11. Usabilidad

Cualquier tonto puede hacer algo complejo; Se necesita un genio para hacer algo simple.
-Albert Einstein

La usabilidad se relaciona con lo fácil que es para el usuario realizar una tarea deseada y el tipo de asistencia al usuario que proporciona el sistema. A lo largo de los años, un enfoque en la facilidad de uso se ha revelado como una de las formas más económicas y sencillas de mejorar la calidad de un sistema (o, más precisamente, la percepción de calidad del usuario).

La usabilidad comprende las siguientes áreas:

• *Funciones del sistema de aprendizaje* . Si el usuario no está familiarizado con un sistema en particular o con un aspecto en particular, ¿qué puede hacer el sistema para facilitar la tarea de aprender? Esto podría incluir proporcionar características de ayuda.

• *Uso eficiente de un sistema* . ¿Qué puede hacer el sistema para que el usuario sea más eficiente en su operación? Esto podría incluir la capacidad para que el usuario redirija el sistema después de emitir un comando. Por ejemplo, el usuario puede desear suspender una tarea, realizar varias operaciones y luego reanudar esa tarea.

• *Minimizar el impacto de los errores* . ¿Qué puede hacer el sistema para que un error de usuario tenga un impacto mínimo? Por ejemplo, el usuario puede desear cancelar un comando emitido incorrectamente.

• *Adaptar el sistema a las necesidades del usuario.* ¿Cómo puede el usuario (o el propio sistema) adaptarse para facilitar la tarea del usuario? Por ejemplo, el sistema puede completar automáticamente las URL basadas en las entradas anteriores de un usuario.

• *Incremento de la confianza y satisfacción* . ¿Qué hace el sistema para dar al usuario la confianza de que se está tomando la acción correcta? Por ejemplo, proporcionar comentarios que indiquen que el sistema está realizando una tarea de larga duración y la medida en que se completa la tarea aumentará la confianza del usuario en el sistema.

11.1. ESCENARIO GENERAL DE USABILIDAD

Las porciones de los escenarios generales de usabilidad son estas:

• *Fuente de estímulo*. El usuario final (que puede estar en un rol especializado, como un administrador del sistema o de la red) es siempre la fuente del estímulo para la usabilidad.

• *Estímulo* . El estímulo es que el usuario final desea utilizar un sistema de manera eficiente, aprender a usarlo, minimizar el impacto de los errores, adaptar el sistema o configurar el sistema.

• *Medio ambiente*. Las acciones del usuario que conciernen a la facilidad de uso siempre ocurren en el tiempo de ejecución o en el tiempo de configuración del sistema.

• *Artefacto* . El artefacto es el sistema o la parte específica del sistema con el que el usuario está interactuando.

• *Respuesta* . El sistema debe proporcionar al usuario las funciones necesarias o anticiparse a las necesidades del usuario.

• *Medida de respuesta*. La respuesta se mide por el tiempo de la tarea, la cantidad de errores, la cantidad de tareas realizadas, la satisfacción del usuario, el conocimiento del usuario, la relación entre las operaciones exitosas y las operaciones totales, o la cantidad de tiempo o datos perdidos cuando se produce un error.

La tabla 11.1 enumera los elementos del escenario general que caracterizan la usabilidad.

Tabla 11.1. Escenario general de usabilidad

Portion of Scenario	Possible Values
Source	End user, possibly in a specialized role
Stimulus	End user tries to use a system efficiently, learn to use the system, minimize the impact of errors, adapt the system, or configure the system.
Environment	Runtime or configuration time
Artifacts	System or the specific portion of the system with which the user is interacting
Response	The system should either provide the user with the features needed or anticipate the user's needs.
Response Measure	One or more of the following: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge, ratio of successful operations to total operations, or amount of time or data lost when an error occurs

La Figura 11.1 muestra un ejemplo de un escenario concreto de usabilidad que podría generar utilizando la Tabla 11.1 : El usuario descarga una nueva aplicación y la está utilizando de manera productiva después de dos minutos de experimentación.

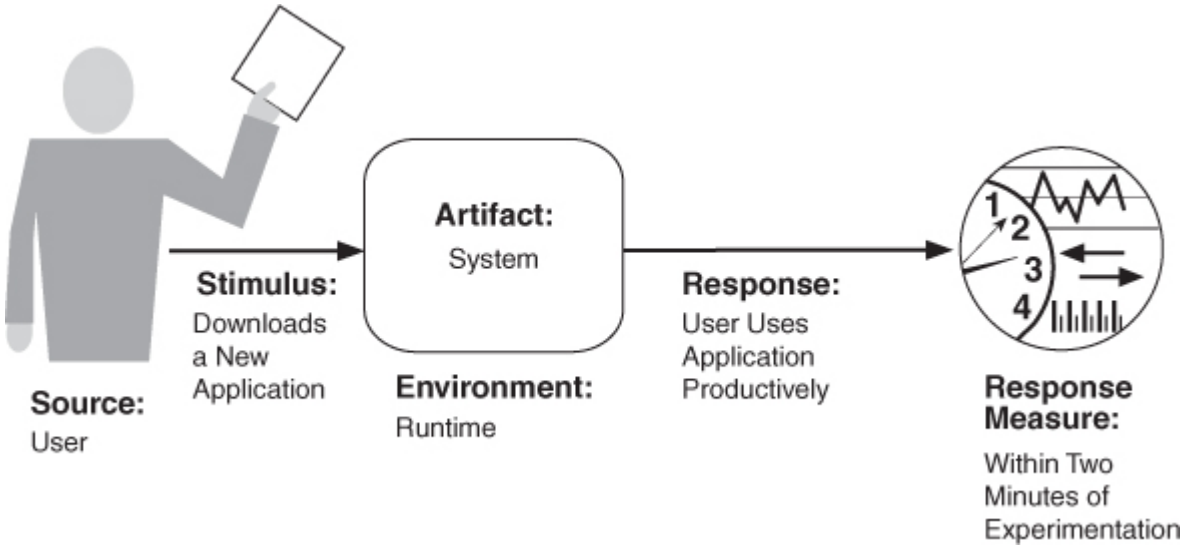

Figura 11.1. Escenario de usabilidad concreto de muestra.

11.2. TÁCTICAS PARA LA USABILIDAD

Recuerde que la facilidad de uso tiene que ver con lo fácil que es para el usuario realizar una tarea deseada, así como el tipo de soporte que el sistema le brinda al usuario. Los investigadores en interacción humano-computadora han utilizado los términos *iniciativa de usuario* , *iniciativa de sistema* e *iniciativa mixta*para describir cuál de la pareja humano-computadora toma la iniciativa para realizar ciertas acciones y cómo se produce la interacción. Los escenarios de usabilidad pueden combinar iniciativas desde ambas perspectivas. Por ejemplo, al cancelar un comando, el usuario emite una cancelación (iniciativa del usuario) y el sistema responde. Sin embargo, durante la cancelación, el sistema puede poner un indicador de progreso: iniciativa del sistema. Por lo tanto, cancelar puede demostrar iniciativa mixta. Utilizamos esta distinción entre el usuario y la iniciativa del sistema para discutir las tácticas que utiliza el arquitecto para lograr los distintos escenarios.

La Figura 11.2 muestra el objetivo del conjunto de tácticas de usabilidad en tiempo de ejecución.

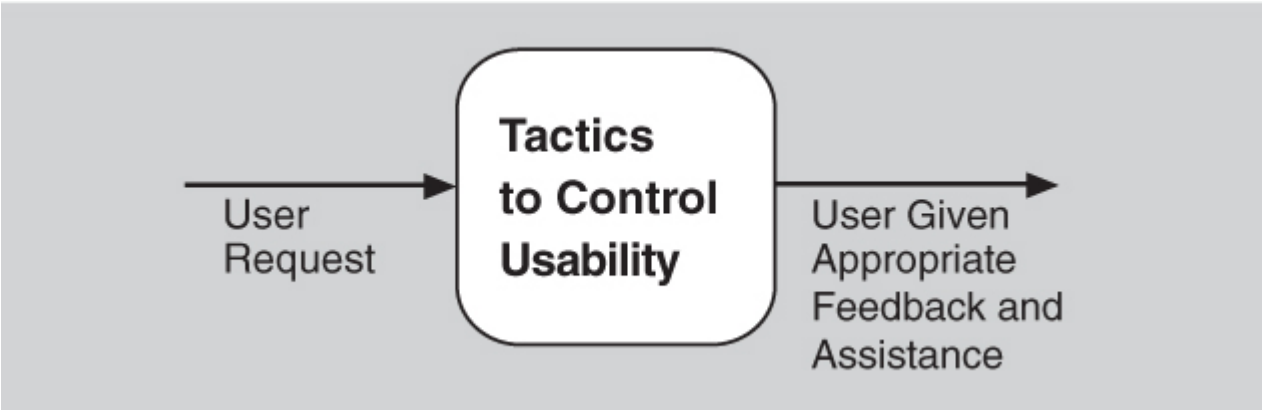

Figura 11.2. El objetivo de las tácticas de usabilidad en tiempo de ejecución.

¡Separe la interfaz de usuario!

Una de las cosas más útiles que un arquitecto puede hacer para que un sistema sea utilizable es facilitar la experimentación con la interfaz de usuario mediante la construcción de prototipos rápidos. La creación de un prototipo, o varios prototipos, para permitir que los usuarios reales experimenten la interfaz y proporcione su retroalimentación paga enormes dividendos. La mejor manera de hacer esto es diseñar el software para que la interfaz de usuario pueda cambiarse rápidamente.

Las tácticas de modificabilidad que vimos en el Capítulo 7 apoyan perfectamente esta meta, especialmente estas:

• Aumente la coherencia semántica, encapsule y coubique las responsabilidades relacionadas, que localizan las responsabilidades de la interfaz de usuario en un solo lugar

• Restrinja las dependencias, lo que minimiza el efecto de rizado a otro software cuando cambia la interfaz de usuario

• Aplazar el enlace, que le permite realizar elecciones de interfaz de usuario críticas sin tener que recodificar

La vinculación diferida es especialmente útil aquí, porque puede esperar que la interfaz de usuario de su producto enfrente la presión de cambiar durante las pruebas e incluso después de que salga al mercado.

Las herramientas de generación de interfaz de usuario son consistentes con estas tácticas; la mayoría produce un único módulo con una interfaz abstracta para el resto del software. Muchos proporcionan la capacidad de cambiar la interfaz de usuario después del tiempo de compilación. Puede hacer su parte restringiendo las dependencias en el módulo generado, en caso de que luego decida adoptar una herramienta diferente.

Mucho trabajo en diferentes patrones de separación de interfaz de usuario ocurrió en los años 80 y 90. Con el advenimiento de la web y la modernización

del patrón modelo-vista-controlador (MVC) para reflejar las interfaces web, MVC se ha convertido en el patrón de separación dominante. Ahora el patrón MVC está integrado en una amplia variedad de marcos diferentes. (Consulte el Capítulo 14 para una discusión de MVC). MVC facilita la presentación de múltiples vistas de los datos, apoyando la iniciativa del usuario, como veremos a continuación.

Muchas veces los atributos de calidad están en conflicto entre sí. La usabilidad y la modificabilidad, por otro lado, a menudo se complementan entre sí, porque una de las mejores formas de hacer que un sistema sea más utilizable es hacerlo modificable. Sin embargo, este no es siempre el caso. En muchos sistemas, las reglas comerciales controlan la IU, por ejemplo, especificando cómo validar la entrada. Para realizar esta validación, la IU puede necesitar llamar a un servidor (lo que puede afectar negativamente el rendimiento). Para sortear esta penalización de rendimiento, el arquitecto puede elegir duplicar estas reglas en el cliente y en el servidor, lo que dificulta la evolución. ¡Ay, la vida del arquitecto nunca es fácil!

Existe una conexión entre el logro de la usabilidad y la modificabilidad. El proceso de diseño de la interfaz de usuario consiste en generar y luego probar un diseño de interfaz de usuario. Las deficiencias en el diseño se corrigen y el proceso se repite. Si la interfaz de usuario ya se ha construido como una parte del sistema, entonces el sistema debe modificarse para reflejar el último diseño. De ahí la conexión con la modificabilidad. Esta conexión ha dado lugar a patrones estándar para admitir el diseño de la interfaz de usuario (ver barra lateral).

Iniciativa de usuario de soporte

Una vez que el sistema se está ejecutando, la facilidad de uso se mejora al proporcionarle al usuario información sobre lo que está haciendo el sistema y al permitirle que responda de manera adecuada. Por ejemplo, las tácticas que se describen a continuación (*cancelar* , *deshacer* , *pausar / reanudar* y *agregar)* ayudan al usuario a corregir errores o a ser más eficientes.

El arquitecto diseña una respuesta para la iniciativa del usuario al enumerar y asignar las responsabilidades del sistema para responder al comando del usuario. Aquí hay algunos ejemplos comunes de iniciativa de usuario:

• *Cancelar* . Cuando el usuario emite un comando de cancelación, el sistema debe escucharlo (por lo tanto, existe la responsabilidad de tener un escucha constante que no esté bloqueado por las acciones de lo que se esté cancelando); el comando que está siendo cancelado debe

ser terminado; cualquier recurso utilizado por el comando cancelado debe ser liberado; y los componentes que colaboran con el comando cancelado deben ser informados para que también puedan tomar las medidas apropiadas.

• *Deshacer* . Para admitir la capacidad de deshacer, el sistema debe mantener una cantidad suficiente de información sobre el estado del sistema para que se pueda restaurar un estado anterior, a solicitud del usuario. Dicho registro puede tener la forma de "instantáneas" de estado, por ejemplo, puntos de control, o como un conjunto de operaciones reversibles. No todas las operaciones se pueden revertir fácilmente: por ejemplo, cambiar todas las apariciones de la letra "a" a la letra "b" en un documento no puede revertirse cambiando todas las instancias de "b" a "a", porque algunas de esas instancias de "b" puede haber existido antes del cambio original. En tal caso, el sistema debe mantener un registro más detallado del cambio. Por supuesto, algunas operaciones, como tocar una campana, no se pueden deshacer.

• *Pausa / reanudar* . Cuando un usuario ha iniciado una operación de larga duración, por ejemplo, descargando un archivo grande o un conjunto de archivos desde un servidor, a menudo es útil proporcionar la capacidad de pausar y reanudar la operación. La pausa efectiva de una operación de larga duración requiere la capacidad de liberar temporalmente los recursos para que puedan ser reasignados a otras tareas.

• *Agregado* . Cuando un usuario está realizando operaciones repetitivas u operaciones que afectan a un gran número de objetos de la misma manera, es útil proporcionar la capacidad de agregar los objetos de nivel inferior en un solo grupo, de modo que la operación se pueda aplicar a la grupo, liberando así al usuario de la monotonía (y el potencial de errores) de hacer la misma operación repetidamente. Por ejemplo, agregue todos los objetos en una diapositiva y cambie el texto a fuente de 14 puntos.

Iniciativa del sistema de apoyo

Cuando el sistema toma la iniciativa, debe confiar en un modelo del usuario, la tarea que realiza el usuario o el estado del sistema en sí. Cada modelo requiere varios tipos de aportes para llevar a cabo su iniciativa. Las tácticas de *iniciativa del sistema de soporte* son aquellas que identifican los modelos que utiliza el sistema para predecir su propio comportamiento o la intención del usuario. Al encapsular esta información, será más fácil adaptarla o modificarla. La adaptación y

modificación pueden basarse dinámicamente en el comportamiento del usuario anterior o fuera de línea durante el desarrollo. Estas tácticas son las siguientes:

• *Mantener el modelo de tarea* . El modelo de tareas se utiliza para determinar el contexto, de modo que el sistema pueda tener una idea de lo que el usuario está intentando y brindar asistencia. Por ejemplo, saber que las oraciones comienzan con letras mayúsculas permitiría que una aplicación corrija una letra minúscula en esa posición.

• *Mantener modelo de usuario* . Este modelo representa explícitamente el conocimiento del usuario del sistema, el comportamiento del usuario en términos del tiempo de respuesta esperado y otros aspectos específicos de un usuario o una clase de usuarios. Por ejemplo, mantener un modelo de usuario le permite al sistema controlar la selección del mouse para que no se seleccione todo el documento cuando se requiere el desplazamiento. O un modelo puede controlar la cantidad de asistencia y las sugerencias que se proporcionan automáticamente a un usuario. Un caso especial de esta táctica se encuentra comúnmente en la *personalización de la* interfaz de usuario , en donde un usuario puede modificar explícitamente el modelo de usuario del sistema.

• *Mantener el modelo del sistema* . Aquí el sistema mantiene un modelo explícito de sí mismo. Esto se usa para determinar el comportamiento esperado del sistema para que el usuario pueda recibir los comentarios apropiados. Una manifestación común de un modelo de sistema es una barra de progreso que predice el tiempo necesario para completar la actividad actual.

La figura 11.3 muestra un resumen de las tácticas para lograr la usabilidad.

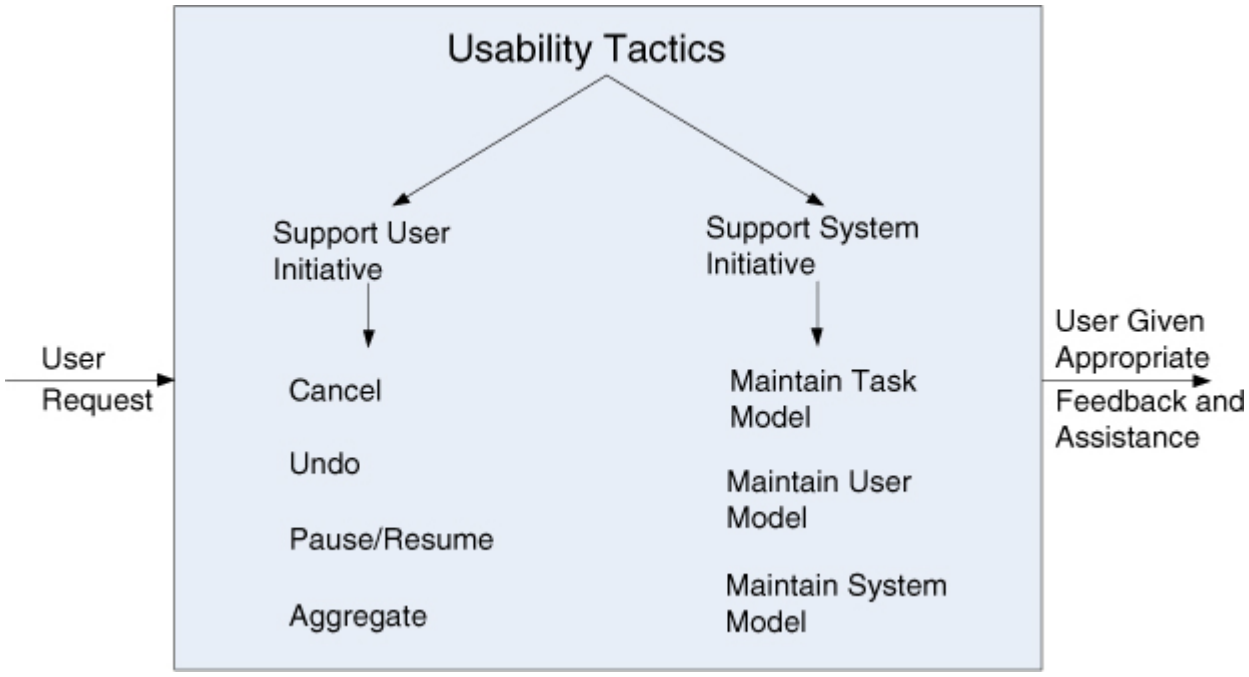

Figura 11.3. Tácticas de usabilidad

11.3. UNA LISTA DE VERIFICACIÓN DE DISEÑO PARA USABILIDAD

La Tabla 11.2 es una lista de verificación para respaldar el proceso de diseño y análisis para la usabilidad.

Tabla 11.2. Lista de verificación para apoyar el proceso de diseño y análisis de usabilidad

Category	Checklist
Allocation of Responsibilities	Ensure that additional system responsibilities have been allocated, as needed, to assist the user in the following: ▪ Learning how to use the system ▪ Efficiently achieving the task at hand ▪ Adapting and configuring the system ▪ Recovering from user and system errors
Coordination Model	Determine whether the properties of system elements' coordination—timeliness, currency, completeness, correctness, consistency—affect how a user learns to use the system, achieves goals or completes tasks, adapts and configures the system, recovers from user and system errors, and gains increased confidence and satisfaction. For example, can the system respond to mouse events and give semantic feedback in real time? Can long-running events be canceled in a reasonable amount of time?
Data Model	Determine the major data abstractions that are involved with user-perceivable behavior. Ensure these major data abstractions, their operations, and their properties have been designed to assist the user in achieving the task at hand, adapting and configuring the system, recovering from user and system errors, learning how to use the system, and increasing satisfaction and user confidence. For example, the data abstractions should be designed to support *undo* and *cancel* operations: the transaction granularity should not be so great that canceling or undoing an operation takes an excessively long time.
Mapping among Architectural Elements	Determine what mapping among architectural elements is visible to the end user (for example, the extent to which the end user is aware of which services are local and which are remote). For those that are visible, determine how this affects the ways in which, or the ease with which, the user

	will learn how to use the system, achieve the task at hand, adapt and configure the system, recover from user and system errors, and increase confidence and satisfaction.
Resource Management	Determine how the user can adapt and configure the system's use of resources. Ensure that resource limitations under all user-controlled configurations will not make users less likely to achieve their tasks. For example, attempt to avoid configurations that would result in excessively long response times. Ensure that the level of resources will not affect the users' ability to learn how to use the system, or decrease their level of confidence and satisfaction with the system.
Binding Time	Determine which binding time decisions should be under user control and ensure that users can make decisions that aid in usability. For example, if the user can choose, at runtime, the system's configuration, or its communication protocols, or its functionality via plug-ins, you need to ensure that such choices do not adversely affect the user's ability to learn system features, use the system efficiently, minimize the impact of errors, further adapt and configure the system, or increase confidence and satisfaction.
Choice of Technology	Ensure the chosen technologies help to achieve the usability scenarios that apply to your system. For example, do these technologies aid in the creation of online help, the production of training materials, and the collection of user feedback? How usable are any of your chosen technologies? Ensure the chosen technologies do not adversely affect the usability of the system (in terms of learning system features, using the system efficiently, minimizing the impact of errors, adapting/ configuring the system, and increasing confidence and satisfaction).

11.4. RESUMEN

El soporte arquitectónico para la usabilidad implica permitir que el usuario tome la iniciativa, en circunstancias como cancelar un comando de ejecución prolongada o deshacer un comando completado, y agregar datos y comandos.

Para poder predecir las respuestas del usuario o del sistema, el sistema debe mantener un modelo explícito del usuario, el sistema y la tarea.

Existe una fuerte relación entre el soporte del proceso de diseño de la interfaz de usuario y la modificabilidad del soporte; esta relación es promovida por patrones que imponen la separación de la interfaz de usuario del resto del sistema, como el patrón por capas.