DZone

RPA vs. BPM, dos lados de la misma moneda

por Phani Krishna Kollapur Gandla · Sep. 05, 17 · DevOps Zone

Después de la nube, Robotic Process Automation (RPA) es la próxima palabra de moda, una tendencia emergente y una nueva generación de tecnología que está cobrando impulso en la era del Machine Learning y la Inteligencia Artificial.

Según el informe de Forrester Wave, Q1 2017 : "Las empresas se encuentran bajo una inmensa presión para digitalizar las operaciones, y ven un futuro en el que las operaciones de rutina están totalmente automatizadas. Estas empresas consideran RPA como parte de su estrategia de automatización".

Forrester estima que, para 2021, habrá más de 4 millones de robots realizando tareas administrativas, de ventas y relacionadas. La gestión de los robots y su gobierno será un problema creciente y los análisis avanzados agravarán estas preocupaciones a medida que los proveedores impulsen a RPA a un mayor valor.

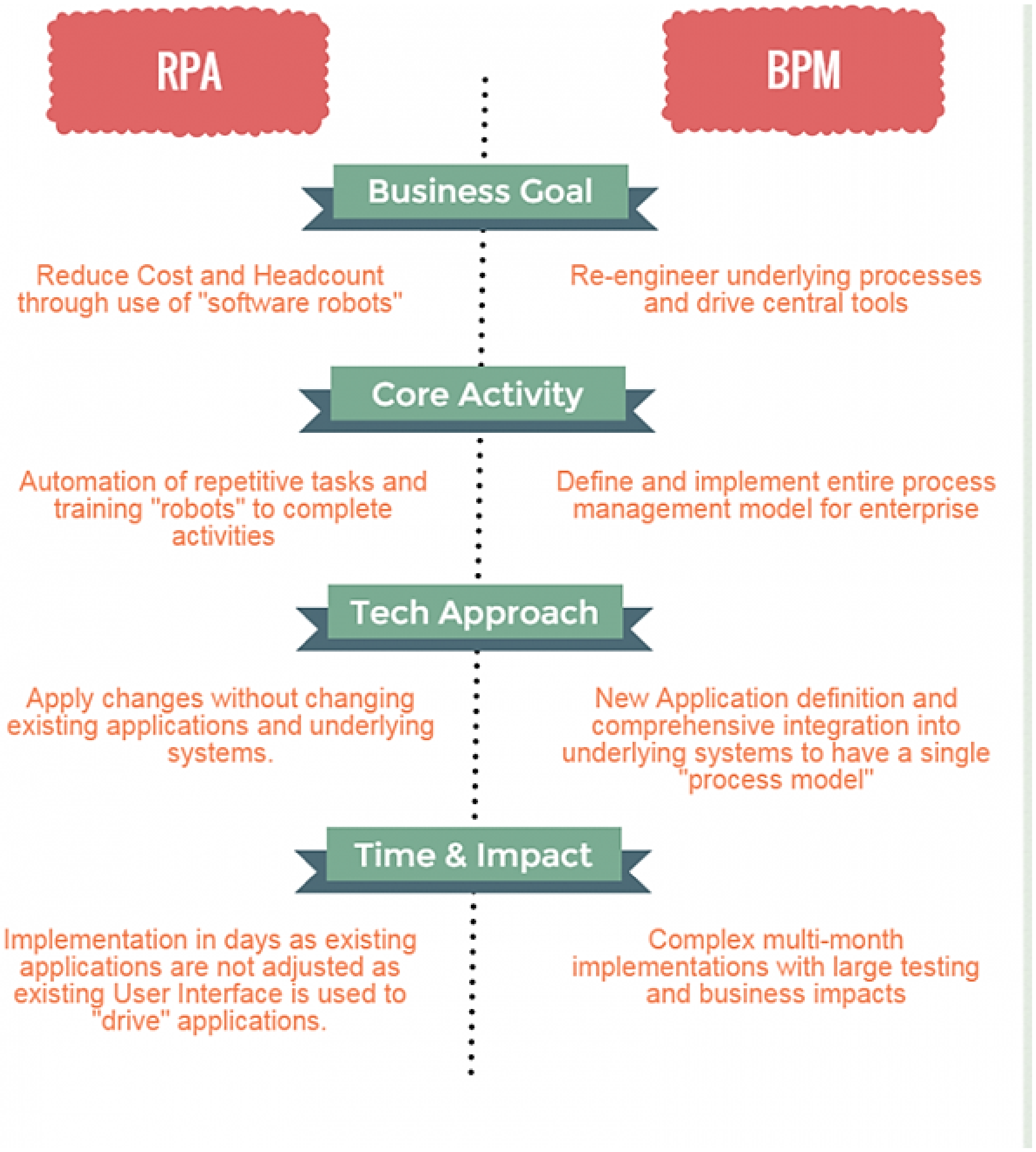
Pero a menudo uno podría obtener o plantear una pregunta sobre cuán diferente es RPA de Business Process Automation? ¿No es Business Process Automation lo mismo?

Las plataformas BPA / BPM les permiten a los equipos crear flujos de trabajo de automatización de procesos que pueden integrarse con sistemas múltiples y diversificados que intercambian información y manejan escenarios que abarcan la automatización de ciertas tareas humanas. El mayor inconveniente es que estas actividades requerirían API o acceso a la base de datos en esos sistemas, implican codificación y desarrollo y consumen mucho tiempo. ¿Qué sucede si los sistemas que se van a conectar no admiten ninguna integración de API, están fuera de la red o si el cliente no está dispuesto a compartir o dar acceso a API / Base de datos?

Aquí es donde superan las plataformas RPA; ya que tienen un marco muy sofisticado para configurar robots de software para capturar y ejecutar inteligentemente tareas repetitivas no programáticas que anteriormente requerían un ser humano para realizarlas.

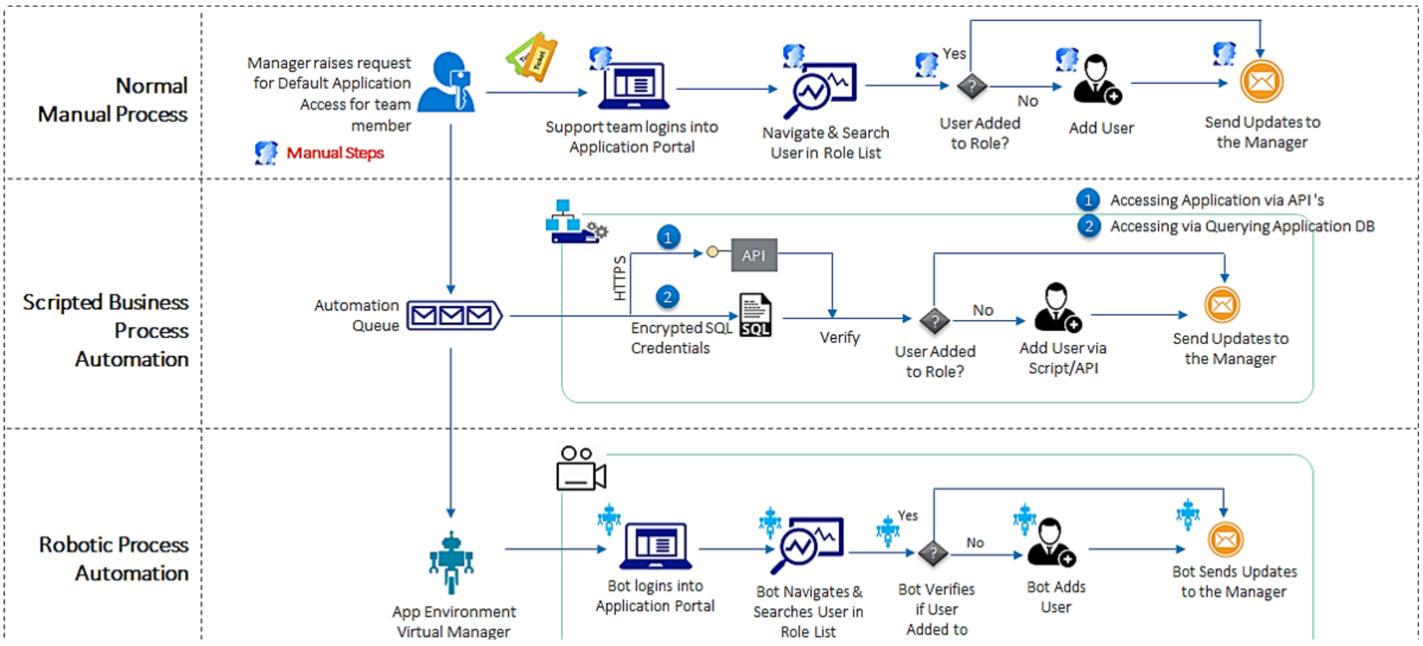
La buena noticia es que RPA está completamente libre de códigos y agnóstico de plataforma.

Una comparación de muestra de Agile Business Technologies:



RPA		BPM
Reduce Cost and Headcount through use of "software robots"	**Business Goal**	Re-engineer underlying processes and drive central tools
Automation of repetitive tasks and training "robots" to complete activities	**Core Activity**	Define and implement entire process management model for enterprise
Apply changes without changing existing applications and underlying systems.	**Tech Approach**	New Application definition and comprehensive integration into underlying systems to have a single "process model"
Implementation in days as existing applications are not adjusted as existing User Interface is used to "drive" applications.	**Time & Impact**	Complex multi-month implementations with large testing and business impacts

A continuación se muestra un escenario empresarial simple y de ejemplo para proporcionar acceso de usuario basado en roles a la Aplicación de forma manual y utilizando el proceso BPA y RPA.


Normal Manual Process
Manager raises request for Default Application Access for team member
Manual Steps
Support team logins into Application Portal
Navigate & Search User in Role List
Yes
No
User Added to Role?
Add User
Send Updates to the Manager
Scripted Business Process Automation
Accessing Application via API's
Accessing via Querying Application DB
Automation Queue
HTTPS
API
Encrypted SQL Credentials
SQL
Verify
No
User Added to Role?
Add User via Script/API
Send Updates to the Manager
Robotic Process Automation
App Environment Virtual Manager
Bot logins into Application Portal
Bot Navigates & Searches User in Role List
Yes
No
Bot Verifies if User Added to
Bot Adds User
Bot Sends Updates to the Manager

Un flujo de trabajo típico sería el siguiente:

- Project o IT Manager plantea una solicitud de acceso a la aplicación basada en roles para el miembro del equipo.
- El ticket es creado, enviado y reconocido por el ingeniero de soporte.
- El equipo de soporte inicia sesión en la aplicación con credenciales de administrador.
- Navega, busca y verifica si el usuario ya está presente como parte del rol.
- Si no está presente, agrega al usuario a la aplicación.
- Envíe las actualizaciones al administrador del proyecto y cierre el ticket.

Invocar un proceso similar a través de la automatización de procesos de negocio con guiones requeriría la creación de un nuevo flujo de procesos que implique el acceso de back-end a través de llamadas API / DB activadas desde una Cola de Automatización. Estas actividades requerirían una cuenta de servicio o credenciales de base de datos, que deben encriptarse y persistir para realizar las acciones respectivas en el sistema. La implementación del flujo de trabajo de proceso personalizado podría tener que pasar por el ciclo de vida de desarrollo de aplicaciones.

En el contraste, las herramientas de RPA ayudan a los equipos de soporte a registrar varias actividades de rutina realizadas (Capturar y Responder). Las herramientas RPA líderes en la industria como Automation Anywhere, UiPath, Blue Prism tienen incorporados diseñadores de estudio para crear visualmente modelos de automatización sin ningún código. Las operaciones manuales completas pueden ser grabadas, programadas y activadas por un robot según sea necesario. Los modelos se pueden construir con facilidad, en menor duración y con cero codificación.

Entonces, ¿qué modelo debería uno seleccionar?

Bueno, la respuesta depende del objetivo del proyecto de automatización que se construirá. Si la intención es automatizar las actividades humanas principales y repetitivas, RPA será la mejor opción y, por otro lado, si los flujos de los procesos de automatización que se implementarán son complejos, implica la simplificación de múltiples sistemas y plataformas, el software BPM sería el objetivo. a la plataforma.

Temas: RPA, AUTOMATIZACIÓN, BPM, IT, DEVOPS

7 Ways to Know You've Aced Continuous Integration

by Brian Dawson MVB · Mar 12, 18 · DevOps Zone

One of my favorite moments at Jenkins World last year came when tech guru Jez Humble, during his keynote address, led the crowd in a quick show of hands about who's truly doing continuous integration (CI). Since the seats were filled with some of the most dedicated DevOps pros you'll find anywhere, one would assume that everybody would be totally on the ball with continuous integration.

Humble asked a few leading questions:

- Who isn't feeding builds into their organization's main pipeline regularly?
- Who can't fix a software issue in under 10 minutes? Those who can't or don't, put your hands down.
- A number of hands dropped after each question. Finally, a fraction of the original number of hands were still in the air.

Humble's message: A large percentage of people think they're doing continuous integration, but they really aren't.

"Continuous integration is really hard," Humble told the crowd. "And a lot of people redefine it to

"Continuous integration is really hard," Humble told the crowd. "And a lot of people redefine it to mean whatever they're already doing."

Are you really doing continuous integration? It's an important question. Continuous delivery (CD) and DevOps, after all, are disrupting the market and providing businesses with a huge competitive advantage. Continuous delivery is built on the tried-and-true practices of continuous integration. Organizations that seek to recognize the benefits of continuous delivery have often failed to fully understand the concepts of continuous integration.

As Humble related at Jenkins World during his keynote address, the organizations that are doing continuous integration correctly are all following a few basic rules. For example, developers' working copies are synchronized with a shared mainline at least daily, preferably several times a day. Each integration gets verified by an automated build to detect errors as quickly as possible. Organizations that aren't following these steps aren't really doing continuous integration properly.

Continuous integration is a development team practice that generates real benefits for an entire organization. Engineers in charge of implementing continuous integration practices want to achieve those benefits and follow the modern practices their peers are following. They're just running into problems trying to make it happen.

They hear about how other organizations are implementing continuous integration, and then make decisions about whether they can do the same inside their four walls. But every organization is different. The DevOps team may have a vision for what continuous integration looks like in their organization, but it might not fit squarely inside commonly accepted definitions of CI.

Organizations that don't follow the core principles of continuous integration very likely will run into problems delivering crisp, functioning builds on a regular basis. Over time, the initiative will lose momentum and team members will become disenchanted. People who are resistant to change (which is most of us) will revert back to their old practices, if they don't see evidence of the benefits of change, even in small increments. In this case, you have multiple problems. Your builds are still riddled with errors. Your team has lost faith in the implementation. You've lost critical time, and now you need to restart the project.

Organizations that don't implement continuous integration correctly often face cultural issues. Engineers are great at solving technical problems, but CI requires a shift in culture. Culture is hard to change. You can bring in a continuous integration tool and check most of the boxes that apply to what CI is supposed to represent, but succeeding with CI requires a change in how you work and *how you work together*. If the culture of the team doesn't change, they're going to have a hard time implementing continuous integration.

Bottom line: Doing continuous integration is like putting on a raincoat over your best suit. You have committed to aggressive delivery schedules and promised stellar versions of the company's new app because you're confident that the coat provided by your continuous integration principles will protect your project. If the raincoat isn't waterproof, your suit will get ruined. Same situation with CI. Leave your project open to the elements, and it could end up underwater.

If it sounds like properly aligned CI processes are designed to eliminate errors, that's not true. Continuous integration itself is a process designed to embrace failure. We want to create a system where developers can fail often - and fail fast, so they can find and fix errors early and quickly. Continuous integration is like a guardrail on a windy road - giving developers the confidence to drive fast, because they're protected if their builds veer too close to the edge.

The core principles and practices of continuous integration date back at least 15 years, when Martin Fowler introduced the term, and they apply today just as they applied then. Here are seven practices organizations need to follow to truly do CI correctly:

1. **Commit to the mainline:** This is table stakes for continuous integration. A developer can set up an automated build and have the build run on every commit. But if the culture is to not commit frequently, it won't matter. If a developer waits three weeks to commit or branches off for three weeks, he has delayed the integration and broken the principles. If a build breaks, the team has to sort through three weeks of work to figure out where it broke.
2. **Maintain a single-source repository:** In complex applications, developers often branch and maintain changes off of a trunk (branch) or main. The branching creates complexity and prevents everyone working with a single source of truth. Teams need to commit/merge to trunk or main at least once per day, or even better for every change.
3. **Automate the build:** This is a practice most organizations tend to do well. However, what some who claim to practice CI are simply doing is scheduled builds (i.e. nightly builds), or continuous builds but they are not actually testing or validating each build. Without validation of the build, you are not doing continuous integration.
4. **Make builds self-testing:** The first step of the validation process is to know that a build with problems actually failed. The next step is to determine if the product of the build is operational and that the build performs as we expect it to. This testing should be included as part of the build process. This consists of fast functional and non-functional testing.
5. **Build quickly:** If it takes too long to build an app, developers will be reluctant to commit changes regularly or there will be larger changesets. In either case, no one will spot a failure quickly. By building quickly and integrating quickly, you can isolate changes quickly. If it takes hours to run, you might have 20 to 30 more changes during that time and it will be difficult to quickly spot problems.
6. **Test in a clone:** Validation processes verify that software performs as expected in its intended environment. If you test in a different kind of environment, it may give you false results.
7. **Fix broken builds immediately:** It's critical for development teams to find problems fast and fix them immediately, so they don't move downstream. Years ago, Toyota instituted a "stop-the-line" approach where workers could pull a rope and stop the manufacturing process if they spotted a problem. CI sets up a process where builds are validated and committed continuously, so if something goes wrong, it'll be easy to fix.

Despite all the challenges organizations face implementing true continuous integration, it's important to note how far the software development community has come in following modern processes that create true value for their operations. Many are working hard to make changes and improve their DevOps practices. The biggest obstacles to pristine CI are our cultural, emotional and technical attachments to legacy technologies. These are the enemy of change. Even when you get beyond the "culture of can't," carrying the baggage of legacy practices can provide a formidable obstacle.

The other challenge organizations face is overplaying their hands. Everybody wants to transform their operations using methodologies like CI, but few want to talk honestly about how they still need to improve their processes.

More than a decade after continuous integration first became an industry term, we're still trying hard to do it right.

Want to learn more about Jenkins? **Sign up for the Jenkins Continuous Information Newsletter**. Brought to you in partnership with CloudBees.

Topics: DEVOPS, CONTINUOUS INTEGRATION, JENKINS WORLD