|  |  |
| --- | --- |
| **Carrera/s** | Ingeniería en Informática |
| **Sigla Curso** | AIN6501 |
| **Modalidad** | Presencial |
| **Versión PDA** | 2015 |
| **Material de apoyo (insumos y equipamiento) para la actividad** | PPT: Testing de Software |

|  |
| --- |
| **NOMBRE DE LA ACTIVIDAD:**  **Casos** |
| **Objetivos de la actividad:**  Analizar un caso de una empresa logrando:   * Detectar buenas prácticas * Detectar de manera correcta las malas prácticas de la empresa. * Definir un modelo de buenas prácticas. * Responder de manera analítica las preguntas planteadas.   **Instrucciones**   * Trabajo para desarrollar en parejas. * Lea atentamente el caso planteado y luego responda las preguntas planteadas.   **Caso : STARSUPS, Perdidas de proyectos de S.W.**  *Hay emprendedores perdiendo su proyecto por usar malas prácticas software*.  Emprender en proyectos basados en la Web está de moda. Sintetizando mucho, detrás de este tipo de emprendimiento, y de estas startups, hay una idea de negocio, un equipo humano promotor y un desarrollo software.  Aunque estos proyectos de emprendimiento están basados en construir funcionalidad sobre tecnología, es obvio decir que no todos los emprendedores que lanzan estas startups tienen conocimientos sólidos y gran experiencia (kilómetros y cicatrices) en desarrollo software. En unas ocasiones porque no es lo suyo, y su perfil es, por ejemplo, económico, y en otras porque tienen demasiada poca experiencia en desarrollo software, guerras similares y supervivencia a naufragios.  La historia comienza en una empresa, con un proyecto software que empieza a tener éxito. Como al principio hay poco software desarrollado nada parece indicar que el propio software sea un riesgo.  Llegan los primeros clientes, y demandan nuevas funcionalidades, nuevas versiones, que deben estar en producción rápidamente. Y así se hace. Pero a nadie parece preocuparle la calidad con que se está desarrollando el software, el negocio es lo primero… y lo único.  Pero el tiempo pasa rápidamente, y cada vez hay más software desarrollado sin control y más gente desarrollándolo. Así que, poco a poco, casi sin darnos cuenta, aquellas funcionalidades que antes rápidamente se desarrollaban, cada vez requieren de más tiempo. Pero cliente, negocio e inversores no están dispuestos a darnos ese tiempo.  Y comienza entonces la fase de las soluciones desesperadas. Comienza la presión, y las jornadas de trabajo nocturnas y fines de semana. Crece el estrés. La gente que no aguanta y abandona el equipo, llevándose el conocimiento de cómo se desarrollaron ciertas partes del software.  Para intentar solucionar el problema se incorporan nuevos desarrolladores. Nuevos desarrolladores que al no conocer el sistema quitan tiempo a los más veteranos. Todo esto acaba bloqueando la producción de software, la puesta en producción de nuevas funcionalidades, haciendo lenta a la joven empresa y disparando el gasto.  Lo anterior ni es nuevo, ni mucho menos es exclusivo de las startups. Un gran número de empresas más tradicionales que desarrollan software viven esto cada día, y desde hace muchos años. Pero la mayoría sobrevive, o mal vive, durante años porque son empresas grandes, algunas con clientes cautivos, con otras fuentes de ingresos, o que se mueven en negocios que van más lentos. Y pueden permitirse ese gran desperdicio económico.  Pero las startups no se pueden permitir ni desperdicios económicos ni frenar la velocidad en que ponen en producción nuevas funcionalidades. Y además… ¿Cómo le vas a explicar a los inversores esta historia sobre por qué el desarrollo de negocio ahora es tan lento y que intentar mal acelerarlo es peor?  **Análisis de caso y diseño de setup de buenas prácticas:**  Análisis de caso y diseño metodológico básico de un setup de buenas prácticas de calidad para el ciclo de vida de desarrollo de software.   1. Analice y determine al menos 10 buenas prácticas que llevó a la empresa a tener éxito en algún momento. Justifique su respuesta explicando cada una de las buenas prácticas. 2. Determine cuáles fueron las malas prácticas realizadas por los gerentes de la empresa. Genere un listado con 10 malas prácticas. Justifique su respuesta explicando porque considera que estas son malas prácticas y indicando cuál es la consecuencia de cada una. 3. Genere un setup de buenas prácticas que permitan mejorar esta empresa. Indicando para cada etapa del desarrollo software buenas prácticas y justificando cada una.    1. Indique 5 buenas prácticas de Calidad asociadas a una correcta toma de requerimientos informáticos.    2. Indique 5 buenas prácticas de Calidad asociadas a la etapa de Desarrollo del Software.    3. Indique 5 Buenas Prácticas de Calidad asociadas a la etapa de Testing de Software.    4. Indique 5 buenas prácticas de Calidad asociadas la implementación de un producto de software. |

Desarrollo

1R: -Contratar a nuevos desarrolladores

-Cumplir con el desarrollo del software en el tiempo acordado

-Buena comunicación con el cliente

2R: 1-No comentar el código: ya que al llegar un nuevo programador este no sabrá las funcionalidades de cada método lo que retrasaría el desarrollo

2-No utilizar patrones de diseño: porque los patrones de diseño permiten estandarizar el lenguaje de los programadores y ofrecer soluciones a problemas comunes ya resueltos, lo que permite ahorrar mucho tiempo de desarrollo

3-No establecer un ciclo de vida del software para detectar errores rápidamente

4-Desarrollar código de manera intuitiva: esto a larga nos puede generar errores en el código que no seamos capaces de controlar o saber la razón exacta del fallo del programa

5-No introducir a gente que domine el negocio: introducir a gente que domine el negocio en etapas tempranas del desarrollo puede ayudar a tomar una correcta toma de requerimientos

6-No tener un plan de contingencia en caso de que un empleado renuncie y se enferme aumentando el tiempo de desarrollo

7-No realizar estimaciones razonables: por falta de una buena estimación de tiempo aumento la presión y el estrés del equipo lo que hizo que muchos desarrolladores optaran por abandonar el proyecto

8- No incluir el testing en todas las fases de desarrollo, si incluimos el testing en todas las fases podemos disminuir en un gran porcentaje la cantidad de errores a la hora del despliegue del software

9-Llevar una bitácora con los testing realizados para que al encargarle esa tarea a otro empleado él ya esté al tanto de las funcionalidades que ya han sido probadas

10-

3R:-Toma de Requerimientos:

1. Conocimiento completo del proceso
2. Realizar una buena planificación: establecer reuniones para capturar las necesidades de los clientes
3. Asesorar al cliente: para no gastar tiempo y recursos innecesarios en procesos que pudieron haber sido más simples
4. Utilizar herramientas adecuadas para capturar requerimientos: para refinar los requerimientos de una manera eficaz y acercarse de a poco a las necesidades deseadas
5. Tratar de descomponer requerimientos en lo más pequeño que se pueda ya que así será más fácil realizar una estimación y desarrollo de estos

-Desarrollo del Software

1. Retroalimentación constante: constantemente revisar el software en busca de posibles mejoras

2. Documentar Continuamente: en caso de que un desarrollador se enferme o renuncie, la persona que lo reemplace le será más fácil entender las funcionalidades del software

3. Participación activa de los clientes para no perder tiempo programando funcionalidades que los clientes no pidieron o se entendieron de otra forma

-Testing de Software

1. Insertar el testing en el ciclo de desarrollo del SW: el testing debe ir en cada una de las etapas para disminuir la cantidad de errores y mantener un estándar alto de calidad

2. Documentar continuamente: si llega una nueva persona a probar el SW sabrá el estado de la fase de testing por lo que no perderá tiempo probando funcionalidades que ya fueron testeadas

3. Implementar un ambiente para testing: lo ideal es que el ambiente de testing sea lo mas parecido al de producción para que al implementarlo disminuya el porcentaje de fallos

4. Generar métricas para ver la calidad actual y compararla con versiones anteriores

5. Generar casos de prueba claros para ayudar a que los testers puedan realizar un trabajo eficiente

-Implementación de SW:

1. Mantener un tiempo en marcha blanca dónde se mantengan sistemas antiguos y nuevos de manera paralela para que los usuarios se familiaricen con el nuevo software

2. Hacer seguimiento a las aplicaciones desarrolladas para detectar errores

3. Involucrar a los usuarios ya que ellos son los que usarán el sistema y de acuerdo a ellos validaremos que nuestro software funcione correctamente

4. Hacer respaldo de las bases de datos en caso de que cualquier cosa pueda fallar

5. Verificar el ambiente de producción si cumple con los requisitos del SW para en un futuro no tener problemas de velocidad o memoria