but the productivity of the cybersecurity workforce has not been a focus of the improvements. To date, the security industry has focused on developing new tools and capabilities with little concern for interoperability, information sharing, and integration among various vendors' products.

The rapid development of vendor's products based on multiple proprietary interfaces has created a flourishing cottage industry of unique solutions. While analysts may have access to a plethora of specialized tools, they lack the resources to translate numerous data inputs into situational awareness and mitigation actions in the timeframes required to detect and mitigate increasing threats. Unless the cybersecurity workforce is provided with capabilities to dramatically improve their productivity, we will need to expend an ever increasing portion of an organization's budget on cybersecurity with unsatisfactory results.

The focus on workforce education (e.g., the NIST National Initiative for Cybersecurity Education [8]) has received positive attention as we attempt to educate our way out of some of the critical shortfalls of the cyber personnel challenge. However, despite the proliferation of new public and private training and education programs, reports from industry and government sectors indicate that there is still significant difficulty finding and retaining qualified cybersecurity professionals.

A January 2015 GAO report recommended that the Office of Personnel Management establish a core set of metrics to help identify and close mission-critical gaps such as those in the

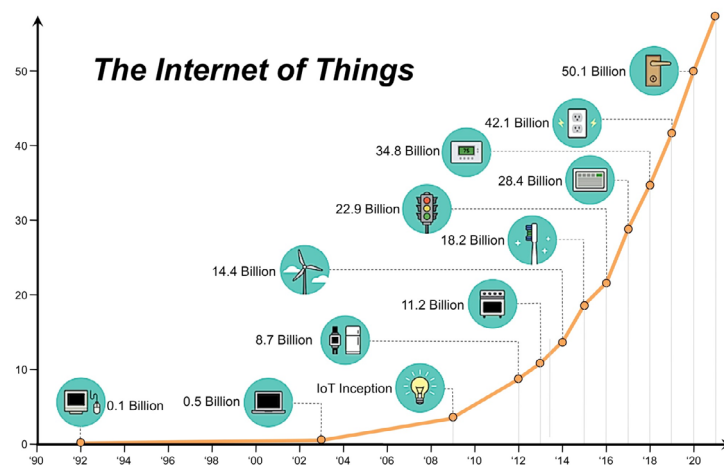


Figure 2: The cyber ecosystem has been expanding much faster than the workforce can scale up to protect it, and the growth is expected to continue long into the future. Source: theconnectivist.com

cyber workforce [9]. Officials launched an initiative from 2013 to 2014 to categorize the cybersecurity specialty areas to better support these metrics and these categories may help to more intelligently allocate cyber talent across the government [10]. While these categories could conceivably help to better place cyber talent within the government, its solution is still based on a strategy to fill the cyber talent gap through additional hiring.

This has been a prevailing strategy the Government has been using to meet the cyber challenge. According to a 2014 RAND report, over 92,863 civilian cyber employees across the Federal Government, one of every 22 Federal workers, works in this field [11]. This indicates that the US Government is attempting to step up to its growing cybersecurity challenges, but in the long run, the availability of cyber personnel will not be sufficient to counter the growing number of intrusions within an expanding cyber ecosystem.

Given the current and projected situation in cyber, it is unreasonable to assume that training and education alone will close this gap between the growing need for secure cyber systems. Advances in technology within the cyber ecosystem are necessary to increase the productivity of the available cyber workforce. One promising approach to help accomplish this is to use automation to increase the security and resiliency of the entire cyber ecosystem.

The Secure and Resilient Cyber Ecosystem

Automation has improved the per-hour productivity of the workforce across industries from manufacturing, to agriculture, to data analysis, to business workflow. Computers and information technology have been integral in developing more efficient processes that have enabled professionals to greatly accelerate time-consuming and repetitive tasks. But further than increasing process efficiency, automation has allowed us access to a vast array of new products and services such as online banking, Software as a Service, and many other new business models. Truly, automation has revolutionized modern industries.

Automation holds similar promise for the cybersecurity sector. Cyber workers today must complete many repetitive tasks which could be automated to free up the cyber labor force to spend more time on the substantive analysis that leads to better and

more informed decisions. But the promise of automation reaches much further than improving today's cyber defense systems. The speed and reliability of automated processes could enable the evolution of a secure and resilient cyber ecosystem with new capabilities that are not yet attainable by today's enterprises.

The Department of Homeland Security, in partnership with the Department of Defense, has been exploring these possibilities through work on a concept called Integrated Adaptive Cyber Defense (IACD) which is currently being prototyped in a laboratory environment at the Johns Hopkins University Applied Physics Laboratory (JHU/APL).³ The principles underlying this approach were outlined earlier this year in an article of IEEE's Computer magazine titled "Cybersecurity: From Months to Milliseconds" by Dr. Fonash and Dr. Schneck, [12] The elements essential to IACD and a secure and resilient cyber ecosystem are illustrated in Figure 3.

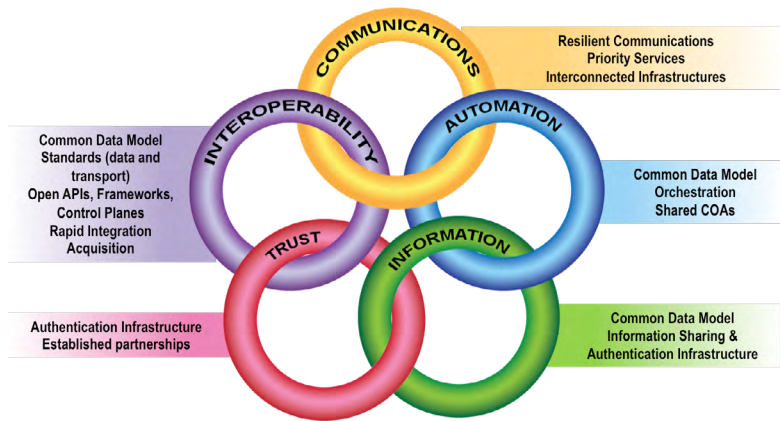


Figure 3: Essential Elements for IACD and a secure and resilient cyber ecosystem.

IACD seeks to enable enterprises to detect and respond to threats; share standardized, machine-readable information on threat indicators; and offer improved visual data analytics all within seconds or minutes. Cyber analysts equipped with these capabilities would be able to make much better informed and faster decisions to defend their cyber networks. IACD has the further potential to allow defenses to respond to intrusions and blunt attacks before adversaries have the ability to compromise their systems and networks.

In many organizations today, these processes usually take months and often don't happen at all. Investigators working on IACD have identified inter-related challenges barring the development of integrated adapted cyber defenses within today's cyber ecosystem. Chief among these is a lack of security automation.

As discussed above, adversaries are improving their capabilities faster than defenders can keep up. Regardless of the lack of cybersecurity analysts to defend information systems, the ability of people to complete all of the necessary procedures and analysis to make effective decisions to counter cyber-attacks is unlikely or impossible. Automation is clearly a promising solution, but there are barriers that have prevented the formation of automated and adaptive cyber defenses.

Currently, tools supplied by different vendors have a limited ability to work together. This means that consumers of cybersecu-

rity tools and services can only take advantage of the automation and integration of suites of tools offered by single vendors. They become locked in to a single vendor and it is often costly and risky to integrate security tools from outside their vendor suite, including many of the newest and most innovative products.

Exacerbating the problem is that it is almost impossible for organizations to address all the functions in the Cybersecurity Framework with a single vendor's product suite. This problem forces organizations to incorporate products from multiple vendors that don't necessarily use the same interfaces, data definitions, or syntax, thus requiring them to use costly middleware and manual intervention to obtain situational awareness, take appropriate mitigation actions or share information in a timely manner.

The IACD team has proposed that current products could be adapted to support common APIs and a common messaging technology that would enable interoperable tools from different vendors to work together to effectively defend enterprise systems in cyber relevant time. Developing these plug-and-play capabilities would enable enterprises to easily swap out tools with any product or set of products that adheres to the specifications of these common APIs. This would enable cybersecurity professionals to easily incorporate new or updated solutions into their architectures with minimal customized configuration.

Defining a consistent set of APIs and messaging technology involves an evolution toward a stable and consistent functional taxonomy that does not change as tools are updated or products are changed. This common set of functions also means that cyber workers would no longer need to manually update automated courses of action each time they make a change to their networks. This consistent foundation would enable automated courses of action to evolve to increasing levels of complexity, relatively independent of the underlying tools supporting them. Organizations would then have the option to automate aspects of their network defense to best address their enterprise risk and the freedom to optimize the use of their cybersecurity professionals.

Further, once these functional elements are normalized across enterprises, new opportunities emerge for cooperation across the broader cyber ecosystem. Automation has the potential to provide analysts with increased situational awareness of their enterprise and the overall ecosystem security and health, and the ability to share threat indicators and automated courses of action. As enterprises adopt infrastructures to more effectively establish trust in information sources with those information sources, these ecosystem-wide solutions show increasing promise.

One example of cooperation across the ecosystem is the development of an information sharing infrastructure that would provide the capability to share indicators and automated COAs before the adversary is able to compromise other members of the community. This information sharing infrastructure could bring together communities with common attributes where there can be a trust relationship to automatically share the current state of intrusions and allow a collection of organizations to band together against common adversaries. These relationships could provide analysts detailed information on an intrusion from the local community within seconds of an intrusion.

IACD Experiment Uses Automation to Increase Workforce Efficiency

To assess the viability of this vision of the cyber ecosystem, as part of the IACD initiative, JHU/APL constructed a testbed that would study challenges that stand in the way of the evolution of the secure and resilient cyber ecosystem. This testbed system employs commercial off the shelf solutions connected through orchestration and custom connectors to provide a set of interrelated services.

In the experiment, existing products and services that support the individual security elements of the testbed system (Boundary Protections, Network Protections, and Host Protections) were connected through a messaging infrastructure to coordinate the response to intrusion indicators. Two experiments were run on this testbed to explore two questions related to automation. First, could automation be used to reduce the cyber workforce manual efforts on a common intrusion type, and if so, what was the level of the reduction in workforce hours required to respond to these events. The second was the percent of time that the automation execution of externally shared indicators for intrusion events could be safely executed. Together these two experiments explored how automation could be used to close the gap in the cyber workforce.

The results of the initial experiments are shown in Figure 4. Only one type of intrusion was tested, a new executable file that the testbed had not previously been seen was dropped on a host. Within the operational infrastructure of the system, approximately 50,000 new files per day were found within the infrastructure. Of these files, operators, on a daily basis, only had time to examine 65 of the new files deemed to have the highest risk of being malware on the system.

Without IACD methods, the JHU/APL investigators observed that for each file on the system assigned to a Tier I analyst, that analyst spent between 10 minutes and 11 hours to determine whether the file represented malware. This was accomplished through the use of reputation services and products that executed the file to examine the result. At this point, the analyst could identify whether or not the new file was of concern and pass the information along to the response element of the team for mitigation.

Using the IACD capability, each file was evaluated between one second and 10 minutes to determine whether or not the file was known to be malicious.⁴ Once the file was determined to be malicious, the IACD automation prepared and executed a block of the communication between the malware and the Internet. A malware indicator for the malicious file was then produced and automatically shared so that other organizations represented in the test environment would be automatically protected prior to this attack promulgated to those downstream organizations.

The use of automation on the incoming new file indicators allowed for 3,500 to 14,000 of these daily new file events to be analyzed and subsequently blocked. While short of the 50,000 new file events received per day, this far exceeded the 65 events that could be manually processed earlier. The methods used to scale from 3,500 to a maximum of 50,000 new file events per day could be achieved through additional investments in the tools used to automatically execute and analyze the results of that execution. Within the lab environment, the

Operations Timeline Comparison

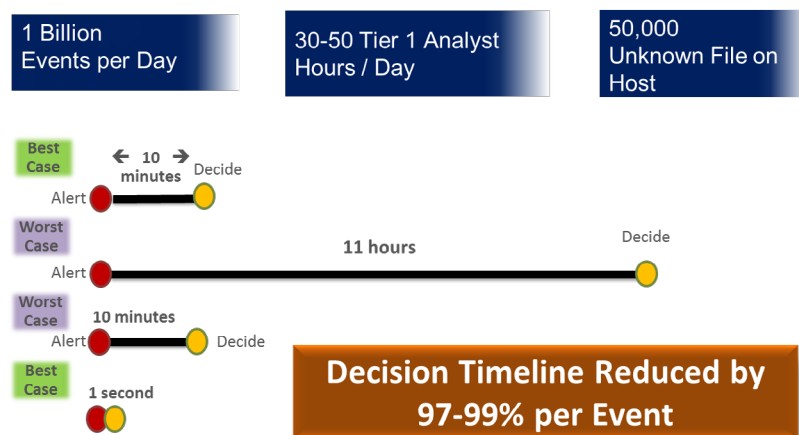


Figure 4: Automated Response Results

individual execution of a potentially malicious file was set for a maximum of ten minutes, so scaling to 50,000 files would take a maximum of 139 computational hours per day, easily achievable through 6 parallel processors. Note that this is a maximum number, since many of the files would likely be identified within a second through reputation services or would be the same as an already processed file.

For organizations that automatically execute responses based on the shared information from this initial attack vector, the malware was mitigated before any attack has actually been attempted with zero analysts' hours from that site involved in the protection. This sharing to trusted organizations is described below.

The available attack indicators were shared from simulated external organizations. These indicators provide the result of intrusion analysis from a source organization and include source IP domains and details on the activity that could be blocked to prevent attacks on the potential victim organization. Once received, these indicators can be compared to existing activity within an organization, and if the block does not significantly increase the risk of failure to an organization, the source site could be blocked.

One objection to automated response based on shared indicators has been that the percentage of intrusions that would in fact be mitigated though an automated response is small. To study this question, JHU/APL analyzed the shared indicators that are currently available across the defense industrial base. On average, JHU/APL receives approximately 135 indicators each day. Looking at this activity over one week (944 total indicators), the IACD lab would recommend automated responses for a total of 561 of the shared indicators based on the experiments described above. The automatic processing of these indicators included comparison of the potential blocked sites to an historical profile of accessed IP addresses. This processing time would take between 6 and 207 seconds with an average processing time per indicator of 50 seconds. As in the previous experiment, the indicators were independent and run in parallel, so the processing of indicators would scale with computing resources.

The analysis determined that of these 561 indicators, 416 could be automatically mitigated with no risk to the organization

based on the historical usage of the IP address associated with the indicator. As a result, almost 74% of the indicators recommended for automated response could have been automatically mitigated with no manual cyber workforce involvement.

In looking at the results as they relate to the cyber workforce, the malware that was managed automatically was the type that could be easily automated; yet, most organizations do not have the level of automation or integration to provide this automation. It is estimated that for this case 30-50 analyst hours per day were gained through this use of automation in both malware that entered as new files and from external indicators.

These initial findings showed a profound increase in workforce productivity, yet these tests should be viewed as early steps toward automation of the cyber industry. The results of this investigation demonstrated that automation is capable of substantially reducing the workload of cyber analysts with existing technologies. Automation enables routine problems to be addressed by machines and tools transferring the knowledge from an analyst to an automated course of action. Analyst productivity increases and also allows analysts to focus on more difficult and new problems. If the barriers to automation could be resolved, and automated cyber tools allowed to advance and take root broadly across the cyber ecosystem, the cumulative number of hours saved shows promise to overcome the cyberworkforce challenge.

Indicators Processed		
Today	Past Week	Average/Day
295	944	135
Indicators Processing Time (seconds)		
Average Time	Minimum	Maximum
50	6	207
Response Action Recommendations & Automation (Today)		
Total Recommended	Number Automated	% Approved for Automation
561	416	73.98%

Figure 5: Percentage of Automation

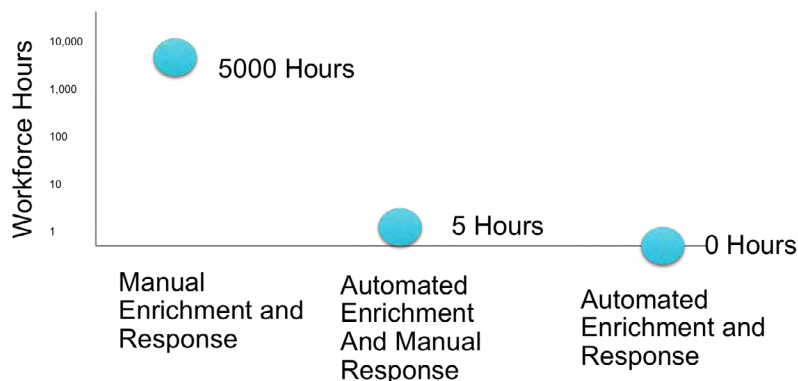


Figure 6: Workforce Reduction.

Result: Meeting the Need by Leveraging the Available Workforce

Training and education are initial steps to address the cyber workforce shortage, but as the threat landscape continues to grow and the cyber ecosystem expands, we will be forced to identify new ways to meet this challenge. Automation is one likely candidate. Automating responses in an integrated adaptive cyber defense environment has proven to be effective in the lab and could be implemented externally in real world systems across the cyber ecosystem. The investigation has shown that applying automation to the challenges faced by the cyber workforce will provide greater efficiencies and empower them with tools to improve intrusion identification, reduce response time, and begin to reduce the cyber workforce shortage across the public and private sectors.

Figure 6 shows a summary of the workforce hours gained through the use of automation in the APL experiment. The figure shows the overall reduction in work hours gained from automating the triage and response to a specific event: unknown file on an end user workstation. While this is only a single incident, fully responding to all indicators of this type of incident would require 5000 hours every day to enrich and respond to the incident. With the tested automation, this was reduced to 5 hours with enrichment alone, and reduced to zero hours with full automation. These hours can then be used for other tasks or to reduce the required workforce hours. This assumes a 0.01% malicious file rate from 50,000 source files/day and a one-hour response for each malicious file discovered.

This promising demonstration of IACD lab work showed that automation can provide significant advantages to help the struggling cyber workforce. It also serves as a jumping off point for further exploration of how automation may benefit cyber defense and the larger cyber ecosystem. A possible area of investigation might include associating the risk of an automated response to an outcome. Risk evaluation, a generally manual activity, absorbs considerable analyst time and frequently involves senior leadership to determine an appropriate risk acceptance level on systems and networks.

To enable the adoption of the described level of automation across the cyber ecosystem, existing information sharing standards (such as those available from NIST, the International Organization for Standardization, International Engineering Task Force, and the Object Management Group) can serve as a basis for orchestration products in tier 1 incident response. As the standards and open APIs available in security products mature, they should incorporate additional information essential to supporting automation, such as suggested course of action, context of the intrusion, and behavioral indicators linked to suggested responses. A focus for improving standards along the lines of supporting an automated workflow rather than just a syntax for exchanging information will support results of the type observed in the experiments described above.

Another opportunity to alleviate the stress and workload of the cyber workforce might include additional investigation of automated responses to system and network activity. This would build upon other elements that share intrusion indicators to help determine if an automated response can be taken with low risk.

Conclusion

Employing technology and automation to reduce workload and increase efficiency has been developed for centuries and has been vital to the economic success of our country. This new recent example merits continued investigation and support for implementation into systems and networks. The promise of benefit to the cyber workforce through automation provides a strong motivation to incorporate existing capabilities into systems and networks and to support and encourage the further development of cybersecurity automation.

REFERENCES

1. NIST Framework for Improving Critical Infrastructure Cybersecurity, Feb 2014
2. <<http://www.sans.org/media/critical-security-controls/CSC-5.pdf>>
3. <<http://www.counciloncybersecurity.org/workforce/>>
4. <[http://www.ey.com/Publication/vwLUAssets/EY-global-information-security-survey-2014/\\$FILE/EY-global-information-security-survey-2014.pdf](http://www.ey.com/Publication/vwLUAssets/EY-global-information-security-survey-2014/$FILE/EY-global-information-security-survey-2014.pdf)>
5. <<http://www.gao.gov/assets/670/668415.pdf>>
6. <<http://www.nist.gov/cyberframework/upload/cybersecurity-framework-021214.pdf>>
7. <<http://csrc.nist.gov/publications/drafts/800-53-rev4/sp800-53-rev4-ipd.pdf>>
8. <<http://csrc.nist.gov/nice/>>
9. January 2015 GAO report, "Federal Workforce, OPM and Agencies Need to Strengthen Efforts to Identify and Close Mission-Critical Skills Gaps (<<http://www.gao.gov/products/GAO-15-223>>),"
10. Ibid.
11. 2014 RAND report, "HACKER5 WANTED An Examination of the Cybersecurity Labor Market, (<http://www.rand.org/content/dam/rand/pubs/research_reports/RR400/RR430/RAND_RR430.pdf>)," "
12. Phyllis Schneek, Peter Fonash. Computer "Cybersecurity: From Months to Milliseconds" Volume 48 issue 1.

NOTES

1. The cyber ecosystem is global and includes government and private sector information infrastructure; the variety of interacting persons, processes, information, and communications technologies; and the conditions that influence their cybersecurity. (DHS, Blueprint for a Secure Cyber Future, November 2011, p. D-2.)
2. The Crosstalk article, "Challenges to a Trustworthy Cyber Ecosystem" addresses many of these growing problems. <<http://m.crosstalkonline.org/issues/3/12/>>
3. For more information on the IACD concept, visit <<https://secwww.jhuapl.edu/iacdcommunityday/>>
4. The difference in time depended on whether or not the new file was found in the reputation database or whether the file needed to be executed to determine if malicious behavior resulted. Automated execution of the file was set to a ten-minute threshold.

ABOUT THE AUTHORS



Dr. Peter Fonash is CTO for the Office of Cybersecurity and Communications in the National Protection and Programs Directorate (NPPD) within the US Department of Homeland Security (DHS). He is also an adjunct faculty member of the University of Tulsa's College of Engineering and Natural Sciences and an advisory board member of George Mason University's Volgenau School of Engineering. Fonash received an MS in engineering from the University of Pennsylvania, an MBA from the University of Pennsylvania Wharton School, and a PhD in information technology and engineering from George Mason University.



Dr. Tom Longstaff is the Principal Cyber Strategist for the Resilient Cyber Systems Branch of the Applied Physics Laboratory. Tom works with a wide variety of areas including technology transition of cyber R&D, information assurance, security architecture, intelligence, and global information networks. Tom's academic publications span topics such as malware analysis, information survivability, insider threat, and intrusion detection. Tom is Chair of the Computer Science, CyberSecurity, and Information Systems Engineering Programs at The JHU Whiting School of Engineering.

**CIVILIAN TALENT IS MISSION-CRITICAL.
LET'S GET TO WORK.**

Work for Naval Air Systems Command (NAVAIR) and you'll support our Sailors and Marines by delivering the technologies they need to complete their mission and return home safely. NAVAIR procures, develops, tests and supports Naval aircraft, weapons, and related systems. It's a brain trust comprised of scientists, engineers and business professionals working on the cutting edge of technology.

You don't have to join the military to protect our nation. Become a vital part of NAVAIR, and you'll have a career with endless opportunities. As a civilian employee you'll enjoy more freedom than you thought possible.

Discover more about NAVAIR. Go to www.navair.navy.mil.

Equal Opportunity Employer | U.S. Citizenship Required

**NAVAIR
CIVILIAN**

CHOICE IS YOURS.

People-driven Process-enabled Software Development: A 21st Century Imperative

Azad M. Madni, USC and Intelligent Systems Technology

Abstract. In the 21st century, software will continue to “grow” in a sociotechnical ecosystem comprising customers, end users, developers, maintainers, testers, and other stakeholders. Their continued participation is crucial to software acceptance both in the DOD and the commercial sector. In the recent past, software has been a process-driven product. However, with increasing software complexity, it is becoming apparent that the people aspect of software deserves greater attention and emphasis. The people aspect comprises people decisions, personnel skillset, training, motivation, creativity, and talent. This paper explores the shift from process-driven to people-driven, process-enabled software development. The key enablers to accomplish this shift are also discussed. This paper concludes with a reminder that while people are becoming increasingly important in software development, process will continue to be a key enabler.

Introduction

Software has been a process-driven product for the last few decades. This view has inadvertently de-emphasized the importance of people in the software lifecycle [1]. The reality today is that: People with appropriate training perform software-related activities, often subject to governing standards and legacy constraints within development environments to achieve desired outcomes. Today with ever-increasing software sophistication, human ingenuity is being challenged like never before. No longer does it suffice to just follow a disciplined development process because people are becoming increasingly crucial in performing trade-off analysis and in creating a satisfying user experience [1-2]. In addition, people are key to ensuring that software performance, quality attributes, schedule and cost objectives are being met. Exclusive focus on software process can potentially stifle human creativity and inhibit human contributions throughout the software lifecycle. Furthermore, as software continues to grow in complexity and humans continue to become more and more an integral part of software-based systems, predictable software behavior is becoming crucial to software system safety [3].

Today the proportion of software in systems continues to increase dramatically. This recognition has led to the creation of the term “software-intensive systems.” And people contribute in a variety of ways to software-intensive systems. For example, humans create new paradigms, explore the software design tradespace, discover patterns and trends, provide decision rationale, attempt to explain anomalous behavior, and assure smooth integration of people and software. Yet, the importance of people in the software lifecycle continues to be underemphasized. This is surprising in that software is largely a people creation that is maintained, supported, and adapted by people. People are also responsible for software quality, and yet scant attention is devoted to the talent, training, creativity and motivation of people responsible for assuring software quality [4]. Clearly, process will always play an important role, but more as an enabler than a driver. This paper argues that to achieve dramatic advances in software quality, the people dimension needs to become a central focus with process as an enabler. After all, software innovation is primarily the result of human creativity, passion and motivation. While process will continue to play an important role in the software life cycle and provide context for collaboration, the process perspective will be a necessary and valuable adjunct to the people perspective as software continues to increase in complexity [5]. People-driven software spans the 5 P’s: people, purpose, passion, patterns, perspectives, and processes. Table 1 presents the key elements underlying the shift in mindset from process-driven to people-driven, process-enabled software development.

There are several compelling reasons to make people the primary focus in software development today (Table 1). First, software is a creation of people, and quite frequently for the use of people. Exclusive focus on process can stifle creativity, and compromise user acceptance. Second, safety is becoming an increasingly important consideration in software-intensive systems. Safety subsumes predictable software behavior in the face of disruptive events [3]. It is important to note that processes do not automatically address safety concerns. It is people who introduce safety concerns in the software life cycle. Third, with the need for adaptive processes (e.g., agile), and the need for adaptable systems (to survive and operate in changing operational environments), the shift toward people-driven development is becoming inevitable [6-8]. Finally, with the advent of multi-domain software that cuts across multiple domains (e.g., electrical, optical, mechanical) and multiple disciplines (e.g., physics, social sciences, cognitive science), software complexity has increased dramatically. Collectively, these trends speak to the need for people-driven, process-enabled software development and use (Figure 1).

Figure 2 presents a notional graph illustrating the approximate relationships between process importance and software complexity, and between people importance and software complexity. As shown in this figure, as software complexity increases, software development becomes less and less process-driven, and more and more people-driven, albeit process-enabled. A key implication of this trend is that if the developing organization expects software to grow in scale and complexity, the organization is better off adopting people-driven, process-enabled software development practices [1,3,4,9].

Process-Driven	People-Driven, Process Enabled
Process flows	Technical stories
Process enforcement	Process guidance
Process prescription	Software patterns
Process integration	People collaboration
Process recipe	Human creativity/innovation
Disciplinary focus	Transdisciplinary perspective
Process knowledge	Human imagination
Process discipline	People passion

Table 1: From Process-Driven to People-Driven Process-enabled Development

⇒In the recent past, several developments have collectively pointed to a much needed shift from process-driven to people-driven software development. First and foremost, is the uncertainty about the operational environment, rate of maturation of promising technologies, and personnel turbulence resulting from retirements, layoffs and personnel moves. Second, software is becoming increasingly more complex because of ever-increasing scale, and ever-growing need for adaptability in light of the changing roles of humans in relation to software. These trends are being driven by the need for systems to be long-lived and capable of coping with unknown operational environments. Third, organizations are increasingly turning to adaptive processes such as agile development paradigm, which is increasingly being viewed as a source of competitive advantage when applied correctly. It requires an accomplished team of developers, effective leadership in pulling the team together, and a change in mindset associated with traditional process-driven development in which roles are important but individual people are viewed as interchangeable/substitutable parts, with people availability trumping people skillset [5,9,10].

Alistair Cockburn, in his book “Characterizing People as Non-linear, First Order Components in Software Development” argues that predictable processes require components with predictable behavior. And, people are anything but predictable. Treating humans as interchangeable components or replaceable parts in software development is a misjudgment. Human behavior tends to be variable and nonlinear. Humans exhibit an uncanny ability to succeed in novel ways, while also exhibiting a disconcerting capacity to fail in unimagined ways. It is the failure to account for these factors in software development that inevitably result in schedule and cost over-runs. In fact, it is fair to say that humans strongly figure in both project successes and failures [3].

Unfortunately, the mistaken belief that people are interchangeable resources is deeply ingrained in business thinking. It dates back to Frederick Taylor’s Scientific Management approach for performing repetitive tasks such as running a factory [11]. However, for highly creative work such as software development, this view is clearly inapplicable. And today, with the advent of smart manufacturing, manufacturing also no longer abides by this tenet. Another key tenet of Taylor’s theory is that the people doing the work are not best-suited to determining how best to do the work. While this tenet may hold, to a degree, on the factory floor, it is untrue of software development. In fact, people attracted to software engineering tend to be the best and the brightest, with the culture of youth pervading the field [3, 11].

So, what is it that people bring to software? People bring imagination, novel insights, storytelling ability, and an uncanny ability to discern and exploit patterns [2, 4]. These capabilities have the potential to transform software development in unprecedented ways to achieve dramatic improvement in software quality, responsiveness, cycle times, and life cycle costs. Some of the unique human capabilities that bear on software quality and costs are presented in Table 2.

A people-driven, process-enabled view of software goes well beyond the process perspective. It is sensitive to business concerns and constraints, implications of software-related decisions on short-term, mid-term, and long-term concerns of a program or business. It

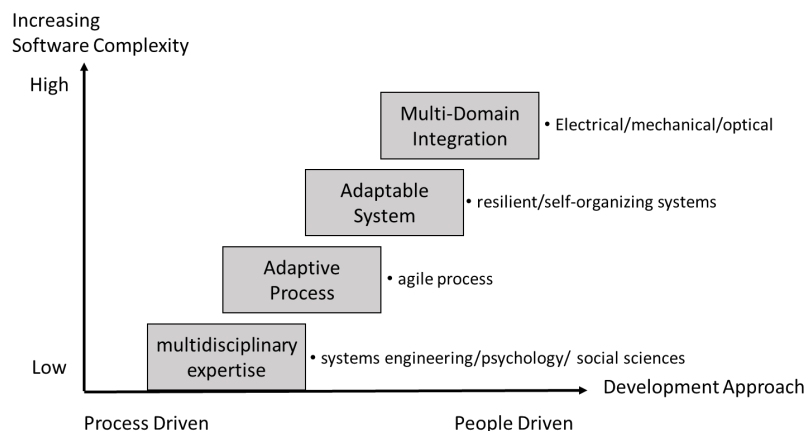


Figure 1: Developments Contributing to Increasing Software Complexity

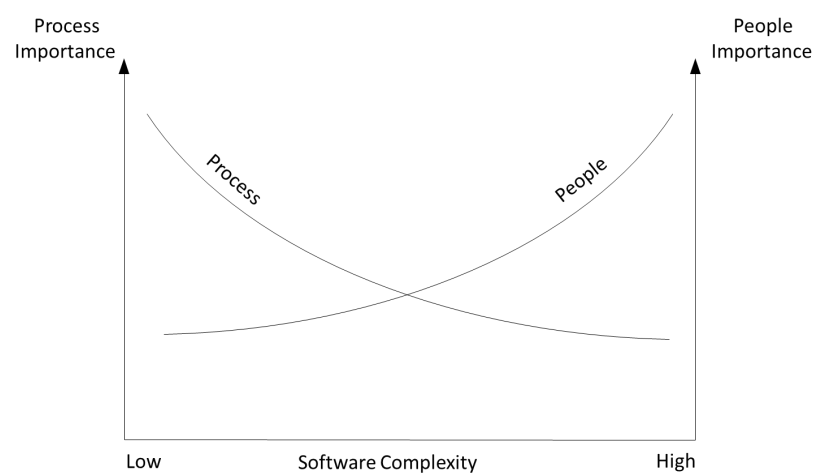


Figure 2: Increasing Software Complexity Driving Paradigm Shift

• Systems Thinking	Think holistically to understand “big picture,” relationships, and interdependencies
• Associative Thinking	Exploit metaphors and analogies to simplify software architectures, and circumvent constraints
• Storytelling	Engage all stakeholders in upfront software engineering to ensure their timely participation, contributions, and acceptance
• Visual Analysis	Discern patterns and trends that can be exploited in software simplification and implementation
• Abstractions	Abstract details to develop a mental representation that informs development of scalable and extensible software
• Tradeoff Analysis	Place right emphasis on conflicting objectives to create responsive software that meets stakeholder needs while satisfying schedule, budget, technical, and legacy constraints

Table 2. Unique Human Capabilities that Bear on Software Quality

is cognizant of the available skillset in both management and development teams. It shows understanding of programmatic and technical trade-offs, and the importance of collaboration and full stakeholder participation in the software lifecycle. The latter is essential for reasoned compromise that addresses stakeholders’ concerns and resolves issues. It is also essential for stakeholder acceptance of collaboratively made decisions, and elimination of extraneous design iterations and rework [1].

The people-driven view of software is especially sensitive to the required skillset and available expertise when it comes to the selection of the software development process (e.g., spiral, waterfall, evolutionary prototyping, incremental commitment) [3]. With a people perspective, software development process selection is not based just on problem particulars (i.e., objectives, schedule, budget,

risks) but also availability (or lack thereof) of the required talent and skillset in the development team [12]. The maturity and the experience of the team members and leadership play a pivotal role in defining use cases, specifying architecture, and developing the right set of abstractions.

From Process-Driven to People-Driven, Process-Enabled Development

People-driven development is more than stakeholders influencing and agreeing on what is being created. It is more than empowering engineering teams and the activities they perform to develop software. And it is more than directing software users in the use of software. It is in fact all of the above. People-driven development means humans playing an active role in software-related trade-offs, designing the software, managing the software development process, and even distributing software development activities to the development team members. People-driven development is also influenced by culture and power distance [13]. Compounding the problem is the “clash of values” between developers and program managers [14]. And, of course, human behavior exhibits nonlinearity and variability [2, 15]. These factors influence both the development process and the software product. Cockburn [15] and Madni [2, 4] identify specific factors that influence the outcome: humans are social beings who perform best in face-to-face collaboration; humans are inconsistent and inconsistency shows up over time; humans exhibit variability from day-to-day and place-to-place; humans generally want to do the right thing for their organizations.

These characteristics bear directly on process. It is important to recognize that process enforcement can vary from strict to loose. In light of human characteristics and ever-growing system complexity, loose process enforcement is preferable to strict enforcement. In cases, where strict process enforcement is required, there is a need for performance support for humans to behave consistently.

Software lifecycle processes provide a structured disciplined means to guide the development of complex, real world software [16]. This software spans: primary processes (acquisition, supply, development, operation, maintenance); supporting processes (documentation, configuration management, quality assurance, reviews and audits, problem resolution); and organizational processes (management, infrastructure, maintenance, improvement, training). The question that needs to be asked is in where do lifecycle processes benefit software design and where do they become an impediment. For most support and organizational processes, following software life cycle process is a benefit. Also, periodic architecture and design reviews help to ensure design quality, and traceability between requirements and design elements to ensure design completeness. However, there are times where strict process enforcement becomes a hindrance to creativity and innovation [17]. In this case, humans can “dial back” on strict process enforcement and adopt loose process enforcement. This shift puts people in charge of the process, making it people-driven, process enabled software. This recognition is at the heart of adaptive software development in general, and agile development in particular.

Agile processes (or agile, for short) are a prime example of people-driven, adaptive development. Agile relies on process acceptance by the development team, not process imposition by management [6-8, 12]. In other words, only developers themselves can choose to follow an adaptive process. This is especially true of extreme programming (XP), which requires disciplined execution, with developers making all the decisions and generating all time estimates. This is a huge cultural shift

for management in that it requires sharing of responsibility between developers and management [12].

Measuring software productivity is a challenge with adaptive processes. In this regard, Robert Austin distinguishes between measurement-based and delegatory management in software development. Measurement-based management is best suited to repetitive work with minimal knowledge requirements and easily measured outputs. For software development, the delegatory style of management is appropriate. Delegatory management calls for developers to decide how to do the work. In fact, this approach is central to the agile philosophy. This does not mean that developers have to do it all. In fact, developers rely on management for guidance when it comes to business needs. Finally, in adaptive development, change is an expected and frequent occurrence. Consequently, people need to be kept apprised as they continually adapt the process to fit changing needs [18].

Recent Trends

Several recent developments make a people-driven view of software both attractive and eminently viable. Three of the more compelling advances that bear on a people-driven view are: Model-Based Engineering, Experiential Design and Visual Analytics; and Interactive Technical Storytelling in Virtual Worlds [1,9]. Each is discussed next.

Model Based Engineering transforms traditional approaches in a number of ways. First, it replaces document-centric engineering with software models at the center of the development process. The model serves as the sole source of truth, from which documents can be created on demand. Second, model-based software engineering assures consistency among the different perspectives embodied in the model. Third, model-based software engineering can provide different lenses for different stakeholders allowing them to explore the consequences of changes in assumptions, constraints, and resource/data availability.

Experiential Design and Visual Analytics is the combination and use of context-sensitive visualization interfaces and analytical reasoning methods to enable visual debugging, simplification, and redesign of both systems and processes. As importantly, visual analytics appeals to all stakeholders because it transforms calculation results into easy-to-assimilate visuals, patterns and trends [9].

Interactive Technical Storytelling in Virtual Worlds is a means to engage all stakeholders in upfront engineering to ensure that the inputs and concerns of all stakeholders are known and addressed when conducting trade-off analysis [1]. By providing each stakeholder with an appropriate “lens” into story execution, meaningful inputs and concerns from all stakeholders can be elicited and resolved through timely, multi-stakeholder collaboration [9]. Virtual Worlds are simulated environments within which stories unfold and stakeholders explore the consequence of “what-if” assumptions, decisions and tradeoffs. By providing appropriate “lenses” for the different stakeholders, instrumented virtual world offers a convenient means for knowledge acquisition, data collection, and uncovering surprising behaviors.

What it Takes

To realize this shift in mindset, requires advances on several fronts: a) a persuasive value proposition of people-driven development; b) demonstration of how people-driven software delivers superior value than process-driven software; and c) a risk-mitigated, staged process to gradually make the transition from process-driven to people-driven development.

Value Proposition: Software development is a collaborative process that involves multiple stakeholders who need to jointly explore tradeoffs and reach consensus. Expanding stakeholder participation is critical. An experiential, stakeholder-oriented interface [9] is the key to ensuring full stakeholder participation early and throughout software development.

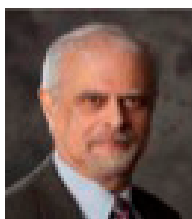
Demonstration of the value proposition: The demonstration should highlight the key elements of a people-centric approach to developing high quality software. The approach should highlight: an experiential interface with "lenses" for various stakeholders; an interactive storytelling capability to engage the various stakeholders from their respective perspectives; an instrumented virtual world that supports story execution and that can be collaboratively and individually explored by the different stakeholders under a variety of "what-if" assumptions, parameter values, and technical and programmatic tradeoffs [9].

Staged transition: The transition from a process-driven view of software development to a people-driven, process-enabled view has to be accomplished in stages. It is a cultural change for both management and developers. In this regard, the first stage is the transition from traditional use cases to stories. The second stage is the introduction of storytelling in virtual worlds with stakeholder-oriented "lenses" that allow stakeholders to explore the software tradespace and understand the CONOPS when the stories execute in the virtual world. The third stage is story-enabled collaborative trade-off studies supported by sensitivity analysis and comparative evaluation.

Conclusion

Software has been a process-driven discipline for quite some time. While "process" will continue to be a key enabler of software development, the people aspect will continue to gain in importance as software continues to grow in scale and complexity requiring greater human involvement. Surprisingly, the growing importance of people in software development has not produced a paradigm shift in software development. And yet it is people that bring ingenuity, imagination, and creativity that can dramatically improve software quality and development efficiency and effectiveness. This paper emphasizes the importance of people-driven, process-enabled software development. Additionally, in light of growing emphases on people, four significant advances are identified as key enablers of this transformation: model-driven engineering, experiential interfaces, visual analytics, and interactive storytelling in virtual worlds. Looking down the line, as various relevant technologies mature, software quality and development will increasingly depend on the "people factor," with process continuing to be an important enabler, but not the sole driver.

ABOUT THE AUTHOR



Dr. Azad Madni is a Professor and Director of the Systems Architecting and Engineering Program in the Viterbi School of Engineering at the University of Southern California. He is also the founder and Chief Scientist of Intelligent Systems Technology, Inc. He received his B.S., M.S., and Ph.D. degrees in engineering from UCLA. His government research sponsors include DOD, DARPA, AFRL, AFOSR, ONR, NAVAIR, NAVSEA, SPAWAR, ARL, ARI, RDECOM, DHS S&T, DTRA, NIST, DoE, and NASA. His research has also been sponsored by commercial companies including Boeing, Northrop Grumman, Raytheon, Hughes, Orincon, and General Motors. He is an elected Fellow of AAAS, AIAA, IEEE, INCOSE, SDPS, and IETE. He is listed in the major Marquis' Who's Who, including Who's Who in America.

Phone: 213-740-9211

E-mail: azad.madni@usc.edu

REFERENCES

1. Madni, A.M. "Expanding Stakeholder Participation in Upfront Systems Engineering", *System Engineering*, 2015
2. Madni, A.M. "Integrating Humans With and Within Software and Systems: Challenges and Opportunities," (Invited Paper) *CrossTalk, The Journal of Defense Software Engineering*, May/June 2011, "People Solutions."
3. Cockburn, A. "People and Methodologies in Software Development," Ph.D. Dissertation, University of Oslo Press, University of Oslo, February 2003.
4. Madni, A.M. "Integrating Humans with Software and Systems: Technical Challenges and a Research Agenda," *Systems Engineering*, Vol. 13, No. 3, pp. 232-245, Autumn (Fall) 2010.
5. Madni, A.M. "Thriving on Change through Process Support: The Evolution of the ProcessEdge™ Enterprise Suite and TeamEdge™," *International Journal on Information - Knowledge - Systems Management*, Special Issue Vol. 2, No. 1, pp. 7-32, 2000.
6. Cockburn, A. "Agile Software Development: The Cooperative Game, 2nd Edition, Addison-Wesley Professional, October 2006.
7. Madni, Azad M. "AgileTecting™: A principled approach to introducing agility in systems engineering and product development enterprises." *JIDPS 12.4 (2008): 49-55*.
8. Madni, A. M. "Agile systems architecting: Placing agility where it counts." *Conference on Systems Engineering Research (CSER)*, 2008.
9. Madni, A.M., Spraragen, M., and Madni, C.C., "Exploring and Assessing Complex System Behavior through Model-Driven Storytelling," *IEEE Systems, Man and Cybernetics International Conference*, special session "Frontiers of Model Based Systems Engineering", San Diego, CA, Oct 5-8, 2014.
10. Austin, R.D. *Measuring and Managing Performance in Organizations*, Dorsett House Publishing, 1996.
11. Taylor, F.W. "The principles of Scientific Management, New York and London, Harper and Brothers, 1911.
12. Taylor, F. W. "Shop Management," New York and London, Harper and Brothers, 1903
13. Geert, H., and Hofstede, G. "Cultures and organizations: Software of the mind." McGraw-Hill, New York (1991).
14. Cockburn, A. "Software development as Community Poetry Writing. Cognitive, cultural, sociological, human aspects of software development." *Annual Meeting of the Central Ohio Chapter of the ACM*, 1997.
15. Cockburn, A. "Characterizing people as non-linear, first-order components in software development" *International Conference on Software Engineering 2000*. 1999.
16. Suryanarayana, G., Sharma, T., Samarthayam, G. "Software Process versus Design Quality: Tug of War", *IEEE Computing Edge*, Aug 2015.
17. Madni, C.C., and Madni, A.M., "Web-enabled collaborative design process management: application to multiphase module design." *Systems, Man, and Cybernetics*, 1998. 1998 *IEEE International Conference on*. Vol. 3. IEEE, 1998.
18. Bramble, P. "Patterns for Effective Use Cases, Addison-Wesley Professional, August 2002

Developer Training: Recognizing the Problems and Closing the Gaps

Mike Lyman, Cigital

Abstract. Problems with the way we have historically trained developers, and continue to do so, gets in the way of learning to do secure development. The same errors are repeated over and over again. Without learning the fundamentals, developers can get software to work but may lack the knowledge required to understand what is happening when it breaks. Since the goal of an attacker is to get software to break and do things the developer did not intend, that lack of understanding can become a dangerous blind spot.

1. Introduction

If the aviation industry operated like the software industry, the new Boeing 787 Dreamliner would have to crash several times before Boeing could learn the lessons we learned with the Comet back in the 1950s. The de Havilland Comet was the first production commercial jetliner and it suffered a series of crashes early in its service life. During the investigations into the crashes, it was discovered that the hull experienced metal fatigue as the cabin was repeatedly and rapidly pressurized and depressurized and encountered rapid temperature change while changing altitude. In some areas of the structure there were special stress points that experienced more problems than others. One of these special stress points existed in the corners of the square-cornered windows on the original versions of the Comet. The square corners caused levels of stress two to three times greater than the rest of the fuselage. The metal would fail after a number of flight cycles with one of the crashes taking place in as few as 900 flights.

Thank goodness the aviation industry doesn't operate like the software industry and the lessons learned are passed on for all. We now design airliners with an understanding of metal fatigue and the windows on airliners are now designed to have rounded corners. The Dreamliner will not suffer from problems the industry learned to fix with the Comet decades ago.

Sadly, the software industry continues to use square-cornered windows. Too many cannot seem to learn the lessons of failure others have already learned and repeat the same errors over and over again. There is clearly something wrong with the way we train developers.

2. Recognizing the Problems

One of the first things we need to realize is counterintuitive. We have to start by recognizing that we have to overcome our training. Problems with the way we have historically trained developers, and continue to do so, get in the way of learning to do secure development. Before we can address our gaps, we have to address our problems.

Lack of Formal Training

There is an old joke that tells us to remember that fifty percent of all doctors graduated in the bottom half of their class. The software industry suffers the same reality with the added burden that many of our developers have no formal, college level development training. Many of our developers came at the field from other disciplines and discovered they have had the skills that it takes to write code and make software work. Not only did half of our developers graduate at the bottom of their class, many never even took the class!

The fact that so many of us are self-taught means that many of us lack the fundamentals underpinning the way our software and computers work. These fundamentals help developers understand things that are going on under the hood when we talk about issues like buffer overflows or integer overflows. Without those fundamentals, developers can still get software to work but may lack the knowledge required to understand what is happening when it breaks. Since the goal of an attacker is to get software to break and do things the developer did not intend, that lack of understanding can become a dangerous blind spot.

Bad Habits from Instructors

Even when we do have formal instruction it can cause us problems. When I was a computer science student back in the 1980s, I remember one of my professors telling us to "forget that extra stuff, just concentrate on the lesson." That "extra stuff" included things like error checking and limiting user input that my partner and I were adding into our program because we had already accomplished the lesson and had time to do the "extra stuff." Translated, he told us to just get it to work and move on. Lesson learned.

Current computer science majors tell me their instructors are still telling them to do this. In the secure code reviews I have done over the last eight years, I see way too much of the "just get it to work and move on" in the code I have reviewed. While not an intended lesson, it is a lesson way too many developers learned and took to heart. Get it to work and move on.

Another bad habit can best be summarized by a conversation with a friend who was a computer science professor for a major university when we bumped into each other at a security conference. He said he once had a chance to see the code from a major commercial product and that it looked like the "junk" his students wrote. I pointed out his students probably did write it. It was a humorous moment but like so much humor, there is a painful truth behind it. How are developers supposed to move on from the "junk" we write as college students to being able to write quality code?

Narrow Focus of Lessons

To be fair though, these are issues with the nature of instruction. Lessons tend to be narrowly focused because lessons with too many topics overwhelm students. Early on in their development training, students lack the foundations to understand the bigger picture so it is hard to get them to understand the lesson in a larger context. Additional topics make grading harder for the instructors; encouraging students to reach beyond the narrow lesson just adds to the instructor's work load. The added grading time may increase the chance that the instructors begin to just see if it runs and then move on which reinforces the unfortunate "get it to work and move on" lesson.

“ The fundamentals remain the same but the technology is moving so fast that books have to be based on pre-release versions in order to be available when the technology is released. Even if we use the latest materials, they are often out of date when we use them. ”

Do the Professor and Trainers Even Know About Secure Development?

Asking professors to look beyond the narrow lessons and to include the security aspects of things assumes they understand the security implications themselves. This is a big assumption. Many of us were never taught the security side of development as we learned to develop software; it is a stretch to then expect us to be able to now turn around and teach it. It does little good to complain that instructors are not teaching secure development if we do not first train the trainers.

Out-of-date Training

Another major problem we have to overcome is our training is usually out of date. The fundamentals remain the same but the technology is moving so fast that books have to be based on pre-release versions in order to be available when the technology is released. Even if we use the latest materials, they are often out of date when we use them. By the time a lot of developers use the training, it is even more out of date than it was when they learned it. The technology stacks we are using and the frameworks we build upon are changing so fast that it is difficult to keep up, especially when you are busy working.

Learn it Quickly and Then Use It

Because most developers are busy working, there is a huge incentive to learn just enough to get something working and then move on; that old habit we learned from our professors when we got started. Learn a new language fast and start coding. The updated frameworks have new features developers need and they need them fast because the next release is just around the corner. Follow the examples showing how to use the new features, get it working and move on to the next thing.

Insecure and Incomplete Examples

The problem with this approach is our examples have problems. Like all lessons, they tend to be narrowly focused to teach what is needed to be taught so they often show stripped down code that leaves off a lot of things we need for quality code. Unfortunately, those stripped down examples find their way into production code.

Beyond the narrow lessons, all too often the examples showed us how to do things exactly wrong.

One of the most glaring examples is how many developers learned to do database driven applications. The books and articles first showed us to connect to databases using connection strings like these:

```
strConnection = "SERVER=db.example.com; DRIVER={SQL
Server}; DATABASE=northwind; uid=sa; pwd=;"
```

Connect to the database server with the sa account (the system admin account) and use a blank password. Why? All too often the answer from a developer was "I don't know. But the book says do it that way so we're going to do it that way." No least privilege and a very weak password.

We were also shown to do database queries exactly wrong. We were taught to create database queries by creating a stub of a query and using string concatenation to insert the dynamic values we needed to tailor the query for the specific need at runtime. Examples looked like this:

```
mySQL = "SELECT * FROM tblBooks WHERE title like '%" &
txtUserInput & "%'";
```

Most often, this dynamic content came from the user without any examples of input validation. This is exactly how to create a SQL Injection attack and it is what we were taught to do by the examples we were following. Combine these examples with the example connection strings that the books showed us to use and the attacker could do anything they wanted to in the database, the other databases on the server or to the server itself. Thank you insecure examples.

3. Overcoming the Problems

Before we can begin to address the gaps in our development training that lead to security issues, we have to address the problems our historical approaches to training have caused.

Stop Teaching Bad Habits

One of the first things we need to do to overcome these problems is to stop teaching the students bad habits. When students move beyond the lesson and start adding in "extras" like error checking and limited user input, do not discourage them. This may mean instructors have to look deeper to grade the actual lesson but doing so saves employers from having to overcome the "get it to work and move on" lesson so many students have taken to heart. It would be beneficial to actually encourage the students to do more than just get it to work, especially when they have moved beyond the basics and on to more advanced lessons.

We also need to stop letting students write junk code. They need to use meaningful variable names, properly document their code and follow standards. The code needs to be easy to read. Many developers do not learn these lessons until they have to maintain junk code and feel that pain. Early lessons that exist only to force them to do maintenance on badly written code and feel that pain will help encourage them to write better code. As an added benefit, the better code will be easier to grade for the instructors.

“ We have been doing it wrong for a long time. The tragedy isn't that we cannot seem to learn the lessons of history but that we have learned the lessons and all too often we have failed to pass them on. ”

Remind Students There is More than the Narrow Lesson

While there may be no way to get away from narrowly focused lessons, we should constantly be reminding students that the lessons do have a narrow focus and in the real world there is more you must do. Instructors should show the lessons applied in a larger context. Periodic summary projects where several narrow lessons are wrapped into a larger project specifically focused on the bigger pictures will reinforce the concept. Part of this bigger picture must include the security implication of what the students are creating.

Train the Trainers

Unfortunately, to inject the security picture into our lessons, the instructors have to know that security picture. Like so many of us, they probably never learned the security implications of what they are doing. It is critical that we train the trainers so they know secure development. If they do not start to teach new developers how to write secure code from the beginning, we will always be playing catch up. We will have to teach them not to use square-cornered windows while they are on the job. We need to teach the teachers so they can properly teach their students.

Purge the Bad Examples

We must purge bad examples from our lessons. We learn from examples and using insecure and badly written examples creates bad habits that have to be unlearned later. The development training publishing houses are doing a much better job of this today than they used to, but they need to remain vigilant. They need security focused reviewers going over code examples just like we need security focused reviews of real code. When we are the instructors we need to be careful about the examples we use. We need to go over our lesson plans, especially old ones, and purge the insecure examples.

Students Have to Do Their Part

All of this cannot just fall on the trainers and producers of training materials. Students, especially professionals learning new technologies, must do their part. They have to remain aware that lessons are deliberately narrow and remember there is a bigger picture. We have to remember that while the instructors are only grading the lesson they taught, there are a lot of other important things we have to do with our code. We have to be

aware that examples we follow are also narrowly focused and be diligent about making sure we learn what else we need to be taking care of in our code. We have to be aware that the examples we follow may not be the most secure way of doing things, especially when the examples are old and out of date. We have to be active students when it comes to security.

4. Closing the Gaps – Learning from History and Today

Once we begin to overcome our training, we can begin to close the gaps that lead to security problems.

The most fundamental gap is that developer training ignores a wealth of history of software failures. We have suffered from decades of buffer overflows and race conditions, input validation failures and injection attacks. We have created software with unnecessary features that cause security issues enabled by default rather than being an optional feature users have to enable. We have been doing it wrong for a long time. The tragedy isn't that we cannot seem to learn the lessons of history but that we have learned the lessons and all too often we have failed to pass them on. The bigger tragedy is those lessons, when we do pass them on, are reserved for “secure development” classes instead of being incorporated into development classes in general.

It would not be difficult. Inject the lessons from failure right along with the actual lessons. We are all used to sidebars in our books. While the main lessons teach new developers how to write code without the issues that cause us problems, the sidebars can tell the stories of how we once did it wrong. They can tell the stories of the money it cost companies or the lives lost. They can link to the historic issues in the Common Weakness Enumerations and Common Vulnerabilities and Exposures databases that MITRE maintains. Show the developers that bad code has consequences beyond a bad grade in class. Rather than teach students to do input and then teach validation at a much later date, teach them to accept limited input that is immediately validated and move on to how to accept less restrictive input at a later date. When teaching numeric types, also teach how computer treat numbers differently than humans do. Have lessons early on that deliberately show the impact of numeric overflows. When having to pass commands to other command processors, teach students to use parameterization mechanisms so the receiving end can easily tell what the command the developer specified was and what is input into that command from other sources. Show them what happens when they do not do it that way. And all along, have sidebars talking about real examples of what happened when we got it wrong in the past.

We've got to get past teaching developers this is the way to do something and later teaching them the secure way to do it. Let's just teach them the right way from the beginning and never let them learn the insecure way.

Even when we learn the right way up front, developers will still have to learn about software failures that occur today. These may be repeats of the lessons from history or they may be new lessons as attackers continue to become more creative in attacking our software. Often, this can come from continuing

ABOUT THE AUTHOR



Mike Lyman is a senior security consultant at Cigital. His areas of expertise include secure code review, vulnerability assessments and training developers in secure development. Mike spent 12 years with SAIC helping create their software assurance offering for DoD customers at Redstone Arsenal, AL; pioneering most of the processes and procedures used by the practice. He has been a CSSLP since 2008 and a CISSP since 2002.

21351 Ridgetop Circle, Suite 400

Dulles, VA 20166-6503

Phone: (800) 824-0022

Fax: (703) 404-9295

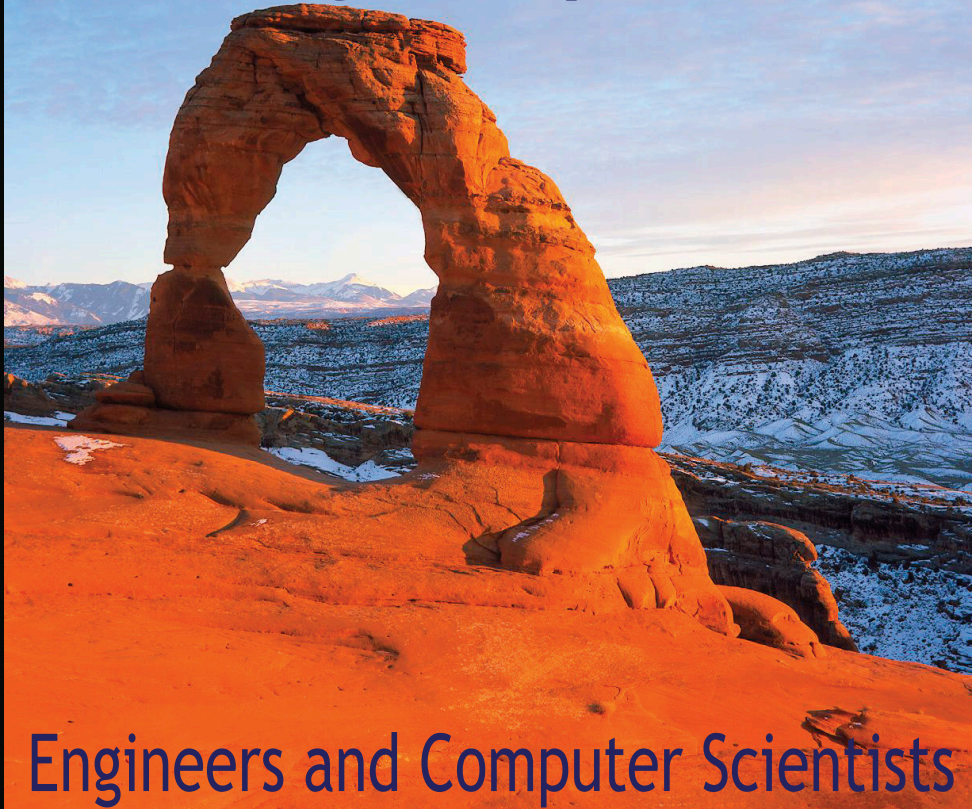
E-mail: communications@cigital.com

secure development training as we continue our education. Some of this needs to come during our code reviews as we cover the problems in our code. We need to deploy lightweight static analysis tools to the developer workstations that can catch mistakes as we create them, similar to the way spell checkers work today, and then the tools teach them the right way to do things. The instructors, code reviewers and tools must stay up to date with the latest trends.

Within organizations, trends in their own code should be shared organization wide, especially when there is a significant failure. Imagine the airline industry if the lesson learned from the Comet had not been shared. If organizations are brave enough, they can share the lessons learned with those outside the organization similar to the way Microsoft's SDL blog did on occasion after a patch Tuesday. Share the lesson learned. The wider the lessons are shared, the better for all of us.

Other engineering disciplines have successfully merged learning from failures into their basic education and continuing to learn from new failures in their continuing education. They no longer make the same mistakes over and over like the software industry does. Because of this, we no longer have to worry about boarding a new airliner with square windows feeding metal fatigue problems in the fuselage. Wouldn't it be nice if the developers of our shiny new software had also learned the lessons of history and did not recreate problems we learned to avoid long ago?

Hiring Expertise



Engineers and Computer Scientists

The Software Maintenance Group at Hill Air Force Base is recruiting **civilians** (U.S. Citizenship Required). Benefits include paid vacation, health care plans, matching retirement fund, tuition assistance, paid time for fitness activities, and workforce stability with 150 positions added each year over the last 5 years.

Become part of the best and brightest!

Hill Air Force Base is located close to the Wasatch and Uinta mountains with skiing, hiking, biking, boating, golfing, and many other recreational activities just a few minutes away.



Send resumes to:

309SMXG.Recruiting@us.af.mil
or call (801) 777-9828



www.facebook.com/309SoftwareMaintenanceGroup

Building and Operating a Software Security Awareness and Role-Based Training Program

Mark S. Merkow, CISSP, CISM, CSSLP, Charles Schwab Co., Inc.

Abstract. Today's application software developers are encouraged (and rewarded) by developing lots of code quickly but are usually not provided the adequate time and skills to build it with security and quality as top of mind requirements. Traditional education that prepares programmers for new technologies, new languages, and new platforms don't arm learners with the skills they need to meet the demands of organizations that require resilient and high-quality applications that can be constructed quickly at acceptable costs. It falls then on each organization to fill these gaps and help learners to 'un-learn' bad habits and form new good habits as they fulfill their duties.

The Three Pillars of Secure Software Development

Software security can only be achieved once three elements are in place and operating effectively – Policies/Standards, Education, and Assessment, as shown in Figure 1 below.

Standards and policies are mandated to document the security requirements on development projects, process documentation on how software should be developed, and the specific security standards for the security activities within the secure software development life cycle. Education then builds on these standards and other documents using both awareness techniques and education with technical courses arranged intelligently. Assessments then provide the means to test, verify, and measure all related activities. Assessments include evaluation of the secure development processes, testing the applications themselves, and assessing the overall program to provide a continuous improvement feedback model.

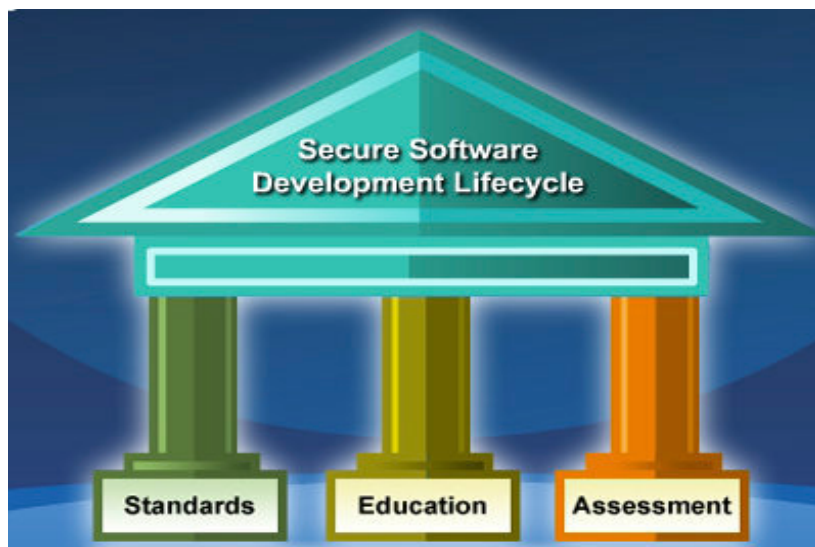


Figure 1. The Pillars of a Secure Development Lifecycle [1]

Education is the middle pillar of a secure SDLC for a very good reason – Education Provides Context.

Context Is Key

Without proper context, any mandates for high-quality and secure applications won't get communicated effectively to those who need to know. It's of little use to run around with your hair on fire shouting that applications are vulnerable to Cross Site Scripting, SQL Injections, Buffer Overruns, and so forth if the people you're screaming at have little clue as to what they're hearing and even fewer skills or know-how to do something about it. To this end, while prevention is always better than remediating problems and rework, programmers are typically faced with learning their applications are insecure long after they're released them to Production and to the malicious users who are pervasive throughout the Internet.

While Secure Software Development is only one of the topics within an overall Information Security Awareness and Training Program, varied levels of awareness and training are needed to get through to the right people in their various roles within the SDLC. An effective program for building the right level of detail for each group of stakeholders uses a layering approach that builds on foundational concepts that are relevant and timely for each role in each phase.

Principles for Software Security Education

The following are some basic principles as consideration for what should be included or addressed when setting up an Education Program:

Executive Management sets the mandate. With management mandates for secure application development that are widely communicated, you're given the appropriate license-to-drive a program from inception forward to continuous improvement. Software security programs that start out as a grassroots initiative typically fail to 'catch fire' across the organization. You'll need this executive support for establishing a program, acquiring an adequate budget and staff, and keeping the program going in the face of setbacks or delays.

Awareness and Training must be rooted in company goals, policies, and standards for software security. Establishing, then using documented organizational goals, policies, and controls for secure application development as the basis for your awareness and training program creates a strong connection to developer actions that lead to compliance and Defense in Depth brought to life.

Learning media must be flexible and be tailored to the specific roles within your SDLC. Not everyone can attend an in-person instructor-led course so alternatives should be provided, like Computer-Based Training, recorded live presentations, and so forth.

Learning should happen as close as possible to point where it's needed. A lengthy course that covers a laundry list of problems and solutions won't be useful when a specific issue crops up and the learner can't readily access whatever was mentioned related to the issue.

Learning and practicing go hand-in-hand. As people personally experience the 'how to' of learning new skills, the better the questions they ask, and the quicker the knowledge becomes a regular practice.

Use examples from your own environment. The best examples of security problems come from your applications. When people see issues with code and systems they're already familiar with, the consequences of exploiting the code's vulnerabilities hit close to home, and become more real and less theoretical. Furthermore, demonstrating where these examples stray from internal standards for secure software helps people make the connection between what they should be doing vs. what they've been doing.

Add Learning Milestones into your training and education program. People are less motivated to learn and retain discrete topics and information if learning is treated as a "check box" activity. People want milestones in their training efforts that show progress and help them to gain recognition and demonstrate progress. As you prepare a learning curriculum for your various development roles, build in a way to recognize people as they successfully advance through the courses and make sure everyone knows about it.

Make your program company-culture relevant. Find an icon or internally well-known symbols in your organization that resonate with employees and incorporate them in your program or build your program around them.

BOLO – Be On The Lookout for people who participate in your awareness and training program who seem more enthusiastic or engaged than others. These people are your candidates for becoming Internal Application Security Evangelists or Application Security Champions. People love thought leaders, especially when they're local, and you can harness their enthusiasm and interest to help you advance your program and your cause.

Keeping track of program maturity keeps people striving for improvements. There are a number of approaches to measuring the maturity of your software security program (Application Security Maturity Model (ASM) [2], Building Security In Maturity Model (BSIMM) [3], Software Assurance Maturity Model (OpenS-AMM) [4], and others. It makes no difference which model you use provided you use it consistently. Some organizations have discovered ways of using these models to compare internal software development organizations and use it as a yardstick for some friendly competition and goal-setting. Other measurement tools (described later) are useful for determining the competency of the individual development staff members themselves to make sure they've acquired the right skills and proven their ability to apply those skills specific to your organization's needs and goals.

Getting People's Attention

In general, software security is not the most exciting topic or engaging topic around – although it can be! Apathy is rampant and too many conflicting messages from prior attempts at software security awareness typically causes people's eyes to glaze over, which leads to even more apathy and disconnection from the conversation.

Peter Sandman, who operates a Risk Communication practice, has identified a strategy for communication that's most appropriate for software security awareness as well as other issues where apathy reigns but the hazards are serious (e.g. Radon poisoning, employee safety, etc.) The strategy,

called "Precaution Advocacy" is geared to motivating people by overcoming boredom with the topic [5]. Precaution Advocacy is used on high-hazard low-outrage situations in Sandman's Outrage Management Model. The advocacy approach arouses some healthy outrage and uses this attention into mobilizing people to take precautions or demand precautions.

Software security is a perfect issue where it's difficult to overcome apathy and disinformation and motivate people to address the issues that only people can address and solve.

Precaution Advocacy suggests using 4 ways to getting people to listen – then learn:

Learning Without Involvement - The average television viewer pays little attention to the commercials, but nevertheless knows dozens of advertising jingles by heart. Repetition is the key here. Posters, closed-circuit TV segments, mouse pads, elevator wraps, etc., are some useful examples.

Make Your Campaign Interesting/Entertaining - If you can arouse people's interest or entertain them, you'll get their attention and eventually you won't need so much repetition. It's best if you can make your awareness efforts impart interesting or entertaining messages often and liberally.

Need to Know – Whetting people's appetite to learn encourages the learner to seek information, and it's easy to deliver information to people who are actively seeking it. Sandman advises developers of awareness programs to focus less on delivering the information, and more on motivating their audience to want to receive it. Empowering people helps you educate them. The more people understand that insecure software is a software engineering, human-based problem (not a network security problem), the more they'll want to learn how best to prevent these problems. Making software security a personal issue for those who can effect improvements, then giving them the tools and skills to use will make them more valuable team members and leads to better secured application software.

Ammunition - Psychologist Leon Festinger's "theory of cognitive dissonance" argues that a great deal of learning is motivated by the search for ammunition to reduce the discomfort (or "dissonance") that people feel when they have done something or decided something they're not confident is wise [6]. Overcoming cognitive dissonance is a vital step early in your awareness program, so people experience your information as supportive of their new behavior, rather than as hostile to their old behavior. People also need ammunition in their arguments with others. If others already believe that software security is a hazardous community-wide issue – with no cognitive dissonance – they won't need to pay so much attention to the arguments for doing it.

The intent is not to frighten people or lead them to believe the sky is falling, but to motivate people into changing their behavior in positive ways that improve software security and contribute to the organization's goals and success. As your program progresses, metrics can show how improvements in one area lead to reduced costs in other areas, simpler and less frequent bug fixing, improved pride of code ownership, and eventually best practices and reusable code components that are widely shared within the development community.

Awareness vs. Education

Security awareness is the effective sharing of knowledge about potential and actual threats and an ability to anticipate and communicate what types of security threats that developers face day after day. Security awareness and security training are designed to modify any employee behavior that endangers or threatens the security of the organization's information systems and data.

Beginning with an awareness campaign that's culturally-sensitive, interesting, entertaining, memorable, and engaging gives you the head start you need to effect positive changes. Awareness needs reach everyone who touches software development in your organization – from requirements analysts to post-production support personnel. As you engage your workforce, be sure to keep the material fresh and in-step with what's going on inside your organization. Provide employees with the information they need to engage in the follow on steps of training and education, and make those steps easy to complete and highly-visible to anyone who's looking. Awareness needs to begin with an assumption of zero-knowledge – don't assume your developers understand application security concepts, principles, and best practices – lay it out so it's easy to find and easy to assimilate. Define common terms (e.g. Threat, Exploit, Defect, Vulnerability, etc.) so everyone understands them the same way, reducing confusion.

As your awareness efforts take hold and people come to realize how their approach to software development affects the security of applications – and they begin asking the right questions – they'll be prepared to 'do' something about it and that's where education programs can take root. The BITS Software Security Framework, published in January 2012 notes that "education and training program in a mature software security program represents the "lubricant" to behavior change in developers and as a result, is an essential ingredient in the change process." [7]

Moving into the Education Phase

While awareness efforts are never considered 'done', they can be progressively more detailed, preparing people for an education regimen that's tailored to their role.

People will naturally fall into one of several specific roles and each role has specific needs for specific information:

Architects and Leads

- Secure code starts with secure requirements and design
- Secure code does not equal secure applications
- Security is required through all phases of the process
- Consider threat modeling and code review processes

Developers

- Match training to level of awareness and technologies used

Testers

- Someone must verify the security of the end product
- Testers can vary in capability from developer level to data input personnel
- Information Security Personnel
- They know security, they don't necessarily know about application development or application-specific security concerns.

Management

- Project and program management, Line management and Upper management
- Need to understand the risks so they release the budgets to address them

Bundles of courses can be assembled to address basic or baseline education, introductory education, and advanced or expert education. Figure 2 is an example of how courses might be bundled to address the various roles and the levels of educating.

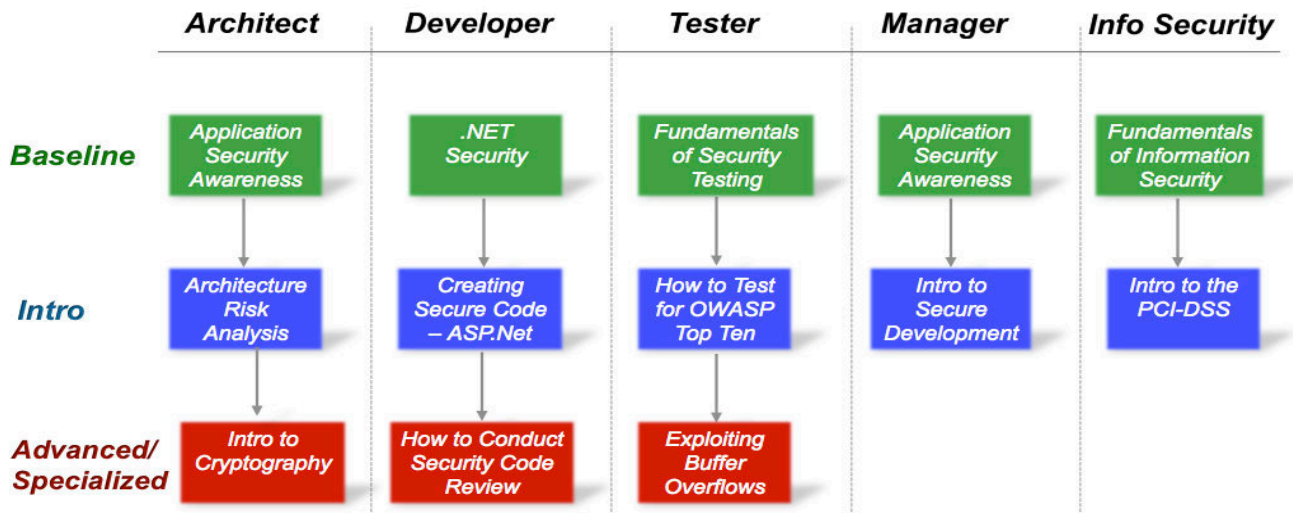


Figure 2. Bundles of Courses Stratified By Role in the SDLC [8]

Strategies for Rolling Out Training

Following are a few suggested approaches for rolling out your training program:

- Everybody gets everything
 - > Broadly deploy training level by level
- Core training plus security specialists
 - > Specialists by functional groups or projects
- Base training plus candidates for “Software Security Champions or Evangelists”
 - > Less training for all, but a few go to people embedded in groups or projects
 - > Multi-level support for developers with Base training
- Start slow
 - > Roll out to test group or organization
 - > Mix and match models and test

Selecting one of these or a hybrid of strategies will depend on several factors that are specific to your organization. These factors include geographical diversity where your development teams are located, separation or concentration of groups who are responsible for mission critical applications, existing infrastructures for educating employees, number of people available to conduct training, etc.

Measuring Success

The OWASP OpenSAMM Maturity Model describes a Level II of program maturity that supports a role-based education curriculum for development staff, under the Education and Guidance domain of OpenSAMM [9]:

“Conduct security training for staff that highlights application security in the context of each role’s job function. Generally, this can be accomplished via instructor-led training in 1-2 days or via computer-based training with modules taking about the same amount of time per person. For managers and requirements specifiers, course content should feature security requirements planning, vulnerability and incident management, threat modeling, and misuse/abuse case design. Tester and auditor training should focus on training staff to understand and more effectively analyze software for security-relevant issues. As such, it should feature techniques for code review, architecture and design analysis, runtime analysis, and effective security test planning. Expand technical training targeting developers and architects to include other relevant topics such as security design patterns, tool-specific training, threat modeling and software assessment techniques. To rollout such training, it is recommended to mandate annual security awareness training and periodic specialized topics training. Course should be available (either instructor-led or computer-based) as often as required based on head-count per role.”

At Level III maturity of Education and Guidance in OpenSAMM, the notion of certification for development roles appears. These types of certifications may be available internally though a custom-developed certification process, or available in the marketplace through such programs like ISC2’s CSSLP [10] and/or SANS’ GSSP [11] programs.

To help in determining the competency levels of your own staff to make sure that the education and practice you’re providing development teams is functioning as intended, two models may be useful for conducting your own assessments and potential certifications.

Life Cycle Skill Areas	Skills
Software Requirements Skills <i>References:</i> [ACM 2004] [Lapante 2009] [Robertson 2012] [SWEBOK 2014] [Wiegers 2013]	Software Requirements Elicitation Software Requirements Analysis Software Requirements Specification Software Requirements Verification and Validation Software Requirements Process and Product Management
Software Design Skills <i>References:</i> [IEEE 1016-2009] [IEEE 12207-2008] [IEEE 15528-2008] [SWEBOK 2014]	Software Design Fundamentals Software Design Strategies and Methods Software Architectural Design Software Design Quality Analysis and Evaluation
Software Construction Skills <i>References:</i> [ACM 2004] [Fowler 1999] [Hunt 1999] [McConnell 2004] [SWEBOK 2014]	Software Construction Planning Managing Software Construction Detailed Design and Coding Debugging and Testing Integrating and Collaborating
Software Testing Skills <i>References:</i> [IEEE 730-2002] [IEEE 829-2008] [IEEE 1012-2012] [Myers 2011] [SWEBOK 2014]	Software Test Planning Software Testing Infrastructure Software Testing Techniques Software Testing Measurement and Defect Tracking
Software Sustainment Skills <i>References:</i> [IEEE 12207-2008] [ISO/IEC/IEEE 24765:2010] [IEEE 828-2012] [Lapham 2006] [SWEBOK 2014]	Software Transition Software Support Software Maintenance

Table 1: Software Engineering Life Cycle Skill Areas and Skills

IEEE’s Software Engineering Competency Model (SWECOM)

The IEEE Competency Model [12] describes competencies for software engineers who participate in developing or modifying software-intensive systems. Skill areas, skills within skill areas, and work activities for each skill are specified. Activities are specified at five levels of increasing competency. The model includes case studies of how the SWECOM model can be applied by management, employees themselves, new hires, and curriculum designers. Table 1 lists the Development Life Cycle skill areas for each phase of the cycle.

Software Engineering Institute’s Software Assurance Competency Model

SEI’s Software Assurance (SwA) Competency Model [13] is intended to help to create a foundation for assessing and advancing the capability of software assurance professionals. Competencies range across knowledge areas and units -- providing a span of competency levels 1 through 5, as well as a decomposition into individual competencies based on knowledge and skills. This model also provides a framework for an organization to adapt the model’s features to the organization’s particular domain, culture, or structure.

The five levels of competency are characterized as follows:

L1 – Technician

- Possesses technical knowledge and skills, typically gained through a certificate or an associate degree program, or equivalent knowledge and experience
- May be employed in a system operator, implementer, tester, or maintenance position with specific individual tasks assigned by someone at a higher level
- Main areas of competency: System Operational Assurance, System Functionality Assurance, and System Security Assurance
- Major tasks: tool support, low-level implementation, testing, and maintenance

L2 – Professional Entry Level

- Possesses application-based knowledge and skills and entry-level professional effectiveness, typically gained through a bachelor's degree in computing or through equivalent professional experience
- May perform all tasks of L1. May also manage a small internal project; supervise and assign sub-tasks for L1 personnel; supervise and assess system operations; and implement commonly accepted assurance practices
- Main areas of competency: System Functionality Assurance, System Security Assurance, and Assurance Assessment
- Major tasks: requirements fundamentals, module design, and implementation

L3 – Practitioner

- Possesses breadth and depth of knowledge, skills, and effectiveness beyond L2, and typically has two to five years of professional experience
- May perform all tasks of L2. May also set plans, tasks, and schedules for in-house projects; define and manage such projects and supervise teams on the enterprise level; report to management; assess the assurance quality of a system; implement and promote commonly accepted software assurance practices
- Main areas of competency: Risk Management, Assurance Assessment, and Assurance Management
- Major tasks: requirements analysis, architectural design, tradeoff analysis, and risk assessment

L4 – Senior Practitioner

- Possesses breadth and depth of knowledge, skills, and effectiveness and a variety of work experiences beyond L3, with 5 to 10 years of professional experience and advanced professional development at the master's level or with equivalent education/training
- May perform all tasks of L3. May also identify and explore effective software assurance practices for implementation, manage large projects, interact with external agencies, etc.
- Main areas of competency: Risk Management, Assurance Assessment, Assurance Management, and Assurance Across Lifecycles
- Major tasks: assurance assessment, assurance management, and risk management across the lifecycle

L5 – Expert

- Possesses competency beyond L4; advances the field by developing, modifying, and creating methods, practices, and principles at the organizational level or higher; has peer/industry recognition; typically includes a low percentage of an organization's workforce within the SwA profession (e.g., 2 % or less).

The model also helps to establish proficiency targets for each job and role, and can be useful when assembling a team with distinct skill requirements to address application development, based on a risk assessment for the application under development.

A Checklist For Establishing a Software Security Awareness and Education Program

The checklist is offered to help remind you of key principles and practices that are proven for success. Consider these elements as you're formulating your overall customized program.

Summary

Secure Software Development is an all-encompassing and difficult problem to address and solve, and requires dedication, effort, and time to build an effective program. Awareness and Education are vital for success and require a many-hats approach that includes psychology, creativity, engaging materials, formal structures for learners to navigate, and a solid-rooting in how people learn and apply new skills in their jobs. Using the tips, recommendations, tools, and providing sufficient budgets and time to pull all these elements together, you will be well on the way to building the best program possible for yourself, your developers, and your organization.

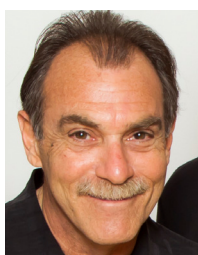
REFERENCES

1. Webcast Library. (n.d.). Retrieved September 30, 2015, from <<https://www.securityinnovation.com/security-lab/webcasts/webcast-library/>>
2. Application Security Maturity Model. (n.d.). Retrieved September 30, 2015, from <<http://web.securityinnovation.com/application-security-maturity-model/>>
3. Building Security In Maturity Model | BSIMM. (n.d.). Retrieved September 25, 2015, from <<http://bsimm.com/>>
4. Software Assurance Maturity Model (SAMM): A guide to building security into software development. (n.d.). Retrieved September 25, 2015, from <<http://www.opensamm.org/>>
5. Sandman, Peter. "Motivating Attention: Why People Learn about Risk . . . or Anything Else". Retrieved September 26, 2015 from <<http://www.psandman.com/col/attention.htm>>
6. McLeod, S. A. (2008). Cognitive Dissonance Theory - Simply Psychology. Retrieved September 26, 2015 from <<http://www.simplypsychology.org/cognitive-dissonance.html>>
7. BITS Software Assurance Framework. (2012, January 15). Retrieved September 26, 2015 from <<http://www.bits.org/publications/security/BITSSoftwareAssurance0112.pdf>>
8. Webcast Library. (n.d.). Retrieved September 30, 2015, from <<https://www.securityinnovation.com/security-lab/webcasts/webcast-library/>>
9. OpenSAMM Education and Guidance. (n.d.). Retrieved September 25, 2015, from <https://www.owasp.org/index.php/SAMM_-_Education_&__Guidance_-_2>
10. (Certifications) - CSSLP. (n.d.). Retrieved September 26, 2015, from <<https://www.isc2.org/csslp/default.aspx>>
11. Security Certifications: Software Security. (n.d.). Retrieved September 28, 2015, from <<http://www.giac.org/certifications/software-security>>
12. Software Engineering Competency Model (SWECOM). (n.d.). Retrieved September 29, 2015, from <<http://www.computer.org/web/peb/swecom>>
13. Digital Library. (2013, March 11). Retrieved September 29, 2015, from <<http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=47953>>

✓	Requirement for Program Success
	Executive Management establishes the mandate for software security and budgets the time, expense, and delegation of authority to improve software security.
	Company goals, policies, standards, and controls are in place for software security throughout the SDLC
	Learning media is geared to your audience based on their availability, geographic dispersion, access to materials (intranet-based vs. Internet-based), language considerations, sensitivity of time zones where personnel are located
	Reference tools are readily available to developers and are usable for just-in-time access for solving specific software security issues
	Examples of high quality and secure source code are available to show developers what needs to be accomplished and why
	Code examples come from familiar internal sources
	Courses are stratified by well-defined roles in the SDLC
	Progress of courses and completion of course bundles include reward and recognition steps that further motivate learners
	A metrics program has been established to show trends over time and help to identify components that are working as planned vs. those that need intervention or changes
	Program maturity is measurable and is used consistently

Table 2: Checklist For Establishing a Software Security Awareness and Education Program

ABOUT THE AUTHOR



Mark Merkow, CISSP, CISM, CSSLP works at Charles Schwab Co., Inc. in Phoenix, AZ as a technical director for Application Software Security. He has over 40 years of experience in Information Technology from a variety of roles, including Applications Development, Systems Design, Security Engineering, and Security Management. Mr. Merkow holds a Masters in Decision and Information Systems from ASU, a master's of Education in Distance Learning, and a bachelor's degree in Computer Information Systems. Mark has been working on software security issues since 1998, and has authored or co-authored 14 books, including, "Secure and Resilient Software Development" (2010, Auerbach Publications), and Information Security Principles and Practices, 2nd Edition (2014 Pearson Education).



Homeland Security

The Department of Homeland Security, Office of Cybersecurity and Communications (CS&C) is responsible for enhancing the security, resiliency, and reliability of the Nation's cyber and communications infrastructure and actively engages the public and private sectors as well as international partners to prepare for, prevent, and respond to catastrophic incidents that could degrade or overwhelm these strategic assets. CS&C seeks dynamic individuals to fill critical positions in:

- Cyber Incident Response
- Cyber Risk and Strategic Analysis
- Networks and Systems Engineering
- Computer & Electronic Engineering
- Digital Forensics
- Telecommunications Assurance
- Program Management and Analysis
- Vulnerability Detection and Assessment

To learn more about the DHS, Office of Cybersecurity and Communications, go to www.dhs.gov/cybercareers. To apply for a vacant position please go to www.usajobs.gov or visit us at www.DHS.gov.

Strategic Human Resource Management of Government Defense R&D Organizations

Kadir Alpaslan Demir, Ph.D.,
Turkish Naval Research Center Command, Istanbul, Turkey

Abstract. Sustainable growth is an important concept in strategic management. Sustainable growth rate is a metric commonly used in private sector to determine whether a firm has a sustainable growth or not. Because the main goal of private firms is to make money, sustainable growth rate is calculated using financial measures in the current strategic management literature. However making a profit is not among the main goals of government and non-profit research organizations. Therefore, to benefit from this important concept in strategic management of government defense R&D organizations, we reformulate the sustainable growth rate measurement to account for one of the most important capital of government defense R&D organizations. This important capital is R&D experience. With the help of sustainable growth rate, R&D managers are able to investigate future human resource scenarios of the organization and strategically manage the project workload and the R&D efforts.

1. Introduction

Government research and development (R&D) organizations (GRDO) play an important role in national innovation systems [1]. One of the latest reports from National Science Foundation indicates that a significant portion of R&D conducted in USA is the work of government R&D organizations. Federal government conducts 11.6% of total national R&D and 29.6% of total R&D is funded by the federal government [2]. Government R&D organizations have a strategic role in national R&D and they have to be managed strategically. Most of the strategic management literature focuses on issues related to private organizations. Therefore, strategic management theories and concepts in the current literature have a private sector focus. To make a profit is the main motive for private R&D firms. However, public research institutions including government R&D organizations have different primary motives [5]. They have a different and complementary role in the national innovation systems. Consequently, investigation into the behavior of such organizations require a different perspective than the current strategic management perspective, mostly focusing on private sector. For example, an important concept such as sustainable growth is quite relevant and useful to any kind of organization. Currently, sustainable growth is formulated using financial concepts and terms [3]. With the current definition and formulation, sustainable growth rate (SGR) cannot be used for government R&D organizations. So, in this study, we redefine and reformulate

SGR. As a result, GRDOs are now able to benefit from this important concept in strategic human resource management.

We experience that the sustainable growth rate metric can be quite informative and useful in strategic planning or budget meetings. The budgets of government R&D organizations are generally determined by higher authorities that are outside stakeholders. SGR will help to justify early or immediate hiring of R&D staff for future projects.

In our previous studies [4], we conducted many interviews and case studies with project managers regarding project management metrics. We found out that if the managers do not easily and clearly understand how the metric is derived, they are unlikely to use it [4]. Therefore, rather than developing metrics with complex formulations derived from complex theories, development of simple and easy to use metrics should be preferred. In many cases, having approximate results with explanatory power is preferred over having precise results as a result of complex and costly measurements [4]. The measurements should be simple, inexpensive, and effortless. The developed metrics should be simple in nature, easily applicable, and easy to understand. In the development sustainable growth rate metric, we abide by these principles. As a result, SGR is formulated in such a way that it is easy to understand, measure, and use.

2. Sustainable Growth Rate Metric

Many practitioners working in R&D, especially the ones working in government defense R&D institutions, will recognize the importance of domain and field experience. Having experienced researchers and engineers is essential in creating and sustaining a successful R&D organization. Consequently, R&D experience lies in the heart of sustainable growth rate (SGR) metric. In SGR calculation, our focus is measuring R&D personnel experience and workload and how well we maintain a certain level of experience with respect to workload in the organization. To calculate SGR for government R&D organizations, we use some of the well-known metrics. Let's briefly explain these metrics.

R&D Experience of an R&D Worker

An R&D worker is an employee who directly takes part in R&D related work. This worker may be a scientist, a research engineer, or a research technician. R&D experience is measured in terms of months or years depending on the granularity of the measurement.

Total R&D Experience of the Department/ Organization

To calculate the total R&D experience, we sum the R&D experience of all R&D workers in the department or in the organization.

Average R&D Experience of the Department/Organization

The average is calculated by dividing the total R&D experience by the number of R&D staff. This metric is quite useful

in presenting how experienced the organization is. Clearly, high figures are desired.

Future Total/Average R&D Experience

Naturally, organizations maintain an employment database, which is especially important for government defense R&D organizations. In most cases, development of defense systems must be conducted in secure environments [7,9]. Therefore, the employment databases include employee records with security clearances and background checks due to the classified nature of the projects. Using these databases, it is possible to extract personnel turn-over rate of the organization. This turn-over rate can be used to develop an expected yearly R&D experience change rate. If the organization has a research agenda that includes long-term plans for the projects to be undertaken, an analysis of these plans will reveal the personnel requirements for the future projects. Using the data derived from the analyses of the employment database records and the personnel requirements of the prospective projects, we can build computer simulations to predict the future state of the R&D personnel in our organization.

Future total R&D experience is an estimation of the R&D staff experience of the organization for a future date. It is calculated with the following formula:

$$\text{Future R\&D Experience} = \text{Current R\&D Experience} \times (1 + \text{Expected Yearly R\&D Experience Change})^{\text{Number of Years}}$$

$$FE = CE \times (1 + EYEC)^{NY}$$

In the formula, FE is the future R&D experience and CE is the current R&D experience. Expected yearly R&D experience change (EYEC) is the expected percent of change in the experience. It may be a positive or a negative number and it is calculated from statistics based on historical data. NY, number of years, is the difference between the date of calculation and the specified future date.

Future average R&D experience can be calculated similarly.

R&D Experience Change Rate of the Department/Organization

This metric is a combination of total and average R&D experience change rate for the department or the organization. Note that the total experience also accounts for the number of engineers. Therefore, the number of engineers is inherently included in the measurements. Each metric has a coefficient. The total of the coefficients is 1. These coefficients are determined based on the evaluation of the needed workforce. If the projects are complex and require highly experienced staff, then the coefficient of the average R&D experience (*Coefficient_a*) should be high. If the projects require manpower rather than experience, then the coefficient of the total R&D experience (*Coefficient_t*) should be higher. We currently set these coefficients to 0.5.

$$\text{R\&D Experience Change Rate} = \text{Coefficient}_t \times \frac{\text{Future Total R\&D Experience}}{\text{Current Total R\&D Experience}} + \text{Coefficient}_a \times \frac{\text{Future Average R\&D Experience}}{\text{Current Average R\&D Experience}}$$

Current Workload

Generally, government R&D institutions conduct business by responding to project requests from other agencies. If both parties negotiate on a project contract, then the government R&D organization issue a project charter. For accounting and auditing purposes, this project charter with a unique ID is linked to a government account. Then all the R&D work done related to the project is tracked using this account. Today, many government R&D organizations use automated systems to keep track of project effort. Even, use of certain automated systems is enforced by government auditing regulations. Consequently, the R&D effort for each project can be automatically measured by such automated accounting systems. In addition to the R&D project related activities, some government R&D organizations also tasked with activities such as certification, national standards development, providing consultancy to other agencies, and investigation of accidents and mishaps etc. These activities may be treated as a project and they can be tracked with automated effort tracking systems.

The current workload or project load may be derived from the project contracts in place. The project effort is expressed in terms of man-month. In SGR calculation, the current number of projects or the effort remaining for the projects are used to express the workload. Since the number of projects is a quite rough metric, we prefer to use effort remaining for the current projects in the SGR calculation.

Future Workload

Governments develop R&D agendas based on their national priorities. These agendas include long-term plans or responsibilities for certain government agencies and organizations. Naturally, government R&D institutions are a part of these agendas based on their strategic expertise areas. Defense R&D institutions have long-term plans since defense projects take years. Hence, these long-term plans help us in predicting the future workload for government R&D organizations especially for the ones in defense. It is possible to estimate the future workload based on statistics derived from project measurements database.

The future workload is estimated for a specific future date. This future date may be a year, 5 years, or 10 years later. The following formula is used to estimate the future workload:

$$\text{Future Workload} = \text{Current Workload} \times (1 + \text{Expected Yearly Workload Change})^{\text{Number of Years}}$$

$$FW = CW \times (1 + EYWC)^{NY}$$

In the formula, FW and CW denote the future and current workload respectively. EYWC, expected yearly workload change, is the expected percent of change in the workload. This change rate may be derived from statistics based on historical project data. NY, number of years, is the difference between the date of calculation and the specified future date.

Workload Change Rate of the Department/Organization

This metric is the workload change rate of the organization between the date of calculation and the specified future date. The workload change rate is calculated with the following formula:

$$\text{Workload Change Rate} = \frac{\text{Future Workload}}{\text{Current Workload}}$$

Sustainable Growth Rate (SGR)

The sustainable growth rate (SGR) is calculated by dividing the R&D experience change rate by the workload change rate. These rates are calculated for a specific time period. This period is determined by the scope of the strategic analysis:

$$\text{Sustainable Growth Rate} = \frac{\text{R\&D Experience Change Rate}}{\text{Workload Change Rate}}$$

Ideally, our goal is increasing the experience of the organization more than the workload. A higher SGR means that the organization has more experienced R&D staff dealing with the same amount of work. Here, we have a basic assumption that the current workload is satisfactorily handled with the current R&D staff. It is possible to relax this assumption by

calculating an adjustment rate for the difference between the current and the required manpower. If the SGR is higher than 1, then we conclude that the organization has a sustainable growth and the organizational R&D experience increase is higher than the workload increase. If the SGR is equal to 1, then we have an optimal growth since our R&D experience and workload is growing at the same rate. Finally, if the SGR is lower than 1, we have an unsustainable growth. The reason we still use the term growth is that based on the strategic management literature the organization is actually growing in terms of workload or number of projects. However, the growth is unhealthy because we will have less experienced staff per work package compared to the earlier states of the organization. We emphasize that R&D work is inherently complex and requires experienced knowledge workers. If the workload increase is faster than the experience increase, the staff will be overwhelmed and they will not be able to complete the projects with the required outcome or the necessary quality. Table 2 lists the interpretations of the SGRs.

3. Case Study

To explain the use of SGR, we developed a case study using a fictional organization. There are two reasons for not using a real organization. First, some of the data needed for the SGR calculation may be considered sensitive in many organizations. Second, we want to provide a striking example to better illustrate the use of SGR metric. The name of the organization is "Unmanned Aerial Vehicles Research Institute". This institute has 4 research departments and each department employs various number of engineers. Figure 1 shows the organizational structure and the number of engineers in the departments.

The R&D experience of systems engineering research department engineers is shown in Table 3.

We focus on only one of the departments to keep the case study simple. Let's develop a future scenario of personnel turn-over for the systems engineering research department. To make the case interesting and challenging, we assume the following conditions:

1. We start to lose some of our experienced engineers due to retirement or moving to a high-paying private sector job.
2. It is hard to attract experienced research engineers and therefore we could only hire recent graduates or inexperienced young researchers.
3. Our executive managers recognize the increasing workload and they hire additional engineers to increase the departmental capacity over the years.

Note that these conditions are for the sake of this particular case study. Different organizations have different conditions. Based on the conditions, managers may develop various future scenarios. Calculating sustainable growth rate using these scenarios help managers to shape the strategic human resource policies of the organization.

Metric	Definition	Unit
R&D Experience of an R&D Worker	Time spent on R&D work by an R&D worker	Months or Years
Total R&D Experience of the Organization/Department	The total experience of R&D staff in the organization/department	Months or Years
Average R&D Experience of the Organization/Department	The average experience of R&D staff in the organization/department	Months or Years
R&D Experience Change Rate	The rate of R&D experience change within the specified time	Without Unit
Current Workload	The project workload of the organization/department in terms of number of projects or remaining work for the contracted projects	Man-month
Future Workload	The estimated workload at a specific future date	Man-month
Workload Change Rate	The rate of workload change within the specified time	Without Unit

Table 1: Metrics used in the Calculation of Sustainable Growth Rate (SGR)

Sustainable Growth Rate (SGR)	Interpretation
Sustainable Growth Rate (SGR) > 1	Growth is sustainable.
Sustainable Growth Rate (SGR) = 1	Growth is optimal.
Sustainable Growth Rate (SGR) < 1	Growth is unsustainable.

Table 2: Interpretation of Sustainable Growth Rate (SGR)

Table 4 presents a simulation of personnel turnover for the next decade for the systems engineering research department. The R&D experience of each engineer is listed in the corresponding row. When an engineer leaves, we fill the position by hiring a new engineer. Highlighted cells indicate the hiring of a new engineer. For example, systems engineer 1 has 26 years of R&D experience in 2015. This engineer retires at the end of 2015. We hire a new engineer with no experience in 2016. In Table 5, we show the estimated future workload derived from the long-term plans in the research agenda of the institute. As previously listed under assumed conditions, the executive managers recognize the increase in the project workload and they hire young energetic engineers when a new project starts. Table 4 also shows that the number of engineers is doubled over a ten-year period as the number of projects increases.

We developed the case study in such a striking way that it presents how the use of SGR brings out a dangerous trend for the organization. The workload and the number of engineers is doubled over a decade. By just looking at these numbers, most executive managers would assume that the department will be in good shape since as the number of projects increases, the number of engineers increases accordingly. However, let's investigate how the R&D experience changes over the same period. Figures 2 and 3 show the total and average R&D experience and workload change for the systems engineering department between 2015 and 2025. In this period, even though the number of engineers is doubled, the total R&D experience stays almost the same, and the average experience decreases dramatically. In addition,

Unmanned Aerial Vehicles Research Institute Systems Engineering Research Department	
R&D Engineer	As of 2015 R&D Experience (Years)
Systems Engineer 1	26
Systems Engineer 2	22
Systems Engineer 3	20
Systems Engineer 4	18
Systems Engineer 5	11
Systems Engineer 6	6
Systems Engineer 7	4
Systems Engineer 8	4
Systems Engineer 9	1
Total R&D Experience for Systems Eng. Research Dept. (Years)	112
Average R&D Experience for Systems Eng. Research Dept. (Years)	12.4

Table 3: R&D Experience of Research Engineers in the Systems Engineering Research Department (in Years)

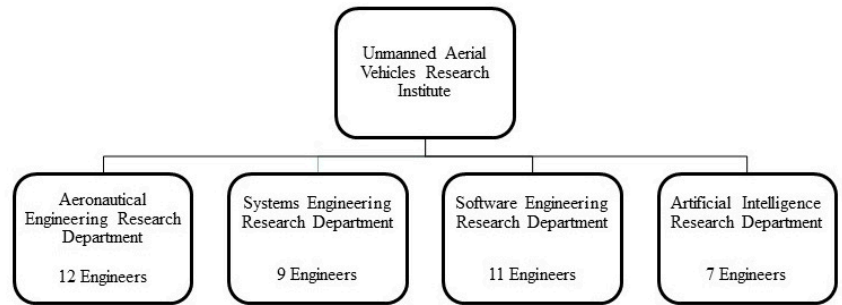


Figure 1: Research Departments of Unmanned Aerial Vehicles Research Institute

tion, while the workload is increasing, the average R&D experience is decreasing. This means that the department will have inexperienced staff to deal with an increasing amount of work. This is clearly a danger sign for the organization. Adding inexperienced staff to a project has short-term side effects. They need to be trained and this training is generally conducted by the experienced staff. While the number of research engineers in employment is an important metric, it is insufficient to explain a key aspect of the R&D organizations, which is the accumulation of expertise and research project experience. In essence, if organizational strategists only focus on the number of engineers, they are likely to miss this important aspect. Use of sustainable growth rate metric helps us to investigate trends for this important aspect.

In Calculation 1 we calculate the sustainable growth rate for the systems engineering research department. We use the simulated data from Table 4. Alternatively, we could derive the necessary measures from historical data. SGR calculation for 2020 is presented.

SGR for the period between 2015-2020

$$R\&D \text{ Experience Change Rate} = \text{Coefficient}_t \times \frac{\text{Future Total R\&D Experience}}{\text{Current Total R\&D Experience}} + \text{Coefficient}_a \times \frac{\text{Future Average R\&D Experience}}{\text{Current Average R\&D Experience}}$$

$$R\&D \text{ Experience Change Rate} = 0.5 \times \frac{86}{112} + 0.5 \times \frac{6.1}{12.4} = 0.63$$

$$\text{Workload Change Rate} = \frac{\text{Future Workload}}{\text{Current Workload}}$$

$$\text{Workload Change Rate} = \frac{162}{108} = 1.5$$

$$\text{Sustainable Growth Rate} = \frac{R\&D \text{ Experience Change Rate}}{\text{Workload Change Rate}} = \frac{0.63}{1.5} = 0.42$$

$SGR_{2020} = 0.42$

Calculation 1: SGR for the period between 2015-2020

For the period between 2015 and 2020, the SGR is 0.42. The sustainable growth rates for the systems engineering research department up to 2025 are presented in Figure 4. As the figure shows, SGR is in a downward trend indicating a human resources problem for the future. Even though the research department is growing in terms of projects, this growth is unsustainable. It is probable that these projects suffer from quality and other problems because these projects will be carried out by inexperienced R&D staff.

Systems Engineering Research Department	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
Systems Engineer 1	26	0	1	2	3	4	5	6	7	8	9
Systems Engineer 2	22	23	24	25	0	1	2	3	4	5	6
Systems Engineer 3	20	21	22	23	24	25	1	2	3	4	5
Systems Engineer 4	18	19	20	2	3	4	5	6	7	8	9
Systems Engineer 5	11	12	13	14	15	3	4	5	6	7	8
Systems Engineer 6	6	7	8	10	11	12	13	14	15	16	17
Systems Engineer 7	4	5	6	7	8	9	10	11	12	13	0
Systems Engineer 8	4	5	6	7	8	9	10	0	1	2	3
Systems Engineer 9	1	2	3	4	5	6	7	8	9	10	11
Systems Engineer 10			0	1	2	3	4	5	6	7	8
Systems Engineer 11			2	3	4	5	6	7	0	1	2
Systems Engineer 12					0	1	2	3	4	5	6
Systems Engineer 13						4	5	6	7	8	9
Systems Engineer 14						0	1	2	3	4	5
Systems Engineer 15								1	2	3	4
Systems Engineer 16								2	3	4	5
Systems Engineer 17									1	2	
Systems Engineer 18									0	1	
Total R&D Experience for Systems Eng. Research Dept. (Years)	112	94	105	98	83	86	75	81	89	106	110
Average R&D Experience for Systems Eng. Research Dept. (Years)	12,4	10,4	9,5	8,9	6,9	6,1	5,4	5,1	5,6	5,9	6,1

Table 4. A Sample Future Scenario of the Systems Engineering Research Department Staff Development – R&D Experience Change of Each Engineer over the Years (Highlighted cells indicate a replacement or an addition of a new engineer)

Systems Engineering Research Department	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
Number of Projects	4	4	5	5	5	6	6	7	7	8	8
Workload (Man-Month)	108	108	135	135	135	162	162	189	189	216	216

Table 5: Systems Engineering Research Department Project Workload Plan for the Next Decade

4. Discussion

Soft concepts such as project success, management effectiveness, or software quality are hard to measure and validate. In many cases, project managers prefer easy to apply, low cost, and approximate measurements rather than hard to use, costly, and precise metrics. Therefore, rather than having costly and precise measurements, approximate metrics, simply providing an indication of what is happening, are sometimes preferable. One of the most successful project management metric suites is Earned Value Management (EVM) metrics. In essence, the application of EVM gives the program managers a number that is either greater than 1, equal to 1, or smaller than 1. EVM provides an indication of how well the project is doing in terms of cost and schedule performance. Even though, there are many sources for error in the calculation of EVM metrics due to imprecise project planning and estimation, EVM is still a valuable tool and concept. This study is inspired from EVM. It is developed similar to EVM on purpose, so that the R&D department and program managers will quickly recognize the similarity and it is easily adopted. Note the similarity between SPI (schedule performance index), CPI (cost performance index) and SGR. All of them provide a number that is either greater than 1, equal to 1, or smaller than 1. As a result, for the SGR

calculation, having an indication of the future status is more important than having a precise sustainable growth rate.

For a government defense R&D organization, there will be an R&D program portfolio mostly guided by governing authorities. Most organizations have certain core competencies and they will try to maintain these competencies to have a competitive advantage. Trying to be an expert at some areas will be a smart strategic choice. Therefore, one of the assumptions regarding this study is that the future projects in these organizations will be similar to the current ones to maintain expertise areas defined in the mission statements. Most projects government defense R&D organizations deal with will be long-term projects evolving over time. Consequently, strategic long term planning of human resources is essential to be successful in these projects.

The sustainable growth rate metric may be applied at different levels or for specific areas. For example, the sample case provided focused on only one department. Therefore, SGR metric is applied at the departmental level. In addition, SGR metric may easily be applied at the organizational, research unit, or program level. The metric may also be applied for a specific R&D research area. For strategic planning purposes, managers may need to predict the future human

resource status of the organization for a strategic research area. For example, it is possible to calculate SGR for a domain specific language design for unmanned vehicle autonomous control software research area. Hence, the sustainable growth rate calculation is actually a generic framework that can be customized based on the needs of the strategic planners. The R&D managers can build a strategic portfolio for specific research areas calculating sustainable growth rate for each area in the portfolio. In this aspect, SGR becomes an essential tool for strategic R&D planning.

Another discussion point may be that the productivity of one talented researcher with some years of experience may be higher than another researcher with the same amount of experience. Some researchers will be more talented and productive than others. However, for all organizations, there will be an average talent across the organization. This average talent will not change dramatically in a short time frame, unless extraordinary staff changes occur. SGR is calculated on an organizational hierarchy level, not on the individual level. Therefore, SGR calculations assume that there will be an average talent and it is unlikely that this average will change dramatically in a couple of years for R&D organizations.

Another argument may be that years of experience do not necessarily translate into value. The researchers may just be marking time and not developing. We believe this would be hardly the case, since R&D managers are not supposed to sit idle when employees are not producing any value. Normally, the managers should be taking necessary precautions to prevent such cases and they should find ways to get the most out of their researchers. It is unlikely that the governing bodies will fund the projects of such underperforming organizations that produce little or no value.

5. Conclusion

Managing human resources is difficult for government R&D organizations due to many reasons including government regulations and security considerations. In addition to acquiring necessary clearances, hiring, staffing, and letting go of employees are bound by strict regulations. A previous survey study [8] confirms that in project management of software intensive systems development, staffing and hiring is more challenging in government organizations than it is in commercial organizations. As a result, a strategic management view and long term planning of human resources are essential in achieving a sustainable organizational growth. Metrics such as sustainable growth rate helps R&D managers in realization of a healthy organizational growth. As a strategic element in national innovation systems, government R&D organizations require good strategic management. With the development of a metric such as sustainable growth rate for government R&D organizations, we contribute to the current body of strategic management literature apart from the main stream. Note that while measuring SGR helps to examine an important aspect of organizational growth, it should not be the only measurement. We clearly need development of other metrics to investigate an important and abstract concept such as sustainable growth. We view the development of SGR based on human capital as a first step in this line of studies. As a measurement tool, SGR helps the executive managers to

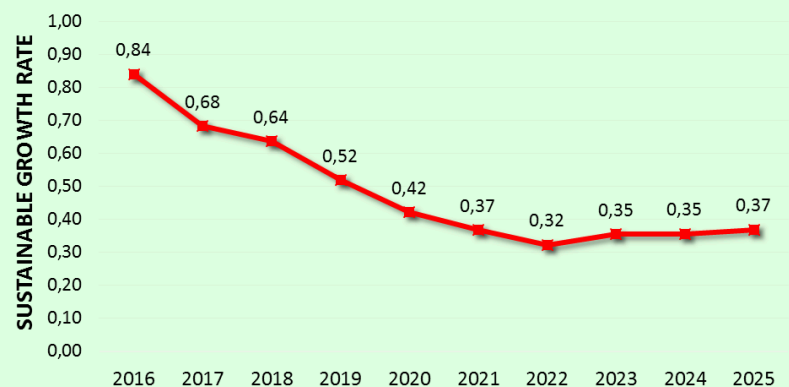
Figure 2. Total R&D Experience and Project Workload for Systems Engineering Research Department 2015-2025



Figure 3. Average R&D Experience and Project Workload for Systems Engineering Research Department 2015-2025



Figure 4. Sustainable Growth Rate (SGR) for Systems Engineering Research Department 2016-2025



strategically manage the R&D human resources, the most important capital of the organization.

Certain issues affect the productivity and research output. For example, adaptation of efficient processes increases productivity or increasing the paperwork may lower productivity. One possible extension to SGR metric may be the addition of adjustment factors to account for productivity differences resulted from increases or decreases in process efficiency.

As a conclusion, using SGR metric, it is possible:

- To investigate the future scenarios related to R&D experience changes for a department/organization/program/research area.
- To take actions such as hiring staff early enough for the time needed to transfer critical knowledge from experienced researchers before they leave.
- To communicate the likely future human resource status of the organization/department/program to outside stakeholders.
- To convince the governing stakeholders to fund the acquisition of human resources early enough so that the program/project does not become endangered due to loss of critical/experienced staff.

Essentially, SGR metric may be used for various purposes during strategic human resource planning for organizations with a strategic research portfolio consisting of long term projects. This study may be considered a framework for the application of sustainable growth concept to government defense R&D organizations. It is possible to extend and modify SGR metric for the specific purposes of managers.

6. Disclaimer and Acknowledgements

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any affiliated organization or government. Preliminary findings of this research was presented in the 4th International Conference on Leadership, Technology, Innovation, and Business Management (ICLTIM 2014), November 20-22, Istanbul, Turkey, 2014 [5] and subsequently included in [6].

ABOUT THE AUTHOR



Dr. Kadir Alpaslan Demir graduated from Turkish Naval Academy with a B.S. in Computer Engineering in 1999. After graduation, he worked as a Naval Officer in Navy Submarines. In 2003, he started his graduate education in Naval Postgraduate School, Monterey, CA, USA. Between 2003 and 2005, he completed one M.S. in Computer Science with a focus in Computer

Forensics and another M.S. in Software Engineering. Between 2005 and 2008, Dr. Demir completed his doctoral study with the research titled "Measurement of Software Project Management Effectiveness" and was awarded a Ph.D. degree in Software Engineering. He took a post as a faculty member in Department of Computer Engineering in Turkish Naval Academy. Dr. Demir taught undergraduate and graduate courses including software engineering, systems engineering for C4I, project management, research methods in science and engineering, and computer security. Since 2011, he has worked at Turkish Naval Research Center Command. Dr. Demir worked as a software developer and development team leader. Currently, he works as a program manager/assistant program manager in various mission-critical defense projects. He developed the organizational systems engineering processes and frequently participates in process improvement efforts. His research interests include software project management, project management measurement and metrics development, process improvement, change management, R&D and innovation management, systems and software modeling, formal methods, UAV systems simulation. Dr. Demir is currently pursuing a Master's in Business Administration with a focus in strategic management of technology, innovation, and human resources.

E-mail: kadiralpaslandemir@gmail.com

Web page: <http://www.softwaresuccess.org/About-Me.php>

Phone: +90 532 333 3988

Notes: *I appreciate your feedback regarding your experiences on the use of sustainable growth rate metric if you choose to apply it in your organization. Please send an e-mail to kadiralpaslandemir@gmail.com.*

REFERENCES

1. OECD, Public Research Institutions: Mapping Sector Trends. OECD Publishing, 2011. <http://dx.doi.org/10.1787/9789264119505-en>
2. National Science Board, Science and Engineering Indicators 2014. Arlington VA: National Science Foundation (NSB 14-01) 2014.
3. Higgins, Robert C, "How much growth can a firm afford?" Financial management 1977: 7-16.
4. Demir, Kadir Alpaslan. Measurement of software project management effectiveness. PhD Dissertation, Naval Postgraduate School, Monterey, CA, December 2008.
5. Demir, Kadir Alpaslan, and Tolga, Ihsan Burak. A Sustainable Growth Rate Metric Based on R&D Experience for Government R&D Organizations. 4th International Conference on Leadership, Technology, Innovation, and Business Management (ICLTIM 2014), November 20-22, Istanbul, Turkey, 2014.
6. Demir, Kadir Alpaslan, and Tolga, Ihsan Burak. "A Sustainable Growth Rate Metric Based on R&D Experience for Government R&D Organizations" Journal of Global Strategic Management, 16, 2014: 26-36.
7. Demir, Kadir Alpaslan. "Challenges of weapon systems software development." Journal of Naval Science and Engineering 5.3 (2009).
8. Demir, Kadir Alpaslan. A Survey on Challenges of Software Project Management. Proc. of the Software Engineering Research and Practice, pp. 579-585, Las Vegas, USA, 2009.
9. Demir, Kadir Alpaslan. Analysis of TLCharts for Weapon Systems Software Development. Master's Thesis, Naval Postgraduate School, Monterey, CA, December 2005.

PixelCAPTCHA

A Unicode Based CAPTCHA Scheme

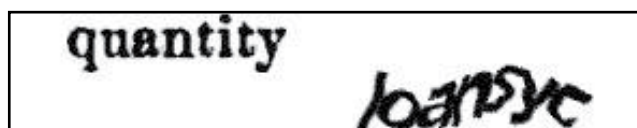
Gursev Singh Kalra, [Salesforce.com](https://www.salesforce.com)

Abstract. This paper will discuss a new visual CAPTCHA [1] scheme that leverages the 64K Unicode code points from the Basic Multilingual Plane (plane 0) to construct the CAPTCHAs that can be solved with 2 to 4 mouse clicks. We will review the design principles, the security mechanisms and its various features. We will also discuss the potential attack vectors on the proposed CAPTCHA scheme. The proposed CAPTCHA scheme will also be available as an open source Java library in near future.

Introduction

A “Completely Automated Public Turing Test to tell Computers and Humans Apart,” or “CAPTCHA,” is used to prevent automated software from performing actions that degrade the quality of service of a given system. CAPTCHAs aim to ensure that the users of applications are human and ultimately aid in preventing unauthorized access and abuse.

There are several types of CAPTCHA schemes that present audio and/or visual challenges to the user. These challenges require a human to interpret them and supply the solution that is validated by the server to allow or disallow the request. Image shows a reCAPTCHA [2] example

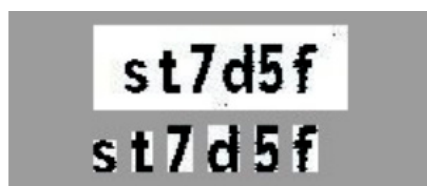


This paper has two main sections: the first discusses challenges with existing visual CAPTCHA schemes and the second section discusses the new PixelCAPTCHA scheme in detail.

Challenges with existing CAPTCHA schemes

Most of the visual CAPTCHAs rely on English alphabets and numerals, which makes them keyboard and locale sensitive. When the conforming fonts are used to build CAPTCHAs, solving the CAPTCHA can be as easy as running the Optical Character Recognition engines after removing the noise. The visual CAPTCHAs often have a similar threat model where the attackers follow three-step process to automatically solve them. The first step is to remove the noise, also called as preprocessing. Individual characters are then segmented in the second step followed by third step of character classification.

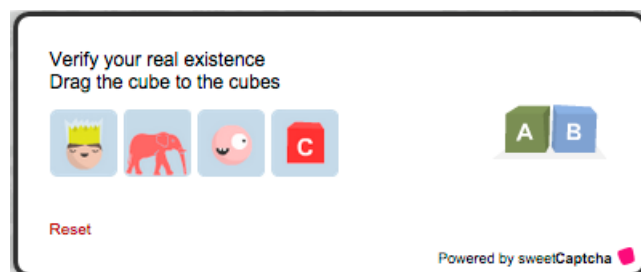
The image below shows a simple segmentation example [3].



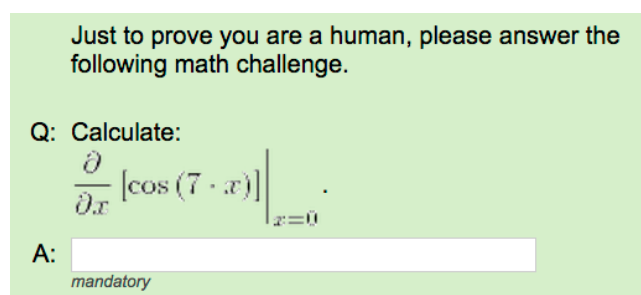
Character segmentation is typically the more difficult part of the automated CAPTCHA solving process. Once individual characters are segmented, computer algorithms can do a very good job of classifying the individual character images to corresponding alphabet/numeric characters. Given the small number of possible characters in the existing CAPTCHAs, automated classifiers can be written for a large number of the existing CAPTCHA schemes. The automated solvers can also use supervised machine learning algorithms to extract the features from the large number of test CAPTCHAs and solve new CAPTCHAs with high accuracy.

Owing to a common threat model between the visual, text based CAPTCHAs, researchers have experimented with alternate forms of CAPTCHAs with different threat models. These CAPTCHAs require you to interpret real world images, videos, solve calculus problems, read advertisements etc. Several of these alternate CAPTCHA implementations require a prior manual generation effort into building a knowledge base on which the users are tested. A few examples are sweetcaptcha, random.irb.hr etc...

The image below shows an example sweetCaptcha [4].



The image below shows a calculus based CAPTCHA [5].



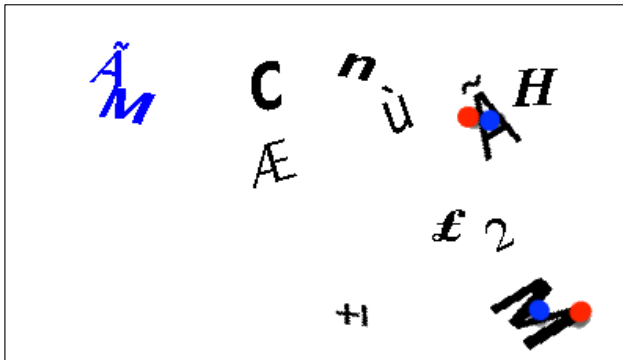
PixelCAPTCHA

Let us now look at a simple PixelCAPTCHA example and build an intuition around the proposed CAPTCHA scheme. On the image below, you will see 2 blue characters on the left and 10 characters in black on the right. You will also see the helper blue and red dots on the CAPTCHA. These dots are only for demonstration and to help build an intuition.

The blue characters are the challenge and the black characters contain the solution among other random characters. To solve the CAPTCHA, you will need to identify the black characters similar to the blue characters and click on the black ones. In the image below, the blue dots on the black characters are the actual solution coordinates and the red dots are the points where the user has clicked. The set of mouse click

coordinates makes your solution, which is submitted to the server for verification. The server computes the sum of minimum distance between the correct solution coordinates and the ones submitted by the user.

In the current example, the server will compute the two distances between the red and the blue dots and sum them up to arrive at a total distance/deviation from the correct solution coordinates. This deviation then will be compared against a pre-computed threshold for a given CAPTCHA to make a decision. The comparison threshold is different for each CAPTCHA and is calculated during the CAPTCHA generation process.



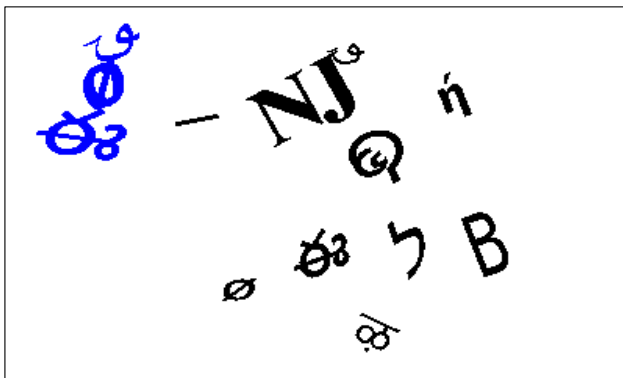
Features

Now that we have built an intuition for the CAPTCHA scheme, we will look at the various supported features in this section. These features allow you to control the CAPTCHA size, orientation, choose a character set and more.

Configurable Challenge and Response Count

The image discussed above was an example of a minimum complexity CAPTCHA that has 2 challenges and 10 responses to choose from, with a character set limited to 0 to 255 Unicode range. The PixelCAPTCHA library supports multiple configurations using which you can build CAPTCHAs that have 2 to 4 challenges, 10 to 12 responses to choose from and any set of characters from the Unicode Basic Multilingual Plane [6] (a.k.a. plane 0). Below is an additional PixelCAPTCHA example.

The image below shows a CAPTCHA with 3 challenges and 11 responses derived from 0-4095 Unicode code points

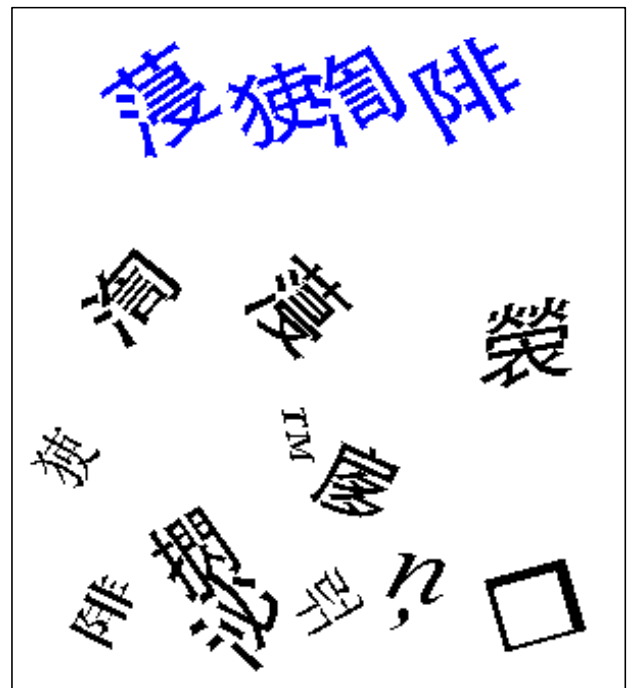


CAPTCHAs for Various Device Types

The PixelCAPTCHA library accepts custom dimensions and calculates the minimum and maximum Font sizes to be used to draw individual characters for the CAPTCHAs in addition to adjusting the challenge text orientation as discussed below.

Horizontal Orientation: When the CAPTCHA width is more than the height, the CAPTCHA gets a horizontal orientation, and the challenge text is drawn vertically to the left side of the CAPTCHA. The horizontal orientation is more appropriate for desktops or devices with larger screen areas. The examples discussed so far are of the horizontal orientation.

Vertical Orientation: When the CAPTCHA height is more than the width, the CAPTCHA gets a vertical orientation, and the challenge text is horizontally drawn to the top of the CAPTCHA. It may be easier for mobile users to view and solve vertical CAPTCHAs on their devices. The image below shows an example of vertical CAPTCHA generated by the PixelCAPTCHA library.



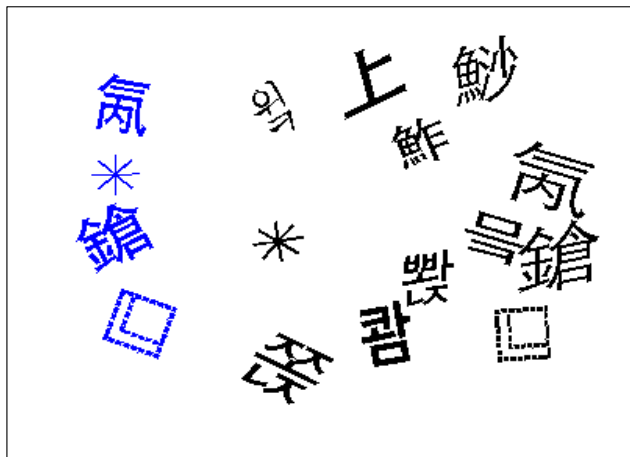
Configurable Character Set

Based on your requirements you can choose the character set for your CAPTCHA. The character set can be a list of Unicode code points from the Unicode plan 0 or multiple character ranges. This allows you to test out user acceptability before finalizing the CAPTCHA configuration for your application. A few examples to custom character sets are shown below:

- 0-255 – configures the PixelCAPTCHA library to use characters only from 8-bit ASCII character range.
- 0-4095 – configures the PixelCAPTCHA library to use characters only from 0 to 4095 Unicode code points.
- 65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90 – instructs the library to include only the uppercase characters

It is however recommended that the entire Unicode plane 0 code points be used for enhanced security as discussed in the security analysis section below.

The image shows a CAPTCHA with 4 challenges and 12 responses derived from the Unicode plane 0.

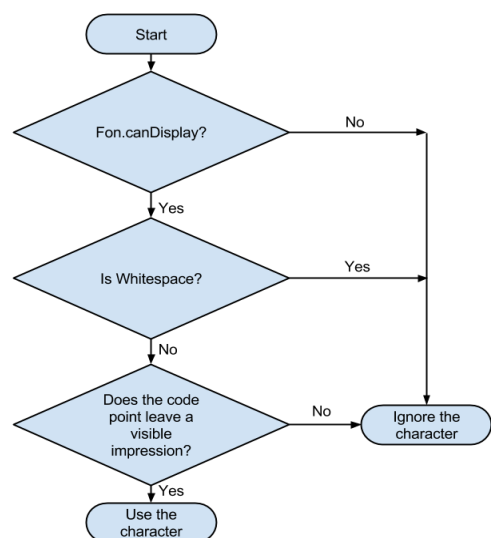


Printable Character Identification

Since the CAPTCHA relies on character correlation in order to be solved, it is very important that the characters leave a distinct impression on the CAPTCHA images. To be able to visually represent the various characters for correlation on the CAPTCHA image, the challenge and response Unicode code points could not be whitespaces and the Font [7] used to draw those must have valid glyphs [8] for each of the chosen Unicode code points. After checking for these conditions, the generated CAPTCHAs continued to have missing characters and the effect was more pronounced with smaller Unicode code points.

For example, when 0 to 255 Unicode code point range was used, more number of CAPTCHAs had missing characters. Further analysis revealed that Unicode code points 0 through 32 did not leave any visible imprints on the CAPTCHA images.

To ensure that the generated CAPTCHAs had visible and distinct characters, the following checks (shown in image below) were applied to each code point for all the characters in the Unicode plane 0 to identify those characters.



The table shows the results of the analysis. To conclude: in addition to code points corresponding to white space and non-valid glyphs, there was a significant number of code points that would not leave a visible imprint on the images.

Unicode Range	Total # of characters	# Of Whitespaces	# Of Font.canDisplay	Expected printable characters	Actual Count
0-255	256	10	256	246	189
0-4095	4096	10	3080	3070	2958
0-65535	65536	26	51878	51852	51580

Inbuilt Cache

The PixelCAPTCHA library has its own, in memory cache to store CAPTCHA solutions and corresponding identifiers. The current release does not require any persistent storage configuration. The storage times out individual CAPTCHA solutions and expires CAPTCHAs on single access, providing better security.

Service Provider Interface

The PixelCAPTCHA library also provides a new Service Provider Interface that you can implement to create wrappers around other CAPTCHA libraries and use them in your code without making extensive changes to your primary application. This allows you to decouple PixelCAPTCHA from your main application code.

Functional and Usability Benefits

The proposed CAPTCHA scheme offers the following additional benefits:

- It allows users to avoid typing and solve CAPTCHAs with a few clicks. This will offer usability improvements on the mobile devices where typing is a challenge and few taps can be used to solve the CAPTCHAs.
- The CAPTCHA scheme is independent of language, keyboard style and locale.
- The CAPTCHA generation process is completely automated.

Security Analysis

The suggested CAPTCHA scheme has been designed with several security features in mind and this section will visit those in some detail.

Probabilistic Analysis of Protection Against Random Guessing

In the very first image of this whitepaper, we discussed a CAPTCHA that has 2 challenges and 10 possible solution options to choose from. Let us assume that an attacker tries to solve that CAPTCHA by randomly selecting coordinates from any 2 of the 10 solution characters. The probability of correctly guessing the solution is $2/(10 \times 9) = 1/45$. This assumes that the order of the mouse clicks is irrelevant. That is, the attacker can click on the potential solution in any order.

For example, let's say that the two challenge characters are A and B, drawn vertically in that order, with A on the top. Consider the following two scenarios:

First: To solve the CAPTCHA, a user may be required to click on A and B in the solution area in any order, making up two possible solutions. The server will check for two possible solutions. This has a benefit of better usability, but offers less security in cases when attackers plan to randomly guess solution coordinates.

Second: To solve the CAPTCHA, a user will be required to click on A and B in the solution area in that particular order, making up only possible solution. The server will also check for only one solution. This has a benefit of higher security, but has a usability challenge.

The PixelCAPTCHA library can be configured to run in either of the two modes and instructed to honor or ignore the order of the clicks. The table below shows different CAPTCHA configurations and corresponding probabilistic analysis of protection against random guessing when ordered or unordered solution clicks are required.

Challenge Count	Response Count	Probability	
		Unordered Clicks	Ordered Clicks
2	10	1/45	1/90
2	11	1/55	1/110
2	12	1/66	1/132
3	10	1/120	1/720
3	11	1/165	1/990
3	12	1/220	1/1320
4	10	1/210	1/5040
4	11	1/330	1/7920
4	12	1/495	1/11880

Higher Classification Complexity

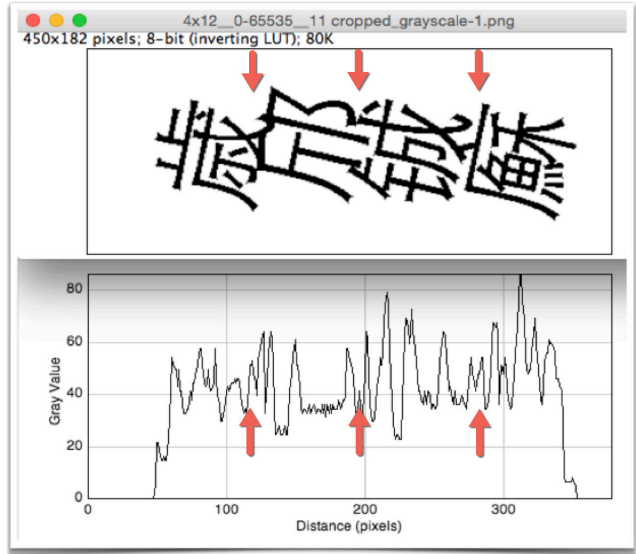
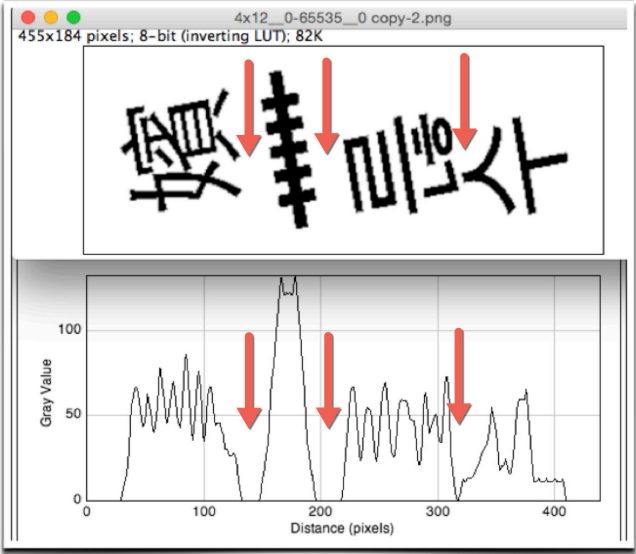
The Basic Multi Lingual Unicode plain consists of 65536 code points. It was concluded during the analysis that out of the possible 65536 Unicode code points, 51,000+ code points could be reliably drawn on the CAPTCHA. Having a large character space increases the complexity to write reliable classification algorithms. The Printable Character Identification section above has discussed how these Unicode code points were identified.

Collapsed Challenge Characters to add Segmentation Complexity

Individual character segmentation is an important aspect of automatically solving CAPTCHAs and one of the PixelCAPTCHA features is to add segmentation difficulty to the challenge characters; achieved by a random overlap between successive challenge characters.

The images below show the challenge component of two different CAPTCHAs. The top portion of each image shows the plotted challenge text and the lower portion shows the black pixel distribution [9] along the vertical axis. Since the challenge characters were not collapsed in the upper image, automated algorithms will be able to segment individual characters because of lower number of black pixels along the vertical axis. However, the bottom image has a random overlap between the challenge characters and there was no visible dip in the black pixel count

along the vertical axis. This increases the difficulty to segment individual characters and provides better security.



Random Font Construction

Randomly generated Fonts are used for each CAPTCHA character to augment the character correlation/classification complexity. The bullet points and the image below depict the process used to generate a separate Font for each CAPTCHA character.

- Pick a random logical Font from the list below. The current version uses Logical Fonts [10] as they are always present. The Physical Fonts [10] that may differ between systems.
Font.SANS_SERIF
Font.SERIF
Font.DIALOG
Font.DIALOG_INPUT
Font.MONOSPACED
- Choose random values for the following attributes:
Size
Bold (boolean)
Italic (boolean)

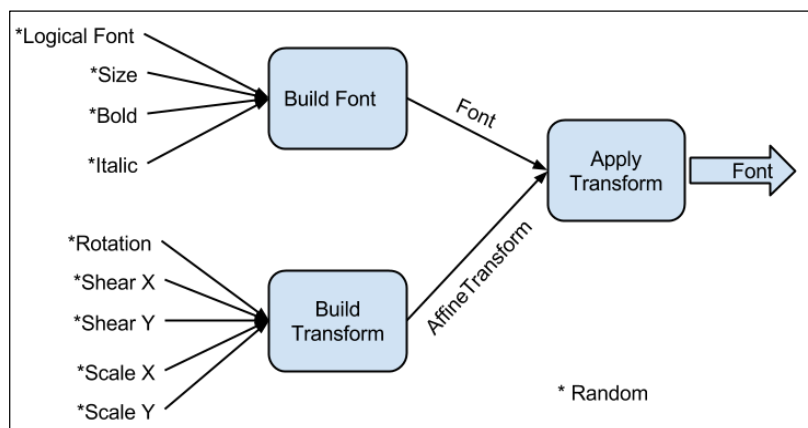
- Construct a Font from the values in step 1 and 2.
- Choose random values for the following
 - Font size multipliers for X and Y-axis (two separate values)
 - Rotation in degrees (positive or negative)
 - Shear values for X and Y-axis (two separate values).
 - To aid text readability and identification, the absolute shear values are restricted to be less than 0.5.
- Use the random scaling, rotation and shear parameters to construct an AffineTransform [11].
- Apply the AffineTransform to the Font constructed in step 3 to get a new Font, which is then used for drawing a character on the CAPTCHA.

Potential Attacks

Having both challenge and the solution text are part of the same CAPTCHA image presents a security risk. A typical attack pattern may be segmentation of the challenge characters and correlation between the segmented challenge and the solution characters. Computer vision and machine learning algorithms may also be leveraged to solve the proposed scheme.

REFERENCES

- <http://en.wikipedia.org/wiki/CAPTCHA>
- <http://www.google.com/recaptcha/intro/index.html>
- <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.3886&rep=rep1&type=pdf>
- <http://sweetcaptcha.com/>
- <http://random.irb.hr/signup.php>
- http://en.wikipedia.org/wiki/Plane_%28Unicode%29
- <http://docs.oracle.com/javase/7/docs/api/java/awt/Font.html>
- <http://en.wikipedia.org/wiki/Glyph>
- <http://imagej.nih.gov/ij/>
- <http://docs.oracle.com/javase/tutorial/2d/text/fonts.html>
- <http://docs.oracle.com/javase/7/docs/api/java/awt/geom/AffineTransform.html>



ABOUT THE AUTHOR



Gursev Singh Kalra is a Sr. Product Security Engineer at Salesforce.com. He worked with McAfee as a Senior Principal Consultant and led multiple software security service lines. He has authored free security tools like JMSDigger, Tesseract, Oyedata, SSLSmart etc. He has written several security-related whitepapers and his research has been voted among the top ten web hacking techniques of 2011 and 2012. He has spoken at conferences like BlackHat, OWASP AppSec, NullCon, Focus, ToorCon, and Infosec Southwest etc.

E-mail: gkalra@salesforce.com

CALL FOR ARTICLES

If your experience or research has produced information that could be useful to others, **CROSSTALK** can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for the areas of emphasis we are looking for:

Supply Chain Risks in Critical Infrastructure

Sep/Oct 2016 Issue

Submission Deadline: Apr 10, 2016

Agile Methods

Nov/Dec 2016 Issue

Submission Deadline: Jun 10, 2016

Please follow the Author Guidelines for **CROSSTALK**, available on the Internet at www.crosstalkonline.org/submission-guidelines. We accept article submissions on software-related topics at any time, along with Letters to the Editor and BackTalk. To see a list of themes for upcoming issues or to learn more about the types of articles we're looking for visit www.crosstalkonline.org/theme-calendar.





Upcoming Events

Visit <http://www.crosstalkonline.org/events> for an up-to-date list of events.

SIGCSE 2016

March 2-March 5, 2016
Memphis, Tennessee
<http://sigcse2016.sigcse.org/>

Software and Supply Chain Assurance

8-9 March, 2016
McLean, VA
<http://register.mitre.org/ssca>

Data Compression Conference

29 March- 1 April, 2016
Snowbird, UT
<http://www.cs.brandeis.edu/~dcc/>

24th High Performance Computing Symposium (HPC 2016)

3-6 April, 2016
Pasadena, CA
<http://www.iue.tuwien.ac.at/hpc2016>

2016 IEEE INFOCOM

10-15 April 2016
San Francisco, CA
<http://infocom2016.ieee-infocom.org>

Mobile Device + Test Conference

17-22 April, 2016
San Diego, CA
<https://mobiledevtest.techwell.com>

IoT Dev + Test Conference

April 17-22, 2016
San Diego, CA
<https://iotdevtest.techwell.com/>

STAREAST Software Testing Conference

1-6 May, 2016
Orlando, FL
<https://stareast.techwell.com>

The 38th International Conference on Software Engineering

14-22 May, 2016
Austin, TX
<http://2016.icse.cs.txstate.edu>

ISCC 2016- IEEE Symposium on Computers and Communications

27-30 June, 2016
Messina, Italy
<http://iscc2016.unime.it>

Down in the Trenches

Security and Defensive Coding

I started my career in computer science back in 1969

when an anonymous donor gifted my high school with a Wang Programmable Calculator (a 300 series, if I remember). It featured 320 words of memory (not Meg, not Kilo, no prefix at all – just 320 words). I learned a now-extinct machine language, allowing us to run 160-word programs and still have 160 words of memory left for data. Error checking? The mark of a “real developer” was that you could fit the quadratic equation into 160 words of code. You had to trust the user to input valid numbers. There was no room for defensive error-checking code.

Eventually, my high school moved up to a GE 635 (later bought out by Honeywell), and it became the H6000 series used by WWMCCS (Worldwide Military Command and Control System). The GE 635 we used was located in Minneapolis, and we had a Teletype Model 33 (with a 300 baud modem!). We’d type our code offline onto paper tape (you had a choice of BASIC or Fortran), and then dial up, upload the code from the tape, run it, and quickly disconnect (long distance calls cost money back then!). Error checking? We were teaching ourselves programming as we coded – and we were our own “users.” Smart enough to write the code? Then we’re smart enough to provide correct input!

In 1974 I joined the Air Force where my first assignment was Offutt AFB, Strategic Air Command, writing code to support the SIOP (Single Integrated Operational Plan), the tactical blueprint for the deployment of nuclear weapons by the United States. Using a Honeywell H6000 system (yes – the same GE 635 I had used in high school) for data processing, we interfaced with a system called PACER (Program Assisged Console Evaluation and Review).

In “A History of Online Information Systems – 1963 – 1976” by Bourne and Hahn (MIT Press), it states, “PACER represented the first real production implementation of an individual online workstation that operated with a coordinated multi-media, multifile capability (alphanumeric text documents, text documents, tabular material, aerial photos and electronic intercept documents).”

Did we do error checking in our code? Pretty much, that was all that we did! We even had a complete “test data base.” Until a

program had been installed and ran against the test data base for several days we wouldn't cut a program over to the "actual" data base. You see, we were writing code in support of nuclear targeting. I'm not exaggerating when I say that at least 50% of my code involved error-checking. When the results of your calculations are used to target a nuclear missile – yeah, you want to error-check everything. Twice. In fact, we had analysts whose sole job was to review, verify and validate program results. By hand. Every. Piece. Of. Data.

Over the years, I have worked on multiple other systems for the services – ranging from relatively small (under 10K lines of code) up to multi-million line systems. And I have never seen a system that did not require some type of "defensive coding" – code that assumes that the users might either accidentally or intentionally provide erroneous input.

I now teach "Enterprise Security" at my University. Enterprise security involves cybersecurity, installation security, data security, and coding security. Sounds like a lot, but I can pretty much sum up all of "Enterprise Security" in a few simple rules:

- Turn on and use a firewall and a GOOD antivirus/anti-spam/malware program – always!
- Make sure you update EVERYTHING automatically – the OS, and all critical program. For those programs that automatically "check for updates monthly" – manually update them. Daily. Because "zero-day vulnerabilities" shouldn't be open for 30 days!
- Backup often. Check your backups. Keep some offsite.
- Educate and test your developers and users regarding good security practices. Use sensible tests to verify understanding (Questions like "True or False: Viruses are bad" are insulting!) Fire those who won't listen or learn. Seriously.
- Understand the CIA triad (Confidentiality, Integrity, Availability).
- If you don't have Physical Security (so you can control access – both physical and electronic) – then it will probably never be secure.
- TRUST NO ONE! THAT INCLUDES OTHER DEVELOPERS!

That last one is pretty interesting – and in fact, merits a full semester course on "Coding Security." Data input from the user? Not trustworthy. Data files that have been used for months or years? Not trustworthy – might have latent bugs just waiting to crash the system. Even secure log in credentials cannot be trusted.

Log in credentials? Not the least bit trustworthy. Seriously, how many of you:

- Change your passwords on a regular basis?
- Use strong 12-character passwords with upper case, lower case, numbers and special characters?
- Use separate passwords for every different account, so that if one account is compromised, all your other accounts are safe?
- NEVER write down your passwords?
- Remember to flush all cached passwords before you let anybody else use your browser, even for a moment?

Developers need to learn to "code defensively," and that's hard. Even simple text input checking is hard. Anybody who has ever used (or worse yet – taught) C or C++ learns to hate input checking. The only really safe way is to read everything into a string buffer, then use various routines to parse input into valid tokens of the appropriate data type. Takes me only about a week to teach this in my C++ Object-Oriented Programming class – but then I spend the rest of the semester teaching and re-teaching the weird rules ("Don't mix getline and cin," "flush doesn't always work the same on all compilers or all operating systems").

I recently taught a semester-long "special topics" course on "How to code securely" – a course requested by 12 of my students. They were interested in learning how to "code defensively" so we spent a semester learning how to write "bad code" and then how to fix it so it's more secure. In fact, when I teach Software Engineering every year, the first assignment I give my students is to write a simple "read in and add three numbers from a user-supplied file" with the caveat that if I can run the program and cause it to crash, they get a zero on the assignment. And they learn how much harder it is to code defensively. And NOT to trust me (or any user) when it comes to supplying valid input data. It helps to imagine that all users of your code are terrorists, who are intent on breaking your code, corrupting your data, and generally trying to destroy "Truth, Justice and the American Way" in every manner they can. Sure motivates you to write defensive code.

A lesson all developers should learn.

David A. Cook, Ph.D.
Professor, Stephen F. Austin State University
cookda@sfasu.edu

CROSSTALK / 517 SMXS MXDED

6022 Fir Ave.
BLDG 1238
Hill AFB, UT 84056-5820

PRSR STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737



Homeland Security

Software and Supply Chain Assurance

Software and Supply Chain Assurance are essential to enabling the Nation's critical infrastructure.

To ensure the integrity of that infrastructure, the software and the IT supply chain must be secure and resilient.

The Software Assurance Community Resources and Information Clearinghouse provides corroboratively developed resources. Visit <https://buildsecurityin.us-cert.gov/swa/index.html> to learn more about relevant programs and how you can become involved.

Software and Supply Chain Assurance must be “built in” at every stage of development and supported throughout the lifecycle.

Visit <https://buildsecurityin.us-cert.gov/bsi/home.html> to learn about the practices for developing and delivering software to provide the requisite assurance. Sign up to become a free subscriber and receive notices of updates.

The Department of Homeland Security provides the public-private collaboration framework for shifting the paradigm to software and supply chain assurance.



NAV AIR



<https://buildsecurityin.us-cert.gov/swa>

CROSSTALK thanks the above organizations for providing their support.