Implementing DevOps Practices in Highly Regulated Environments

Jose Andre Morales, Hasan Yasar, Aaron Volkman
Software Engineering Institute, Carnegie Mellon University

ABSTRACT

In this paper, we discuss implementing DevOps practices in highly regulated environments (HREs). DevOps has become a standard option for entities seeking to streamline and increase participation by all stakeholders in their Software Development Lifecycle (SDLC). For a large portion of industry, academia, and government, applying DevOps is a straight forward process. There is, however, a subset of entities in these three sectors where applying DevOps can be very challenging. These are entities mandated by policies to conduct all or a portion of their SDLC activities in HREs. Often, the reason for an HRE is general security and protection of intellectual property. Even if an entity is functioning in a highly regulated environment, its SDLC can still benefit from implementing DevOps as long as the implementation conforms to all imposed policies. In this paper, we discuss the process of performing a DevOps assessment and implementation in an HRE which we refer to as HRE-DevOps.

CCS CONCEPTS

• **Software and its engineering → Rapid application development**;

KEYWORDS

DevOps, Highly Regulated Environment, Secure DevOps, SDLC, DevOps Assessment

ACM Reference Format:
Jose Andre Morales, Hasan Yasar, Aaron Volkman. 2018. Implementing DevOps Practices in Highly Regulated Environments. In *Proceedings of International Workshop on Secure Software Engineering in DevOps and Agile Development (SecSE 2018).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

A highly regulated environment [8, 10, 13, 14, 18] (HRE) is typically characterized by the following: air-gapped physical spaces and computer systems with heightened security and access controls, segregation of duties, inability of personnel to discuss certain topics outside of closed areas, and the inability to take certain artifacts off premises. In this paper, air-gapped is meant to be a physical space, personnel, computer system, and other technology which are isolated from the rest of the HRE and all HRE external entities. We have mentioned only some of the characteristics of an HRE as the list changes on a case by case basis. This definition of an HRE is not the same as government regulation where policies focused on how to conduct business, financial responsibilities, and disclosure filing, just to name a few, is required for various sectors of industry and overseen by federal agencies such as the U.S. Securities & Exchange Commission [6], the U.S. Food and Drug Administration [5], and the Federal Communications Commission [4].

Each of the previously mentioned obstacles characterizing an HRE can impose several barriers impeding the full incorporation of DevOps [7, 15] practices into a Software Development Lifecycle (SDLC) [22]. In this paper, we focus on implementing the following DevOps principles: Open communication between all stakeholders, Infrastructure as Code (IaC), environment parity, centralized documentation, small task completion and deployment, accurate production environment replication, end user feedback loop, and software artifact versioning. We determined these principles should be present in any DevOps implementation including HREs and an appropriate focus of this work. In general, HREs promote isolation and gaps between persons and projects which is in direct contrast to DevOps whose main goal is to establish open communication between all members and stakeholders of a project. In order to implement DevOps practices in an HRE, the current state of the SDLC process in that HRE must be clearly understood. This can be accomplished as part of a DevOps assessment [11]. Once the assessment is completed, obstacles impeding the current SDLC process should be identified and analyzed one by one to determine if DevOps practices can overcome the barrier. A list of recommendations should be submitted to relevant project stakeholders who must verify which recommendations are implementable within all the boundaries of applicable policies [9]. In some cases, a recommendation may fall outside the scope of DevOps practices and this is acceptable. These recommendations should still be submitted since they are part of the reality of customizing typical DevOps practices into tailored versions benefiting a specific HRE situation. We refer to the process of assessing, implementing and customizing DevOps practices for the SDLC of an HRE as HRE-DevOps. The remainder of this paper discusses each of the steps in the process of assessing and implementing DevOps practices in an HRE.

2 HRE ASSESSMENT

Before commencing an assessment, expectations should be set with HRE personnel. From the beginning, HRE personnel should know that DevOps practices will likely not solve all issues faced in their SDLC. Instead it should be made clear that DevOps practices will be implemented, as much as possible, in a customized manner within imposed policy constraints. Another key point is the assessment of an HRE will include recommendations that may not be part of a DevOps practices but is essential to improving the SDLC. The overall assessment goals should be:

SecSE 2018, May 2018, Porto, Portugal

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

(1) Identify bottlenecks and pain points that occur within the HRE when executing the SDLC.
(2) Recommend DevOps solutions as broadly as possible.
(3) Recommend non-DevOps solutions when improvements cannot be achieved with DevOps approaches.
(4) Establish the HRE's DevOps posture [11] and identify SDLC components which need improvement in order to consider implementing DevOps practices.

It is critical, at the commencement of an assessment, to clearly detail what will be assessed. This includes, personnel, systems, and processes. The focus of the assessment discussed in this paper is to improve the SDLC in an HRE. This includes: current lifecycle process being implemented, technology being used in that process, professional interactions of personnel regarding the process. The culture of the SDLC creates an environment which can include assumptions, beliefs, default decision making, group discussions, teamwork, and reliance between individuals. It is non-trivial to understand the culture of a group of people implementing the SDLC, especially in an HRE [13]. One way to gather understanding, besides interviews, is to observe the personnel in performing daily work tasks on an actual project, asking questions to clarify doubts, and documenting every step of one member for each project role type.

When assessing an HRE, a lot of time must be spent in interviewing personnel, primarily developers, to understand how the SDLC process is currently implemented [19]. Keep in mind the SDLC can be implemented very differently for each HRE, and for different development groups within one HRE. The assessors must understand every HRE's process and obstacles in detail including imposed policies affecting the SDLC in some way. It is critical to understand the policies that dictate that certain components of the SDLC be carried out in a specific way. When assessing a software development group in an HRE, consider the following questions:

(1) How is stakeholder communication accomplished, how often, by what means?
(2) Do all requirements come at the beginning of a project or are some added after work has begun?
(3) How is artifact delivery carried out?
(4) How are HRE external stakeholders given access to artifact documents and project progress?
(5) Is there a feedback loop between stakeholders and end users in the production environment?
(6) Are staging environments used always, never, or only when the production environment is inaccessible?
(7) Can changes be made to an artifact during and after initial deployment?
(8) What is the project authorization process?
(9) What other bottlenecks, not covered here, are causing delays to project commencement and completion?
(10) What is the hardware acquisition process?
(11) Are there any non-HRE development activities contributing to the project and how are they managed?

The process of carrying out an assessment should include several onsite visits, shadowing of daily work and conference calls. Assessor led meetings with HRE personnel should occur to discuss:

Assessor observations of the SDLC process. When presenting an assessor's observations to HRE personnel, it is best to divide the observations by the components of the SDLC. The assessor should first describe their understanding of the process for the specific component and validate its correctness with HRE personnel. For each component, the assessor should point out observations considered standard SDLC practice, observations considered non-standard practice, and expected standard practices not observed. Standard SDLC practices greatly facilitate implementing DevOps since those practices function the same as a non-HRE entity. When a non-standard SDLC practice is observed, HRE personnel should explain in detail why it is in use, the unique conditions that led to its use, and more importantly, if it the result of imposed policies or other measures unique to the HRE. Identifying standard and non-standard practices is based on the assessor's knowledge of observing SDLC implementations in non-HRE entities. Attempt to establish if the practice is in use because it provides a benefit to the SDLC. In this case, consider if replacing it with a DevOps approach can achieve the same or improved benefits.

Obstacles stated by HRE personnel. As discussions about the current SDLC implementation progress, an assessor can expect, in most cases, HRE personnel to quickly reveal the pain points and bottlenecks (collectively referred to as obstacles) making their job harder. In some cases, the personnel are eager to discuss these in search of a solution and should be given high priority. The assessor needs to craft a precise statement that describes the obstacle and validate its correctness with personnel. It must be mutually agreed if a stated obstacle is out of scope of the SDLC. Depending on the scenario, these particular obstacles may not be addressed, and this should be made clear to the HRE personnel. The out of scope items should be documented and included in the final assessment report. For each obstacle, the assessor must determine if it exists due to conditions stemming from imposed polices, the HRE, operational concerns, or some other reason. Unless informed otherwise by HRE personnel, an assessor should assume these reasons are non-negotiable and removing or ignoring them is not an option.

Inferred obstacles observed by the assessor. It is possible that an HRE can have obstacles that personnel do not recognize. In observing the day to day operations of the HRE, an assessor can discover a process negatively affecting the SDLC. The HRE personnel may not be aware of this and not consider it an obstacle. It is important, in these cases, to point out the observed obstacle and its negative impact. The assessor must make clear to personnel why this is an obstacle and detail its negative impact on the SDLC process along with potential benefits from its removal or modification, assuming its existence is not the result of a mandatory imposition. Only if the HRE personnel agrees this is an obstacle which they have overlooked can it be addressed. If there is disagreement, the inferred obstacle should be listed in the final report.

An effective assessment occurs when HRE personnel feel comfortable in open conversation with assessors inside an HRE. This implies no negative impact or retaliation can occur for expressing their concerns and frustrations. Achieving this can be done in negotiations with HRE management. The most effective people to engage are the actual developers performing the day to day tasks of the SDLC for various projects. These persons are closest

to the work, and they are most aware of the obstacles impeding their productivity. Assessments will benefit from the availability of diagrams, manuals, and any other documentation detailing the SDLC process. Reviewing the HRE's software development process document (SDP) and any and desk instructions for developers can give great insight on their current SDLC process and if the day to day activities of developers follows what is instructed in these documents. Walking through an actual completed or ongoing project facilitates the discovery and detailing of obstacles in the SDLC. Any discovered differences between daily work routines and documented processes should be included in an assessment if it impacts the SDLC in a negative way.

3 ANALYZING SDLC OBSTACLES

An obstacle can be described as a composition of three parts: 1. Description of the obstacle, 2. ,the reason for the obstacle's imposition and 3. the method in which the obstacle is imposed. A method can be characterized as a series of actions that when followed enforces the obstacle. The method and reason are critical when determining which obstacles can and cannot be addressed with assessor recommendations. If the reason for an obstacle is policy imposition or obligatory conditions within the HRE, the obstacle cannot be removed. If any other reason is given for an obstacle, it can be considered for removal or modification. Often, actions can be streamlined while still enforcing the obstacle. This does not hold true in specific cases where a single or series of actions are imposed by policy or HRE conditions. Here are some examples of obstacles:

Obstacle 1 Description: No code development details can be shared with operators. **Reason:** Security concerns over proprietary tactics. **Method:** Implement air gapped environments for the physical locations and computer networks of the development and operation teams. **Actions:** 1. Place each team in physically separate buildings 2. Isolate the developer and operator networks by using different IP ranges with internal only access 3. Provide individual email servers filtering out communication between members of different teams 4. Developer software licensing must not be shared between teams. **Reason why it's an obstacle:** Developers cannot discuss during code development if the approaches they are using will be effective when in use. This causes potential for crashes or unexpected side effects that could be avoided if discussions were allowed.

Obstacle 2 Description: All physical software artifacts related to a project must be stored in a single isolated and secure location. **Reason:** Facilitates retrieval of physically secured archived projects for the development team **Method:** Designate a multi-shelf archive closet in an isolated area with badge access. **Actions:** 1. Identify an isolated room with badge access not shared by the development team 2. Clearly mark âĂIJDevelopment Team Archive ClosetâĂİ 3. Provide access during 9am-5pm to all developers **Reason why it's an obstacle:** Developers work well into the night and cannot access artifacts after 5pm. This holds back project progress and/or completion.

Obstacle 3 Description: Projects must be approved by a member of the senior leadership prior to commencement. **Reason:** Administrative management policy **Method:** Member of senior leadership is given project description for approval with signature. **Actions:** 1. Stakeholder provide needs to development team 2. Team formalizes needs into mutually agreed project document containing description and requirements 3. Document officially sent to senior leadership for approval 4. Work commences upon approval receipt. **Reason why it's an obstacle:** A project's approval could have very long delays. Project completion is due within a specified time and there is no compensatory time added for delayed approval. This drastically cuts into the time left to complete a project with developers rushing. This results with, in some cases, an inferior final result.

Obstacle 4 Description: The operational testing environment is air gapped from the development team **Reason:** The operational testing environment is highly secured due to its security requirements **Method:** Approved software changes are hand carried into the operational testing environment on portable media by security personnel **Actions:** 1. Software development team member submits a software change request 2. The software change is packaged by a configuration management team 3. The software change is vetted by the security team to determine its security posture 4. The configuration management team copies the software change onto portable media 5. The physical security team transports the portable media into the operational testing environment **Reason why it's an obstacle:** A software change workflow that involved multiple teams performing manual processes can add multiple days to a deploy-test cycle in the SDLC.

The above summarizes some sample obstacles following our definition of an obstacle's components. Note, the last section labeled "Reason why it's an obstacle" serves to document why this is an obstacle to the SDLC and is not strictly a part of our definition of an obstacle description but should be included to clearly state the reasons why this has been deemed an obstacle. Obstacle 1 is in place to protect personally identifiable information [20] (PII). It is not stated to be a policy so it may be modifiable or removable. Closely reading the actions, it becomes clear that some of these cannot be removed due to HRE conditions. In this case, the actions taken to implement air-gapped environments are governed by their own set of imposed policies likely not removable or modifiable. Obstacle 2 is imposed by the development team's desire to easily access artifacts that are physically secured. This is not a policy and actions can be modified to overcome the obstacle. The reason for obstacle 3 is imposed policy. The actions used to enforce are not themselves policy imposed or part of an HRE. There may be ways to modify the actions to better deal with this obstacle and ensure adherence to the policy but these will need to be discussed and agreed by HRE personnel. Obstacle 4 is isolating highly secure systems and operators. It is enforced by the HRE and the actions are likely mandated by policy. There may be few possible modifications in this obstacle, the assessment team will have to analyze closely with HRE personnel. This small sampling illustrates the importance of understanding the reasons for an obstacle's existence and the actions enforcing it. In all four cases the stated reasons are the critical component to decide if the obstacle can be addressed in some way or not at all.

When interviews and onsite visits are completed, a list of agreed upon obstacles is presented. For each obstacle in the list, the assessment team must decide if a DevOps approach can be put in place to

address it or if some other non-DevOps solution can be used. Here is a list of some obstacles that can be expected to exist in an HRE:

(1) Air-gapped environment
(2) Slow project approval
(3) Slow hardware and software acquisition
(4) No software version control
(5) No centralized document repository
(6) No communication with stakeholders
(7) Loss of project work time
(8) Partial or no production environment access
(9) New requirements late in SDLC
(10) No centralized software installation
(11) High attrition of contract personnel
(12) Slow IT request fulfillments

The above list generalizes the obstacles that could be present in an HRE. It is not exhaustive and each of the development groups within an HRE will have specific circumstances leading to unique obstacles. Once each obstacle has been reviewed and documented, the next step is to make recommendations.

4 DEVOPS AND NON-DEVOPS RECOMMENDATIONS

The goal of an assessment is to remove or alleviate the obstacles HRE personnel face when executing the SDLC. In some cases, DevOps can help reach that goal. In other cases, non-DevOps approaches are appropriate. This mix of approaches is prevalent in HREs. The assessment team should focus on applying DevOps followed by non-DevOps recommendations. It should not be the policy of an assessor to leave obstacles unaddressed if a DevOps approach cannot be applied. Below we discuss possible recommendations for the generalized obstacle list presented in the previous section.

Air-gapped environment. A key component of all HREs is areas that are isolated or air-gapped [1–3, 16, 21]. These silos separate personnel and technology from other groups of the same entity and outsiders. These environments are in direct contrast to cooperation and inclusion of all stakeholders throughout the SDLC process, which is a cornerstone of DevOps principles. Due to imposed policies regarding security, it will be difficult to include all stakeholders throughout the SDLC. One approach is inclusion of persons allowed to be in the HRE representing a stakeholder that is not allowed. These persons can create reports for the stakeholder, after HRE security and management approval, for external distribution. HRE's security focus is on outflow of information and less on inflow. This can allow stakeholders to openly contact HRE personnel, via email or phone, with questions and requests. Personnel can reply within the bounds of security policies. Assessors should ask the details of what is allowed and not allowed in both physical and digital silos. It may be the case that a stakeholder is not allowed physically in a silo but is allowed to view siloed materials online. In the case where a stakeholder is disallowed in both physical and digital, then a representative may be needed. If that is not possible, arrangements can be made for reoccurring meetings in a non-HRE to provide HRE approved updates and status.

Slow project approval. In an HRE, most projects must be submitted via a formal process up a management chain to a senior leadership member who provides approval. This can be a very time-consuming process that absorbs work cycles no longer available to developers. Time is further consumed when multiple senior members claim authority and debate who should rightfully approve a project. Typically, time is not added on to a project schedule when this occurs and developers are rushing to complete the work. This breaks several DevOps principles, primarily the ability to deploy multiple versions of a software during development for testing and feedback. One approach to overcome this is designating a middle management member as a proxy approver for projects. This person is physically located with the developers and can quickly provide authorization following guidelines provided by the senior member. The project is simultaneously sent via the normal approval route while work commences in a timely manner. Since the proxy approves projects based on predetermined guidance by the senior member, chances of a rejection are low. Another approach is to empower the stakeholder requesting the project to approve work commencement. This is possible in scenarios where the stakeholder is authorized to use a source of funding for these projects and is approved to task the development team.

Slow hardware and software acquisition. HREs may have to deal with hardware and software acquisition processes burdened with several approvals and processing steps. These typically occur at the beginning of a project, but could happen at any point, and can significantly delay or consume time. In general, acquisition is not related to DevOps. Solutions to improving the internal acquisition process would be non-DevOps. Dealing with the obstacle can, on the other hand, be eased with DevOps concepts. At the requirements stage of the SDLC, a review of needed software and hardware should be conducted between all parties and led by the developers and end users. Based on the agreed upon needs, extra time can be budgeted to allow for acquisition thus not affecting the development time. A good DevOps environment, even in an HRE, is equipped with modern development technologies. Being DevOps proactive requires developers to explore new tools and techniques as they arise along with creating their own. Adding new tools and techniques can eliminate the need for software and hardware acquisition to cases where very specific items are required.

No software version control. Maintaining custom software and its dependencies in a version control system (VCS) is critical to a successful DevOps implementation. There is a large selection of VCSs available. In some cases, the developers may deliver multiple versions of the same product to different entities. Due to project urgency or imposed policy, those products may lack clearly marked version numbers. This can create a scenario where various groups are testing different versions of the software resulting in mixed feedback causing delays to project completion. A solution is based on how project artifacts are delivered: if via a repository where others download from, then that repository should, at a minimum, have versioning on the developer side and assure users can only download the latest version. This should be accompanied with alerts and updates either in the repository, email, or other communication channels to users informing they may have an older version and consider updating. Depending on the reason for this obstacle, these actions can be less restrictive to the point of labeling new version numbers into the filenames directly. If delivery is made

via a physical medium such as a USB drive, CD, or DVD, ideally it should be marked with a version number. If that is not possible, the developers should assure the items on the device are the expected version. An alternative is to use random numbers for software versions. The number generation method should assure no repetition for large amounts of generated numbers. Recipients will still know if two products are different versions based on the random number. A mapping ability to actual version numbers should be accessible. These devices need to be version marked in some way to avoid confusion if several are stored in one place. An end user should not have to insert a device to discover the version number. In ideal cases, implementing the complete DevOps versioning concept, all stakeholders have access to a centralized repository storing downloadable historical versioned archives of project artifacts.

No centralized document repository. During the SDLC, project stakeholders will create notes, updates, requests, instructions, guidance, and many other types of documents. These documents are often stored in various locations. This results in stakeholders, primarily end users, accessing obsolete documentation on the project that do not include the latest updates, features, installations and usage instructions. In a DevOps process, centralized document repositories are critical to provide visibility to all stakeholders. One approach is to provide outside stakeholders with documentation approved for external use. Another option is to electronically expose the repository's documentation to specific external stakeholders via secure networks having appropriate security access controls. Stakeholders will have access only to what they need to read. Requests can be made for other documentation that can be made available with appropriate approvals.

No communication with stakeholders. Some stakeholders do not have a direct line of communication with the personnel developing their product in an HRE.. This breaks the fundamental DevOps principle of involving all appropriate stakeholders in their applicable steps in the SDLC. The problem is critical in relation to project requestors and end users. An absent requestor can lead to vague or unverified project requirements. An absent end user can result in no or minimal project feedback which is imperative to testing and development in the DevOps process. It must be established at project commencement who the stakeholders are and verify available communication channels. Each stakeholder should have one or more representatives to step in if they are unavailable. This way, full stakeholder participation throughout the SDLC can be achieved.

Loss of project work time. For several reasons, some of which we have already discussed, budgeted project schedule time can be reduced to levels that risk successful project execution. In an HRE, there could be many cases where a project may require start to finish completion in hours or days. The approval processes for software and hardware acquisitions, software change requests, and others can place project completion at risk. Unfortunately, hardware and software acquisition is likely beyond the control of project developers. In these cases, project time extensions must be requested or requirements reduced to a set that is achievable in the remaining time allotment. A DevOps approach to minimize lost project time is being proactive in developer, industry, government, and consumer technology communities. Part of this approach is to acquire, investigate, and create new tools and techniques, for both common and specialized applications. In cases where project approval is delaying work start, one approach would be assigning a senior member of the development team such as an immediate manager that is physically located with the team to give immediate approval to all projects with completion times below some threshold. That threshold could be hours, days, or weeks. The project approval request should simultaneously be submitted into the usual process of approval with an additional marking of "expedited approval due to project schedule pressure."

Partial or no production environment access. For projects developed in an HRE, scenarios will likely arise where the production environment is not available to the development team to create a replica for testing. Testing in a non-accurate replicated production environment is full of risks. This breaks two critical DevOps concepts. First, the ability to complete small tasks of a project and release it as a version to the end user for testing and feedback. This is a very important DevOps piece since modifications can be made on small tasks minimizing expenditure of time and money. Second, without testing on an accurately replicated production environment, no real world feedback can be provided on performance. One approach would be to elicit as much detail as possible about the production environment from all accessible sources and replicate as accurately as possible within the development team's work space as a staging environment. This form of internal testing would provide all stakeholders, especially the developers, some confidence the final product will work as expected in the production environment. Ideally, testing occurs in an accurately replicated production environment with end user feedback. In absence of that scenario, testing in a staging environment which is an accurate as possible replication of the production environment could be the next best choice.

New requirements late in SDLC. Following the DevOps process, requirements for software development are acquired at the requirements stage of the SDLC with all stakeholders involved. In the dynamic nature of HREs, requirements can arise in the later stages of the SDLC process. Accepting a new requirement at later SDLC stages can have broad impacts on the entire project consuming significant time for design, development, and testing. Ideally it is best that no requirements arise in late SDLC stages. If a new late stage requirement must be considered, here is some guidance to help decide if the requirement should be accepted or not. First, identify who is giving the requirement. It should only be the original requestor, an internal developer, or as a result of unexpected production environment changes. The original requestor can create new requirements for essential functions to the original project. Adding non-essential features or enhancements should not be accepted and instead placed on an optional project backlog. An internal developer can provide new requirements based on unique technical domain knowledge of hardware or software. A developer may be aware of fine grained system details that need specific treatment to avoid unexpected outcomes. In this case, the requirement must be approved by the original requestor. If a requirement meets one of the above criteria, the development team, and all other stakeholders, should assess and assure the time needed to implement and test the

requirement is within current time budgets. If needed time goes beyond current time budgets, either a request for additional time is approved or the new requirement is placed on an optional project backlog or rejected. Another option is to swap the new requirement with another requirement that is deemed non-essential and relabeled optional. In order to deal with "late stage" requirements better, developers should follow the DevOps concept of progressing a project through the entire SDLC process one small task at a time. Ideally, a "late stage" requirement will only impact one or a few completed tasks. A small task contains a small amount of source code in a single unit such as a function or class. This simplifies the task of modifying and testing code.

No centralized software installation. Some HREs may lack a centralized process to deploy software across several machines. This can cause machines to run different versions of the same software which can lead to several problems. Not having a centralized software deployment process violates the DevOps principle of environment parity. This principle dictates that all machines used in the same group or for the same purpose are setup and configured identically. For developers, this is critical. As an example, building code using two versions of the same library can produce significantly different outcomes. In the case of production environments, developers should deliver software for testing contained within the staging environments that are used for their own internal testing. The staging environment should either be an exact copy or accurate as possible replica of the expected production environment. Fortunately, there are several system container and machine virtualization technologies available to easily establish environment parity. Developers should deploy one virtual machine for all workstations at the start of each new project. This assures all developers are working on identical systems. The virtual machine should be archived and marked as designated for this project. Internal testing in a staging environment can occur using a virtual machine replicating the production environment. Virtual machines containing new project versions can be placed in a container and sent to external stakeholders for testing. This assures external testing is done in the exact same environments as internal developer testing. Environment parity provides several advantages, the most critical being the elimination of system differences as a source of reported problems when testing or running a project's software. A development team can create a centralized automated service to provision and deploy systems and software. This is the DevOps principle of Infrastructure as Code (IaC). This principle supports environment parity by deploying preconfigured systems, networks, and environments as needed by developers, operators, and testers. At the start of a project all stakeholders should agree on the systems and environments needed to build and test a project. These are built prior to project commencement and can be deployed automatically when needed. The service should track all personnel's deployments and launch automatic updates when changes to the base system occur ensuring environment parity throughout the system's life cycle.

High attrition of contract personnel. In an HRE, not all personnel are full time employees. It may be the case where some personnel are third-party contractors. These contractors work for variable periods of time, from days to years. An environment of contractors working on a project for short periods of time does not establish a long-lasting culture of DevOps within an HRE. Once a person understands the DevOps process being carried out at an HRE, it becomes part of daily work life. When all, or most, members of a group think the same way about DevOps, they are more likely to work seamlessly together as one cohesive unit. Regularly bringing new contract personnel up to speed on the team's DevOps process can delay, sometimes significantly, a project's completion. Ideally, contractors should persist until project completion or for several years with the same group to sustain a long-lasting DevOps culture. If it is unavoidable to retain contractors beyond short periods, a DevOps training should be a pre-requisite for a new contractor, or any team member, to join a group. This training indoctrinates a person in the DevOps process of a specific group. The training should consist of a lecture and sample project where the trainee carries out the project using the learned DevOps. Once completed, the trainee should shadow an established group member with same job type to observe firsthand the DevOps implementation. The group members collectively test the trainee's DevOps understanding and approve the contractor's entry to the group. The time spent training is variable and dictated by the group's suitability assessment. The training should always be performed in the same manner for each new group member and carried out by the same instructor. This person could be a full-time employee or contractor and should only do training for specific HRE groups. In order to sustain consistent and persistent training, a group of instructors should know how to provide training and are themselves trained in a consistent manner to assure all instructors have the same preparation. Instructor training should be dictated and managed by HRE personnel.

Slow IT request fulfillments. All HREs will submit IT requests for a diverse set of needs. In cases where a request holds back project progress, the developers should not be held accountable for internal IT delays. Much like hardware and software acquisition, there may be little that can be done, from a DevOps perspective, to change the internal IT process of fulfilling requests. One approach to alleviate slow requests is to implement the DevOps principle of IaC as previously discussed. Another approach to minimize IT requests is to assess required access, authorizations, systems, configurations, accounts, and other similar needs as part of the requirements gathering stage and fulfilled before any development starts. This can minimize IT requests to only unexpected occurrences. A proactive approach is to routinely assure developers have all they need for daily work. This is easily achievable with an IaC service. In accordance with the DevOps principle of environment parity, establishing a core set of tools and techniques that are available via IaC facilitates the assurance of access to all group members. Under these scenarios, only unique cases where specialization is required would an IT request be submitted.

We have addressed only general obstacles that one should expect in an HRE. There can be several more for specific circumstances in an HRE. The solutions may not always be rooted in DevOps principles. Overall, the goal is to create a more cohesive and inclusive group of stakeholders for a project which is a critical component of any successful DevOps implementation. At this point in the assessment, an internal write up of recommendations for each obstacle is in place. This list should be reviewed by HRE personnel. A list of

recommendations, mutually agreed upon between the assessment team and HRE personnel, is submitted in the final report.

5 IMPLEMENTING RECOMMENDATIONS

Once recommendations are finalized, the next step is for HRE personnel to implement them. The role of the assessor in this stage can be to assist in the implementation or end the engagement. Ideally, the assessment team, which possess DevOps domain knowledge, should assist HRE personnel in implementing recommendations. The best approach is to implement one recommendation at a time. Assessors and HRE personnel should work closely together to identify the processes and systems that will be modified by the implementation and the affected team members. In some cases, modifying existing or acquiring new hardware and software may be required. Given our previous discussion of hardware, software acquisition and IT requests, completing an implementation may take longer than expected. Each implementation may require significant time and effort to place within imposed policies and HRE constraints while others are simple and straightforward. It should be expected that some recommendations may need modifications resulting from discoveries while attempting to implement. HRE personnel should be made aware of this possibility beforehand. As an implementation is completed, a test period with one or more projects should be observed to assure the expected outcome has been achieved. Documenting observations and interviewing HRE personnel for feedback is critical to assure the implementation worked and the personnel using it are satisfied. If this is not the case, assessors and HRE personnel should identify what is not working, the reason for it and attempt to fix. This may not always be achievable and thus some recommendations may only be partially implemented. Once all recommendations have been addressed, assessors should observe their use in a project. The assessors and HRE personnel should decide how many recommendations to fully implement before observing the new SDLC process with DevOps in one or more projects. Issues may arise when all recommendations are in use that are not observable with individual recommendations in action. It is critical to conduct satisfaction surveys and solicit feedback of HRE personnel and stakeholders to assure that overall the assessment and resulting recommendations have improved the SDLC of the HRE entity.

In spite of the challenges an HRE poses to implementing DevOps [17], there are practices that can be implemented in a modified form which adhere to imposed policies:

(1) Communication with all stakeholders can be conducted on a regular basis [12].
(2) Developers can use current technology to assure all project development environments are configured in exactly the same way establishing symmetric environment parity. These technologies can be applied to internal testing and production environment replication.
(3) All written artifacts can be centrally stored and made available to stakeholders.

These practices are generalized and can be applied to almost any HRE entity. Achieving these practices will greatly enhance the DevOps posture for any HRE entity.

6 ASSESSING DEVOPS POSTURE OF AN HRE

A critical component is the ability to measure, or at a minimum demonstrate, how far or near the SDLC DevOps implementation of a group in an HRE is in comparison to a fully implemented ideal DevOps SDLC process before and after an assessment. This measurement will likely be subjective as each group may have their own view on the ideal DevOps SDLC process. The HRE personnel should explain what their ideal DevOps SDLC looks like with guidance from the assessors. In order to acquire the pre-assessment posture, an assessor should survey HRE personnel about the following general desirable properties in their current SDLC:

(1) Length of project life. This is the amount of time from start to finish of a project. Is it within budgeted time or is additional time regularly required?
(2) Source code commits per day. This is a reflection of code development tasks. If a task is too large, it may not be completed and committed in one day. Multiple smaller tasks may be easier to complete and commit in a single day. Is the group satisfied with the commits per day? Does it support scheduled project completion?
(3) Stakeholder communication. How well and often does this occur? Is the group capable to speak with a stakeholder as often as needed or desired?
(4) New employee spin up time. How much time is needed to indoctrinate a new employee in the group's SDLC process?
(5) Consistent development environment. This relates to environment parity. Is the various environments being used by developers and other HRE personnel kept consistent and up to date with tools, updates, features, etc...?
(6) Tooling usage. This refers to development team tools such as chat services, versioning systems, testing and development infrastructures, automated IT request fulfillments, and other similar tools. Are these tools available to developers and do they function effectively? Are their multiple tool sets for the same purpose? Can the current tool set be reduced to exclude those not favored by developers? Consistent appropriate staff. Does the HRE provide long term, well-qualified personnel that are assigned to specific job functions such as: IT infrastructure, software deployment, programmers for specific languages, unit testers, system integration, etc...?
(7) Production deployment. How often is code pushed to production for end user feedback? Is it hourly, daily, weekly, longer? Is deployment performed by one person or a group? Is deployment a consistent, repeatable process or is it a unique effort each time? Is approval required? How long is the wait for approval? Does it negatively affect the deployment process?

For each desirable property listed above, a measure of 1 to 5 can be assigned. A measure of 1 is the worst case possible and a measure of 5 is the ideal case. For example, consider âĂIJCommits per dayâĂİ, a 1 could mean the group has very large tasks that cannot be completed and committed in one working day. A 5 would indicate multiple small incremental tasks are being completed and committed on a daily basis. A group is free to add their own desirable properties to this list. The assessor should assure additional properties pertain to DevOps and do not overlap with

listed properties. The ideal DevOps SDLC expressed by the group in pre-assessment will serve as the threshold to reach with recommendations. After observing all recommendations in action for one or more projects, conduct a post-assessment survey to re-assess how near to the ideal DevOps SDLC the group has reached. This will serve as a measure of how effective the assessment and recommendation implementation has been. For those properties in which the group feels they are still lacking, discuss the reasons why and recommend what can be done to improve.

7 ASSESSMENT FOLLOWUP

At three to six months after completing an assessment, the assessors should return to repeat the post-assessment survey to assure satisfaction has sustained or improved. HRE personnel should be asked if new obstacles have arisen resulting from the assessment. If possible, assessors should observe one or more projects to validate the SDLC is being implemented as recommended. Deviations should be pointed out and remedied if possible and deemed necessary. Negative survey results and new obstacles can be the focus of a new assessment specific for these issues. HRE personnel should be informed that DevOps implementation, especially in an HRE, may involve multiple assessments with each refining the SDLC process to a point where personnel is satisfied and no new issues have arisen.

8 GENERAL HRE DEVOPS STRATEGIES

The following are some general DevOps strategies that can be implemented in an HRE. These strategies address some of the large-scale issues that can arise in these environments with potential approaches to addressing them.

Implementing SDLC in air-gapped environments. One of the biggest obstacles to DevOps in an HRE is air-gapped spaces and systems. This disallows several DevOps principles such as stakeholder collaboration, frequent production environment deployment and end user feedback. One general approach to alleviating this obstacle is to sustain an open environment outside the HRE for development and testing. It is important to realize development itself is typically not a security or proprietary concern. The specific techniques and data that are sometimes in use typically are security or proprietary concerns. It is best to perform as much development as possible in an open environment outside of the HRE. This would allow a straight application of DevOps to the SDLC. Development in the HRE should be restricted to scenarios of security or proprietary concerns. In most HREs, all development is performed in air-gapped environments due to imposed policies or consolidation of work. When carrying out a project in an air-gapped space, consider using a staging environment which replicates the production environment within the air-gapped space. Assuming all the other SDLC components are in one space, adding the staging environment adds the needed infrastructure for the SDLC. This facilitates implementing traditional DevOps within an air-gapped space while assuring no imposed policy or HRE requirement breaches occur. This is assuming there are no imposed policies disallowing full SDLC development to occur within one air-gapped space. The staging environment should exist in its own network and hardware so deployments from developers occur as usual but internally. Personnel should be assigned the roles of end users. They should not have insight of the product outside of requirements. They should interact with the staging environment and provide feedback leading to code improvement.

Fast track DevOps for short project times. It may be the case where a project needs to be completed in hours or days. In these cases the development team may be unable to perform the entire DevOps SDLC process as desired. To deal with short completion times, consider reducing the DevOps of the SDLC to the following critical components:

- Accurate production replication and staging environment testing
- End user feedback
- Regular stakeholder updates

Ideally, the full DevOps process should be carried out regardless of budgeted time. In practice, this is likely very difficult in cases of severe time constraints. Keeping the stakeholder informed and assuring the product will work in the expected manner in the production environment are the key goals. Note that no late stage requirements should be allowed in time constrained projects.

Common simulation environments. As we have previously discussed, it can be a challenge, in some cases, to access and accurately replicate actual production environments. One approach that the HRE DevOps community can take is to ask manufactures of various production environment systems to create a simulated version of their systems. When commencing a project, all stakeholders can come to agreement that a manufacturer's simulation system is suitably equivalent to the actual production environment of a project, meaning the simulation system performs all needed tasks to fulfill the project. Under this agreement, developers can conduct all internal testing using the simulation system as the staging environment and report results to stakeholders. End users can test delivered versions on the same simulation system and provide feedback even within one air-gapped space if needed. By agreeing on a simulation system provided by the manufacturer, environment parity of the staging environment with the production environment and an effective end user feedback loop are assured, thus fulfilling these two core DevOps principles.

Secure DevOps. The term "Secure DevOps" indicates security is given high priority throughout the SDLC and general software development security practices are implemented at each stage of the cycle [17, 21, 22]. This includes security requirements at the requirements gathering stage, secure coding practices at the development stage, testing for vulnerabilities and unexpected behaviors, and creation of misuse cases addressing security concerns such as breaches and attacks. It is important to have a group of security experts as stakeholders. The ultimate goal of Secure DevOps is assuring security is a standard part of the SDLC. When addressing this in an HRE, it is important to identify secure DevOps practices that are already in place due to imposed policies or HRE constraints. It is possible that practices already present may be applied to all or specific projects. Attempt to understand why some practices are only for specific projects and, if appropriate, suggest these practices should be applied to all projects. It may be the case, in some HREs, that secure DevOps is routinely practiced and referred to by

a different name. In these cases, the assessor should review these practices and suggest modifications if beneficial.

9 CONCLUSIONS

In this paper, we have discussed the process of assessing and implementing DevOps practices in highly regulated environments (HRE). The restrictions imposed in an HRE disallow the direct implementation of traditional DevOps to an SDLC process. We have described how to perform a DevOps assessment in an HRE and how to identify bottlenecks and pain points. A process is given to document these problems as obstacles. Several general obstacles that are likely present in an HRE is described along with approaches to overcome or alleviate each one. The solution to obstacles can be DevOps related or not. Some recommendations are likely not part of DevOps but will improve the SDLC in some way. Innovation enters the process by assessors thinking outside the box in finding unique ways to implement DevOps in an HRE to overcome obstacles. The ultimate goal is to place an HRE's DevOps posture as close to an ideal scenario as possible. A method to establish this posture in pre and post assessment is given. In achieving an improved DevOps posture, the underlying SDLC will benefit from both DevOps and non-DevOps enhancements. The resulting DevOps SDLC will be a process that started as a traditional DevOps process which was molded and customized to an HRE group's specific needs and environment. This new process can be labeled an HRE-DevOps SDLC. At the end of an assessment, the key component of the final report will be the list of recommendations that consist of both DevOps and non-DevOps enhancements resulting in a customized version of HRE-DevOps. Once implemented, this customized version will further bring an HRE development team closer to their ideal DevOps SDLC scenario. Our future work includes implementing secure DevOps in an HRE, and defining metrics to determine success of an HRE-DevOps implementation.

ACKNOWLEDGMENTS

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. [DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. Carnegie MellonÂő and CERTÂő are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University. DM18-0273

REFERENCES

[1] [n. d.]. Air gap (networking). https://en.wikipedia.org/wiki/Air_gap_(networking)
[2] [n. d.]. Barak Obama's Top Secret Tent. ([n. d.]). https://web.archive.org/web/20151206091123/http://www.bbc.com/news/world-us-canada-12810675
[3] [n. d.]. Nuclear Power Plant Security and Access Control. ([n. d.]). https://www.nei.org/Master-Document-Folder/Backgrounders/Fact-Sheets/Nuclear-Power-Plant-Security-and-Access-Control
[4] [n. d.]. United States Federal Communications Commission. https://www.fcc.gov/
[5] [n. d.]. United States Food and Drug Administration. https://www.fda.gov/
[6] [n. d.]. United States Securities and Exchange Commission. https://www.sec.gov/
[7] Len Bass, Ingo Weber, and Liming Zhu. 2015. *DevOps: A Software Architect's Perspective* (1st ed.). Addison-Wesley Professional.
[8] Gary Blau, Bharat Mehta, Shantanu Bose, Joe Pekny, Gavin Sinclair, Kay Keunker, and Paul Bunch. 2000. Risk management in the development of new products in highly regulated industries. *Computers & Chemical Engineering* 24, 2 (2000), 659 – 664. https://doi.org/10.1016/S0098-1354(00)00388-4
[9] Constantine Aaron Cois, Joseph Yankel, and Anne Connell. 2014. Modern DevOps: Optimizing software development through effective system interactions.. In *IPCC*. IEEE, 1–7. http://dblp.uni-trier.de/db/conf/ipcc/ipcc2014.html#CoisYC14
[10] Franklin R Edwards. 1977. Managerial Objectives in Regulated Industries: Expense-Preference Behavior in Banking. *Journal of Political Economy* 85, 1 (1977), 147–62. https://EconPapers.repec.org/RePEc:ucp:jpolec:v:85:y:1977:i:1:p:147-62
[11] Nicole Forsgren, Monica Tremblay, Debra Vandermeer, and Jez Humble. 2017. DORA Platform: DevOps Assessment and Benchmarking. (05 2017), 436–440.
[12] Alfonso Fuggetta and Elisabetta Di Nitto. 2014. Software Process. In *Proceedings of the on Future of Software Engineering (FOSE 2014)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/2593882.2593883
[13] Joan Harvey, George Erdos, Helen Bolam, Michael A. A. Cox, John N. P. Kennedy, and David T. Gregory. 2002. An analysis of safety culture attitudes in a highly regulated environment. *Work & Stress* 16, 1 (2002), 18–36. https://doi.org/10.1080/02678370110113226
[14] Lawrence G. Hrebiniak and William F. Joyce. 1985. Organizational Adaptation: Strategic Choice and Environmental Determinism. *Administrative Science Quarterly* 30, 3 (1985), 336–349. http://www.jstor.org/stable/2392666
[15] Michael Httermann. 2012. *DevOps for Developers* (1st ed.). Apress, Berkely, CA, USA.
[16] S. Allcorn M. A. Diamond, H. F. Stien. 2002. Organizational silos: Horizontal organizational fragmentation. *Journal for the Psychoanalysis of Culture & Society* (2002).
[17] Don O'Neill. 2017. Secure DevOps Foundations for Large-Scale Software Systems. *Crosstalk* (2017), 24–27.
[18] Rod Rasmussen, Tim Hughes, J. R. Jenks, and John Skach. 2009. Adopting Agile in an FDA Regulated Environment. In *Proceedings of the 2009 Agile Conference (AGILE '09)*. IEEE Computer Society, Washington, DC, USA, 151–155. https://doi.org/10.1109/AGILE.2009.50
[19] Guoping Rong, He Zhang, and Dong Shao. 2016. CMMI Guided Process Improvement for DevOps Projects: An Exploratory Case Study. In *Proceedings of the International Conference on Software and Systems Process (ICSSP '16)*. ACM, New York, NY, USA, 76–85. https://doi.org/10.1145/2904354.2904372
[20] Paul M. Schwartz and Daniel J. Solove. 2011. The PII Problem: Privacy and a New Concept of Personally Identifiable Information. *NYUL Rev.* 86 (2011), 1814–1894.
[21] Andrew Storms. 2015. How Security can be the Next Force Multiplier in DevOps *(RSA Conference 2015)*.
[22] Hasan Yasar and Kiriakos Kontostathis. 2016. Where to Integrate Security Practices on DevOps Platform. *IJSSE* 7 (2016), 39–50.