coloradodigitalservice / **modern-software-delivery-maturity-index** Public

A resource to help teams evaluate and improve their software delivery.

CC0-1.0 License

14 stars 0 forks

Star Notifications

Code Issues 7 Pull requests Discussions Actions Projects Wiki Security Insig

main Go to file

skcain Merge pull request #12 from coloradodigitalservice/accessibility ... on 28 Oct 2021 8

View code

README.md

Modern Software Delivery Maturity Index

Category	Low	Medium	High
Team Culture	• Lack of accountability • Confusion over roles	• Silos may still exist • Honest conversations are happening but still feel difficult or combative	• Cohesive team dynamic • High levels of psychological safety and trust • Team dynamics marked by collaboration and transparency,

	• Lack of transparency • Working in silos • Culture is described with words such as fear, apathy, toxic • Finger-pointing and scapegoating are common	• The team is willing to acknowledge dysfunctions and working to improve dynamics	with effective retros spurring continuous team improvement • Low levels of turnover/burnout
Purchasing v. Procurement	• Focused on procedural; protocol and risk aversion • Acquisitions are one dimensional and transactional • Purchases occur without strategy or consideration of value • Performance metrics prioritize outputs over outcomes • Vendor diversity is stagnant; failed	• Primarily focused on purchasing, stretched into procurement • Limited options for strategy • Default to prior vendors, templates, and approaches • Multi-factor evaluation rubrics used • Pursuing strategic procurement occurs because of law and regulatory requirements; it is not pursued as a value add activity • Finds new vendors	• Focus on the problems, i.e., what are we trying to solve? Why is a good or service acquired? • Proposals include dynamic criteria (i.e., proofs of concept, pilots, code samples) • Manages key relationships by aligning policy and program goals with procurement strategies and ongoing contract administration • Creates mechanisms to enhance vendor diversity and success • Focus on best value-added, total cost of ownership

		value delivery is consistent. • Focus on the lowest cost per unit		
	Modular Contracting	• No understanding of diverse contracting strategies and negotiation methodologies. • One contract/vendor per project/system is the default norm • IT vendors are overwhelming the same 3-6 vendors and problems persist from project to project without improvement. • No multi-vendor projects active • Unwillingness to consider Open	• Some systems have strong state technical leadership with vendor augmentation • A small number of systems have multiple vendors working on them • Solicitations occasionally seek specialized skills to supplement the existing team • Reluctance to consider Open Source development practices • Difficulty contracting with partners who are willing to work in the open	• A rich pool of vendors frequently compete for modules of systems • Large ongoing systems have multiple vendors supporting different components collaboratively • Seamless handoff between vendors • The state retains ownership and autonomy over systems or State retains direction and autonomy on how a purchased product can be curtailed, modified, and work in service of meeting organizational needs • Defaults to Open Sourcing as much code as possible

		Source development		
	User-Centered Approach	• Confusion or lack of understanding of who will use the output of the project or who will be ultimately impacted • Work is performed with a "solution-first" focus; Assumptions are stated as fact • No engagement with the people that will use the output of the project • Decisions related to priority and implementation are driven primarily by a stakeholder that is close to the project	• A shared understanding of who will use the output of the project; lack of understanding of who will be impacted • Work begins with a "solution-first" focus; Assumptions are stated as fact • Living experts are consulted at the beginning or end of the project, or living experts are consulted so much that they are overwhelmed. • Information collected from living experts is not shared amongst the entire team. • Decisions related to priority and implementation are driven by a best guess of what end-users need	• A shared understanding of the people that will engage and/or are impacted by the output of the project • Work begins with a "problem-first" focus and the team identifies the best path forward; The team regularly evaluates their assumptions • Living experts are regularly engaging with the team in order to drive prioritization and implementation decisions and are properly compensated for their time • Ongoing incremental value is delivered to people who will use the output of the project; shared understanding of the impact • People regularly engage with the output of the project as expected • The output of the project is accessible and equitable for all people who engage with it

		• The team identifies important considerations or constraints late in the process, requiring them to conduct re-work • End-users feel frustrated, disempowered, and alienated • The output of the work has many barriers to access • No authorization to spend time or money on user research	• People engage with the output of the project, though not quite as expected • The output of the work is accessible but has other barriers that prevent people from meaningfully engaging with it (eg. only available in English) • No authorization to spend money on user research	
	Product Ownership	• No Product Owner (PO) in place • PO role is filled by a project or program manager, or individuals who fulfill other roles	• PO role may exist • The product team is engaging regularly with end-users • The product team has access to a single backlog • PO is not yet set up for success in key ways	• PO role exists • Product vision/strategy is clear • PO has decision-making authority • PO is able to effectively prioritize work & has access to good data and analytics • The team is aligned with a product-oriented mindset focused on outcomes, not outputs

		• Product is looked at as a project or collection of projects	(other dimensions of this maturity matrix are low)	
	Agile Software Development	• Using some agile terminology, but really doing something like iterative waterfall approach • Variable work quality • Growing tech debt • Infrequent high-stakes releases • Missing feedback loops	• Some elements of agile are working, but a fully cross-functional approach still missing • Infrequent but regular release cadence • Feedback loops irregular or inconsistently used	• Agile is well understood and embraced by the project team and stakeholders • Right resources/skill sets are in place • Continuous organizational learning and optimization of work processes • Regular low-stakes releases, CI/CD
	DevSecOps	• Lack of automation in testing • Lack of automation in deployment • Manual server config	• Some developer documentation exists but is not consistent • Onboarding a new developer takes a while and isn't straightforward • Many automated tests exist though may not	• Repeatable processes in place with high-quality results • Secure, documented, code released every sprint to production • The system is stable for end-users and regularly updated without impact • Well-defined security model allowing for appropriate access to

	• Security is a checklist, at best • No Continuous Integration	run regularly	State employees, contractors, and volunteers as needed.
Building With Loosely Coupled Parts	• The system is monolithic, if one thing breaks, it impacts the whole system • The system architecture knowledge is not easily shared across teams • New features must be deployed across all systems at the same time.	• One or more parts of the system are loosely coupled • Teams work together in maintaining the systems • Some features can be deployed in one system independently of whether other systems are ready	• The system is composed of logical modular components that all talk to one another • Failures in components are handled gracefully when a single system goes offline • Different teams/vendors are successfully able to take ownership of individual modules • Features regularly ship in different codebases independently of each other

This maturity index was derived from guidance produced by 18F, the UK.gov digital team, and the Harvard Kennedy School of Public Policy.

Assembled and published by the Colorado Digital Service.

Releases

No releases published

Packages

No packages published

Contributors 2


greggles Greg Knaddison


skcain Stephanie