Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e ingenierías

División de Tecnologías para la Integración Ciber-Humana

Computación Tolerante a fallas

Actividad 4: Métodos para la prevención de defectos

Profesor: Michel Emanuel López Franco
Elaborado por: Sanchez Fregoso Jose Manuel
Código: 215476966
Carrera: ingeniería en computación

Tabla de contenido

Introducción

Normalmente en el mundo de desarrollo de software estamos acostumbrados introducir errores o no pensar en escenarios de los cuales se puede encontrar un defecto, bug o error, y es practico simplemente resolverlo una vez encontrado.

Pero en la teoría de la tolerancia de fallas y el análisis profundo del mismo, encontramos desde métodos de cierta forma se considerarían como un consejo para reducir la cantidad de defectos que se ingresan en una aplicación. Pero también existen múltiples métodos los cuales tienen toda una metodología de desarrollo con el propósito de no solo encontrar el error y resolverlo, si no el integrar la información obtenida en el desarrollo para optimizar a futuro el mismo.

Desarrollo

Métodos para la prevención de defectos

Normalmente se considera todos los métodos dentro de una misma categoría, pero debido a la variedad de los resultados a la hora de hacer la investigación, podemos separar dichos métodos en subcategorías que gustaría llamar Métodos heurísticos y Métodos científicos. A partir de este momento me referiré a los “Métodos para la previsión de reportes” como “Los métodos, o método” debido a la longitud del término y aplicando un principio de contexto dentro de la información (Si hablamos de prevención de defectos, no me referiré a los métodos, como técnicas de horneado)

A que me refiero con esto o de donde nace dicha necesidad de dividir los métodos, es muy simple. Los Métodos de carácter Heurísticos fueron técnicas que se adoptaron por la comunidad por medio de a las experiencias de los resultados en base a una innovación para resolver un defecto, dichos métodos se pueden apegar bastante bien aún a método científico, pero tiene el defecto que carece de un “Framework” establecido el cual cuenta con un respaldo científico comprobado (incluso es bastante ambiguo dependiendo los factores con los que se trabajen).

Por otro lado los métodos científicos son aquellos que se les desarrollo toda una metodología de ejecución la cual sin excepción alguna cumple con su trabajo y existen evidencias que respaldan antes el mundo su funcionalidad, dejando de lado la ambigüedad que se puede llegar a producir por un factor (ya sea humano u otro).

Cabe resalta que la clasificación de los métodos es a mi opinión y forma de ver los distintos métodos.

Métodos para la prevención de defectos heurísticos

En esta categoría se encuentran las técnicas que se usan para la prevención, sin embargo, no cuentan con un respaldo científico con un factor ambiental estático (depende de una persona, un software en específico, etc.). Estos métodos son:

- Revisión de código (Code Review): Esta tenca como su nombre lo dice, consiste en la revisión de código por compañeros de equipo, en busca de errores, malas practicas y problemas de diseño.

- Pruebas unitarias: Consiste en el escribir pruebas unitarias para que cada componente de software con el propósito de detectar defectos, buscando el correcto funcionamiento de este.
- Pruebas de integración: Similar a las pruebas unitarias, nomas que ahora buscamos que los diferentes componentes de software funcionen correctamente juntos.
- Automatización de pruebas: La automatización de pruebas tanto unitarias como de integración, nos permiten ejecutarlas de forma rápida y consistente, lo cual ayuda a comprobar defectos de forma eficiente.
- Análisis de código estático: Son herramientas que se usan para para identificar problemas de estilo, convención de programación y errores en código fuente (ESLint, PyLint, etc.).
- La gestión de requisitos: Asegurarse que los requisitos estén claros, bien definidos, y documentados correctamente.
- Estándares de codificación: El establecer estándares de codificación y cumplirlos, vuelve el código fácil de entender y evitar defectos.
- Historial de cambios: Examinar la historia de cambios por medio del software de control de versiones para identificar defectos que no se encontraban ahí, o dar seguimiento a errores que aún siguen sin resolver.
- Análisis estático de seguridad: Utilizar herramientas de código las cuales hacen un análisis estático del sistema para detección de vulnerabilidades de seguridad de Código.
- Planificación y Estimación Realista: Aunque no se crea el establecer plazos realistas y realizar estimaciones de tiempo, ayuda evitar que se generen defectos debido a la presión del tiempo.

Estos son unos cuentos métodos de los autodenominados por mi como heurísticos, ya que algunos de ellos se pueden contar como estáticos o exentos de factores ambientales (personas), pero algunos de ellos si llegan a depender de esto por lo cual puede variar de caso en caso.

Métodos para la prevención de defectos científicos

En este apartado hablaremos de los métodos que se pueden considerar mas robustos con un proceso/metodología de trabajo la cual permite establecer una meta concisa y como desarrollar dicha meta.

Defect Clustering

Este método se basa en el principio de que un pequeño conjunto de tipos de defectos suele ser responsable de la mayoría de los defectos encontrados en el software. Centrado en la identificación y clasificación de patrones recurrentes o grupos de defectos que comparten características similares. En algunas ocasiones se llega a utilizar la combinación de análisis de datos históricos de defectos.

Los pasos para la implementación son:

- Recolección de datos: Se busca el recopilar datos históricos de los defectos anteriores o de iteraciones anteriores del mismo proyecto. Con el fin de incluir

información sobre el tipo de defecto, su gravedad, la fase de desarrollo donde se encontró, etc.

- Análisis de los datos: Como lo dice su nombre, usando técnicas de análisis de datos, se buscan patrones y tendencias de los defectos históricos. Se hace uso de gráficos, estadísticas y otras herramientas.
- Identificación de grupos de defectos: Se buscan grupos de defectos que muestren similitudes en términos de tipo, causa u otra característica relevante. Se pueden usar métodos como una matriz de correlación o técnicas de minería de datos para encontrar estos grupos.
- Clasificación y etiquetado: Una vez agrupados le les puede clasificar y etiquetar según las características comunes, ayudando a entender los defectos predominantes.
- Priorización y enfoque: Con la información de sobre los grupos de defectos, se pueden priorizar los esfuerzos de prevención y corrección.
- Implementación de mejoras: Con base en el análisis de defectos y su clasificación, se implementan mejoras en los procesos de desarrollo y las practicas de calidad para prevenir la recurrencia de defectos similares en el futuro.

Defect taxonomy

Este método se centra en crear una estructura jerárquica de categorías para clasificar y organizar los defectos. Se busca una forma sistemática de categorizar los defectos con el fin de comprender mejor sus naturaleza y origen. Los pasos a llevar a cabo son:

- Definición de categorías: En este paso se identifican las categorías clave en las que se pueden clasificar los defectos. Estas categorías pueden incluir tipos de defectos (de interfaz, de lógica, rendimiento, etc) o cualquier otro criterio relevante.
- Creación de una estructura jerárquica: Se organizan las categorías en una estructura jerárquica, lo que permiten la subcategorización de defectos. Esto puede ser similar a un árbol de decisiones con categorías principales y subcategorías.
- Definición de criterios de clasificación: Establece criterios claros para asignar defectos a categorías especificas en función de sus características. Estos pueden incluir la gravedad, la fase del ciclo de vi da en la que se encontró el defecto, la causa raíz, etc.
- Entrenamiento del equipo: Se busca que los miembros del equipo comprendan y sigan las pautas de clasificación de defectos. Esto garantiza la coherencia en la aplicación de la taxonomía.
- Clasificación de defectos: Cuando detectan defectos. Se asignan las categorías apropiadas según los criterios definidos en la taxonomía
- Análisis de datos: Con la información obtenida se realiza un análisis de y se detectan tendencias. Con el fin de detectar problemas recurrentes y áreas problemáticas en el desarrollo de software.

- Mejora continua: Con los datos del análisis, se implementan mejoras en los procesos y practicas recurrentes para prevenir defectos en las categorías más criticas

Root Cause Analysis(RCA)

El RCA busca de forma sistemática identificar las causas fundamentales o raíces de los problemas o defectos de un proyecto. El objetivo es evitar que los mismos problemas vuelvan a ocurrir abordando sus causas raíz. Los pasos para desarrollarlo son:

- Identificación del problema: Este paso consiste en el identificar un defecto en específico. El defecto puede ser de cualquier tipo de problema, desde defecto de código, hasta defecto de diseño.
- Recopilación de datos: Se reúnen datos relacionados al defecto, como registros de seguimiento, documentación, informes de pruebas y cualquier otra información relevante.
- Análisis de los síntomas: Se examinan los efectos visibles del problema, pueden ser errores de ejecución, fallos del sistema, retraso del proyecto o insatisfacción del cliente.
- Identificación de causas inmediatas: Se determinan las causas inmediatas del problema, que son los pasos específicos que llevaron al problema.
- Profundización en causas raíz: Con las causas inmediatas identificadas, se utiliza la técnica de Ishikawa para profundizar en las causas raíz subyacentes que contribuyeron al problema.
- Plan de acción correctivo: Se desarrolla un plan de acción para abordar la raíz ya identificada, esto puede incluir cambios en procesos, procedimientos, capacitación o modificaciones en el código.
- Implementación de soluciones: Aplican los correctivos según el plan de acción correctivo de manera efectiva
- Seguimiento y prevención: Se hace un semiento del impacto de las soluciones implementadas y asegurándose de que el problema no vuelva a ocurrir

Conclusión

En un principio uno solamente se imagina con tomar buenas practicas o aplicar ciertos principios es mas que suficiente para reducir la cantidad de defectos o prevenirlos, o detectarlos, sin embargo, existe una metodología más allá de eso, donde por medio de procesos nos indican como integrar en el proceso de desarrollo todos los conocimientos para mejorar el proceso.

De forma personal esto es más lo que uno se esperaría del tema, y es fascinante como algo relativamente simple o básico, puede ser una gran herramienta en el área de desarrollo de software e incluso me gustaría que se implementara en mi trabajo.

Referencias

- Calva, R. C. (2011, octubre 14). Poka Yoke: Técnicas para prevenir errores y defectos. gestiopolis; gestiopolis.com. https://www.gestiopolis.com/poka-yoke-tecnicas-prevenir-errores-defectos/
- Felderer, M. (s/f). Improving requirements testing with defect taxonomies. Ucl.ac.uk. Recuperado el 3 de septiembre de 2023, de http://crest.cs.ucl.ac.uk/cow/25/slides/COW25_Felderer.pdf
- Fiabilidad y tolerancia a fallos — documentación de Sistemas en tiempo real - 0.1-rc. (s/f). Readthedocs.Io. Recuperado el 3 de septiembre de 2023, de https://uned-sistemas-tiempo-real.readthedocs.io/es/latest/tema02.html
- Greyrat, R. (s/f). Métodos y técnicas de prevención de defectos. Barcelonageeks.com. Recuperado el 3 de septiembre de 2023, de https://barcelonageeks.com/metodos-y-tecnicas-de-prevencion-de-defectos/
- Inspectorio Inc. (s/f). Cuatro Pasos para Prevenir Defectos de Producción. Inspectorio.com. Recuperado el 3 de septiembre de 2023, de https://inspectorio.com/es/blog/cuatro-pasos-para-prevenir-defectos-de-producci%C3%B3n
- ManageEngine. (s/f). What is root cause analysis? Manageengine.com. Recuperado el 3 de septiembre de 2023, de https://www.manageengine.com/academy/incident-root-cause-analysis.html
- Metdos para la prevencion de defectos software. (s/f). Bing. Recuperado el 3 de septiembre de 2023, de https://www.bing.com/search?q=metdos+para+la+prevencion+de+defectos+software&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=metdos+para+la+prevencion+de+defectos+softwar&sc=10-45&sk=&cvid=9F43C633C73B457EAEE5D2A92D4CE060&ghsh=0&ghacc=0&ghpl=
- Métodos y técnicas de prevención de defectos. (s/f). Myservername.com. Recuperado el 3 de septiembre de 2023, de https://spa.myservername.com/defect-prevention-methods
- Morteo, R. (2021, marzo 27). Categorización de Defectos Funcionales de Software: una propuesta. Los Morteo de México. https://www.morteo.mx/desarrollo-de-software/categorizacion-de-defectos-de-software-una-propuesta/
- Root cause analysis explained: Definition, examples, and methods. (s/f). Tableau. Recuperado el 3 de septiembre de 2023, de https://www.tableau.com/learn/articles/root-cause-analysis
- Roy, S. (2022, noviembre 25). What is Defect Clustering in Software Testing? BrowserStack. https://www.browserstack.com/guide/defect-clustering-in-software-testing
- van Moll, J. H., Jacobs, J. C., Freimut, B., & Trienekens, J. J. M. (2002). The importance of life cycle modeling to defect detection and prevention. 10th

International Workshop on Software Technology and Engineering Practice, 144–155.
- Wikipedia contributors. (2023, agosto 20). Software bug. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Software_bug&oldid=1171285898
- (S/f-a). Safetyculture.com. Recuperado el 3 de septiembre de 2023, de https://safetyculture.com/es/temas/guia-cero-defectos/
- (S/f-b). Softwaretestinghelp.com. Recuperado el 3 de septiembre de 2023, de https://www.softwaretestinghelp.com/7-principles-of-software-testing/
- (S/f-c). Researchgate.net. Recuperado el 3 de septiembre de 2023, de https://www.researchgate.net/publication/270802963_Using_Defect_Taxonomies_for_Testing_Requirements
- (S/f-d). Indeed.com. Recuperado el 3 de septiembre de 2023, de https://www.indeed.com/career-advice/career-development/root-cause-analysis