Riesgos de las Pruebas en el Software

Dr. Luis Gerardo de la Fraga

17 de julio de 2002

Resumen

Los siguientes apuntes han sido tomados del libro: Software Engineering, Principles and Practice; H. van Vliet; 2000; 2nd Ed. ; Wiley

1 Las Fases en el Desarrollo de Software

Cuando se construye una casa, no se comienza simplemente apilando ladrillos unos sobre otros. En vez de ello, se analizan primero los requerimientos y posibilidades del cliente, tomando en cuenta factores tales como estructura familiar, pasatiempos, finanzas, etc. El arquitecto toma en consideración estos factores cuando se diseña una casa. Solamente después de que se está de acuerdo con el diseño, se comienza con la construcción.

Es conveniente de la misma manera cuando se construye software. Primero, el problema a ser resuelto debe analizarse y se describen los requerimientos de una forma muy precisa. Entonces un diseño es realizado en base a estos requerimientos. Finalmente, se comienza con el proceso de construcción, i. e. la programación actual de la solución. Las fases en el desarrollo del software se muestran en la Fig. 1.

El **modelo de proceso** mostrado en la Fig. 1 es muy simple. En la realidad, las cosas serán usualmente más complejas. Por un ejemplo, la fase de diseño frecuentemente es dividida globalmente en, una fase de diseño arquitectural y una fase de diseño detallada, y frecuentemente se distinguen varias fases de pruebas. Los componentes básicos, sin embargo, permanecen en los dados en la Fig. 1. Estas fases deben de ser pasadas por cada proyecto. Dependiendo de la clase de proyecto y el ambiente de trabajo, puede necesitarse un esquema más detallado.

En la Fig. 1, las fases han sido dibujadas de forma secuencial. Para un proyecto dado estas actividades no están separadas de forma estricta con se indican aquí. Las fases usualmente se sobrelapan y pueden sobrelaparse. Esto es, por ejemplo, puede ser posible comenzar con la realización de una parte del sistema mientras algunas de las otras partes no han sido completamente diseñadas. No hay una progresión lineal estricta de la ingeniería de requerimientos al diseño, del diseño a la realización, etc. El regreso a fases tempranas sucede debido al descubrimientos de errores o cambios en los reque-

Figura 1: Una visión simple del desarrollo de software

rimientos.

El esfuerzo relativo dedicado a cada una de las varias actividades requeridas por el sistema deriva en la regla llamada 40–20–40: solamente el 20% del esfuerzo es gastado en la programación (codificación) del sistema, mientras que las fases precedentes (ingeniería de requerimientos y diseño) y las pruebas consumen un 40% del esfuerzo total.

Dependiendo de las condiciones específicas de las fronteras, las propiedades del sistema a ser construido, pueden encontrarse variaciones a esta regla.

Esto no implica que la regla 40–20–40 es la que debe procurarse. Los errores hechos durante la ingeniería de requerimientos son aquellos que son más costosos de reparar. Es mucho mejor dedicar energía en la fase de ingeniería de requerimientos, que tratar de quitar los errores durante la fase de pruebas, que consume mucho tiempo, o peor aún, durante el mantenimiento. De acuerdo con [1], los proyectos que terminan satisfactoriamente tienen una distribución 60–15–25: 60% de ingeniería de requerimientos y diseño, 15% en la realización y 25% en las pruebas. El mensaje es claro: mientras más posponga la codificación, más temprano terminará.

2 Desde las Trincheras

Las historias aquí presentadas son interesantes porque nos enseñan que la ingeniería de software tiene mucha facetas. Las fallas en el desarrollo de proyectos de software frecuentemente no son unidimensionales. Las fallas no son causadas *solamente* con un resbalón técnico en alguna rutina. No son causadas *solamente* por

problemas de comunicación humana. Frecuentemente es una combinación de muchos resbalones pequeños, los cuales se acumulan en el tiempo, y eventualmente resultan en una falla mayor. Cada una de las historias discutidas enseguida muestran un efecto acumulativo. El éxito en el desarrollo de software no llega si solamente empleamos al programador más brillante. O aplicamos la filosofía de desarrollo más nueva. O tenemos la más extensiva consulta de usuarios. O aún si contamos con el mejor administrador. Es necesario contar con lo mejor de todos. Y aún más que eso.

2.1 Ariane 5, vuelo 501

El viaje inaugural de la lanzadera Ariane 5 sucedió el 4 de junio de 1996. Después de cerca de 40 segundos, a una altitud de menos de cuatro kilómetros, la lanzadera se rompió y explotó. Esta pérdida de $500M USD fue causada por un sobreflujo en la conversión de un número en punto flotante de 64 bits a un entero con signo de 16 bits. Desde un punto de vista de ingeniería de software, la historia del Ariane 5 fue interesante porque la falla puede atribuirse a causas diferentes, a diferentes niveles de entendimiento: pruebas inadecuadas, tipo de reuso equivocado, ó una filosofía de diseño equivocada.

La altitud de la lanzadera y su movimiento en el espacio es medida por un Sistema de Referencia Inercial (SRI). Existen dos SRIs operando en paralelo. Tanto su hardware como software son idénticos. La mayoría del hardware y software del SRI fue conservado del Ariane 4. La conversión fatal tomó lugar en una pieza del software del SRI el cual solo tiene significado antes del despegue. Aunque esta parte del servicio del software no tenía propósito después de que el cohete había sido lanzado, se mantenía funcionando por un número adicional de segundos. Este requerimiento fue puesto más de diez años antes por alguna razón peculiar. Esto permitía una reinicialización rápida en la cuenta hacía atrás, en el caso de que esta se interrumpiera muy cerca del lanzamiento. Este requerimiento no se aplicaba al Ariane 5, pero el software fue dejado sin cambios – después de todo, este funcionaba. Debido a que el Ariane 5 es mucho más rápido que el Ariane 4, al cohete alcanzaba una velocidad horizontal mucho más alta en un periodo de tiempo corto después del lanzamiento, resultando en el sobreflujo ya mencionado. Debido a este sobreflujo, el primer SRI cesaba de funcionar. El segundo SRI se activada entonces, pero dado que el hardware y el software eran idénticos, también fallaba el segundo SRI. Como una consecuencia, datos incorrectos fueron transmitidos del SRI a la computadora de abordo. Sobre la base de estos datos incorrectos fueron comandadas deflexiones completas de la boquilla. Estas causaron una carga aerodinámica muy alta la cual resultó en la separación de los impulsores del cohete principal. En este punto fue iniciada la autodestrucción de la lanzadera.

Hay varios niveles en los cuales puede entenderse y explicarse la falla del Ariane 5:

- Fue una falla de software, la cual pudo haberse revelado con pruebas más intensivas. Esto es verdad: el comité de investigación del evento gestionó el uso de simulaciones extensivas para exponer la falla

- La falla fue causada por el reuso de un componente defectuoso. Esto es verdad así

como también es verdad, debido a las características físicas del Ariane 4, que este defecto nunca se hizo aparente. Hubo muchos vuelos exitosos del Ariane 4, usando esencialmente el mismo subsistema de SRI. Aparentemente, el reuso no es composicional: el éxito en el uso de un componente en un ambiente no garantiza su reuso exitoso en otro ambiente.

- La falla fue causada por un defecto en el diseño. El software del Ariane siguió una filosofía de diseño propia del hardware: si un componente se viene abajo, la causa se asume que es aleatoria y es manejada apagando tal parte e invocando un componente de respaldo. En el caso de una falla de software, la cual no es aleatoria, un respaldo idéntico es de pequeño uso. De la parte de software, una línea diferente debió seguirse. Por ende, el componente debió ser cuestionado para dar su mejor estimación de la información requerida.

2.2 Therac-25

El Therac-25 es una máquina radiológica controlada por una computadora. Tenía tres modos de operación:

- Modo en campo luminoso. Esta posición meramente facilita el posicionamiento correcto del paciente.

- Modo de electrones. En terapia con electrones, la computadora controla la energía y corriente del rayo (que es variable) y también controla los imanes que dispersan el rayo a una concentración segura.

- Modo de fotones (en rayos X). En modo de fotones, la energía del rayo es fija. Un atenuador del rayo es puesto entre el acelerador y el paciente para producir un campo de tratamiento uniforme. Una corriente muy alta (100 veces mayor que en el modo de electrones) se requiere a un lado del atenuador del rayo para producir una dosis de tratamiento razonable en el otro lado.

La máquina tiene una mesa movible la cual pone el equipo necesario en posición. La situación básica de peligro es obvia de lo descrito arriba: un rayo de fotones se emitido por el acelerador, mientras que el atenuador del rayo no está en posición. El paciente es tratado entonces con una dosis la cual es demasiado alta. Esto sucedió varias veces. Como una consecuencia, varios pacientes murieron y otros fueron lesionados seriamente.

Uno de los malfuncionamientos del Therac-25 fue conocido como 'Malfunción 54'. Un paciente fue puesto para tratamiento. El operador tecleó los datos necesarios sobre la consola de la computadora la cual estaba situada en un cuarto adyacente. Mientras realizaba esto, él hizo un error: tecleó 'x' (para el modo en rayos X) en vez de 'e' (para el modo en electrones). El operador corrigió su error moviendo en cursor al campo apropiado, tecleó el código correcto y presionó la tecla de intro un número de veces hasta que el cursor estuvo de nuevo sobre la línea de comandos. Él entonces presionó 'B' (Rayo [Beam] encendido). La máquina paró y emitió el mensaje 'Malfunción 54'. Este particular mensaje de error indica una dosis equivocada, muy alta o muy baja. La consola indicaba sustancialmente una dosis baja. El operador co-

nocía que la máquina frecuentemente tenía peculiaridades, y éstas podían arreglarse simplemente presionando 'P' (proceder). De forma que él lo hizo. El mismo mensaje de error apareció de nuevo. Normalmente el operador podía tener contacto de audio y video con el paciente en el cuarto de tratamiento. No fue posible esa vez: el audio no funcionaba y el monitor se había apagado. Se estimó posteriormente que el paciente había recibido una dosis de 16,000–25,000 rad sobre una zona muy pequeña, en vez de la dosis terapéutica de 160 rad. El paciente tuvo desorientación, entró en coma progresivo, fiebre alta y murió tres semanas después del accidente. La autopsia reveló un daño por sobredosis de radiación en el lóbulo temporal derecho del cerebro.

La causa de este evento peligroso fue investigada en la operación del software de la máquina de radiación. Después de que el operador terminó de meter los datos, comienza el posicionamiento físico de la máquina. El doblamiento de los imanes toma cerca de ocho segundos. Después los imanes son puestos en posición, y se checa de nuevo si algo a cambiado. Si el operador realiza cambios y regresa a la línea de comandos dentro de los ocho segundos que toman los imanes en posicionarse, parte de esos cambios resultaran en cambios en los parámetros internos del sistema, pero el sistema nunca 'piensa' que algo ha sucedido y simplemente continua. Con la consecuencia que se ha descrito arriba.

Los accidentes de este tipo fueron reportados a la FDA (Federal Drugs Administration). La FDA requirió al fabricante para realizar las medidas necesarias. El 'corrección' sugerida fueron las siguientes:

> Efectivo inmediatamente, y hasta tener más noticias, la tecla usada para mover el cursos de regreso a través de la secuencia prescrita (i.e. cursos 'UP' inscrito con una flecha hacia arriba) no debe ser usada para edición o cualquier otro propósito.
>
> Para prevenir el uso accidental de esta tecla, la cubierta de la tecla debe de ser removida y fijar los contactos del interruptor en la posición de abierto con cinta eléctrica u otro material aislante. . . .
>
> Deshabilitar esta tecla significa que si en cualquier introducción de datos prescritos es incorrecta entonces un comando de reset 'R' puede ser usado y la prescripción entera reintroducida.

La FDA no se creyó este remedio. En particular, consideraron que el tono de la notificación no era conmensurable a la urgencia para hacerla. La discusión entre la FDA y el fabricante continuaron por un tiempo antes de que una respuesta adecuada fuese dada a esta y otras fallas del Therac-25.

La máquina Therac-25 y su software provienen de modelos anteriores que eran menos sofisticados. En versiones previas del software, por ejemplo, no era posible moverse hacia arriba y abajo de la pantalla para cambiar los campos individuales. Los operadores notificaron que tratamientos distintos frecuentemente requerían casi los mismo datos, los cuales tenían que reteclearse todos cada vez. Para realzar el uso, se adicionó esta característica de mover el cursos alrededor de los campos individuales. Aparentemente, algo que es amigable al usuario puede tener conflicto con seguridad.

También en modelos anteriores, la posición correcta de la mesa giratoria y otros equipos era asegurada por simples candados electromagnéticos. Estos candados son un mecanismo común para asegurar seguridad. Por ende, son usados en los elevadores para asegurarse que las puertas no pueden abrirse si el elevador está entre pisos. En la Therac-25, estos mecanismos de seguridad fueron reemplazados por software. El software fue hecho en un punto simple de falla. Esta sobreconfianza en el software contribuyó a los accidentes del Therac-25, junto con unas inadecuadas prácticas de ingeniería de software y una reacción inadecuada para la administración de incidentes.

2.3 El servicio de ambulancias de Londres

El Servicio de Ambulancias de Londres (SAL) maneja el tráfico de ambulancias en la cuidad de Londres, incluyendo el centro y gran parte de la zona conurbana. Esto cubre un área de cerca de 600 millas cuadradas, cerca de 5000 pacientes por día en 750 vehículos. El SAL recibe cerca de 2000 llamadas por día, incluyendo más de 1300 llamadas de emergencia. El sistema que se discutirá aquí es un sistema despachador auxiliado por computadora (DAC). Tal sistema DAC tiene la funcionalidad siguiente:

- Maneja las llamadas que recibe, aceptando y verificando los detalles del incidente incluyendo la localización del incidente;
- Determina cual ambulancia debe ser enviada;
- Maneja la movilización de la ambulancia y comunica los detalles del incidente a la ambulancia;
- Toma en cuenta la administración de los recursos de las ambulancias, en particular la posición del vehículo para minimizar los tiempos de respuesta.

Un sistema DAC completo es algo complejo. En el pánico, alguien podría llamar y decir que el accidente ha sucedido enfrente de Foyle, asumiendo que cualquiera conoce donde está localizada esta librería. Un componente de un directorio extensivo que incluya la identificación de los teléfonos públicos ayudaría a resolver este tipo de problema. El sistema DAC también contiene un sistema de radio, terminales móviles en las ambulancias y un sistema de localización automática de vehículos.

El proyecto DAC del Servicio de Ambulancias de Londres fue iniciado en al otoño de 1990. La terminación se contempló para enero de 1992. A esa fecha, sim embargo, el software estaba todavía lejos de estar terminado. En los primeros nueve meses de 1992, es sistema fue instalado pieza por pieza sobre diferentes divisiones del SAL, pero nunca fue estable. El 26 y 27 de octubre de 1992, hubo serios problemas con el sistema y se decidió revertirlo a un modo de operación semiautomático. El 4 de noviembre de 1992 el sistema de vino abajo. La Autoridad Regional de Salud estableció un Equipo Examinador para investigar las fallas y la historia que había llevado a ellas. Ellos acabaron con un reporte de 80 páginas, la cual se lee como una novela de suspenso. Aquí se subrayarán algunos de los puntos tratados en el reporte.

La previsión del sistema DAC podría haber tenido más entendimiento. Ningún servicio de emergencia ha intentado llegar tan lejos. El plan fue mover todo el proceso manual – en el cual las formas eran llenadas y transportadas de un empleado al siguiente vía una cinta transportadora – a una autorización completa, en un solo salto. El esquema fue muy ambicioso. Parece ser que los participantes no tenía muy en mente todos los riesgos que estaban tomando.

Mucho antes de que el proyecto comenzará, ya se había cuestionado a una firma consultora administrativa para tener consejos. Ellos habían sugerido que una solución en un solo paquete podría costar 1.5M de libras y tomar 19 meses. Ellos también habían indicado en su reporte que si no se encontraba una solución empaquetada, los costos estimados se incrementarían significativamente. Eventualmente, se escogió una solución no empaquetada, y solo se recordaron los números de este reporte, o eso pareció.

La licitación resultó en réplicas de 35 compañías. Las especificaciones y el calendario de ejecución fueron discutidos con esas compañías. El calendario propuesto fue de 11 meses (esto no es una falla de ortografía). Y aunque muchos proveedores levantaron preguntas sobre el calendario, les fue dicho que eso no era negociable. Eventualmente, 17 proveedores entregaron propuestas completas. Se seleccionó la oferta más baja de cerca de 1M de libras. Esta oferta fue de cerca de 700,000 libras más barata que la siguiente oferta más barata. Nadie parece que se cuestionó esta enorme diferencia. La propuesta seleccionada superficialmente sugería que la compañía ya tenía experiencia en diseño de sistemas para servicios de emergencia. Esto no era una mentira: ellos ya habían desarrollado sistemas administrativos para tales servicios. El sistema SAL también fue mucho más grande que cualquier cosa que ellos hubiesen manejado antes.

El sistema propuesto podría impactar significativamente la forma en que la tripulación de las ambulancias llevaba a cabo su trabajo. Este podría ser por lo tanto lo supremo a tener en su operación completa. Si la tripulación no presionaba los botones correctos en la forma correcta en el tiempo preciso y en el orden correcto, podría resultar un caos. Aún así, hubo muy poco involucramiento de los usuarios durante del proceso de ingeniería de requerimientos.

El sistema DAC proyectado operaría con un objetivo absoluto y una forma imparcial y movilizaría los recursos óptimos para cualquier incidente. Esto abrumaría muchas de las prácticas de trabajo presentes las cuales la administración consideraba anticuadas y no del interés del SAL. Por ejemplo, el nuevo sistema localizaría los recursos disponibles más cercanos de la estación de origen. El siguiente escenario podría ocurrir:

- La tripulación de John tiene que ir a un accidente a unas pocas millas al este de un base.
- Una vez allí, son dirigidos a un hospital a unas pocas millas más al este para asignar al paciente.
- Si llegan otras llamadas y parece que John es el más cercano, a él se le ordena viajar unas pocas millas más al este.
- Y así sucesivamente.

De esta forma, la tripulación podría operar cada vez más lejos de su base y en un territorio que

no les era familiar. Perdía tiempo, porque tomaban vueltas equivocadas, o mas aún tenían que parar para preguntar por las direcciones. También tenían que viajar más para alcanzar su base a la finalización de una tarea. A la tripulación no le gustó el aspecto del nuevo sistema.

También el nuevo sistema no tomaba en cuenta la flexibilidad de las estaciones de emergencia locales, teniendo que decidir cuales recursos localizaba. En el nuevo esquema, la administración de recursos fue centralizada completamente y manejada por el sistema. De esta forma, suponga que John corría abajo hacia donde las ambulancias estaban estacionadas y la computadora le ordenaba tomar el coche número 5. John tenía prisa y posiblemente no puede reconocer el coche número 5, o quizás éste está estacionado detrás de otro coches De manera que John piensa sobre su paciencia para esperar el coche número 5 y decide tomar el coche número 4. Esto significa un problema.

La gente responsable de estos requerimientos tenían mal juicio o ingenuidad sobre lo que los sistemas de computadoras podrían brindar en las prácticas de los humanos. Las computadoras están aquí para ayudar a la gente a hacer su trabajo, y no viceversa. Las comisas de fuerza operacional están condenadas al fracaso.

El caída del sistema el 4 de noviembre de 1992 fue causada por un error de programación menor. Unas tres semanas antes, un programador que había estado trabajando en una parte del sistema olvidó remover una pequeña parte del texto del programa. El código en sí mismo no era dañino. Sin embargo, este localizaba una pequeña cantidad de memoria cada vez que el sistema generaba una movilización de un vehículo. Esta memoria no era liberada. Después de tres semanas, toda la memoria fue consumida y el sistema se cayó.

El proyecto del SAL no falló debido a ese error de programación. Este solo fue la última gota que derramó al vaso. La previsión del proyecto fue demasiado rígida. La administración tanto del Servicio de Ambulancia de Londres como del contratante tenían muy poca o ninguna experiencia en desarrollo de proyectos de software de este tamaño y complejidad. Fueron demasiado optimistas en su asignación de riesgos. Asumieron que toda la gente podría interactuar con el sistema, podrían hacerlo justamente de la forma correcta, todo el tiempo. La administración decidió sobre la funcionalidad del sistema, sin realizar ninguna consulta con la gente que sería sus usuarios primarios. Cualquier proyecto de tales características está condenado a fallar. Desde el mismísimo primer día.

Referencias

[1] B.W. Boehm. Industrial software metrics top 10 list. *IEEE Software*, 4(5):84–85, 1987.