CAPÍTULO I. EXPOSICIÓN DEL PROBLEMA

Los problemas de programación a los que los desarrolladores se enfrentan hoy en día son sorprendentemente similares a aquellos que preocupaban a los ingenieros en los años sesenta. Ya desde aquellos años se le dio el nombre de "crisis del software" a la era donde se reconoce la dificultad de desarrollar y dar mantenimiento al software, misma que se traduce en recurrentes problemas que ocasionan que los sistemas se encuentren atrasados, fuera de presupuesto y/o que no respondan a las necesidades y requerimientos del usuario.

Aunque pudiera pensarse que el software tenga un ciclo de vida mayor que el hardware, éste no es sustancialmente mayor. Por ejemplo, si el hardware aumenta el ancho de palabra de 32 a 64 bits, se hace necesario un nuevo sistema operativo que permita aprovechar esa diferencia. Si a los pocos meses ese nuevo sistema operativo sale al mercado, es necesario que exista software de aplicación que aproveche aquella plataforma, lo que involucra herramientas que permitan construir aquel software, aumentando así la presión por la producción de nuevos productos, ya sea para diferenciarse de otros de la competencia o para mantenerse en la cumbre de la actualización.

A pesar del tiempo transcurrido, lo que se ha llamado "crisis del software" continúa siendo un serio obstáculo para los avances tecnológicos del cómputo. Mientras que los costos del hardware han descendido de manera dramática, los costos del software siguen incrementándose, por lo que en las últimas décadas se han dedicado grandes esfuerzos a definir principios, herramientas, técnicas y métodos que permitan minimizar los problemas que presentan los proyectos de software. En este sentido, nació la ingeniería de software para estudiar los problemas en el desarrollo y mantenimiento del mismo. En este capítulo se hablará sobre dichos problemas, y en el capítulo tres se presentarán aspectos de la ingeniería de software.

1.1 Evolución del software

En su libro "Ingeniería de software, un enfoque práctico", Roger S. Pressman señala lo siguiente en cuanto a la evolución histórica del software:

> "Durante los primeros años de desarrollo de las computadoras, el hardware sufrió continuos cambios, mientras que el software se veía simplemente como un añadido. La programación de computadoras era un arte de 'andar en casa' para el que existían pocos

métodos sistemáticos. El desarrollo de software se realizaba virtualmente sin ninguna planificación... Durante los primeros años era normal que el hardware fuera de propósito general. Por otra parte, el software se diseñaba a medida para cada aplicación y tenía una distribución relativamente pequeña. La producción de software (es decir, de programas desarrollados para ser vendidos a uno o más clientes) estaba en su infancia. La mayoría del software se desarrollaba y utilizaba por la misma persona u organización. La misma persona lo escribía, lo ejecutaba y si fallaba lo depuraba. Debido a que la movilidad en el trabajo era baja, los ejecutivos estaban seguros de que esa persona estaría allí cuando se encontrara algún error. Debido a este entorno personalizado del software, el diseño era un proceso implícito, ejecutado en la cabeza de alguien, y la documentación no existía normalmente." [Pressman, 1988]

Es significativa la observación en relación con los continuos cambios del hardware y la incidencia de esta movilidad sobre la construcción de programas. El software, en esta primera etapa, era claramente un apéndice del hardware, y la evolución de éste impedía pensar en procesos de desarrollo estándar mientras la plataforma fuese modificándose constantemente. Pero queda claro que en esas etapas, la visión del software como algo similar al hardware -en términos de considerarlo un producto- se consolida.

"La segunda era de la evolución de los sistemas de computadora se extiende desde la mitad de la década de los 60 hasta finales de los 70. La multiprogramación, los sistemas multiusuarios, introdujeron nuevos conceptos de interacción hombre-máquina. Las técnicas interactivas abrieron un nuevo mundo de aplicaciones y nuevos niveles de sofisticación del hardware y software. Los sistemas de tiempo real podían recoger, analizar y transformar datos de múltiples fuentes, controlando así los procesos y produciendo salidas en milisegundos en vez de minutos. Los avances en los dispositivos de almacenamiento en línea condujeron a la primera generación de sistemas de gestión de base de datos. La segunda época se caracterizó también por el uso del software como producto y la llegada de las 'casas de software'. El software se desarrollaba para una amplia distribución en un mercado multidisciplinario... Las casas desarrollaron proyectos que producían programas de decenas de miles de sentencias fuente... todos estos programas -todas estas sentencias fuente- tenían que ser corregidos cuando se detectaran fallos, se modificaran por cambios en los requerimientos del usuario o se adaptaran a un nuevo hardware que se hubiera comprado. Estas actividades se llamaron colectivamente *mantenimiento del software*. El esfuerzo gastado en el mantenimiento del software comenzó a absorber recursos en una medida alarmante. Aún peor, la naturaleza personalizada de muchos programas los hacía virtualmente imposibles de mantener. Había comenzado una *'crisis del software'*." [Pressman, 1988]

La estabilidad del hardware se traduce en esta etapa en un fuerte desarrollo del software y, a su vez, en un importante proceso de estandarización de la producción, lo que consolida la visión del software como un producto. Esta es la visión dominante y la que determina las principales características del concepto de software. Pero independientemente de que esta visión pueda resultar adecuada en algunas situaciones, por ejemplo: los sistemas operativos,

procesadores de texto, hojas de cálculo, etc., no todo el software puede ser visto desde la óptica de producto físico.

El mismo Pressman avala más adelante esta visión:

> "El software es un elemento *lógico* en vez de *físico* del sistema... El software es desarrollado, no es fabricado en un sentido clásico." [Pressman, 1988]

Adicionalmente, es preciso señalar que la mayoría del software se construye a la medida, en vez de ensamblar componentes existentes, aún en esta era en la que las metodologías orientadas a objetos promueven la reusabilidad.

Pero continuemos con la visión de Pressman sobre la evolución del software:

> "La tercera era de la evolución de los sistemas de computadoras comenzó a mediados de los 70 y continúa hoy. El sistema distribuido –computadoras múltiples, cada una ejecutando funciones concurrentemente y comunicándose con alguna otra- incrementó notablemente la complejidad de los sistemas informáticos. Redes de área local y global, comunicaciones digitales de alto ancho de banda y creciente demanda de acceso 'instantáneo' a los datos, supusieron una fuerte presión sobre los desarrolladores de software. La tercera era también se caracteriza por la llegada y amplio uso de los microprocesadores y computadores personales... Las computadoras personales han sido la catálisis para el crecimiento de muchas compañías de software. Mientras que las compañías de software de la segunda era vendían cientos o miles de copias de sus programas, las compañías de software de la tercera era venden decenas o incluso centenas de miles de copias."
>
> "La cuarta era en software de computadora está ahora empezando... Ya, las *técnicas de la cuarta generación* para el desarrollo de software... están cambiando la forma en que algunos segmentos de la comunidad informática construye los programas de computadora. Conforme nos movemos en la cuarta era, continúa intensificándose la crisis del software." [Pressman, 1988]

Como vemos, la rápida expansión de la informática llevó a la escritura de millones de líneas de código mucho antes de que se empezaran a plantear metodologías para el diseño y la construcción del software, así como técnicas para resolver los problemas de mantenimiento, confiabilidad, etc. Desde este punto de vista, lo que se ha llamado "crisis del software" es una consecuencia lógica de esta expansión sin control.

Es entendible que los primeros problemas graves en el desarrollo de software hayan sido detectados originalmente en la industria militar de las naciones desarrolladas, dado que ahí es donde se desarrollaron los primeros sistemas de software grandes y complejos, muchos de ellos de tiempo real. Posteriormente, los sistemas de software comerciales y específicos crecieron en tamaño, complejidad y exigencias de desempeño, manifestando los mismos síntomas.

El primer reconocimiento público de la existencia de la llamada "crisis del software" a la que nos referimos en párrafos anteriores, tuvo lugar en una conferencia organizada en 1968 por la Comisión de Ciencias de la OTAN, donde se convocó a un grupo de alrededor de cincuenta expertos. El objetivo de esta conferencia era trazar el rumbo que permitiera salir de la crisis y hacer de la construcción de software una ingeniería: en palabras de Bauer, "establecer y usar principios de ingeniería orientados a obtener software de manera económica, que sea fiable y funcione eficientemente", surgiendo así una nueva disciplina: la ingeniería del software, de la que se hablará en el capítulo tres. A continuación profundizaremos en algunas manifestaciones de la llamada "crisis del software".

1.2 La "crisis del software"

La llamada "crisis del software" se caracteriza por muchos problemas. Pressman la describe de la siguiente forma [Pressman, 1988]:

- La sofisticación del hardware ha dejado atrás la capacidad de construir software que pueda explotar el potencial del hardware.
- La capacidad de construir nuevos programas no puede descansar, debido a la demanda de nuevos programas.
- La capacidad de mantener los programas existentes está amenazada por el mal diseño y el uso de recursos inadecuados.

Asimismo, señala que como respuesta a la "crisis del software", muchas industrias están adoptando prácticas de *ingeniería del software*.

Como ya hemos comentado, el punto básico de la llamada "crisis del software" es que el software es mucho más difícil de construir de lo que nos indica nuestra intuición; se dice que los proyectos de software sobrepasan sus presupuestos y tiempos de desarrollo, generando productos no confiables.

Pero, ¿es realmente una crisis? Analicemos el término "crisis". Esta palabra se define en el diccionario como "un punto decisivo en el curso de algo". Sin embargo, no ha habido un punto decisivo en el desarrollo de software, sino un lento cambio evolutivo. En esta industria, entonces, habríamos tenido una "crisis" que ha estado con nosotros más de treinta años, lo cual es una contradicción para el término. Tendríamos que hablar de una "aflicción crónica". La palabra "aflicción" se define en el diccionario como "algo que causa pena o desastre", y la palabra "crónica" se define como "muy duradero o que vuelve a aparecer con frecuencia; continuando indefinidamente", por lo que podría ser mucho más preciso decir que es una "aflicción crónica" en vez de utilizar el término de "crisis".

Independientemente de la razonabilidad del término, justo es decir que los casos reportados por los cuales se afirma que la "crisis del software" no ha sido superada por la industria, no representan a la mayoría de los proyectos, por lo que tal vez deberíamos reenfocarnos a lo que ciertamente es una realidad: el desarrollo de software es una actividad que, históricamente, ha presentado diversos problemas que han incidido de manera importante en los resultados obtenidos en los proyectos de sistemas. Así, trataremos de cambiar la visión de "proyectos que fracasan" por "proyectos que presentan problemas", aunque una resolución de alguno de estos proyectos podría ser su cancelación.

Entre los problemas en el desarrollo de software a que se hace referencia, y de manera preliminar, se presentan los siguientes:

- Especificaciones: Los proyectos de desarrollo de software se emprenden, frecuentemente, con sólo una vaga definición de los requerimientos del cliente.

- Expectativas: A menudo los sistemas no responden a las expectativas que de ellos tienen los usuarios. La insatisfacción del cliente con el sistema "terminado" se produce demasiado frecuentemente.

- Comunicación: En muchos estilos de desarrollo, la comunicación entre el cliente y el desarrollador es muy escasa, o prácticamente nula.

- Costo: Los costos del software son muy difíciles de prever, y a menudo son muy superiores a lo esperado.

- Facilidad de modificación: La modificación de software es una tarea compleja, costosa y propensa a errores.

- Plazos: El software se suele entregar tarde y con menos servicios que los ofrecidos al inicio del proyecto.

- Calidad: Este es un punto normalmente cuestionable. Es frecuente que los programas fallen demasiado.

- Eficiencia: Los esfuerzos que se hacen para el desarrollo de software no suelen hacer un aprovechamiento óptimo de los recursos con los que se cuenta.

- Métricas: No se tiene tiempo ni cuidado en recoger datos sobre el proceso de desarrollo de software. Sin datos históricos como guía, la estimación de futuros proyectos encontrará diversos problemas. Sin indicadores sólidos de productividad, no se podrán evaluar con precisión la eficacia de las herramientas, técnicas o estándares.

Es difícil agrupar las causas de problemas en el desarrollo de software en categorías mutuamente excluyentes, ya que muchas de ellas se mezclan entre sí.

A continuación se presentarán algunas estadísticas, un estudio británico y un estudio norteamericano, así como casos famosos, que ilustran los problemas en el desarrollo de sistemas.

Algunas estadísticas

Ante el "amarillismo" que pueden despertar las historias de proyectos de sistemas que han presentado graves problemas en los últimos años, la llamada "crisis del software" no podía escaparse de contar con diversas estadísticas que por sí mismas son poco alentadoras. Sin embargo, conocer algunas de ellas nos permitirá revisar diversas causas de problemas en este tipo de proyectos, punto que será sumamente útil en el desarrollo de este trabajo.

Algunos estudios muestran que por cada seis sistemas nuevos, a gran escala, que son puestos en operación, se cancelan dos. Los proyectos promedio de desarrollo de software exceden su tiempo planeado de realización en un 50%, porcentaje que es superado en proyectos a gran escala. Se dice también que 75% de los grandes sistemas presentan fallas de operación que impiden su funcionamiento o bien funcionan de manera distinta a las especificaciones iniciales [Gibbs, 1994].

El desarrollo de sistemas distribuidos también tiene sus propios problemas. En junio de 1994, IBM publicó los resultados de una encuesta hecha entre 24 compañías líderes que habían desarrollado sistemas distribuidos: 55% de los proyectos habían costado más de lo esperado, 68% sobrepasaron los calendarios de trabajo originales y 88% tuvieron que sufrir cambios sustanciales de diseño [Gibbs, 1994]. Si a esto se agrega que un sistema distribuido incrementa los posibles puntos de falla, ya que las redes y los medios de comunicación agregan complejidad y fragilidad a tales sistemas, se verá que la confiabilidad de operación de los sistemas distribuidos puede ser severamente cuestionada.

El estudio de KPMG

La consultora internacional KPMG realizó, tanto en 1989 como en 1995, estudios sobre proyectos que hubiesen fallado en el cumplimiento de sus objetivos y/o que hubiesen excedido su presupuesto original al menos en un 30%, a fin de determinar su frecuencia, causas, las soluciones que se intentaron para remediar los problemas, y el efecto del proyecto en la organización en que ocurrió. En 1995, por ejemplo, se estableció contacto con 250 organizaciones del Reino Unido. Los resultados de ambos estudios, de acuerdo con Robert Glass [Glass, 1998], pueden clasificarse en tres categorías:

- Resultados *predecibles*: Que reflejan lo que ya se podría haber esperado de acuerdo con la literatura de la ingeniería de software.

- Resultados *sorprendentes*: Que en número son mayores a los resultados predecibles, y permiten encontrar nuevos puntos de vista sobre la ingeniería de software.

- *Tendencias*: Observaciones que comentan aspectos que resultan de la comparación entre ambos estudios.

Entre los resultados *predecibles*, se tienen los siguientes:

- Muchos de los proyectos que presentaban problemas eran grandes y sumamente ambiciosos.

- La mayoría de los proyectos fallan por una multiplicidad de causas. Puede o no haber habido una causa dominante, pero sí una combinación de éstas que contribuyeron a los resultados obtenidos.

- Los problemas en la administración del proyecto son más frecuentes, como causa dominante, que los problemas técnicos.

- Sobrepasar el tiempo establecido es más común que sobrepasar el presupuesto (89% contra 62%, respectivamente).

Los resultados *sorprendentes* fueron:

- Quienes contestaron la encuesta respondieron que pensaban que había más problemas en los sectores financiero y de gobierno que en servicios y manufactura, pero los resultados del estudio muestran que todos los sectores son igualmente sensibles.

- El uso de software *llave en mano* no incidió en la reducción de problemas. De los proyectos estudiados, 47% resultaron de la combinación de soluciones *llave en mano* con aplicaciones a la medida, 24% eran sistemas a la medida, y 22% eran soluciones *llave en mano*.

- Más de la mitad de los proyectos empezaron a mostrar síntomas de problemas durante el desarrollo del sistema, y 25% de ellos mostraron tales síntomas durante la planeación inicial.

- La determinación de que el proyecto se encontraba en problemas provino del grupo de desarrollo en 72% de los casos, mientras que sólo en 19% de ellos la detección de la situación problemática surgió en el nivel administrativo.

- "Tecnología nueva para la organización" fue la cuarta causa más común de problemas en los proyectos.

- 55% de los proyectos no habían llevado a cabo ninguna tarea de administración del riesgo, 38% sí habían incluido actividades en este sentido, y el resto no sabía.

En cuanto a *tendencias*, se concluyó que:

- Las organizaciones se mostraron mucho más renuentes a discutir los problemas en sus proyectos en 1995 que en el estudio llevado a cabo en 1989.

- Las cuestiones relacionadas con el uso de la tecnología son una causa mucho más fuerte de los problemas en los proyectos de software: en 1989 sólo 7% la reportaron como causa, mientras que en 1995, 45% de los encuestados la señalaron en este sentido.

Glass agrega, por su parte, que cuestiones relativas al rendimiento y desempeño de los sistemas, así como las aplicaciones que deben registrar y controlar objetos en movimiento (un almacén o un sistema de control de equipaje, por ejemplo) son también dos causas importantes para los problemas de los proyectos de software [Glass, 1998].

Más adelante, cuando en el capítulo tres se hable de las causas de los problemas en los proyectos de desarrollo de software, se señalarán las seis categorías que al respecto concluye este estudio de la KPMG.

El estudio de *The Standish Group*

En una encuesta realizada en 1995 a directivos de tecnologías de la información en Estados Unidos por *The Standish Group*, se determinaron algunas estadísticas interesantes:

- 31.1% de los proyectos son cancelados
- 52.7% de los proyectos costarán 189% de los presupuestos originales
- En 1995, organizaciones de Estados Unidos gastarían $81 mil millones de dólares en proyectos cancelados, y $59 mil millones en proyectos que excedieron sus presupuestos originales
- 16.2% de los proyectos se terminan a tiempo y dentro del presupuesto establecido
- En organizaciones mayores, sólo 9% de los proyectos se terminan a tiempo y dentro del presupuesto asignado

- 42% de los proyectos terminados en organizaciones mayores cumplen con las características y funciones propuestas originalmente
- En organizaciones pequeñas, 78.4% de los proyectos de software serán implementados cumpliendo con al menos un 74.2% de sus características y funciones propuestas originalmente
- De los proyectos que exceden su presupuesto original (hayan terminado o hayan sido cancelados), en promedio lo hacen en un 189%: 178% en grandes compañías, 182% en medianas, y 214% en pequeñas organizaciones
- De los proyectos que exceden sus calendarios de trabajo originales (hayan terminado o hayan sido cancelados), en promedio lo hacen en un 222%: 230% en grandes compañías, 202% en medianas, y 239% en pequeñas organizaciones
- De los proyectos que no cumplen con la funcionalidad y características comprometidas originalmente (hayan concluido dentro o fuera de los tiempos y costos establecidos al inicio), en promedio cumplen en un 61%: 42% en grandes compañías, 65% en medianas, y 74% en pequeñas organizaciones

La encuesta de *The Standish Group* abarcó compañías grandes, medianas y pequeñas en diferentes sectores: bancos, seguros, manufactura, *retail*, salud, servicios, gobiernos locales, estatales y federales, obteniendo respuestas de 365 ejecutivos. También se hicieron entrevistas directas a personal de algunas de esas compañías. Cabe resaltar que en las 365 compañías se tenían 3,682 aplicaciones en desarrollo, y sólo 431 (12%) de ellas estaban dentro de los tiempos y costos estimados en su inicio.

Los proyectos fueron clasificados en tres categorías, de acuerdo con los resultados obtenidos:

Tipo 1. Proyectos terminados a tiempo y dentro del presupuesto asignado, con todas las características y funcionalidad especificadas al inicio.

Tipo 2. Proyectos terminados y en operación pero que sobrepasaron tiempos y costos, además de ofrecer menos características y funciones que las especificadas originalmente.

Tipo 3. Proyectos cancelados en algún punto del ciclo de desarrollo.

La distribución de los tres tipos de proyectos se muestra en la siguiente gráfica:

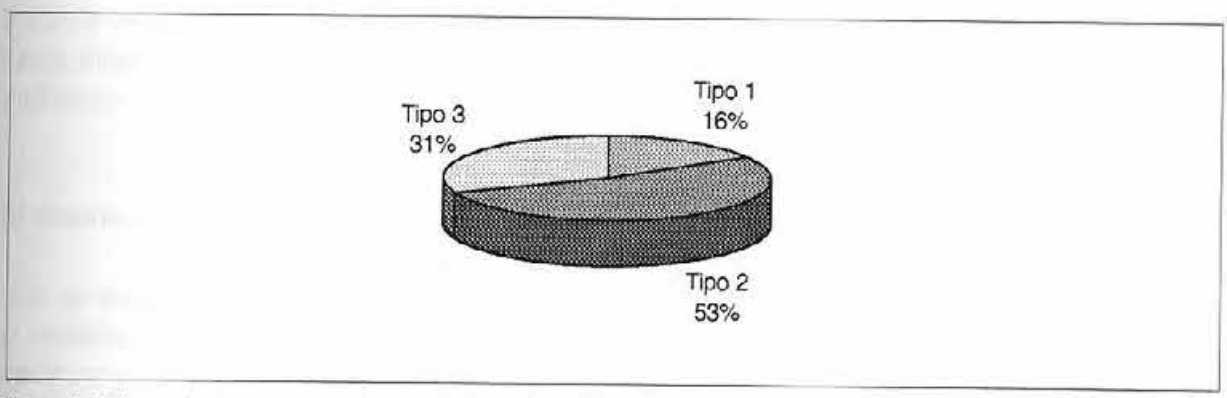

Figura 1. Tipos de proyectos por resultados obtenidos.

Es importante resaltar que estos son los resultados promedio, ya que la distribución no resultó la misma en grandes, medianas y pequeñas organizaciones.

Dos casos famosos

Actualmente la industria enfrenta un reto que ha puesto sobre la mesa el tema de las dificultades en el desarrollo de software: el problema informático del año dos mil. Se conocen las causas y orígenes de este problema, se sabe cómo corregirlo, pero además de aparecer en prácticamente todos los ámbitos de la informática (hardware, software, comunicaciones), tanto en sistemas de información como en componentes, surge la duda: ¿qué hizo el programador? En caso de encontrarse el año "00", ¿programó una caída de sistema, dividirá entre cero, generará un error, ni siquiera hará caso? Los proyectos de conversión que verifican esto para responder a estas y otras preguntas, pero sobre todo para garantizar la continuidad de la operación de las organizaciones, deben finalizar antes del año 2000. Pero son proyectos de software, que como ya hemos mencionado, comúnmente sufren retrasos, en su gran mayoría, pero en este caso no hay prórrogas.

Por otra parte, al hablar del estudio de la KPMG se comentaron las características que en su estudio se tomaron en cuenta para determinar un proyecto como problemático. Sin embargo, debemos mencionar que Glass señala que, en su opinión, un proyecto se sale de control "principalmente por la dificultad de construir el software que requiere el sistema, duplicando recursos tales como presupuesto y costo" [Glass, 1998].

Antes de comentar dos famosos casos de proyectos de software que se salieron de control, justo es mencionar que no todas las historias reflejan fracasos. En los años setenta American Airlines construyó el sistema SABRE, un software para reservación de vuelos que costó $2 mil millones de dólares. SABRE se convirtió en parte de la infraestructura de la industria, y empujó a American Airlines para consolidarse como una de las aerolíneas más importantes a nivel mundial. Como estas, se pueden encontrar múltiples historias de sistemas que sí funcionan y que sí operan: nos rodean en la vida cotidiana, en el sistema financiero, en nuestro automóvil, en fin, nuestro ámbito ya no escapa a equipos y organizaciones donde el software sí hace lo que se pretendía que hiciera al momento de gestarse. Pero como se aprende más de la anormalidad que de la normalidad, a continuación se presentarán dos casos tristemente famosos de proyectos de desarrollo de sistemas que enfrentaron múltiples problemas.

El sistema de control de equipaje del nuevo aeropuerto internacional de Denver

Uno de los casos más estudiados por el impacto económico que tuvo, entre otras razones, es el sistema de control de equipaje del nuevo aeropuerto internacional de Denver. Este aeropuerto tiene el doble de tamaño que Manhattan, con capacidad para permitir el aterrizaje

de tres jets al mismo tiempo. Se pretendía que su sistema de control de equipaje manejara carros similares a aquellos utilizados en las minas, sólo que éstos serían inteligentes: a lo largo de veintiún millas de rieles, cuatro mil "teleautos" independientes se moverían y entregarían equipaje entre los mostradores, puertas y áreas de reclamación de equipaje de veinte aerolíneas diferentes. Un "sistema nervioso central" formado por cien computadoras enlazadas en red coordinaría a cinco mil sensores, cuatrocientos receptores de radio y cincuenta y seis lectores de código de barras para garantizar la llegada segura y oportuna de cada una de las piezas de equipaje.

La apertura del nuevo aeropuerto de Denver estaba programada para octubre de 1993, fue pospuesta para diciembre del mismo año, luego para marzo de 1994 y finalmente para mayo del mismo año. Una de las causas: errores en el software de control de equipaje. En junio, con pérdidas trimestrales de más de un millón de dólares dado el retraso en la apertura del aeropuerto, se aceptó que no se podía predecir la fecha en que el sistema de equipaje podría estabilizarse para que el aeropuerto pudiera comenzar a funcionar. En agosto de 1994 se aprobó la construcción de un sistema de respaldo, mismo que tuvo un costo de $50 millones de dólares, de los cuales $10.5 millones USD correspondieron al software y el resto a cuestiones eléctricas y de modificación de edificios. Finalmente, el aeropuerto fue inaugurado a fines de febrero de 1995, 16 meses después de lo programado, con costos no previstos por $2 mil millones de dólares.

Indudablemente, una causa determinante en la grave problemática presentada en este proyecto, fue que inicialmente se trataba del manejo de equipaje de una sola línea aérea, ya que es práctica común en los Estados Unidos que cada aerolínea implante su propio sistema de control de equipaje al construir un aeropuerto, sin embargo, en el caso de Denver se tomó la decisión de escalar el proyecto de United Airlines para que cubriera a todas las líneas que utilizarían el aeropuerto. Esto es, los responsables del aeropuerto trataron de escalar un sistema relativamente simple a uno muchísimo más complicado, lo que ocasionó una espectacular alza en los presupuestos y en los tiempos de desarrollo. Al final, hubo que regresar a los requerimientos originales para poder hacer que el sistema funcionara.

La mayoría de los problemas que presentaba el sistema estaban relacionados con errores en el software, aunque también los problemas mecánicos jugaron un papel importante. Al final, el nuevo aeropuerto internacional de Denver inició operaciones no con uno, sino con tres sistemas de control de equipaje. El proyecto original de automatización de entrega de equipaje regresó a su primer usuario, la aerolínea United Airlines, que además es quien mueve la mayor cantidad de pasajeros en Denver.

Podríamos resumir que algunos de los riesgos que presentó el sistema fueron:

- Proyecto de enormes dimensiones
- Alta complejidad
- Tecnología nueva

- Gran número de entidades a las que el mismo sistema debía dar servicio
- Alto grado de incertidumbre en cuestiones técnicas y de definición del proyecto
- Tiempos de desarrollo sumamente comprometidos

El sistema de conductores y vehículos del estado de California

El estado de California en los Estados Unidos, a través de su *Department of Motor Vehicles* (DMV), decidió, en 1987, desarrollar una base de datos relacional para registrar a 31 millones de conductores y 38 millones de vehículos, fusionando así ambos sistemas que para entonces tenían más de 28 años de vida. Una de las principales razones para el proyecto era el uso de la nueva tecnología de bases de datos. Entre otros servicios, se esperaba que se pudieran instalar kioscos de trámites durante 1993. Pero en lugar de eso, el Departamento decidió cancelar el proyecto, ya que el costo del proyecto se había incrementado en un 650% respecto al presupuesto original (se iba a requerir invertir $100 millones de dólares más), y la fecha de entrega se disparaba hasta 1998. En total se invirtieron más de $44 millones de dólares durante más de seis años.

Una de las causas de los problemas en este proyecto fue la adopción de tecnología de bases de datos relacionales que no podía satisfacer los requerimientos de un sistema transaccional para dar servicio a más de 40,000 usuarios, con picos de hasta 30 transacciones por segundo, y de un millón por día. Adicionalmente, se reporta que el proyecto no fue apoyado por los niveles directivos, ni hubo colaboración del usuario; adicionalmente se detectaron fallas de planeación, especificaciones pobres y objetivos poco claros, además de sufrir de disputas políticas entre las entidades del estado.

El DMV opera actualmente sus sistemas como módulos separados, añadiendo servicios de valor agregado como son los trámites vía telefónica, utilizando tecnología IVR.

1.3 Proyectos de software estudiados

A fin de presentar casos de proyectos con problemas mucho más cercanos a mi realidad como profesional dedicado al desarrollo de software, a continuación se presenta una tabla con 61 sistemas con los que he tenido contacto en los últimos diez años, y en los que desempeñé diferentes roles, como sigue:

- Programador
- Programador para dar mantenimiento
- Documentador
- Analista-programador