Platform Infrastructure for Agile Software Estimation

Wagner, Nicholas A.

Monterey, California. Naval Postgraduate School

SYM-AM-22-070

PROCEEDINGS
OF THE
NINETEENTH ANNUAL
ACQUISITION RESEARCH SYMPOSIUM

Acquisition Research:
Creating Synergy for Informed Change

May 11–12, 2022

Published: May 2, 2022

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
DEPARTMENT OF DEFENSE MANAGEMENT
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School

Platform Infrastructure for Agile Software Estimation

Nicholas A. Wagner—has been a Research Staff Member at the Institute for Defense Analyses (IDA) since 2019. Before joining IDA, Nicholas was a materials science and engineering PhD student at Northwestern University, where he applied machine learning to the discovery of new electronic materials. While at Northwestern, he was a visiting student at Los Alamos National Lab as well as a research software engineering intern at Google Accelerated Science. His research interests at IDA include the application of data science techniques to cost estimation and program evaluation, crowd forecasting, and the governance of AI. Nicholas is a lifelong nerd and Phoenix native. [nwagner@ida.org]

Abstract

Software acquisition reform is a hot topic in the DoD, but the oversight community is struggling to adapt to changes. I diagnose some issues with the current state of business in the Software Acquisition Pathway and propose a system called Overlord to increase the level of automation in software program management and oversight. My goal is to make life easier for software developers, program managers, and members of the oversight community.

Introduction

Nearly 11 years after Marc Andreessen's claim that "software is eating the world" (Andreessen, 2011), the Department of Defense (DoD) is still struggling to appear appetizing. The department is well aware of its barriers to software acquisition from high-profile reports such as the National Security Commission on AI report in 2021 (Schmidt et al., 2021) and Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study in 2019 (Defense Innovation Board, 2019). Actions are being taken to lower the barrier to acquiring software, such as developing the Software Acquisition Pathway (SAP); building continuous authority to operate DevSecOps platforms like the Navy's Black Pearl, the Army's CReATE, and the Air Force's Platform One; and establishing "software factories" within the services. These innovations have returned results real enough that the model is being matured for scale across the DoD. Defense Deputy Secretary Kathleen Hicks has called for tighter integration of service software factories on a "reasonable" number of service providers and software repositories (Serbu, 2022). Doing so will help control cloud service costs as well as reduce technical barriers to code reuse, bringing DoD software development practices across the enterprise closer in line to commercial ones.

This integration of developer-friendly platform infrastructure is a welcome advance, but there is one key group of users who need more: the oversight community. As acquisition processes change to more closely follow the commercial technology sector's agile approach, the way programs are assessed for cost, schedule, and value must evolve as well. And just as enterprise development infrastructure has enabled a software development revolution for the DoD, an enterprise infrastructure for oversight could also unlock great value for the oversight community. In this paper, I lay out a proposal for a data system I call Overlord to do just that. I begin with what I see as the main job roles affected by oversight of the SAP and what their desires are. I then discuss how the current state of oversight leaves those desires unsatisfied. To inform my proposed solution, I describe two examples of currently existing infrastructure, Platform One and the Cost Analysis Data Environment (CADE), and how they meet or fail to meet the desires of intended users. After, I suggest how the DoD can remedy these problems

with an enterprise data system called Overlord. Finally, I sketch out some ideas for implementation.

The Roles

There are three archetypal job roles in the software acquisition world: developers, overseers, and program managers. Developers are the most straightforward in their desires. They want to build interesting software and be troubled as little as possible with busywork that detracts from that. This is an understandable instinct, but it can lead to phenomena such as "nerd sniping"—when developers become fixated on interesting subproblems to the detriment of the larger goal (Munroe, 2007)—or other behaviors that delay delivery to the end user. "Overseers" is a term used here as a catchall for people who are interested in making sure resources dedicated to a project would not be better used elsewhere. This group includes staff overseeing portfolios of projects, service and OSD cost estimators, inspectors general, the GAO, and the like. Overseers are also interested in a project's success, but they are equally, if not more, concerned with the use of taxpayer funds in a manner that minimizes the risk of waste. Due to their concern with risk, overseers typically desire as much information as possible as a condition of granting resources so they might better forecast which projects will fail. This desire naturally conflicts with developers' desires to focus on their work. In this scenario, program managers are placed in the middle. They focus on acquiring and keeping the resources they need to deliver their project in a manner satisfactory to the customer. This approach usually entails tracking critical metrics of value, assigning work to team members, pushing for adherence to the expected road map, holding meetings to get status updates from developers and remove their obstacles, and providing updates to overseers in turn. Great program managers shoulder these necessary responsibilities while keeping the overhead burden on their developers to a minimum. I summarize these three roles and their relationships in Table 1.

Table 1. Motivations and Risks for Software Acquisition Archetypal Roles

	Developer	***Program Manager***	***Overseer***
Primary Selfish Objective	Wants to minimize time not developing software	Preserves program resources, keeps developers producing	Wants as much information as possible to predict the future
Primary Risk to Organizational Success	Not always focused on risk to program portfolios or taxpayer	Overloads developers with reporting or fails to communicate crucial management info	Not always considerate of developer time pressures

The balance of this tripartite ecosystem has important consequences for the industrial base. Overly expansive documentation requirements for oversight are both literally costly in the sense that record systems and staff to run them cost money, but also in the sense that developers have nonmonetary concerns about how bureaucratic their work is. These concerns can drive off nontraditional vendors and talent. On the other hand, insufficient transparency can make it impossible to make informed assessments of program performance or estimates of future software cost, schedule, and quality. The optimal trade-off is getting the information needed to effectively oversee a software program to the right people with a minimum amount of effort on the part of developers. I discuss the SAP's attempt to thread this needle next.

The Current System of Software Program Oversight

The 2019 DIB SWAP study documented several deficiencies with how software programs were typically managed. Summarized, the DoD treated software like hardware, trapping software programs in a 1970s paradigm of waterfall development

where speed took a back seat to years-long planning. Unscheduled change was viewed as a planning failure rather than responsiveness to user needs. Software was taking too long to acquire, cost overruns were common, and failures to adapt to changing requirements resulted in public embarrassments. The SWAP study called for an alternative to the traditional combination of Joint Capabilities Integration and Development System (JCIDS); Planning, Programing, Budget, and Execution (PPB&E); and Defense Federal Acquisition Regulation Supplement (DFARS). This resulted in the Software Acquisition Pathway (SAP) a year later. SAP programs will be the focus of this paper as they represent the current state of the art when it comes to agile software development.

SAP program reporting is rooted in some of the SWAP study's other recommendations. As part of its work, the SWAP study developed a guide on "Metrics for Software Development" (DIB, 2019, S88-S90). It concluded that the use of source lines of code (SLOC) as a software complexity metric and SLOC/unit time as the corresponding productivity measure for developers was not a productive way to measure the status of programs. The guide dived deep into suggested program metrics covering deployment rates, response rates, code quality, functionality, and importantly to the overseers, software program management, assessment, and estimation. The full list is not pasted here as it is fairly lengthy, but I recommend reading either the source material or the condensed list provided on the Defense Acquisition University's (DAU) page on the SAP's Program Management Metrics and Reporting (DAU, 2022).

The exact reporting requirements for SAP programs are described next. Discussed first is the simplest case of those with less than $20 million of software expenditures and below the Acquisition Category (ACAT) 2 threshold. Currently, the primary recipients of metrics reporting for such programs are program managers, their immediate decision authority (DA), and the Office of the Under Secretary of Defense for Acquisition and Sustainment (OUSD(A&S)).

Small Program Oversight Data

In terms of key data collection, program managers or their end-user sponsors are required to collect four main sets of quantitative information: value assessments, cost estimates, program metrics, and road maps. Value assessments are conducted annually at a minimum and capture outcome-based measures of a program's ability to meet end-user mission goals. Cost estimates are a responsibility of program managers and must be prepared before program execution and updated at least annually. Additionally, program managers must develop a Metrics Plan. The Metrics Plan identifies metrics to be collected in order to manage the software program's day-to-day performance. This plan may overlap with some of the information collected for the value assessment, but it should also contain information on process efficiency, software quality, software development progress, cybersecurity, and cost. A minimal subset of these metrics is required to be provided quarterly to the relevant OUSD, such as OUSD(A&S) or OUSD(R&E). Road maps show planned goals and features of each software iteration over the next 18 months. The OUSD(A&S) provides guidance on all four of these items, including recommending that programs provide data access to approved stakeholders with automated read-only self-service portals. However, the exact metrics and manner of metric collection and sharing is ultimately left up to the programs.

In addition to the previously mentioned requirements, some data explicitly used for SAP review and not program oversight must be reported to OUSD(A&S) semiannually. As of March 5, 2022, the reporting guidance on the DAU's "Program Management Metrics and Reporting" page tells decision authorities of programs using

the SAP to email a form to OUSD(A&S) with 11 metrics twice a year (DAU, 2022). The use of email is intended to be temporary while formal reporting systems are updated. Those metrics are:

1. Avg Lead Time for Authority to Operate (days)
2. Continuous Authority to Operate In-Place
3. Mean Time to Resolve Experienced Cyber Event
4. Mean Time to Experience Cyber Event
5. Avg Deployment Frequency
6. Avg Lead Time
7. Minimum/Maximum Lead Time
8. Avg Cycle Time
9. Change Fail Rate
10. Mean Time to Restore
11. Value Assessment Rating

Larger Program Oversight Data

For programs above $20 million in software expenditures, a Contractor Cost Data Report (CCDR) and Software Resources Data Report (SRDR) must also be filed. CCDRs and SRDRs are filed at program start and then regularly with each major release until program completion. They are Excel or JSON files containing information on expected and actual software size and complexity, cost breakdown, and development schedule. These data are fed into the Cost and Software Data Reporting (CSDR) system managed by the CADE under OSD's Cost Assessment and Program Evaluation (CAPE) office. For programs above the ACAT 2 threshold, an independent cost estimate (ICE) by CAPE is required. CAPE analysts will therefore directly receive metrics in some manner and conduct site visits. CAPE is required to be notified 210 days before the execution decision is made and must hold a kickoff meeting no later than 180 days before the execution decision.

Problems with the Current System

There are three main problems with the current system:

1. Data that would be useful across the enterprise is not available at the enterprise level.
2. Oversight activities take a long time, limiting a program's ability to rapidly execute.
3. Automation is an additional responsibility.

The first point is a consequence of the high level of delegation to program managers for most information collection. Aside from the 11 metrics required by OUSD(A&S) semiannually, the program's metrics are collected in whatever manner the program manager decides. OUSD(A&S)'s system for storing and organizing the information reported to them is unclear. For larger programs, some information is collected within CADE in the SRDRs and CCDRs, but these reports only partially cover information from the value assessments and program metrics. And when CAPE conducts a cost estimate, its analysts are free to collect additional data in yet another manner. As a result, the ability to access and analyze historical program data is compromised. These varied approaches make it difficult for other programs to improve the accuracy of their own estimates, more difficult to assess past performance, and more difficult to develop tools for process automation (Putnam-Majarian & Staiger, 2019). This lack of access to data is reflected in the rather sparse published studies looking at agile

software estimation in the DoD and their small sample sizes (Goljan, 2021; Madachy, 2018).

The second point on long execution times for oversight is largely a result of the first, but lack of historical data is not the sole cause. There is also the issue of a lack of tooling supporting common activities like cost, schedule, and value estimation. Because of difficulty locating and obtaining relevant program data and reliance on manual analysis, cost estimators take a long time to prepare their analyses. Furthermore, the estimates themselves are preserved in a manner inconducive to automated updating or reuse (e.g., as Excel files with expert knowledge embedded in their construction). In a way, the situation is actually reminiscent of the modelling challenges facing the Joint Artificial Intelligence Center (JAIC), which also struggles with data access and model operationalization. I should also note that while the half-year minimum timeline for independent cost estimates is public, the timeliness of program cost estimates is unknown but presumably faster. In industry practice and under USD(A&S) guidance, cost/schedule/value estimates are iteratively refined as time goes on and more data is collected. In the commercial technology industry, estimates are updated in real time or, at the very least, at the cadence of a sprint cycle. If government estimates were compressed to this timescale, there would be much better visibility into problems before they occurred and less of a burden on programs when estimates need to be conducted.

The use of automation was recommended by the SWAP study authors as a solution to the timeliness issue and has since been repeated by other bodies such as the Government Accountability Office (GAO, 2021). This key role of automation was encoded in DoD Instruction 5000.87, Section 3.3.b.11, which demands that SAP programs' "metrics collection will leverage automated tools to the maximum extent practicable" (DoD, 2020, p. 18). But what makes automation practicable? Programs are already resource-constrained fulfilling the needs of their users. The addition of reporting automation development to their existing workload is unlikely to be prioritized.

These challenges are not being faced for the first time. Before proposing a solution, it is helpful to explore the examples of two systems already discussed in this paper, Platform One and CADE.

Infrastructure Examples: Platform One and CADE

Platform One

Platform One is an official DevSecOps Enterprise Services team for the DoD housed within the Air Force Office of the Chief Software Officer. Platform One provides several backend services for software developers in the DoD, such as prescreened software containers, cloud-native access points, identity management capabilities, automated testing pipelines, and coding collaboration tools. These services allow the operation of "software factories," which developers in program offices use to more easily produce operational software. Users are charged fees commensurate with their level of demand on the Platform One team. If no team labor is required, users frequently pay nothing. By turning industry-standard software development tooling, security, and operations into services, Platform One frees developers to stop repeating the 85% of work common to all DoD programs (Platform One, 2020), and lets them complete their program-unique work more quickly. A diagram (Figure 1) provided by Mr. Nicolas Chaillan, former Chief Software Officer of the Air Force, illustrates how Platform One serves as the connective layers between the cloud computing base infrastructure and the application layer being created by program software development teams (see "YOU" in Figure 1). Before the establishment of Platform One, developers with knowledge of

industrial best practices knew these layers were critical, labor-saving capabilities in software engineering. However, each program was expected to pay for and stand up such capabilities on their own, much like how SAP programs today are expected to adopt automation in their oversight reporting. Platform One demonstrates the time and cost savings that can be realized by investing in enterprise-wide capabilities.

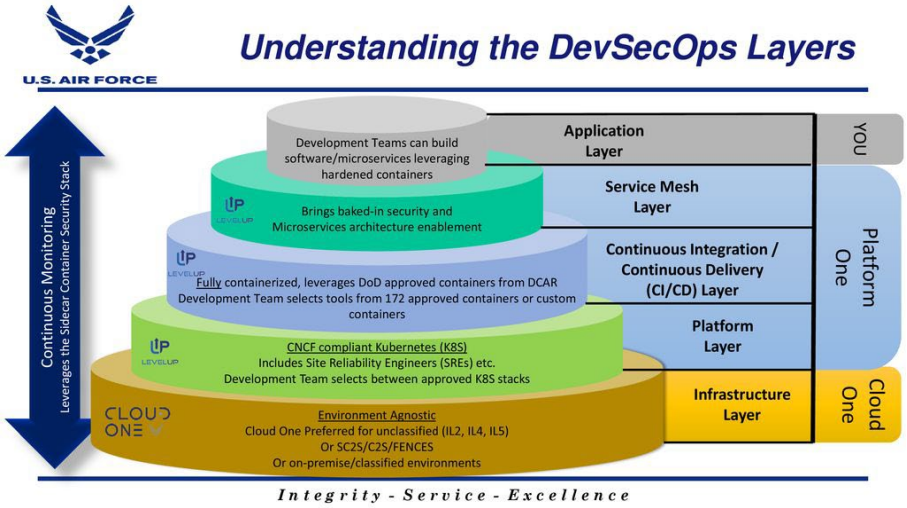

Figure 1. Services Provided by Platform One
(Chaillan, 2020)

CADE

The Cost Analysis Data Environment (CADE) is a data system run by OSD CAPE. It serves as the official repository for Contractor Cost Data Reports (CCDRs/1921s); Cost Analysis Requirements Descriptions (CARDs)/Technical Data (1921-Ts); Selected Acquisition Reports (SARs); and the Software Resource Data Reports (SRDRs) submitted by programs. CADE's vision is to "provide comprehensive data availability and automate common data visualization methods to help depict each program's unique story" (CADE, 2022c). Government overseers can search within their web browsers for data by several categories, including service, program, ACAT, contractor name, and a few others. Once data is located, CADE can provide visualizations and allow users to download the data, typically as an Excel file or PDF. CADE also provides a Data set, Tools, and Modelling Hub (DTMHub) where groups of users can share resources that they manage. These resources might take the form of an Excel spreadsheet that can be downloaded, a software application, or a link to another website. Note that there are some gaps in the data available in CADE on agile software projects. This data is generally more applicable to the waterfall method of software development historically prevalent in DoD software acquisition programs. Only in 2017 did the reporting of agile-focused alternatives to traditional software metrics, like lines of code, become possible on SRDRs, and their inclusion was not mandatory. As mentioned previously, programs with software efforts less than $20 million are not typically required to submit SRDRs, and therefore data on smaller software programs in the SAP is also sparse. Despite these limitations, CADE is better than the many stovepiped systems that it replaced. In recent years, CADE has added functionality for its users and updated documentation at a decent clip. Where CADE perhaps falls short, aside from its data availability issues, is its focus on manual workflows (Figure 2) via graphical user interface (GUI) rather than on automated ones.

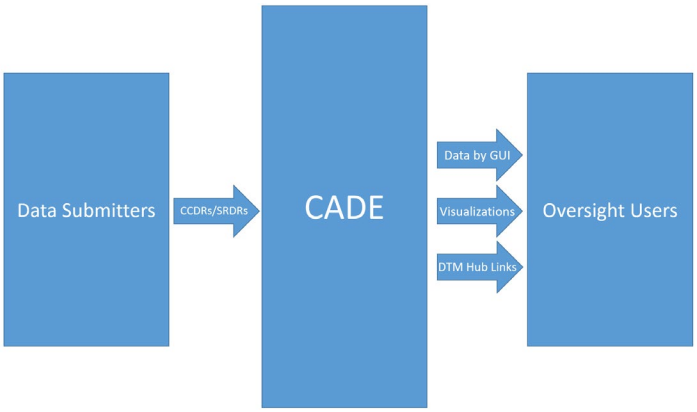

Figure 2. Sketch of CADE's Use for Software Programs

An Enterprise Infrastructure Proposal

I propose a potential enterprise infrastructure combining the commercial technology best practices of Platform One with the demonstrated oversight utility of CADE to address the challenges of data availability, speed, and automation raised previously. For convenience, I will call the hypothetical system "Overlord." A rough sketch for Overlord is presented in Figure 3, with its components discussed in the following sections.

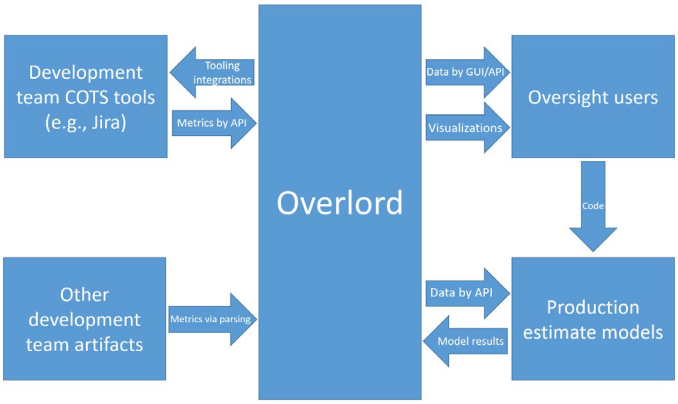

Figure 3. Overlord System Sketch

I begin with the challenge of automated data extraction. I suggest providing automation support for agile software program metrics in an iterative manner beginning with what are already known to be small lifts with high impact. As already known to readers who have utilized popular commercial off-the-shelf (COTS) software development tools such as GitLab, Bitbucket, or GitHub, these applications can instantly supply certain types of information such as commit activity data, team size, merge rates on pull requests, and other information with no manual effort through application programming interfaces (APIs). GitLab and other popular project management tools such as Jira and Asana can also export key agile metrics such as story point counts and backlog burndown rates. By leveraging existing automation and building out more capabilities to do so, effort required of developers can be minimized. Rather than have each program set up their own connections to these APIs to populate reports for emailing, Overlord would provide its own published API that handles receiving this information into a database. Publishing this API is important as it prevents the vendor

lock associated with tying support to one particular tool. After an enterprise team writes the software to connect two APIs, developers on program teams would grant the specific access required to establish a link. The data could then be updated automatically moving forward with a simple script. Since many COTs tools are used by multiple teams, this setup can require development effort once and then be repeated easily, similar to how Platform One leverages commonalities among software development efforts. Of course, not every development team will use COTS solutions for some of these metrics, especially those contained in the value assessments, so Overlord would also support upload and parsing of workhorse formats such as Excel, CSV, PDF, Word, and JSON. This parsing will be much easier with provision of standard templates and detailed guidelines similar to those provided by CADE for SRDRs. Support for other tools will be added gradually over time in order of demand.

There is also some work to be done on common infrastructure regarding how data is pushed out, not just pulled in. CADE, for example, has a web browser GUI. Only within the last year was support added to download SRDR data in bulk (CADE, 2022b). APIs can help here as well, allowing various types of users to pull the exact data they need automatically without necessarily needing to understand the inner workings of Overlord's database. This would enable, among other things, easier integration with enterprise data platforms like Advana and live feeds to cost, schedule, and value models. Another big advantage of an API is low overhead for access role creation. For example, a contractor could be granted access to pull data only from their own past projects, or an academic researcher could be granted access only to non-proprietary fields or aggregated quantities. These types of access are currently forbidden in CADE, out of both a fear that proprietary data will be inappropriately accessed and the infeasibility of someone manually curating data for sharing.

Not that web browsers should be entirely neglected in favor of access by API. Once again, CADE has some features to highlight here. One is that users can search by certain structured fields like program, WBS, service, and so on. These ideas could be extended further by allowing for searches of unstructured full text using something like Solr or Elasticsearch. Such search functionality would be especially vital when many program documents contain vital information in narrative form. Another feature of CADE worth copying is the automatic generation and display of commonly used plots for traditional acquisition programs like average procurement unit costs over time. Overlord would supply similar functionality but with common software development-related plots like product road map, sprint burndown, epic burndown, velocity, and control charts. Users would be free to export the plot data for re-creation in their own preferred plotting tools, if desired.

But the collection, sharing, and display of program data is not the only purpose of this system. The data it collects would also be a key driver of enhancements in development productivity. For instance, one of the common complaints about agile software estimation is the inconsistency of software size measures, such as story points between teams or even the same teams over time. Story point planning is a manual process that is subject to biases and uses lots of personal experience and gut intuition. This is especially hard on newer teams and makes analogizing coding speed estimates to later projects difficult. By capturing historical schedule metadata, extensions for development planning tools like Jira can be built that show the historical point totals and schedule actuals for similar stories, reducing mental effort on the developer's part and increasing consistency. Similarly, teams often struggle with decomposing high-level features into a set of digestible, specific tasks. Overlord can provide suggestions for

developers from close past examples, increasing the thoroughness of planning. As time goes by, the amount of data collected and feedback on suggestions would increase, improving the quality of the suggestions.

There has been plenty of discussion of how machine learning and AI will change the nature of estimating itself in the DoD. Perhaps bucking the trend, I do not suggest building another platform for hosting production machine learning models. I think it would actually be a better use of the oversight community's resources to piggyback on the JAIC's work to put models into operation and host estimation models on their infrastructure. Instead, a hub would be provided on Overlord where users could view model outputs with dashboards and obtain links to the data supplied from Overlord for the model, the model documentation, and code used. This would essentially be an expanded version of CADE's DTMHub.

To illustrate Overlord's potential, imagine the following demonstration:

Alice is a new cost analyst in CAPE. She has been instructed by an SES, Bob, to look at a months-old program to see whether its planned road map is realistic. She turns to a more experienced colleague, Candice, to ask her advice. Candice asks for the program's name and quickly looks it up in Overlord's web application. Seeing that the road map in Overlord looks outdated, she asks Alice whether she can get the latest copy. Alice copies a code example from Overlord's code hub, changes the "INSERT_API_KEY" value to her own, and modifies the project ID number. A few seconds after pressing run, the program's Jira instance returns a list of major stories and their expected dates formatted as a blob of JSON. A later section of Alice's code ingests the JSON file into Overlord, checks its validity, and puts it in Overlord's database. Candice refreshes the page, and a newly updated road map chart appears. She asks whether Alice has examined any prior programs run by the program's manager. Hearing no, Candice queries Overlord for past projects involving the program manager and filters for story point data. Downloading the data as an Excel file, Candice emails the data set to Alice and says she should take a look at what the projected and actual story point velocities were. Before Alice heads back to her desk, Candice also suggests she look at the schedule estimate from CAPE's baseline model for new programs. Candice shares a link on Microsoft Teams to the live model page, which provides a dashboard where users can input a program's parameters and see the predicted likelihood of meeting the schedule from a model retrained every day on fresh data. She notes to Alice that the program manager can see the same model on his program's Jira project page, so she shouldn't expect a lot of disagreement. Alice thanks Candice and walks away.

This vignette illustrates the potential of Overlord to turn time-consuming tasks like updating data, making plots, and sharing estimate models into quick, nearly effortless activities. Following these suggestions, costs to collect and share all kinds of information can be driven down to nearly nothing. This enables a win-win mindset where developers save time while receiving helpful guidance, and overseers get constant access to troves of useful data.

What It Will Take

Overlord can build upon past investments in oversight data systems. As mentioned before, CADE has established a data warehousing functionality for traditional CSDR and other non-software-related data. Some efforts to automate data collection are also underway with the CSDR Planning and Execution Tool (cPet) that creates report templates in Excel or JSON format and validates them when completed (CADE, 2022a). Extending the capabilities of CADE to handle what I envisioned previously is not a far

stretch of the imagination. A question, then, is what role other organizations play. OUSD(A&S) owns the SAP and therefore has both a major stake in the outcome and a head start on managing SAP program data from their collection of semiannual reports. From an outsider's perspective, it seems intuitive that both organizations share a common interest in collaborating on a system like the one I propose. There is also the question of funding. One suggestion is to charge programs a small fee for using the system the way Platform One charges for managed services. The current alternative of emailing forms and waiting 210 days for an ICE would still be a free option. This approach would align the incentives of the Overlord team with the program team's desire for speed and ease of use compared to the alternative. The development of Overlord would require software engineering and data engineering talent like every other modern data initiative. Likely staffing partners include the Cost Estimating and Analysis Data Tools Tiger Team led by OSD CAPE for their data science and cost estimating expertise, as well as members of the service software factories who are well-acquainted with the state of the art in development software tooling. Like the SAP programs themselves, it is best to begin with a small team and follow the same general iterative steps (certified need statement, value assessments, road map, etc.) to build Overlord. With the right talent and development process, Overlord can quickly begin serving user needs in the agile oversight community.

References

Andreessen, M. (2011, August 20). Why software is eating the world. *Andreessen Horowitz*. https://a16z.com/2011/08/20/why-software-is-eating-the-world/

Chaillan, N. (2020, September 10). *DoD Enterprise DevSecOps Initiative and Platform One*. https://www.youtube.com/watch?v=gR0Mzo7WDss&t=2s

Cost Assessment Data Enterprise (CADE). (2022a, March 6). *CSDR Tools*. https://cade.osd.mil/tools/csdr-tools

Cost Assessment Data Enterprise (CADE). (2022b, March 6). *News*. https://cade.osd.mil/News/blog#news_20211012

Cost Assessment Data Enterprise (CADE). (2022c, March 13). *About Us*. https://cade.osd.mil/about/cade

Defense Acquisition University. (2022, March 4). *Program management metrics and reporting*. https://aaf.dau.edu/aaf/software/metrics-and-reporting/

Defense Innovation Board. (2019, May 3). Software is never done: Refactoring the acquisition code for competitive advantage.

DoD. (2020, October 2). *Operation of the software acquisition pathway* (DoDI 5000.87). Department of Defense. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF

Goljan, J. (2021). *Developing a Cost of Delay (CoD) framework for the DoD & analyzing the current state of Air Force agile cost estimation* (AFIT-ENV-MS-21-M-231) [Master of Science in Cost Analysis]. Air Force Institute of Technology. https://scholar.afit.edu/cgi/viewcontent.cgi?article=5949&context=etd

GAO. (2021, September 30). *DoD software acquisition: Status of and challenges related to reform efforts* (GAO-21-105298). https://www.gao.gov/assets/gao-21-105298.pdf

Madachy, R. (2018). *Early phase cost models for agile software processes in the U.S. DoD*. Naval Postgraduate School. https://calhoun.nps.edu/handle/10945/64421

Munroe, R. (2007, December 12). *Nerd sniping.* xkcd. https://xkcd.com/356/

Platform One. (2020). *Platform One intro (extended)* [Video]. https://youtu.be/HFTDlLIlyeA?t=32

Putnam-Majarian, T., & Staiger, J. A., Jr. (2019). Defense trend lines: Leveraging the power of historical data. In A. M. Lungu (Ed.), *Development Research Series. QSM Software Almanac* (5th ed., pp. 31–36). Quantitative Software Management, Inc. https://www.qsm.com/sites/default/files/qsm/2019 QSM Almanac.pdf

Schmidt, E., Work, R., Catz, S., Horvitz, E., Chien, S., Jassy, A., Clyburn, M., Darby, C., Ford, K., Griffiths, J.-M., Louie, G., Mark, W., Matheny, J., McFarland, K., & Moore, A. (2021, March). *Final report: National Security Commission on artificial intelligence.* National Security Commission on Artificial Intelligence. https://www.nscai.gov/wp-content/uploads/2021/03/Full-Report-Digital-1.pdf

Serbu, J. (2022, February 7). Pentagon has plan to fix its software development woes. *Federal News Network*. https://federalnewsnetwork.com/defense-news/2022/02/pentagon-has-plan-to-fix-its-software-development-woes/

Acquisition Research Program
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net