Clean Core: Extensibility Dimension

Today, we'll build on our previous discussion about processes in clean core and dive into how extensions can be implemented properly. This topic is particularly important because extensibility tends to be where things get messy in real-world implementations.

Let's discover how to keep your SAP system clean while still meeting your unique business requirements through proper extension strategies.

por Mayko Silva

Understanding Clean Extensions

1

Definition

Clean extensibility means implementing customizations or modifications to your SAP system in a way that keeps the core system standard while minimizing complexity.

2

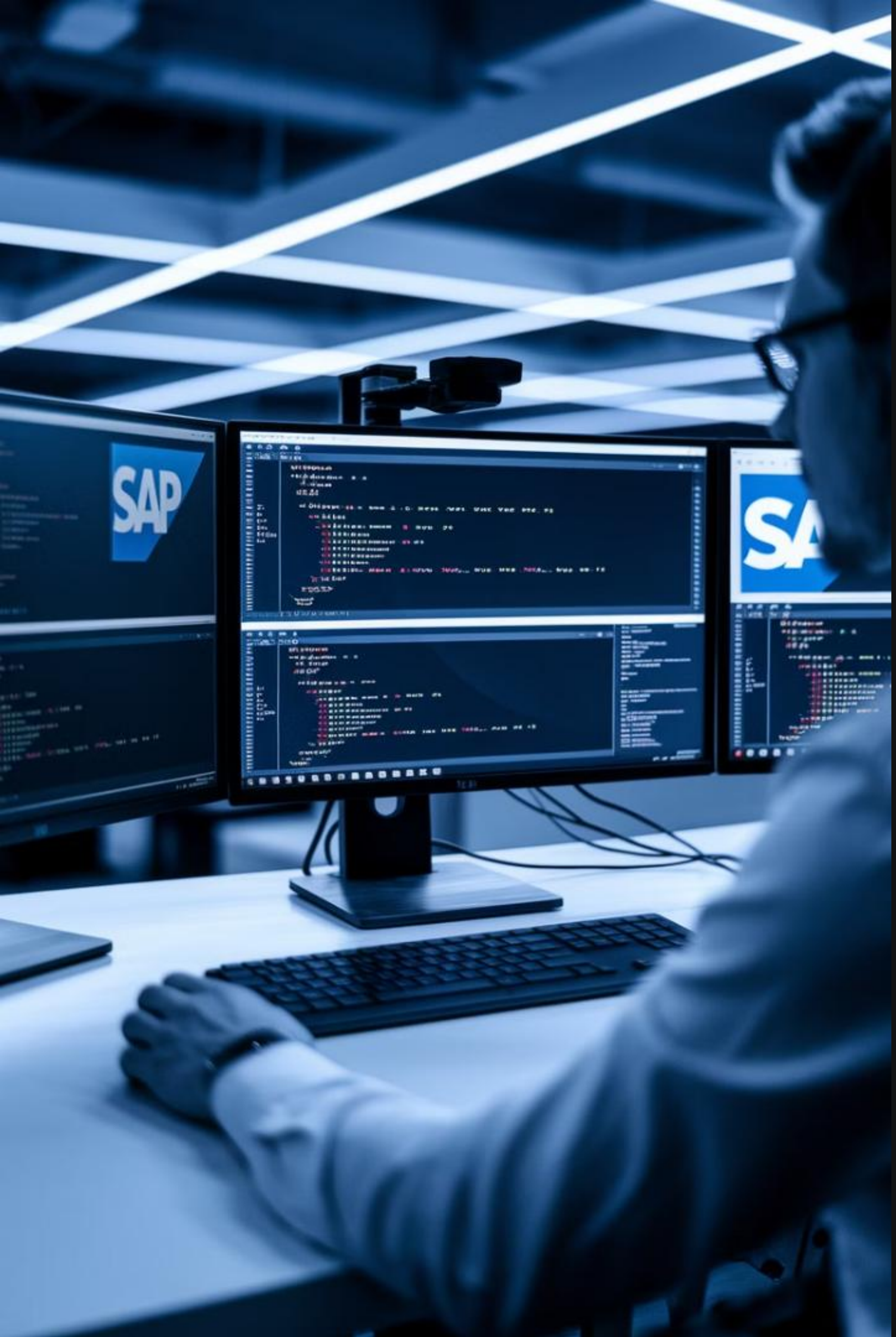
Key Principles

Maintaining a standard core system is essential. You must minimize unnecessary complexity while prioritizing future compatibility to avoid painting yourself into a corner with customizations.

3

Critical Factors

Upgradability is the whole point of a clean core approach. Security considerations are equally important in today's environment, ensuring your extensions don't create vulnerabilities.



Three Key Outcomes for Clean Extensions

Stability

Extensions must work reliably and survive system upgrades without breaking. This is a common challenge many organizations face when custom code fails after an upgrade.

Active Usage

Every extension should serve a clear business purpose and be actively used. Many organizations maintain customizations that nobody uses anymore, creating unnecessary technical debt.

Documentation

Extensions need to be well documented and developed based on best practices. Though often overlooked, proper documentation ensures maintainability and knowledge transfer.

Main Aspects of Clean Extensions

Requirements Understanding

A crystal-clear understanding of what's actually needed is absolutely crucial before deciding to extend. This foundation determines whether an extension is truly necessary.

Avoid Extensions Where Possible

Extensions should be your last resort, not your first solution. Always ask: "Can we achieve this using standard functionality?" The answer might surprise you!

Use Cloud-Compliant Extensions

When extensions are necessary, use cloud-compliant extensions following the three-tier model to ensure compatibility and maintainability.

Practical Approaches to Clean Extensions

Now let's explore the practical side of maintaining clean extensions. The following strategies will help you implement extensions that add value without compromising your clean core.

These approaches represent best practices developed through years of SAP implementations and will guide you through the process of extending your system while maintaining its integrity.

Each strategy addresses a different aspect of the extension lifecycle, from governance to implementation techniques.

1. Establish a Governance Model

Structured Approval Process

Set up a structured process with clear criteria for approving extensions. This isn't about making it difficult to get extensions approved, but ensuring each extension adds value and follows best practices.

Key Governance Elements

Include clear approval workflows defining who needs to sign off on what, documentation requirements specifying what needs to be documented, quality standards establishing the bar for code quality, and regular review processes.

Continuous Oversight

Implement a system to regularly evaluate if existing extensions are still needed and meeting their intended business purpose, preventing extension sprawl.

2. Prefer Standard Over Custom

1

Explore Standard Options

Before writing any custom code, thoroughly investigate if standard features can meet your needs. Many organizations jump into custom development without fully exploring standard options.

2

Evaluate Use Case Frequency

Determine if you're trying to solve a rare use case that doesn't justify customization. Occasional manual workarounds might be more cost-effective than maintaining custom code.

3

Consider Long-Term Costs

Assess whether the customization is worth the long-term maintenance cost. The initial development cost is often dwarfed by ongoing maintenance expenses over the system's lifetime.

3. Use SAP Application Extension Methodology

Identify Right Approach

SAP's methodology helps you identify the appropriate extension approach for different scenarios, ensuring you use the most suitable technique for each business requirement.

Follow Best Practices

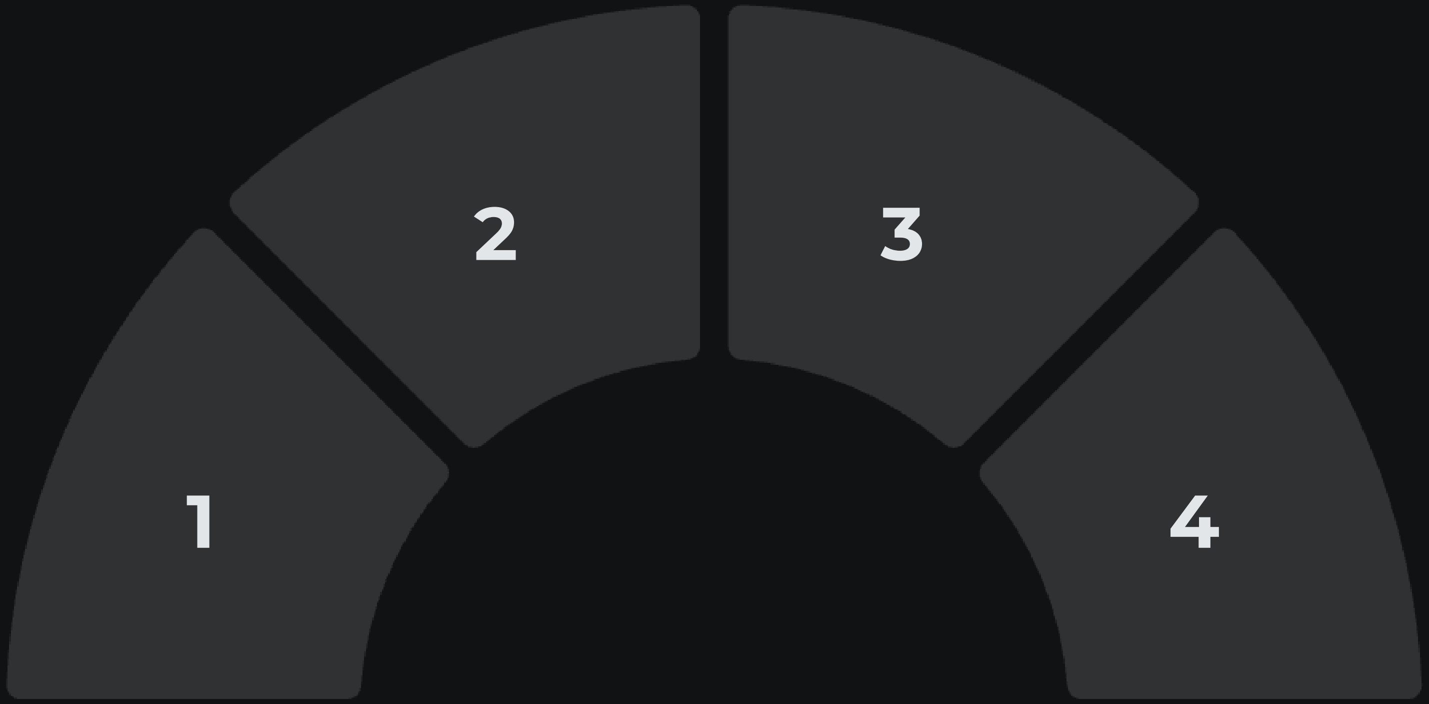
Leverage best practices that SAP has developed over years of experience, avoiding common pitfalls and implementation mistakes.

Maintain Compatibility

Ensure compatibility with future versions by following SAP's recommended approaches, reducing the risk of extensions breaking during upgrades.

Ensure Upgrade Safety

Create extensions that are upgrade-safe by design, minimizing disruption during system updates and reducing maintenance overhead.



4. Ensure Clear Separation

Separation Principle

Your extensions should be clearly separated from the core system, like building an addition to your house that connects seamlessly without modifying the original structure. This separation is fundamental to maintaining a clean core.

SAP BTP as Extension Platform

Whenever possible, use SAP Business Technology Platform (BTP) as your extension platform. It provides a clean separation between your custom code and the core system, creating a boundary that protects both sides.

Upgrade Safety Benefits

This clear separation is what provides the upgrade safety we've discussed. When your extensions are properly isolated, core system upgrades can proceed without breaking your custom functionality.

5. Use Released APIs

1

Stable & Supported

Released APIs are officially supported by SAP, providing stability and reliability for your extensions. They're designed specifically for integration purposes.

2

Upgrade-Safe

These APIs will continue to work after upgrades, protecting your extensions from breaking when the core system is updated. This significantly reduces maintenance overhead.

3

Well-Documented

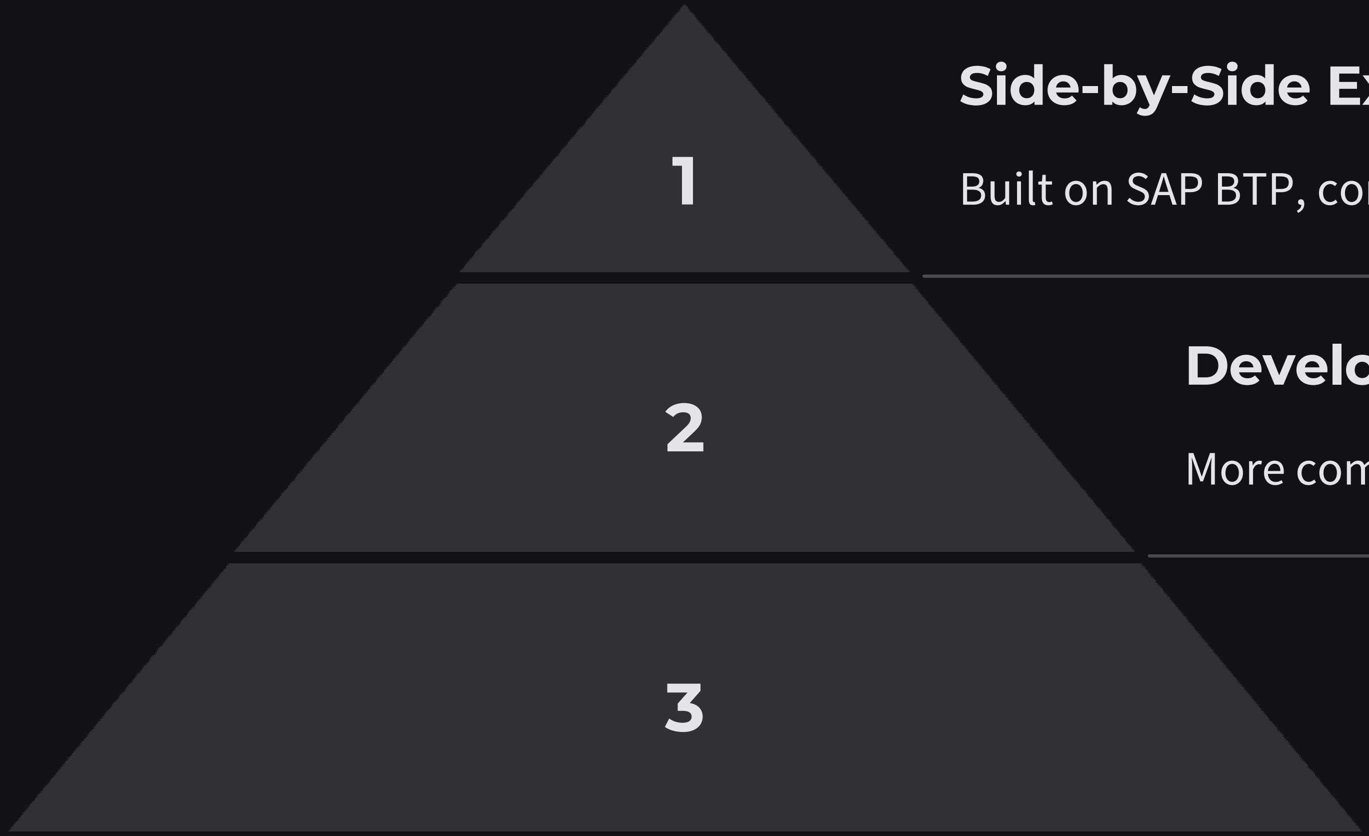
Released APIs come with proper documentation, so you know exactly how to use them correctly. This reduces development time and prevents errors.

4

Properly Maintained

SAP maintains these APIs over time, ensuring they remain compatible with new system versions and security requirements.

6. Choose Clean Extension Options



Side-by-Side Extensions

Built on SAP BTP, completely separated from core

Developer Extensions

More complex, require development skills

Key User Extensions

Simple adaptations without coding

SAP provides several types of extensions, and choosing the right one for your specific need is crucial. Key user extensions are perfect for UI changes and basic process modifications like adding fields to forms or rearranging screens.

Developer extensions use approved extension points that SAP has specifically designed for customization. Side-by-side extensions are ideal for complex custom logic that might not fit neatly into standard processes.

Real-World Example: AutoPro's Challenge

The Situation

AutoPro, a mid-sized automotive parts manufacturer, was struggling with their SAP system after making over 150 custom modifications directly in the core code over 8 years.

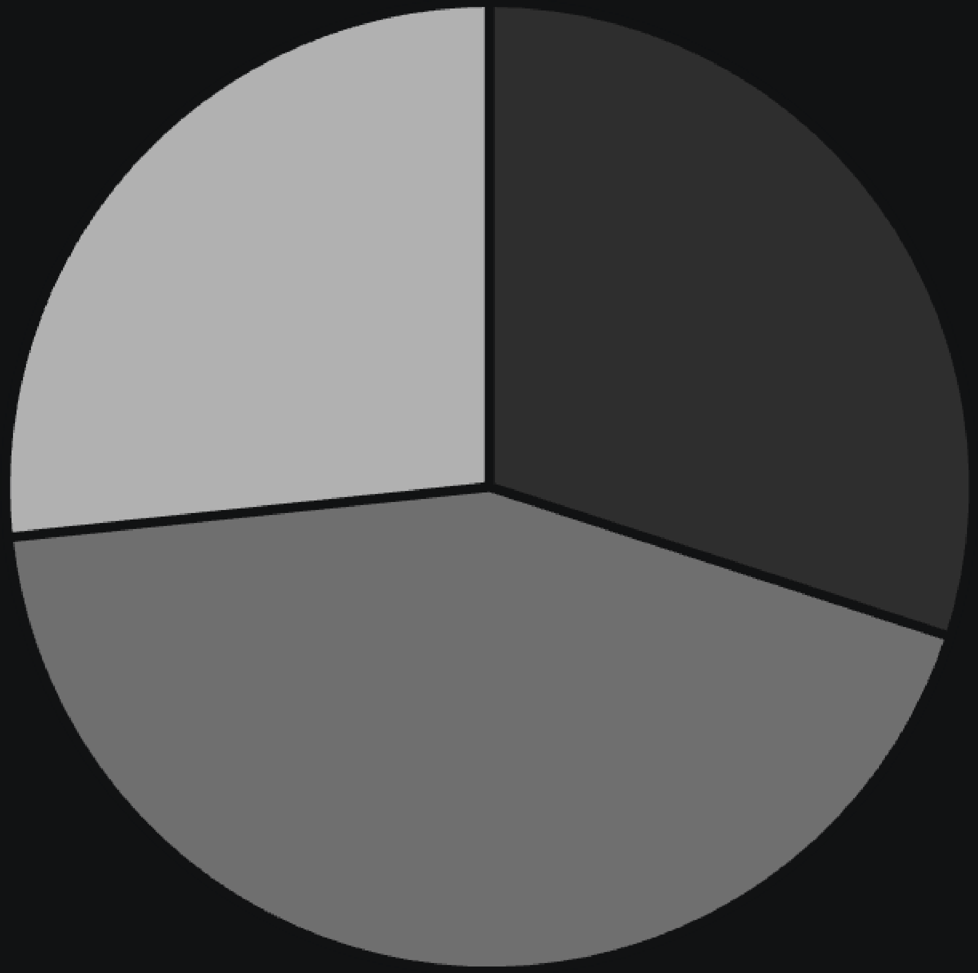
The Consequences

Every upgrade required about 3 weeks of system downtime. They needed 4 full-time developers just to maintain these customizations, and still faced constant performance issues.

The Crisis Point

When facing their next upgrade, the team estimated it would require 6 weeks of downtime—a complete disaster for a manufacturing company that relies on continuous operations.

AutoPro's Customization Analysis



■ Standard Features Available ■ UI-Related ■ Complex Business Logic

When consultants analyzed AutoPro's system, they discovered that about 45 customizations could actually use standard SAP features—the company simply wasn't aware these features existed. This is a common situation in many organizations.

Additionally, about 65 modifications were purely UI-related—changing how screens look, rearranging fields, and similar cosmetic changes. The remaining 40 involved complex business logic that would require more sophisticated solutions.

AutoPro's Transformation Strategy

1

Replace Unnecessary Customizations

They replaced all unnecessary customizations with standard features, eliminating redundant code that was duplicating existing functionality.

2

Move UI Changes to Key User Extensions

All UI modifications were moved to key user extensions, separating the presentation layer from the core system and making it upgrade-safe.

3

Develop Side-by-Side Extensions

For complex business logic, they developed side-by-side extensions on SAP BTP, completely isolating custom code from the core system.

4

Implement Standard API Integration

They implemented standard API integration for everything that needed to communicate with the core system, ensuring stable and supported connections.

AutoPro's Impressive Results

3 days

Upgrade Downtime

Reduced from potential 6 weeks to just 3 days

60%

Custom Code Reduction

Eliminated more than half of their custom code

25%

Performance Improvement

System response time improved significantly

66%

Faster Deployment

New features deployed in about a third of the time

The transformation also allowed AutoPro to reduce their development team from 4 full-time developers to just 2, creating significant cost savings while improving system performance and reliability.

The key takeaway is that modern extension strategies can transform heavily customized systems into manageable, upgradable platforms while preserving business-specific functionality.

Key Takeaways

1

Start With Standard

Always begin with standard functionality

2

Use Right Extension Type

Match extension type to specific needs

3

Document Everything

Thorough documentation is essential

4

SAP's extension methodology

They've put a lot of thought into this, so take advantage of it.

Follow SAP's extension methodology to take advantage of their expertise and established best practices. They've put significant thought into creating approaches that work in real-world scenarios.

Regular review and clean-up of extensions is essential for maintaining a healthy system. Your extension landscape should evolve with your business needs, removing outdated customizations when they're no longer needed.

Next time, we'll explore the Integration dimension of clean core, where we'll discover how to keep your system connections clean and efficient. Integration is often where complexity increases, making it another critical area for maintaining a clean core.