

Driving Revolutionary Change in DoD Software Design and Acquisition



https://www.acq.osd.mil/dsb/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

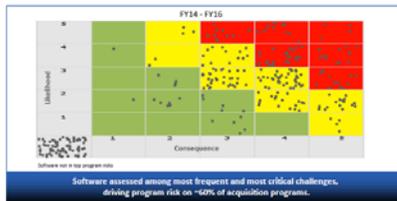


Figure 2. Software Risk Assessed by DoD Program Office

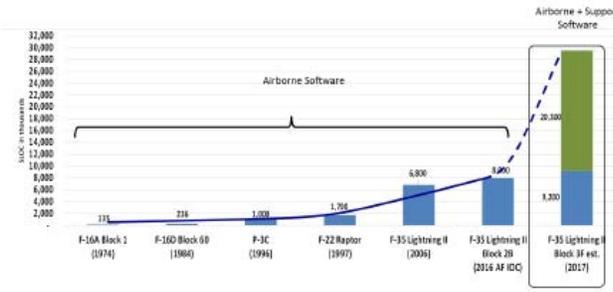


Figure 1. DoD Software Complexity and Growth: Explosive Growth of Source Lines of Code (SLOC) in Avionics Software³

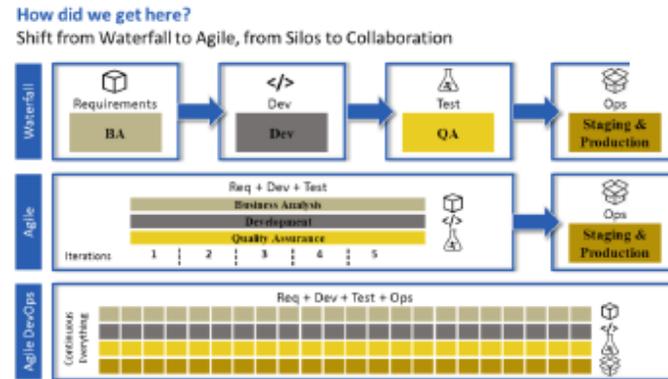


Figure 3. Theories of Software Development⁴

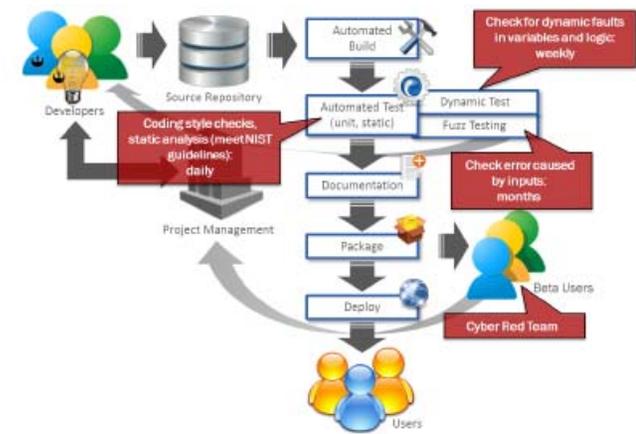


Figure 7. Addressing Cyber in the Software Factory

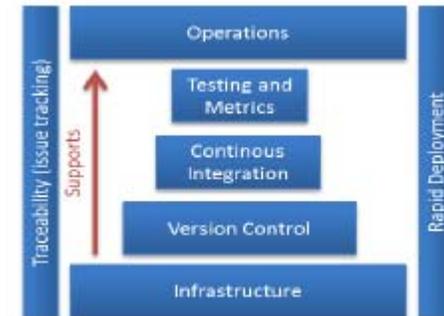


Figure E-1. Software Factory in Source Selection

PSM User's Group Workshop, 12 Sep 2018

“Measures for Iterative Software Development and Acquisition”

NDIA/INCOSE/PSM Iterative Software Development and Acquisition Working Group

Agenda



- **Introduction and Background**
 - DSB Task Force Report and Recommendations
 - Defense Innovation Board (DIB) Recommendations
 - NDIA / INCOSE / PSM Iterative Software Development and Acquisition Working Group
- **Measurement Related Findings and Actions**
- **PSM Workshop**
 - Consider appropriate measures toward industry consensus recommendations to address DSB findings

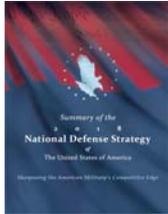
What input should we provide to DoD to support implementation of the DSB task force recommendations?

DoD and Congress are Mandating Rapid Iterative Software Development for Defense Acquisition



NDA 2019 (Sec. 868)

National Defense Strategy



<https://www.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf>

“...streamline rapid, iterative approaches from development to fielding.”

Defense Science Board (DSB) Task Force on the Design and Acquisition of Software for Defense Systems



https://www.acq.osd.mil/dsb/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

Recommendations:

- Evaluation criteria: efficacy of offeror’s SW factory
- Adopt continuous, iterative development
- Risk reduction and metrics for new programs**
- Transition for current and legacy programs in development, production, and sustainment
- Build competency in DoD and contractor workforce
- Source selection preference for delivery of SW factory framework to USG

H.R.5515 - John S. McCain National Defense Authorization Act for Fiscal Year 2019
115th Congress (2017-2018) | [Get alerts](#)

SEC. 868. IMPLEMENTATION OF RECOMMENDATIONS OF THE FINAL REPORT OF THE DEFENSE SCIENCE BOARD TASK FORCE ON THE DESIGN AND ACQUISITION OF SOFTWARE FOR DEFENSE SYSTEMS.
(a) IMPLEMENTATION REQUIRED.—Not later than 18 months after the date of the enactment of this Act, the Secretary of Defense shall, except as provided under subsection (b), commence implementation of each recommendation submitted as part of the final report of the Defense Science Board Task Force on the Design and Acquisition of Software for Defense Systems.

<https://www.congress.gov/bill/115th-congress/house-bill/5515/text>

“Not later than 18 months after the date of the enactment of this Act, the Secretary of Defense shall... commence implementation of each recommendation submitted as part of the final report of the Defense Science Board Task Force on the Design and Acquisition of Software for Defense Systems.”

Iterative Software Development and Acquisition Working Group

Advise and inform DoD on implementation recommendations (policies, guidance, RFP language, source selection criteria, training, metrics)

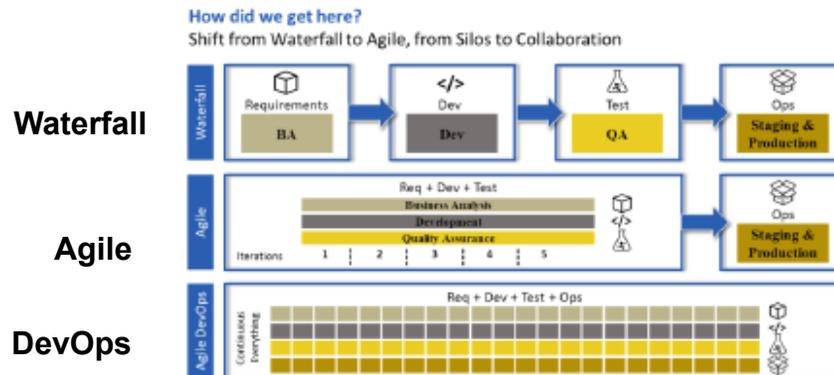


Figure 3. Theories of Software Development⁴

Iterative Software Development and Acquisition Working Group (ISDAWG)



Charter:

- Provide industry recommendations and resources to advance the use of continuous iterative software methods in DoD programs and acquisition
- Address recommendations of DSB Software Design and Acquisition Task Group

Participation:

- NDIA Systems Engineering Division, INCOSE, Practical Software and Systems Measurement (PSM)
- For participation contact the WG task leads:
Joseph.Elm@L3T.com, Geoff.Draper@harris.com, Garry.Roedler@lmco.com, Cheryl.I.jones128.civ@mail.mil

Potential Outcomes:

RFP language	<ul style="list-style-type: none">• Requirements for software factory capability, iterative SW development processes• Documentation, reviews, and CDRLs consistent with iterative development• Source selection guidance and evaluation criteria
Guidance	<ul style="list-style-type: none">• Guidance for iterative SW development (planning, architecture, design, continuous I&T, etc.)• Supplier reporting and monitoring of iterative SW development• Cost estimation techniques• Measures for iterative SW development and status monitoring• Strategies for program transition to SW iterative development methods (development and sustainment)
Education & Training	<ul style="list-style-type: none">• Recommendations for developing acquisition workforce skills for software iterative methods

Findings



DEPARTMENT OF DEFENSE | DEFENSE SCIENCE BOARD

Table of Contents

Executive Summary.....	1
1. Introduction.....	3
1.1 The Importance of Software in Defense Systems.....	3
DoD Software Growth.....	3
DoD Software Risk Assessment.....	4
1.2 Silicon Valley Baedeker: Theories of Software Development.....	5
Waterfall Development.....	6
Agile Development.....	6
Agile DevOps.....	6
Iterative Development: Agile, Spins, and Spirals.....	6
2. Finding: Continuous Iterative Development for the Department of Defense.....	7
2.1 DoD Software Processes.....	7
2.2 Commercial Software Processes.....	7
2.3 Software Factory.....	9
2.4 Addressing Cyber.....	10
2.5 Importance of Architecture.....	11
2.6 The Right Conditions for Iterative Development in Defense Systems.....	11
2.7 The Case For and Against Iterative Development for DoD Systems.....	14
3. Finding: Commercial, the DoD, and Its Partners: Case Studies.....	17
3.1 Differences and Similarities of DoD and Commercial Software Development.....	17
3.2 Defense Prime Contractors State of Play.....	18
4. Finding: Acquisition Strategies and Contracting Approaches.....	20
4.1 Software Acquisition Misalignment.....	20
4.2 Defense Acquisition Could Use Continuous Iterative Development in Many Types of Programs.....	22
Ongoing Small-scale Major Development Programs (Hybrid Model).....	22
Ongoing Large-scale Major Development Programs.....	22
New Programs.....	22
Legacy Programs.....	23
5. Recommendations.....	24
Recommendation 1: Software Factory.....	24
Recommendation 2: Continuous Iterative Development.....	24
Recommendation 3: Risk Reduction and Metrics for New Programs.....	25
Recommendations 4: Current and Legacy Programs in Development, Production, and Sustainment.....	25
Recommendation 5: Workforce.....	26
Recommendation 6: Software is Immortal – Software Sustainment.....	27
Recommendation 7: Independent Verification and Validation for Machine Learning.....	27

DSB Task Force on Design and Acquisition of Software for Defense Systems

Table of Contents | i

Continuous iterative development

SW practices in commercial industry vs. DoD and defense industry

Acquisition practices and contracting approaches

- DoD waterfall approach was largely abandoned by commercial industry years ago
- Rapid and continuous SW development is essential for quick response to adversaries
- DoD must incentivize contractor base to take advantage of modern software best practices

“The Task Force strongly believes greater adoption of continuous iterative development and its associated best practices will result in significantly improved acquisition performance.

The assessment of the Task Force is that an iterative approach to software development and sustainment is applicable to the DoD and should be adopted as quickly as possible.”

Recommendations - 1

DSB Task Force on Design and Acquisition of Software Systems



Summary of Recommendations	
<p>1. <u>Software Factory</u> – A key evaluation criteria in the source selection process should be efficacy of the offeror’s software factory.</p>	
<ul style="list-style-type: none"> Establish a common list of source selection criteria (draft App E; IDE, tools, SW, CM, issues, reqts, cloud) 	<ul style="list-style-type: none"> DoD has limited iterative development expertise – focus on acquisition
<p>2. <u>Continuous Iterative Development</u> – DoD and defense industrial base partners should adopt continuous iterative development best practices for software, including through sustainment.</p>	
<ul style="list-style-type: none"> Identify Minimally Viable Product (MVP) approaches, delegate acquisition authority to PM Engage Congress to change statutes for rapid iterative approach 	<ul style="list-style-type: none"> DAE and SAE/MDA should require for all programs entering MS-B (ACAT I,II,III) Incorporate in regular program reviews (e.g., DABs, IPRs, SRBs), with waivers only by exception
<p>3. <u>Risk Reduction and Metrics for New Programs</u> – For all new programs, starting immediately, implement best practices in formal program acquisition strategies:</p>	
<ul style="list-style-type: none"> Modernize cost/schedule estimates and measures (SLOC > historical measures, adopt NRO approach for DIB WBS schedule, staff, cost, productivity) 	<ul style="list-style-type: none"> Require PMs to build status estimation framework (e.g., burndown measures for sprints, epics, releases, velocity, control chart, cumulative flow)

Recommendations - 2

DSB Task Force on Design and Acquisition of Software Systems



Summary of Recommendations	
4. <u>Current and Legacy Programs in Development, Production, and Sustainment</u> –USD(A&S) should task PMs/PEOs to plan transition to a software factory and continuous iterative development.	
<ul style="list-style-type: none">• Prime contractors should transition execution to a hybrid model, within contractual constraints	<ul style="list-style-type: none">• Business case for transition of legacy programs where development is complete.
5. <u>Workforce</u> – The U.S. Government does not have modern software development expertise in its program offices or the broader functional acquisition workforce. This requires Congressional engagement and significant investment immediately.	
<ul style="list-style-type: none">• Services need to develop workforce competency (prioritize acquisition strategy, source selection)	<ul style="list-style-type: none">• Prime contractors must build internal competencies in modern SW methodologies, SW factories.
6. <u>Software is Immortal: Software Sustainment</u> – RFPs ... should specify the basic elements of the software framework supporting the software factory... reflected in source selection criteria	
<ul style="list-style-type: none">• Repositories; test infrastructure/tools; docs; etc.• Availability, cost, compatibility, licensing should be part of source selection criteria	<ul style="list-style-type: none">• Delivered to USG at each production milestone• Selection preference based on ability of USG to reconstitute SW framework, binaries, tests, tools.
7. <u>IV&V for Machine Learning</u> – Machine learning is an increasingly important component of a broad range of defense systems, including autonomous systems, and will further complicate the challenges of software acquisition.	

Recommendation 3: Risk Reduction and Metrics for New Programs



Recommendation 3: Risk Reduction and Metrics for New Programs

For all new programs, starting immediately, the following best practices should be implemented in formal program acquisition strategies.

The MDA (with the DAE, the SAE, the PEO, and the PM) should allow multiple vendors to begin work. A down-select should happen after at least one vendor has proven they can do the work, and should retain several vendors through development to reduce risk, as feasible.

The MDA with the Cost Assessment and Program Evaluation office (CAPE), the USD(R&E), the Service Cost Estimators, and others should modernize cost and schedule estimates and measurements. They should evolve from a pure SLOC approach to historical comparables as a measurement, and should adopt the National Reconnaissance Office (NRO) approach (demonstrated in Box 5) of contracting with the defense industrial base for work breakdown schedule data to include, among others, staff, cost, and productivity.

The MDA should immediately require the PM to build a program-appropriate framework for status estimation. Example metrics include:¹⁸

- Sprint Burndown: tracks the completion of work throughout the sprint.
- Epic and Release Burndown: tracks the progress of development over a larger body of work than a sprint.
- Velocity: the average amount of work a team completes during a sprint.
- Control Chart: focus on the cycle time of individual issues—the total time from “in progress” to “complete.”
- Cumulative Flow Diagram: shows whether the flow of work across the team is consistent; visually points out shortages and bottlenecks.

There may be short-term costs in transitioning to iterative development (e.g., software factory, training). However, based on the experience of the commercial sector, net costs can be expected to decrease after adopting iterative development.

Defense Innovation Board (DIB)



<https://innovation.defense.gov/>



ADAM M. GRANT
Professor, Wharton School of Business



DANNY HILLIS
Computer Theorist & Co-Founder, Applied Inventions



ERIC LANDER
President and Founding Director, The Broad Institute



ERIC SCHMIDT
Technical Advisor, Alphabet, Inc



JENNIFER PAHLKA
Founder & Executive Director, Code for America



MARNE LEVINE
Chief Operating Officer, Instagram



MICHAEL MCQUADE
Former SVP Science and Technology, United Technologies



MILO MEDIN
Vice President, Access Services, Google Capital



NEIL DEGRASSE TYSON
Director, Hayden Planetarium



REID HOFFMAN
Co-founder, LinkedIn & Partner, Greylock Partners



RICHARD MURRAY
Professor, California Institute of Technology



WALTER ISAACSON
President and CEO, Aspen Institute



WILLIAM H. MCCRAVEN
Chancellor, The University of Texas System



Defense Innovation Board unveils 'Ten Commandments of Software'

FedScoop - Apr 27, 2018

The Defense Innovation Board (DIB) unveiled an initial version of its "Ten Commandments of Software" at a public meeting Thursday in ...



Defense Innovation Board proposes new metrics for assessing DOD ..

FedScoop - Jul 12, 2018

The Defense Innovation Board, since it was established in April 2016, ... attention has turned to evolving the way the DOD acquires software.

DIB proposed metrics for DoD software acquisition

Defense Innovation Board added 12 new photos.
July 11 at 3:28 PM

🔥 Live: Board Deliberation on Metrics for Software Development and Acquisition Programs #SWAP Join Us:
<https://innovation.defense.gov/Meetings/>

✅ Defense Innovation Board Metrics for Software Development: https://media.defense.gov/.../DIB_METRICS_FOR_SOFTWARE_DEVELO...

SOFTWARE ACQUISITION AND PRACTICES (SWAP) STUDY
DIRECTED BY SECTION 872 OF THE 2018 NATIONAL DEFENSE AUTHORIZATION ACT

"To undertake a study on streamlining software development and acquisition regulations..."

MISSIONS' CHARGE	ENABLES	DELIVERABLE 4: DATA M
...

Defense Innovation Board – Ten Commandments of Software



WORKING DOCUMENT // DRAFT

CLEARED
For Open Publication

Defense Innovation Board
Ten Commandments of Software Apr 20, 2018 5

Version 0.14, last modified 15 April 2018 Department of Defense
OFFICE OF PREPUBLICATION AND SECURITY REVIEW

Executive Summary

The Department of Defense (DoD) must be able to develop and deploy software as fast or faster than its adversaries are able to change tactics, building on commercially available tools and technologies. Recognizing that “software” can range from off-the-shelf, non-customized software to highly-specialized, embedded software running on custom hardware, it is critical that the right tools and methods be applied for each type. In this context we offer the following ten “commandments” of software acquisition for the DoD:

1. Make computing, storage, and bandwidth abundant to DoD developers and users.
2. All software procurement programs should start small, be iterative, and build on success – or be terminated quickly.
3. Budgets should be constructed to support the full, iterative life-cycle of the software being procured with amount proportional to the criticality and utility of the software.
4. Adopt a DevOps culture for software systems.
5. Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years.
6. Every purpose-built DoD software system should include source code as a deliverable.
7. Every DoD system that includes software should have a local team of DoD software experts who are capable of modifying or extending the software through source code or API access.
8. Only run operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities.
9. Data should always be encrypted unless it is part of an active computation.
10. All data generated by DoD systems - in development and deployment - should be stored, mined, and made available for machine learning.

https://media.defense.gov/2018/Apr/22/2001906836/-1/-/1/0/DEFENSEINNOVATIONBOARD_TEN_COMMANDMENTS_OF_SOFTWARE_2018.04.20.PDF

DIB Proposed Software Metrics for DoD

https://media.defense.gov/2018/Jul/10/2001940937/-1/-1/0/DIB_METRICS_FOR_SOFTWARE_DEVELOPMENT_V0.9_2018.07.10.PDF



WORKING DOCUMENT // DRAFT

Defense Innovation Board Metrics for Software Development

Version 0.9, last modified 9 Jul 2018

Software is increasingly critical to the mission of the Department of Defense (DoD), but DoD software is plagued by poor quality and slow delivery. The current state of practice within DoD is that software complexity is often estimated based on number of source lines of code (SLOC), and rate of progress is measured in terms of programmer productivity. While both of these quantities are easily measured, they are not necessarily predictive of cost, schedule, or performance. They are especially suspect as measurements of program success, defined broadly as delivering needed functionality and value to users. Measuring the health of software development activities within DoD programs using these obsolete metrics is irrelevant at best and, at worst, can be misleading. As an alternative, we believe the following measures are useful for DoD to track performance for software programs and drive improvement in cost, schedule, and performance.

#	Metric	Target value (by software type) ⁱ				Typical DoD values for SW	
		COTS ⁱⁱ apps	Custom -ized SW ⁱⁱⁱ	COTS HW/OS ^{iv}	Real-time HW/SW ^v		
Deployment Rate Metrics	1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
	2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops) ^{vi}	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
	3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
Response Rate Metrics	4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing ^{vii}	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
	5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A	?
Code Quality Metrics	6	Automated test coverage of specs / code	N/A	>90%	>90%	100%	?
	7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
	8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
	9	% code available to DoD for inspection/rebuild	N/A	100%	100%	100%	0%
Program Management, Assessment and Estimation Metrics	10	Complexity metrics	#/type of specs structure of code		# programmers #/skill level of teams		Partial/ manual tracking
	11	Development plan/environment metrics	#/type of platforms		#/type deployments		
	12	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total \$

DRAFT GQM Measurement Framework Derived from DIB Proposed Measures



Just a potential starting point offered as input to PSM workshop...

Category	Goal	Questions	#	Metrics (DoD DIB)	Metrics (Harris rough DRAFT as input to NDIA WG)	Comments
Deployment Rate	Prioritize speed in delivering value to end users through new operational capabilities. -Automated development and deployment -Automated testing (unit level, system level) -Iterative deliver-value-now mentality	How quickly can we deliver initial capability for new products?	1	Time from program launch to deployment of simplest useful functionality. (target 6-12 mo.)	Initial capability cycle time: time from program award (startup review, or SW code start) to completion of initial v1.0 product release from factory (excluding field deployment). (weeks)	Measure: calendar weeks (plan vs. actual) Start: program award; End: v1.0 release date May be partial functionality, but must provide useful operational capability to end users. Excludes deployment to field which may not be under contractor control, therefore is not a predictable measure.
		How quickly can we add and deliver high priority capabilities for an existing operational product?	2a	Time to field high priority functions (spec > ops) (target < 3 mo.)	Incremental capability cycle time: cycle time for release of incremental product capability enhancements (v1.x) Deployment frequency.	Target: <3 mo Measure: calendar weeks (plan vs. actual) Start: sprint start for new operational capability; End: baseline release for validated new operational capability outcome from the factory baseline release. Excludes deployment to field which may not be under contractor control, therefore is not a predictable measure.
		How quickly can new security vulnerabilities be patched and deployed to fielded products?	2b	Time to fix newly found security hole (find > ops)	Patch cycle time: time for patching a new security vulnerability and releasing a new operational baseline, (v1.x.y))	Target: <1 wk Measure: calendar days Start: receipt of vendor vulnerability patch; End: release/deployment of patched and verified baseline update. Verification and accreditation of patches is conducted as an outcome of the software factory process. Time from identification of vulnerability (e.g., SANS) to availability of patch from the vendor (e.g., Oracle) also impacts overall end user cycle time, but is not under contractor control and is separate from this metric.
		What is the "lead time" duration from code committed to a repository to availability of tested functionality?	3	Time from code committed to code in use (target <1 mo)	Factory cycle time: time from final code development submission to CM through factory build, I&T, validation and availability of tested baseline release.	Target: <1 mo Measure: calendar days Start: submission of final code to CM for build; End: baseline release of validated capability ready for operational deployment (outcome from factory). A necessary condition for rapid evolution of delivered SW functionality. Shorter product delivery times demand faster feedback, which enables tighter coupling to user needs. Deployment time to the field is not wholly under contractor control, so is excluded from this metric but does affect availability to users.

...