Ingeniería de Software para la operación y gestión IT en organizaciones medianas

Ricardo Di Pasquale

Universidad Católica Argentina, República Argentina.
rdipasquale@uca.edu.ar

Resumen: En este artículo estamos interesados en proponer la aplicación de las técnicas y metodologías de la Ingeniería de Software en la Operación y Gestión de la Infraestructura IT de determinadas organizaciones. Dichas organizaciones se describen como de tamaño mediano con capacidad de gestión y desarrollo IS/IT propio. Se describe la visión y experiencia de un Ingeniero de Software en dichas organizaciones. Particularmente se enfoca en la diferencia de criterios más comunes entre las gestiones clásicas llevadas a cabo por los analistas IT y la aplicación de metodologías y métricas por parte de los Ingenieros de Software. Se fundamenta la necesidad de que que la conducción de las mejoras en gestión y operación IT de una organización de este estilo sean direccionadas por la Ingeniería de Software. Se presentan brevemente las principales tecnologías "Platform as a Service (PaaS)". Se elabora un catálogo de técnicas y metodología de la Ingeniería de Software a aplicar en la gestión y operación IT. Se exponen lecciones aprendidas, recomendaciones, líneas futuras de trabajo y conclusiones.

1 Introducción

En una organización de tamaño medio pueden convivir soluciones de hardware (servidores) de distinta naturaleza, desde pequeños servidores individuales a cajones de servidores tipo Blade. Este tipo de situaciones de convivencia heterogénea no sólo se da con el hardware, sino con las soluciones de software. Es común encontrar organizaciones medianas que han implementado combinaciones de desarrollos ad hoc, plataforma SAP, soluciones Oracle, Microsoft, etc. Por tanto, el sujeto de estudio de este artículo son las organizaciones de mediana magnitud con capacidad de gestión IT autónoma, incluso las que hayan avanzado tecnológicamente hasta el punto de virtualizar infraestructura.

Los grandes esfuerzos realizados por la industria a la hora de normalizar la operación y gestión IT desembocaron en la proliferación de marcos de referencia como ITIL, que consiste en una librería de buenas prácticas y guías operativas no necesariamente coherentes entre sí que deben seguirse con el fin de mejorar la calidad y eficiencia de las operaciones IT.

Los ingenieros y arquitectos de software, si bien suelen destacar los logros por el apego a ciertos marcos de trabajo como CMMI o metodologías ágiles, suelen hacer críticas a enfoques del estilo ITIL. Solo para ilustrar el punto anterior, se puede decir que, desde el punto de vista de BPM (*Business Process Management*), los ingenieros de software pueden observar un uso incorrecto del concepto de "Proceso", o mínimamente un peligroso abuso de notación, tal como se señala en [Betz, 2011].

Si bien, dentro de la organización de ITIL, existe un esfuerzo importante por llevar ITIL a las organizaciones pequeñas y medianas, son raros los casos de éxito documentados en dichas organizaciones.

El objetivo de este artículo no es criticar la implementación de ITIL (u otras alternativas) a la gestión y operación IT de las organizaciones en cuestión, sino proponer en base a la evidencia, que la conducción de las mejoras en gestión y operación IT de una organización de este estilo sean direccionadas por la Ingeniería de Software. Dicha propuesta no choca de ninguna manera con la implementación de ITIL u otros estándares.

2 El devenir de Hardware en Software

El proceso de virtualización de la infraestructura mejoró de manera notable el servicio entregado por las áreas de infraestructura de las organizaciones. Pero muchos de los vicios acarreados de la era del "hardware real" siguen en pie. Al poder expandir la infraestructura de manera más fácil, se da la situación en la que una organización mediana llega a tener una cantidad desmedida de servidores (por ejemplo: más servidores que empleados). Cada servidor, independientemente de si su naturaleza es real o virtual, requiere tiempos de operación. Por tanto, estos tiempos, y estos riesgos, comienzan a multiplicarse al elevarse la cantidad de servidores. Complejidades adicionales se agregan cuando las organizaciones medianas, sujetos del presente artículo, no tienen totalmente pulidos los esquemas automatizados de operación. Para ilustrar este hecho con un ejemplo, vale traer a colación la complejidad que podría acarrear la instalación de actualizaciones de los sistemas operativos, sobre todo en ambientes en donde no se encuentran debidamente clasificadas las mismas (según su importancia o criticidad o naturaleza): se pierden horas valiosas de recursos capacitados para actualizar (y en algunos casos reiniciar) una cantidad desmedidamente elevada de servidores.

La percepción de la infraestructura como código fuente proviene del concepto conocido como *Infrastructure as code (IaC)*. El mismo, tal como se observa en [Wittig et al., 2016], consiste en el *proceso de administrar, operar y proveer infraestructura de cómputo (servidores virtuales o reales) y su configuración a través de archivos de definición procesados por software de automatización, en vez de la configuración física del hardware o el uso de herramientas de configuración interactivas*. Puntualmente, esta disciplina está asociada a la provisión de entornos virtuales, servers, etc. La

implementación de *IaC* implica la existencia de Administración de la Configuración[1] (*Configuration Management*). La implementación de este mecanismo, además de proveer eficiencia y repetibilidad (ya provistas por la Administración de la Configuración), provee ciertas ventajas, como la generación de un repositorio de software[2] donde comienzan todos los despliegues de aplicaciones; el software generado sirve también como documentación de la infraestructura; la adopción de repositorios de código brinda todas las ventajas que son necesarias en la disciplina de *Software Configuration Mamagement* (SCM) [Pressmann, 2009], como por ejemplo el mantenimiento de líneas base, la gestión de versiones, el historial de modificaciones y el *branching* de versiones.

Las herramientas más importantes dentro del marco de *IaC* y la gestión de recursos en nube son, por difusión en la industria e implementaciones exitosas, *Puppet* [Loope, 2011] y *Chef* [Marschall, 2013], que son, específicamente, herramientas de *Configuration Management*. La irrupción de plataformas de *cloud computing*, y más específicamente, la posibilidad de poder establecer nubes de recursos privadas de manera eficiente, permitió la evolución y adopción de este tipo de herramientas de manera natural.

Como se verá más adelante, la pieza clave para la aplicación de la ingeniería de software en la operación IT es la tecnología de contenedores, donde la herramienta por excelencia es Docker [Turnbull, 2015]. Esta plataforma provee la posibilidad de desplegar aplicaciones de cualquier índole sobre contenedores de software. Estos son abstracciones similares a sandbox en donde corren, controladas, las aplicaciones. A diferencia del esquema de virtualización de servidores, no corren un sistema operativo, ni virtualizan los recursos del hardware que se encuentra en una capa de abstracción debajo. De esta manera, un desarrollador puede disponer de un entorno exactamente igual al que utilizará su aplicación en ambiente de producción. Desde el punto de vista de operaciones IT, las ventajas de la utilización de contenedores, radica en la posibilidad de proveer un servicio más cercano al *zero downtime* o, incluso, a niveles de *alta disponibilidad*, mediante la elaboración de una estructura de despliegue de contenedores, que incluya varias copias del mismo en distintos procesos, de modo que la ejecución puede "migrar" entre nodos.

La combinación de todas las tecnologías anteriores, permite la elaboración del concepto de *Platform as a Service (PaaS)* [Kavis, 2014], que consiste en el servicio de poner a disposición del cliente toda una plataforma de software en la nube que incluye el hardware donde corre. En el caso de este trabajo, se habla de nubes privadas de una organización. *OpenShift* es una plataforma de código abierto provista por Red Hat para la gestión de Private PaaS clouds [Pousty et al., 2014], y parece perfilarse como el gestor adecuado a la tipificación de la organización, debido tanto a los costos de implementación, como a la diversidad de tecnologías soportadas.

[1] Ya sea vía ITIL, con otro marco procedimental, o ad hoc.
[2] Típicamente git

3 La irrupción de DevOps

Existe una cantidad notable de literatura que refleja, de forma novelada [Kim et al., 2014], o de forma de crónica [Reed, 2015], el devenir de las tareas de los ingenieros de operaciones IT en la industria tecnológica. Todos estos relatos resumen vivencias de varios perfiles de profesionales de sectores IS/IT dentro de las organizaciones. Dentro del área de operaciones IT, existe un factor común en estos relatos: el hartazgo respecto de la tarea sumamente repetitiva de despliegue de aplicaciones[3], así como de la impotencia ante las caídas de servicio (sean por problemas de la infraestructura o por problemas del código).

En la conferencia Agile 2008, Andrew Clay Shafer y Patrick Debois disertaron sobre la aplicación de metodologías ágiles en el ámbito de la gestión de infraestructura. El término comenzó, entonces, a popularizarse hasta tomar dimensión propia y ser adoptado por una comunidad cada vez más importante en el ámbito IT. Tal como se puede observar en [Hüttermann, 2012] , el término DevOps *es una contracción de Development y Operations (IT) que consiste en una conjunto de patrones que tienden a mejorar la colaboración entre los sectores de desarrollo y operaciones, direccionando metas e incentivos compartidos, así como, procesos y herramientas compartidas. Es importante destacar que la cultura DevOps tiende a destacar a la gente por encima de los procesos y que considera inevitable los conflictos entre los sectores de Desarrollo y Operaciones*.

4 La relación entre los sectores de IS/IT

El estado del arte de las relaciones entre sectores IS/IT en las organizaciones que han mejorado sus procesos a través de la virtualización de recursos, pero que no se han despegado de la cultura de la infraestructura real se puede representar de la siguiente manera:

[3] En los casos referenciados se trata generalmente de sitios web transaccionales de cierta envergadura.

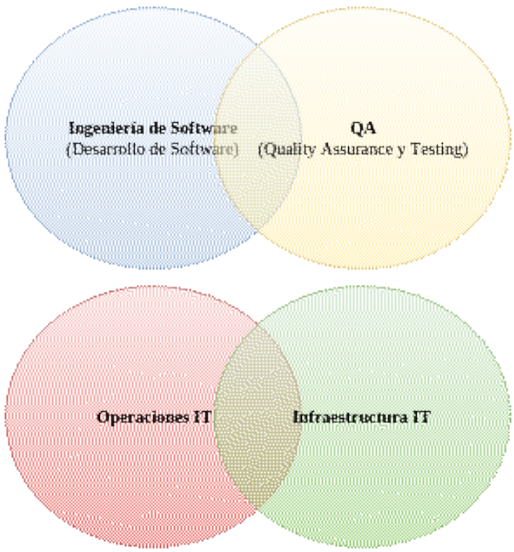

Fig.1 - Relaciones entre sectores IS/IT en organizaciones de referencia.

Cuando se avanza con el proceso de virtualización, racionalizando los servidores e introduciendo la noción de nube de recursos, y donde se comienzan a utilizar las técnicas y metodologías de la ingeniería de software en la operación y gestión IT, las relaciones entre los sectores de IS/IT comienzan a tomar otra forma, en donde se integra fuertemente la operación IT al desarrollo:

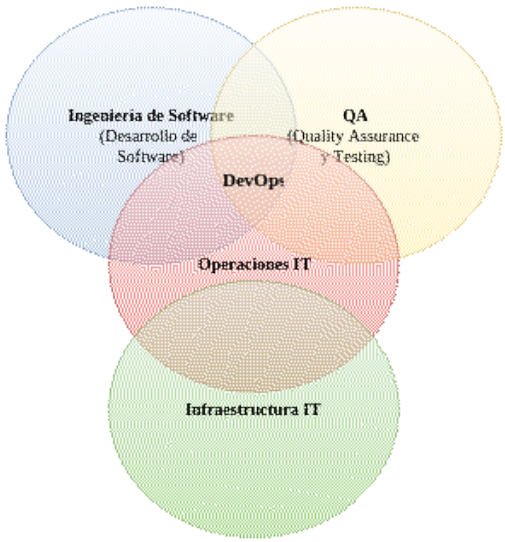

Fig.2 - Relaciones entre sectores IS/IT en organizaciones que adoptan DevOps.

5 La aplicación de Ingeniería de Software a la operación y gestión IT

La idea de automatizar operaciones de IT no es nueva ni revolucionaria. Históricamente los administradores POSIX (Linux y Unix) han utilizado herramientas de scripting para lograr dicho objetivo. Con los esquemas de virtualización, seguramente se incrementó la tendencia a programar operaciones. Lo que puede considerarse *nuevo*, históricamente hablando, es la presentación de los recursos de hardware y operaciones como un recurso más de software. Si realmente la gestión y operación mayoritaria de recursos IT es, ahora, software, podemos aplicar, entonces, técnicas y metodologías de la Ingeniería de Software para la gestión de los mismos. Y es entonces donde aparece un nuevo concepto: No se trata de envolver varias tecnologías en un paquete nuevo y ponerles un nombre tecnológicamente atractivo, sino en la aplicación de una disciplina ya establecida (la Ingeniería de Software) sobre un dominio nuevo (la gestión y operación IT).

¿Cuál es la Diferencia entre *IaC* y la ingeniería de software aplicada a la operación y gestión IT? Una cosa es programar un script para hacer una tarea de operación o gestión IT. Otra muy distinta es conceptualizar dicha tarea o dicho recurso como software, incorporarlo al proceso de desarrollo de aplicaciones, someterlo a Software Configuration Management [Pressman, 2009], a Integración Continua (Continuous Integration [Duvall et al., 2007]), a Pruebas Unitarias [Massol et al., 2004] o Test Driven Development [Stephens et al., 2010], etc. En efecto, se trata de algo nuevo, y no de, únicamente, la construcción de una colección de scripts que automaticen tareas.

Las principales metodologías, herramientas y técnicas de la ingeniería de software a aplicar, pueden enumerarse de la siguiente manera:

- **Reingeniería** [Pressmann, 2009]: Los servidores de aplicación existentes deberán someterse a un proceso de reingeniería. No alcanza con una refactorización, ya que no es el objetivo convertir los servidores reales o virtuales en contenedores o scripts del Configuration Manager, sino racionalizarlos, orientarlos a servicio y construir, entonces, los nuevos artefactos de software en función de dichas especificaciones. Esta idea se hace extensiva a las pŕacticas de operaciones IT. Este tipo de procesos, llevará indefectiblemente a una racionalización de los servidores: En una organización con una cantidad desmedida de servidores (virtuales), conviene elaborar la reingeniería de servidores como primera etapa, ya que el tiempo invertido en su mantenimiento, y la complejidad de su gestión pueden ser prohibitivos. Menos servidores virtuales implican menos instancias de sistemas operativos requiriendo recursos de hardware.
- **Refactorización**: No todo el código de las aplicaciones existentes está habilitado para independizarse de los servidores, y comenzar a correr en contenedores. Es entonces, fundamental, no afectar la funcionalidad de dicho código, pero refactorizarlo para poder, entonces llevarlo a un esquema de contenedores, construyendo dichos artefactos.

- **Software Configuration Management** [Pressman, 2009]: Dado que una vez montada la estructura PaaS, prácticamente todos los componentes son software, se estará en condiciones de compartir el repositorio de Software, de manera de contar tanto con el código fuente de las aplicaciones, las estructuras de datos, las especificaciones de los contenedores, las configuraciones de los servidores, los servidores en sí mismos y otros componentes. El poder que brinda la posibilidad de establecer líneas base no sólo del código fuente de un producto, sino de toda la pila de tecnología asociada es, realmente, notable, y debe considerarse un triunfo de la Ingeniería de Software.
- **Integración Continua** [Duvall et al., 2007]: Esta metodología ágil tiene como fundamento construir constantemente los paquetes entregables de software, así como aplicar el proceso automatizado de testing, con el fin de mejorar los ciclos de desarrollo, así como, la detección temprana de fallos. La posibilidad de incluir en este circuito a los contenedores y la infraestructura, permite no sólo extender el alcance de la técnica, sino convertirla en una práctica necesaria a la hora de garantizar el funcionamiento de los releases incrementales del software de aplicación. Cabe destacar que la plataforma más difundida de integración contínua, Jenkins [Smart, 2015], se integra con Docker mediante un esquema de plugins.
- **Pruebas Unitarias** [Massol et al., 2004] y **Test Driven Development** [Stephens et al., 2010]: Cuando los desarrolladores escriben pruebas unitarias, sea bajo el paradigma TDD, DDT [Stephens et al., 2010] o no, establecen un punto de partida para la corrida de una prueba, la prueba en si, la evaluación de los resultados de la prueba (oráculo) y la limpieza del entorno. Dos nuevos niveles de abstracción puede incorporarse al paradigma, dado que se pueden utilizar contenedores para las pruebas, así como probar los contenedores; y, utilizar la infraestructura para las pruebas, así como probar la infraestructura mediante pruebas unitarias automatizadas.
- **Planificación de Sistemas**. Plan Táctico - Plan anual de Operaciones: En este tipo de planificación, suelen observarse requerimientos al sector de desarrollo por parte del sector de gestión IT cuando es necesario reemplazar equipamiento. Este tipo de requerimientos suele capturar recursos valiosos de todas las áreas. Bajo el nuevo paradigma, los tiempos se reducen drásticamente, y, el proyecto resultante puede someterse a las otras técnicas de Ingeniería de Software, de manera que, también, los riesgos se reducirán.
- **Métricas**. Durante el proceso de incorporación de la tecnología PaaS, es esperable, que, bajo el marco de la Ingeniería de Software, surjan modelos de métricas específicos para los “nuevos” artefactos de software.

6 Lecciones aprendidas, conclusiones y futuras líneas de Trabajo

Las habilidades necesarias para el perfil del analista IT a cargo de la operación o gestión de infraestructura IT necesariamente van a combinarse con las habilidades del desarrollador de software. O, más bien, la división de tareas en una organización, conducirá a especializar al analista IT en la organización del hardware a bajo nivel; y, por otro lado, a recursos con formación de desarrolladores, en analistas *DevOps* para la mayoría de las tareas de operación IT, así como de algunas de las tareas de gestión IT. Incluso, en la práctica, se da que el desarrollador de software no sólo genera el código de las aplicaciones que desarrolla, sino que puede desarrollar la infraestructura en donde corre el código. La experiencia concreta indica que es más productivo formar equipos especializados de *DevOps* con las habilidades necesarias para desarrollar el código de la infraestructura, con la ventaja que comparten el arte de la programación como nexo comunicativo fundamental.

Una lección importante es no innovar de manera temprana con plataformas Legacy complejas como Sap u Oracle (sobre todo si hay involucrados clusters reales). Es por eso, que se recomienda no incluir este tipo de plataformas en el primer alcance del proyecto, esperando adquirir la experiencia suficiente como para tomar la mejor decisión en una iteración posterior.

Otra lección importante es saber entender que las métricas y monitoreos de las plataformas durante un proyecto de este estilo pueden ser sensibles para los integrantes de los equipos, por lo que se recomienda buscar la manera de que la responsabilidad por el monitoreo funcional de la infraestructura recaiga en un equipo Independiente de los Analistas IT.

Si bien es posible establecer un marco *PaaS* sin una herramienta de gestión (como *OpenShift*), tal como se ilustra en [Hane, 2015], se evidenció que no es la mejor manera de introducir *PaaS* en la organización, dado que las plataformas facilitan gestiones engorrosas para equipos que no están demasiado familiarizados con contenedores de software.

Se evidenció que a medida que pasa el tiempo, la configuración de los servidores de la infraestructura se vuelven cada vez más inconsistentes o diferentes de lo que se esperaría de la línea base (o template). Este fenómeno se conoce como *Configuration Drift* [Morris, 2015] y se puede deber a cambios ad hoc, actualizaciones del sistema operativo o software de base, o desorden o "entropía" general.

Como líneas futuras de investigación conviene destacar un fenómeno difícil de medir: la ocurrencia de situaciones en las que un desarrollador o un analista de QA certifican cierto funcionamiento, que no ocurre en la infraestructura productiva. Estos casos se los suele denominar "*En mi máquina funciona*". Se espera que estos incidentes desaparezcan de la organización, pero, en función de que es imposible asegurar que esto ocurra, es conveniente hacer un seguimiento de los mismos para analizarlos en el futuro.

Otra línea de investigación futura tiene que ver con la incorporación, o no, de los bloques legacy monolíticos (como Sap y plataformas complejas Oracle, o similares).

Como conclusión general se puede destacar que la aplicación de la Ingeniería de Software a la Operación y Gestión IT en las organizaciones referenciadas es totalmente posible. Es independientemente de la implementación de marcos estándares como ITIL. No se contradice con dichos estándares. Es posible tener las expectativas de mejoras que los marcos de referencia de la Ingeniería de Software suelen brindar, aplicadas a las operaciones y gestión IT. De hecho, las mejoras esperadas exceden las que puede prometer la aplicación sistemática de técnicas y metodologías, ya que se espera una mejor integración entre las áreas de Desarrollo y Operaciones y Gestión IT, lo que redunda, en un mejor ambiente de trabajo.

Referencias

[Betz, 2011] - "ITIL®, COBIT®, and CMMI®: Ongoing Confusion of Process and Function" - Charles T. Betz - BPTrends 10/2011 - Online al 31/03/2016 en http://www.bptrends.com/publicationfiles/10-04-2011-ART-Ongoing%20Confusion%20of%20Process%20and%20Function-Betz-Final.pdf

[Duvall et al., 2007] - "Continuous Integration: Improving Software Quality and Reducing Risk" - Paul M. Duvall, Steve Matyas & Andrew Glover - 2007 - Addison-Wesley - ISBN: 978-0-321-33638-5.

[Hane, 2015] - "Build Your Own PaaS with Docker" - Oskar Hane - Pakt Publishing - 2015 - ISBN: 978-1-78439-394-6.

[Hüttermann, 2012] - "DevOps for Developers" - Michael Hüttermann - 2012 - Apress - ISBN: 978-1430245698.

[Kavis, 2014] - "Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)" - Michael Kavis - Wiley - 2014 - ISBN: 978-1-118-61761-8.

[Kim et al., 2014] - "The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win" - Gene Kim, Kevin Behr & George Spafford - 2014 - ISBN: 978-0-9882625-8-4.

[Käsper, 2015] - "Building high availability systems using Dockers" - Taivo Käsper - University of Tartu - 2015.

[Loope, 2011] - "Managing Infrastructure with Puppet" - James Loope - 2011 - O'Reilly

[Marschall, 2013] - "Chef Infrastructure Automation Cookbook" - Matthias Marschall - 2013 - Pakt Publishing - ISBN: 978-1-84951-922-9.

[Massol et al., 2004] - "JUnit in action" - Vincent Massol & Ted Husted - 2004 - Manning - ISBN: 1-930110-99-5.

[Morris, 2015] - "Infrastructure as Code: Managing Servers in the Cloud" - Kief Morris - O'Reilly Media - March 2015.

[Pousty et al., 2014] - "Getting started with OpenShift" - Steven Pousty & Katie J. Miller - 2014 - O'Reilly - ISBN: 978-1-491-90143-4.[Pressman, 2009] - "Software Engineering: A Practitioner's Approach" (7th International ed.) - Roger S. Pressman - 2009 - McGraw-Hill.

[Reed, 2015] - "Devops in practice" Third Release - J. Paul Reed - 2015 - O'Reilly - ISBN: 978-1-491-91306-2.

[Smart, 2015] - “Jenkins: The definitive guide” Second Release - John Fergusson Smart - 2015 - O’Reilly - ISBN: 978-1-449-30535-2.
[Stephens et al., 2010] - “Design Driven Testing: Test Smarter, Not Harder. Program and test from the same design” - Matt Stephens & Doug Rosenberg - 2010 - Apress - ISBN: 978-1-4302-2943-8.
[Turnbull, 2015] - “The Docker Book: Containerization is the new virtualization” - James Turnbull - 2015 - Independiente.
[Wittig et al., 2016] - “Amazon Web Services in Action” - 2016 - Manning Press - ISBN: 978-1-61729-288-0.
[Zheng et al., 2015] - “Integrating Containers into Workflows: A Case Study Using Makeflow, Work Queue, and Docker” - Chao Zheng & Douglas Thain - Department of Computer Science and Engineering, University of Notre Dame - 2015 - on line al 31/03/2016 en http://ccl.cse.nd.edu/research/papers/wq-docker-vtdc15.pdf - ACM 978-1-4503-3573-7/15/06.