# Introducción

Idealmente, un documento *ágil* es apenas bueno o apenas suficiente para la situación que tenemos a mano. La documentación es una parte importante de los proyectos de desarrollo ágiles, pero a diferencia de los tradicionalistas que a menudo ven la documentación como una estrategia de reducción de riesgos, los agilistas comúnmente ven la documentación como una estrategia que aumenta el riesgo total del proyecto y, por lo tanto, se esfuerzan en ser tan eficientes como sea posible cuando se trata de documentación. Los agilistas escriben documentación cuando esa es la mejor manera de lograr las metas relevantes, pero a veces se ha probado que es mejor cumplir las metas que escribir documentación estática. Este artículo resume las “buenas prácticas” comunes que los *agilistas* han adoptado con respecto a la documentación.

# 1. Buenas Prácticas para escribir Documentación Efectiva

Las siguientes prácticas nos ayudarán a mejorar nuestro enfoque para escribir documentación:

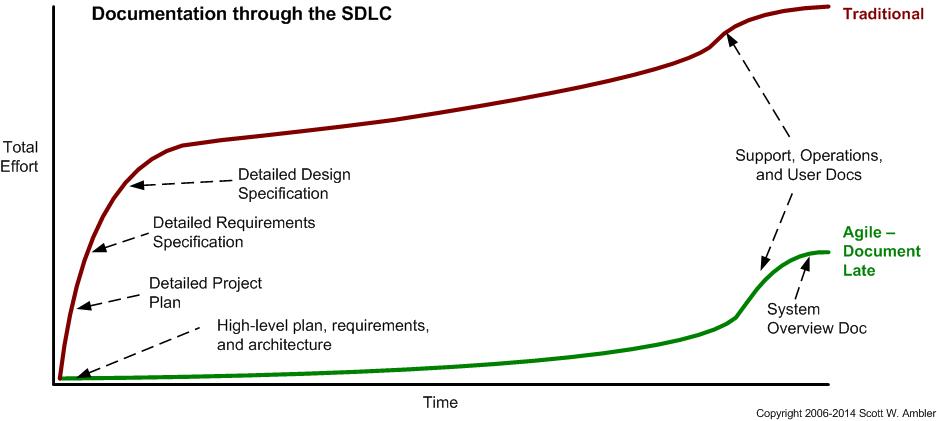
1. Prefiera especificaciones ejecutables en vez de documentos estáticos.
2. Documente conceptos estables, no ideas especulativas.
3. Genere documentación del sistema

## 1.1 Prefiera especificaciones ejecutables en vez de documentos estáticos

La mayoría de la información capturada en los documentos de especificaciones tradicionales, tales como la especificación de requerimientos, o especificaciones de diseño, puede ser capturada como “especificaciones ejecutables” en forma de pruebas. Cuando tomas un enfoque TDD (Desarrollo Manejado por Pruebas), escribes efectivamente especificaciones detalladas sobre una base JIT (Just in time, o, sobre la hora). Con TDD escribes una prueba, ya sea a nivel de aceptación de cliente o nivel de desarrollador antes de escribir la suficiente funcionalidad para cumplir esa prueba. Las pruebas son usadas con dos propósitos: especifican los requerimientos/arquitectura/diseño, y validan tu trabajo. Este es un ejemplo de la práctica Información de una Única Fuente.

## 1.2 Documente Conceptos Estables, no Ideas Especulativas

Como se puede ver en la figura 1, la estrategia ágil es posponer la creación de todos los documentos tanto como sea posible, crearlos justo antes de que los necesitemos vía una práctica llamada “documento atrasado”. Por ejemplo, los resúmenes del sistema se escriben mejor al final del desarrollo de una entrega porque conocemos lo que realmente construimos. De la misma manera, la mayoría de las documentaciones de usuario y soporte es mejor escribirlas al final del ciclo de vida del proyecto. Sin embargo, esto no significa que toda la documentación deba escribirse al final. Igual querremos tomar notas para esos tipos de documentos a través del desarrollo para que no perdamos información crítica. Estas notas puede ser solo información puntual y puede que no haya necesidad de “pulir” los documentos hasta estar bien cerca de entregarlos.



**Figura 1.** Documentación a través del ciclo de vida de desarrollo de software.

Al esperar documentar la información hasta que se haya estabilizado, reducimos tanto el costo como el riesgo asociado con la documentación. El costo se reduce debido a que no tenemos que perder tiempo documentando información que cambia, lo cual, a la vez, nos motiva a actualizar la documentación. El riesgo se reduce debido a que hay mucho menos posibilidades de que la documentación existente esté desactualizada. Si escribimos documentación que contiene información que aún no ha sido estabilizada, entonces nos arriesgamos a tener que volver a trabajar la documentación cuando ésta cambie. En otras palabras, no queremos invertir mucho tiempo documentando ideas especulativas tales como los requerimientos o diseños de las etapas tempranas del proyecto. En lugar de eso, mejor esperar al final del ciclo de vida, cuando la información se ha estabilizado y cuando sabemos cuál es la información útil. La desventaja es que nuestros esfuerzos de documentación podrían estar unas pocas iteraciones atrasadas con respecto a los esfuerzos de desarrollo de software.

Una versión extrema de esta práctica es esperar hasta que hayamos terminado y recién entonces escribir la documentación. La ventaja principal es que estás escribiendo sobre algo conocido y estable (la entrega del software que acabamos de construir). Pero, hay varias desventajas de este enfoque:

* Probablemente hayas olvidado algunas de las razones detrás de las decisiones que hiciste, lo cual, claramente es un problema si esto es importante de documentar.
* Podrías no tener disponibles a las personas que escriben la documentación porque pueden haberse movido a otros proyectos.
* Podrías no tener financiamiento para hacer el trabajo.
* Podría dejar de existir el deseo de escribir la documentación.

## 1.3 Genere Documentación del Sistema

Las herramientas de modelado de software modernas pueden hacer ingeniería inversa al código existente y presentar una gran variedad de vistas sobre esa información. En resumen, podemos ahorrar mucho dinero generando la mayoría de la documentación del sistema que necesitamos usando las herramientas disponibles.

# 2. Buenas Prácticas para la Simplificación

Las siguientes prácticas nos ayudaran a simplificar la documentación que escribimos:

1. Mantenga la documentación lo suficientemente simple, pero no tan simple.
2. Escriba el mínimo de documentos con poca redundancia o repeticiones.
3. Ponga la información en el lugar más apropiado.
4. Muestre la información públicamente.

## 2.1 Mantenga la documentación lo suficientemente simple, pero no tan simple

La documentación exhaustiva no asegura el éxito de un proyecto, de hecho, aumenta las posibilidades que falle. La documentación debería ser concisa: generalmente los resúmenes/mapas son preferidos por sobre la documentación detallada. Siga las prácticas del Modelamiento Ágil (AM – Agile Modeling): Use las Herramientas más Simples, Cree Contenido Simple, y Represente Modelos Simplementa cuando cree la documentación. La mejor documentación es la más simple que logra que se haga el trabajo. No cree documentos de 50 páginas que pueden ser hechas en 5. No cree un documento de 5 páginas cuando se puede hacer con 5 viñetas. No cree diagramas elaborados y complejos cuando se puede hacer con un bosquejo. No repita información en ninguna parte cuando se pueda usar una referencia. Escriba en forma puntual. Documente solo lo suficiente para entregar un contexto útil. Empiece con un documento que es lo suficientemente minimalista para las necesidades de los clientes y vaya aumentándolo a medida que se necesite. Para determinar cuál es la verdadera cantidad mínima de documentación requerida por los clientes, explore activamente (pregunte) en cómo pretenden usar la documentación y por qué la quieren usar de esa manera.

El intercambio básico acá es la “seguridad” de tener el documento contra la confianza en su contenido. ¿Qué prefiere tener, un documento del sistema de 500 páginas que probablemente tenga una cantidad significativa de errores en él pero harto detalle, o un resumen de alto nivel de 10 páginas? El documento grande probablemente tenga toda la información de sistema que necesite para la mantención y mejora del mismo, pero, ¿confiaría en la información que hay en él? El documento más corto probablemente no tenga la información detalla que necesite, pero le entregará un mapa de donde puede indagar en el código fuente, u otros documentos, por detalles. Usted quizás confíe más en este documento porque es más corto, en el peor de los casos, se podría actualizar fácilmente o simplemente reescribirse si encuentra que es groseramente inexacto, debido a que está relacionado con conceptos de alto nivel tales como la arquitectura del sistema, la cual cambia más lentamente que la minuta detallada contenida en el documento más grande. Es importante entender que no estamos diciendo que el documento más extenso es automáticamente de menor calidad que el más corto, pero sí que es mejor percibirlo así hasta que se pruebe lo contrario.

## 2.2 Escriba el mínimo de documentos con poca redundancia o duplicados

Debemos esforzarnos por viajar lo más ligeros como sea posible, escribiendo la suficiente documentación para la situación a mano, la cual es apenas suficiente para lograr su propósito. Una manera de lograr esto, es escribir documentos más grandes a partir de los más pequeños. Por ejemplo, documentar la arquitectura del sistema en una página HTML, el diseño de pantallas en otra página (o wiki), y así sucesivamente, para finalmente unir todas las páginas con una que tenga un índice con enlaces a cada tema (tabla de contenidos). La ventaja de esto es que la información fue definida en un solo lugar, de tal manera que no hay opción para que hayan duplicados.

## 2.3 Ponga la información en el lugar más apropiado

¿Dónde alguien querría un pedazo de documentación? ¿Es esta decisión de diseño, mejor documentarla en el código, agregarla como nota en el diagrama, o mejor ponerla en un documento externo? ¿Es un requerimiento específico, mejor capturado como parte de un caso de uso, una especificación de regla de negocio, o como una prueba ejecutable? La respuesta a este tipo de preguntas debería estar guiada por las necesidades del cliente que necesita esa información. También es guiada por nuestro deseo de seguir el principio de Trabajo de Calidad, deberíamos esforzarnos por registrar la información una vez que ésta mejora más nuestro trabajo (por ejemplo, Información de una Única Fuente – SingleSource Information). También deberíamos considerar temas como la indexación, linkeo y accesibilidad cuando se está escribiendo documentación porque no siempre sabemos quién será eventualmente el consumidor de la información.

## 2.4 Muestre la información públicamente

Cuando los modelos son mostrados públicamente – en una pizarra, o un sitio web interno – estamos promocionando la transferencia de información y así la comunicación a través de la aplicación de lo que ***Cockburn*** se refiere como “radiador de información”. Mientras mayor sea la comunicación de nuestro proyecto, menor será la necesidad de documentación detallada porque las personas ya sabrán lo que estamos haciendo. Dicho esto, no olvidemos indicar el estado de nuestro trabajo para que la gente pueda ponerlo en contexto – tratarán a un modelo a un bosquejo de manera muy diferente a si está debidamente aprobado y con línea base para la entrega actual de software.

# 3. Buenas Prácticas para Determinar Qué Documentar

Las siguientes prácticas nos ayudarán a determinar qué documentar:

1. Documente con un propósito.
2. Enfóquese en las necesidades de los clientes actuales del documento.
3. El cliente determina la suficiencia.

## 3.1 Documente con un propósito

Se debe documentar solo si satisface una meta clara, importante e inmediata para el total de los esfuerzos del proyecto. No olvide que este propósito debe ser a corto o a largo plazo, puede apoyar directamente a los esfuerzos de desarrollo o puede que no. También recuerde que cada sistema tiene sus propias y únicas necesidades de documentación, es tamaño en particular no sirve para todo – una implicación de que no podrá seguir un “proceso repetitivo” y entregar el mismo conjunto de documentación (o plantillas) en cada proyecto, al menos no si está interesado en ser realmente efectivo.

## 3.2 Enfóquese en las necesidades de los clientes actuales del documento

Usted quiere trabajar de cerca con la audiencia del documento de tal manera de saber lo que ellos realmente requieren de ella. Para hacer esto, hay que identificar a los potenciales usuarios o clientes de la documentación, que creen que necesitan y luego negociar el subconjunto mínimo que realmente necesitan. Para descubrir lo que ellos creen que requieren, hay que preguntarles acerca de qué hacen, cómo lo hacen y cómo quieren trabajar con la documentación que se les entregará. Entendiendo las necesidades de los usuarios podremos entregar la documentación necesaria y suficiente, y entregarla donde ellos realmente la necesitan encontrar – no importa que tan bien esté escrito un documento si nadie sabe que existe.

## 3.3 El cliente determina la suficiencia

En una institución financiera de Canadá tenían una política que decía que no se podía hacer la transición de un sistema a alguien a menos que esta persona estuviera dispuesta a aceptarlo. Ellos inspeccionarían el código y los artefactos de apoyo y si sentían que los artefactos no estaban a la par de lo que necesitaban, había que mejorarlos una y otra vez. A veces, tendremos que trabajar en mejorar los artefactos en conjunto, otras veces no. Esta práctica provee una compuerta de calidad justa y efectiva entre los desarrolladores y los usuarios del trabajo. Como escritores de documentación, es nuestro trabajo asegurarnos de que tenga un verdadero significado y entregue valor, el rol del usuario es validar que lo hemos hecho así.

# 4. Buenas Prácticas para Determinar Cuándo Documentar

Las siguientes prácticas nos ayudarán a determinar cuándo documentar:

1. [Iterar, iterar, iterar](http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm#Iterate)
2. Encuentre mejores maneras de comunicarse
3. Empiece con modelos que realmente estén actualizados
4. Actualice solo cuando duele

## 4.1 [Iterar, iterar, iterar](http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm#Iterate)

Idealmente se debe crear documentación a través de todo el ciclo de vida del software, cuando tiene más sentido, aunque eso normalmente es cerca del final del ciclo. Cuando escribimos documentación, debemos tomar un enfoque evolutivo (iterativo e incremental) para su desarrollo para que tengamos retroalimentación de su valor actual. En otras palabras, escribir un poco, mostrárselo a alguien, obtener la retroalimentación, actuar sobre esa retroalimentación y luego iterar. Los mejores documentos son escritos iterativamente, no todo de una vez. Un enfoque iterativo nos permite enfocarnos en lo que la audiencia realmente necesita.

## 4.2 Encuentre mejores maneras de comunicarse

Highsmith cree que el problema principal con la comunicación es el entendimiento, no la documentación, por lo tanto, no se debe sobrevalorar el valor de la documentación. Documentación bien escrita apoya a la memoria organizacional efectivamente, pero es una manera pobre de comunicarse durante el proyecto. Nuestra meta es asegurar que los desarrolladores de mantención entienden cómo trabaja el sistema para que pueda evolucionar en el tiempo, no producir una montaña de documentación que pueda o no pueda usarse. Nuestra meta es asegurar que nuestros usuarios trabajen con nuestro sistema efectivamente, no que tengan un bonito sistema de ayuda disponible para ellos. Nuestra meta es permitir nuestro apoyo al staff de soporte y operaciones, no taparlos con papel.

La documentación apoya la transferencia de conocimiento, pero solo es una de las muchas opciones disponibles para hacerlo y como se muestra en la figura 2, a menudo no es la mejor opción. La implicancia de este diagrama es que deberíamos elegir el mejor medio de comunicación disponible dada la situación. Las opciones de documentación, en particular, las “especificaciones en papel” son la opción menos deseable, no la más deseable. Las conversaciones con los stakeholders, tenerlos activamente involucrados con el desarrollo, estar disponible para trabajar en cualquier problema con ellos, muchas veces va más lejos que la mejor de las documentaciones. La documentación se convierte en una mejor opción para nosotros cuando es mayor la distancia (física o temporal) entre los individuos que nos estamos comunicando.

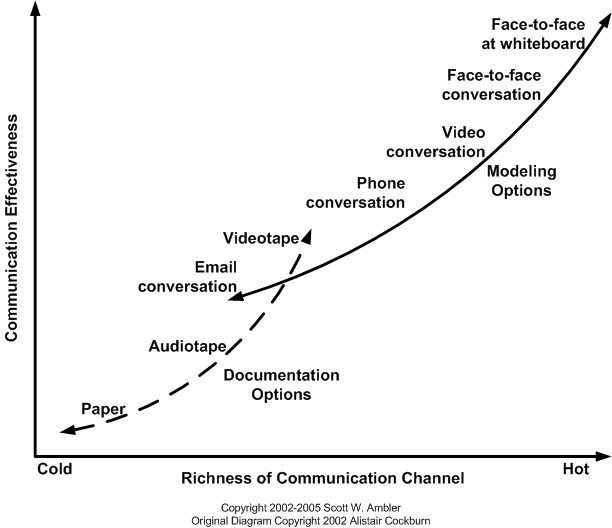
[](http://agilemodeling.com/essays/communication.htm)

Figura 2. Comparando la efectividad de los distintos modos de comunicación.

Es importante reconocer que cuando especificamos detalles de cómo se hace algo y luego le pasamos el documento a alguien para que lo siga, hemos logrado efectivamente quitarle toda la gracia a la actividad para esa persona porque hemos hecho el pensamiento por ellos. ¿Cuál sería la motivación para un trabajador intelectual, tal como un profesional TI, de seguir instrucciones detalladas? Nunca olvidemos que los desarrolladores rara vez confían en la documentación, particularmente en la documentación detallada porque usualmente no está sincronizada con el código, así que minimicemos la cantidad de documentación que entregamos.

## 4.3 Empiece con modelos que realmente estén actualizados

Si hemos elegido mantener nuestro diagrama de despliegue UML, nuestro diagrama de flujo de interfaces y nuestro diagrama físico de datos al día a través del desarrollo, entonces esa es una buena señal de que los modelos de valor deberían ser la base de nuestra documentación. Los modelos desactualizados no dan ningún valor al proyecto.

## 4.4 Actualice solo cuando duele

En este aspecto, los documentos son solo como modelos, mi recomendación es seguir la práctica “Actualice solo cuando duele”. Los documentos ágiles, al igual que modelos ágiles, deberían ser solo suficientemente buenos. Muchas veces un documento puede estar desactualizado y no importa mucho. Por ejemplo, hay veces que uno trabaja con un manual de referencia de una versión más actual (o más antigua) que el software que se está utilizando, no es la situación ideal, pero no hay tanta diferencia como para que duela debido a que los manuales son relativamente buenos. Serían suficientes los manuales un poco más desactualizados o actualizados (de más de 1 o 2 años de antigüedad)? Probablemente no porque puede que hayan los cambios suficientes para que haya una pérdida de productividad al leer documentación demasiado atrasada y el costo podría ser más grande que el de conseguir manuales al día.

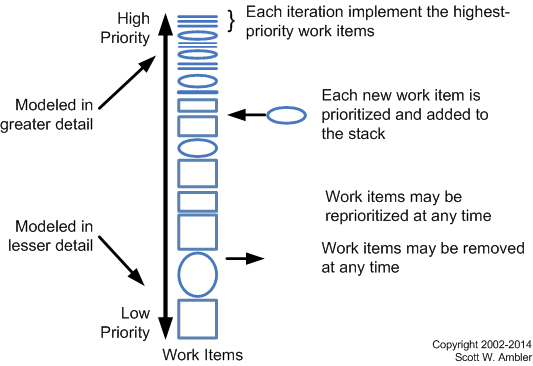
# 5. Buenas Prácticas Generales

Las siguientes prácticas nos ayudarán a mejorar nuestros esfuerzos de documentación:

1. Trate la documentación como a un requerimiento
2. Requiera o solicite personas que justifiquen los requerimientos de documentación.
3. Reconozca que necesita la documentación.
4. Consiga a alguien con experiencia en escribir documentación.

## 5.1 Trate la documentación como a un requerimiento

Una filosofía ágil común es tratar la documentación como a cualquier otro requerimiento: debe ser estimado, priorizado y puesto en el stack de ítems de trabajo (ver figura 3) junto con todos los otros ítems de trabajo que debemos realizar. La necesidad de escribir un documento claramente es un requerimiento tal como lo es la necesidad de escribir una característica de la aplicación. Cualquier inversión que hagamos en documentación es una inversión que podríamos haber hecho en una nueva funcionalidad y viceversa, así que alguien debe hacer una decisión a conciencia de cuánto se debe invertir. Al tratar la documentación como un requerimiento hacemos de su creación una decisión visible y explícita para que consideren los stakeholders. Fundamentalmente, la inversión en documentación es una decisión de negocio, no una técnica: no deberíamos crear documentación porque el proceso lo dice sino porque los stakeholders lo dicen.

[](http://agilemodeling.com/essays/changeManagement.htm)

**Figura 3.** Administrando el trabajo como un stack priorizado.

## 5.2 Requiera o solicite personas que justifiquen los requerimientos de documentación

¿La persona que pidió la documentación, sabe lo que está pidiendo y por qué lo está pidiendo, o está pidiéndolo porque se les dijo que lo hicieran? ¿Están los usuarios pidiendo documentación porque fueron plantados por los desarrolladores y ahora piden todo para poder obtener algo? ¿El solicitante entiende las concesiones que se deben hacer para documentar y que la documentación tiene un costo? La experiencia dice que cuando exploramos los temas de documentación con los usuarios, rápidamente se descubre que lo están haciendo porque desconfían de nosotros, ellos a menudo no entienden las implicancias de lo que están pidiendo y a menudo no saben que hay alternativas (por ejemplo, menos documentación). Una buena pregunta a realizar es para qué pretenden usar la documentación y cómo realmente usan la documentación. Cuando hacemos esto, a menudo descubrimos que los usuarios no usan toda la documentación, que más que nada la piden como un seguro o respaldo por escrito. Hay muchas maneras mejores de manejar estos temores que la documentación. Algunos temas importantes:

1. Debemos entender el costo total de propiedad del documento (TCO), y esforzarnos en maximizar el ROI de los stakeholders para proporcionar el mejor valor posible a nuestra organización.
2. Alguien debe elegir explícitamente hacer la inversion de documentar.
3. El beneficio de tener documentación debe ser mayor que el costo de crearla y mantenerla.
4. Preguntarse si NECESITAMOS la documentación o si la queremos.

|  |  |
| --- | --- |
|  |  |

## 5.3 Reconozca que necesita la documentación

Algunas observaciones importantes:

1. **La Documentación es tan parte del sistema como el código fuente**. Además del software en el que trabajamos, probablemente también querremos entregar mínimamente manuales de usuario, documentación de soporte, de operaciones y resúmenes del sistema.
2. **La meta principal de tu equipo es desarrollar software, su meta secundaria es permitir nuestro próximo esfuerzo**. Sí, la construcción de software que funcione de alta calidad que satisfice las necesidades de los clientes o usuarios es importante, pero asegurar que la gente que viene después de ti, pueda mantenerlo, mejorarlo, operarlo y darle soporte también es importante.
3. **La Documentación aún sirve a un propósito**. En particular, queremos capturar información de alto nivel en la documentación pero no detalles. Igual querremos capturar información importante permanentemente, y a veces las regulaciones requieren ciertos niveles de documentación.
4. **Los Agilistas de hecho están escribiendo documentación**. La encuesta DDJ's 2008 Modeling and Documentation enconró que los equipos ágiles fueron muy similares a los equipos tradicionales al escribir documentación entregable, tal como manuales de usuario, documentación de operaciones, etc. La figura 4 resume los resultados de la encuesta pertinente a documentación entregable. Lo importante es que esta encuesta debiera a desmitificar lo que la gente piensa cuando se refiere al desarrollo y documentación de software ágil.

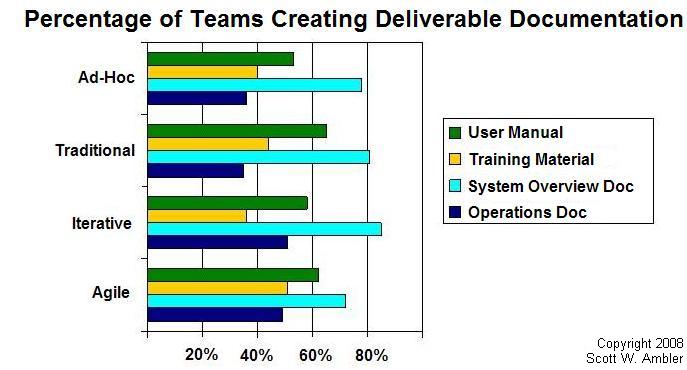
[](http://www.ambysoft.com/surveys/modelingDocumentation2008.html)

Figura 4. Creación de documentación entregable.

## 5.4 Consiga a alguien con experiencia en escribir documentación

Los escritores técnicos traen muchas ventajas en lo que respecta a escribir documentación porque ellos saben cómo organizar y presentar la información de manera efectiva. ¿No tiene acceso a un escritor técnico? Aquí hay algunas estrategias:

* Considere leer y seguir algún curso corto de Escritura Técnica para saber los fundamentos.
* Trate de escribir documentación con un compañero, tal como es significativa la programación en pares, hay un valor similar en la documentación en pares.
* Mantenga la documentación compartida para que varias personas puedan trabajar sobre ella.
* Compre software text-to-speech que le permita escuchar lo que ha escrito, es una gran manera de descubrir los pasajes que están pobremente escritos.