ACTUALIZACIÓN INFORMÁTICA

CPAP

CURSO DE POSGRADO

Técnicas y Gestión de las Pruebas de Software

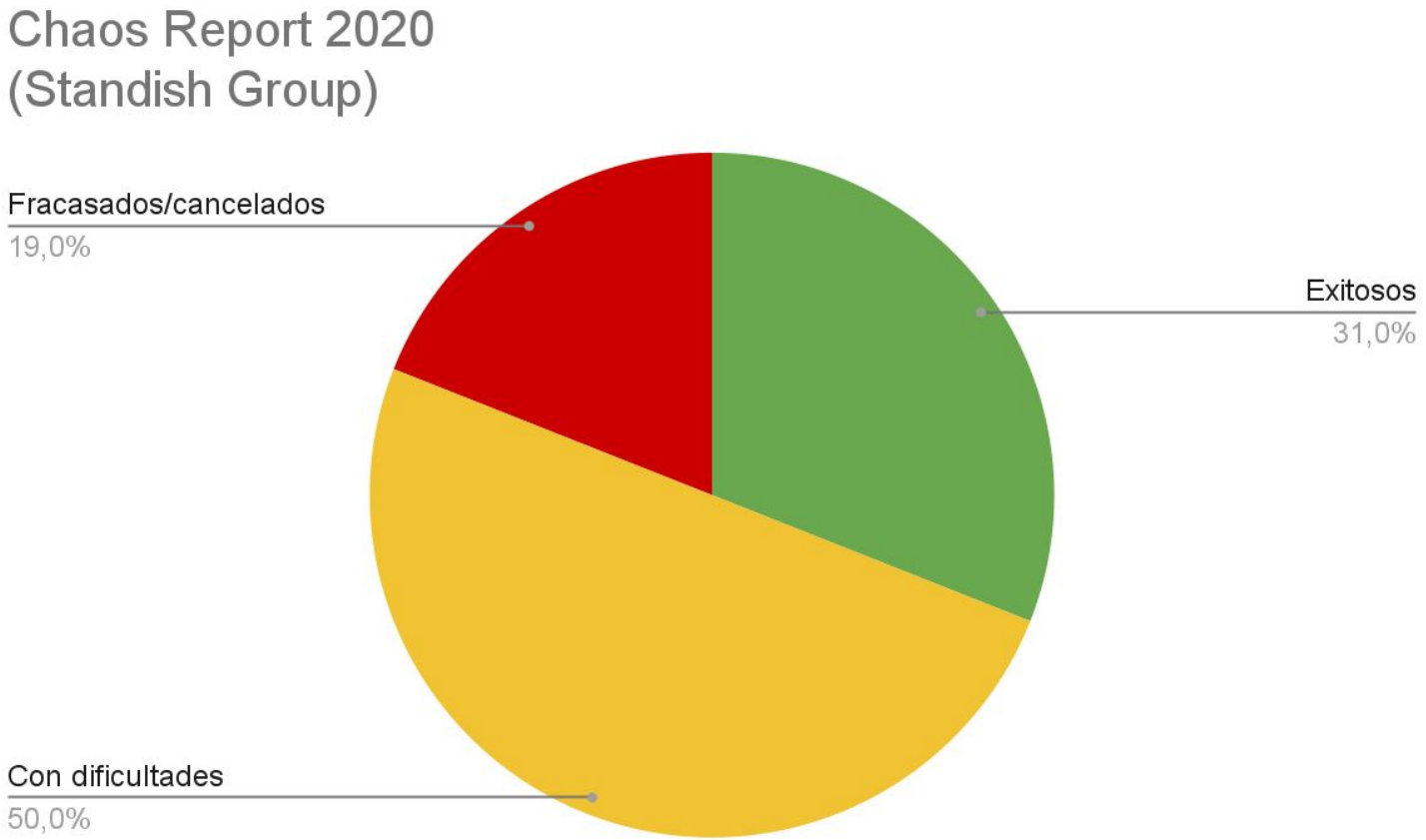
Cecilia Apa

DOCENTE

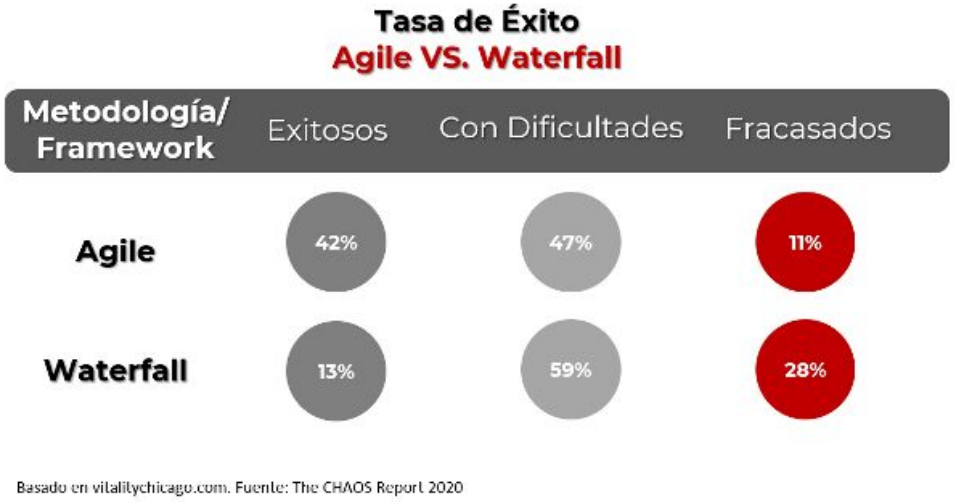
ACTUALIZACIÓN INFORMÁTICA
CPAP

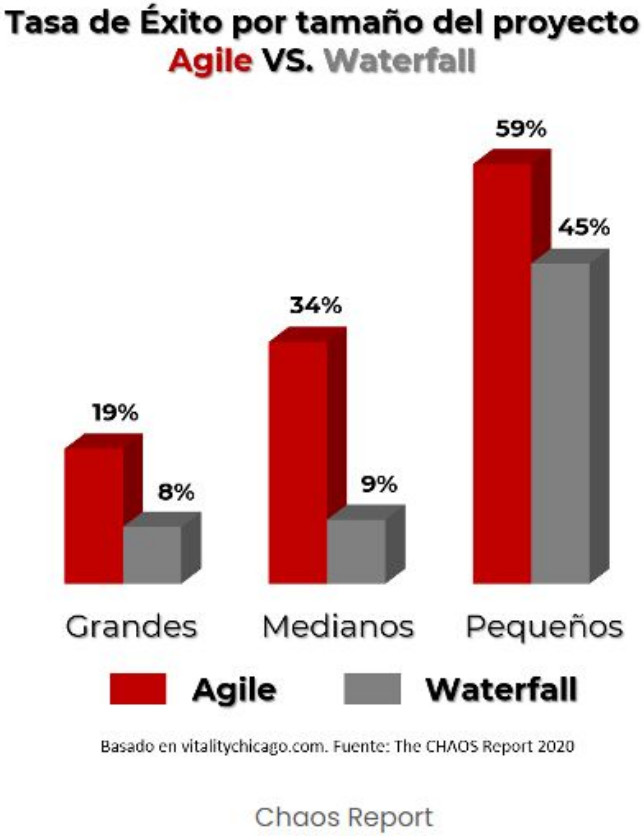
Motivación

Desarrollo de software - actividad de alto riesgo



The Chaos Report: Agile y el éxito en los proyectos de TI[1]

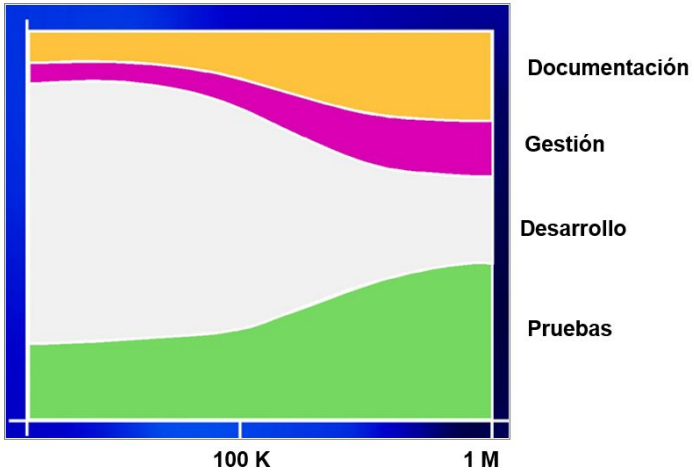




[1] https://scrumcolombia.org/the-chaos-report-agile-y-el-exito-en-los-proyectos-de-ti/

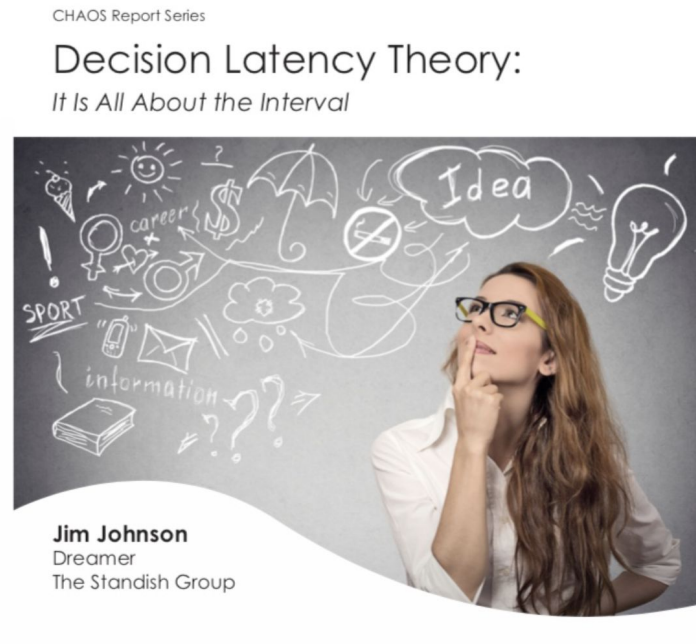
Chaos report 2015

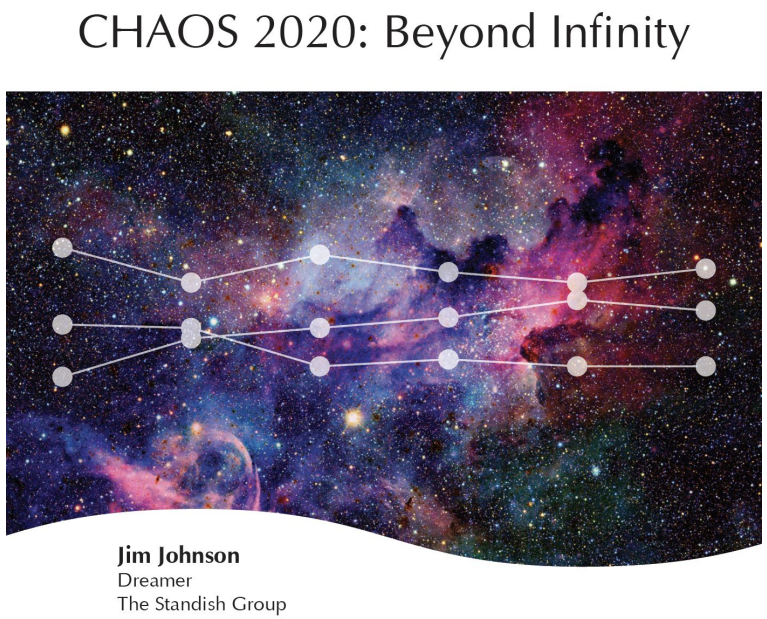
- Una de principales causas de fracaso en los proyectos es el bajo nivel de pruebas
- Esta dificultad es tanto más grave cuando el proyecto es grande
  - La complejidad crece exponencialmente al tamaño del proyecto
  - La parte de las pruebas es proporcionalmente mayor en proyectos complejos



Google: “Chaos report” + “testing”

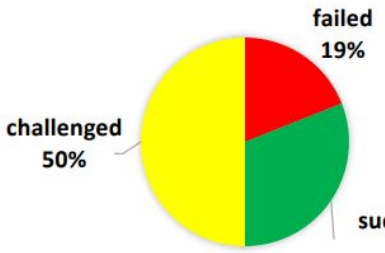
Nos quedamos sin compatibilidad hacia atrás…



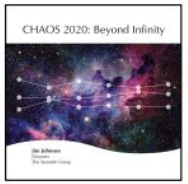


Project Success
Quick Reference Card

Based on CHAOS 2020: Beyond Infinity Overview. January'2021, QRC by Henny Portman



Modern measurement (software projects)



Good Sponsor, Good Team, and Good Place are the only things we need to improve and build on to improve project performance.

The Good Place is where the sponsor and team work to create the product. It's made up of the people who support both sponsor and team. These people can be helpful or destructive. It's imperative that the organization work to improve their skills if a project is to succeed. This area is the hardest to mitigate, since each project is touched by so many people. Principles for a Good Place are:

- The Decision Latency Principle
- The Emotional Maturity Principle
- The Communication Principle
- The User Involvement Principle
- The Five Deadly Sins Principle
- The Negotiation Principle
- The Competency Principle
- The Optimization Principle
- The Rapid Execution Principle
- The Enterprise Architecture Principle

Successful project Resolution by Good Place Maturity Level:

highly mature	50%
mature	34%
moderately mature	23%
not mature	23%

The Good Team is the project's workhorse. They do the heavy lifting. The sponsor breathes life into the project, but the team takes that breath and uses it to create a viable product that the organization can use and from which it derives value. Since we recommend small teams, this is the second easiest area to improve. Principles for a Good Team are:

- The Influential Principle
- The Mindfulness Principle
- The Five Deadly Sins Principle
- The Problem-Solver Principle
- The Communication Principle
- The Acceptance Principle
- The Respectfulness Principle
- The Confrontationist Principle
- The Civility Principle
- The Driven Principle

Successful project Resolution by Good Team Maturity Level:

highly mature	66%
mature	46%
moderately mature	21%
not mature	1%

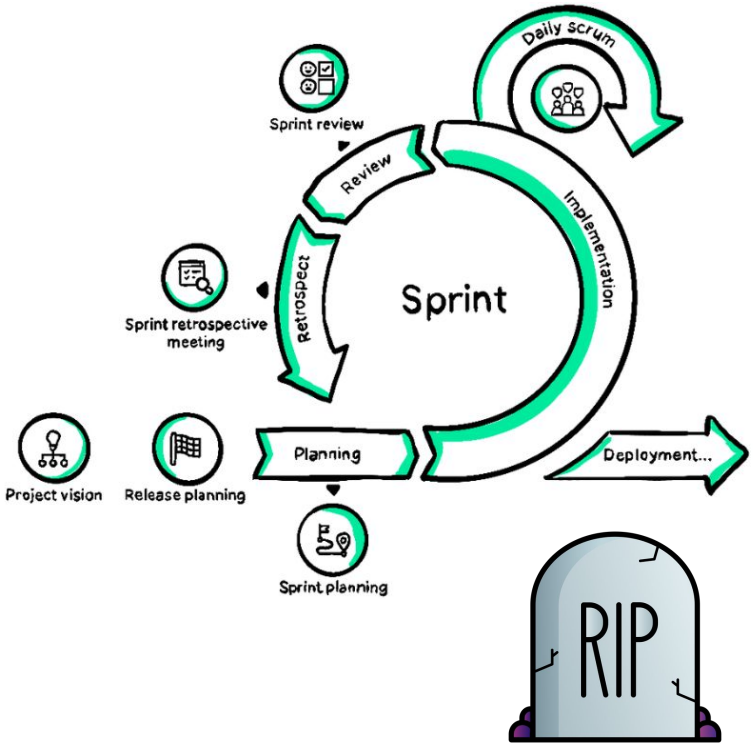
The Good Sponsor is the soul of the project. The sponsor breathes life into a project, and without the sponsor there is no project. Improving the skills of the project sponsor is the number-one factor of success – and also the easiest to improve upon, since each project has only one. Principles for a Good Sponsor are:

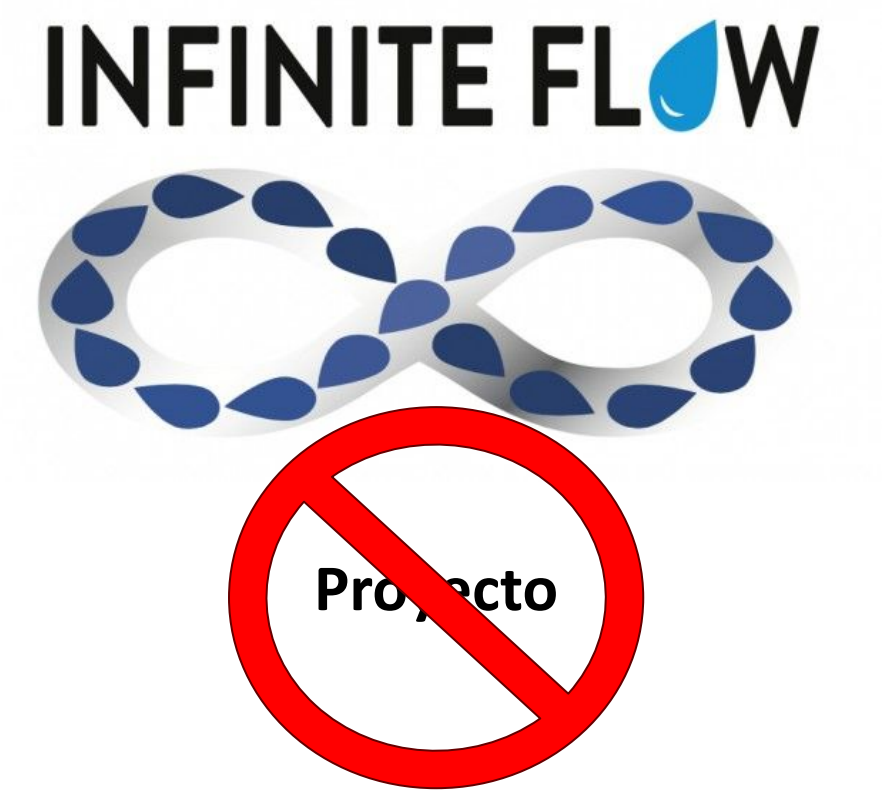
- The Decision Latency principle
- The Vision Principle
- The Work Smart Principle
- The Daydream Principle
- The Influence Principle
- The Passionate Principle
- The People Principle
- The Tension Principle
- The Torque Principle
- The Progress Principle

Successful project Resolution by Good Sponsor Maturity Level:

highly mature	67%
mature	33%
moderately mature	21%
not mature	18%

Predicciones CHAOS Report 2020





[1] https://scrumcolombia.org/the-chaos-report-agile-y-el-exito-en-los-proyectos-de-ti/

Períodos de evolución del desarrollo de software (60 años hacia atrás y 20 hacia adelante)

- 1960 - 1980 *Wild West*
- 1980 - 2000 *Waterfall*
- 2000 - 2020? *Agile*
- 2020 - 2040? *Infinite Flow Period*
  - Sin presupuesto de proyecto, planes de proyecto, PM's o Scrum Masters.
  - Habrá un presupuesto para el *pipeline*, y para gestionar el *pipeline*.
  - Esto se logrará reduciendo y eliminando la mayoría de la mayoría de las actividades de gestión de proyecto actuales.
  - La descripción funcional del trabajo a realizar entrará en el *pipeline* y saldrá en forma de software listo para usar.
  - El cambio ocurrirá continuamente, pero en pequeños incrementos que mantendrán todo actualizado, útil y más aceptable para los usuarios, en lugar de sorprenderlos con un resultado de "gran explosión".[1]

[1] https://hennyportman.wordpress.com/2021/01/06/review-standish-group-chaos-2020-beyond-infinity/

Tendencias en las pruebas

- La complejidad de los “proyectos” es creciente
  - Software destinado a contextos de altas exigencias.
  - Fuerte integración entre sistemas (nuevos, legados, externos)
    - ¿Dónde está la complejidad?
  - Sistemas incorporados a equipamiento: automóvil, audiovisual, electrónica, telefonía, transporte público, armamento, centrales nucleares, petróleo,…
  - Aplicaciones móviles con altas exigencias de calidad de uso y dinamismo

- Las exigencias son enormes y globales
  - Muy alta adecuación funcional a las necesidades del usuario
  - Muy altas exigencias de fiabilidad, usabilidad, seguridad, integridad

Tendencias en las pruebas

- Las pruebas, en sentido amplio, deben formar parte del proceso de desarrollo.
  - Es mucho más difícil hacer las pruebas únicamente a posteriori del desarrollo.
  - Pruebas Ágiles: fuerte integración entre desarrollo, involucramiento de usuarios, pruebas.
  - Más aún pensando en un pipeline de flujo continuo

- Acciones multidisciplinarias, multidimensionales
  - Mejorar los procesos de desarrollo de software
    - Mejores prácticas, mejora continua, normalización
  - Mejorar las herramientas de desarrollo
    - Prevenir, detectar, facilitar la corrección de errores
  - Mejorar los procesos de Prueba de Software
    - Todo se juega durante el proceso de desarrollo
    - Es muy difícil agregar calidad a posteriori
    - Difundir las buenas prácticas e impulsar la certificación de profesionales

¿Qué dificultades enfrentan las pruebas?

Dificultades de las pruebas

- Visibilizar el valor agregado de las pruebas
  - Lo importante es la creación de productos innovadores, la algorítmica, la adecuación al negocio…
  - Probar genera un costo adicional cuyo aporte es poco visible
  - Solo se aprecia cuando previene problemas importantes

Me recuerda a…

Dificultades en las pruebas

- Por naturaleza, las pruebas nunca son completas ni exhaustivas.
  - Imposibilidad práctica (y generalmente también teórica) de ejercer todos los casos de prueba posibles.
- Las pruebas llegan generalmente demasiado tarde
  - Al final del proyecto (liberación, sprint, etc), con poco tiempo y pocos recursos
  - Sin capacidad de intervenir en decisiones tempranas, reduciendo su potencial de mejora de la calidad

Evolución de la disciplina

- El área de Pruebas de Software está ocupando un lugar creciente
  - Como lo ha sido el área de Procesos, con ISO 9000 y CMM-I
  - Como lo ha sido el área de Gestión de Proyectos, con PMI
  - Nueva norma ISO 29119
  - Metodologías Agiles para las Pruebas de Software.
  - Pruebas de aplicaciones para celulares

- Una de las claves es ver las pruebas en su contexto global, como aporte a la mejora de los procesos de desarrollo y a la calidad de los productos

Comunidades y Conferencias en Uruguay

‹TestingUy›

https://testinguy.org/

https://www.linkedin.com/company/under-test/

https://qualitysenseconf.com/

Otras conferencias en el mundo

- Conference of the Associations for Software Testing (CAST) (Canada)
- Agile Testing Days (Potsdam, Germany)
- Test India (Dehli)
- EuroSTAR (Software Testing Analysis and Review) (La Haya)
- Software Testing and Validation (Paris)
- BZ Media Software Test and Performance Conference (Boston)
- Future Test (New York)
- ICSTEST-E Conference of Software Testing (Bilbao)
- QA & Test (Bilbao)
- QAI Canada Internation Quality Conference
- QUEST – Quality Engineered Software and Testing Conference
- STANZ Conference (Software Testing in Australia & New Zeland)
- Software Testing Managers Workshop
- The Workshop on Performance and Reliability

Certificaciones en Testing

ISTQB® International Software Testing Qualifications Board

www.istqb.org

SSTQB Spanish Software Testing Qualifications Board

www.sstqb.es

iSQI® International Software Quality Institute

https://corporate.isqi.org/

GASQ

www.gasq.org

ACTUALIZACIÓN INFORMÁTICA
CPAP

Introducción

Temas a tratar hoy

- Visión general de las pruebas de software
- Introducir terminología
- Presentar los grandes principios básicos de las pruebas
- Introducir la noción de procesos fundamentales de las pruebas

Aseguramiento de la calidad y pruebas

Aseguramiento de la Calidad

- Un modelo de acciones planeadas y sistemáticas necesarias para alcanzar un adecuado grado de confianza tal que un elemento o producto es conforme con los requerimientos técnicos establecidos.

Pruebas

- El proceso de operar un sistema o componente bajo condiciones especificadas, observando y registrando los resultados, evaluando ciertos aspectos de ese sistema o componente.

Norma IEEE 610 – Standard glossary of software engineering terminology

¿Qué es la calidad?

"La calidad es valor para una persona a la que le interesa"
Cem Kaner[1]

[1] Extraído del libro: *Introducción a las Pruebas de Sistemas de Información, Federico Toledo*

Pruebas de software

- Permite la **medida** de la calidad del software basada en la cantidad de defectos
  - Medidas de características de aspectos funcionales y no funcionales
- Proporciona **confianza** en la calidad del software
  - Por evaluación y por corrección de errores encontrados
- Ayuda a reducir los **riesgos**
  - Pruebas adecuadas permiten detectar errores tempranamente
- Impacta positivamente el **aseguramiento de la calidad**
  - Lecciones aprendidas ayuda a mejorar la calidad de los procesos para próximos proyectos

¿Por qué las pruebas son necesarias?

- Para reducir los riesgos operacionales
  - Financieros, reputación, impactos graves, daños físicos
- Para aumentar la calidad del software
  - Costos de prevención mucho más baratos que los correctivos
- Para cumplir con requerimientos contractuales o legales
  - Contratos y reglamentaciones públicas
- Para cumplir con estándares
  - Políticas de privacidad, procedimientos de seguridad, requerimientos de seguridad de funcionamiento

¿Cuánto hay que probar?

- Es importante la relación costo-beneficio
- Las pruebas debe permitir que los responsables tomen decisiones fundamentadas
  - Riesgos y restricciones confrontados a la entrega del software en el estado actual
- Riesgos
  - Técnicos, del negocio, del producto, del proyecto
- Restricciones
  - Presupuesto, plazos

Probar vs. depurar (testing vs. debbugging)

Probar

- Determinar si un elemento bajo pruebas contiene fallas.
- Medir o estimar la calidad de un elemento/componente.
- Tomar decisiones binarias, SI/NO, hay presencia de defectos.
- Puede ser realizado por el programador, probador o usuario.

Depurar

- Buscar la causa de las fallas encontradas.
- Identificar el defecto en el código.
- Análisis de soluciones posibles, y realización de las correcciones necesarias.
- Realizado por el programador (generalmente el que hizo ese componente).

Más que pruebas

- Muchas actividades de prueba en diferentes momentos
  - Actividades previas
    - Planificación, estrategia, diseño de casos de prueba
  - Actividades durante
    - Controlar resultados, evaluar criterios de salida, informar al cliente
  - Actividades posteriores
    - Generación de reportes, lecciones aprendidas, finalización de pruebas

- El proceso incluye pruebas estáticas y revisiones
  - Revisión de requerimientos, diseños, códigos
  - Verificación de normas y convenciones
  - Evaluación de factores no-funcionales
  - Detección (manual o automática) de ciertos errores

Objetivos de las pruebas desde diferentes puntos de vista

Punto de vista	Objetivo
Dentro del proceso de desarrollo (pruebas unitarias, de integración y de sistema)	Identificar y corregir la mayor cantidad posible de defectos. Decidir si el software puede ser entregado.
Prueba de recepción	Verificar conformidad con los requerimientos
Prueba de mantenimiento (regresión)	Verificar que no se han agregado defectos o efectos secundarios como consecuencia de cambios en el software
Prueba operacional	Verificación de la confiabilidad, disponibilidad y otras características
Pruebas alfa y beta	Evaluación de pertinencia y calidad del software previo a una entrega o difusión

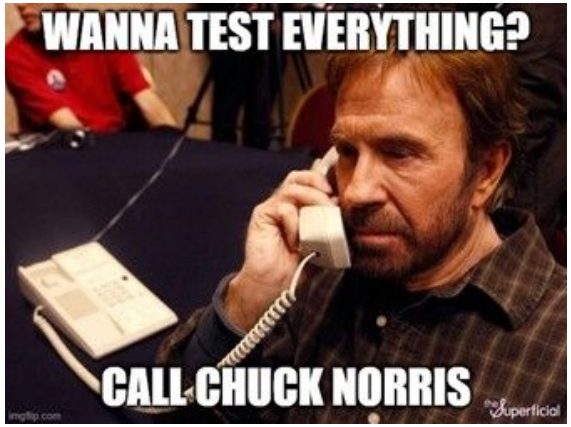
Siete principios básicos de las pruebas (1)

Principio 1: las pruebas ponen en evidencia defectos

- El testing muestra la existencia de errores, no su ausencia.
- Reduce la probabilidad de defectos no detectados
- No se puede demostrar que todos los defectos han sido identificados.

Principio 2: las pruebas exhaustivas son imposibles

- Probar “todo” no es factible, salvo en casos muy triviales.
- Evaluar los casos más interesantes que reduzcan el riesgo y aumenten la confianza.



Siete principios básicos de las pruebas (2)

Principio 3: Probar en fases tempranas

- Comenzar al inicio del desarrollo del software
- Realizar revisiones de los requerimientos, diseño y código
- La corrección de defectos en los requerimientos pueden costar más de 100 veces si son detectados en la implantación

"Hacer las pruebas al final de un proyecto es como pedir examen de sangre cuando el paciente ya está muerto."
Scott Barber[1]

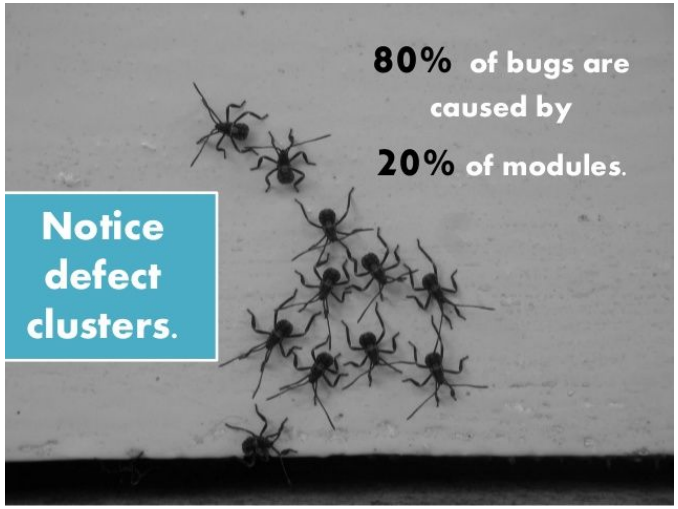
Video ilustrativo:
https://www.youtube.com/watch?v=75wa8Lx4yc4&ab_channel=AplanaSoftware

[1] Extraído del libro: *Introducción a las Pruebas de Sistemas de Información, Federico Toledo*

Siete principios básicos de las pruebas (4)

Principio 4: Agrupamiento de los defectos (defect clustering)

- La mayoría de las fallas están originadas en unos pocos módulos
- Realizar un análisis del origen de las fallas a partir de pruebas en fases tempranas
- Repetir esas pruebas hacia el final del proceso



Siete principios básicos de las pruebas (3)

Principio 5: Paradoja "del pesticida"

- Repetir el mismo tipo de prueba puede perder eficacia en el largo plazo
- Revisar periódicamente para evitar este fenómeno

Principio 6: Las pruebas son dependiente del contexto

- Sistemas similares (en apariencia) pueden tener objetivos muy diferentes y requisitos muy diferentes
- Definir los objetivos de las pruebas según el contexto
- Adecuar esos objetivos según el nivel de riesgo aceptable

Principio 7: Falacia de la ausencia de errores

- El objetivo es que el software sea conforme a las especificaciones y no contenga errores.
- No se puede demostrar que así sea, pero igual hay que usar el software de todas formas.

La psicología de las pruebas (1)

- Los desarrolladores sienten que son los más idóneos para encontrar defectos en su propio programa
  - Aunque generalmente no poseen la objetividad y la independencia necesaria
  - Tienden a cometer los mismos errores al probarlo
  - Es preferible utilizar a personas independientes y objetivas
  - Fomentar la colaboración entre esos dos grupos, en lugar de la competencia o la oposición

La psicología de las pruebas (2)

- Buscar defectos puede ser percibido como una agresión hacia el producto (y hacia quien lo construyó)
  - Comunicar claramente los objetivos de las pruebas a todo el equipo de desarrollo, de forma a evitar confusiones
  - Presentar los resultados obtenidos de forma constructiva, en lugar de mostrarlos como críticas destructivas
  - Evitar criticar a los responsables del proyecto, arquitectos o a los programadores

- Desarrolladores y testers se ven como adversarios en lugar de como colaboradores
  - Insistir en que ambos grupos están “en el mismo barco”
  - Recordar que el objetivo común es construir un software conforme a los requisitos y mejorar la calidad del producto y de los procesos
  - Concentrarse en las observaciones y resultados objetivos, dejando de lado las opiniones subjetivas
  - Tener siempre presente la forma en la cual el otro va a recibir las malas noticias

Niveles de independencia

Nivel 0
- Pruebas realizadas por el desarrollador

Nivel 1
- Pruebas realizadas por otro desarrollador

Nivel 2
- Pruebas realizadas por testers del mismo proyecto

Nivel 3
- Pruebas realizadas por terceras partes (externas al proyecto pero en la misma organización)

Nivel 4
- Pruebas realizadas por una organización externa

Principales roles y responsabilidades

- Jefe de Proyecto de Pruebas (Test leader): gestión integral del proyecto
- Analista de Pruebas (Test Analyst): qué elementos probar y con qué objetivos
- Diseñador de Pruebas (Test Designer): qué técnicas utilizar y casos de prueba
- Ejecutante de Pruebas (Tester): ejecución y registro de resultados
- Apoyo técnico al equipo de pruebas: construir e instalar ambientes, configuración de herramientas

Estos roles toman formas y lugares diferentes según la metodología de desarrollo y la organización de los equipos.

Algunas conclusiones

- Reconocer importancia de las pruebas dentro del proceso de desarrollo
- Existen principios básicos a tener en cuenta, independientemente del proyecto particular
- La actitud de los grupos dentro del equipo de desarrollo es diferente con relación a la tarea de Pruebas
- Importancia de las pruebas independientes