

Agile Manifesto for a Holistic AppSec Environment



```
bool success = false;

size_t size = a_segments.size();
if (size)
{
    tSet = static_cast<CSegment*>(a_segments[size-1])->a_T1;
    success = true;
}

return success;
```

```
////////////////////////////////////
CTrack::query_segment_index(int32_t ind, float time)

bool success = false;

size_t size = a_segments.size();
for (size_t i = 0; i < size; i++)
{
    CSegment* segment = static_cast<CSegment*>(a_segments[i]);
```

```
float t0 = segment->a_T0;
float t1 = segment->a_T1;

if (((t0 - Epsilon < time) && (t1 + Epsilon > time)) //interval
{
    index = (int32_t)i;
    success = true;
}
}

}
```

```
45 //////////////////////////////////////////////////
46 CContext::CContext(
47     : a_windowed(tr
48     , a_fileCreate
49     , a_fileCreate
50     , a_GenNodeEn
51     , a_fileTrack
52 {
53 }
54
55 //////////////////////////////////////////////////
56 std::string CCont
57 {
58     auto it = a_Re
59     ASSERT(it != a
60
61     return TAKE(CA
62 }
63
64 //////////////////////////////////////////////////
65 void CContext::Loc
66 {
```

```

67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Table of Contents

Agile Manifesto at 20.....	3
Agile’s culture of collaboration builds trust in your software.....	4
The Four Principles of Holistic Agile Application Security	5
Developers and testers over security specialists.....	5
Securing as you work over securing after you’re done	6
Implementing features securely over adding security features.....	7
Mitigating risks over fixing bugs	7
Applying the Four Principles.....	8



Agile Manifesto at 20

It's been more than 20 years since the [Agile Manifesto](#) turned software development on its head. Widespread adoption of the methodology across both IT and non-IT teams has changed the way software is developed and businesses are run. Although a few crucial sectors such as healthcare and aerospace still use waterfall development (you'd hardly want iterative development builds pushed to your pacemaker), most software is now developed in short sprints, and builds are routinely pushed out as they are developed.

When the pandemic prompted businesses to pivot to new ways of working, agile methodologies, tools, and processes provided a framework that enabled success in remote and hybrid workspaces. Although critics might gripe that some of these adoptions are not "true agile," the genius of the agile methodology—and the reason it's been so successful over these past two decades—is that it's built on a culture of collaboration and feedback that breaks down silos and allows groups to work more efficiently, flexibly, and happily.



Agile's culture of collaboration builds trust in your software

The four core values of the original Agile Manifesto

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Each of these values prioritizes collaboration and feedback to build organizations that can respond to situations as they unfold instead of sticking to a perhaps-outmoded plan. This results in a nimble organization that can respond to uncertain times like those we're currently navigating.

Over 20 years ago, when the Agile Manifesto was released, the internet had not yet moved to the cloud, nor had the Internet of Things (IoT) pushed into our homes. Today, in this era when every business is a software business—either your business is using software or building software, and often it's both—the flexibility of the agile methodology is crucial. The latest [State of Agile report](#) confirms that the path forward isn't about IT aligning to business, it's about IT being fully integrated into business and leading the way to business agility.

As business and software have become intertwined, it's more important than ever that trust is built into the software we depend on. The evolution from agile to DevOps and then to DevSecOps is a natural one. The DevOps and DevSecOps methodologies provide automation and closer collaboration between development and operations. This enables agile software development, which in turn enables the faster and more flexible deployment of more-secure software that can quickly respond to a changing environment. Agile AppSec principles help you build a [holistic AppSec environment](#) and drive software development that runs at velocity and delivers **secure** high-quality software.

However, even after more than two decades of evangelism about the [secure software development life cycle \(SSDLC\)](#) too many organizations still think of security as synonymous with testing, and silo security activities away from the software development process. Security is too often seen as something separate from and external to software development, and security activity outcomes are presented in documents and outputs that don't synch with software development activities. All of this slows your ability to integrate secure development into your operations.


Let's take a look at how applying the agile methodology to holistic AppSec practices can help you build secure software.

The Four Principles of Holistic Agile Application Security

The four principles of the Agile Manifesto can be used as a starting point for holistic application security practices. The four principles of holistic agile AppSec prioritize

- **Developers and testers** over security specialists
- **Securing as you work** over securing after you're done
- **Implementing features securely** over adding security features
- **Mitigating risks** over fixing bugs

Like the original core values of the Agile Manifesto, these holistic AppSec core values prioritize collaboration and feedback so organizations can respond to situations as they unfold instead of sticking to a predefined plan. Each of these principles shifts the responsibility for security left in the development cycle. However, the only way to implement this shift is to integrate it into the core agile values of collaboration, continuous feedback, flexibility, and teams working independently and taking responsibility for managing their own work. Adding a set of values like these to your existing agile process will help you build a nimble organization in which security development doesn't just shift left—it shifts everywhere.



The goal of a holistic agile AppSec is to make it natural and efficient to build security into an agile process.

Developers and testers over security specialists

The key to this principle is instilling the belief throughout your development teams that **security is not someone else's job**. Shifting security left means that security is increasingly the responsibility of developers and testers. Shifting security everywhere allows organizations to treat application security vulnerabilities as defects and fix them as they happen. This not only makes more-secure software, it saves time and money, as it's much harder to chase down security defects once they've populated into your code.

Experienced security specialists are valuable, but few agile teams have the luxury of their own dedicated security specialists. The simple fact is that at most organizations, there are far more application developers than there are information security professionals. Sixty-six percent of organizations surveyed in a recent study by 451 Pathfinder said their organizations don't have enough information security professionals.

This means that most of the time, software development teams are responsible for their own security and they can't wait for an external security review before the code moves forward on the assembly line. Security must be integrated into code development and testing. Teams must own security the same way they own user experience, reliability, performance, and all other nonfunctional requirements.

In a holistic AppSec development environment, security specialists function like specialists of any other discipline (performance, reliability, user experience, etc.). They help teams understand how to achieve their security goals. Security specialists can also take on tasks that require specialized knowledge or experience, like integrating security tools into the development tool chain and the CI/CD environment. Security specialists can also be involved as you define user stories and prioritize your backlog of test findings, but teams should own security as an integral part of their own development workflows.



Integrate
security activities
into what you are
already doing.

Securing as you work over securing after you're done

When it comes to creating, releasing, and maintaining functional software, most organizations have been using the agile methodology for years. However, when it comes to securing that software, too many development teams still think of security as interference—something that throws up hurdles and forces them to do rework, keeping them from getting cool new features to market.

How do you make security activities a part of your development process and build a truly secure software development life cycle? How do you shift not just left but everywhere, and build security into your workflows without impacting velocity? You don't want security activities that make developers stop what they're doing, go to another tool to remediate, and then come back to what they were doing. That level of distraction derails development velocity and often forces developers to revisit work they may have done weeks earlier.

A defect solved at the point of code creation is easiest to correct, whereas one that has escaped to the testing phase typically involves more people to remediate: developers to re-engage the affected code and testers to retest. [The Systems Sciences Institute at IBM](#) reported that it was six times more expensive to fix a bug found during implementation than to fix one identified during design. Furthermore, according to IBM, the cost to fix bugs found during the testing phase could be 15 times higher than the cost of fixing those found during design.

One important way to implement a holistic AppSec security environment is to integrate security feedback and information into your developer tools. This allows security specialists to do their work however they like but using the same tools your developers are already working with. If your development team uses a whiteboard, sticky notes, and an online tool, the security priorities have to be visible alongside other tasks you are tracking.

Even better, implementing automated security tools will allow developers to do a quick scan of code and get timely and relevant feedback as they work. Using tools that run security scans and return information about security vulnerabilities, and also give guidance about how to remediate them, will go a long way toward shifting security everywhere through your development cycle and ensure you're releasing better, more-secure software.

Implementing features securely over adding security features

It's more efficient to focus on using the iterative nature of agile holistic AppSec practices to evolve the security of your software than it is to succumb to the temptation to add a lot of security features all at once for a quick fix. Although your primary focus should always be achieving your software's business mission, you cannot achieve that without building trust into your software. Now that much of the responsibility for secure development is shifting onto development teams—who are not security specialists—you'll need to think about how to integrate security into your holistic AppSec development environment.

On the macro level, this means using intrinsic methods of adding security. Security needs to be multidimensional. The systems architecture, systems engineering, and security engineering teams must work together from the beginning to define and build more-secure systems. These teams need to understand the system components, how they are put together, how much protection each component has, the information flow between components, how the components interact, where the single points of failure are, the interactions with the supply chain, the automatic updates or patches, and what would happen if there was malicious code in those updates.

On the micro level, this means using secure practices to protect your software and your users. You'll want to add security via secure-by-default frameworks instead of relying on explicit calls to security libraries. Abide by the first rule of cryptographic security: Do *not* to roll your own crypto. Don't build your own custom implementations of security functions. Leave the implementation of security features like authentication controls, password storage schemes, and cryptographic algorithms to the professionals whose day-to-day work focuses on these areas.



Use tried-and-true secure implementations instead of building it yourself.

Mitigating risks over fixing bugs

It is tempting to rely on quantitative milestones instead of doing the harder work of thinking qualitatively about risk. It's easy to go on a hunt for as many security vulnerabilities as you can find, count up your mitigation numbers, and declare the software "secure." But this avoids the harder work of considering the risks specific to your business, your users, your data, and your software.

Risk management is about figuring out the right way to deal with a risk. Maybe you write some code; maybe you monitor and react to bad behavior; maybe you use legal and contractual controls instead of technical controls. Agile holistic AppSec development favors taking a high-level view of what could go wrong instead of distilling "security" down to a giant list of individual bugs that need to be squashed.

Every organization has a different risk profile and tolerance, and a holistic threat modeling approach consists of two essential steps.

1. Review your system's major software components, security controls, assets, and trust boundaries.
2. Model threats against your existing countermeasures to evaluate potential outcomes.

Although threat modeling is more difficult than simply going on a bug hunt, it's the most effective way to detect problems early in the SDLC. Threat models can help spot design flaws, evaluate new forms of attacks, maximize your testing budget, and identify holes in your requirements process. You'll save money by remediating problems **before** releasing software and performing costly code rewrites.

Applying the Four Principles

The purpose of these agile AppSec principles is to guide and inspire you, not to enforce a new set of rigid rules. What was groundbreaking about the original Agile Manifesto two decades ago was its simplicity, and the way it turned a top-down rigid development process on its head. Instead of central planning and hierarchical delivery of coding tasks, agile empowered small groups of developers to self-organize, collaborate, and innovate. Agile's emphasis on flexibility and responding to changing situations as they develop has been a key to its success.

As the software industry evolves and infiltrates all aspects of business life, we now find ourselves challenged not only to deliver nimble software at velocity, but to deliver nimble **secure** software at velocity. When software runs everything from our factory lines to our vacuum cleaners to our automobile navigation systems, it's essential that businesses build trust into their software. By building on the collaboration and feedback culture that's at the heart of agile development, and adding these DevSecOps-specific principles, you can build a robust holistic AppSec development environment.

Not sure where to start?

Black Duck® has all the tools and services you need to achieve not just software compliance and software quality but true software security. We can help you avoid data breaches and compliance gotchas by implementing security at every step of your software development life cycle, from developer to deployment. At Black Duck, we help organizations build secure, high-quality software faster.

Let Black Duck help you build secure, high-quality software faster.

[Ready to learn more?](#)



About Black Duck

Black Duck[®] offers the most comprehensive, powerful, and trusted portfolio of application security solutions in the industry. We have an unmatched track record of helping organizations around the world secure their software quickly, integrate security efficiently in their development environments, and safely innovate with new technologies. As the recognized leaders, experts, and innovators in software security, Black Duck has everything you need to build trust in your software. Learn more at www.blackduck.com.