DIB Guide: Detecting Agile BS

Agile is a buzzword of software development, and so all DoD software development projects are, almost by default, now declared to be "agile." The purpose of this document is to provide guidance to DoD program executives and acquisition professionals on how to detect software projects that are really using agile development versus those that are simply waterfall or spiral development in agile clothing ("agile-scrum-fall").

Principles, Values, and Tools

Experts and devotees profess certain key "values" to characterize the culture and approach of agile development. In its work, the DIB has developed its own guiding maxims that roughly map to these true agile values:

Agile value	**DIB maxim**
Individuals and interactions over processes and tools	"Competence trumps process"
Working software over comprehensive documentation	"Minimize time from program launch to deployment of simplest useful functionality"
Customer collaboration over contract negotiation	"Adopt a DevSecOps culture for software systems"
Responding to change over following a plan	"Software programs should start small, be iterative, and build on success – or be terminated quickly"

Key flags that a project is not really agile:

- Nobody on the software development team is talking with and observing the users of the software in action; we mean the *actual* users of the *actual* code.[14] (The PEO does not count as an actual user, nor does the commanding officer, unless she uses the code.)
- Continuous feedback from *users* to the development team (bug reports, users assessments) is not available. Talking once at the beginning of a program to verify requirements doesn't count!
- Meeting requirements is treated as more important than getting something useful into the field as quickly as possible.
- Stakeholders (dev, test, ops, security, contracting, contractors, end-users, etc.)[15] are acting more-or-less autonomously (e.g., 'it's not my job.')
- End users of the software are missing-in-action throughout development; at a minimum they should be present during Release Planning and User Acceptance Testing.

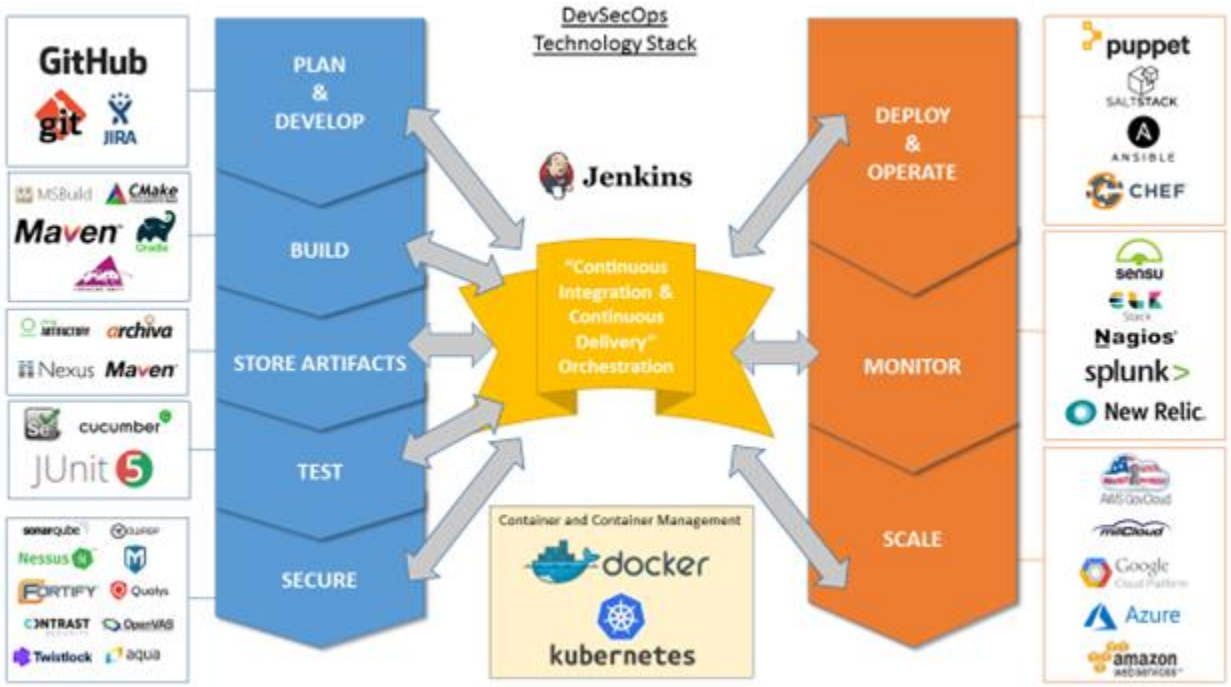
[14] Acceptable substitutes for talking to users: Observing users working, putting prototypes in front of them for feedback, and other aspects of user research that involve less talking.
[15] Dev is short for development, ops is short for operations

- DevSecOps culture is lacking if manual processes are tolerated when such processes can and should be automated (e.g., automated testing, continuous integration, continuous delivery).

Some current, common tools in use by teams using agile development (these will change as better tools become available):[16]

- Git, ClearCase, or Subversion - version control system for tracking changes to source code. Git is the *de facto* open source standard for modern software development.
- BitBucket or GitHub - Repository hosting sites. Also provide issues tracking, continuous integration "apps" and other productivity tools. Widely used by the open source community.
- Jenkins, Circle CI or Travis CI - continuous integration service used to build and test BitBucket and GitHub software projects
- Chef, Ansible, or Puppet - software for writing system configuration "recipes" and streamlining the task of configuring and maintaining a collection of servers
- Docker - computer program that performs operating-system-level virtualization, also known as "containerization"
- Kubernetes or Docker Swarm for Container orchestration
- Jira or Pivotal Tracker - issues reporting, tracking, and management

Graphical version:



Questions to Ask Programming Teams

[16] Tools listed/shown here are for illustration only: no endorsement implied.

- How do you test your code? (Wrong answers: “we have a testing organization,” “OT&E is responsible for testing”)
    - Advanced version: what tool suite are you using for unit tests, regression testing, functional tests, security scans, and deployment certification?
- How automated are your development, testing, security, and deployment pipelines?
    - Advanced version: what tool suite are you using for continuous integration (CI), continuous deployment (CD), regression testing, program documentation; is your infrastructure defined by code?
- Who are your users and how are you interacting with them?
    - Advanced version: what mechanisms are you using to get direct feedback from your users? What tool suite are you using for issue reporting and tracking? How do you allocate issues to programming teams? How to you inform users that their issues are being addressed and/or have been resolved?
- What is your (current and future) cycle time for releases to your users?
    - Advanced version: what software platforms to you support? Are you using containers? What configuration management tools do you use?

Questions for Program Management

- How many programmers are part of the organizations that owns the budget and milestones for the program? (Wrong answers: “we don’t know,” “zero,” “it depends on how you define a programmer”)
- What are your management metrics for development and operations; how are they used to inform priorities, detect problems; how often are they accessed and used by leadership?
- What have you learned in your past three sprint cycles and what did you do about it? (Wrong answers: “what’s a sprint cycle?,” “we are waiting to get approval from management”)
- Who are the users that you deliver value to each sprint cycle? Can we talk to them? (Wrong answers: “we don’t directly deploy our code to users”)

Questions for Customers and Users

- How do you communicate with the developers? Did they observe your relevant teams working and ask questions that indicated a deep understanding of your needs? When is the last time they sat with you and talked about features you would like to see implemented?
- How do you send in suggestions for new features or report issues or bugs in the code? What type of feedback do you get to your requests/reports? Are you ever asked to try prototypes of new software features and observed using them?
- What is the time it takes for a requested feature to show up in the application?

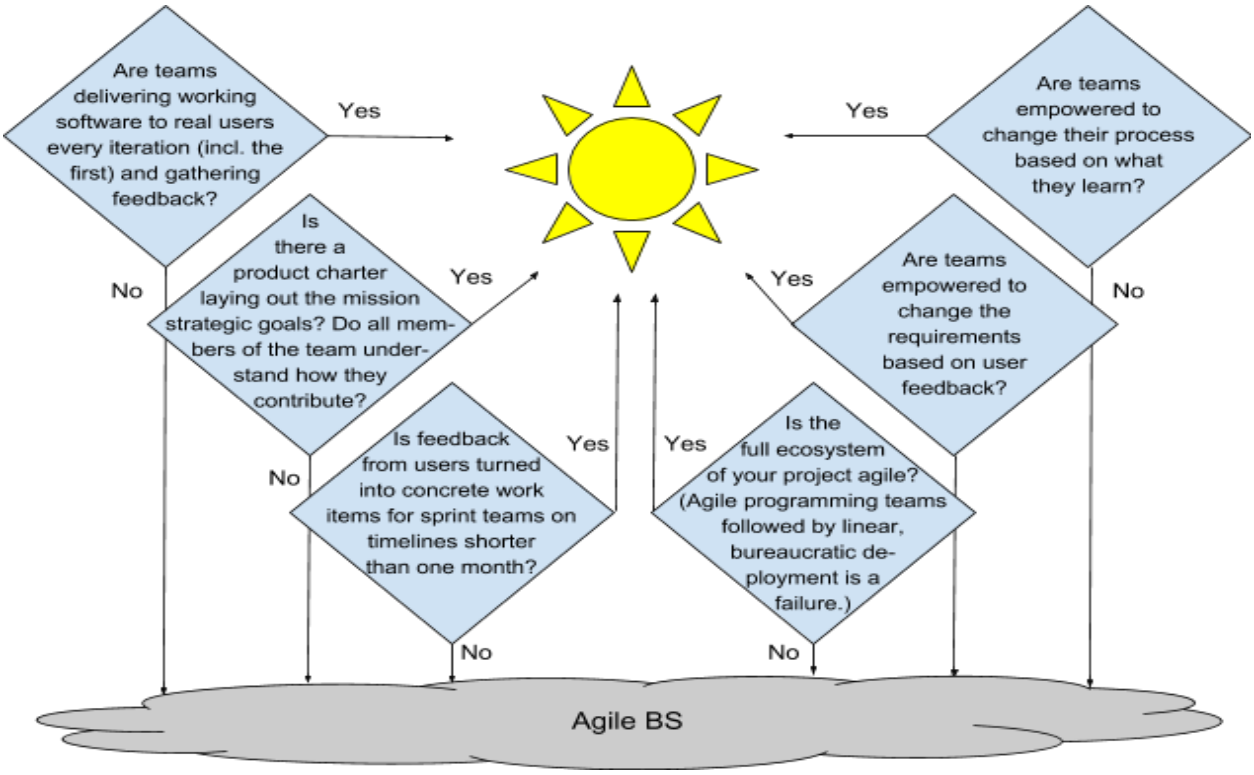
Questions for Program Leadership

- Are teams delivering working software to at least some subset of real users every iteration (including the first) and gathering feedback? (alt: every two weeks)

- Is there a product charter that lays out the mission and strategic goals? Do all members of the team understand both, and are they able to see how their work contributes to both?
- Is feedback from users turned into concrete work items for sprint teams on timelines shorter than one month?
- Are teams empowered to change the requirements based on user feedback?
- Are teams empowered to change their process based on what they learn?
- Is the full ecosystem of your project agile? (Agile programming teams followed by linear, bureaucratic deployment is a failure.)

For a team working on agile, the answer to all of these questions should be “yes.”

Graphical version:



More information on some of the features of DoD software programs are included in the DIB’s “Ten Commandments of Software,” the DIB’s “Metrics for Software Development,” and the DIB’s “Do’s and Don’ts of Software.”