INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

TECNICATURA SUPERIOR EN
Ciencia de Datos e
Inteligencia Artificial

TESTEO DE SOFTWARE

Módulo Testing

Tema: Fundamentos de la calidad

Docente: Carolina Ahumada

Antes de comenzar con nuestro primer encuentro, los invito a reflexionar con el siguiente artículo:

ERRORES DE CÓDIGO DESASTROSOS DE LA HISTORIA DE LA INFORMÁTICA

Escrito por José María López, 26 de abril de 2017 a las 08:30

El software es cada vez más complejo. De ahí que habitualmente surjan bugs o errores de código. Normalmente se descubren a tiempo y se solucionan. No siempre es así. En un artículo anterior comenté que Windows 3.1 (1993) tenía entre 4 y 5 millones de líneas de código. Windows XP llegaba a los 45 millones de líneas. Sirvan esos datos como ejemplo de lo difícil que es programar y encontrar errores en el código, y eso teniendo en cuenta que los programadores cuentan con herramientas semiautomáticas.

Los errores de código son inevitables. Por suerte, siempre hay alguien que los encuentra y, en muchos casos, comparte esa información con el fabricante del software para que cree un parche o actualización que repare dicho error. Pero una cosa es un programa de ordenador que utilizan millones de personas y otra un software interno de una empresa o de un ente público. En esos casos, pocos tienen acceso a ese programa y es difícil encontrar bugs o errores de código a tiempo. En otros casos, al programar no se tienen en cuenta los posibles problemas que surgirán al cabo de muchos años. Veamos algunas de las consecuencias desastrosas relacionadas con errores de código.

Y2K O EL FIN DEL MUNDO EN 2000

Quienes vivimos el paso de 1999 al año 2000 recordamos la alarma que surgió en la prensa motivado por uno de los errores de código más curiosos.

El error Y2K o Y2K bug afectaba a computadoras creadas a principios de los 90. Para ahorrar memoria, escasa en estas primeras máquinas, los años se expresaban con dos dígitos, creyendo que no durarían más allá del siglo XX donde prácticamente todos los años empezaban por 19. Como consecuencia, el 1 de enero de 2000, esas máquinas volvían al 1 de enero de 1980 (al menos en el caso de MS-DOS y Windows).

¿Qué consecuencias tuvo el error Y2K? No se desató el Apocalipsis, tal y como parecía según ciertas informaciones alarmistas. Pero sí ocurrieron cosas, que pudieron solventarse, como alarmas o alertas en plantas de energía, modelos de telefonía móvil

antiguos que borraban los mensajes recibidos, máquinas tragaperras que dejaron de funcionar, transacciones bancarias que se rechazaron, etc.

THERAC-25 Y LA RADIACIÓN

Un error de programación del software del Therac-25, una máquina de radioterapia provocó serios accidentes entre 1985 y 1987. En concreto, un número indeterminado de pacientes fueron irradiados con dosis excesivas de radiación, más de 100 veces la dosis recomendada, lo que causó daños graves e incluso la muerte de varios de ellos.

NORAD Y LA TERCERA GUERRA MUNDIAL

NORAD es el acrónimo de North American Aerospace Defense Command o Mando Norteamericano de Defensa Aeroespacial.

Durante la Guerra Fría empleó supercomputadoras que, en algunos casos, reportaron falsas alarmas de ataques por parte de la URSS y que, de haberse tenido en cuenta, hubieran derivado en una más que probable Tercera Guerra Mundial.

En concreto, se tienen constancia de falsas alarmas en noviembre de 1979 y junio de 1980. La causa fue una batería de pruebas que reportaron falsos avisos que pudieron tomarse como ciertos de no haberse revisado.

LOS NO MUERTOS DEL ST. MARY'S MERCY MEDICAL CENTER

El hospital o centro médico St. Mary's Mercy de Grand Rapids, Michigan, actualizó su software de gestión de pacientes en 2003, lo que provocó que más de 85.000 pacientes aparecieran en su base de datos como fallecidos.

Por suerte, esa información era errónea, y es que, al actualizar el software, éste no hizo una correcta conversión del código numérico correspondiente a pacientes dados de alta asignándoles el código que correspondía a pacientes fallecidos.

Este error no afectó a vidas humanas, pero sí fue un problema para las empresas aseguradoras, los pacientes que recibieron facturas y documentación indicando que habían fallecido y otros problemas burocráticos derivados de la digitalización del sistema sanitario norteamericano.

LA CAÍDA DE LA RED DE AT&T

El 15 de enero de 1990, la mitad de la red de AT&T en Estados Unidos dejó de funcionar. Esto provocó que, en nueve horas, 75 millones de llamadas no pudieran realizarse. Si bien las primeras informaciones culparon del problema a un grupo de hackers, luego se vio que la causa era una actualización de software y un error en una línea de código de la actualización que se replicó por toda la red de AT&T.

LOS DESASTROSOS COMIENZOS DEL ARIANE 5

El cohete Ariane 5, fabricado por la Agencia Espacial Europea, tuvo unos inicios problemáticos, y es que, en su vuelo inaugural, conocido como vuelo 501, explotó a los 40 segundos de despegar y a 3.700 metros de altitud.

En el cohete no viajaba nadie, se trataba de un vuelo de prueba no tripulado. Pero sí produjo pérdidas millonarias, ya que el propósito del cohete era situar en órbita futuros satélites de comunicaciones o para experimentación científica. Se estima que unos 500 millones de dólares en carga perdida y meses de desarrollo.

¿Qué causó la explosión? Como en los casos anteriores, errores de código. Literalmente, el problema surgió al convertir un número de coma flotante de 64 bits en entero de 16 bits.

Básicamente, un error relacionado con valores decimales en las instrucciones del despegue. Curiosamente, ese código erróneo se había heredado del software de los cohetes Ariane 4, el modelo anterior cómo bien nos indica el autor, la existencia de errores en el software es algo seguro, por ende, el foco del trabajo estará en encontrarlos… lo cual, no es tarea fácil.

En la década del 60, el surgimiento de los lenguajes de alto nivel y el aumento de la potencia de las máquinas, generó una creciente demanda de sistemas de computación complejos, lo que dio lugar a los primeros grandes sistemas de software. Sin embargo la mayoría de estos fracasaron por el desarrollo de software de mala calidad. Este hecho dio lugar al nacimiento de la Ingeniería de Software, la cual estableció una serie de pautas que guían al desarrollo de software[1]… sin embargo, los problemas continuaron:

En la década de 1990, las principales corporaciones reconocieron que cada año se desperdiciaban miles de millones de dólares en software que no tenía las características ni la funcionalidad que se habían prometido.

Lo que era peor, tanto el gobierno como la industria se preocupaban por la posibilidad de que alguna falla de software pudiera afectar infraestructura importante y provocara pérdidas de decenas de miles de millones de dólares.[2]

Pero antes de profundizar más en las implicancias del software de buena o mala calidad, les propongo que reflexionemos sobre el concepto de calidad en sí, ¿Qué es calidad?

EVOLUCIÓN DE LA CALIDAD

Dentro de las organizaciones industriales, el concepto de calidad siempre estuvo presente, entendiéndose cómo: “El grado en que un producto cumple con las especificaciones técnicas que se establecieron cuando se diseñó”.

Posteriormente, el mismo evolucionó con las normas UNE 66-001, a través del cual se definió como “La adecuación al uso de un producto, el conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer las necesidades expresadas o implícitas”.

Finalmente, el concepto de calidad dejo de enfocarse en el producto, para trascender hacia todos los ámbitos de la organización, definiendo la calidad como: “Todas las formas a través de las cuales la organización satisface las expectativas de sus clientes, empleados, entidades implicadas financieramente y toda la sociedad en general”.

En paralelo con esta evolución de la definición, fue progresando también los mecanismos a través de los cuales las organizaciones gestionan la calidad:

1. En un comienzo, se aplicaba el concepto de **CONTROL DE CALIDAD**, donde un departamento era el responsable de inspeccionar cada producto y verificar que el mismo cumpla las especificaciones. El control de calidad se define como el conjunto de técnicas y actividades, de carácter operativo, utilizadas para verificar los requisitos relativos a la calidad del producto o servicio.

2. En los años 50 nace el término Quality Assurance, o **ASEGURAMIENTO DE LA CALIDAD**. Este nuevo enfoque engloba el conjunto de actividades planificadas

sistemáticas y necesarias para dar confianza de que un producto o servicio va a satisfacer los requerimientos establecidos. Actualmente, esto se conoce como **GESTIÓN DE LA CALIDAD**.

3. Finalmente, junto con la última definición de calidad que vimos, nace el término de **CALIDAD TOTAL** o Excelencia, el cual es un enfoque que engloba toda la organización. Es una estrategia de gestión cuyo objetivo es que la organización satisfaga de una manera equilibrada las necesidades y expectativas de los clientes, empleados, accionistas y sociedad en general.

¿QUÉ ES LA CALIDAD DEL SOFTWARE?

Si hablamos de la calidad del software, una de las primeras definiciones aseguraba que ***"la calidad de un programa o sistema se evaluaba de acuerdo al número de defectos por cada mil líneas de código. (KLOC: Kilo Lines Of Code)"…***

Con el avance del desarrollo y de los enfoques de calidad, la IEEE Std. 610-1990, definió a la calidad del software como ***"el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario".***

Finalmente, Pressman[1] define que la calidad de software es: ***"el proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan."***

De esta última definición podemos realizar el siguiente análisis:

1. Es un **PROCESO** eficaz de software, que establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.

   a. Los procesos de administración del proyecto generan las verificaciones y equilibrios que ayudan a evitar que el proyecto sea un caos, lo cual contribuye a la mala calidad.

[1] **Roger S. Pressman. "Ingeniería del software. Un enfoque práctico" 7ma edición. Editorial: Mc. Graw Hill.**

b. Las prácticas de ingeniería del software permiten que los desarrolladores analicen el problema y diseñen soluciones sólidas, las cuales son críticas para construir un software de alta calidad.

c. Las actividades de administración del cambio, y revisiones técnicas aportan a generar sistemas de calidad superior.

2. Genera **UN PRODUCTO ÚLTIL**, este tipo de producto es el que entrega contenido, funciones y características que el usuario final necesita, y además, lo hace de una manera confiable y libre de errores. Un producto útil será aquel que siempre satisfaga los requerimientos explícitos e implícitos (cómo por ejemplo la facilidad de uso) qué son esperados en software de alta calidad.

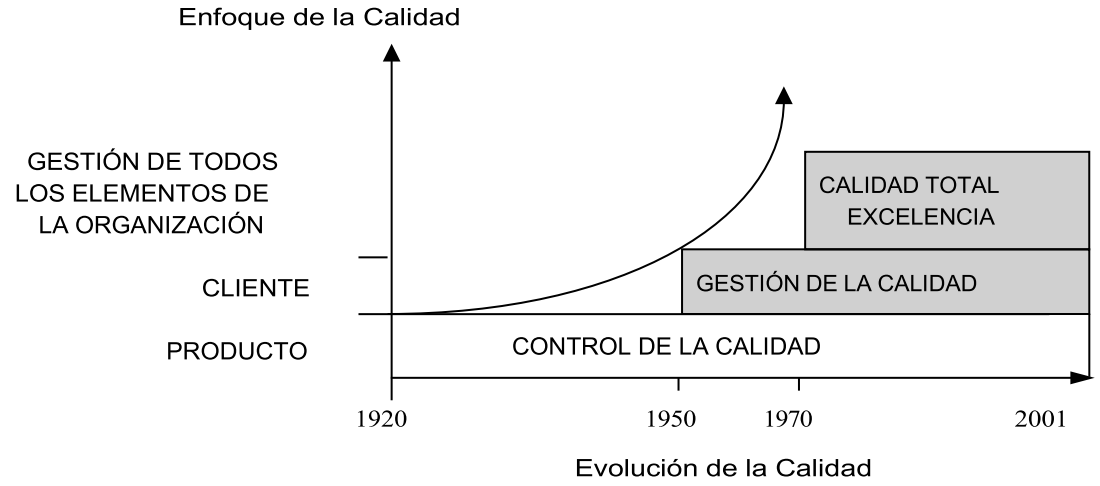
3. AGREGA VALOR para el productor y para el usuario de un producto.

a. La organización que elabora el software obtiene valor agregado porque el software de alta calidad requiere un menor esfuerzo de mantenimiento, menos errores que corregir y poca asistencia al cliente.

b. La comunidad de usuarios obtiene valor agregado porque la aplicación provee una capacidad útil en forma tal que agiliza algún proceso de negocios.

Podemos apreciar las etapas de evolución del enfoque de la calidad en el siguiente gráfico:

ETAPAS DE EVOLUCIÓN DEL ENFOQUE DE LA CALIDAD



	Años		
	20	**70**	**90**
Sistemas de producción	En masa		Ajustada

Concepto de calidad	Conformidad con especifi- caciones	Satisfacción de las nece- sidades de los clientes	Satisfacción del cliente, de los em- pleados, accionistas y sociedad

Etapas de la gestión de la calidad	Control de calidad	Gestión de calidad	Calidad Total - Excelencia

Entendiendo ahora, qué significa la calidad del software, es que nos surge la siguiente duda.. ¿Qué atributos se espera que tenga un software de calidad?...

Esta pregunta fue respondida por varios autores, los cuales elaboraron sus propias listas de factores, sin embargo nosotros nos enfocaremos en el estándar ISO 9126, el cuál fue específicamente desarrollado para identificar los atributos claves del software de computo.

Este estándar nos dice que los atributos claves de la calidad son seis:

- FUNCIONALIDAD. Grado en el que el software satisface las necesidades planteadas según las establecen los atributos siguientes: adaptabilidad, exactitud, interoperabilidad, cumplimiento y seguridad.

- CONFIABILIDAD. Cantidad de tiempo que el software se encuentra disponible para su uso, según lo indican los siguientes atributos: madurez, tolerancia a fallas y recuperación.

- USABILIDAD. Grado en el que el software es fácil de usar, según lo indican los siguientes sub-atributos: entendible, aprendible y operable.

- EFICIENCIA. Grado en el que el software emplea óptimamente los recursos del sistema, según lo indican los sub-atributos siguientes: comportamiento del tiempo y de los recursos.

- FACILIDAD DE RECIBIR MANTENIMIENTO. Facilidad con la que pueden efectuarse reparaciones al software, según lo indican los atributos que siguen: analizable, cambiable, estable, susceptible de someterse a pruebas.

- PORTABILIDAD. Facilidad con la que el software puede llevarse de un ambiente a otro según lo indican los siguientes atributos: adaptable, instalable, conformidad y sustituible.

EL COSTO DE LA CALIDAD

Llegados a esta instancia nos preguntamos ¿Es posible lograr un software perfecto? ¿Cuánto tiempo llevaría? Y si se pudiera ¿Sería Rentable?

Es normal escuchar argumentos como “Sí, la calidad es importante, pero cuesta demasiado tiempo y dinero.” Sin embargo, ante esto hay que tener en cuenta que no solo la calidad es cara, sino también la mala calidad.

La mala calidad la pagan tanto los usuarios finales que conviven con una herramienta defectuosa, así como también la organización que realizó el desarrollo y debe de darle mantenimiento.

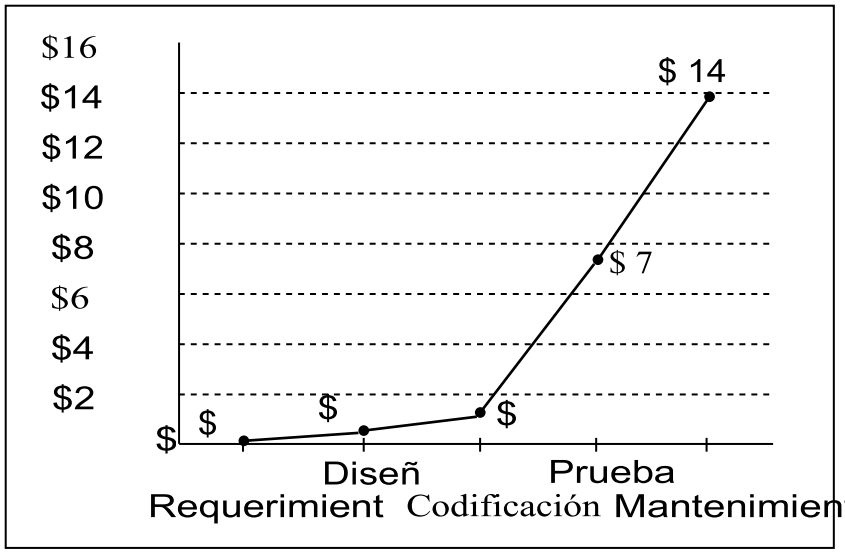
¿QUÉ ES EL COSTO DE LA CALIDAD?

El COSTO DE LA CALIDAD incluye todos los costos en los que se invierte para ir en pos de una buena calidad, sumados a los costos posteriores surgidos de la falta de calidad. Estos costos se pueden clasificar en tres:

1. **Costos de Prevención:** En el cual se incluyen los gastos de actividades de administración requeridas para planear y coordinar las actividades de control y aseguramiento de la calidad, así como los costos de planificación de pruebas y capacitación.

2. **Costos de Evaluación:** en el cual se incluyen los costos de las revisiones técnicas, de realizar las pruebas y depurar, etc...

3. **Costos de Fallas:** que representa el costo de la inversión realizada para subsanar errores



Cómo se visualiza en el gráfico anterior, el costo de corrección de errores aumenta a medida que se avanza en el proyecto, por lo cual, los costos de una mala calidad se terminarán pagando en la fase de mantenimiento, representando un gasto mucho mayor que si se hubiese implementado un proceso de gestión de calidad.

LA CALIDAD NO ES NEGOCIABLE

Distingamos entre calidad interna y calidad externa:

- Calidad externa: es lo que perciben los usuarios del sistema. Un interfaz de usuario lento y poco intuitivo es un ejemplo de baja calidad externa.

- Calidad interna: se refiere a aquellos aspectos que normalmente no son visibles al usuario, pero que tienen un profundo efecto en la mantenibilidad del sistema. Cosas como consistencia del diseño del sistema, cobertura de pruebas, legibilidad del código, refactorización, etc.

Generalizando, un sistema con alta calidad interna puede, aun así, tener una baja calidad externa. Pero un sistema con baja calidad interna rara vez tendrá buena calidad externa. Es difícil construir algo sobre unos cimientos podridos.

Aquí trataremos a la calidad externa como parte del alcance. En algunos casos puede tener sentido, desde el punto de vista de negocio, liberar una versión del producto que tenga un internaza de usuario torpe y lento, y más tarde liberar una versión mejorada. Dejo esa decisión al Dueño de Producto, ya que él es el responsable de definir el alcance.

Sin embargo, la calidad interna es algo que no puede ser discutido. Es responsabilidad del equipo mantener la calidad del sistema bajo toda circunstancia y simplemente no es negociable.

¿Y cómo distinguimos la diferencia entre aspectos de calidad externa y aspectos de calidad interna?

Supongamos que el PO dice “Respeto a las estimaciones de esfuerzo de 6 puntos de historia, estoy seguro de que podríamos usar algún ***parche rápido*** para hacerlo en la mitad de tiempo”.

Está intentando usar la calidad interna como una variable. ¿Cómo lo sé? Porque quiere que reduzcamos la estimación de la historia sin “pagar el precio” de reducir el alcance. La expresión “parche rápido” debería disparar una alarma en sus mentes…

¿Y por qué no permitimos esto? Es que sacrificar la calidad interna es, prácticamente siempre, una idea terrible. El tiempo que se ahorra es mucho menor que el coste, tanto a

corto como a largo plazo. Una vez que permites que una base de código comience a deteriorarse es muy duro volver a conseguir su calidad original más adelante.

En lugar de eso intento reconducir la discusión hacia el alcance. “Ya que es importante para ti que entreguemos esta historia pronto, ¿podemos reducir su alcance de forma que sea más rápida de implementar? Quizás podríamos simplificar el manejo de errores y convertir “Manejo Avanzado de Errores” en una historia separada que reservemos para el futuro. O podríamos reducir la prioridad de otras historias de forma que podamos centrarnos en esta”.

Una vez que el Dueño de Producto aprende que la calidad interna no es negociable, normalmente se hace muy bueno en manipular las otras variables.

¿CÓMO LOGRAR LA CALIDAD DE SOFTWARE?

Para poder influir en la calidad del software de manera positiva, Pressman nos recomienda llevar a cabo una buena administración del proyecto y una correcta aplicación de la ingeniería del software. Para ello recomienda ejecutar las siguientes cuatro actividades:

- Utilizar **MÉTODOS DE LA INGENIERÍA DEL SOFTWARE** para el desarrollo de un acertado análisis y diseño del sistema.

- Utilizar **TÉCNICAS DE ADMINISTRACIÓN DE PROYECTOS**, de manera que se utilicen las estimaciones para verificar el cumplimiento de fechas.

- Se comprendan las dependencias de las actividades programadas.

- Se lleve una correcta gestión de los riesgos para evitar caos.

- Implementar **CONTROL DE CALIDAD**, el cual incluye el conjunto de acciones que aseguran que todo el trabajo cumpla sus metas de calidad. Entre estas actividades se incluye la revisión de modelos, inspección de código, implementación de pruebas y seguimiento de métricas de calidad.

¿QUÉ ES SQA?

La IEEE define al aseguramiento de la calidad como: “Una guía planificada y sistemática de todas las acciones necesarias para proveer la evidencia adecuada de que un producto cumple los requerimientos técnicos establecidos. Un conjunto de actividades diseñadas para evaluar el proceso por el cual un producto es desarrollado o construido.”

Por otro lado la norma ISO 9000:2000 define SQA cómo la parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de calidad.

El Aseguramiento de la Calidad del Software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza que el software satisfará los requisitos dados de calidad. Este aseguramiento se diseña para cada aplicación antes de comenzar a desarrollarla y no después. El Aseguramiento de la Calidad del Software engloba:

- Un enfoque de gestión de calidad.
- Métodos y herramientas de Ingeniería del Software.
- Revisiones técnicas formales en el proceso del software.
- Una estrategia de prueba multi-escala.
- El control de la documentación del software y de los cambios realizados.
- Procedimientos para ajustarse a los estándares de desarrollo del software.
- Mecanismos de medición y de generación de informes.

Algo importante a tener en cuenta en este punto, es que SQA no es lo mismo que el control de calidad. El público general al hablar de SQA lo relacionan directamente con el testing, revisiones, etc… Estos elementos son solo una parte del área del control de la calidad del software.

Sin embargo con solo realizar testing no podemos asegurar que nuestro producto sea de calidad. El aseguramiento de calidad (QA, por las siglas de quality assurance) es la definición de procesos y estándares que deben conducir a la obtención de productos de alta calidad y, en el proceso de fabricación, a la introducción de procesos de calidad. El

control de calidad es la aplicación de dichos procesos de calidad para eliminar aquellos productos que no cuentan con el nivel requerido de calidad. En síntesis…

Control de Calidad
· Busca detectar problemas en los productos que se generan de los trabajos. · Verifica que los productos cumplan con los estandares de calidad. · Revisa el Producto y su contenido.
Aseguramiento de la Calidad (SQA)
· Asegura que los procesos se adhieran a los estándares y planes definidos. · Evalúa que los procesos, planes y estándares esten alineados a los estándares organizacionales. · Revisa el proceso.

A pensar y reflexionar!

A continuación, reunidos en grupos, los invito a responder el siguiente cuestionario respecto a los temas de la clase. Saquen sus propias conclusiones!! No lleva calificación, pero debes subirlo en el espacio de Testeo de software “Evidencia – Clase 1” para validar parte de tu aprendizaje.

1. ¿Qué entiendes por Calidad?

2. ¿Qué es la calidad de software?

3. ¿Cuáles son los costos de la calidad?

4. ¿Qué es el aseguramiento de la calidad de Software?

5. ¿Cuáles son las Diferencias entre el SQA y el Control de Calidad?

6. ¿Cuál es la participación que deben tener los miembros de un equipo de desarrollo de software, en definiciones de calidad de una entrega?

7. ¿Cómo resolverías el pedido del dueño del producto para adelantar la fecha de entrega de un requerimiento? ¿Con qué argumentos?

REFERENCIAS BIBLIOGRÁFICAS

Pressman, R. S. (2010). *Ingeniería del software: Un enfoque práctico* (7.ª ed.). McGraw-Hill.

Toledo, F. (2016). *Introducción a las pruebas de sistemas de información*. Universidad ORT Uruguay.