El modelo de desarrollo para un Proyecto Fin de Carrera en Ingeniería Técnica en Informática

Agustín Cernuda del Río

Departamento de Informática - Universidad de Oviedo
Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo
Facultad de Ciencias - C/ Calvo Sotelo, S/N - 33007 Oviedo
guti@lsi.uniovi.es

Resumen

El Proyecto Fin de Carrera de la Ingeniería Técnica en Informática presenta al alumno una problemática especial, y la actividad del profesor como guía adquiere gran relevancia. Además, este primer proyecto de un alumno puede tener gran influencia en su posterior comportamiento como profesional. Pensando en ciertos problemas que aquejan a la industria del desarrollo de software, parece apropiado aprovechar el Proyecto Fin de Carrera para que el alumno afronte su vida profesional prevenido contra estos problemas. En este documento se realizan diversas reflexiones sobre un posible modelo de gestión del proyecto que responda a estos fines, modelo que estamos aplicando en nuestros proyectos de Ingeniería Técnica en Informática. Precisamos que la problemática referida se hace especialmente patente en proyectos *nuevos*, no *de mantenimiento*, y nos limitaremos a ese ámbito. En este caso, además, hablaremos sólo de proyectos *individuales*.

1. La gestión de proyectos en el mundo profesional

El desarrollador profesional de software debe afrontar un panorama de notables desafíos. Dos tercios de los proyectos terminan claramente fuera de coste y plazo, los grandes proyectos muestran un retraso medio del 25% al 50% de la estimación inicial, y cuanto mayor es el proyecto mayor es la probabilidad de cancelación [2]. De 600 empresas encuestadas, el 35% tenía al menos un proyecto fuera de control [3]. Existen grandes dificultades para planificar costes y plazos, así como para mantener una calidad adecuada de los productos, un bajo coste de mantenimiento y un conjunto de prestaciones apropiado.

Muchos de los problemas que plantea el desarrollo de software no son de índole estrictamente técnica. Gran parte de estos inconvenientes no obedecen a dificultades con los lenguajes de programación, el análisis o el diseño, sino en el campo de la gestión de proyectos. Algunos de los errores clásicos que influyen de manera determinante en la pérdida del control de un proyecto [3] pueden verse en la Ilustración 1.

I.	Expectativas poco realistas
II.	Hazañas, ilusiones
III.	Planificación excesivamente optimista
IV.	Gestión de riesgos insuficiente
V.	Abandono de la planificación bajo presión
VI.	Escatimar en las actividades iniciales y/o en el control de calidad, en favor de la codificación
VII.	Programación a destajo
VIII.	Exceso de requisitos
IX.	Control insuficiente

Ilustración 1. Algunos errores clásicos.

Estos patrones de conducta (y podrían citarse muchos otros) se cuentan entre los más nocivos para la profesión, y son causa directa del fracaso de buena parte de los proyectos informáticos. Puede verse que en gran medida no se ven involucrados conocimientos estrictamente académicos, sino más bien hábitos de comportamiento. Según Roger Burlton, "las organizaciones que intentan implantar la disciplina de la ingeniería del software antes que la de gestión de proyectos están abocadas al fracaso".

2. El papel del Proyecto Fin de Carrera

A lo largo de sus estudios, un alumno tiene ocasión de caer en este tipo de hábitos e incluso de sufrir las consecuencias. Habrá comprobado más de una vez, por ejemplo, que *desear* terminar una práctica en una noche no lo hace *posible*. No obstante, es nuestra opinión que esa experiencia circunstancial y oportunista no es, en general, suficiente para su formación.

Apoya esta idea el hecho de que los desarrolladores profesionales incurren una y otra vez en este tipo de comportamientos "suicidas", bien porque no los identifican como la causa de fracasos anteriores, bien porque no perciben que las cosas puedan transcurrir de otra manera. Desgraciadamente, está muy extendida la idea de que en el desarrollo de software real "no hay tiempo" para actividades como control de calidad, buen análisis o diseño, saneamiento del código, etc. No parece razonable esperar que un estudiante con tan limitada experiencia en el desarrollo llegue por sí mismo a conclusiones más acertadas que los profesionales en un plazo tan breve.

Se impone, pues, transmitir al alumno unos hábitos que en el futuro le protejan frente a estos errores clásicos, de manera que al menos sea consciente de cuándo está haciendo un mal trabajo. Aunque estas normas básicas de actuación deben promoverse en todo momento y circunstancia durante el desarrollo de los estudios, entendemos que la piedra de toque definitiva es el Proyecto Fin de Carrera (en adelante PFC), puesto que:

- El alumno estará centrado en el proyecto, y podrá ser más consciente de su propio proceso de desarrollo que en los casos en que dicho desarrollo es sólo un medio para alcanzar conocimientos de alguna otra materia.
- El desarrollo del PFC guarda mayor parecido con el desarrollo de un proyecto real en cuanto a objetivos, técnicas y (especialmente) responsabilidades que el alumno debe asumir.
- Esta situación es la primera en la que se da tal parecido (en la mayoría de los casos).

Si se acepta esta idea, durante el desarrollo del PFC sería conveniente adoptar métodos de trabajo que formasen al alumno con vistas a evitar los males endémicos del desarrollo de software que se han citado más arriba. A nuestro juicio, sería muy conveniente contemplar los siguientes aspectos:

- Ciclos de vida
- Estimación
- Planificación
- Gestión de riesgos
- Calidad
- Documentación

El resto de este artículo está dedicado a proponer algunas prácticas cuyo fin es potenciar estos elementos para la consecución de los objetivos planteados anteriormente. Las normas de desarrollo de PFC que aplicamos recogen estas ideas. En cada apartado se presentarán, a modo de resumen, los hábitos recomendados, y una referencia a los "errores clásicos" de la Ilustración 1 a los que se pretende combatir en cada caso.

3. Ciclos de vida

Un punto fundamental para que el desarrollo de un PFC transcurra de manera satisfactoria es la selección de un ciclo de vida para el mismo. Quedarán después por resolver muchos detalles *tácticos* de planificación, pero esa tarea será aún más difícil sin una decisión previa a nivel *estratégico*.

La falta de visibilidad es un problema muy preocupante (y frecuente) en los proyectos de desarrollo de software. Quien va a recibir un producto llave en mano en cierta fecha lejana pero no tiene información fiable sobre el progreso del proyecto está asumiendo graves riesgos. Si hay errores de base se descubrirán demasiado tarde; si el proyecto fracasa, se habrá gastado eventualmente todo el dinero.

En el caso de un PFC este efecto es igualmente pernicioso. No es raro que se dé por supuesto un desarrollo en cascada, con las fases académicas tradicionales de análisis, diseño, codificación y pruebas. Este esquema, sin embargo, es difícil de aplicar con éxito en un entorno profesional (a menos que se gestione con gran habilidad el conjunto de requisitos o el proyecto sea muy similar a otros anteriores); no digamos para un alumno sin experiencia en

planificación ni estimación. Debe desarrollar dotes casi adivinatorias para definir en un solo paso la solución a un problema de envergadura notable. Durante el desarrollo afrontará una gran incertidumbre sobre la fecha de terminación o la marcha de los trabajos, ya que durante gran parte del proyecto no tendrá nada tangible o ejecutable.

Proponemos como remedio discutir de manera explícita cuál va a ser el ciclo de vida a adoptar, dependiendo de las circunstancias del proyecto. En este entorno, creemos que es adecuado casi en todos los casos algún ciclo de vida que produzca resultados intermedios: desarrollo en espiral, entrega por etapas, desarrollo evolutivo o similares. El alumno debe *ver algo* funcionando periódicamente. Eso le dará confianza, le ayudará a concretar el proyecto y le permitirá ejercitar la difícil habilidad de la estimación.

En particular, creemos también que en el caso de estudiantes es imperativo (y así se refleja en nuestras normas) el desarrollo de al menos un prototipo, ya sea de la interfaz de usuario, de viabilidad tecnológica o ambos a la vez. Por supuesto, el prototipo se desechará sin paliativos. El alumno habrá adquirido experiencia sobre cómo *no* debe hacer las cosas la segunda vez; parece la única forma de asimilar técnicas nuevas reduciendo el impacto negativo en la calidad del producto final, y ayuda además a mitigar la notable sensación de incertidumbre del alumno (recordemos que probablemente es la primera vez que debe resolver un problema en el que el enunciado no está cerrado, sino que es muy vago). Simplemente, se trata de aplicar el viejo consejo: *build one to throw away* [1].

Prácticas recomendadas:

- Elección explícita del ciclo de vida al principio del proyecto
- Realización obligatoria de al menos un prototipo (de IU, de viabilidad tecnológica o ambos)

Problemas mitigados: IX, I, II, III, VIII

4. Estimación

La estimación suele resultar especialmente difícil, porque sólo puede realizarse de forma rigurosa si se tiene implantado un buen sistema de medición sobre proyectos anteriores; y aun en ese caso pasa por fases de progresiva precisión, pero inicialmente hay que admitir un abultado margen de error. En la práctica, juegan un importante papel la experiencia y el instinto del desarrollador.

A la luz de esto, es evidente que un alumno de un PFC no se encuentra en una situación ideal para realizar buenas estimaciones (y así suele percibirlo él mismo). La estrategia que proponemos pasa por inculcar los siguientes principios, que serán de aplicación en el mundo profesional:

- Ya que el problema no tiene solución satisfactoria, lo mejor que tenemos es el criterio del desarrollador.
- Una estimación **no se negocia**.
- El desarrollador se compromete con sus propias estimaciones, que debe tratar de cumplir.
- Hay que dividir las tareas hasta llegar a un tamaño muy pequeño, y estimar esas mini-tareas. La estimación final se obtiene combinando las parciales (no se “inventa”).

Conviene hacer especial hincapié en que las estimaciones no se negocien. El alumno debe estimar el tiempo o esfuerzo lo mejor que pueda; una vez lo ha hecho, el deseo de su jefe (o profesor) no altera los hechos, y por tanto tampoco debe alterar la estimación (salvo que se le aporte información adicional que influya en la misma). El alumno debe aprender a dar (y reconocer) las malas noticias con valentía. Ceder y hacer las estimaciones más cortas sin ningún motivo racional es una de las prácticas más nocivas en el desarrollo de software.

En el ámbito de la estimación, puede ser de utilidad que el alumno mantenga una tabla de tareas, en la que vaya incluyendo cada tarea prevista, la fecha prevista de terminación, y la fecha real. En esta tabla quedará constancia de los errores de estimación, tanto por defecto como por exceso. El objetivo de esto no es “penalizar” al alumno por sus errores, sino hacerle consciente de las cifras previstas y reales (dándole así datos para

el futuro) y hacer que practique con frecuencia la estimación. Esta también es práctica obligatoria en nuestros PFC (Ilustración 2). La estimación de tareas pequeñas (de pocas jornadas, o incluso menos) ayuda a disminuir el error global, ya que los errores por exceso y por defecto se anulan. Por último, cada una de las tareas debe ser *binaria*: se entrega en cierta fecha o no, sin términos medios ni porcentajes.

Prácticas recomendadas:

- El alumno realiza sus estimaciones y se compromete con ellas.
- Se estima con frecuencia, y sobre tareas pequeñas.
- Se mantiene un registro riguroso de estimaciones, cumplimientos e incumplimientos.
- Las tareas están terminadas ó al 100% ó al 0%, sin paliativos.

Problemas mitigados: I, II, III

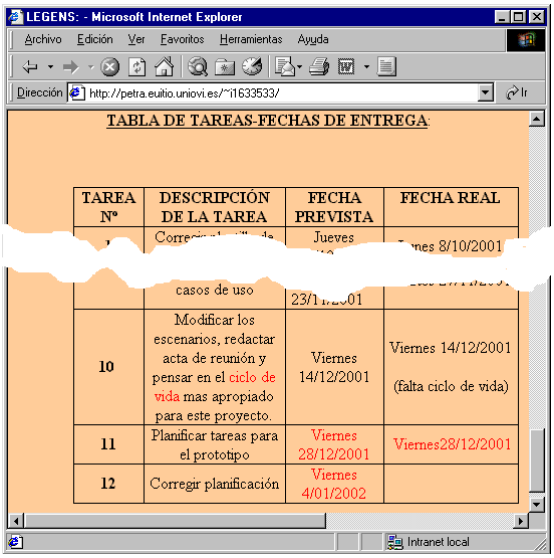

Ilustración 2. Tabla de tareas y fechas en la página web de un PFC

5. Planificación

Las dificultades de la planificación derivan en gran medida de las que plantea la estimación, añadiendo la colocación de las tareas, el establecimiento de dependencias y precedencias, etc. Cobra aquí particular importancia el efecto de las desviaciones.

Con el fin de evitar los problemas mencionados en el apartado 1, conviene vigilar estrechamente el desarrollo del PFC para evitar algunos males frecuentes, que tienen su raíz en una actitud muy común: **el abandono de la planificación bajo presión**. Cuando el plan se incumple, el desarrollador simplemente empieza a trabajar a destajo para acabar lo antes posible, ignorando el plan. Durante el PFC se debería hacer hincapié en evitar esto.

Proponemos que el alumno *siempre* trabaje sobre un plan. Además, este plan debe contener hitos binarios (que se cumplen o se incumplen en su totalidad); cuando se incumple un hito, se analizan las causas y se modifica la planificación de manera adecuada. No se abandona el plan. Sin embargo, es lícito rehacer el plan cuantas veces sea necesario. No se obliga al alumno a acertar en sus apuestas (al fin y al cabo, está aprendiendo), pero cuando falla en su apuesta, siempre debe sustituirla por otra.

Asimismo, el incumplimiento de hitos obliga a replanificar: se supone que el retraso producido probablemente se reproducirá de manera global en el proyecto. No es aceptable asumir que el tiempo perdido se recuperará por sí solo más adelante.

Proponemos también que el alumno lleve un "diario del proyecto", en el que anote los sucesos importantes. Los incumplimientos del plan y las acciones correctoras se reflejan siempre en este diario.

Prácticas recomendadas:

- El plan contiene hitos que se cumplen al 100% ó al 0%.
- El incumplimiento de un hito conlleva una replanificación.
- En el diario del proyecto queda constancia escrita de ambos sucesos.
- El alumno siempre trabaja sobre un plan; puede modificarlo repetidamente, pero no abandonarlo.

Problemas mitigados: V, VII, IX

6. Gestión de riesgos

Es muy frecuente que en el entorno profesional no se realice una gestión de riesgos efectiva; lógicamente, en un PFC esto es aún más habitual.

Sin embargo, el obligar al alumno a realizar una gestión de riesgos siquiera mínima tiene un gran valor didáctico. En primer lugar porque le hace consciente de los peligros que acechan al proyecto, y cuando alguno se manifieste estará sensibilizado; y en segundo lugar porque la gestión de riesgos es también un ejercicio adicional de planificación y gestión. Basta con un sencillo documento de una o dos páginas, en el que el alumno identifique los peligros importantes y adopte acciones preventivas (que pueden ser tan simples como cambiar el orden de ciertas tareas).

Un riesgo muy importante en los PFC (y que no se produce tanto en entornos profesionales) es el alargamiento indefinido del proyecto. Es frecuente en todo tipo de alumnos, pero más en el caso de alumnos que encuentran trabajo antes de obtener la titulación. Para mitigar este riesgo, es aconsejable que el tutor obligue a un progreso constante, aunque sea muy lento; esto entronca con el hábito ya mencionado de que toda tarea tenga una fecha de entrega prevista que el alumno se esfuerce por cumplir.

Prácticas recomendadas:

- Una gestión de riesgos, siquiera elemental.
- Norma de obligado cumplimiento: un avance constante (aunque sea lento) del proyecto. En caso de abandono manifiesto, cancelación del mismo.

Problemas mitigados: IX, V

7. Calidad

Quizás la enseñanza más importante que el alumno puede obtener del PFC es que la calidad no es una cualidad adicional que se puede incorporar al proyecto como una prestación más (o renunciar a ella), sino que debe ser algo intrínseco a cualquier proyecto. Y por una sencilla razón: la calidad es precisamente el camino para terminar el proyecto en menos tiempo y con menos esfuerzo [3].

El PFC es el momento adecuado para luchar contra el mito de que los proyectos reales son artesanía. Si el alumno termina su formación técnica y ya en la etapa final de la misma ve que lo estudiado en otras asignaturas sobre la calidad es sólo teoría, es de esperar que cederá con toda facilidad a las presiones del entorno profesional para desarrollar más rápido, "a costa de la calidad".

No resulta fácil resumir en pocas actividades cómo debe vigilarse la calidad; es una guerra que exige luchar en muchos frentes. Simplemente, debe garantizarse que el alumno comprende claramente que se espera de él que desarrolle software de calidad, *en su propio beneficio* (por puro egoísmo). Deben aplicarse todas las enseñanzas que se transmiten en el Plan de Estudios sobre cómo debe desarrollarse el software.

Una actividad que puede resultar muy beneficiosa en este sentido es la realización de revisiones técnicas formales (RTF), que nosotros proponemos según el modelo descrito en [4]. Además de mejorar la calidad del proyecto en cuestión, si estas RTF se realizan de manera cruzada entre miembros de distintos proyectos, pueden constituir un excelente ejercicio de análisis, diseño y codificación en un problema distinto del que ocupa a cada alumno en su propio PFC, aportándoles una experiencia extra y una visión más amplia del desarrollo.

Por supuesto, son aplicables las mismas reservas que en las RTF entre profesionales: hay que conseguir que los alumnos no consideren a los participantes como enemigos o censores, sino como personas que están ayudándole a terminar su trabajo antes y mejor. De hecho, como personas que están *haciéndole parte del trabajo*. Cada defecto señalado en una RTF ahorrará al alumno muchas horas de depuración, y así debe percibirlo.

Como inconveniente, cabe señalar que los alumnos de otros PFC probablemente no estén en buena situación para comprender un problema que les es ajeno. Esto exige del profesor una selección adecuada del material que se somete a RTF cruzada con otros alumnos, de manera que sea *digerible* por ellos. Otras partes críticas del sistema que no puedan ser revisadas por estos alumnos pueden someterse a RTFs en las que sólo

participen el alumno afectado y algunos profesores (al menos, lógicamente, el director del proyecto).

Prácticas recomendadas:

- En general, aplicar de manera decidida y rigurosa los fundamentos del desarrollo.
- Realizar revisiones técnicas formales, cruzadas con otros alumnos o sólo con profesores.

Problemas mitigados: VI, VII, V

8. Documentación

En realidad, este punto es simplemente otra faceta del problema de la calidad. La necesidad de una buena documentación es un caballo de batalla en los entornos profesionales, y también en los académicos.

Aquí merece la pena llamar la atención sobre la documentación relacionada con la gestión del proyecto en sí. Cuando se habla de "documentación", con frecuencia se piensa en documentación de análisis y diseño, que describe el programa, probablemente con el fin de facilitar su mantenimiento. Pero siendo este un tema importante, no es el único. Conviene reflexionar sobre la documentación del proyecto, que no se refiere de manera directa al programa sino a lo que rodea a su desarrollo.

Como ya se ha dicho, en muchos casos los problemas de un proyecto no son técnicos, sino de gestión. La falta de control del proyecto, la ausencia de planes precisos y ordenados o la falta de información sobre decisiones adoptadas previamente puede ser tan nociva como una mala descripción del código fuente. Existe un problema adicional con este tipo de documentos, que nuevamente va más allá del aspecto técnico: la documentación sobre el proyecto corre el riesgo de convertirse en burocracia, que el alumno percibe como inútil y que realiza cumpliendo un mero trámite.

La documentación sería motivo para un estudio pormenorizado por sí misma, pero aquí nos limitaremos a algunos apuntes relacionados con el tema que nos ocupa.

Utilidad. Es fundamental que la documentación sea para el alumno *una herramienta* y que la perciba como tal. El director del proyecto puede aprovechar algunas de las dificultades específicas que el PFC plantea al alumno (indefinición, incertidumbre, falta de control, dificultad...) para que dicho alumno encuentre en sus propios documentos el apoyo que necesita. Contra la indefinición, una buena lista de requisitos, que pueda consultar y actualizar con frecuencia; contra la incertidumbre, una buena planificación, y así sucesivamente.

Creemos que puede ayudar a esto el que el alumno mantenga una buena organización y accesibilidad de los documentos; imponemos como norma obligatoria el mantenimiento de una página web del proyecto, donde se encuentra toda la documentación (Ilustración 3).

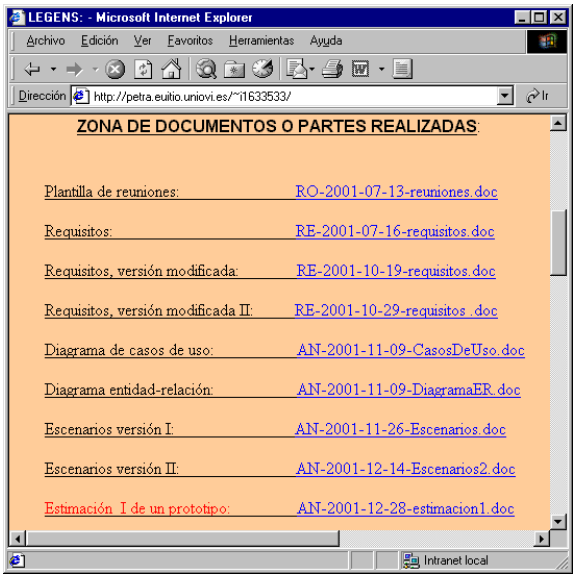

Ilustración 3. Página web con la documentación de un PFC

Burocracia. Hay que evitar a toda costa que el alumno perciba la documentación como un ejercicio de burocracia gratuita. Además de conseguir que la documentación le resulte útil y

aprecie esa utilidad, hay que procurar que se reduzca al mínimo posible.

El que la documentación no tiene por qué convertirse en burocracia queda patente si pensamos en los documentos realmente necesarios (aparte de la documentación de análisis/diseño habituales):

- Un documento de requisitos preciso (probablemente entre 1 y 5 páginas). Esto puede considerarse documentación de análisis, pero tiene un papel activo en la gestión del proyecto.
- Un solo documento de planificación y estimación (tipo Microsoft Project)
- Un documento de gestión de riesgos (1-2 páginas)
- Un "diario del proyecto" en el que se reflejen los sucesos importantes (y en este no tiene importancia una mayor longitud)
- Un historial de fechas de entregas parciales (las mismas consideraciones que en el apartado anterior)
- Otros documentos "históricos" (como actas de reuniones, RTFs, etc.)

Este simple conjunto de documentos permite realizar la mayor parte de las acciones sugeridas aquí, y parece claramente manejable.

Sincronización. El típico problema de que la documentación se adecue a la realidad, y cambie con ella, puede abordarse también adoptando los hábitos de trabajo adecuados. Si el alumno *piensa* sobre sus documentos, y se guía por ellos, cada nueva decisión quedará reflejada en primer lugar de forma natural en los documentos afectados, y posteriormente se llevará a la práctica. Esto puede ser más difícil de conseguir en la documentación que alude directamente al código, pero en los documentos del proyecto (requisitos, planificación, etc.) parece viable, puesto que la codificación es una tarea a la que el alumno está habituado y ha adquirido ciertos patrones de comportamiento, pero el desarrollo de un PFC es una experiencia nueva.

9. Conclusiones

En este artículo se realiza una reflexión sobre el especial papel que el Proyecto Fin de Carrera cumple en la formación de un Ingeniero Técnico en Informática de cara a su futura actividad profesional. Se identifican algunos de los problemas más graves del desarrollo de software en entornos profesionales, y se analiza su relación con algunos aspectos del desarrollo de un PFC; simultáneamente, se sugieren algunas prácticas recomendables con el fin de que el alumno adquiera hábitos que le conviertan en un profesional mejor preparado para luchar contra los problemas descritos.

En general, de lo que se trata es de que el alumno perciba que el desarrollo con arreglo a las directrices de la profesión que se transmiten en los estudios universitarios resulta *práctico* y redunda en su propio beneficio. Asimismo, se pretende que adquiera una clara conciencia de que el dominio de la técnica, por sí solo, no garantiza un buen rendimiento profesional; se necesitan ciertos hábitos de trabajo (y herramientas adicionales) que ofrezcan un marco adecuado a la labor técnica y la apoyen.

Referencias

[1] Frederick P. Brooks. *The Mythical Man-Month, Anniversary Edition: Essays on Software Engineering*. Addison-Wesley Pub Co; ISBN: 0201835959, Julio 1995.

[2] Jones, Capers. Assessment and Control of Software Risks. Englewood Cliffs, N. J.: Yourdon Press (1994).

[3] McConnell, Steve. *Desarrollo y gestión de proyectos informáticos*. McGraw-Hill, 1997.

[4] Software Formal Inspections Guidebook (NASA-GB-A302). Office of Safety and Mission Asurance, NASA (Agosto de 1993). Disponible en http://satc.gsfc.nasa.gov/fi.