Sonatype

Successful Agile Development Efforts Require Automated "Golden" Policies

Yesterday's Golden Repository Approach is No Longer Sufficient

Augment Your Golden Repository with Golden Policies

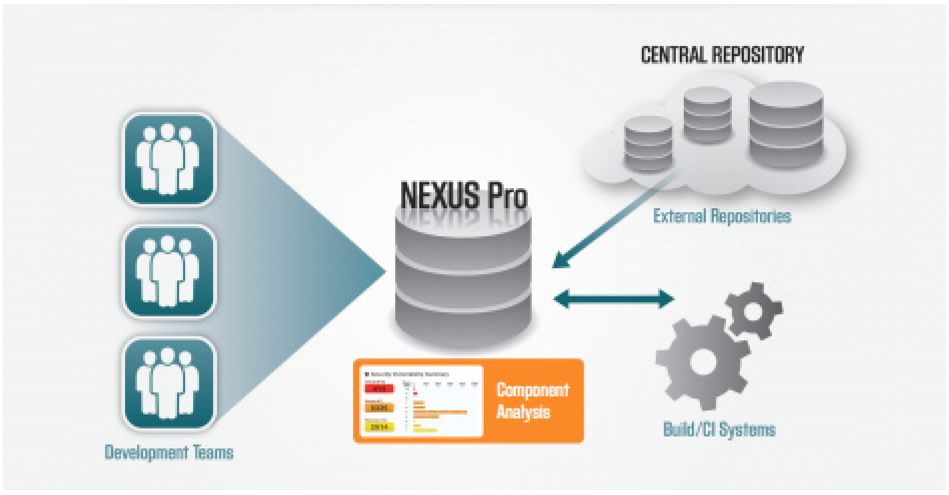
Organizations turn to various technologies and processes to ensure that component-based development efforts produce trusted applications. One approach is to limit access to an approved set of components using a "golden repository." Before any component is placed into the repository, it goes through an approval process that includes a security, legal and architecture review. While this approach makes sense on the surface, it has negative unintended consequences. The golden repository approach no longer works in agile, component-based development efforts because developers are under pressure to deliver fast and component volume, complexity, diversity and release cadence overwhelms the approval and maintenance processes required to keep the golden repository current.

While the repository manager remains a foundation for component management, a new approach is needed to ensure that developers deliver trusted applications. An approach that leverages automated policies and that provides guidance and enforcement throughout the entire software lifecycle. In short, you need a golden policy approach, not a golden repository.

Nexus Provides a Solid Foundation for Component Management

Many organizations have turned to the Nexus repository manager as a key part of their application tool chain. Because today's applications are constructed of components – with up to 80-90% of each application being comprised of components - organizations simply can't keep pace with the volume, variety, complexity and release cadence of components.

A reliable, scalable repository manager provides a foundation for component management.



Nexus Pro is the industry leading Repository Manager that provides enterprise features allowing organizations of any size to effectively acquire, store and share components. It is the Cadillac of repository managers. Nexus Pro has been designed to support large enterprise-level usage. It is simply unmatched in terms of capabilities, reliability/scalability and market share. Nexus drives success for tens of thousands of organizations.

We worked hand-in-hand with these customers to understand their requirements for component governance. They wanted to limit their developers to approved components that would ensure trusted applications - applications free from security, licensing and quality issues. And we addressed these "golden repo" requirements directly in the product. It worked, but only to a point.

The Golden Repo is Not Sufficient for Today's Development Approach

So, what's missing? Nothing is missing if you need to simply retrieve, store and share components, but organizations are looking, or should be looking, for more. They are looking to effectively manage and govern their components. They want to share the best components, components that speed development without placing the organization at security, licensing and architecture risk.

That's where a limited, repository-only approach to component management can be problematic. Put another way: using a golden repository to manage components by restricting usage to only those components approved by your security, licensing and architecture teams actually has a ton of unintended consequences.

So what went wrong? Well, the problem is not a functional limitation with Nexus – Nexus procurement support does a fine job of controlling what goes into the repository. The issue is a process problem that leads to unexpected consequences:

- **Developers bypass the repository** – Developers are measured by their ability to quickly deliver application functionality that fulfills a business need. While developers are concerned about the quality, reliability, scalability and performance of the application, they may not be completely invested in security and licensing implications. They aren't being negligent; they simply lack the information and time to address these concerns – especially if they are under heavy pressure to make tight development deadlines. What do they do if they turn to the golden repository for a component they need and they find it isn't there? Well, they bypass the repository to meet deadlines – or if they settle for what is in the repository, they might be using sub-optimal components.

- **Golden repos don't support experimentation** – What if developers want to experiment with a new component, one that has not been vetted by the organization? Well, they may start using a component with the full intention of seeking approval, but they may forget, and when the application goes into production, so does the non-vetted component.

- **Approvals can't keep pace with components** – Given that applications are comprised of up to 80-90% components, organizations that rely on an approval process can't keep up with the volume, variety, complexity and release cadence of components. Even if the approval process is automated with some form of workflow, it won't keep pace with agile-development deadlines.

- **Components become stale and outdated** – Given the volume and release cadence of components, organizations can't keep up with the velocity of new releases. This means that previously blessed components are stale – in this case, components in your golden repository actually put you at risk.

- **New vulnerabilities are not identified** – A solid repository manager approach needs to be augmented by the ability to identify newly discovered vulnerabilities, proactively notify appropriate constituents, provide the ability to prioritize and triage work, and finally, provide remediation and deployment assistance. A repository manager alone is not designed to provide this functionality, it needs to be complemented by a complete component lifecycle management capability that helps identify new vulnerabilities as they arise in previously approved component versions.

- **Production applications are not supported** – While a repository manager plays an instrumental role in the development and deployment of applications in production, it doesn't do anything to support an operational application. The repository alone won't provide the ability to identify new component vulnerabilities and determine which applications are at risk.

- **Different department or risk profiles are difficult to manage** – While it may be possible to organize distinct repositories so they contain different sets of components based on different department or application risk profiles, this approach is unwieldy and difficult to manage.

So how do you avoid these consequences? You could use multiple repositories to overcome these issues. Some that are open, some that are locked down. Some that support sensitive applications, some that support non-sensitive applications. Unfortunately this still results in problems:

- There is no easy way to move between the repositories and identify components that are approved or not approved without going through a series of "let's change the repo URL and see what breaks." This often leads to organizations delaying the reconciliation process until late in the development cycle.
- It's not easy for developers to see what components are likely to be approved, what components will be approved automatically or whether the component will be automatically denied.
- Managing multiple repositories for this purpose creates an administrative burden that could be managed more effectively using policies that codify your security, licensing and architecture standards.

Your Component Governance Approach Must Be Developer Friendly

So what is the answer? The repository manager is a key part of the answer but it must be augmented with component management and governance that supports the entire software lifecycle. And just like your repository management strategy must be designed for developers - your governance approach will only be effective if it works for developers.

Your component management approach must be developer friendly or it will fail. It should be fast, precise, contextual, actionable and continuous.

That means that your approach must include these qualities:

- **Fast** - No one has the time to wait hours for a scan to complete or days to wade through superfluous interpretation. Developers can't wait for an approval-laden process, even if it is automated with workflow. They need automated policies that provide timely guidance including up-front flexibility with additional control over production deployment.
- **Precise** – Given the number of components that organizations have to address, information about the exact version of the component being used is an absolute requirement. It's not good enough to state: "Struts is approved", when Struts 2.3.15.2 is good and Struts 2.3.15.0 will let the hackers in and you will get Pwned.
- **Contextual** – Developers need information that relates to their application. SQL Injection vulnerabilities are irrelevant if the application logic isn't dependent on a database and CopyLeft licenses may not be a problem for internal applications or services.
- **Actionable** – You can't stop once the problem has been identified – you have to help the developer fix the problem. And the easier you make it, the more likely the developer will fix it. Even if they don't completely understand the risk, if it's easy to apply the fix, the developer will do it.
- **Continuous** - Applications that have "left the building" don't age like wine (it's more like milk!). You need an approach that will monitor applications for newly discovered threats. And your monitoring approach can't be invasive – you should monitor components for new vulnerabilities and map these discoveries to your application inventory.

A Golden Policy Approach is the Answer

So how does this change your repository approach? In short, you need to move from a repository centric approach to an automated policy approach. Policies provide the flexibility and control you need to speed development efforts while ensuring trusted production applications. An automated policy approach designed for speed, precision, context, continuity & action allows you to get the most out of your component-based development efforts while supporting your security, licensing and architecture risk standards.

A golden policy approach augments your repository manager and provides component governance that speeds development efforts and ensures the delivery and ongoing support of trusted applications.

Fast

- The component inventory and analysis must be delivered instantaneously – developers need up-to-date component intelligence throughout the software lifecycle.
- Automated policies need to replace approval-laden approaches – automated guidance and enforcement keeps pace with components and agile delivery cycles.
- Providing information early in the development cycle shifts remediation left – optimal component selection prevents problems while proactive monitoring speeds new vulnerability discovery.

Precise

- The component inventory must be version specific and capable of identifying altered components – an accurate, up-to-date inventory provides a sound governance foundation.
- Component analysis and insight on security, licensing and architectural data must take into account all sub components (dependencies and transitive dependencies.)
- Accuracy must span component security, licensing and usage intelligence – broad, accurate meta-data is key to support security, legal/compliance, and architecture policies.
- Guidance that drives component selection, triage and remediation efforts must be trusted – developers must trust the guidance or it will be disregarded.

Contextual

- Vulnerability identification, guidance and enforcement actions must align with the application context – information must be version specific to be relevant.
- Component intelligence and guidance must be delivered in context – governance can't be limited to the repo, it's needed throughout the development tool chain, for example the IDE and CI Server.
- Governance must be organization and application aware – automated policies should accommodate different application and organization requirements.

Actionable

- While inventory and vulnerability discovery start the process, governance must be actionable – if guidance doesn't result in remediation, production apps will remain vulnerable.
- Agile development will be stalled without effective release management capabilities – support for pushing new applications, functionality and fixes to production is key.
- It's not just developers, guidance should support action for multiple constituents – DevOps needs release management support, managers need risk assessment support, etc.

Continuous

- Applications and components aren't static, on-going production monitoring is key to protect against new threats – proactive identification and triage that leads to action is key.
- Risk management and compliance needs continuous support – organizations need constant visibility into their overall application portfolio.
- Perimeter-based approaches will fail – continuous support is needed throughout the entire software lifecycle.

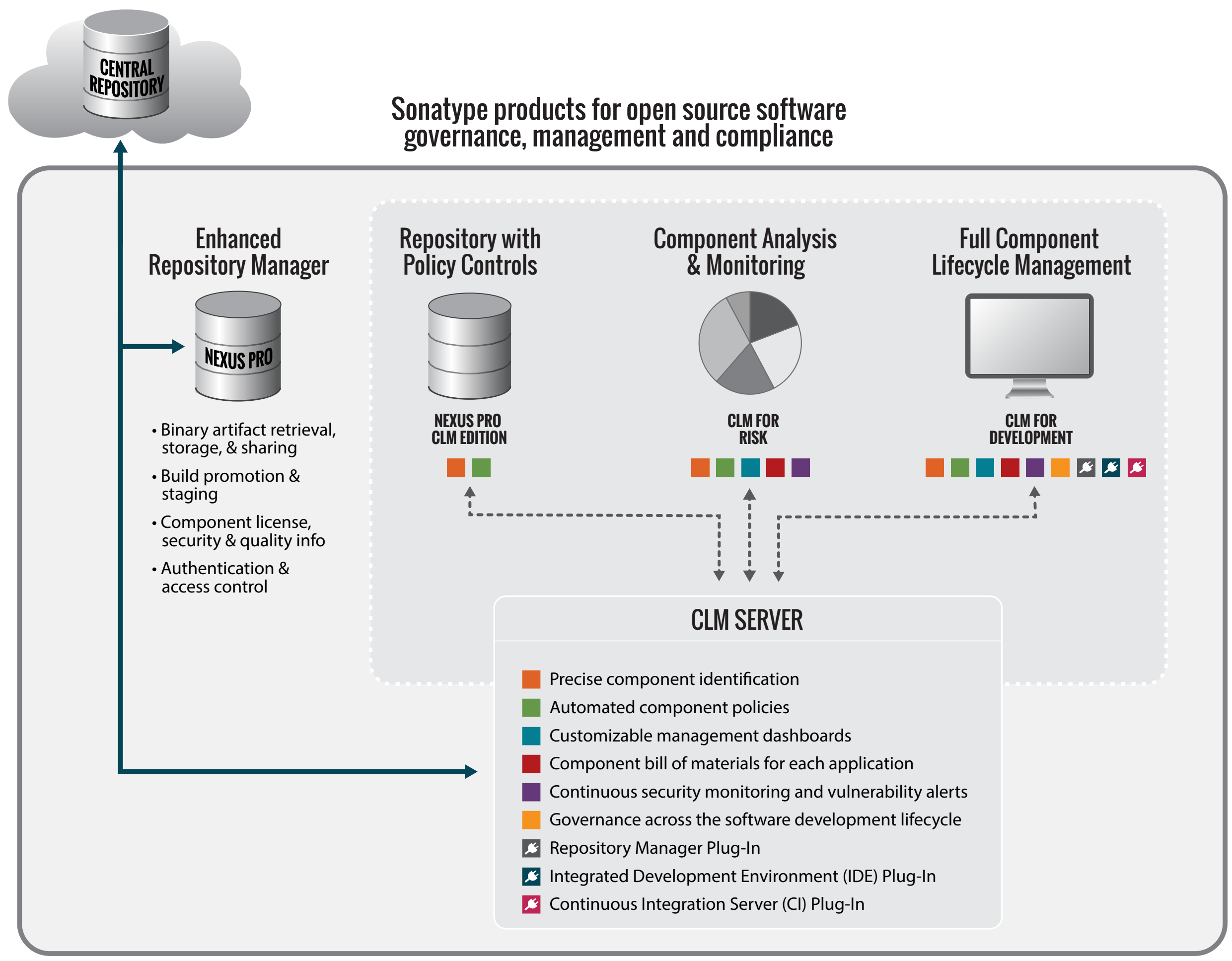
In short, you need a **Golden Policy Approach, not a Golden Repository Approach.**

Sonatype CLM Addresses the Golden Policy Across the Application Tool Chain

Sonatype provides the only solution that supports the golden policy approach. So how do you get started? Sonatype provides a flexible path depending on your organization's requirements.

Only Sonatype provides a complete set of solutions that are designed for how applications are constructed today. Only Sonatype provides an automated approach that leverages golden policies that guide development and production efforts throughout the entire software lifecycle.

- Extend Nexus with release management policies that automate compliance – **Nexus Pro CLM Edition**
- Support the entire development lifecycle with integrated component intelligence and guidance in the IDE, build/CI environments – **CLM for Development**
- Reduce production risk by continuously monitoring your component inventory for newly discovered vulnerabilities and drive response by proactively notifying appropriate constituents – **CLM for Risk**


CENTRAL REPOSITORY
Sonatype products for open source software governance, management and compliance
Enhanced Repository Manager
NEXUS PRO
• Binary artifact retrieval, storage, & sharing
• Build promotion & staging
• Component license, security & quality info
• Authentication & access control
Repository with Policy Controls
NEXUS PRO CLM EDITION
Component Analysis & Monitoring
CLM FOR RISK
Full Component Lifecycle Management
CLM FOR DEVELOPMENT
CLM SERVER
Precise component identification
Automated component policies
Customizable management dashboards
Component bill of materials for each application
Continuous security monitoring and vulnerability alerts
Governance across the software development lifecycle
Repository Manager Plug-In
Integrated Development Environment (IDE) Plug-In
Continuous Integration Server (CI) Plug-In