Mejora de la calidad en uso de una herramienta didáctica para crear interfaces gráficas en *Java™*

Hernán Molina
Facultad de Ingeniería
Universidad Nacional de La Pampa
Calle 110 , General Pico, La Pampa
hmolina.ing.unlpam@gmail.com

Resumen

El uso de herramientas de software didácticas puede beneficiar el proceso de enseñanza/aprendizaje del cuerpo de conocimientos en consideración. Estos beneficios son posibles siempre y cuando estas herramientas posean, en alguna medida, ciertas características, tales como Calidad en Uso*. En este artículo se presentan los resultados del proceso de mejora al que fue sometida la herramienta* jGUIAr*, diseñada y creada para asistir en la enseñanza y aprendizaje de la construcción de interfaces gráficas en* Java™*, en el contexto de una asignatura de* Programación Orientada a Objetos*. La mejora de* jGUIAr *se lleva a cabo siguiendo una estrategia de medición, evaluación y mejora (SIQinU) y modelos de calidad relacionados.*

1. Introducción

La enseñanza y aprendizaje de nuevos conceptos y paradigmas de programación puede beneficiarse del uso de herramientas que faciliten y conduzcan el aprendizaje de los elementos que componen el cuerpo de conocimientos a incorporar, y posibiliten la construcción de nuevas capacidades mediante la experimentación [1, 4]. Como caso particular, la incorporación de los conceptos del paradigma *Orientado a Objetos* propone un verdadero desafío a estudiantes formados en el paradigma procedural, más aún si a ello le agregamos los conceptos y mecanismos empleados en la construcción y programación de interfaces gráficas de usuario. Con este último propósito fue creada la herramienta *jGUIAr* (Java™ *Graphical User Interface Architect*) [9], en el contexto de la asignatura *Programación Orientada a Objetos* de las carreras de informática de la *Facultad de Ingeniería* de la *Universidad Nacional de La Pampa*.

La incorporación de una nueva herramienta a un proceso de enseñanza/aprendizaje requiere de una cuidadosa evaluación y análisis de su capacidad de ser un facilitador, y no un obstáculo, de tales procesos, más aún siendo una herramienta de reciente creación y en su primer versión. Por tal motivo, *jGUIAr* fue sometida a una evaluación para determinar si satisface al propósito para el que fue creada y, en caso contrario, guiar un camino de mejora para satisfacer dicho requerimiento [7]. Dicho requerimiento se corresponde al concepto de *Calidad en Uso*, definido como el grado en el que un producto puede ser utilizado por usuarios para satisfacer sus necesidades para lograr objetivos específicos con efectividad, eficiencia, satisfacción y libre de riesgos[1], en contextos de uso específicos [3]. Tal característica puede ser especificada en términos de aspectos o atributos medibles de menor granularidad, como se verá en las siguientes secciones.

En este trabajo se presentan los resultados del proceso de mejora llevado a cabo sobre la herramienta *jGUIAr*, que incluyen:

- el diseño de la evaluación y la prueba de usuarios, que determina un marco consistente que guía todo el proceso de mejora –resumido en la sección 4,
- una primer evaluación y correspondiente análisis de los resultados, tras la cual se proponen acciones de cambio en los casos necesarios –expuestos brevemente en la sección 5,
- la implementación de los cambios propuestos en la herramienta, obteniendo una nueva versión –presentada en la sección 6, y
- una posterior evaluación de la nueva versión, para determinar si los cambios implementados produjeron una mejora respecto de los resultados de la

[1] La libertad de riesgos no se considera en esta evaluación ya que no se aplica al tipo de producto a evaluar.

evaluación previa –sección 7.

En la sección 2 se presenta la primer versión de *jGUIAr* describiendo su funcionalidad y aspectos más relevantes. En la sección 3 se describe brevemente la estrategia utilizada para llevar adelante la medición, evaluación y mejora de *jGUIAr*. En la sección final del artículo se enuncian las consideraciones finales pertinentes y se proponen acciones futuras para continuar con la mejora de la herramienta.

2. Punto de partida: *jGUIAr 1.0*

jGUIAr permite diseñar y crear interfaces gráficas utilizando los componentes más comunes de la librería o paquete *AWT* (*Abstract Window Toolkit*) de *Java*^{TM}. Ya que el propósito de la herramienta es didáctico, el diseño de la interfaz se realiza de forma lógica siguiendo las reglas de composición del paquete *AWT*, utilizando una metáfora gráfica, y los pasos que se realizarían escribiendo directamente el código fuente. La metáfora utilizada en este caso es una estructura jerárquica de tipo árbol n-ario en la que los nodos representan objetos o instancias de alguna de las clases de componentes del *AWT* (ver Figura 1a).

La creación y configuración de cada componente es guiada por interfaces simples que solicitan los parámetros requeridos por uno de los constructores del tipo de componente. Además, *jGUIAr* permite crear instancias de los *layouts* del *AWT* para ser asignados a los contenedores deseados y así configurar la disposición de los componentes agregados a ellos.

La aplicación permite, cuando se lo requiera, generar el código fuente de la interfaz gráfica en construcción, así como compilar y ejecutar la aplicación para ver los resultados reales del diseño creado. El código fuente generado se mantiene limpio, ya que no agrega elementos o configuraciones que no fueron incluidas en el diseño, y se presenta de forma ordenada, incluyendo comentarios que describen los diferentes pasos en la construcción de la interfaz. Para su compilación y ejecución desde la herramienta, el código fuente se almacena en archivos *.java* (en un directorio o carpeta de la aplicación) correspondiente a la clase principal de la interfaz, pudiendo continuar con su edición posterior con el editor deseado.

La interfaz de *jGUIAr 1.0* se compone de cuatro áreas, según se muestra en la Figura 1a:

- *Componentes de la interfaz*: Lista los componentes creados, mostrando el nombre de la clase a la que pertenece y el nombre del identificador con el que aparecerá en el código fuente. Un menú contextual permite editar o eliminar los componentes. Para un contenedor, también permite agregar nuevos componentes de cualquiera de los tipos soportados (ver Figura 1b).
- *Jerarquía de agregación de componentes*: Muestra la agregación de los componentes que conformarán la interfaz. Cada componente se muestra usando el nombre de la clase a la que pertenece y su identificador, y ofrece el mismo menú contextual usado en la lista de componentes (ver Figura 1b).
- *Área de configuración de componentes*: Espacio donde se muestran los formularios que permiten editar la configuración de los componentes, así como la vista previa del código fuente de la interfaz diseñada.
- *Código fuente del objeto*: Muestra una vista previa del código fuente *Java*^{TM}asociado al objeto seleccionado (de la lista de objetos de la aplicación o en la jerarquía de agregación) que incluye su declaración e instanciación.

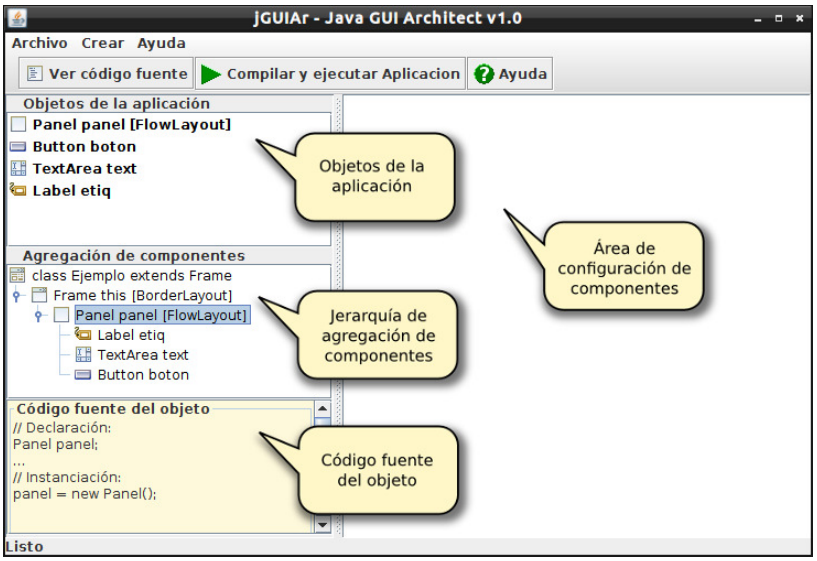
Además, la interfaz de *jGUIAr 1.0* cuenta con una barra de herramientas con accesos rápidos para mostrar la interfaz diseñada (compilar y ejecutar) y para previsualizar el código fuente de la misma. La entrada de menú *"Crear"* en la barra de menú permite la creación de componentes (así como el menú contextual de un contenedor, como se muestra en la Figura 1b) y de *layout managers* del *AWT*.

Los fundamentos didácticos de la herramienta se explican con mayor amplitud en [9].

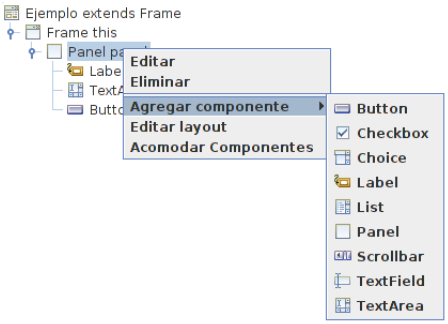
3. Estrategia de medición, evaluación y mejora

La estrategia utilizada en este trabajo, *SIQinU (Strategy for Improving Quality in Use)*, fue diseñada para comprender y mejorar requerimientos de *Calidad en uso* de sistemas de software de forma iterativa [6]. *SIQinU* se construye sobre otra estrategia de medición y evaluación de propósito general, llamada *GOCAME (Goal-Oriented and Context-Aware Measurement and Evaluation)* [13], compuesta por tres elementos integrados: (i) un proceso, (ii) un marco conceptual y (iii) métodos y herramientas instanciadas para los dos primeros. *SIQinU* extiende el proceso de *GOCAME* e instancia modelos y métodos incorporando la retroalimentación sucesiva de los resultados de evaluación y la implementación de cambios a los sistemas para producir nuevas versiones que reflejen un mayor grado de satisfacción de los requerimientos definidos.

La elección de *SIQinU* se fundamenta en que dicha estrategia fue aplicada con éxito en otro caso de estudio



(a) La interfaz de edición de *jGUIAr 1.0*.



(b) Menú contextual de un componente *Panel* (contenedor) en *jGUIAr 1.0*.

Figura 1: La interfaz de edición de *jGUIAr 1.0*.

[6] para evaluar y mejorar aspectos de calidad en uso de *JIRA*, una herramienta comercial para seguimiento de defectos de software, y además porque provee los mecanismos necesarios para llevar adelante el proceso de mejora de forma integrada.

A continuación se describen brevemente el proceso y el marco conceptual de esta estrategia.

3.1. El Proceso y marco conceptual de *SIQinU*

El proceso de *SIQinU* se enfoca en la mejora incremental y continua de la *Calidad en Uso* de un sistema de software, guiada por resultados de medición y evaluación de requerimientos no funcionales, para conocer el nivel de satisfacción de los mismos, y, posteriormente, la implementación de acciones correctivas correspondientes, orientadas a elevar dicho nivel de cumplimiento.

Las actividades y artefactos del proceso se construyen sobre los conceptos y relaciones especificados en el marco conceptual que acompaña a la estrategia, denominado *C-INCAMI* [13]. Este marco, que toma su nombre de los términos principales utilizados para especificar actividades de medición y evaluación –*Contextual Information Needs, Concept Models, Attributes, Metrics and Indicators*, permite especificar, de forma estructurada, el diseño e implementación de las actividades de medición y evaluación, haciendo posible la comparación consistente con resultados obtenidos en diferentes instancias. Además, *SIQinU* utiliza este marco conceptual para instanciar modelos de *Calidad en Uso* y, eventualmente, de *Calidad Externa* (inherentes al sistema de software), asociados a atributos medibles de la entidad a evaluar y mejorar.

El marco conceptual *C-INCAMI* se compone de cuatro módulos:

- *requerimientos no funcionales*: incluye la definición de la necesidad de información para diversas categorías de entidad y la especificación de los requerimientos no funcionales, mediante modelos de concepto que agrupan características o conceptos calculables y atributos cuantificables de las entidades.
- *contexto*: permite especificar el contexto relevante en el que se lleva a cabo la medición y evaluación –descripto mediante propiedades de contexto, así como el contexto en el que son aplicables las especificaciones reutilizables del marco *C-INCAMI* (tales como Modelos de Concepto, Métricas, entre otros). Las propiedades de contexto son tomadas del dominio de aplicación y cuantificadas por medio de métricas apropiadas;
- *medición*: permite especificar las entidades concretas a ser medidas, las métricas –que son tomadas de un catálogo de especificaciones reutilizables [10]- que cuantifican los atributos de los modelos de concepto y el registro de los valores medidos.
- *evaluación*: permite especificar indicadores elementales y derivados para valuar componentes de los modelos de concepto, modelos de agregación

para el cálculo global del concepto foco o de alto nivel, y los criterios de decisión para interpretar los resultados obtenidos.

El proceso de *SIQinU* se estructura en seis fases principales que describimos brevemente a continuación:

1 *Especificar los requerimientos y criterios de evaluación para Calidad en Uso*: se diseñan las tareas, se define el tipo de usuario, se especifica el contexto de uso y los requerimientos no funcionales (características y atributos) para *Calidad en Uso*. Luego, se seleccionan métricas e indicadores.

2 *Realizar la evaluación y análisis de Calidad en Uso*: se realiza la prueba con usuarios, quienes realizan las tareas diseñadas, y se recolectan los datos de uso del sistema (mediante el uso de registros) para los atributos especificados –tales como efectividad, eficiencia, completitud, entre otros. Luego, se realiza la evaluación de *Calidad en Uso* y el análisis preliminar.

3 *Derivar y especificar los requerimientos y criterios de evaluación para Calidad Externa* (ante la necesidad de mejora) en función de los resultados de evaluación de *Calidad en Uso* no satisfactorios de la fase anterior. Se seleccionan métricas para cuantificar los atributos identificados durante el diseño de requerimientos.

4 *Realizar la evaluación y análisis de Calidad Externa*. A partir de las especificaciones de la fase anterior, se implementan la medición y evaluación siguiendo las métricas seleccionadas y los indicadores diseñados.

5 *Recomendar e implementar acciones de mejora para los requerimientos no satisfactorios de Calidad Externa*, re-evaluando, posteriormente, el nivel de satisfacción –fase 4– para determinar la mejora en los requerimientos de *Calidad Externa* involucrados.

6 *Re-evaluar Calidad en Uso y analizar acciones de mejora.* La nueva versión del sistema se somete a la misma evaluación de *Calidad en Uso* con los usuarios finales en el mismo contexto (fase 2). Se analizan los resultados de esta última evaluación en función de los cambios realizados al sistema para satisfacer *Calidad Externa* (fase 5). Si fuera necesaria la mejora, se continúa con la fase 3.

4. Diseño de la evaluación

A continuación se resume el diseño de la evaluación de *Calidad en Uso* utilizado para guiar el proceso de mejora de *jGUIAr*, utilizando para ello el marco conceptual *C-INCAMI* (presentado originalmente en [8]). Este diseño, resultado de la fase 1 del proceso de *SIQinU*, establece los instrumentos y actividades que deberán repetirse para cada versión de la herramienta para determinar su grado de cumplimiento a los requerimientos no funcionales especificados, estableciendo, en cada iteración, la necesidad de realizar cambios para obtener un mayor grado de cumplimiento. Este diseño incluye:

- la *necesidad de información*, que define el propósito de la evaluación, el concepto foco, el punto de vista y la entidad a evaluar (en este caso, la herramienta *jGUIAr*),
- el *contexto*, que caracteriza la evaluación, necesario para asegurar la comparabilidad de futuros resultados,
- las *tareas* que llevarán a cabo los usuarios durante la *prueba de usabilidad* para recolectar los datos a medir,
- los *requerimientos no funcionales*, que determinan los *atributos* cuantificables a medir, asociados a las *características* a evaluar, en este caso *Calidad en Uso* y sus sub-características, y
- el *diseño de la medición y evaluación*, que consiste en las *métricas* utilizadas para cuantificar los atributos identificados, los *indicadores* que interpretan las características y los *instrumentos* para obtener y sistematizar el computo de las mediciones y la evaluación.

4.1. Necesidad de Información

La necesidad de información guia el diseño e implementación de la evaluación, y, a la vez, posibilita la comparación coherente de los sucesivos resultados para los aspectos de calidad en los cuales se enfoca la evaluación (ver Tabla 1).

Tabla. 1: Especificación de la necesidad de información para la evaluación de *jGUIAr*.

Propósito	Mejorar
Concepto calculable	Calidad en Uso
Punto de vista	Estudiante de Programación Orientada a Objetos
Categoría de entidad	Herramienta didáctica de software
Entidad	jGUIAr

4.2. Contexto

Esta especificación incluye propiedades (y los valores correspondientes) de las entidades relevantes –aquellas que afectan al uso de la herramienta, a los procesos

de medición y evaluación o a la interpretación de los resultados de los mismos. Estas propiedades se especifican y cuantifican siguiendo el marco *C-INCAMI*: realizando mediciones siguiendo una métrica asignada a la propiedad (atributo de una entidad). Algunas de las propiedades de contexto identificadas para la evaluación de *Calidad en Uso* de *jGUIAr 1.0* se presentan a continuación, agrupadas por entidad y destacando su valor en negrita:

- **Usuarios**
  - Tipo de usuario=**programador**
  - Nivel de experiencia=[ninguna | **estudiante** | junior | senior]
  - Paradigma de programación conocido= **procedural**
  - Tiempo de experiencia en programación procedural: **4420 horas**[2]
  - Experiencia en uso de interfaces gráficas=[ninguna | básica | **intermedia** | avanzada | experto]
  - Experiencia en el uso de la herramienta=[ninguna | una vez | **al menos tres veces**[3] | más de cinco veces];
  - Experiencia en el uso de herramientas similares=[**ninguna** | una vez | al menos tres veces | más de cinco veces]
  - Experiencia en el dominio=[ninguna | **principiante** | intermedio | avanzado | experto]
  - Motivación=[el usuario no utiliza la herramienta en su actividad | el usuario utiliza raramente la herramienta en su actividad | **el usuario utiliza frecuentemente la herramienta en su actividad** | el usuario utiliza siempre la herramienta en su actividad].
- **Herramienta**
  - Plataforma de software=***Java™***;
  - Categoría de entidad=**herramienta didáctica de software**;
  - Dominio=**interfaces gráficas**;
  - Etapa de versión entregable=[alfa | **beta**[4] | estable]
- **Ambiente**
  - Lugar de la prueba=**laboratorio de programación**
  - Privacidad=[**espacio dedicado** | espacio compartido]
  - Pluralidad de usuarios=[individual | **grupal**]
- **Proceso de prueba**
  - Tiempo de la prueba=**1 hora**
  - Técnica de registro=**archivos de registro**
  - Tipo de intervención=[intrusivo | **no intrusivo**]
  - Conocimiento de la técnica de registro por parte del usuario=[**consciente** | no consciente]
  - Cantidad de usuarios por tarea=**1**

Además, se registra como propiedad de contexto, para la entidad **Tarea**, las propiedades *estructura* –con valor "`estructurada`"- y *nivel de dificultad* de cada tarea. También, se registran las características de *hardware* y *software* de las computadoras donde se ejecuta la herramienta.

[2][140 horas/semana (Programación Procedural) +120 horas/semana (Estructura de datos y algoritmos)] x 17 semanas.

[3]Los estudiantes utilizan la herramienta durante una clase práctica guiada y en dos ejercicios de laboratorio.

[4]La herramienta fue sometida, en diciembre de 2013, a una prueba con usuarios avanzados en el dominio y se efectuaron los cambios apropiados.

4.3. Tareas

Durante la prueba de usabilidad *jGUIAr*, los usuarios realizan un conjunto de tareas. Éstas se diseñaron en función del propósito de la herramienta, siguiendo un conjunto de lineamientos basados en una filosofía de enseñanza de la programación *Orientada a Objetos* [5]. Las sucesivas tareas responden a diferentes niveles de dificultad, incorporando nuevos conceptos y desafíos de forma iterativa e incremental:

1 *Explorar* la configuración de los componentes de una interfaz gráfica simple, previamente creada con *jGUIAr*,

2 *Configurar* los parámetros de creación de los componentes de las interfaces gráficas dadas, observando los cambios resultantes en la interfaz,

3 *Agregar componentes* nuevos a la interfaz gráfica,

4 *Cambiar la disposición* de los componentes existentes para obtener una interfaz gráfica más apropiada y mejor diseñada que la original.

5 *Agregar contenedores y componentes*, utilizando la disposición y configuración que considere más apropiada al propósito solicitado.

6 *Crear una interfaz gráfica* desde el comienzo, a partir de una imagen de muestra.

A modo ilustrativo, en la *Tarea 3* (correspondiente al nivel de dificultad *Agregar componentes*), el usuario debe:

> *Partiendo de la interfaz* Default.jgui *modificada anteriormente (Tarea 2): agregar dos botones al panel 'panel': con identificadores 'bAnt' y 'bSig' y etiquetas "<" y ">" respectivamente. Observar los cambios en el código fuente y en la interfaz resultante. Guardar los cambios hechos a* Default.jgui.
>
> *Sub objetivos:*
>
> 1. *Crear Button bAnt con etiqueta "<"*
> 2. *Agregar bAnt al panel 'panel'*
> 3. *Crear Button bSig con etiqueta ">"*
> 4. *Agregar bSig al panel 'panel'*
> 5. *Compilar y ejecutar aplicación*
> 6. *Ver código fuente*
> 7. *Guardar*

Sólo la descripción de la tarea es entregada a los usuarios para realizar la prueba. Los sub-objetivos son usados por el evaluador para determinar la completitud de la tarea realizada.

4.4. Requerimientos No Funcionales

La evaluación de *Calidad en uso* de *jGUIAr* se enfoca en dos conceptos o características principales: *Usabilidad Real (Actual Usability, sinon. Usability in Use)*,

```
Quality in use (sinon. User Experience)
1. Actual Usability (sinon. Usability in use)
    1.1 Effectiveness
        A_1.1.1 Tasks effectiveness
    1.2 Efficiency
        A_1.2.1: Tasks efficiency on completeness
        A_1.2.2: Tasks efficiency on effectiveness
2. Satisfaction
    2.1 Universality
        A_2.1.1 User satisfaction on universality
    2.2 Usefulness
        A_2.2.1 User satisfaction on usefulness
    2.3 Learnability in use
        2.3.1 Communicability
            A_2.3.1.1 User satisfaction on communicability
        2.3.3 Readability
            A_2.3.3.1 User satisfaction on readability
        2.3.4 Navigability
            A_2.3.4.1 User satisfaction on navigability
        2.3.5 Familiarity
            A_2.3.5.1 User satisfaction on familiarity
        2.3.6 Appropriateness
            A_2.3.6.1 User satisfaction on appropriateness
```

Figura 2: Requerimientos no funcionales de *Calidad en Uso* utilizados para la evaluación de *jGUIAr*.

definido como *"grado con que usuarios especificados alcanzan objetivos específicos con efectividad, eficiencia, facilidad de aprendizaje en uso y sin brechas comunicacionales en un contexto de uso especificado"* [12]); y *Satisfacción*, definido como *"grado en que las necesidades del usuario son satisfechas cuando un producto es usado en un contexto de uso especificado"* [3]. Estos conceptos se incluyen en un modelo de requerimientos no funcionales, presentado en la Figura 2, incorporando los atributos medibles a partir de los cuales se interpretan los primeros.

4.5. Medición y Evaluación

El diseño de la evaluación se completa con la especificación de:

- las métricas, que determinan cómo cuantificar los atributos incluidos en los requerimientos de *Calidad en Uso* (recordar la Figura 2), junto a los instrumentos de medición necesarios para su obtención,
- los indicadores que interpretarán los atributos y características del árbol de requerimientos no funcionales, y
- la sesión de prueba durante la cual se aplican los instrumentos para obtener las mediciones.

4.5.1. Medición e instrumentos

Los atributos de una entidad se cuantifican utilizando *métricas* que determinan métodos e instrumentos para obtener el valor correspondiente. Las métricas seleccionadas para cuantificar los atributos incluidos en el modelo de concepto de *Calidad en Uso* (recordar la Figura 2) para evaluar la herramienta *jGUIAr* se resumen en la Tabla 2.

Tabla. 2: Métricas que cuantifican los atributos del modelo de concepto de *Calidad en Uso*

Atributo A1.1.1
Métrica: Porcentaje de tareas efectivamente completadas por los usuarios (TE) **Escala**: Numérica/Proporción **Unidad**: %(tarea) **Método**: $TE = \sum_{i=1}^{n} tE_i/n$ n es la cantidad de tareas
Atributo A1.2.1
Métrica: Nivel de eficiencia de tareas respecto de completitud (ETTC) **Escala**: Numérica/Proporción **Unidad**: %(usuarios)/segundos **Método**: $ETTC = \sum_{i=1}^{n} ETC_i/n$ n es la cantidad de tareas
Atributo A1.2.2
Métrica: Nivel de eficiencia de tareas respecto de efectividad (ETTE) **Escala**: Numérica/Proporción **Unidad**: %(usuarios)/segundos **Método**: $ETTE = \sum_{i=1}^{n} ETE_i/n$ n es la cantidad de tareas
Atributo A2.1.1 a A2.3.6.1
Métrica: Promedio de Nivel de satisfacción (PNS) **Escala**: Numérica/Absoluta **Unidad**: Satisfacción **Método**: $PNS = \sum_{i=1}^{n} NS_i/n$ donde n es la cantidad de items de satisfacción

Todas estas métricas son indirectas o derivadas (dependen de otras métricas para producir un valor). Cada una de estas métricas relacionadas cuantifican atributos de la misma u otra entidad relacionada, aunque no se incluyen en el modelo de conceptos diseñado. Por ejemplo, la métrica *Nivel de eficiencia de tareas respecto de completitud (ETTC)*, utilizada para cuantificar el atributo *A1.2.1*, depende de otra métrica indirecta, que a su vez, depende de dos métricas directas (ver Tabla 3).

Tabla. 3: Métricas utilizadas para calcular la métrica ETTC y los atributos que cuantifican.

Métricas relacionadas a ETTC	
Atributo: Eficiencia de tarea respecto de completitud **Métrica Indirecta**: Nivel de eficiencia de tarea respecto de completitud (ETC) **Escala**: Numérica/Proporción **Unidad**: %(usuarios)/segundos **Método**: PUCT/TPCT	**Métricas relacionadas a ETC**
	Atributo: Usuarios que completaron la tarea **Métrica Directa**: Porcentaje de usuarios que completaron la tarea (PUCT) **Escala**: Numérica/Absoluta **Unidad**: %(usuarios) **Método**: Calcular la proporción de usuarios que completaron la tarea.
	Atributo: Tiempo de completado de tarea **Métrica Directa**: Tiempo promedio de completado de una tarea (TPCT) **Escala**: Numérica/Absoluta **Unidad**: segundos **Método**: Calcular el promedio de tiempo que los diferentes usuarios utilizaron para completar la tarea

Para llevar a cabo las mediciones, se utilizan dos instrumentos. El primero es el registro automático de eventos en la herramienta para cuantificar los atributos

Tabla. 4: Cuestionario para cuantificar los atributos de Satisfacción.

A2.1.1 User satisfaction on universality	
Q1.	La herramienta se adapta a mis necesidades y habilidades.
A2.2.1 User satisfaction on usefulness	
Q2.	La herramienta me permitió completar los ejercicios solicitados.
Q3.	Entendí los conceptos y componentes involucrados en la creación de una interfaz gráfica.
A2.3.1.1 User satisfaction on communicability	
Q4.	Encuentro fácilmente en la interfaz las funciones u opciones que quiero utilizar.
Q5.	Reconozco fácilmente el propósito o funcionalidad asociada a los elementos de la interfaz.
Q6.	Las respuestas de la herramienta a mis acciones son las que yo esperaba.
Q7.	La herramienta ofrece una respuesta a cada una de las acciones que ejecuto.
Q8.	Encuentro razonable e intuitiva la forma de interactuar con la herramienta.
Q9.	Utilicé la ayuda provista por la aplicación.
Q10.	La ayuda me resultó útil para resolver mis dudas.
A2.3.3.1 User satisfaction on readability	
Q11.	Entiendo claramente las etiquetas y mensajes mostrados por la herramienta.
A2.3.4.1 User satisfaction on navigability	
Q12.	Me muevo fácilmente por las diferentes vistas/pantallas de la herramienta.
A2.3.5.1 User satisfaction on familiarity	
Q13.	Los componentes de la interfaz me son familiares y sé cómo usarlos.
A2.3.6.1 User satisfaction on appropriateness	
Q14.	La interfaz de la herramienta me permite comprender la estructura de los datos manipulados.

relacionados a *Usabilidad Real*. Estos eventos se almacenan en archivos de registro que los usuarios entregan como parte los resultados de la prueba.

El segundo instrumento es un cuestionario sobre la experiencia del usuario en el uso de la herramienta, quien lo completa de forma anónima una vez finalizadas las tareas (ver Tabla 4), utilizado para cuantificar los atributos de la característica *Satisfacción*. El cuestionario incluye afirmaciones para las cuales el usuario indica su grado de acuerdo según una escala *Likert* de cinco valores (desde *"totalmente en desacuerdo"* hasta *"totalmente de acuerdo"*).

4.5.2. Evaluación

El diseño de la evaluación involucra: *i)* criterios de decisión para los valores de indicadores, *ii)* modelos elementales para calcular valores de indicadores elementales a partir de los valores medidos de los atributos, y *iii)* modelos globales para calcular los valores de indicadores derivados para las características de alto nivel de los requerimientos no funcionales. Estos elementos del diseño de la evaluación se configuraron a partir de resultados de una prueba con usuarios expertos [14],

El criterio de decisión utilizado en esta evaluación es el siguiente:

Satisfactorio	[100..80]
Marginal	(80..60]
No satisfactorio	(60..0]

A modo de ejemplo, el indicador que interpreta el atributo *Eficiencia de tareas respecto de completitud* (código 1.2.1) incluye un modelo elemental que convierte valores medidos por la métrica *ETTC* (recordar la Tabla 2) a valores normalizados del indicador elemental (I_{ETTC}) según la siguiente ecuación (1):

$$I_{ETTC} = \begin{cases} 100 & \text{if } Met_{ETTC} > 0{,}625, \\ Met_{ETTC} * 160 & \text{si } Met_{ETTC} <= 0{,}625. \end{cases} \quad (1)$$

El modelo global utilizado es el modelo LSP [2], cuya fórmula de cálculo (ver ecuación 2) se basa en una suma ponderada de los valores de indicadores I_i, según pesos W_i, que permite modelar diferentes grados de preferencia o exigencia, utilizando un parámetro r, para cada nivel del árbol de requerimientos no funcionales.

$$e_0 = (W_1 I_1^r + ... + W_k I_k^r)^{1/r}, \quad (2)$$

donde $(W_1 + ... + W_k) = 1, W_i > 0, i = 1, ..., k..$

4.5.3. Sesión de prueba

El diseño de la sesión de prueba determina un conjunto de lineamientos que deben seguirse para posibilitar la objetividad de los resultados [15].

La prueba de la herramienta *jGUIAr* se lleva a cabo con los usuarios voluntarios que se encuentran cursando la asignatura, durante una clase práctica, dedicándole 1 hora a completar las tareas y responder el cuestionario. Según define la estrategia *SIQinU* (en su primer etapa), se especifican y registran las propiedades del contexto relevante. Antes de comenzar, se explica el procedimiento a los usuarios y se les entrega una copia impresa de las tareas a realizar y el cuestionario de satisfacción (a completar luego de finalizar las tareas). Se recuerda a los participantes de la prueba que el objeto a ser evaluado es la herramienta *jGUIAr*, destacando que, tanto los registros de eventos como el cuestionario, son anónimos. Cada usuario resuelve las tareas de forma individual.

Durante la realización de las tareas, un facilitador responde dudas sobre el dominio de la herramienta, no así sobre el uso de la misma, para no influir en los resultados de la evaluación.

Una vez concluida la prueba, se recolectan los archivos de registro de eventos de la sesión de trabajo de los usuarios y la encuestas correspondientes.

5. Evaluando *jGUIAr 1.0*

En esta sección se presentan los resultados de la evaluación de *Calidad en Uso* realizada a la versión *1.0* de

Tabla. 5: Valores de indicadores resultantes de la evaluación de *Calidad en Uso* de *jGUIAr 1.0*.

Quality in use (user experience)	**64.30**
1. Actual Usability (sinon. Usability in use)	**54.17**
1.1 Effectiveness	79.21
A1.1.1 Tasks effectiveness	79.21
1.2 Efficiency	28.23
A1.2.1: Tasks efficiency on completeness	29.13
A1.2.2: Tasks efficiency on effectiveness	27.32
2. Satisfaction	**87.22**
2.1 Universality	89.09
A2.1.1 User satisfaction on universality	89.09
2.2 Usefulness	90.91
A2.2.1 User satisfaction on usefulness	90.91
2.3 Learnability in use	82.13
2.3.1 Communicability	74.45
A2.3.1.1 User satisfaction on communicability	74.45
2.3.3 Readability	77.27
2.3.3.1 User satisfaction on readability	77.27
2.3.4 Navigability	81.82
2.3.4.1 User satisfaction on navigability	81.82
2.3.5 Familiarity	92.73
2.3.5.1 User satisfaction on familiarity	92.73
2.3.6 Appropriateness	85.45
2.3.6.1 User satisfaction on appropriateness	85.45

jGUIAr (fase 2 del proceso de *SIQinU*), siguiendo lo especificado en la Sección 4, originalmente presentados en [7]. En esta prueba participaron 12 estudiantes. Aunque la cantidad de usuarios excede la recomendada para pruebas de usabilidad (cinco) [11], la misma se justifica por la necesidad de cubrir los diferentes tipos de usuarios que van más allá del perfil genérico preestablecido (ver Sección 3.3 Contexto).

El resultado de la evaluación se materializa en un valor de indicador para cada componente del modelo de conceptos de *Calidad en Uso* (atributos y características), tal como se muestra en la Tabla 5, interpretados, según el criterio de decisión especificado, como *satisfactorio* (verde), *marginal* (amarillo) o *no satisfactorio* (rojo).

5.1. Análisis de los resultados

La característica *Actual Usability* obtuvo un valor de satisfacción marginal, por lo que se analizan los resultados de las dos características que lo componen. La característica 1.2 *Eficiencia* obtuvo un valor *no satisfactorio*. Ésta depende, en parte, del tiempo utilizado para completar la tarea (cuya métrica asignada es utilizada por las métricas asociadas a los atributos 1.2.1 y 1.2.2). En el gráfico *"Tiempos de completado de tareas"* (Figura 3), se observa que los tiempos fueron bajando hasta la tarea 3 y luego aumentando en mayor proporción. A partir de la tarea 4 se incorpora el uso de *Layout Managers*[5] (recordar los niveles de dificultad de las tareas vistos en la Sección 4.3). Los registros de las

[5]Utilizados para distribuir los componentes en una interfaz gráfica siguiendo diferentes estrategias o esquemas.

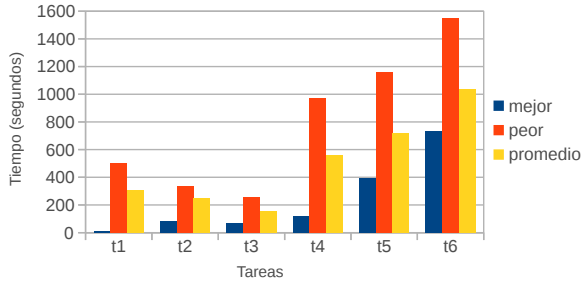


Figura 3: Tiempos de completado de las tareas para *jGUIAr 1.0*.

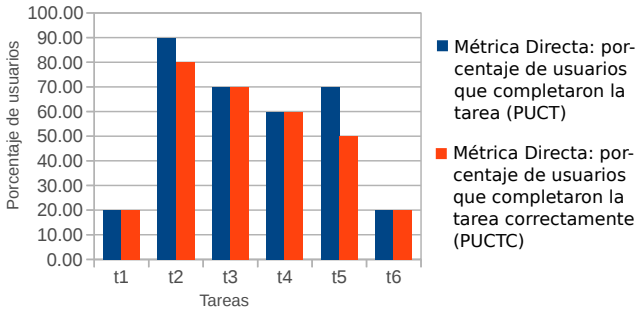


Figura 4: Completitud y correctitud en la realización de las tareas para *jGUIAr 1.0*.

pruebas, evidencian que los usuarios tienen dificultad para encontrar la funcionalidad *"Acomodar componentes"* asociados al uso de *layouts*, aumentando el tiempo necesario para completar la tarea.

La *eficiencia* también se ve afectada por el logro de los sub-objetivos previstos para cada una, considerando la completitud (atributo 1.2.1) y la correctitud (atributo 1.2.2) con la que fueron realizadas por parte de los usuarios.

Respecto de la *completitud de tareas*, computado por la *métrica directa "Porcentaje de usuarios que completaron la tarea"* (ver Figura 4), se observa que ninguna de las tareas fue completada por todos los usuarios, la tarea 2 fue la que completaron la mayor cantidad de usuarios (90 %), las tareas 1 y 6 fueron completadas por la menor cantidad de usuarios (sólo un 20 %) y el resto (tareas 3 a 5) fueron completadas por entre un 60 % y 70 % de los usuarios.

Respecto de la *correctitud en las tareas completadas*, computada por la métrica directa *"Porcentaje de usuarios que completaron la tarea correctamente"* (ver Figura 4), no se observaron errores en las tareas 1, 3, 4 y 6 para los usuarios que completaron la tarea.

La segunda característica que compone *Actual Usability* es *Efectividad*, que resultó con un valor marginal (recordar la Tabla 5). El único atributo que compone

esta característica se cuantifica mediante una métrica indirecta que depende de la completitud y correctitud de las tareas realizadas por los usuarios. En este caso, basada, no en la cantidad de usuarios que completaron las tareas, sino en la cantidad de sub-objetivos completados de cada tarea .

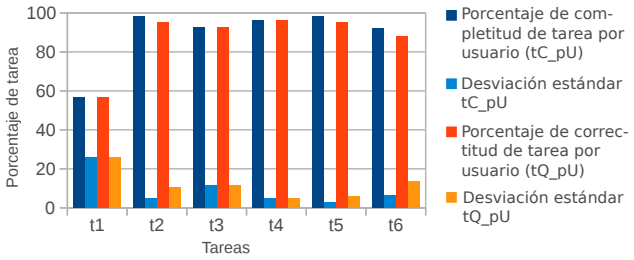


Figura 5: Completitud y correctitud en la realización de las tareas considerando los sub-objetivos para *jGUIAr 1.0*.

En el gráfico de la Figura 5 se representan los valores obtenidos para estos dos atributos. La métrica indirecta *"Porcentaje de completitud de tarea por usuario (tC_pU)"* cuantifica el grado de completitud de cada tarea, y la métrica indirecta *"Porcentaje de correctitud de tarea por usuario (tQ_pU)"* cuantifica el grado de correctitud de cada tarea, ambos promediados entre la cantidad de usuarios. Para una mejor interpretación, se consideran los desvíos estándar de cada métrica.

Sorpresivamente, las tareas relativamente simples, como la tarea 1 (explorar la configuración), son las que menos sub-objetivos completaron los usuarios. El nivel baja levemente en la tarea 3 (agregar componentes), donde se incorporan nuevos desafíos, mejorando hasta la tarea 5 (agregar contenedores y componentes), que incluye sub-objetivos similares a los ya experimentados, aunque en mayor cantidad que en la tarea 3. Luego, en la tarea 6 (crear interfaz), que representa un nuevo desafío, se observa una caída de nivel, similar a los observados en la tarea 3.

Respecto de la característica *2. Satisfacción*, aunque resultó con una valoración satisfactoria, se debe prestar atención a la sub-característica *"Learnability in Use"*, cuyas sub-características *"Communicability"* y *"Readability"* resultaron con valores marginales. En las preguntas *Q4*, *Q5*, *Q6*, *Q9* y *Q11*, utilizadas para computar los valores de satisfacción de las características mencionadas, al menos dos usuarios respondieron con un valor medio o menor en la escala utilizada (presentada en la Sección 4.5), en concordancia con los problemas observados en el análisis de las métricas asociadas a efectividad y eficiencia.

En resumen, claramente, el problema principal de la herramienta se encuentra en el tiempo utilizado por los usuarios para completar las tareas. Luego, en menor medida, se observan problemas para completar los sub-objetivos de las tareas propuestas.

Cabe aclarar que no todos los usuarios habían utilizado la herramienta previamente como se había especificado en el perfil del usuario. Este hecho se registra durante la sesión de prueba, ante los comentarios de los usuarios.

5.2. Cambios propuestos

A partir de los problemas observados durante el análisis preliminar, y siguiendo con el proceso de la estrategia *SIQinU* aplicado en este estudio (presentado en la Sección 3.1), se deben derivar requerimientos de *Calidad externa* que pudieran influir en los resultados obtenidos (fase 3). No obstante, y considerando que muchos de los problemas de *Calidad en uso* son evidentes, se adelanta parte de la etapa de *Recomendación de acciones de mejora* para los requerimientos de *Calidad en uso* que resultaron marginales y no satisfactorios (fase 5). Ambos resultados se plasman en la Tabla 6, donde, para cada atributo del modelo de *Calidad en uso* en cuestión, se resumen los problemas observados y los cambios propuestos para solucionarlos.

6. Aplicando los cambios: *jGUIAr 1.1*

Siguiendo los cambios propuestos a la versión 1.0 de *jGUIAr*, en función de los resultados de la primer evaluación, se lleva adelante su implementación, produciendo una nueva versión de la herramienta. Los cambios realizados, que pueden observarse en la Figura 6, se resumen a continuación.

Se agregaron iconos representativos a todas las opciones del menú contextual de los componentes de la interfaz, incluyendo la opción *"Acomodar componentes"* para contenedores (ver Figura 7). También se incorporaron textos informativos flotantes (*tooltips*).

Se incorporaron botones a la barra de herramientas para crear los tipos de componentes y *layouts* del *AWT* soportados, utilizando un icono representativo para cada caso, también con textos flotantes.

Se agregó, de forma siempre visible, un *Área de código fuente* de la aplicación completa, utilizando coloreado de sintaxis para facilitar la lectura del mismo. El código fuente se actualiza automáticamente en pantalla cuando se efectúan cambios en cualquiera de los componentes de la interfaz gráfica o en la jerarquía de agregación. El *área de configuración de componentes*, siempre visible en la versión 1.0, se mantiene "minimizada" arriba del espacio del código fuente y se expande al requerir configurar algun componente. También se

Tabla. 6: Cambios propuestos a los problemas encontrados.

Atributo	Problema de Calidad en Uso	Cambios propuestos
1.2.1 Tasks efficiency on completeness 1.2.2 Tasks efficiency on effectiveness 2.3.1.1 User satisfaction on communicability	- Los usuarios demoran en acceder a la funcionalidad “acomodar componentes” de un contenedor. - Los usuarios no reconocen o interpretan el significado de algunas etiquetas de la herramienta. - Algunos usuarios no utilizaron la ayuda provista por la herramienta.	- Hacer más visible o intuitivo al acceso a la funcionalidad “Acomodar componentes” de un contenedor. - Revisar la pertinencia de las etiquetas de los menús y botones de la interfaz; proveer textos descriptivos contextuales para los botones y opciones de menú. - Incluir un componente contextual para acceder a la ayuda referida al componente o acción en cuestión.
1.1.1 Tasks effectiveness 1.2.1 Tasks efficiency on completeness	- Algunos usuarios omitieron el subobjetivo “ver el código fuente resultante” (*1) - Algunos usuarios no utilizaron la funcionalidad esperada (editar) para ver la configuración de los componentes (*1). - Algunos usuarios no cambiaron el nombre de la aplicación/interfaz gráfica. - Algunos usuarios omitieron el subobjetivo “Guardar la interfaz gráfica”. - Algunos usuarios omitieron el subobjetivo “Compilar la aplicación”	- Incluir un mecanismo que facilite acceder a los atributos o configuración de un componente (Ejemplo: Texto flotante sobre el componente o una sección de la interfaz que muestre estas propiedades al seleccionar un componente –se hace actualmente con el código fuente). - “Ver código fuente”: incluir una vista que muestre constantemente el código fuente actualizado de la interfaz gráfica en edición. - “Compilar la aplicación”: incluir un indicador visual para notificar que existen cambios en la interfaz gráfica que no han sido compilados/vistos en la interfaz resultante. - “Guardar los cambios”: incluir un indicador visual para notificar que existen cambios en la interfaz gráfica que no han sido guardados.
1.1.1 Tasks effectiveness 1.2.2: Tasks efficiency on effectiveness	- Algunos usuarios confundieron componentes del AWT. - Idem *1	- Revisar los iconos de los diferentes componentes y proveer un texto descriptivo breve para cada uno. - Ver código fuente: ver recomendaciones anteriores.
2.3.3.1 User satisfaction on readability	- Los usuarios no interpretan correctamente el significado de algunos mensajes de la herramienta.	- Revisar la pertinencia de las etiquetas de los diálogos mostrados al usuario

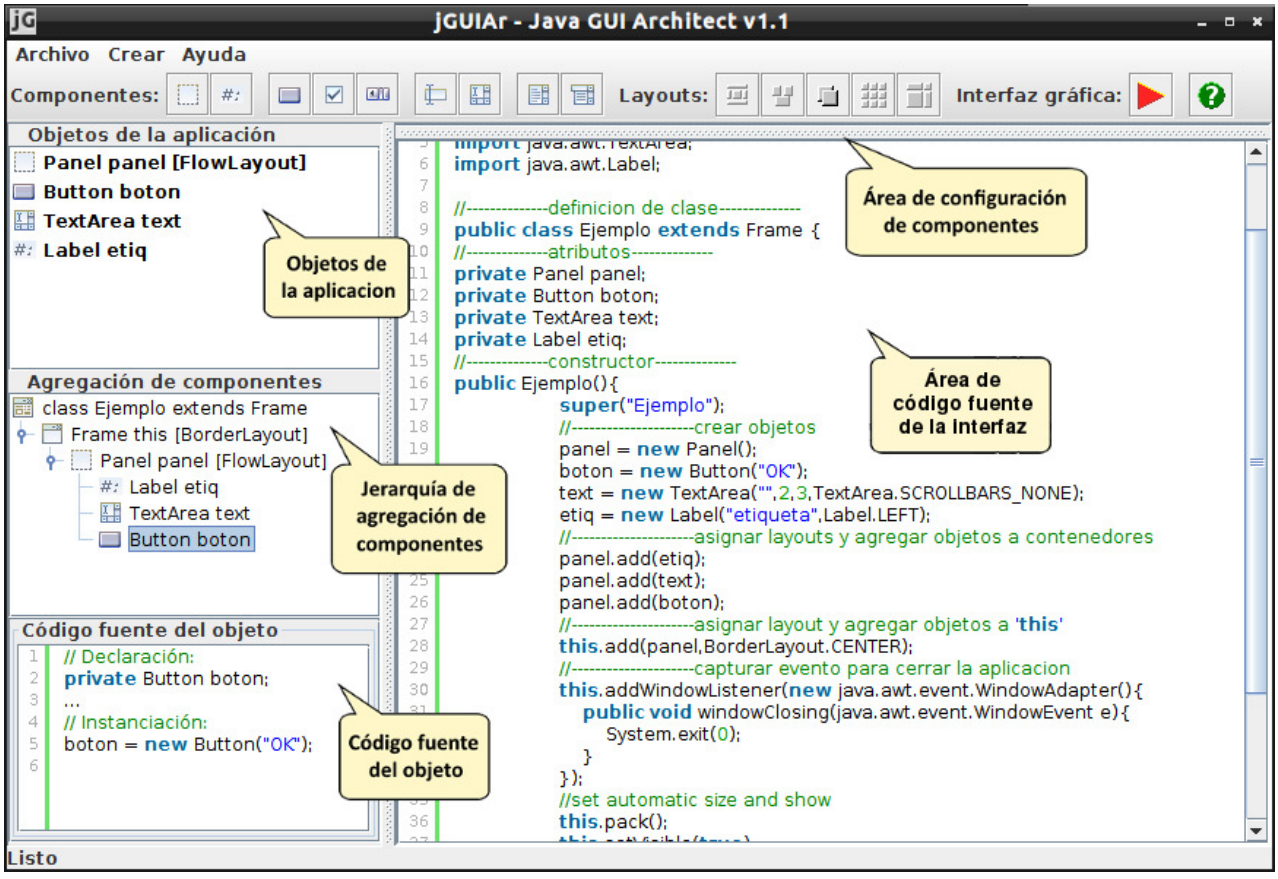


Figura 6: La interfaz de edición de *jGUIAr 1.1*.

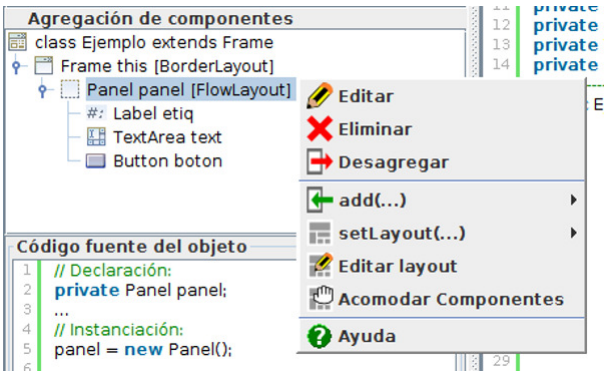


Figura 7: Menú contextual de un contenedor en *jGUIAr 1.1*.

incorpora el coloreado de sintaxis al área de *Código fuente de objetos* para facilitar su lectura.

Se modificó el botón para compilar y ejecutar la aplicación (en la barra de herramientas), utilizando un icono representativo e intuitivo (metáfora de *ejecutar* o *"play"*) cuyo color varía para advertir al usuario que hay cambios en la aplicación que no fueron compilados/observados. Cuando la configuración actual de la interfaz gráfica fue compilada y observada, el botón se muestra de color verde; cuando existen cambios sin compilar, el botón se muestra de color rojo.

Se incluyó un botón a la barra de herramientas para acceder a la ayuda general de la aplicación. También se incorporó un botón de ayuda contextual al menú contextual que ofrece cada uno de los componentes creados en la aplicación, que permite acceder directamente a la ayuda asociada a ese tipo de componente (ver Figura 7).

Se implementó un botón contextual para acceder rápidamente a la ventana de edición de un componente, permitiendo, desde allí, realizar todas las operaciones accesibles desde el menú contextual del mismo. Este botón se muestra junto a cada componente cuando es seleccionado, tanto en la lista de objetos de la aplicación como en el árbol de agregación de componentes.

Se incorporaron marcas visuales para distinguir aquellos componentes que aún no fueron configurados de manera de lograr un resultado coherente en la interfaz final. Por ejemplo, un contenedor sin componentes agregados se destaca en color rojo, tanto en la lista de componentes como en el árbol de agregación de componentes; de la misma forma ocurre con un componente agregado en un contenedor cuando no se le han aplicado restricciones, y el *layout* utilizado así lo requiere (por ejemplo, *BorderLayout* ofrece restricciones *CENTER*, *NORTH*, *SOUTH*, entre otras).

También, para agilizar la edición y emular el proceso de programación manual, se incorporó la posibilidad de agregar y desagregar componentes a la interfaz, independientemente de su creación o eliminación (ver Figura 7). En la versión anterior la creación de un componente estaba atada a su agregación a algún contenedor de la interfaz, de forma que, para moverlo a otro contenedor, debía eliminarse y volverse a crear.

Tabla. 7: Valores de indicadores resultantes de la evaluación de *Calidad en Uso* de *jGUIAr v1.1*.

Quality in use (user experience)	**82.60 +**
1. Actual Usability (sinon. Usability in use)	**73.68**
1.1 Effectiveness	87.59 +
A1.1.1 Tasks effectiveness	87.59 +
1.2 Efficiency	55.97 +
A1.2.1: Tasks efficiency on completeness	56.48 +
A1.2.2: Tasks efficiency on effectiveness	55.47 +
2. Satisfaction	**100**
2.1 Universality	100
A2.1.1 User satisfaction on universality	100
2.2 Usefulness	100
A2.2.1 User satisfaction on usefulness	100
2.3 Learnability in use	100
2.3.1 Communicability	100 +
A2.3.1.1 User satisfaction on communicability	100 +
2.3.3 Readability	100 +
2.3.3.1 User satisfaction on readability	100 +
2.3.4 Navigability	100
2.3.4.1 User satisfaction on navigability	100
2.3.5 Familiarity	100
2.3.5.1 User satisfaction on familiarity	100
2.3.6 Appropriateness	100
2.3.6.1 User satisfaction on appropriateness	100

7. Evaluando *jGUIAr 1.1*

Siguiendo con lo especificado en el diseño presentado en la Sección 4, se replica la evaluación para la versión *1.1* de *jGUIAr* para determinar si los cambios incorporados produjeron una mejora en los niveles de satisfacción de *Calidad en Uso*.

En esta instancia, la prueba se realizó con 10 estudiantes de la asignatura donde se pretende aplicar la herramienta, en las mismas condiciones diseñadas para la primer prueba, para hacer posible una comparación consistente de los resultados.

Los resultados cuantitativos de la evaluación para los requerimientos no funcionales de *Calidad en Uso* se presentan en la Tabla 7, destacando con un **"+"** los valores de aquellos requerimientos que pasaron de un nivel *no satisfactorio* a *marginal* o de *marginal* a *satisfactorio* (recordar los *criterios de decisión* definidos en el diseño de la evaluación).

7.1. Análisis de los resultados

Puede observarse rápidamente, en el resultado global de la evaluación de *Calidad en uso*, que se produjo una mejora significativa: de un valor *marginal* de *64.3* (ver Tabla 5) se llegó a un valor *satisfactorio* de *82.6*.

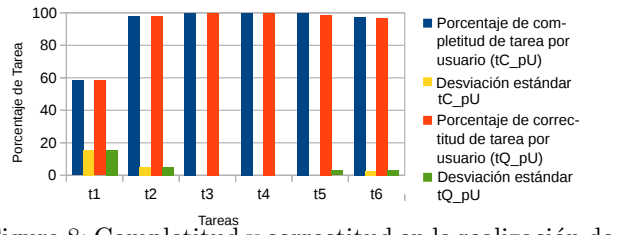


Figura 8: Completitud y correctitud en la realización de las tareas considerando los sub-objetivos para *jGUIAr 1.1*.

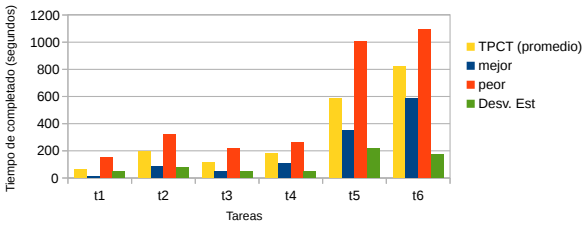


Figura 9: Tiempos de completado de las tareas para *jGUIAr 1.1*.

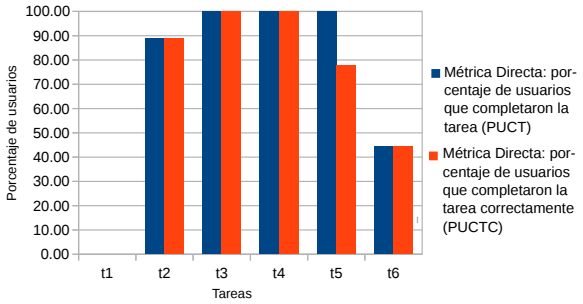


Figura 10: Completitud y correctitud en la realización de las tareas para *jGUIAr 1.1*.

Analizando las causas de este incremento, podemos observar que la mejora se produjo, en mayor o menor proporción, en todos los elementos o componentes de los requerimientos no funcionales. Particularmente relevantes resultan los incrementos en los atributos asociados a las características *1.1 Efectividad* y *1.2 Eficiencia*.

En el primer caso, se debió a una leve mejora en el atributo *1.1.1 Task effectiveness*, relacionado a la *efectividad de tareas respecto de completitud y correctitud*, considerando la cantidad de sub-objetivos de las tareas completadas por los usuarios. En el gráfico de la Figura 8 se representan los valores obtenidos por las métricas utilizadas indirectamente para cuantificar el atributo *1.1.1*. Aquí podemos observar que, exceptuando la tarea 1, las tareas 3 a 5 fueron totalmente completadas por los usuarios (*"Porcentaje de completitud de tarea por usuario (tC_pU)"*), con algunos errores para la tarea 5; la tarea 2 y 6 fueron completadas correctamente (*"Porcentaje de correctitud de tarea por usuario (tQ_pU)"*) en, al menos, un 96 % de los sub-objetivos.

En el caso de la característica *1.2 Eficiencia*, se produjo una mejora más significativa para los dos atributos asociados, ambos dependiendo del *tiempo utilizado para completar la tarea*. Como se observa en el gráfico de la Figura 9, los tiempos de completado de todas las tareas bajaron respecto de la primer evaluación, destacándose la mejora en los tiempos de la tarea 4, en la que se incorpora el uso de *layouts*, que en la evaluación anterior presentaba un incremento significativo del tiempo necesario para completarla.

Por otro lado, la *eficiencia* depende del logro de los sub-objetivos previstos para cada tarea, considerando la completitud (atributo 1.2.1) y la correctitud (atributo 1.2.2) con la que fueron realizadas por parte de los usuarios. En la Figura 10 se grafican los valores obtenidos por las métricas directas *"Porcentaje de usuarios que completaron la tarea"* y *"Porcentaje de usuarios que completaron la tarea correctamente"*, donde se observa una mejora importante en la cantidad de usuarios que completaron las tareas 2 a 5, y, en menor magnitud, en la tarea 6. Por el contrario, ningún usuario completó la tarea 1, a diferencia de la primer evaluación donde un 20 % de los usuarios completaron la tarea.

Respecto de la evaluación obtenida para la característica 2. Satisfacción, se ha obtenido el máximo puntaje en todos sus atributos.

A la vista de los resultados obtenidos se puede sostener que los principales problemas de *Calidad en uso* fueron solucionados en gran medida, principalmente, aquellos relacionados a: la demora en acceder a la funcionalidad *"Acomodar componentes"*, relacionada al uso de *layout managers*; la omisión de observar el código fuente de la interfaz siendo editada, y la omisión de compilar la aplicación para observar la interfaz gráfica creada. En segundo lugar, pero no menos importante, el agregado de la barra de herramientas para la creación de componentes y *layouts*, así como la posibilidad de desagregar componentes de la estructura de la interfaz, han permitido reducir el tiempo de edición.

La incorporación de ayuda detallada por tipo de componente ofrece otro elemento importante en la mejora de la evaluación, ya que provee un mecanismo de auto-aprendizaje y descubrimiento de la funcionalidad de la herramienta.

Aún persisten ciertos problemas que influyeron en el desempeño al completar las tareas 1 y 6. En el primer caso –la tarea de explorar la configuración de la interfaz de ejemplo-, se volvió a experimentar una amplia falta de completitud de los sub-objetivos planteados. Dado que los mecanismos para acceder a la información de los componentes fueron utilizados en las tareas subsiguientes, es posible que se deba a un problema en la

formulación de la tarea propuesta al usuario. En el caso de la tarea 6, se observan demoras en la creación de los componentes y la decisión del *layout* a utilizar, por lo que es posible que se deba, por un lado, a una cuestión de experiencia en el uso de este aspecto de $Java^{TM}$, y por otro lado, a la cantidad de sub-objetivos que requiere la misma para ser completada.

Para complementar estos resultados, y a diferencia de la primer evaluación, se solicitó a los usuarios, una vez completado el cuestionario de satisfacción, escribir las apreciaciones y recomendaciones que tuviera para la herramienta, expresándose de forma libre. De estos comentarios, 5 usuarios expresaron que el uso de la herramienta les facilitó la comprensión y aprendizaje de los conceptos de interfaces gráficas, además de destacar que les resultaba una guía para construir una interfaz, facilitando la correctitud del código, y poder obtener rápidos resultados de los cambios que realizaban. Además, los usuarios mencionaron funcionalidades que creen serían útiles disponer para facilitar la creación de interfaces gráficas, que se resumen en las siguientes:

- Utilizar valores por defecto para diferentes campos obligatorios (tales como identificadores y tamaños de componentes),
- Incorporar una opción para los contenedores de desagregar todos sus componentes,
- Permitir el agregado de múltiples componentes de una vez a los contenedores.
- Pensar una forma alternativa de asignar las restricciones a un *GridBagLayout* (uno de los layouts más complejos del paquete *AWT*).
- Utilizar textos flotantes informativos para los campos correspondientes a los parámetros de creación de los componentes, sobre tipo o formato válido,
- Implementar la función de deshacer la eliminación de componentes.
- Incorporar un asistente para la creación de múltiples botones.
- Incluir la opción de crear menús (componentes Menu, MenuItem y MenuBar).

8. Consideraciones finales y trabajo futuro

Como resultado de este trabajo, se han logrado mejorar aspectos relacionados a la *Calidad en uso* de una herramienta didáctica de reciente creación, diseñada para asistir en el proceso de enseñanza y aprendizaje de los conceptos y mecanismos involucrados en la creación de interfaces gráficas en $Java^{TM}$, mediante la librería de AWT (Abstract Window Toolkit).

Cabe destacar la importancia de haber utilizado una estrategia de mejora estructurada como *SIQinU* ya que permitió descomponer e identificar de forma detallada y explícita los elementos o aspectos que afectan a la calidad en uso de la herramienta (mediante el uso del modelo conceptual C-INCAMI) y, además, definir mecanismos de medición y evaluación claros que permitieron guiar un proceso repetible y consistente, base fundamental para emprender el proceso de mejora.

Este proceso debe llevarse adelante cuidando de mantener la consistencia y coherencia durante todas las etapas para permitir un seguimiento bidireccional de los requerimientos no funcionales identificados, pasando por los valores medidos, su evaluación e interpretación, hasta los cambios propuestos e implementados. Esto permite determinar cómo dichos cambios influyen o afectan a los requerimientos no funcionales identificados.

Además del uso del proceso de mejora, también se destaca la importancia de contar con heurísticas de usabilidad, tanto genéricas como propias del dominio, ya que los cambios concretos hacia la mejora deben ser propuestos por un evaluador, más allá de las actividades especificadas por el proceso. En este caso, estos cambios son particularmente importantes ya que afectarán los mecanismos de enseñanza y aprendizaje de un cuerpo de conocimientos no trivial.

Como se planteó al comienzo de este artículo, resulta de gran importancia realizar evaluaciones de *Calidad en uso* de herramientas didácticas, particularmente cuando se tratan de nuevos desarrollos. Estas evaluaciones ofrecen una perspectiva de cómo los estudiantes incorporan y procesan los nuevos conocimientos a través de la herramienta, lo que podría ofrecer herramientas para ajustar el proceso de enseñanza de tales conocimientos.

El próximo paso en el desarrollo de este trabajo será la implementación de los cambios propuestos en la Sección 7 y, posteriormente, la realización de una nueva evaluación de *Calidad en uso* para comprobar que los cambios incorporados no hayan afectado de forma negativa el desempeño de los usuarios en el uso de la herramienta. Con estos resultados se pretende incorporar a *jGUIAr* a la asignatura en la cual se originó. Finalmente, la herramienta se pondrá a disposición de la comunidad interesada junto con los resultados de estas evaluaciones.

Referencias

[1] Ben-ari, M and Ragonis, N. and Ben-bassat Levy, R.: A Vision of Visualization in Teaching Object-

Oriented Programming, In Proceeding of 2nd Program Visualization Workshop, pp. 83-89, 2002.

[2] Dujmovic, J.: A Method for Evaluation and Selection of Complex Hardware and Software Systems. En CMG 96 Proceedings, (The Computer Measurement Group, Inc.), págs. 368–378, 1996.

[3] ISO/IEC CD 25010.3: Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models, 2009.

[4] Jiménez C., Leoncio and Zaraté, Pascale and Vidal, Elizabeth: Analogías Gráficas como Método de Aprendizaje en un Curso de Programación Orientada a Objetos, Revista Ingeniería Informática, edición Nro. 13, Noviembre, ISSN: 0717–4195, 2006.

[5] Kölling, M. and Rosenberg, J.: Guidelines for Teaching Object Orientation with Java, Proceedings of the 6th conference on Information Technology in Computer Science Education, (ITiCSE 2001), pp. 33–36. Canterbury, 2001.

[6] Lew, P. and Olsina, L. and Becker, P. and Zhang, L.: An Integrated Strategy to Systematically Understand and Manage Quality in Use for Web Applications. Requirements Engineering Journal, Springer London, Vol.17, No. 4, pp. 299-330, 2011.

[7] Molina, H. and Olsina, L.: Evaluando la Calidad en Uso de una Herramienta Didáctica para Crear Interfaces Gráficas en *Java*^{TM}. Memorias del 3er Congreso Nacional de Ingeniería Informática / Sistemas de Información (CONAIISI 2015). Buenos Aires, Argentina Noviembre 2015. ISBN 978-987-1896-47-9.

[8] Molina, H. and Olsina, L.: Diseñando la Evaluación de Calidad en Uso de una Herramienta Didáctica para Crear Interfaces Gráficas en *Java*^{TM}. Anales del 15º Simposio Argentino de Ingeniería de Software. Jornadas Argentinas de Informática e Investigación Operativa (43JAIIO). Buenos Aires, Argentina. ISSN: 1850-2792. XLIII Sept. 2014.

[9] Molina, H.: jGUIAr: una herramienta para la enseñanza y el aprendizaje de interfaces gráficas en JavaTM. Actas del 1er Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI). Red de carreras de Ingeniería Informática / Sistemas de Información del CONFEDI (RIISIC), Córdoba, CD-ROM UTN-FRC, pp. 11, ISSN: 2346-9927, 2013.

[10] Molina, H. and Papa, F. and Martín, M. de los A. and Olsina, L.: "Semantic Capabilities for the Metrics and Indicators Cataloging Web System". In: Engineering Advanced Web Applications, Matera M. and Comai S. (Eds.), Rinton Press Inc., US, pp. 97-109, ISBN 1-58949-046-0. 2004.

[11] Nielsen, J. How Many Test Users in a Usability Study? Junio 4, 2012. Nielsen Norman Group. Online: `http://www.nngroup.com/articles/how-many-test-users/`.

[12] Olsina, L. and Lew, P. and Dieser, A. and Rivera B.: Updating Quality Models for Evaluating New Generation Web Applications. In Journal of Web Engineering, Special issue: Quality in new generation Web applications, Abrahão S., Cachero C., Cappiello C., Matera M. (Eds.), Rinton Press, USA, 11 (3), pp. 209-246, 2012.

[13] Olsina, L. and Papa, F. and Molina, H.: How to Measure and Evaluate Web Applications in a Consistent Way. Chapter 13 in Springer book, HCIS: Web Engineering: Modeling and Implementing Web Applications, Rossi G., Pastor O., Schwabe D. and Olsina L. (Eds), Springer-Verlag, London pp. 385–420, ISBN 978-1-84628-922-4, 2008.

[14] Rubin, J. and Chisnell, D. Handbook of Usability Testing, Second Edition (2008). Wiley ISBN: 978-0470185483.

[15] Uehling, D. L.: Usability Testing Handbook. DSTL-94-002, Data Systems Technology Laboratory Series, NASA/Goddard Space Flight Center, 1994.