Diseñando la Evaluación de Calidad en Uso de una Herramienta Didáctica para Crear Interfaces Gráficas en *Java™*

Hernán Molina y Luis Olsina

GIDIS Web, Facultad de Ingeniería,
Universidad Nacional de La Pampa, Argentina
hmolina.ing.unlpam@gmail.com,olsinal@ing.unlpam.edu.ar

Resumen En este artículo se describe el diseño de la evaluación de *Calidad en Uso* que se llevará a cabo sobre la herramienta *jGUIAr*, diseñada y creada para asistir en la enseñanza y aprendizaje de la construcción de interfaces gráficas con el paquete AWT del lenguaje *Java™*. La herramienta será utilizada en el contexto de una cátedra de *Programación Orientada a Objetos* para introducir al estudiante en la creación de interfaces gráficas. La evaluación, cuyo objetivo es mejorar la herramienta para su utilización en dicho contexto, fue diseñada siguiendo una estrategia de medición y evaluación (SIQinU) y partiendo de un modelo de calidad de referencia relacionado (2Q2U).

Palabras Clave: Calidad en Uso, Medición, Evaluación, Herramienta didáctica, Interfaz Gráfica de Usuario.

1. Introducción

El uso de herramientas didácticas constituye un recurso muy importante en la enseñanza y aprendizaje de nuevos conceptos y paradigmas de programación, ya que ofrecen un acercamiento visual a los elementos que constituyen el cuerpo de conocimientos a incorporar [1,4], a la vez que facilitan la asimilación de nuevas capacidades mediante la experimentación.

El caso de la enseñanza y aprendizaje de los conceptos y capacidades asociadas a la construcción de interfaces gráficas constituye uno de los escenarios ideales para el uso de estas herramientas, tal como ocurre en la asignatura *Programación Orientada a Objetos* de las carreras de informática de la *Facultad de Ingeniería* de la *Universidad Nacional de La Pampa*. En dicha asignatura, los estudiantes incorporan por primera vez los conceptos y experiencias de la programación orientada a objetos, utilizando el lenguaje de programación *Java™*, así como de la creación y programación de interfaces gráficas –en el mismo lenguaje. Para asistir al estudiante en este último aspecto, se ha desarrollado –a fines de 2013- una herramienta, llamada *jGUIAr (*Java™ *Graphical User Interface Architect)* [8]. Esta herramienta permite construir una interfaz gráfica

–utilizando componentes de la librería AWT[1]- de forma lógica, a diferencia del enfoque WYSIWYG[2] en el que el usuario *"dibuja"* los componentes en un lienzo que representa la ventana de la aplicación final. En dicho enfoque, se ocultan los detalles del código fuente que genera la interfaz gráfica dibujada y, en muchos casos, resulta sobrecargado de ajustes –necesarios para producir el diseño realizado-, dificultando la comprensión a programadores novatos. En *jGUIAr*, la interfaz gráfica se compone mediante el uso de una metáfora gráfica conocida que representa la agregación de componentes en una interfaz gráfica (ver Figura 1), facilitando la comprensión de cómo se relacionan los diferentes componentes para conformar la interfaz. También es posible ver el código fuente que genera la interfaz gráfica construida –que se mantiene libre de ajustes no solicitados- y ejecutar la aplicación para ver el resultado real. De esta forma, la herramienta también permitiría reducir el tiempo transcurrido entre el diseño y la comprobación de la interfaz gráfica resultante, haciendo posible la rápida experimentación de las diferentes configuraciones de los componentes gráficos. Con todo esto, *jGUIAr* pretende acortar la brecha existente entre el código fuente y la interfaz gráfica resultante, facilitando al estudiante establecer una relación entre ambos.

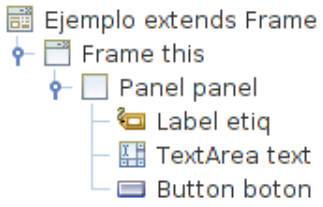

Figura 1. Estructura que representa la agregación de componentes de la interfaz diseñada en *jGUIAr*.

No obstante, la incorporación de *jGUIAr* como herramienta didáctica en la asignatura mencionada debe realizarse con ciertas previsiones. Concretamente, se debe tener la certeza de que posee los niveles de usabilidad y satisfacción necesarios para garantizar que su uso produzca los resultados esperados y que, a la vez, no entorpezca el proceso de enseñanza y aprendizaje del cuerpo de conocimientos correspondiente. Por tal motivo, y como se adelantó en [8], se llevará a cabo una evaluación de *Calidad en Uso* con usuarios reales para determinar la conformidad de la herramienta con un conjunto de requerimientos no funcionales diseñados para el propósito de la misma. Dicha evaluación se realizará siguiendo la estrategia de medición y evaluación *SIQinU (Strategy for Improving Quality in Use)* [7] y el modelo de calidad *2Q2U (Quality, Quality in use, actual Usability and User experience)* [10], debido a que constituyen un enfoque integrado que incluye todos los elementos necesarios. En este artículo se presenta el diseño de la evaluación que se llevará a cabo, siguiendo el enfoque y modelo mencionados, quedando pendiente la implementación de la misma, a

[1] La librería *"Abstract Window Toolkit"* permite utilizar componentes nativos a la plataforma

[2] *"What You See Is What You Get"*

realizarse durante la segunda mitad del primer semestre del año 2014, durante el dictado de la asignatura *Programación Orientada a Objetos* de las carreras antes mencionadas.

El resto del artículo se estructura de la siguiente forma: en la siguiente sección se describen brevemente los elementos que conforman la estrategia *SIQinU*; en la Sección 3 se presenta el diseño de la evaluación de *Calidad en Uso* de la herramienta *jGUIAr*, explicitando las diferentes especificaciones realizadas; finalmente, en la Sección 4 se exponen las consideraciones finales, así como trabajos relacionados y futuros.

2. Estrategia Integrada de Medición y Evaluación: SIQinU

La evaluación de la herramienta *jGUIAr* se lleva a cabo siguiendo una estrategia de medición y evaluación, llamada *SIQinU*, diseñada para comprender y mejorar requerimientos de calidad de software de forma iterativa [7]. *SIQinU* se construye sobre otra estrategia de medición y evaluación de propósito general, llamada *GOCAME (Goal-Oriented and Context-Aware Measurement and Evaluation)*, compuesta por tres elementos consistentemente integrados: (i) un marco conceptual, (ii) un proceso y (iii) métodos y herramientas instanciadas para los dos primeros. *SIQinU* reutiliza todos estos elementos, extendiendo la definición del proceso –e instanciando modelos y métodos- para incorporar actividades específicas de mejora y retroalimentación sucesiva de los resultados de evaluación con el objetivo de obtener nuevas versiones del producto que reflejen un mayor grado de satisfacción de los requerimientos de *Calidad en Uso* definidos. A continuación se describen brevemente los tres componentes que constituyen esta estrategia.

2.1. Marco Conceptual y Modelo de Calidad

La repetitividad y consistencia de la estrategia *SIQinU* se apoya en la especificación explícita y consistente de las actividades a llevar a cabo gracias al uso del marco conceptual *C-INCAMI (Contextual Information Needs, Concept Models, Attributes, Metrics and Indicators)* [11], que define de forma estructurada y explícita los conceptos involucrados en las actividades de medición y evaluación de requerimientos no funcionales en proyectos de software y web. El marco C-INCAMI sigue un enfoque *orientado a objetivos* en el que el diseño e implementación de la medición y la evaluación se enfocan en satisfacer una necesidad de información especificada inicialmente. Además, el marco es *sensible al contexto* ya que permite especificar de forma clara y estructurada los metadatos y datos que describen el contexto relevante en el que se llevan a cabo las actividades de medición y evaluación. Adicionalmente C-INCAMI está centrado en la organización, ya que la definición de los conceptos involucrados se realiza en el marco de proyectos que una organización gestiona, contemplando aspectos de reusabilidad y consistencia de dichas actividades.

Los conceptos del marco C-INCAMI están organizados en cuatro módulos principales (ver Figura 2):

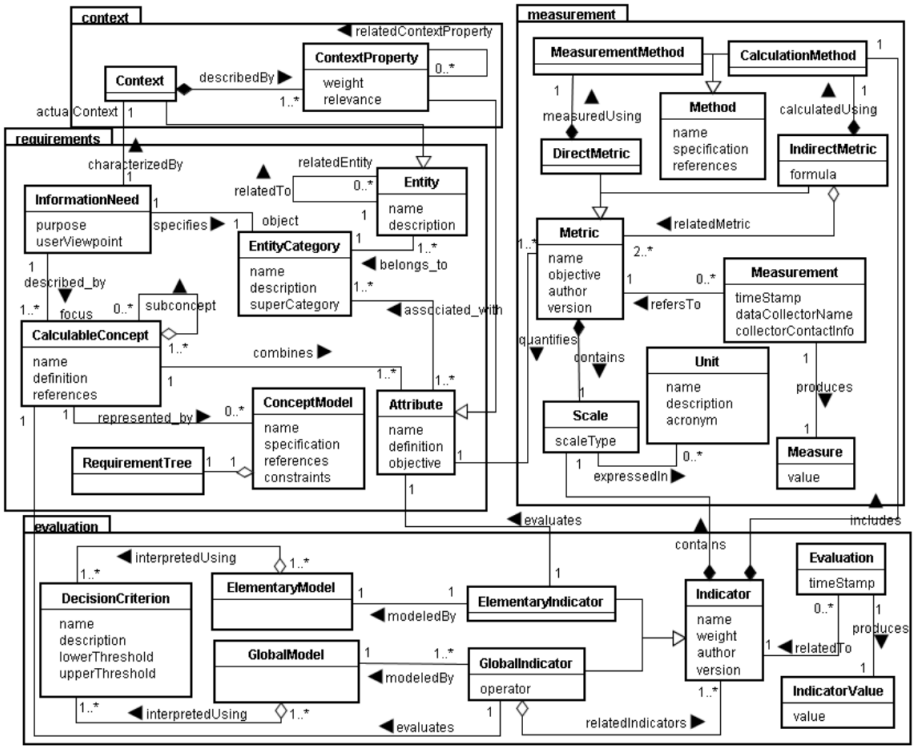

Figura 2. Principales Componentes del Marco Conceptual C-INCAMI.

- el *módulo de requerimientos no funcionales* trata con la definición de la necesidad de información (*InformationNeed*) para diversas categorías de entidad (*EntityCategory*) y la especificación de los requerimientos asociados, representados en modelos de concepto (*ConceptModel*) mediante la agrupación de conceptos calculables (*CalculableConcept*) y atributos cuantificables (*Attribute*) de las entidades. De esta forma, es posible instanciar, por ej., modelos de calidad interna, *Calidad en Uso*, entre otros. Estos modelos guían las actividades posteriores de medición y evaluación;
- el *módulo de contexto* incluye conceptos y relaciones que permiten especificar el contexto relevante en el que se lleva a cabo la medición y evaluación así como el contexto en el que son aplicables las diferentes especificaciones reutilizables del marco C-INCAMI (tales como Modelos de Concepto, Métricas, entre otros). Las especificaciones de contexto (*Context*) se describen mediante propiedades de contexto (*ContextProperty*) tomadas del dominio de aplicación y cuantificadas por medio de métricas apropiadas;

- el *módulo de medición* trata con la especificación de las entidades (*Entity*) concretas a ser medidas, la selección de las métricas (*Metric*) –de un catálogo de especificaciones reutilizables [9]- para cuantificar los atributos de los modelos de concepto seleccionados y el registro de las mediciones obtenidas;
- el *módulo de evaluación*, trata con la definición de los indicadores (*Indicator*) que interpretarán los modelos de concepto seleccionados, el diseño de los criterios de decisión (*DecisionCriteria*) y del modelo de agregación para el cálculo global (*GlobalModel*) que proveerá una interpretación al concepto foco de la evaluación para cada una de las entidades medidas.

SIQinU propone un marco de modelos de calidad llamado *2Q2U* [10], instanciando el término *ConceptModel* de C-INCAMI. Este marco, propuesto como una extensión y mejora a los modelos de calidad del estándar ISO 25010 [3], fue diseñado para cubrir los aspectos distintivos de las aplicaciones web. No obstante, es lo suficientemente flexible para ser adaptado a la evaluación de aplicaciones de software tradicionales. El modelo *2Q2U* cubre tres vistas de calidad: *Calidad Interna*, *Calidad Externa* y *Calidad en Uso* (ver Figura 3) y contempla las dependencias e influencias que existen entre las características de las diferentes vistas, en concordancia con el proceso definido por la estrategia (descripto a continuación).

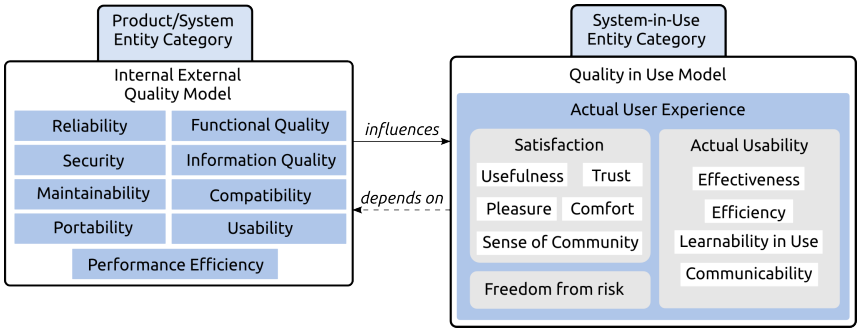

Figura 3. Vistas de calidad y características principales del modelo de calidad 2Q2U.

2.2. El Proceso de SIQinU

Como se introdujo previamente, el proceso definido para la estrategia *SIQinU* se enfoca en la mejora incremental y continua de la *Calidad en Uso* de un sistema de software, guiada por la medición y evaluación –destinadas a entender el nivel de satisfacción de los requerimientos no funcionales- y la implementación de acciones correctivas correspondientes.

El proceso utiliza *2Q2U* para instanciar modelos de *Calidad Externa* y *Calidad en Uso*, mapeando cuestiones de uso real con atributos medibles de *Calidad Externa* (inherentes al producto de software). Se diseñan las tareas de medición y evaluación de forma consistente y repetible, según el marco conceptual *C-INCAMI*, para evaluar la satisfacción de los requerimientos de *Calidad Externa* y de *Calidad en Uso*. En función de los resultados, y de forma iterativa

e incremental, se efectúan los cambios necesarios al software –siguiendo métodos y técnicas específicas de la estrategia- y se re-evalúa la satisfacción de los requerimientos, verificando la mejora hasta alcanzar los niveles deseados.

El proceso de *SIQinU* se estructura en seis fases principales (Figura 4), descriptas brevemente a continuación:

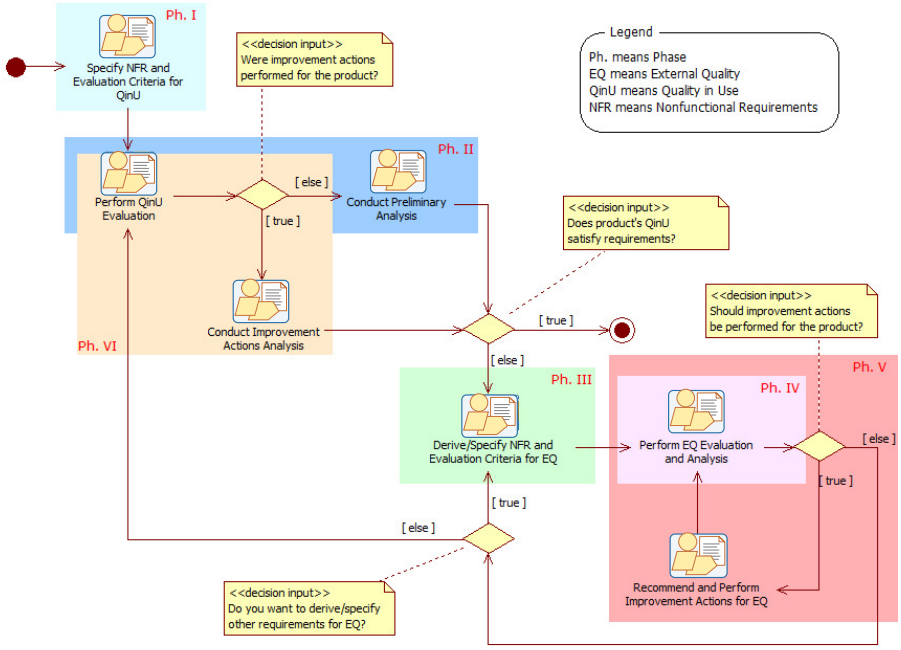

Figura 4. Panorama del proceso de *SIQinUs*.

1. *Especificar requerimientos y criterios de evaluación para Calidad en Uso*: involucra diseñar tareas, definir tipo de usuario, especificar contexto de uso y requerimientos no funcionales (características y atributos) para *Calidad en Uso*. A partir de aquí, se seleccionan métricas e indicadores.
2. *Realizar evaluación y análisis de Calidad en Uso*: se recolectan los datos de uso del producto (*logs*), obtenidos durante la realización de las tareas diseñadas en la fase anterior, para los requerimientos de *Calidad en Uso* 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 requerimientos y criterios de evaluación para Calidad Externa* (si fuera necesaria la mejora) en función de 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. *Llevar a cabo la evaluación y análisis de Calidad Externa.* A partir de las especificaciones de la fase anterior, se realiza la medición y evaluación basado en 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 producto 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 satisfacción en función de los cambios realizados para satisfacer *Calidad Externa* (fase 5). Si fuera necesaria la mejora, se continúa en la fase 3.

2.3. Métodos y Herramientas

SIQinU integra el método *WebQEM (Web Quality Evaluation Method)* [11] junto a la herramienta *C-INCAMI_Tool* que da soporte al mismo y al marco conceptual *C-INCAMI*. El método *WebQEM* especifica cómo llevar a cabo la especificación de requerimientos funcionales, el diseño y la ejecución de la medición y la evaluación.

3. Diseño de la Evaluación de la Herramienta jGUIAr

Como se introdujo al principio del artículo, la herramienta *jGUIAr* será sometida a una evaluación de *Calidad en Uso* para determinar si satisface un conjunto de requerimientos para ser utilizada como herramienta didáctica para la enseñanza y aprendizaje en la construcción de interfaces gráficas con *AWT* de *Java*^{TM}, en el contexto de la asignatura *Programación Orientada a Objetos*, en la cual el estudiante es introducido al tema por primera vez.

La evaluación consiste de (i) una prueba con usuarios reales (estudiantes de la asignatura mencionada) para recabar los datos necesarios, (ii) la evaluación y análisis de los mismos para determinar si se requieren cambios en la herramienta y, (iii) proceder con los mismos (si fueran necesarios) y realizar una nueva evaluación para determinar la mejora. La evaluación se lleva a cabo siguiendo la estrategia *SIQinU*. Para los aspectos específicos de la prueba de usuarios (tales como definición de tareas, selección de métricas, diseño de la sesión de prueba, entre otros), se utilizó como guía el *Manual de Pruebas de Usabilidad*[3] [13].

3.1. Definición de Tareas

Como primer paso (fase 1 de *SIQinU*) se definieron las tareas (con sus objetivos y sub-objetivos) que se espera que los usuarios puedan completar con

[3] Desarrollado para la División de Tecnología de Sistemas de Datos del Centro de Vuelo espacial Goddard

la misma. Para ello se tuvieron en cuenta, además de la guía mencionada, los lineamientos presentados en [5], basados en la filosofía de enseñanza de la programación *Orientada a Objetos* presentada en el mismo trabajo. Así, las sucesivas tareas deben responden a diferentes niveles de dificultad, incorporando nuevos conceptos y desafíos de forma iterativa e incremental:

1. *Explorar*: se pide al usuario que explore los componentes y configuraciones de una interfaz gráfica simple, previamente creada con la herramienta.
2. *Configurar*: se pide al usuario que haga modifique los componentes de las interfaces gráficas dadas y vea los cambios resultante en la interfaz .
3. *Agregar componentes*: se pide al usuario que modifique los ejemplos provistos agregando nuevos componentes a la interfaz gráfica.
4. *Cambiar disposición*: se pide al usuario que modifique los ejemplos provistos cambiando la disposición de los componentes existentes para obtener una interfaz gráfica más apropiada y mejor diseñada que la original.
5. *Agregar contenedores y componentes*: se pide al usuario que agregue nuevos contenedores y componentes, utilizando la disposición y configuración que considere más apropiada al propósito solicitado.
6. *Crear interfaz*: se pide al usuario que cree una interfaz gráfica desde cero a partir de una imagen de muestra.

A modo ilustrativo, se describe a continuación la primer tarea diseñada (correspondiente al nivel de dificultad *Explorar*):

> *Tarea 1. Objetivo: Explorar la configuración de los componentes incluidos en la interfaz* Default.jgui. *Analizar el código fuente generado y observar la interfaz resultante en cada caso.*
>
> *Sub objetivos:*
>
> 1. *Editar Frame* `this`
> 2. *Editar Panel* `panel`
> 3. *Editar TextArea* `text`
> 4. *Editar Label* `etiq`
> 5. *Editar Button* `boton`
> 6. *Ver código fuente*
> 7. *Compilar y ejecutar aplicación*

3.2. Especificación de la Necesidad de Información

A continuación, se procede con el diseño de la evaluación, siguiendo el proceso de *SIQinU*, según el cual se realiza la especificación de la *necesidad de información* que guiará el resto de las actividades. Siguiendo el marco conceptual C-INCAMI, la necesidad de información se especifica como se muestra en la Tabla 1.

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

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

3.3. Especificación del Contexto

Continuando con el proceso, debe especificarse el contexto en el que se lleva a cabo la evaluación. Para ello se identifican las entidades y sus propiedades relevantes a la situación, es decir, que afectan al uso de la herramienta, al proceso de medición y evaluación o a la interpretación de los resultados del mismo. Cada propiedad de contexto se cuantifica y especifica como un atributo medible (a partir de una métrica a partir de la cual se obtienen medidas –ver Figura 2). A continuación se listan algunas de las propiedades de contexto identificadas, indicando los valores que puede asumir (en caso de utilizar una escala categórica) y, en negrita, el valor real.

- **Usuarios**
  - Tipo de usuario=**programador**
  - Nivel de experiencia=[ninguna | **estudiante** | junior | senior]
  - 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** | 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***^{TM};
  - Categoría de entidad=**herramienta didáctica de software**;
  - Dominio=**interfaces gráficas**;
  - Etapa de versión entregable=[alfa | **beta**[4] | estable]
- **Ambiente**
  - Lugar de la prueba=**centro de cómputos**
  - Privacidad=[**espacio dedicado** | espacio compartido]
  - Pluralidad de usuarios=[individual | **grupal**]
- **Proceso de prueba**
  - Tiempo de la prueba=**1 hora**
  - Tipo de registro=[intrusivo | **no intrusivo**]
  - Técnica de medición=**archivos de registro**
  - Conocimiento de la técnica de medición por parte del usuario=[**consciente** | no consciente]
  - Cantidad de observadores=**4**
  - Cantidad de usuarios por tarea=**1**

También se registra como propiedad de contexto para **Tarea**, la *estructura* –con valor "`estructurada`"- y el *nivel de dificultad* de cada tarea, tal como se describió anteriormente. Además, se registran las características de *hardware* de las computadoras donde se ejecuta la herramienta, aunque aquí se omiten por cuestiones de espacio.

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

Tabla. 2. Especificación de propiedad de contexto *Etapa de versión entregable*.

Nombre	**Etapa de versión entregable**	**Categoría de Entidad**	Aplicación de Software
Definición	Etapa que describe la estabilidad de una pieza de software entregable a medida que progresa su ciclo de vida.	**Propiedad de Contexto relacionada**	–
Relevancia	Puede afectar la interpretación de resultados ya que en diferentes etapas del entregable pueden observarse diferentes niveles de calidad.	**Escala Categorica (Valores Categóricos)**	[Alpha | Beta | Estable]
		Multiplicidad de valores	1
Prop. URI	`http://myorg.com/context/SoftwareReleaseLifeCycleStage`	**Medidas (Valor)**	Beta

Para cada propiedad de contexto seleccionada debe proveerse una fundamentación de su relevancia para la situación, como parte de su especificación, tal como se define en el modelo de C-INCAMI. A modo ilustrativo, en la Tabla 2 se presenta una especificación más completa de la propiedad de contexto *Etapa de versión entregable*, listada previamente.

3.4. Diseño de Requerimientos No Funcionales

A continuación, en base a la necesidad de información definida, y teniendo en cuenta el contexto especificado, se diseñan los requerimientos no funcionales de *Calidad en Uso* deseables para la herramienta, tomando como punto de partida el modelo de calidad 2Q2U [10] (ver Figura 5). Algunas características del modelo original no fueron incluidas –tales como *Sense of community* y *Freedom from risk*- por no considerarse aplicables al contexto actual.

El modelo de requerimientos no funcionales incluye, además, los atributos medibles relacionados a cada característica (indicados en la figura mencionada con una letra "A" como prefijo al código del atributo), que serán el punto de partida para calcular indicadores de satisfacción para los conceptos calculables de alto nivel.

El modelo resultante incluye dos conceptos o características principales: *Usabilidad Real* (*Actual Usability*), definido como *"grado con el que usuarios especificados pueden lograr objetivos específicos con efectividad, eficiencia, facilidad de aprendizaje en uso (learnability in use) y sin brechas comunicacionales en un contexto de uso especificado"* [10]; y *Satisfacción*, definido como *"grado con el cual las necesidades del usuario son satisfechas cuando un producto es usado en un contexto de uso especificado"* [3].

3.5. Instrumentos de Medición

Continuando con el proceso definido para *SIQinU*, se establecen los mecanismos de medición y métricas asociadas que permitirán cuantificar los atributos incluidos en los requerimientos no funcionales de *Calidad en Uso*.

El primero de los mecanismos diseñados es el registro automático de determinados eventos en la herramienta que permitirán cuantificar los atributos

```
Quality in use (sinon. User Experience)
1. Actual Usability (sinon. Usability in use)
    1.1 Effectiveness
        A1.1.1 Tasks effectiveness
    1.2 Efficiency
        A1.2.1: Tasks efficiency on completeness
        A1.2.2: Tasks efficiency on effectiveness
2. Satisfaction
    2.1 Universality
        A2.1.1 User satisfaction on universality
    2.2 Usefulness
        A2.2.1 User satisfaction on usefulness
    2.3 Learnability in use
        2.3.1 Communicability
            A2.3.1.1 User satisfaction on communicability
        2.3.3 Readability
            A2.3.3.1 User satisfaction on readability
        2.3.4 Navigability
            A2.3.4.1 User satisfaction on navigability
        2.3.5 Familiarity
            A2.3.5.1 User satisfaction on familiarity
        2.3.6 Appropriateness
            A2.3.6.1 User satisfaction on appropriateness
```

Figura 5. Requerimientos no funcionales de *Calidad en Uso* diseñados para la evaluación de *jGUIAr*.

```
11/30|10:30:24|Se abrió una interfaz gráfica|--> Default.jgui desde /home/hernan/jGUIAr/samples/
11/30|10:30:30|Aplicacion obtiene foco
11/30|10:30:31|Se va a editar objeto|--> Frame this [BorderLayout]
11/30|10:30:33|Se cancela la edición del objeto|--> Frame this [BorderLayout]
11/30|10:30:33|Se termina edición
11/30|10:30:35|Se va a editar objeto|--> Panel panel [FlowLayout]
11/30|10:30:37|Se cancela la edición del objeto|--> Panel panel [FlowLayout]
11/30|10:30:37|Se termina edición
11/30|10:30:39|Se va a editar objeto|--> Label etiq
11/30|10:30:41|Se cancela la edición del objeto|--> Label etiq
11/30|10:30:41|Se termina edición
```

Figura 6. Extracto de archivo de registro de eventos de jGUIAr realizando la Tarea 1.

relacionados a *Usabilidad Real* (*Actual Usability*), asociados a la realización de las tareas diseñadas. En la Figura 6 se muestra un extracto de un archivo de eventos registrado durante la realización de la Tarea 1 (ver Sección 3.1).

Por otro lado, para cuantificar los atributos de la característica *Satisfacción*, al igual que en [6], se diseñó un cuestionario que se le solicitará completar al usuario, de forma anónima, una vez completadas las tareas propuestas. El cuestionario incluye afirmaciones sobre la experiencia del usuario en el uso de la herramienta, cubriendo todos los atributos relacionados a las sub-características de *Satisfacción* (ver Tabla 3). Para cada afirmación, el usuario deberá indicar su grado de acuerdo en una escala *Likert* de cinco valores que van desde *"totalmente en desacuerdo"* hasta *"totalmente de acuerdo"*.

Las preguntas asociadas a *Comunicabilidad* (característica 2.3.1) fueron formuladas a partir del método propuesto en [12]. Las interpretación de las respuestas se utilizará para recomendar cambios en la interfaz de la herramienta

para solucionar los problemas de comunicabilidad encontrados, que deberán ser re-evaluados en la siguiente iteración del proceso de *SIQinU*.

Tabla. 3. Cuestionario diseñado 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.

3.6. Diseño de la Medición y Evaluación

Siguiendo el proceso definido para *SIQinU*, se procede al diseño/selección de las métricas que permitirán cuantificar los atributos incluidos en los requerimientos de *Calidad en Uso* (ver Figura 5). En el caso de asignar, a dichos atributos, métricas indirectas (que dependen de otras métricas), deberán incluirse las métricas de las que depende, junto a los atributos que cuantifican. En el diseño de la medición de *jGUIAr*, para el atributo *Eficiencia de tareas respecto de completitud* (codigo 1.2.1) se asignó la métrica *Nivel de eficiencia de tareas respecto de completitud (ETTC)*, cuyo cálculo depende de otra métrica que cuantifica el atributo *Eficiencia de tarea respecto de completitud* (no incluido en los requerimientos especificados). En la Tabla 4 se muestra la dependencia de estas métricas relacionadas, incluyendo la escala y método (o procedimiento) de cálculo o medición correspondiente. El diseño de la medición se completa cuando todas las métricas que cuantifican los requerimientos no funcionales hayan sido expresadas en función de métricas directas. Las mediciones correspondientes a estas últimas, se obtienen del procesamiento de los registros de eventos de la ejecución de las tareas por parte de los usuarios.

Posteriormente, se procede al diseño de los indicadores que permitirán calcular e interpretar valores de satisfacción para los requerimientos no funcionales.

El diseño de la evaluación involucra la definición de criterios de decisión para los valores de indicadores, el diseño de modelos elementales para calcular valores de indicadores elementales a partir de los valores medidos sobre los atributos, y la selección y diseño de modelos globales para calcular valores de indicadores globales para las características de alto nivel de los requerimientos no funcionales.

Tabla. 4. Métricas utilizadas para cuantificar el atributo 1.2.1.

Atributo 1.2.1: Eficiencia de tareas respecto de completitud **Métrica Indirecta**: Nivel de eficiencia de tareas respecto de completitud (ETTC) Escala: Numérica/Proporción; Unidad: %(usuarios)/segundos Método: Promedio de ETC entre todas las tareas
Métricas relacionadas
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
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

En el caso de la evalución de herramienta *jGUIAr*, no existían datos de evaluaciones previas sobre esta herramienta, por lo que no era posible establecer de forma precisa criterios de decisión para los resultados. Por ello, se realizó una prueba con usuarios avanzados –incluyendo al desarrollador mismo de la aplicación-, tal como se propone en [13], realizando las tareas diseñadas para los usuarios finales. De esta forma, los valores de desempeño registrados se utilizaron para establecer los criterios de decisión que permitieran interpretar los resultados de la evaluación con los usuarios finales.

El criterio de decisión diseñado para esta evaluación es el siguiente:

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

A modo de ejemplo, el indicador diseñado para el atributo *Eficiencia de tareas respecto de completitud* (código 1.2.1) incluye un modelo elemental que mapea valores medidos a partir de la métrica correspondiente (M_{ETTC}) a valores normalizados del indicador elemental (I_{ETTC}) según la siguiente ecuación:

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

Finalmente, para el modelo global se utilizará el modelo LSP [2], cuya fórmula de cálculo (ver ecuación 2) se basa en una suma ponderada (según pesos W_i) de los valores de indicadores (I_i) que permite modelar diferentes grados de preferencia o exigencia para cada componente, utilizando un parámetro (r), pudiendo variar desde un requerimiento obligatorio hasta uno opcional (donde el punto medio es equivalente a una suma ponderada simple) 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 $(W_1 + ... + W_k) = 1, W_i > 0, i = 1, ..., k..$

Finalmente, se diseña la sesión de prueba con los usuarios finales. La prueba se llevará a cabo durante una clase práctica de la asignatura, dedicándole una hora a completar las tareas y responder el cuestionario. Se comenzará explicado el procedimiento a los usuarios recordándoles que el objeto a ser evaluado es la herramienta y no ellos mismos, destacando que, tanto los registros de eventos de la herramienta como el cuestionario, son anónimos y confidenciales. Cada usuario resolverá las tareas de forma individual sin interrumpir al resto.

Se entregará una copia impresa de las tareas a realizar y el facilitador las leerá en voz alta para todos los usuarios, resolviendo dudas que pudieran surgir, únicamente sobre el dominio.

Durante la realización de las tareas, los facilitadores podrán hacer responder dudas que guíen a los usuarios en las tareas sin decirles cómo debe completarla. Además, el facilitador, cumpliendo el rol de observador, tomará nota de problemas que los usuarios tengan asociados al uso de la herramienta, como complemento para ayudar a interpretar los datos de los registros de eventos.

Una vez concluidas las tareas y la encuesta, se retirará la misma y el archivo de registro de eventos de la sesión de trabajo del usuario. Posteriormente se procederá a procesar los datos de los eventos para llevar a cabo la implementación de la medición y evaluación, continuando con las fases tal como se definen en el proceso de *SIQinU*.

4. Consideraciones Finales y Trabajos Futuros

En este artículo se ha presentado el diseño de la evaluación de *Calidad en Uso* para la herramienta *jGUIAr* (diseñada para construir interfaces gráficas en *Java*^{TM}), para determinar su grado de satisfacción para ser utilizada como instrumento didáctico en la asignatura *Programación Orientada a Objetos* de la *Universidad Nacional de La Pampa*. El diseño de la evaluación se llevó a cabo siguiendo la estrategia *SIQinU* –desarrollada por investigadores de la misma institución- que integra de forma consistente un marco conceptual, un proceso y métodos y herramientas coordinados para facilitar la mejora de productos de software (entre otros).

A lo largo del artículo se han expuesto los resultados de las etapas correspondientes al diseño de la medición y la evaluación, destacando la importancia de contar con las especificaciones explícitas de los diferentes elementos de diseño; particularmente de la especificación del contexto, aspecto clave en una evaluación de *Calidad en Uso* para una correcta interpretación de los resultados.

Más allá de los trabajos citados en este artículo, respecto de la estrategia [7] y modelos de calidad [10] para evaluación utilizados (y trabajos relacionados [6]), se encuentran, en la literatura relacionada, pocos trabajos donde se reporte el diseño de evaluaciones de calidad de herramientas didácticas. Por otro lado, en [5] y [4] se reportan herramientas y filosofías utilizadas en la enseñanza de conceptos relacionados a la programación orientada a objetos.

Posteriormente a la escritura de este artículo, se procederá a llevar adelante la sesión de prueba con los usuarios finales, los estudiantes de la asignatura men-

cionada, para luego procesar los datos e implementar la evaluación. El análisis de los resultados de dicha etapa determinará si debe realizarse una evaluación de *Calidad Externa* y recomendar los cambios pertinentes, tal como se define en el proceso de *SiQinU*.

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. Leoncio Jiménez C., Pascale Zaraté, Elizabeth Vidal, 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., Qanber A. M., Rafique I., Wang X., and Olsina L.: Using Web Quality Models and Questionnaires for Web Applications Evaluation. IEEE proced., QUATIC, pp. 20-29, (2012)
7. Lew P., Olsina L., 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)
8. 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)
9. Molina H., Papa F., Martín M. de los A., 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)
10. Olsina L., Lew P., 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)
11. Olsina L., Papa F., 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, (2008)
12. Prates R., de Souza C., Barbosa S.: A method for communicability evaluation of user interfaces. Interactions, ACM, 7(1), pp. 31-3, 2000.
13. Uehling, D. L., Usability Testing Handbook. DSTL-94-002, Data Systems Technology Laboratory Series, NASA/Goddard Space Flight Center (1994)