Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico

Ana M. Moreno
Facultad de Informática
Universidad Politécnica de Madrid, España

M. Isabel Sánchez
Facultad de Informática
Universidad Carlos III de Madrid, España

Resumen

La usabilidad es uno de los atributos de calidad claves en el desarrollo de software. En este trabajo, se presenta una aproximación para mejorar la usabilidad de un sistema software aplicando un proceso específico de diseño para usabilidad. El contenido de este artículo forma parte de una investigación llevada a cabo dentro del proyecto de la Unión Europea IST STATUS, relacionado con el desarrollo de técnicas y procedimientos para mejorar la usabilidad, desde momentos tempranos del desarrollo de software. Esta aproximación difiere de la idea tradicional de medir y mejorar la usabilidad una vez finalizado el sistema. En este trabajo nos centraremos en la posible mejora de la usabilidad en el momento de diseño de software. Para el lo, se ha identificado, a partir de la literatura y de la experiencia de los socios industriales del proyecto, lo que se ha denominado patrones de usabilidad. Los patrones de usabilidad representan veinte mecanismos de usabilidad, por ejemplo, deshacer, ca ncelar, múltiples idiomas, etc., que mejoran la usabilidad del sistema final y que tienen un efecto en el diseño del sistema software donde se implante. Junto con estos patrones de usabilidad, también se presentan posibles soluciones de diseño para incorpo rar los mecanismos de usabilidad correspondientes en el diseño del sistema software. Las soluciones de diseño se han obtenido por medio de un proceso inductivo que garantiza que éstas son posibles soluciones aunque no tienen porque ser únicas.

Palabras clave: Arquitectura software, diseño software, usabilidad software

1. Introducción

Una de las razones por las que la investigación en el campo de arquitecturas software atrae cada vez más interés es la relación directa que existe entre las decisiones arquitec tónicas y la satisfacción de ciertos atributos de calidad [1]. La idea que subyace en esta relación es que una arquitectura software necesita ser diseñada de forma explícita para que el sistema final satisfaga unos atributos de calidad específicos. Los est udios más extendidos en este tema hacen referencia a atributos de calidad como rendimimiento, mantenibilidad, ...[2].

Otro atributo de calidad que está adquiriendo cada vez más importancia es la usabilidad del software [3]. Por tanto, sería interesante estudi ar la relación entre la arquitectura software y la usabilidad del sistema final. En este contexto surge el proyecto STATUS (SofTware Architectures That support USability) [1], cuyo objetivo es el desarrollo de técnicas y procedimientos a incorporar durante el diseño de un sistema software con el fin de conseguir mejoras en la usabilidad del sistema a construir. Esta perspectiva contrasta con la alternativa tradicional que consiste en medir la usabilidad del sistema una vez que éste se ha construido y mejorarla después iterativamente. La propuesta que se está abordando en el proyecto STATUS consiste en adelantar ese ciclo de evaluación/mejora al momento arquitectónico, tal como muestra la Figura 1.

[1] STATUS project: EU funded project IST–2001–32298.

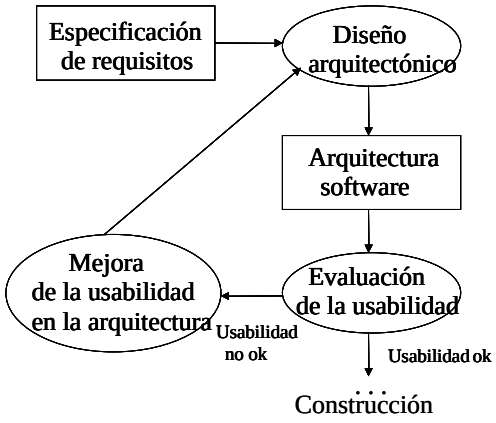


Figura 1. Método de diseño para usabilidad propuesto en STATUS

El proceso de diseño comienza con la construcción de un modelo de la arquitectura software a partir de un conjunto de requisitos funcionales. Aunque los ingenieros software no diseñarán este modelo preliminar aposta para que sea poco fiable o tenga bajo rendimiento, la mayoría de requisitos no funcionales son por lo general estudiados más adelante. Así, el diseño preliminar obtenido es evaluado con respecto a ciertos atributos de calidad, concretamente la usabilidad en nuestro caso. Si el valor de usabilidad obtenido es adecuado con respecto a lo solicitado en los requisitos el diseño finalizaría y se pasaría a las etapas de desarrollo posteriores. Si la usabilidad necesita mejorarse, se entraría en un ciclo de mejora en la arquitectura software y nuevas evaluaciones, hasta que el nivel de usabilidad sea el adecuado. Este proceso no implica que no se evalúe la usabilidad del sistema una vez terminado, simplemente se pretende adelantar el ciclo de evaluación/mejora de la usabilidad con el fin de ahorrar esfuerzos en el proceso de desarrollo.

En [4] y [5] se pueden encontrar posibles técnicas para evaluar la usabildad en el momento de diseño, desde un punto de vista dinámico mediante simulación de la arquitectura, y estático mediante un análisis de los modelos de diseño, respectivamente.

En este paper nos centraremos en las posibles mejoras de usabilidad que se pueden realizar en el momento arquitectónico. Para ello, en la sección 2, se presenta la aproximación adoptada para descomponer la usabilidad en varios niveles de abstracción que se acercan progresivamente a la arquitectura del software. Estos niveles progresivos son representados por los conceptos de atributos de usabilidad, propiedades de usabilidad y patrones de usabilidad.

Seguidamente, en la sección 3, se muestra cómo incorporar a la arquitectura software las características de usabilidad que representan los patrones de usabilidad. Para ello, se utilizará el concepto de patrón arquitectónico, que representa, en términos de componentes y las relaciones entre ellos, posibles soluciones para incorporar, en el diseño, aspectos que mejoran la usabilidad del sistema final.

Por último, la sección 4 presenta una breve discusión sobre la utilización de la aproximación adoptada en este trabajo.

2. Atributos, Propiedades y Patrones de Usabilidad

La usabilidad de los sistemas software se suele evaluar sobre el sistema finalizado, intentando asignar valores a los atributos de usabilidad clásicos [6] [7] [8] :

- Aprendizaje — rapidez y facilidad con las que los usuarios pueden realizar trabajo productivo con un sistema que no conocen, junto con la facilidad con la que se recuerda la forma en la que se debe operar con el sistema.
- Eficiencia en el uso — el número de tareas por unidad de tiempo que el usuario puede realizar con el sistema.
- Fiabilidad — también llamada “fiabilidad en el uso”, se refiere al porcentaje de errores cometidos por el usuario en el uso del sistema y el tiempo que se tarda en recuperarse de estos errores.

- Satisfacción — las opiniones subjetivas que forman los usuarios al utilizar el sistema.

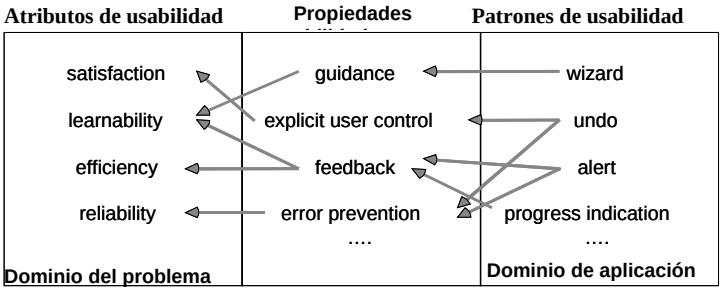
Sin embargo, el nivel de abstracción de estos atributos de usabilidad es demasiado alto como para poder estudiar los mecanismos que deben aplicarse a la arquitectura del software para mejorarlos. Por tanto, la filosofía que se ha seguido en STATUS ha sido descomponer estos atributos en dos niveles intermedios de conceptos que se acercan más a la solución software: las propiedades de usabilidad y los patrones de usabilidad.

El primer nivel consiste en relacionar los atributos de usabilidad antes mencionados con unas propiedades de usabilidad específicas que determinan las características de usabilidad que han de ser mejoradas en el sistema. Las propiedades de usabilidad pueden verse también como los requisitos que debe satisfacer un sistema software para que sea usable (por ejemplo, proporcionar retroalimentación al usuario, proporcionar el control explícito del usuario, proporcionar orientación al usuario, etc.). El segundo nivel se planteó para identificar unos mecanismos concretos que podrían incorporarse a la arquitectura de software para mejorar la usabilidad del sistema final. Estos mecanismos se han denominado patrones de usabilidad y abordan alguna necesidad indicada por una propiedad de usabilidad. Cabe mencionar que los patrones de usabilidad no proporcionan una solución software concreta para ser incorporada a la arquitectura del software, simplemente sugieren algún mecanismo abstracto que podría utilizarse para mejorar la usabilidad (por ejemplo, proporcionar *undos*, alertas, agregaciones de comandos, *wizards*, etc.).

El procedimiento que se ha seguido para identificar la relación entre los atributos, las propiedades y los patrones de usabilidad se describe en detalle en [9]. Se adoptó una aproximación *top-down* desde los atributos de usabilidad (definidos en la literatura) hasta la identificación de los patrones de usabilidad, pasando por las propiedades de usabilidad (derivadas , por una parte, de las heurísticas y guías indicadas en la literatura para la mejora de la usabilidad, y por otra, de la experiencia de los partners industriales del proyecto).

En la Tabla 1, se presenta un subconjunto de dicha relación. Esta tabla muestra cómo las propiedades de usabilidad relacionan los patrones con los atributos de usabilidad en un sentido cualitativo (una flecha indica que una propiedad afecta positivamente a un atributo, es decir, mejora este atributo). Por ejemplo, el patrón *wizard* mejora el aprendizaje: el patrón *wizard* utiliza la propiedad de *guidance* para guiar al usuario, paso por paso, en la realización de una tarea compleja; por otra parte, el *guidance* mejora el atributo de usabilidad aprendizaje. Los patrones de usabilidad pueden abordar una o varias propiedades de usabilidad, y las propiedades de usabilidad pueden mejorar uno o varios atributos de usabilidad.

Tabla 1. Relaciones entre Atributos, Propiedades y Patrones



El concepto de patrón de usabilidad ya ha sido utilizado en la literatura. En general, este concepto puede definirse como "una descripción de las soluciones que mejoran los atributos de usabilidad" [10]. Los aspectos de usabilidad que tratan estos patrones se refieren básicamente a las interfaces de usuario, por lo que se denominan también patrones de interfaz [11] o patrones de diseño de la interacción [12]. Tal como indican autores como Welie y Troetteberg [13], aunque que existen varias colecciones de patrones, no ha surgido aún un conjunto aceptado de éstos. No parece haber consenso acerca del formato y del enfoque de los patrones de interfaz.

Algunos ejemplos de patrones de interfaz son [11] [12] [13]: feedback, wizard, proporcionar al usuario toda la información que necesita en la misma ventana, marcar los campos necesarios para rellenar un formulario, etc

Las diferencias entre los patrones de usabilidad que se proponen en este trabajo y los patrones de interfaz o de usabilidad clásicos existentes en la literatura estriban básicamente en el hecho de que los patrones clásicos se basan en la mejora de la interfaz de la aplicación, lo que significa que estos patrones se implementan principalmente

durante la fase de diseño de la interfaz y, en general, afectan a los componentes de bajo nivel, como el pseudo código. Por otra parte, los patrones de usabilidad reseñados en este trabajo abordan los mecanismos considerados con la arquitectura del software, tratando los aspectos de usabilidad en los estadios tempranos del proceso de desarrollo. Por ejemplo, la solución que propone Welie [13] para el patrón de *feedback* se basa en "Proporcionar una indicación válida del progreso. Típicamente, el progreso se contabiliza como el tiempo que queda para terminar, el número de unidades procesadas o el porcentaje del trabajo realizado. El progreso puede representarse por medio de un *widget*, tal como una barra de progreso. La barra del progreso debe tener una etiqueta que indica el progreso relativo o la unidad de medición". Tal como se verá más adelante, la solución que se propone en este trabajo para este mismo patrón contempla los componentes que deben incorporarse a la arquitectura del software y las relaciones entre éstos.

En la segunda columna de la Tabla 2 se enumeran los patrones de usabilidad que se proponen en el STATUS. La primera columna de la tabla indica las propiedades de usabilidad que se relacionan con cada patrón.

Tabla 2. Lista de Patrones de Usabilidad

Propiedad de Usabilidad	**Patrón de Usabilidad**
NATURAL MAPPING	
CONSISTENCY (functional, interface, evolutionary)	
ACCESSIBILITY (internationalisation)	Different languages
CONSISTENCY, ACCSESIBILITY (multichannel, disabilities)	Different access methods
FEEDBACK	Alert
ERROR MANAGEMENT, FEEDBACK	Status indication
EXPLICIT USER CONTROL, ADAPTABILITY (user expertise)	Shortcuts (key and tasks)
ERROR MANAGEMENT (error prevention)	Form/field validation
ERROR MANAGEMENT (error correction),	Undo
GUIDANCE, ERROR MANAGEMENT	Context-sensitive help
GUIDANCE, ERROR MANAGEMENT	Wizard
GUIDANCE, ERROR MANAGEMENT	Standard help
GUIDANCE, ERROR MANAGEMENT	Tour
MINIMISE COGNITIVE LOAD, ADAPTABILITY, ERROR MANAGEMENT (error prevention)	Workflow model
ERROR MANAGEMENT (error correction)	History logging
GUIDANCE, ERROR MANAGEMENT (error prevention)	Provision of views
ADAPTABILITY (user preferences)	User profile
ERROR MANAGEMENT, EXPLICIT USER CONTROL	Cancel
EXPLICIT USER CONTROL	Multi-tasking
MINIMISE COGNITIVE LOAD ERROR MANAGEMENT (error prevention)	Commands aggregation
EXPLICIT USER CONTROL	Action for multiple objects
MINIMISE COGNITIVE LOAD, ERROR MANAGEMENT (error prevention)	Reuse information

Es importante subrayar que las propiedades de *Natural Mapping* y *Consistency* no pueden disponerse alrededor de unos patrones de usabilidad concretos. Esto se debe a que estas propiedades requieren la realización de distintas tareas y actividades durante todo el proceso de desarrollo en vez de la aplicación de soluciones particulares al nivel arquitectónico. Por ejemplo, para proporcionar natural mapping entre las tareas del usuario y las tareas que deben implementarse en el sistema, es necesario que se tenga en cuenta este objetivo durante la educción de los requisitos del software en el proceso de análisis y que el diseño se haga según estos requisitos. Lo mismo se puede decir de la consistencia, que implica distintas actividades a lo largo del proceso de desarrollo del sistema original o de versiones nuevas de éste.

3. Patrones Arquitectónicos de Usabilidad

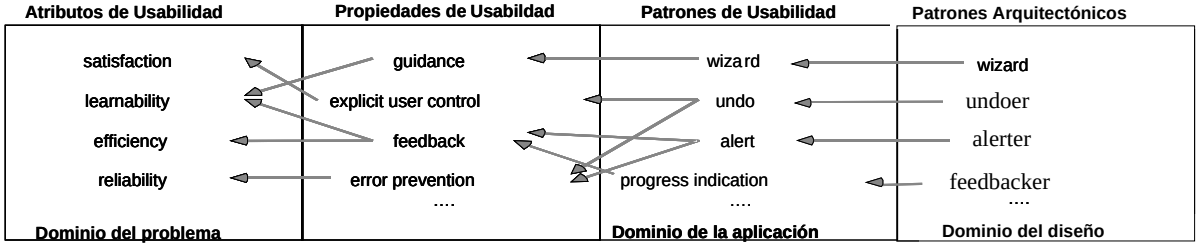
El concepto de patrón más ampliamente utilizado en el desarrollo del software es el patrón de diseño, que se utiliza particularmente en el paradigma orientado a objetos. En este contexto, un patrón de diseño es una descripción de las clases y de los objetos que trabajan conjuntamente para resolver un problema concreto [14]. Estos patrones muestran

una solución a un problema, que se ha obtenido a partir de su uso en otras aplicaciones diferentes. No obstante, debe señalarse que un patrón de diseño no debe verse como una solución única u original, sino como una posible solución.

En el proyecto STATUS, además de la idea del patrón de usabilidad, también se usa el concepto de patrón arquitectónico. Dado que el patrón de usabilidad se ha definido como un mecanismo a ser incorporado en el diseño de una arquitectura software a fin de abordar una propiedad de usabilidad concreta, un patrón arquitectónico determinará cómo se incorporará este patrón de usabilidad en una arquitectura software; es decir, qué efecto tendrá la inclusión del patrón de usabilidad en los compo nentes de la arquitectura del sistema. Abstrayendo la definición del patrón de diseño, un patrón arquitectónico puede definirse como una descripción de los componentes de un diseño y de la comunicación entre ellos para proporcionar una solución a un patrón de usabilidad. Al igual que los patrones de diseño, los patrones arquitectónicos reflejarán una posible solución a un problema: la incorporación de un patrón de usabilidad concreto en un diseño software.

Por lo tanto, el patrón a rquitectónico es el último eslabón en la cadena atributo, propiedad y patrón de usabilidad, y conecta la usabilidad del sistema software con la arquitectura de éste. Así, se puede añadir otra columna a la Tabla 1, tal como se muestra en la Tabla 3.

Tabla 3. Relaciones entre los atributos, propiedades y patrones de usabilidad y patrones arquitectónicos



3.1. Procedimiento de obtención de los patrones arquitectónicos de usabilidad

A continuación, se describe el procedimiento que se ha seguido para identificar los patrones arquitectónicos que representan soluciones de diseño para los patrones de usabilidad propuestos. Este procedimiento se compone de dos partes:

1. La aplicación de un proceso de inducción para abstraer los patrones arquitectónicos a partir de unos diseños particulares para varios proyectos desarrollados tanto por investigadores como por profesionales. Para ello, se adoptaron los siguientes pasos:

   1.1. Se pidió a los diseñadores que construyesen modelos de diseño para varios sistemas sin incluir los patrones de usabilidad.

   1.2. Para cada patrón de usabilidad, se pidió a los diseñadores que se modificasen sus diseños anteriores para incluir la funcionalidad correspondiente a los patrones considerados.

   1.3. Para cada patrón de usabilidad, mediante un proceso de abstracción, se obtuvo el patrón arquitectónico correspondiente a partir de las modificaciones introducidas por los desarrolladores en el diseño.

2. La aplicación de los patrones arquitectónicos resul tantes del paso anterior a varios desarrollos para validar su viabilidad.

Para ilustrar este proceso, se muestra a continuación la inducción de l patrón arquitectónico relacionado con el *status indication* a partir de una aplicación de gestión de pedidos y mesas de un restaurante.

El diagrama de secuencia representado en la Figura 2 y el diagrama de clases representado Figura 3 muestran parte del diseño de esta aplicación, concretamente la parte relacionada con la introducción del menú pedido por el cliente del restaurante. La Figura 4 y la Figura 5 muestran los diagramas de secuencia y clases, respectivamente, considerando en este caso la inclusión del patrón de usabilidad *status indication* para la funcionalidad anterior. Se

aprecia como la incorporación del *feedbacker* proporciona al us uario la información sobre lo que está haciendo el sistema.

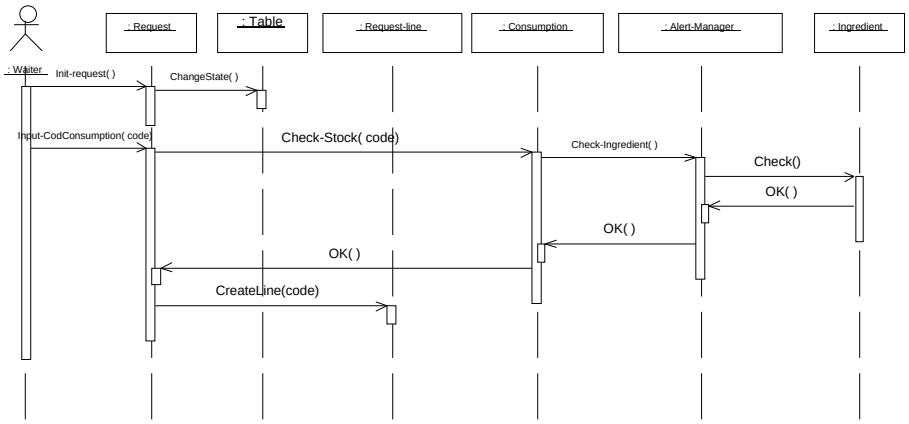


Figura 2. Diagrama de interacción sin patrón de usabilidad

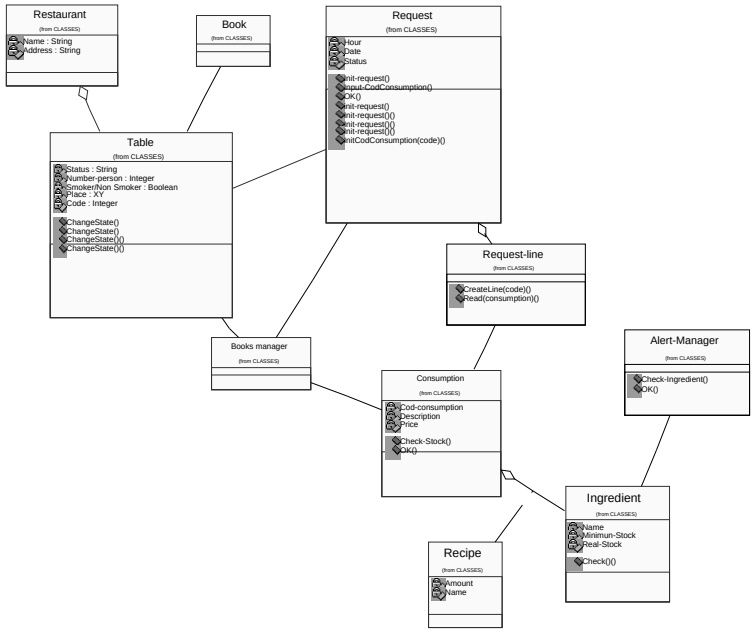


Figura 3. Diagrama de clases sin el patrón de usabilidad

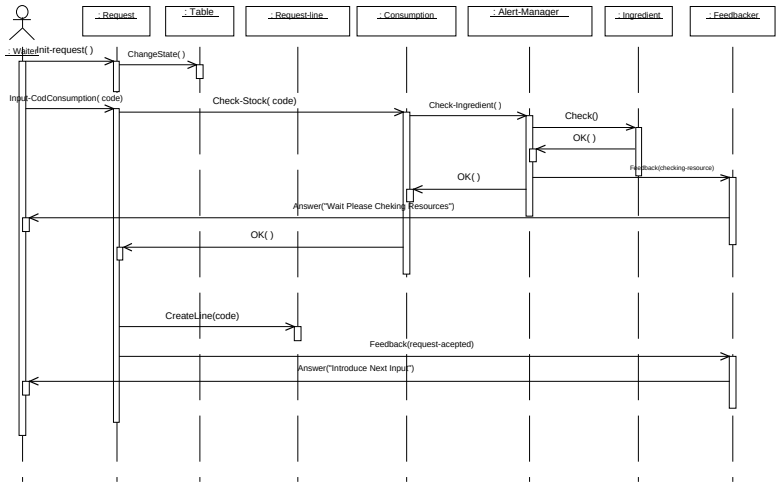


Figura 4. Diagrama de interacción con patrón de usabilidad

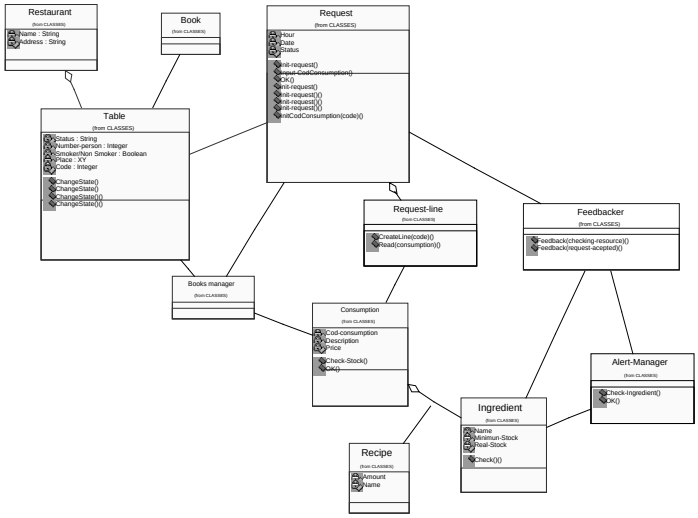


Figura 5. Diagrama de clases con patrón de usabilidad

A partir de este diseño y otros tanto s creados por otros desarrolladores, se ha abstraído una solución general de diseño, que se muestra en la Figura 6.

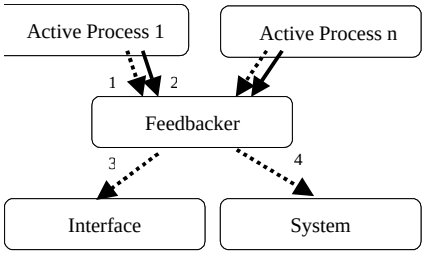


Figura 6. Solución genérica para el patrón feedbacker

Asimismo, se ha aplicado este proceso inductivo a los otros patrones de usabilidad para desarrollar los patrones arquitectónicos respectivos. Este proceso se encuentra detallado en [14].

3.2. Descripción de los Patrones Arquitectónicos

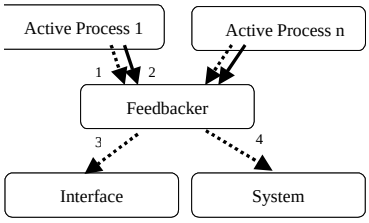
Como este trabajo tiene como objetivo último el proporcionar un conjunto de recomendaciones arquitectónicas para mejorar la usabilidad de los sistemas software, éstas se describirán en un catálogo de patrones arquitectónicos. Cada patrón de este catálogo debe describirse según los siguientes elementos:

- **Nombre del patrón** – Los patrones deben tener nombres sugestivos que, en una o dos palabras, dan una idea del problema que abordan y de su solución.
- **Problema** – Describe cuándo aplicar el patrón y en qué contexto. En el caso de los patrones arquitectónicos, el problema se refiere a un patrón de usabilidad concreto que ha de ser materializado.
- **Solución** – Describe los elementos que conforman la arquitectura, sus relaciones, sus responsabilidades, etc. La solución no describe un diseño definitivo, ya que un patrón puede verse como una plantilla que se aplica en muchas situaciones diferentes. En concreto, la solución para un patrón específico se hará explícita a partir de una:
  - **Representación gráfica** – Una figura que representa los componentes de la arquitectura y sus iteraciones. Las flechas numeradas entre los distintos componentes representarán las iteraciones. Las flechas con las líneas ininterrumpidas indican el flujo de datos, mientras que las líneas interrumpidas representan el flujo de control entre los componentes.
  - **Participantes** – Una descripción de los componentes que participan en la solución propuesta y las iteraciones (representadas por flechas) para determinar cómo deben asumir sus responsabilidades.
- **Ventajas para la usabilidad** – Descripción de los aspectos de usabilidad (propiedades de usabilidad) que pueden mejorarse al incluir el patrón apropiado.
- **Beneficios de usabilidad** – Una argumentación razonada acerca del impacto que tiene la aplicación del patrón en la usabilidad, es decir, cuáles son los atributos que se han mejorado y cuáles pueden haberse empeorado. En un primer momento, esta característica se completará con la información procedente de otros autores o de la experiencia de los socios del consorcio. Sin embargo, una vez que se hayan aplicado los patrones a aplicaciones reales, se rellenará este campo con los resultados de la experiencia empírica.
- **Consecuencias** – Impacto del patrón en otros atributos de calidad, como la flexibilidad, la portabilidad, la mantenabilidad, etc. Al igual que la característica anterior, ésta se rellenará con los resultados de la experiencia empírica.
- **Patrones relacionados** – Cuáles son los patrones arquitectónicos más íntimamente relacionados con éste, y cuáles son las diferencias entre ellos.
- **Implementación del patrón en OO** – Los patrones arquitectónicos proporcionados se pueden aplicar en cualquier paradigma de desarrollo. Sin embargo, debido a que estos patrones se han obtenido y se han refinado para aplicaciones OO, se proporcionarán guías tendentes a abordar la aplicación de los patrones en este campo. Básicamente, se describirán las clases que se derivan de los componentes principales del patrón. Estas guías se ilustran por medio de un ejemplo que se muestra en el apartado siguiente.
- **Ejemplo** de la aplicación del patrón considerado.

A continuación, se muestra a modo de ejemplo la descripción del patrón arquitectónico “Feedbacker” . Análogamente se han descrito los erstantes patrones arquitectónicos que proporcionan soluciones de diseño a los patrones de usabilidad propuestos en el proyecto STATUS:

- Nombre del patrón: Feedbacker
- Problema: Proporcionar al usuario información relacionada con el estado actual del sistema (patrón de usabilidad *Status indication*).
- Solución:

- Representación gráfica:



- Participantes:
  - *Active process i* : este módulo se representa más de una vez, ya que pueden ejecutarse al mismo tiempo varios procesos que solicitan la retroalime ntación (1), y cada proceso activo enviará (1) al *feedbacker* la información que desea que sea retroalimentada.
  - *Feedbacker*: este módulo se encarga de recibir la petición y los datos (1) (2), que indican el tipo de retroalimentación que se solicita y los dat os que deben retroalimentarse desde cada proceso activo. Además, debe conocer el destinatario de esta retroalimentación que enviará a otra parte del sistema y/o a la interfaz (3) para informar al usuario. En [13] , se indican algunas guías acerca de la forma en la que se puede visualizar esta retroalimentación en la interfaz de usuario, por ejemplo, cuántas veces debe refrescarse o dónde debe colocarse la información particular. Estos detalles deben tenerse en cuenta durante el diseño de bajo nivel.
  - Interfaz: la interfaz se encarga de recibir la retroalimentación y visualizarla para el usuario (3).
  - System: este componente es optativo y representa otras partes del sistema que deben ser informadas de la retroalimentación (4).
- Ventajas para la usabilidad: una indicación del estado del sistema proporciona al usuario retroalimentación (feedback) acerca de lo que está haciendo el sistema en ese momento y del resultado de cualquier acción que tome.
- Beneficios de la usabilidad: la retroalimentación proporciona al usuario información acerca del trabajo que está realizando el sistema y de en qu é medda la aplicación está procesando . Así, este patrón aumenta el atributo satisfacción.
- Consecuencias:
  - Este patrón impide una carga adicional del sistema al evitar reintentos por parte de los usuarios [13] .
  - Este patrón mejora la mantenibilidad del sistema porque canaliza mejor la información retroalimentada , a diferencia de cuando la retroalimentación se emite de forma indiscriminada por cualquier módulo del sistema.
- Patrones relacionados:
- Implementación OO: Este patrón arquitectónico dará lugar a una clase *feedbacker* especializada en informar al usuario y al sistema de lo que está pasando. De esta forma, todas las clases que desean informar de algo deben notificar al responsable de la retro alimentación, el *feedbacker*, para que éste pueda distribuir de forma adecuada esta información, ya sea dentro o fuera del sistema.
- Ejemplo: Esta sección describiría en detalle alguno de los ejemplos utilizados para obtener este patrón, verbigracia, el ejemplo mostrado en la Figura 5 y la Figura 6.

4. Discusión

Después de generar los patrones arquitectónicos, se pr opone presentar un conjunto de guías prácticas que proporcionen a los profesionales información acerca de:

- Cómo seleccionar un patrón arquitectónico, por ejemplo, a partir de los atributos de usabilidad que han de potenciarse en cada diseño y cómo determinar su impacto sobre otros atributos de calidad.
- Cómo utilizar un patrón arquitectónico para su incorporación a un diseño determinado.

El esfuerzo por mejorar la arquitectura con respecto a la usabilidad que se presenta en este trabajo se relaciona con otra parte importante del proyecto STATUS, que consiste en la valoración de la arquitectura con respecto a la usabilidad. En el proyecto, esta valoración se realiza de dos formas: una valoración arquitectónica basada en escenarios y una valoración arquitectón ica basada en la simulación. El resultado de esta evaluación será un conjunto de carencias que tiene una arquitectura software determinada con respecto a ciertos atributos o parámetros de usabilidad. Por tanto, los patrones de usabilidad podrían utilizarse para implementar soluciones que mejoren la usabilidad para las carencias detectadas.

Sin embargo, la idea de los patrones arquitectónicos puede también utilizarse con independencia de la evaluación de la arquitectura, ya que proporcionan soluciones de di seño para ciertos requisitos (cualquiera de las propiedades de usabilidad incluidas en la especificación de requisitos). Se espera que la consideración de estos requisitos de usabilidad desde el primer momento del desarrollo y más tarde en el diseño por me dio de los patrones arquitectónicos, proporcione mejoras en la usabilidad del sistema final.

Hay que tener en cuenta que la usabilidad del sistema software final ha de ser validada y medida una vez que éste se haya construido y esté operativo. Por tanto, tendremos que esperar hasta que estos resultados se hayan aplicado en proyectos reales para obtener los datos empíricos que puedan verificar adecuadamente los beneficios, ahora intuidos, del uso de los patrones arquitectónicos con respecto a la usabilidad de los sistemas. En el momento de la escritura de este paper los patrones están siendo aplicados a un proyecto real desarrollado por uno de los partners industriales. Tan pronto como el sistema esté desarrollado se realizarán las evaluaciones clásicas de u sabilidad con el fin de comprobar las mejoras en la usabilidad final del sistema construido.

5. Bibliografía

[1]. J Bosch. Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach, Pearson Education, Addison-Wesley, 2000.

[2] P. Bengtsson, J. Bosh. Analyzing software architecture for modifiability. Journal of Systems and Software, 2000.

[3] X. Ferré, N. Juristo, H. Windl, L. Constantine. Usability Basics for Software Developers. *IEEE Software*, vol 18 (11), p. 22 - 30

[4] S. Uchitel, R. Chatley, J. Kramer and J. Magee. LTSA-MSC: Tool Support for Behaviour Model Elaboration Using Implied Scenarios. Proceedings of TACAS '03, Warsaw April 2003.

[5] E. Folmer, J. Bosch. Usability patterns in Software Architecture. Aceptado en HCII 2003

[6] L. L. Constantine, L. A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage -Centered Design*. Addison-Wesley, Nueva York, NY, 1999

[7] J. Nielsen. *Usability Engineering*. AP Professional, 1993.

[8] B. Shackel. "Usability – context, framework, design and evaluation". En *Human Factors for Informatics Usability*. pp 21-38. B. Shackel and S. Richardson (eds.). Cambridge University Press, 1991.

[9] A. Andrés, J. Bosch, A Charalampos, R. Chatley, X. Fer re, E. Forlmer, N. Juristo, J. Magee, S. Menegos, A. Moreno. *Usability attributes affected by software architecture* . Deliverable 2. STATUS project, junio de 2002. Http://www.ls.fi.upm.es/status

[10] D. Perzel, D. Kane (1999). *Usability Patterns for Applications of the World Wide Web.* PloP'99

[11] G. Cascade. *Notes on a Pattern Language for Interactive Usability* , Proceedings of the Computer Human Interface Conference of the ACM, Atlanta, Georgia, 1997.

[12] J. Tidwell. *Interaction Design Patterns*. Pattern Languages of Programming 1998, Washington University Technical Report TR 98-25.

[13] M. Welie, H Troetteberg. *Interaction Patterns in User Interfaces*. PloP'00.

[14] E. Gamma, R Helm, R Johnson, J Glissades. *Design Patterns. Elements of Reusable Obje ct-Oriented Software*. Addison Wesley, 1998.

[15] N. Juristo, M. López, A. Moreno, M Sánchez. *Techniques and Patterns for Architecture -Level Usability Improvements* . Deliverable 3.4. STATUS project. Http://www.ls.fi.upm.es/status.