Machine Translated by Google

# 2

# El mes del hombre mítico

Machine Translated by Google

# Restaurant Antoine

Fondé En 1840

**AVIS AU PUBLIC**

*Faire de la bonne cuisine demande un certain temps. Si on vous fait attendre, c'est pour mieux vous servir, et vous plaire.*

## ENTREES (SUITE)

Côtelettes d'agneau grillées 2.50
Côtelettes d'agneau aux champignons frais 2.75
Filet de boeuf aux champignons frais 4.75
Ris de veau à la financière 2.00
Filet de boeuf nature 3.75
Tournedos Médicis 3.25
Pigeonneaux sauce paradis 3.50
Tournedos sauce béarnaise 3.25
Entrecôte minute 2.75
Filet de boeuf béarnaise 4.00
Tripes à la mode de Caen (commander d'avance) 2.00
Entrecôte marchand de vin 4.00
Côtelettes d'agneau maison d'or 2.
Côtelettes d'agneau à la parisienne 2
Fois de volaille à la brochette 1.50
Tournedos nature 2.75
Filet de boeuf à la hawaïenne 4.00
Tournedos à la hawaïenne 3.25
Tournedos marchand de vin 3.25
Pigeonneaux grillés 3.00
Entrecôte nature 3.75
Châteaubriand (30 minutes)

## LÉGUMES

Epinards sauce crême .60
Broccoli sauce hollandaise .80
Pommes de terre au gratin .60
Haricots verts au berre .60
Chou-fleur au gratin .60
Asperges fraîches au beurre .90
Carottes à la crème .60
Pommes de terre soufflées
Petits pois à la française .75

## SALADES

Salade Antoine .60
Salade Mirabeau .75
Salade laitue au roquefort .80
Salade de laitue aux tomates .60
Salade de légumes .60
Salade d'anchois 1.00
Fonds d'artichauts Bayard
Salade de laitue aux oeufs .60
Tomate frappée à la Jules César .60
Salade de coeur de palmier 1.00
Salade aux pointes d'asperges .60
Avocat à la vinaigrette .60

## DESSERTS

Gâteau moka .50
Méringue glacée .60
Crêpes Suzette 1.25
Glace sauce chocolat .60
Fruits de saison à l'eau-de-vie .75
Omelette soufflée à la Jules César (2) 2.00
Omelette Alaska Antoine (2) 2.50
Cerises jubilé 1.25
Crêpes à la gelée .80
Crêpes nature .70
Omelette au rhum 1.10
Glace à la vanille .5
Fraises au kirsch
Pêche Melba

## FROMAGES

Roquefort .50
Camembert .50
Liederkranz .50
Gruyère .50
Fromage à la crême Philadelphie .50

## CAFÉ ET THÉ

Café .20
Café brulôt diabolique 1.00
Café au lait .20
Thé glacé .20
Thé .20
Demi-tasse

## EAUX MINERALES—BIERE—CIGARES—CIGARETTES

White Rock
Vichy
Cliquot Club
Bière locale
Canada Dry
Cigarettes

Roy b. Alciatore, Propriétaire

713-717 Rue St. louis Nouvelle Orléans, louisiane

Machine Translated by Google

# 2

# El mes del hombre mítico

La buena cocina finge el tiempo. Si te hacen esperar es para servirte mejor y complacerte.

MENÚ DEL RESTAURANTE ANTOINE. NUEVA ORLEANS

Machine Translated by Google

# 14 El mítico mes del hombre

Más proyectos de software han fracasado por falta de tiempo calendario que por todas las demás causas combinadas. ¿Por qué es tan común esta causa de desastre?

En primer lugar, nuestras técnicas de estimación están poco desarrolladas. Más seriamente, reflejan una suposición silenciosa y completamente falsa: es decir, que todo irá bien.

En segundo lugar, nuestras técnicas de estimación confunden falazmente esfuerzo con progreso, ocultando la suposición de que hombres y meses son intercambiables.

En tercer lugar, como no estamos seguros de nuestras estimaciones, los administradores de software a menudo carecen de la cortés terquedad del chef de Antoine.

Cuarto, el progreso del cronograma está mal monitoreado. Las técnicas probadas y rutinarias en otras disciplinas de la ingeniería se consideran innovaciones radicales en la ingeniería de software.

Quinto, cuando se reconoce un retraso en el cronograma, la respuesta natural (y tradicional) es agregar mano de obra. Como apagar un fuego con gasolina, esto empeora las cosas, mucho peores. Más fuego requiere más gasolina, y así comienza un ciclo regenerativo que termina en desastre.

El seguimiento del cronograma será objeto de un ensayo aparte. Consideremos otros aspectos del problema con más detalle.

## Optimismo

Todos los programadores son optimistas. Quizás esta hechicería moderna atraiga especialmente a aquellos que creen en finales felices y hadas madrinas. Quizás los cientos de frustraciones esenciales ahuyenten a todos menos a aquellos que habitualmente se centran en el objetivo final. Quizás se deba simplemente a que las computadoras son jóvenes, los programadores son más jóvenes y los jóvenes siempre son optimistas. Pero sea cual sea el proceso de selección, el resultado es indiscutible: "Esta vez seguramente se ejecutará" o "Acabo de encontrar el último error".

Así, la primera suposición falsa que subyace a la programación de sistemas es que todo irá bien, es decir, que cada tarea avanzará sólo durante el tiempo que "debería" durar.

Machine Translated by Google



La omnipresencia del optimismo entre los programadores merece más que un análisis involuntario. Dorothy Sayers, en su excelente libro The Mind of the Maker, divide la actividad creativa en tres etapas: la idea, la implementación y la interacción. Entonces, un libro, una computadora o un programa surgen primero como una construcción ideal, construida fuera del tiempo y el espacio, pero completa en la mente del autor. Se realiza en el tiempo y el espacio, mediante pluma, tinta y papel, o mediante alambre, silicio y ferrita. La creación se completa cuando alguien lee el libro, usa la computadora o ejecuta el programa, interactuando así con la mente del creador.

Esta descripción, que la señorita Sayers utiliza para iluminar no sólo la actividad creativa humana sino también la doctrina cristiana de la Trinidad, nos ayudará en nuestra tarea actual. Para los creadores humanos de las cosas, lo incompleto e inconsistente de nuestras ideas sólo se hace evidente durante la implementación. Así, la escritura, la experimentación y el "trabajo" son disciplinas esenciales para el teórico.

En muchas actividades creativas el medio de ejecución es intratable. La madera se parte; manchas de pinturas; Los circuitos eléctricos suenan. Estas limitaciones físicas del medio limitan las ideas que pueden expresarse y también crean dificultades inesperadas en su implementación.

La implementación, entonces, requiere tiempo y sudor, tanto por los medios físicos como por las insuficiencias de las ideas subyacentes. Tendemos a culpar a los medios físicos por la mayoría de nuestras dificultades de implementación; porque los medios no son "nuestros" en el sentido en que lo son las ideas, y nuestro orgullo colorea nuestro juicio.

La programación informática, sin embargo, crea con un medio sumamente manejable. El programador construye a partir de puro pensamiento: conceptos y representaciones muy flexibles de los mismos.
Debido a que el medio es manejable, esperamos pocas dificultades en su implementación; de ahí nuestro optimismo generalizado. Como nuestras ideas son defectuosas, tenemos errores; De ahí que nuestro optimismo sea injustificado.

En una sola tarea, la suposición de que todo irá bien tiene un efecto probabilístico sobre el cronograma. De hecho, podría ser como **planned,**

Machine Translated by Google



porque existe una distribución de probabilidad para el retraso que se producirá, y "sin retraso" tiene una probabilidad finita. Sin embargo, un gran esfuerzo de programación consta de muchas tareas, algunas encadenadas de un extremo a otro. La probabilidad de que todo salga bien se vuelve extremadamente pequeña.

## El'Mes-Hombre

El segundo modo de pensamiento falaz se expresa en la misma unidad de esfuerzo utilizada para estimar y programar: el mes-hombre. De hecho, el costo varía como producto del número de hombres por el número de meses. El progreso no. De ahí que el mes-hombre como unidad para medir el tamaño de un trabajo sea un mito peligroso y engañoso. Implica que los hombres y los meses son intercambiables.

Los hombres y los meses son bienes intercambiables sólo cuando una tarea puede dividirse entre muchos trabajadores sin comunicación entre ellos (figura 2.1). Esto es cierto cuando se cosecha trigo o se recoge algodón; ni siquiera es aproximadamente cierto en el caso de la programación de sistemas.

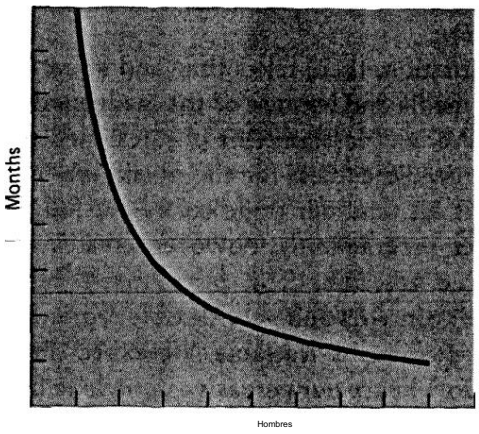


Fig. 2.1 Tiempo versus número de trabajadores: tarea perfectamente divisible

Machine Translated by Google



Cuando una tarea no se puede dividir debido a restricciones secuenciales, la aplicación de más esfuerzo no tiene ningún efecto en el cronograma (figura 2.2). Para tener un hijo se necesitan nueve meses, sea cual sea. cuantas mujeres se asignan. Muchas tareas de software tienen esto Característica debido a la naturaleza secuencial de la depuración.

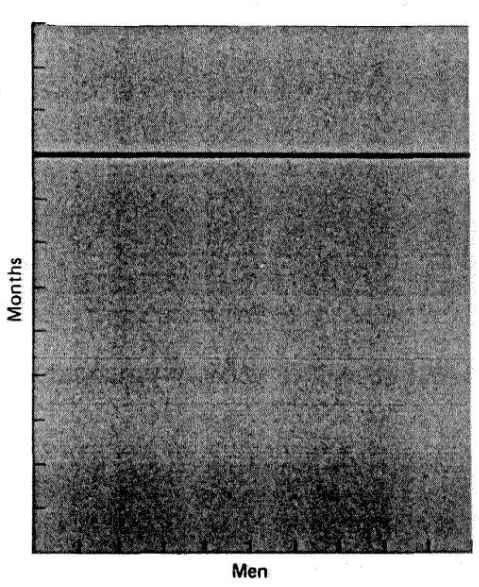


Fig. 2.2 Tiempo versus número de trabajadores: tarea no divisible

En tareas que pueden dividirse pero que requieren comunicación entre las subtareas, el esfuerzo de comunicación debe ser añadido a la cantidad de trabajo a realizar. Por lo tanto lo mejor que se puede hacer es algo peor que un comercio equitativo de hombres por meses(Figura 2.3).

Machine Translated by Google



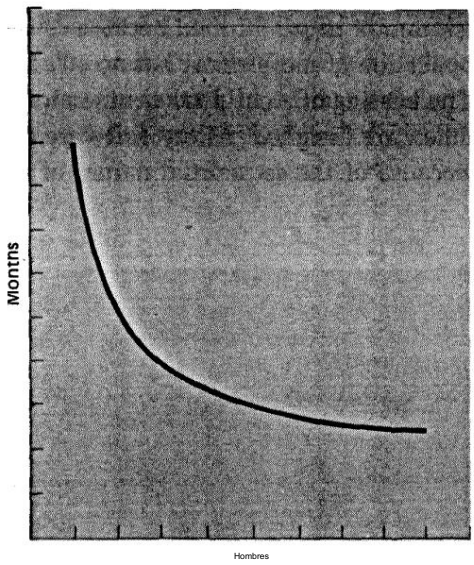


Fig. 2.3 Tiempo versus número de trabajadores: tarea divisible que requiere comunicación

La carga añadida de la comunicación se compone de dos partes, formación e intercomunicación. Cada trabajador debe estar capacitado en la tecnología, los objetivos del esfuerzo, la estrategia general y el plan de trabajo. Esta formación no puede fraccionarse, por lo que esta parte del esfuerzo añadido varía linealmente con el número de trabajadores. [1]

La intercomunicación es peor. Si cada parte de la tarea debe coordinarse por separado entre sí, el esfuerzo aumenta como n(nI)/2. Tres trabajadores requieren tres veces más intercomunicación por parejas que dos; cuatro requieren seis veces más que dos.

Si además es necesario que haya conferencias de tres, cuatro, etc., trabajadores para resolver las cosas de forma conjunta, la cosa empeora aún más. El esfuerzo adicional de comunicar puede contrarrestar completamente la división de la tarea original y llevarnos a la situación de la figura 2.4.

Machine Translated by Google



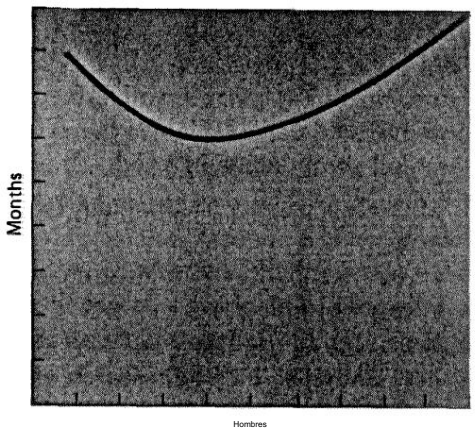


Fig. 2.4 Tiempo versus número de trabajadores: tarea con interrelaciones complejas

Dado que la construcción de software es inherentemente un esfuerzo de sistemas (un ejercicio de interrelaciones complejas), el esfuerzo de comunicación es grande y rápidamente domina la disminución del tiempo de las tareas individuales provocada por la partición. Agregar más hombres alarga, no acorta, el cronograma.

## Prueba de sistemas

Ninguna parte del programa se ve tan afectada por las restricciones secuenciales como la depuración de componentes y las pruebas del sistema. Además, el tiempo necesario depende del número y la sutileza de los errores encontrados. En teoría, este número debería ser cero.
Debido al optimismo, normalmente esperamos que el número de errores sea

Machine Translated by Google



más pequeño de lo que resulta ser. Por lo tanto, las pruebas suelen ser la parte de la programación que menos se programa.

Durante algunos años he utilizado con éxito la siguiente regla general para programar una tarea de software:

- /3 planificación
- /6 codificación
- / Prueba de 4 componentes y prueba temprana del sistema
- /4 prueba del sistema, todos los componentes en mano.

Esto difiere de la programación convencional en varios aspectos importantes:

1. La fracción dedicada a la planificación es mayor de lo normal. Aun así, apenas es suficiente para producir una especificación detallada y sólida, y no es suficiente para incluir la investigación o exploración de técnicas totalmente nuevas.

2. La mitad del cronograma dedicado a la depuración de los completados.
   El código es mucho más grande de lo normal.
3. La parte que es fácil de estimar, es decir, la codificación, recibe sólo una sexta parte del cronograma.

Al examinar proyectos programados convencionalmente, he encontrado que pocos permitieron la mitad del cronograma proyectado para pruebas, pero que la mayoría efectivamente gastó la mitad del cronograma real para ese propósito. Muchos de estos se cumplieron según lo previsto hasta y excepto en
pruebas del sistema. [2]

En particular, no dejar suficiente tiempo para probar el sistema es particularmente desastroso. Dado que el retraso se produce al final del cronograma, nadie se da cuenta del problema del cronograma hasta casi la fecha de entrega. Las malas noticias, llegadas tarde y sin previo aviso, inquietan a los clientes y a los directivos.

Además, un retraso en este punto tiene repercusiones financieras y psicológicas inusualmente graves. El proyecto cuenta con personal completo y el costo por día es máximo. Lo que es más grave, el software debe respaldar otros esfuerzos comerciales (envío de computadoras, operación de nuevas instalaciones, etc.) y los costos secundarios de retrasarlos son muy altos, ya que casi es hora de enviar el software.

Machine Translated by Google



De hecho, estos costos secundarios pueden superar con creces a todos los demás. Por lo tanto, es muy importante dejar suficiente tiempo para probar el sistema en el programa original.

## Estimación despiadada

Obsérvese que para el programador, como para el chef, la urgencia del cliente puede regir la finalización programada de la tarea, pero no puede regir la finalización real. Una tortilla, prometida en dos minutos, puede parecer que va bien. Pero cuando no se ha endurecido en dos minutos, el cliente tiene dos opciones: esperar o comérselo crudo. Los clientes de software han tenido las mismas opciones.

El cocinero tiene otra opción; él puede subir la temperatura. El resultado suele ser una tortilla que nada puede salvar: quemada en una parte y cruda en la otra.

Ahora bien, no creo que los administradores de software tengan menos coraje y firmeza inherentes que los chefs, ni que otros administradores de ingeniería. Pero la programación falsa para coincidir con la fecha deseada por el cliente es mucho más común en nuestra disciplina que en otras partes de la ingeniería. Es muy difícil hacer una defensa vigorosa, plausible y arriesgada para el empleo de una estimación que no se deriva de ningún método cuantitativo, está respaldada por pocos datos y está certificada principalmente por las corazonadas de los gerentes.

Es evidente que se necesitan dos soluciones. Necesitamos desarrollar y publicar cifras de productividad, cifras de incidencia de errores, reglas de estimación, etc. Toda la profesión sólo puede beneficiarse compartiendo esos datos.

Hasta que las estimaciones tengan una base más sólida, los administradores individuales necesitarán endurecer su columna vertebral y defender sus estimaciones.

con la seguridad de que sus malas corazonadas son mejores que las estimaciones derivadas de los deseos.

## Desastre del programa regenerativo

¿Qué se hace cuando un proyecto de software esencial está retrasado? Agregue mano de obra, naturalmente. Como las Figs. 2.1 a 2.4 sugieren que esto puede ayudar o no.

Machine Translated by Google



Consideremos un ejemplo. [3] Supongamos que una tarea se estima en 12 meses-hombre y se asigna a tres hombres durante cuatro meses, y que hay hitos mensurables A, B, C, D, que están programados para caer al final de cada mes (Fig. 2.5).

Ahora supongamos que no se alcanza el primer hito hasta que hayan transcurrido dos meses (figura 2.6). ¿Cuáles son las alternativas que enfrenta el gerente?

1. Asuma que la tarea debe realizarse a tiempo. Supongamos que sólo se estimó erróneamente la primera parte de la tarea, por lo que la figura 2.6 cuenta la historia con precisión. Entonces quedan 9 meses-hombre de esfuerzo y dos meses, por lo que se necesitarán 4V£ hombres. Suma 2 hombres a los 3 asignados.
2. Asuma que la tarea debe realizarse a tiempo. Supongamos que el La estimación total fue uniformemente baja, de modo que la figura 2.7 realmente describe la situación. Entonces quedan 18 meses-hombre de esfuerzo y dos meses, por lo que se necesitarán 9 hombres. Suma 6 hombres a los 3 asignados.

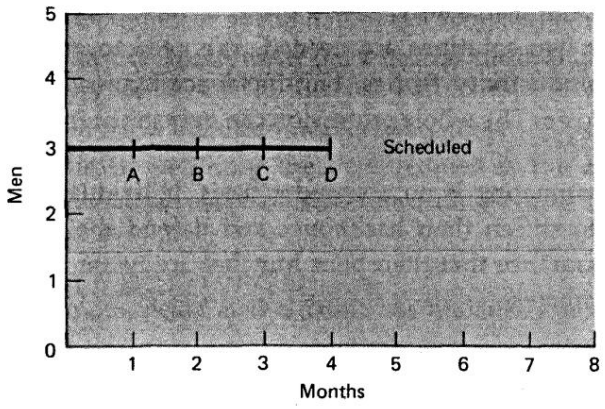


Figura 2.5

Machine Translated by Google



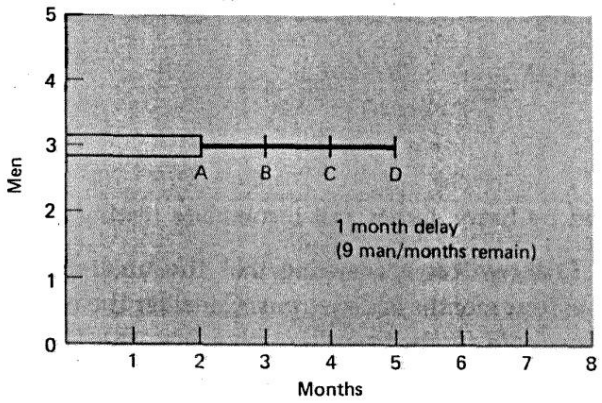


Figura 2,6

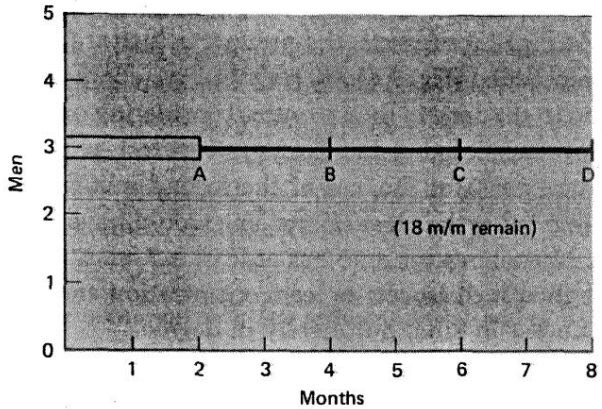


Figura 2.7

Machine Translated by Google



3. Reprogramar. Me gusta el consejo de P. Fagg, un experimentado ingeniero de hardware: "No cometas pequeños deslices". Es decir, deje suficiente tiempo en el nuevo cronograma para garantizar que el trabajo pueda realizarse cuidadosa y exhaustivamente, y que no sea necesario reprogramarlo nuevamente.
4. Recorta la tarea. En la práctica, esto tiende a suceder de todos modos, una vez que el equipo observa un retraso en el cronograma. Cuando los costos secundarios del retraso son muy altos, esta es la única acción factible. Las únicas alternativas del gerente son recortarla formal y cuidadosamente, reprogramarla o observar cómo la tarea se recorta silenciosamente mediante un diseño apresurado y pruebas incompletas.

En los dos primeros casos, insistir en que la tarea inalterada se complete en cuatro meses es desastroso. Consideremos, por ejemplo, los efectos regenerativos de la primera alternativa (figura 2.8). Los dos nuevos hombres, por competentes y rápidos que sean reclutados, necesitarán que uno de los hombres experimentados los capacite en la tarea. Si esto lleva un mes, se habrán dedicado 3 meses-hombre a trabajos que no están en la estimación original. Además, la tarea, originalmente dividida en tres partes, debe dividirse en cinco partes; por lo tanto, se perderá parte del trabajo ya realizado y será necesario prolongar las pruebas del sistema.
Así, al final del tercer mes, quedan sustancialmente más de 7 meses-hombre de esfuerzo y 5 personas capacitadas y un mes disponibles. Como sugiere la figura 2.8, el producto llega tan tarde como si no se hubiera añadido nadie (figura 2.6).

Para esperar terminar en cuatro meses, considerando sólo el tiempo de entrenamiento y no la repartición y las pruebas de sistemas adicionales, sería necesario agregar 4 hombres, no 2, al final del segundo mes. Para cubrir los efectos de la repartición y de las pruebas del sistema, habría que añadir otros hombres más. Ahora, sin embargo, se tiene al menos un equipo de 7 hombres, no de 3; por lo tanto, aspectos como la organización del equipo y la división de tareas son diferentes en tipo, no sólo en grado.

Observe que al final del tercer mes las cosas se ven muy negras. A pesar de todo, el hito del 1 de marzo no se ha alcanzado

Machine Translated by Google



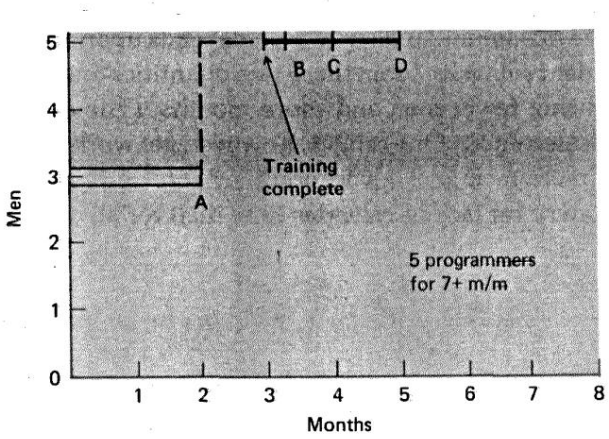


Figura 2.8

el esfuerzo directivo. La tentación de repetir el ciclo es muy fuerte, añadiendo aún más mano de obra. Ahí está la locura.

Lo anterior supuso que sólo se desestimó el primer hito. Si en marzo 1 se hace la suposición conservadora de que todo el cronograma era optimista, como lo muestra la figura 2.7, se quieren agregar 6 hombres solo a la tarea original. El cálculo de los efectos del entrenamiento, la repartición y las pruebas del sistema se deja como ejercicio para el lector. Sin lugar a dudas, el desastre regenerativo arrojará un producto peor, más adelante, que el que daría la reprogramación con los tres hombres originales, sin aumentar.

Simplificando demasiado escandalosamente, afirmamos la Ley de Brooks:

Agregar mano de obra a un proyecto de software tardío lo hace más tardío.

Se trata entonces de la desmitificación del mes-hombre. El número de meses de un proyecto depende de sus consecuencias secuenciales.

Machine Translated by Google



tensiones. El número máximo de hombres depende del número de subtareas independientes. De estas dos cantidades se pueden derivar cronogramas utilizando menos hombres y más meses. (El único riesgo es la obsolescencia del producto). Sin embargo, no se pueden conseguir cronogramas viables utilizando más hombres y menos meses. Más proyectos de software han fracasado por falta de tiempo calendario que por todas las demás causas combinadas.