UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Tesis Doctoral

Marco para la Captura de Requisitos de Usabilidad en Entornos de MDD

Yeshica Isela Ormeño Ayala

Directores:

Dr. Óscar Pastor López

Dr. José Ignacio Panach Navarrete

Diciembre 2023

Marco para la Captura de Requisitos de Usabilidad en Entornos de MDD

Esta tesis fue realizada por:
Yeshica Isela Ormeño Ayala

Tutores
Dr. Óscar Pastor López (Universitat Politècnica de València)
Dr. José Ignacio Panach Navarrete (Universitat de València)

Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camìno de Vera s/n, Edificio 1F
46022, Valencia, España
Tel. (+34) 963 877 007 ext. 83533
Fax: (+34) 963 877 359
Web: http://www.pros.upv.es

Comentarios: La tesis es presentada para la obtención del grado de Doctor en Informática por la Universidad Politécnica de Valencia.

Derechos: © Yeshica Isela Ormeño Ayala, 2023.

Dedicatoria

A mi madre porque siempre ha sido el pilar que me sostuvo durante esta travesía, gracias por estar ahí cuando más te necesito.

A mis hijos por todos los momentos de ausencia, gracias por vuestra comprensión y paciencia.

Agradecimientos

Comienzo agradeciendo la generosidad de mis directores, el Dr. Ignacio Panach y el Dr. Óscar Pastor. Ignacio, gracias por guiarme y acompañarme, pero sobre todo gracias por tu paciencia y confianza que me brindabas durante todo este tiempo. Gracias Óscar por recibirme en el grupo de investigación PROS, por apoyarme y ampliar mis horizontes. Gracias a todos mis compañeros de laboratorio L104 (Nelly, Sergio, Raúl, Karolyne, Nathalie, José Luis y Yhu) y del laboratorio L204 (Ana, Urko, los Perris, Otto y María Eugenia) con quienes pase inolvidables momentos en la UPV y mis amigos (Nubia, Flabio, Ani, Raúl, Eva, Alejandro, Lili y demás integrantes del grupo el pulpo anhelado) que siempre me dieron todo su apoyo y muchos alientos para seguir adelante pase lo que pase, gracias por vuestra amistad.

Gracias a mis colegas del DAII en especial al profesor Lauro por sus sabios consejos, a Esthercita que como madre me entendía en los avatares de la vida, y a todos que de una u otra forma confiaron en mi persona y me brindaron su apoyo incondicional. Un agradecimiento especial a la Universidad Nacional de San Antonio Abad del Cusco, que a través del programa Yachayninchis Wiñarinanpaq en convenio con Concytec Fondecyt hicieron posible el financiamiento de esta tesis.

Resumen

La investigación desarrollada en esta tesis representa un marco novedoso para capturar requisitos de usabilidad durante el desarrollo de un sistema software. Estos requisitos, están representados como alternativas de diseños de Interfaces de Usuario (IU). El objetivo es desarrollar un proceso de captura de requisitos de usabilidad basado en entrevistas estructuradas con el apoyo de una herramienta que ayude a resolver problemas como: (1) la omisión de la usabilidad desde las primeras etapas de desarrollo, en general, las características de usabilidad solo se tienen en cuenta al diseñar las interfaces en las últimas etapas de desarrollo; (2) resulta tedioso la captura de requisitos para analistas que no son expertos en usabilidad; (3) los métodos y herramientas que se utilizan para desarrollar software no admiten la elicitación de requisitos de usabilidad. A partir de estos problemas encontrados en la literatura se definen las preguntas de investigación: ¿Es posible capturar requisitos de usabilidad en etapas iniciales de desarrollo al mismo tiempo que los requisitos funcionales? Para responder a esta pregunta, la tesis ha definido un método de elicitación de requisitos de usabilidad llamado UREM (por sus siglas en inglés, Usability Requirements Elicitation Method) y ha propuesto un método para tratarlo dentro de entornos MDD.

El desarrollo de este trabajo de investigación se ha llevado a cabo siguiendo la metodología Design Science. Esta metodología considera dos ciclos: el primer ciclo es un ciclo de ingeniería en el que se diseña un método para incluir requisitos de usabilidad durante el proceso de elicitación de requisitos. El segundo ciclo corresponde a la validación del método propuesto mediante una evaluación empírica dentro de un contexto académico.

La propuesta de captura de requisitos de usabilidad mediante UREM consiste en la definición de una estructura de un árbol donde las guías de usabilidad y las guías de diseño de IU están almacenadas. El árbol se define como un grafo conectado sin ciclos y una raíz; compuesto de 4 elementos: pregunta, respuesta, grupo de preguntas y diseño.

Las preguntas y las alternativas de diseño (respuestas) son extraídas de las guías de usabilidad y de diseño, y marcan el camino por el cual el

analista navega hasta llegar a los nodos hoja que son los diseños de la interfaz de usuario que se han alcanzado durante el proceso de captura de requisitos de usabilidad. Son los usuarios finales quienes eligen la alternativa más adecuada dependiendo de sus requisitos y/o siguiendo las recomendaciones ya preestablecidas en la estructura del árbol. La construcción del árbol la lleva a cabo un experto en usabilidad y puede ser utilizado en reiteradas ocasiones, generando así diversas alternativas de diseño de interfaz de usuario.

La tesis presenta el trabajo relacionado en tres áreas: elicitación de requisitos de usabilidad, uso de guías de usabilidad e ingeniería empírica de software.

Resum

La investigació desenvolupada en aquesta tesi representa un marc nou per a capturar requisits d'usabilitat durant el desenvolupament d'un sistema programari. Aquests requisits, estan representats com a alternatives de dissenys d'Interfícies d'Usuari (IU). L'objectiu és desenvolupar un procés de captura de requisits d'usabilitat basat en entrevistes estructurades amb el suport d'una eina que ajude a resoldre problemes com: (1) l'omissió de la usabilitat des de les primeres etapes de desenvolupament, en general, les característiques d'usabilitat només es tenen en compte en dissenyar les interfícies en les últimes etapes de desenvolupament; (2) resulta tediós la captura de requisits per a analistes que no són experts en usabilitat; (3) els mètodes i eines que s'utilitzen per a desenvolupar programari no admeten l'elicitació de requisits d'usabilitat. A partir d'aquests problemes trobats en la literatura es defineixen les preguntes d'investigació: És possible capturar requisits d'usabilitat en etapes inicials de desenvolupament al mateix temps que els requisits funcionals? Per a respondre a aquesta pregunta, la tesi ha definit un mètode d'elicitació de requisits d'usabilitat anomenat UREM (per les seues sigles en anglés, Usability Requirements Elicitation Method) i ha proposat un mètode per a tractar-lo dins d'entorns MDD.

El desenvolupament d'aquest treball de recerca s'ha dut a terme seguint la metodologia Design Science. Aquesta metodologia considera dos cicles: el primer cicle és un cicle d'enginyeria en el qual es dissenya un mètode per a incloure requisits d'usabilitat durant el procés d' elicitació de requisits. El segon cicle correspon a la validació del mètode proposat mitjançant una avaluació empírica dins d'un context acadèmic.

La proposta de captura de requisits d'usabilitat mitjançant UREM consisteix en la definició d'una estructura d'un arbre on les guies d'usabilitat i les guies de disseny d'IU estan emmagatzemades. L'arbre es defineix com un graf connectat sense cicles i una arrel; compost de 4 elements: pregunta, resposta, grup de preguntes i disseny.

Les preguntes i les alternatives de disseny (respostes) són extretes de les guies d'usabilitat i de disseny, i marquen el camí pel qual l'analista navega fins a arribar als nodes fulla que són els dissenys de la interfície d'usuari que s'han aconseguit durant el procés de captura de requisits d'usabilitat. Són els usuaris finals els qui trien l'alternativa més adequada depenent dels seus requisits i/o seguint les recomanacions ja preestablides en l'estructura de l'arbre. La construcció de l'arbre la duu a terme un expert en usabilitat i pot ser utilitzat en reiterades ocasions, generant així diverses alternatives de disseny d'interfície d'usuari.

La tesi presenta el treball relacionat en tres àrees: elicitació de requisits d'usabilitat, ús de guies d'usabilitat i enginyeria empírica de programari.

Abstract

The research developed in this thesis represents a novel framework for capturing usability requirements during the development of a software system. These requirements are represented as alternative User Interface (UI) designs. The objective is to develop a usability requirements capture process based on structured interviews with the support of a tool that helps solve problems such as: (1) the omission of usability from the early stages of development, in general, the characteristics of Usability is only taken into account when designing interfaces in the later stages of development; (2) it is tedious to capture requirements for analysts who are not usability experts; (3) the methods and tools used to develop software do not support the elicitation of usability requirements. Based on these problems found in the literature, the research questions are defined: Is it possible to capture usability requirements in initial stages of development at the same time as functional requirements? To answer this question, the thesis has defined a usability requirements elicitation method called UREM (Usability Requirements Elicitation Method) and has proposed a method to treat it within MDD environments.

The development of this research work has been carried out following the Design Science methodology. This methodology considers two cycles: the first cycle is an engineering cycle in which a method is designed to include usability requirements during the requirements elicitation process. The second cycle corresponds to the validation of the proposed method through an empirical evaluation within an academic context.

The proposal to capture usability requirements through UREM consists of the definition of a tree structure where the usability guides and UI design guides are stored. The tree is defined as a connected graph without cycles and a root; composed of 4 elements: question, answer, group of questions and design.

The questions and design alternatives (answers) are extracted from the usability and design guides, and mark the path along which the analyst navigates until reaching the leaf nodes, which are the user interface designs that have been achieved. during the usability requirements

capture process. It is the end users who choose the most appropriate alternative depending on their requirements and/or following the recommendations already pre-established in the tree structure. The construction of the tree is carried out by a usability expert and can be used repeatedly, thus generating various user interface design alternatives.

The thesis presents related work in three areas: usability requirements elicitation, use of usability guides, and empirical software engineering.

Índice General

Estructura de la Tesis

Siguiendo la normativa de la Universidad Politécnica de Valencia para la tesis por compendio de artículos, la estructura de este trabajo se ajusta a las siguientes cuatro partes:

Parte I (**Introducción**). La primera parte de la tesis presenta la motivación de la investigación, la descripción del problema, los objetivos del trabajo, la relación de artículos científicos publicados para el cumplimiento de los objetivos de la tesis y la metodología seguida para desarrollar la investigación.

Parte II (**Publicaciones**). La segunda parte de la tesis, compuesta por cuatro capítulos (capítulos 1, 2, 3 y 4) contiene el compendio de artículos científicos que resultan de la investigación realizada para la tesis. Las contribuciones están ordenadas cronológicamente, y su formato ha sido adaptado al formato de esta tesis.

Parte III (**Discusiones**). En la tercera parte de la tesis se realiza una discusión general de los resultados relacionando los aportes de la tesis con el contexto de la investigación.

Parte IV (**Conclusiones**). La cuarta y última parte de la tesis presenta las conclusiones sobre el trabajo realizado y las futuras líneas de investigación.

PARTE I

INTRODUCCION I

Los temas que se cubren en esta parte son:

Esta parte presenta la motivación para realizar la tesis, incluyendo el análisis del problema a resolver, los objetivos a alcanzar, y las preguntas de investigación que conducirán a la construcción del marco de desarrollo de requisitos de usabilidad. Además, se describe la metodología seguida con la que se llevó a cabo la investigación, así como las contribuciones y el contexto de la tesis.

1.1 Motivación y Planteamiento del Problema

La interacción persona ordenador ha desarrollado guías y recomendaciones para mejorar la usabilidad en los sistemas de información que son usualmente aplicados en las etapas finales del proceso de desarrollo software. Por otro lado, la comunidad de la ingeniería del software ha desarrollado métodos conocidos para capturar requisitos funcionales en etapas tempranas, siendo los requisitos como la usabilidad postergada a etapas finales conjuntamente con otros requisitos no funcionales. La captura de requisitos de usabilidad permite a los ingenieros de software, diseñadores, y analistas crear software que no solo cumpla con los requisitos funcionales [1].

Además, no existen métodos que capturen requisitos de usabilidad durante el desarrollo del software en ambas comunidades y la mayoría de trabajos para optimizar la usabilidad se centran en el uso real de la aplicación final [2]. Un claro ejemplo de este problema se manifiesta en la aplicación del paradigma de desarrollo dirigido por modelos en donde los métodos y herramientas no soportan la captura de requisitos de usabilidad.

El desarrollo de interfaces de usuario, que va desde los primeros requisitos hasta la implementación del software, se ha convertido en un proceso costoso y lento en el ciclo de vida del desarrollo de software (SDLC) [3]. Este proceso sería más efectivo si se incluyeran los requisitos de usabilidad para que el software cumpla con los requisitos de los usuarios y además brinde una interacción con el software acorde con el tipo de tarea a realizar. Existen propuestas para utilizar guías de diseño que mejoren la usabilidad pero cómo relacionar estas guías con la elicitación de requisitos es un ámbito aún no explorado [4].

Las áreas de la Interacción Persona Ordenador (IPO) e Ingeniería del Software (IS) tienen como objetivo común desarrollar sistemas usables. En ambas comunidades, la usabilidad suele considerarse en las últimas etapas del proceso de desarrollo de software, cuando las interfaces ya han sido diseñadas. El incluir características de usabilidad en estas últimas etapas podría afectar a la arquitectura del sistema. Para minimizar este problema, la usabilidad debe incluirse en la etapa de captura de requisitos [5], [6]. La comunidad de la IS tiene una amplia experiencia en la obtención temprana de requisitos y existen métodos sólidos. Sin embargo, estos métodos solo se centran en los requisitos funcionales (RF), y los requisitos no funcionales (NFR) como la usabilidad han sido olvidados en esta etapa temprana. Según muchos autores, cumplir con los requisitos funcionales no es suficiente para crear y asumir que un producto es de calidad [7]. La usabilidad es un factor clave para obtener niveles de aceptación.

Model-Driven Development (MDD) ha sido bastante popular en la comunidad académica [8] en los últimos años, y se han introducido varias propuestas diferentes para desarrollar sistemas de software. MDD es un paradigma de desarrollo de software que se basa en modelos y transformaciones de modelos para obtener un producto final mediante la generación automática de código considerando algunas reglas de transformación.

En un campo donde la tecnología cambia rápidamente, una metodología basada en modelos es una opción válida por algunas razones:

• El dominio del conocimiento está representado en modelos, siendo éstos independientes de la tecnología [9],

• La solución para el desarrollo de un sistema software no se ve afectada por la evolución de la plataforma hardware.

• Cuando se considera una nueva tecnología como plataforma de destino para desarrollar software, no es necesario volver a describir todo el sistema sino generar un nuevo modelo específico de plataforma (PSM) que incluya los cambios en la plataforma de destino.

• Las tareas relacionadas con el ciclo de vida del desarrollo (mantenimiento, nuevos requisitos, proceso de actualización) son menos complicadas de realizar [10].

Esta tesis presenta un método para el proceso de elicitación de requisitos de usabilidad (UREM, por sus siglas en inglés, Usability Requirements Elicitation Method) representados en diseños de IU construidos siguiendo guías de usabilidad, de diseño, estándares e ISOs dentro del entorno de MDD con el apoyo de una herramienta de soporte. El método tiene como objetivo representar los requisitos de usabilidad mediante alternativas de diseños de IU que serán seleccionados por el usuario final durante la captura de requisitos. Este método propone representar los diseños de las interfaces en modelos conceptuales que después puedan ser la entrada a un proceso de desarrollo MDD.

En resumen, el enunciado del problema en esta tesis es:

No existe un método para capturar los requisitos de usabilidad que tenga en cuenta guías de diseño y recomendaciones de usabilidad que ayuden a analistas poco expertos en el desarrollo de sistemas usables bajo el enfoque MDD.

Nuestro trabajo tiene como objetivo definir un método de captura de requisitos de usabilidad (UREM) para analistas que no son expertos en ingeniería de usabilidad y deseen incorporar la especificación de requisitos de usabilidad en un entorno de MDD.

1.2 Objetivos y Preguntas de Investigación

El objetivo principal de la Tesis es definir UREM: un método estructurado basado en normas y guías de usabilidad que incorporan requisitos de usabilidad durante la captura de requisitos mediante entrevistas entre el analista y el usuario final, obteniendo diseños de IU como resultado de las entrevistas.

Para lograr el objetivo principal, es necesario responder las siguientes preguntas de investigación (RQ), que debido a su amplitud son subdividas en sub preguntas de investigación (SQ):

- RQ1: ¿Es posible capturar requisitos de usabilidad en etapas iniciales de desarrollo software?

    - SQ1.1: ¿Que métodos, guías de usabilidad, estándares y normas se requieren en el proceso de captura de requisitos de usabilidad que apoyen la labor del analista?
    - SQ1.2: ¿Es posible desarrollar una estructura de árbol que facilite el proceso de captura de requisitos en un entorno MDD?
    - SQ1.3: ¿Es posible representar alternativas de diseño de IU en una estructura de árbol en base a las guías de usabilidad y diseño para la captura de requisitos de usabilidad?
- RQ2: ¿Qué impacto produce UREM en la captura de requisitos de usabilidad?
    - SQ2.1 ¿Cuál es el impacto del uso de las guías de usabilidad en el diseño de IU?
    - SQ2.2 ¿Cuál es el impacto de la aplicación del UREM en un contexto académico?
    - SQ2.3 ¿Cuál es el impacto de las recomendaciones de usabilidad propuestas por UREM?

Para contestar estas preguntas, se plantean los siguientes objetivos específicos:

Objetivo 1 (RQ1). Para contestar la RQ1, se identificarán las limitaciones y problemas existentes en el desarrollo del software por la ausencia de mecanismos que garanticen una adecuada captura de requisitos de usabilidad. Para contestar la SQ1.1, se analizarán métodos, estándares, normas y guías de usabilidad existentes en la literatura que deben ser incluidas en el desarrollo del software y durante el diseño de IU. Para contestar la SQ1.2, se definirá un mecanismo de captura de requisitos de usabilidad que consiste en desarrollar una estructura de árbol en base a preguntas, grupo de preguntas y respuestas, que resulten en diseños de IU usables. Para contestar la SQ1.3 se implementarán las guías de usabilidad y diseños dentro de la estructura del árbol que conduzcan a la generación de diseños de IU usables.

Objetivo 2 (RQ2). Para contestar la RQ2, se realizará el experimento empírico. El experimento, está orientado a responder las SQ2.1, SQ2.2 y SQ2.3, es un experimento con 2 réplicas para comparar UREM con un método de elicitación de requisitos de usabilidad no estructurado (y sin guías de usabilidad). Los diseños de IU son el resultado de la captura de requisitos de usabilidad realizado y se plasman en los diseños de IU obtenidos al final de la entrevista.

1.3 Compendio de Artículos

Como resultado de la investigación se han elaborado y publicado cuatro artículos de investigación que abarcan las preguntas de investigación y responden más explícitamente a las sub preguntas de investigación definidas.

1.3.1 Mapping study about usability requirements elicitation

Ormeño, Yeshica, Ignacio Panach y Óscar Pastor. En International Conference Advanced Information Systems Engineering (CAiSE 2013). Springer 2013, págs. 672-687, DOI: https://doi.org/10.1007/978-3-642-38709-8_43.

Este artículo publicado en la conferencia CORE A CAiSE aborda la sub pregunta de investigación SQ1.1: ¿Que métodos, guías de usabilidad, estándares y normas se requieren en el proceso de captura de requisitos de usabilidad que apoyen la labor del analista?

En el primer artículo se ha desarrollado un estudio sistemático siguiendo la metodología de Kitchenham, cuyo objetivo es identificar las propuestas existentes para la elicitación de requisitos de usabilidad desde las primeras etapas de desarrollo software, la misma que ha sido subdividida en 6 sub preguntas referentes a: 1) Métodos para elicitar los requisitos de usabilidad. Los métodos existentes inician el proceso de elicitación de los NFRs mediante técnicas tradicionales (entrevistas, cuestionarios, etc.) teniendo que ser personalizables en caso de aplicarse a otros contextos diferentes, es decir deben ser adaptados. Además, solo proporcionan soporte básico a la gestión de requisitos por medio de extensiones para la captura de requisitos. 2) Métodos para elicitar requisitos de interacción. Se caracterizan porque realizan un análisis exhaustivo de los requisitos para encontrar y aliviar los problemas de interacción donde los modelos están basados en el análisis sistemático de un conjunto de propiedades de interfaces estándar, y/o patrones estructurales, buscando potenciar la usabilidad y experiencia de usuario. 3) Guías de usabilidad utilizadas para elicitar los requisitos de usabilidad. Las guías encontradas ayudan a superar en parte el obstáculo de la integración de la usabilidad y su significado por los stakeholders. No obstante, para su aplicación se requiere la interpretación de un experto en usabilidad. 4) Herramientas de apoyo a la elicitación de requisitos. Las herramientas son de apoyo y presentan funcionalidad limitada cuando se orientan a la elicitación de requisitos. En general, están orientadas a la identificación de requisitos para que las interfaces de usuario sean más comprensibles por los usuarios. Se utilizan más en el diseño de sistemas interactivos, pero su uso exige cierto grado de esfuerzo en la comprensión y aplicación por parte del analista. 5) Tipo de notación para la elicitación de los requisitos. Las notaciones son utilizadas por los métodos en sus diferentes fases de desarrollo. Algunos tipos de representación son patrones, escenarios y plantillas. En algunos métodos se han utilizado más de una notación en combinación con más de un artefacto, siendo de gran uso para el analista, aunque no son tan comprendidos por el usuario final. y 6) Entorno de validación empírica. Los casos de estudio, experimentos o pruebas de concepto que se plantean dentro del plano académico e industrial no muestran métricas explícitas que determinen el nivel de usabilidad logrado por el sistema. Además, los métodos están desarrollados para ciertas características de usabilidad consideradas de mayor impacto sobre la funcionalidad. Las listas de verificación, sesiones y gestión de escenarios son los artefactos generalmente utilizados para evaluar la usabilidad. Generalmente, la usabilidad se evalúa mediante encuestas en términos de efectividad, eficiencia y satisfacción.[1].

Analizando los resultados del estudio sistemático, podemos concluir que existe una clara línea de investigación en el campo de los requisitos de usabilidad en entornos MDD. Por lo general, los métodos MDD históricamente se han centrado en modelar el comportamiento y la persistencia, pero relegando la interacción y particularmente la usabilidad a una implementación manual. Esta implementación manual contradice claramente el paradigma MDD, que aboga por que el analista trabaje con modelos conceptuales holísticos, en los que se puedan representar todas las características del sistema (incluidas las características de usabilidad).

1.3.2 Towards a proposal to capture usability requirements through guidelines

Ormeño, Yeshica, Ignacio Panach, Nelly Condori y Óscar Pastor. En International Conference Research Challenges in Information Science (RCIS 2013). IEEE 2013, Págs.1-12, DOI: 10.1109/RCIS.2013.6577677

Este artículo publicado en el congreso CORE B RCIS, aborda la sub preguntas de investigación SQ1.2: ¿Es posible desarrollar una estructura de árbol que facilite el proceso de captura de requisitos en un entorno MDD?.

En este segundo artículo se define el proceso para capturar los requisitos de usabilidad consistente en construir una estructura de árbol utilizando las guías de diseño de interfaz usuario y las guías de usabilidad que ayudan al analista a capturar los requisitos de usabilidad. El enfoque se basa en un formato de pregunta-respuesta de tal manera que los requisitos se capturan con una entrevista con el usuario final. El resultado de la entrevista es un conjunto de diseños que el sistema debe satisfacer. Si especificamos estos diseños formalmente, podemos transformarlos en primitivas conceptuales de un método MDD existente.

Los componentes del modelo para el árbol son: 1) Las preguntas, que son formuladas en base a las diversas alternativas de diseño que existen para la especificación de los componentes de una IU extraídas de las guías de diseño y estándares de usabilidad existentes. Se pregunta al usuario que alternativa es de su preferencia. 2) Las respuestas, que son establecidas como opciones exclusivas para ser presentadas al analista, quien elige cuál se adapta mejor a los requisitos. La decisión del analista no solo se basa en los criterios del usuario final, sino que toma en consideración las respuestas que están definidas en el árbol en base a las guías de usabilidad según el tipo de usuario, tarea y contexto. Estas son las respuestas que son recomendadas al usuario durante su elección. 3) Los grupos de preguntas, que están formadas por un conjunto de preguntas, agrupadas por una característica de diseño de IU. Las preguntas no son mutuamente excluyentes, es decir, se deben consultar todas ellas al usuario independientemente de las respuestas que se elijan. 4) Los diseños, que son las hojas del árbol alcanzadas a través de las alternativas que el analista ha ido eligiendo como resultado de las selecciones realizadas por el usuario final.

La estructura de árbol y la transformación entre los diseños y el método MDD se definen una sola vez y se pueden reutilizar indefinidamente para desarrollar cualquier sistema.

1.3.3 A proposal to elicit usability requirements within a model-driven development environment.

Ormeño, Yeshica, Ignacio Panach, Nelly Condori y Óscar Pastor. En International Journal of Information System Modeling and Design (2014) 5(4), Págs.1-21, DOI: http://dx.doi.org/10.4018/ijismd.2014100101

Este artículo publicado en una revista internacional aborda la sub preguntas de investigación SQ1.3: ¿Es posible representar alternativas de diseño de IU en una estructura de árbol en base a las guías de usabilidad y diseño para la captura de requisitos de usabilidad?

En este tercer artículo se presenta el proceso para elicitar requisitos de usabilidad basado en alternativas de diseño propuestas y lineamientos de usabilidad existentes. El enfoque se basa en la construcción de una estructura de árbol que representa todas las alternativas de diseño. Se explica en detalle cómo construir la estructura de árbol y cómo usarla. El usuario final participa en el proceso, eligiendo la alternativa de diseño que mejor se ajuste a sus requerimientos. La navegación comienza desde la raíz del árbol y continua mientras el analista hace las preguntas a los usuarios. La posible navegación entre dos nodos de la estructura de árbol puede ser: i) De un grupo de preguntas a una pregunta, o a otro grupo de preguntas; ii) De una pregunta a una respuesta iii) De una respuesta a una pregunta o a un grupo de preguntas o a un diseño.

El enfoque ha sido validado con 4 sujetos a través de una demostración de laboratorio. En el ejemplo, se han utilizado dos guías de usabilidad: ISO 9126-3 y los criterios ergonómicos. Nuestro enfoque acepta tantas guías como el analista quiera considerar. Una contradicción entre dos guías no significa un problema, ya que el usuario final decide la alternativa de diseño más adecuada. Sin embargo, es importante mencionar que demasiadas recomendaciones para los posibles diseños pueden confundir a los usuarios finales.

Como resultado del proceso de elicitación obtenemos algunos modelos conceptuales incompletos. En los próximos pasos de desarrollo, el analista debe mejorar estos modelos con primitivas que representen la funcionalidad y la apariencia visual del sistema para obtener un sistema completamente funcional.

1.3.4 An Empirical of a Usability Requirements Elicitation Method based on Interviews

Ormeño, Yeshica, Ignacio Panach y Óscar Pastor. En Information and Software Technology (2023), Págs. 107324, DOI: https://doi.org/10.1016/j.infsof.2023.107324

Este artículo publicado en la revista JCR IST (Q2 en JCR) aborda las sub preguntas de investigación SQ2.1 ¿Cuál es el impacto del uso de las guías de usabilidad en el diseño de IU?, SQ2.2 ¿Cuál es el impacto de la aplicación del UREM en un contexto académico? y SQ2.3 ¿Cuál es el impacto de las recomendaciones de usabilidad propuestas por UREM?

En el cuarto artículo se ha realizado un experimento que compara entrevistas estructuradas con entrevistas no estructuradas para obtener requisitos de usabilidad. Las entrevistas estructuradas se operacionalizan con UREM, que es un método basado en un árbol de decisiones en el que el analista guía la entrevista navegando por la estructura del árbol. Cada rama del árbol incluye una pregunta para el usuario final con posibles respuestas. Además, se recomienda la respuesta que cumpla más con las guías de usabilidad existentes. Con el método de entrevista no estructurada, el analista debe obtener requisitos de usabilidad sin ninguna guía. En el experimento, el tratamiento de control se denomina entrevista no estructurada. La evaluación se realiza para analizar cuatro variables de respuesta: 1) Efectividad en la elicitación de requisitos de usabilidad;2) Efectividad en la aplicación de las guías de usabilidad; 3) Eficiencia; y 4) la satisfacción tanto del analista como la del usuario final. Como resultados significativos, UREM es más efectivo en la obtención de requisitos de usabilidad y también más efectivo en el diseño de interfaces que cumplen con las guías de usabilidad.

Se han aprendido algunas lecciones durante la realización del experimento: 1) El esfuerzo para construir el árbol con UREM es alto. Esto es algo que no se analizó en el experimento, pero el esfuerzo requerido no es despreciable en base a la experiencia vivida por los experimentadores. Cabe destacar que este esfuerzo se amortiza debido a que la misma estructura de árbol es útil para cualquier desarrollo futuro; 2) Las recomendaciones realizadas durante la navegación por la estructura del árbol pueden ser diferentes según las guías de usabilidad utilizadas para construir el árbol. Si bien la mayoría de las guías de usabilidad coinciden en las características que optimizan la usabilidad, existen algunas guías que pueden presentar algunas contradicciones. Al final, el experto en usabilidad que construye la estructura de árbol es quien elige las guías de usabilidad más adecuadas para las recomendaciones; 3) La mayoría de los usuarios finales aceptaron las recomendaciones de usabilidad. Este valor podría haber sido diferente si los sujetos hubieran tenido más experiencia en las características de usabilidad.

1.4 Metodología de la Investigación

Para el desarrollo de la tesis se ha seleccionado la metodología “Design Science” (DS) [11] por su enfoque en la investigación de proyectos de Sistemas de Información e Ingeniería de Software a través de la experimentación, observación del estudio y análisis de resultados. Todo ello hace de esta metodología una guía adecuada para llevar a cabo la investigación.

DS se basa en el diseño e investigación de artefactos en un contexto. Los artefactos que estudiamos están diseñados para interactuar con un contexto problemático a fin de mejorar en ese contexto. Esta tesis aplica la metodología DS para investigar cómo se pueden capturar requisitos de usabilidad a partir de la gestión de un modelo basado en guías y estándares de usabilidad que promuevan el diseño de interfaces de usuario usables, y que satisfagan los requisitos del usuario.

El objeto de estudio de cualquier proyecto basado en DS es “estudiar un artefacto interactuando en su contexto del problema”, a lo cual la metodología lo denomina tratamiento. Cuando se menciona “artefacto” se refiere a un elemento de software (por ejemplo, método, aplicación de software, etc.) diseñado por los investigadores del proyecto DS y se usa por personas como solución a un problema.

El objeto de estudio de esta tesis es: proponer UREM (nuestro artefacto) para resolver el problema de capturar requisitos de usabilidad mediante entrevistas estructuradas (preguntas y respuestas) que se realicen en el proceso de diseño de IU. La siguiente Fig.1 muestra la relación existente.

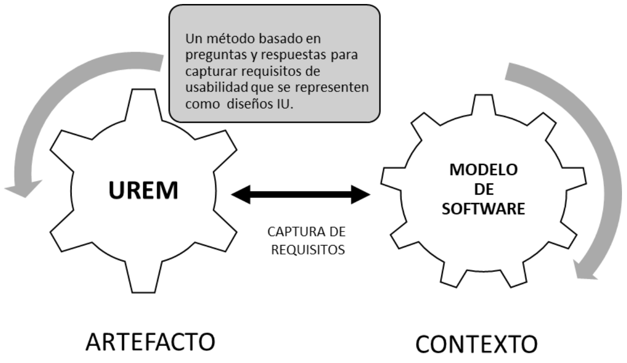

Un método basado en preguntas y respuestas para capturar requisitos de usabilidad que se representen como diseños IU.
UREM
CAPTURA DE REQUISITOS
MODELO DE SOFTWARE
ARTEFACTO
CONTEXTO

Figura 1. Artefacto que captura requisitos de usabilidad interactuando con el contexto para resolver un problema de ese contexto.

Como resultado de esta investigación se pretende que la aplicación de UREM contribuya a la captura de requisitos de usabilidad en etapas tempranas del desarrollo software facilitando la generación de diseños de IU usables. El tratamiento, el artefacto y las investigaciones asociadas a la creación de este método brindan un avance en la investigación científica.

1.4.1 Marco Metodológico Aplicado a la tesis

Para alcanzar los objetivos y responder a las preguntas de investigación, la metodología provee un marco de trabajo que consiste en dos contextos interactuando con el proyecto DS. Se tiene dos contextos que son: el contexto social y el contexto de conocimiento, como se muestra en la siguiente figura.

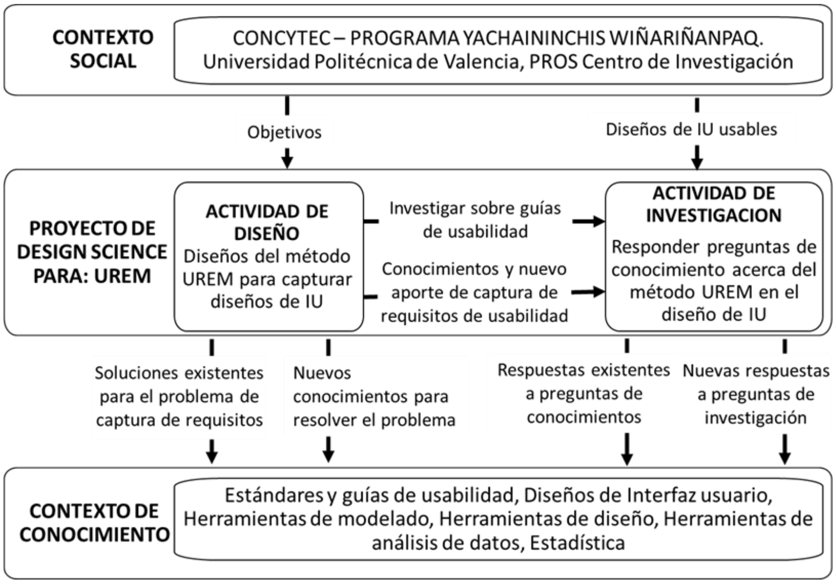

CONTEXTO SOCIAL
CONCYTEC – PROGRAMA YACHAININCHIS WIÑARIÑANPAQ. Universidad Politécnica de Valencia, PROS Centro de Investigación
Objetivos
Diseños de IU usables
PROYECTO DE DESIGN SCIENCE PARA: UREM
ACTIVIDAD DE DISEÑO Diseños del método UREM para capturar diseños de IU
Investigar sobre guías de usabilidad
Conocimientos y nuevo aporte de captura de requisitos de usabilidad
ACTIVIDAD DE INVESTIGACION Responder preguntas de conocimiento acerca del método UREM en el diseño de IU
Soluciones existentes para el problema de captura de requisitos
Nuevos conocimientos para resolver el problema
Respuestas existentes a preguntas de conocimientos
Nuevas respuestas a preguntas de investigación
CONTEXTO DE CONOCIMIENTO
Estándares y guías de usabilidad, Diseños de Interfaz usuario, Herramientas de modelado, Herramientas de diseño, Herramientas de análisis de datos, Estadística

Figura 2. Marco de trabajo de la metodología DS aplicado a la tesis

El contexto social representa a las partes interesadas del proyecto incluyendo a las personas o instituciones que financian el proyecto y/o definen los objetivos o requisitos para UREM. Las partes interesadas se dividen en 2 grupos. El primero lo conforman las partes interesadas que patrocinan el proyecto de investigación:

- Universidad Nacional de San Antonio Abad del Cusco – CONCYTEC PROCIENCIA .
- Departamento de Sistemas y Computación de la Universidad Politécnica de Valencia.
- PROS Centro de Investigación.

El segundo grupo lo conforman las partes interesadas que son beneficiarios directo del UREM.

- Universidades e investigadores en el área de desarrollo de software dirigido por modelos.
- Analistas de sistemas y desarrolladores de software

El contexto de conocimiento representa la literatura científica existente que se ha utilizado para poder llevar a cabo la investigación. En esta tesis, el contexto de conocimiento incluye las fuentes primarias de conocimiento como son la literatura científica, profesional, técnica y comunicaciones orales en las disciplinas HCI, desarrollo de software dirigido por modelos, usabilidad, ingeniería de requisitos, estudios empíricos.

1.4.2 Ciclo de Diseño y Ciclo Empírico

La metodología DS para realizar las actividades de diseño e investigación en un proyecto, provee de 2 ciclos iterativos y anidados: i) Ciclo de Diseño y ii) Ciclo Empírico. Cada ciclo está compuesto de tareas y cada tarea involucra resolver problemas de diseño y preguntas de conocimiento.

i) Ciclo de diseño

El ciclo de diseño es un proceso orientado al diseño del artefacto de la investigación y puede ser visto como un sub-ciclo de un tipo de ingeniería enfocado a la resolución de problemas. El ciclo de ingeniería está compuesto de 4 tareas de diseño (TD).

- TD1. Investigación del problema. Identificar las causas del problema, para poder ser mejorado.
- TD2. Diseño del tratamiento. Diseñar artefactos para tratar el problema, se especifican los requisitos, se estudian tratamientos existentes, para ver si se adapta el tratamiento o si se diseña un nuevo tratamiento.
- TD3. Validación del tratamiento. Verificar que el diseño del tratamiento abarca el problema.
- TD4. Implementación del tratamiento. Tratar el problema con el artefacto diseñado.

De estas 4 tareas del ciclo de ingeniería, el ciclo de diseño abarca las tareas, como se muestra en la Figura 3.

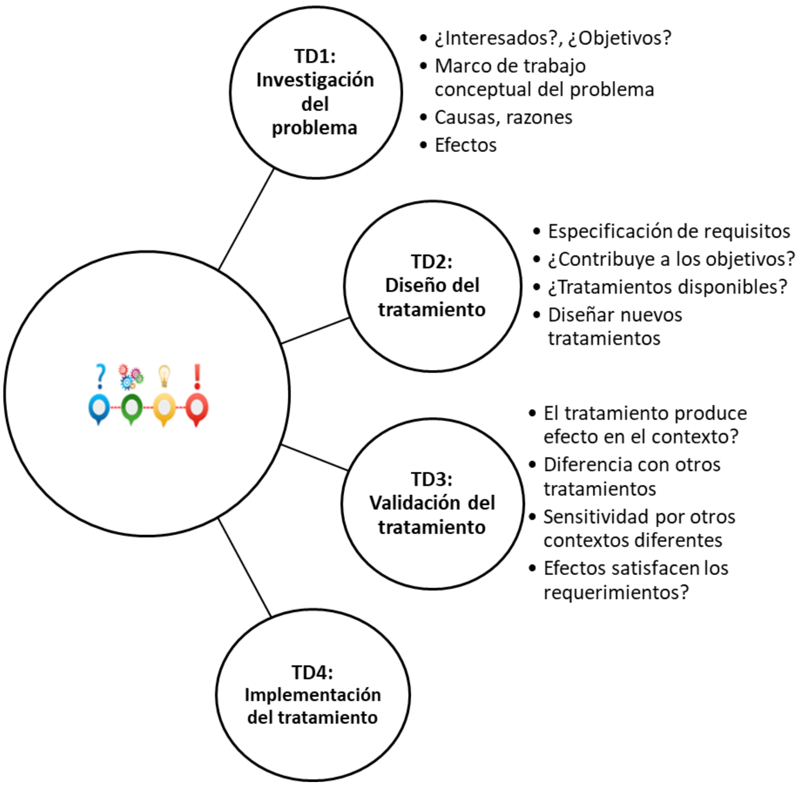

TD1: Investigación del problema
• ¿Interesados?, ¿Objetivos?
• Marco de trabajo conceptual del problema
• Causas, razones
• Efectos
TD2: Diseño del tratamiento
• Especificación de requisitos
• ¿Contribuye a los objetivos?
• ¿Tratamientos disponibles?
• Diseñar nuevos tratamientos
TD3: Validación del tratamiento
• El tratamiento produce efecto en el contexto?
• Diferencia con otros tratamientos
• Sensitividad por otros contextos diferentes
• Efectos satisfacen los requerimientos?
TD4: Implementación del tratamiento

Figura 3. Ciclo de diseño de la metodología de DS. Adaptado de [19]

En el desarrollo de esta Tesis, aplicamos un Ciclo del Diseño con las Tareas (TD) indicando en qué parte, capítulo o sección de la tesis se encuentran:

- TD1) Problema de Investigación: Definido por el investigador y la necesidad de investigar un método de captura de requisitos de usabilidad a partir de entrevistas (Parte II, Sección 1).
- TD2) Estado del Arte: Investigar propuestas existentes relacionadas con métodos de captura de RF y NFR, requisitos de interacción, notación, guías, validaciones empíricas (Parte II, Sección 1).

- TD3) Definir el método estructurado: A partir de la estructura de un árbol donde se definen preguntas y respuestas para generar alternativas de diseño de IUs. Estas preguntas y respuestas fueron extraídas de la revisión de guías y estándares de usabilidad existentes en la literatura (Parte II, Sección 2).
- TD4) Definir alternativas de diseños de IU: Al definir las preguntas de la estructura en árbol, cuando las preguntas tienen más de una respuesta (alternativa de diseño), se utilizan guías de usabilidad para recomendar la alterativa apropiada en base a los estándares y guías de usabilidad. Se asignan preguntas y respuestas a cada alternativa que conducen a la especificación de un diseño de IU (Parte II, Sección 3).
- TD5) Definir recomendaciones de usabilidad: Cuando las preguntas cuentan con más de una alternativa que conlleva a los diseños de IUs, se proporciona las alternativas que contienen recomendaciones de usabilidad para saber qué alternativa es más adecuada (Parte II, Sección 4).

ii) Ciclo Empírico

El ciclo empírico es un proceso orientado a contestar preguntas de conocimiento científico de manera racional, donde el investigador diseña la configuración de la investigación (o estudio empírico, como por ejemplo un experimento) y analiza los datos producidos de esta experimentación. El ciclo empírico se muestra en la Figura 4, y se compone de 5 tareas que se identifican con (TE).

- TE1. Análisis del problema de investigación, que consiste en definir las preguntas de investigación sobre las cuales vamos a realizar el estudio, y reclutar los sujetos del experimento de quienes obtenemos los datos.
- TE2. Diseño de la investigación, que consiste en diseñar el estudio empírico definiendo las variables y las métricas (como medirlas), definir los problemas experimentales (los problemas que los sujetos tienen que resolver), definir los tratamientos de la investigación, y definir los métodos estadísticos que serán utilizados para obtener resultados.

- TE3. Validación de la investigación, que consiste en validar las amenazas que puedan afectar el estudio empírico y a los resultados. Utilizamos 4 tipos de validaciones [12]: validez de la conclusión, validez interna, validez del constructo, y validez externa. Se describe cómo se ha minimizado o cubierto las amenazas del experimento para cada tipo de validación.
- TE4. Ejecución del experimento, que consiste en ejecutar el experimento empírico según el diseño del experimento.
- TE5. Análisis de los datos, que consiste en analizar los datos obtenidos en el experimento de acuerdo a los métodos estadísticos definidos en el diseño del experimento.

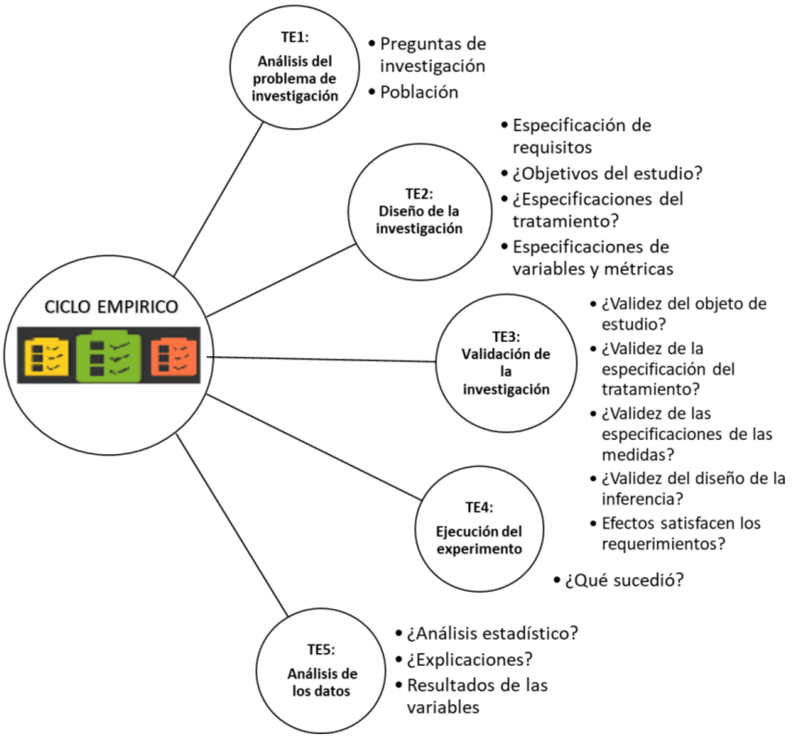

TE1: Análisis del problema de investigación
• Preguntas de investigación
• Población
TE2: Diseño de la investigación
• Especificación de requisitos
• ¿Objetivos del estudio?
• ¿Especificaciones del tratamiento?
• Especificaciones de variables y métricas
CICLO EMPIRICO
TE3: Validación de la investigación
• ¿Validez del objeto de estudio?
• ¿Validez de la especificación del tratamiento?
• ¿Validez de las especificaciones de las medidas?
• ¿Validez del diseño de la inferencia?
• Efectos satisfacen los requerimientos?
TE4: Ejecución del experimento
• ¿Qué sucedió?
TE5: Análisis de los datos
• ¿Análisis estadístico?
• ¿Explicaciones?
• Resultados de las variables

Figura 4. Ciclo Empírico de la metodología DS. Adaptado de [13]

En esta Tesis se ha definido un ciclo empírico que se muestran en la Parte II Sección 4:

El ciclo empírico TE: La validación de UREM y sus diseños de IU incluyendo recomendaciones para optimizar la usabilidad se encuentra en la Parte II Sección 4. A continuación, se muestran las tareas (TE) relacionadas con este ciclo:

- TE1) Análisis del problema de investigación: Se definieron 5 preguntas de investigación. El experimento consiste en 2 réplicas, los sujetos son estudiantes del Grado y Master de Ingeniería Informática de la Universidad Nacional de San Antonio Abad del Cusco (Perú).
- TE2) Diseñar un experimento para validar UREM: El investigador propuso 2 problemas experimentales en contextos diferentes, donde cada problema requiere la elicitación de distintos requisitos de usabilidad. En el experimento participan sujetos con dos roles: el rol de analista que elicita los requisitos de usabilidad y diseña las IUs, y el rol del cliente que explica sus requisitos y valida el resultado. La captura de requisitos de usabilidad se realiza con un método de entrevistas no estructurado y UREM (haciendo uso del árbol implementado para este proceso) para comparar el grupo de control con el tratamiento respectivamente. Después de realizar la entrevista con uno u otro método, el analista debe dibujar los diseños de IUs que satisfagan los requisitos de usabilidad del cliente. Las variables y las métricas utilizadas en la experimentación son: Efectividad aplicada en dos contextos: Efectividad en la captura de requisitos (se mide como el porcentaje de requisitos de usabilidad satisfechos por el analista usando el método no estructurado y UREM), y efectividad en la aplicación de las guías de usabilidad (se mide como el porcentaje de requisitos de usabilidad que han sido incluidos en el diseño de la IU usando el método no estructurado y el UREM). Eficiencia (se mide como el ratio del tiempo destinado en la captura de requisitos de usabilidad sobre la efectividad lograda en la captura de requisitos por el analista con el método no estructurado y UREM). Satisfacción aplicada desde dos perspectivas: Satisfacción del analista que

diseña las IU (se mide como el nivel de satisfacción del analista durante la elicitación de requisitos usando el método no estructurado y UREM) y satisfacción del usuario final quien utilizará las IUs (se mide a través del cuestionario CSUQ (https://garyperlman.com/quest/quest.cgi), para el método no estructurado y UREM). Para la satisfacción del analista se mide en términos de Facilidad de Uso Percibida, Utilidad Percibida y la Intención de Uso a través de un cuestionario de escala de Likert de 5 puntos) para el método no estructurado y el método UREM.

- TE3) Validación de la investigación: El experimento valida las amenazas que puedan afectar el estudio empírico y a los resultados, utilizamos 4 tipo de validaciones [12]: validez de la conclusión (Poder estadístico bajo, Supuestos transgredidos de estadística, Pesca, Fiabilidad de las medidas, Fiabilidad de la implementación de los tratamientos y Heterogeneidad aleatoria de los sujetos), validez interna (Historia, Maduración, Instrumentación, Selección, Mortalidad y Rivalidad compensatoria), validez del constructo (Explicación preoperacional inadecuada de los constructos, Sesgo mono-operación, Sesgo mono-metodo y Homogeneidad del problema), y validez externa (Interacción de selección y tratamiento, Interacción de entornos y tratamiento e Interacción de historia y tratamiento). Se describe cómo se ha cubierto y minimizado las amenazas del experimento para cada tipo de validación.
- TE4) Ejecutar el experimento para validar el método: El experimento se ejecuta en 2 réplicas, el investigador elaboró dos listas de requisitos de usabilidad para cada problema y se desarrollaron sesiones de capacitación del manejo de UREM a todos los sujetos experimentales días antes del experimento. Además de una introducción de UREM con una duración de 10 minutos antes del experimento, se realiza un cuestionario demográfico para saber el nivel de conocimiento de captura de requisitos, diseño de IU y guías de usabilidad de cada uno de los sujetos.
- TE5) Analizar resultados del método: El análisis muestra el resultado del Eficiencia, Eficacia y la Satisfacción del método no estructurado y UREM.

La Figura 5 muestra los ciclos aplicados a la Tesis.

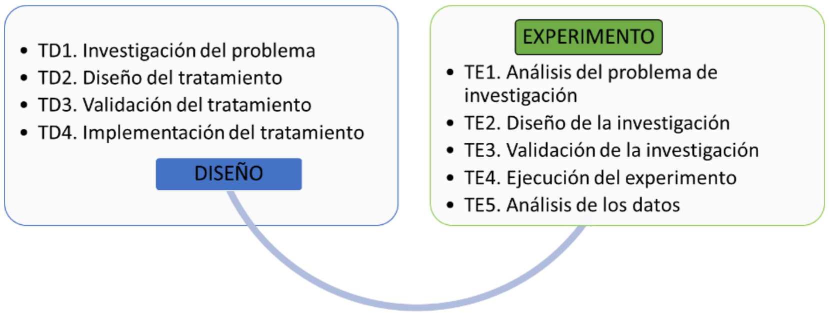

TD1. Investigación del problema
TD2. Diseño del tratamiento
TD3. Validación del tratamiento
TD4. Implementación del tratamiento
DISEÑO
EXPERIMENTO
TE1. Análisis del problema de investigación
TE2. Diseño de la investigación
TE3. Validación de la investigación
TE4. Ejecución del experimento
TE5. Análisis de los datos

Figura 5. Ciclos aplicados a la tesis

1.5 Contribuciones de la tesis

Esta tesis presenta los siguientes aportes:

Contribución 1: **Definición de un método de captura de requisitos de usabilidad** basado en un árbol de decisiones. El método captura requisitos de usabilidad a través del diseño de interfaces usuario mediante la estructura de un árbol que contiene las guías de usabilidad y diseño. El árbol lo construye un experto en usabilidad.

Contribución 2: **Una herramienta** para apoyar el método de requisitos de usabilidad descrito en la Contribución 1.

Contribución 3: La **validación** del método propuesto mediante una evaluación comparativa empírica**.**

1.6 Contexto de la tesis

Este trabajo de investigación se ha desarrollado en el contexto del Centro de Investigación PROS (Centro de Investigación en Métodos de Producción de Software), y DSIC (Departamento de Sistemas de Información y Computación) de la Universitat Politècnica de València, España.

Este trabajo ha sido financiado por la Universidad Nacional de San Antonio Abad del Cusco a través del Consejo Nacional de Ciencia y Tecnología Secretaría Nacional de Educación Superior, Ciencia Tecnología e Innovación Tecnológica-CONCYTEC de Perú, bajo el Programa Yachayninchis Wiñarinanpaq UNSAAC.

PARTE II

COMPENDIO DE PUBLICACIONES II

Los temas que se cubren en esta parte son:

2.1 Mapping Study about Usability Requirements Elicitation

The HCI community has developed guidelines and recommendations for improving the usability system, usuability applied at the last stages of the software development process. On the other hand, the SE community has developed sound methods to elicit functional requirements in the early stages, but usability has been relegated to the last stages together with other non-functional requirements. Therefore, there are no methods of usability requirements elicitation to develop software within both communities. An example of this problem arises if we focus on the Model-Driven Development paradigm, where the methods and tools that are used to develop software do not support usability requirements elicitation. In order to study the existing publications that deal with usability requirements from the first steps of the software development process, this work presents a mapping study. Our aim is to compare usability requirements methods and to identify the strong points of each one.

Versión del autor del artículo: Ormeño, Y. I., & Panach, J. I. (2013). Mapping study about usability requirements elicitation. In Advanced Information Systems Engineering: 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013. Proceedings 25 (pp. 672-687). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38709-8_43

1.1 Introduction

The goal of developing usable systems has been dealt with by the Human Computer Interaction (HCI) and Software Engineering (SE) fields. In both communities, usability is usually considered in the last stages of the software development process, when the interfaces have already been designed. Including usability characteristics at these last stages could affect the system architecture. To minimize this problem, usability should be included at the requirements elicitation stage [5], [20]. The SE community has broad experience in early requirements elicitation and there are sound methods. However, these methods are mainly focused on functional requirements and Non-Functional Requirements (NFR) have historically been forgotten at this early stage.

According to many authors, fulfilling functional requirements is not enough to create a quality product [49]. Usability is a key factor in obtaining good acceptance rates.

In this study, we aim to identify the existing methods for capturing usability requirements. To do this, we perform a Mapping Study (MS) based on the works performed by Kitchenham [29]. A MS provides an objective procedure for identifying the nature and extent of the research that is available to answer a particular question. These studies are also useful to identify gaps in current research and to suggest areas for further investigation. Of all the software development methods, we focus on the Model-Driven Development (MDD). MDD aims to develop software by means of a conceptual model, which is the input for a model compiler that generates the system code implementation. The SE community has been working with this paradigm, and, nowadays, there are sound methods and tools (e.g. OO-Method [39], WebRatio [2], OOHDM [12]). However, to the authors' knowledge, none of these methods deal with usability. In general, existing MDD methods deal with usability when the models that represent the functional requirements have been defined and the code has been generated. At this stage, if the analyst needs to improve the system usability, the code must be modified manually. Moreover, some changes require the architecture to be re-worked [5], [20]. These are the reasons why more efforts should be made to include usability in MDD methods, and this MS aims to be a step forward this direction.

Our long term target is twofold: (1) to improve current practices of usability requirements elicitation; and (2) to enhance the existing MDD methods to support usability requirements elicitation. The MS can help us to identify the advantages and disadvantages of each existing capture method, as a previous step for our target. However, the MS is not exclusive to MDD; it can analyze in detail any software development method that includes usability requirements elicitation.

This study is structured as follows. Section 2 reviews related works about usability requirements elicitation. Section 3 describes the design process of the MS. Section 4 shows the results obtained from the study. Section 5 presents a discussion about the results. Section 6 presents our conclusions and future work.

1.2 Related Work

Usability has been studied in several mapping studies and systematic reviews. The MS provides a systematic and objective procedure for identifying all the information that is available to answer a particular research question, topic area, or phenomenon of interest [29]. This section summarizes the different studies on requirements elicitation techniques, NFRs, and development methods based on usability.

First, we focus on studying techniques for capturing requirements that deal with usability. In this area, Dieste [13] updates a Systematic Review (SR) where interview-based techniques seem to be the most effective capture techniques. Carrizo [7] presents a framework to support decision-making, where some capture techniques respond better to certain project features than other capture techniques. Second, we focus on NFRs, since usability is considered by many authors to be a NFR. In the state-of-the-art written by Chung [11], the reviewed works are classified into six are- as: software variability, requirements analysis, requirements elicitation, requirements reusability, requirements traceability, and aspect-oriented development. Svensson [50] performs a SR to identify: elicitation requirements, metrics, dependencies, cost estimation, and prioritization as important areas for managing quality requirements. Mellado [34] carries out a SR about security requirements engineering in order to summarize evidence regarding security. The precision and reliability of the information are his main contribution. Mehwish [33] reports a SR to collect evidence of software maintainability prediction. The results suggest that there is little evidence for the effectiveness of these predictions. Third, we focus on studies that deal with methods to build usable systems. Folmer [20] performs a survey to explore the feasibility of a framework that can be applied to usability at the architectural level, taking into account design methods for usability design and evaluation tools. He concludes that there are no techniques for dealing with usability at the architectural level. In Fernandez's work [18], the objective of the MS is to summarize the current knowledge of methods in order to evaluate usability in web applications. The results show the need for usability evaluation methods that are specific to the web.

In summary, we state that most of the existing research publications related to usability are focused on: inclusion of usability features at the design stage; usability evaluation at early phases; methods to assess usability at the implementation stage; usability evaluation throughout the web development process; and techniques for usability specification during the software development process. However, we have not found mapping studies or SRs focused on usability requirements elicitation at early phases. We aim to study the existing literature concerning usability requirements elicitation in order to summarize current knowledge. This information will be used in a future work to design a framework for usability requirements elicitation using existing guidelines.

1.3 Mapping Study Design

The MS provides a wide overview of a research area to identify the quantity and type of research and results available within it. We considered the following elements: research questions, search strategy, selection criteria, quality assessment, data extraction strategy [28]. Next, we apply these elements to our MS.

Our **research question** is: “What are the proposals to elicit usability requirements throughout the software development process?”. It includes methods, notations, guidelines, tools, and empirical validations which are related to the usability area. The main goal is divided into six subgoals since the general research question is very abstract and involves many concepts. Each subgoal has been formulated as a research sub question. These are: *SQ 1.1 Methods to elicit usability requirements*. It aims to study whether or not the proposed methods (including NFR methods) can capture usability requirements at early stages; *SQ1.2 Methods to elicit interaction requirements*. It aims to study the existing methods to elicit interaction requirements related to usability. These methods are included because some authors improve usability by means of visual characteristics; *SQ 1.3 Usability guidelines to elicit usability requirements*. It aims to study the recommendations that help the analyst to identify usability requirements; *SQ 1.4 Tools to support usability requirements elicitation*. It aims to study the tools or prototypes that support the methods to elicit usability requirements; *SQ1.5 Notations to elicit usability requirements*. It aims to identify the existing representations in which the usability requirements are depicted. The target is to identify which notations are the most frequently employed for capturing usability requirements; *SQ1.6 Empirical validation environment*. It aims to study whether the proposal to elicit requirements was validated in an academic context or in industry.

The **search strategy** is composed of:

Defining the search sources. These sources are based on digital libraries that include peer-reviewed literature, such as: IEEExplore, ACM Digital Library, Springer Link, and Science Direct. Our main tool for searching in all these libraries was Sciverse Scopus, since it allows searching in all the mentioned digital libraries (among others). The sources explored were the proceedings of conferences, journals, books, and workshops. The search area is restricted to the computer science area. The search period is from 2000 to 2011.

Building and applying the search string. The search string is a set of terms to obtain the publications that answer the research question. Our search string is composed of two substrings: Usability Requirements and Software Engineering. With the first we collect publications related to how to elicit Usability Requirements, including software quality features and works related to requirements elicitation. The second substring is related to Software Engineering concepts based on requirements elicitation.

Search string = *(Usability Requirement) AND (Software Engineering)* Usability Requirement = *(usability requirement OR user requirement OR usability elicitation OR interaction requirement OR non-functional OR usability guidelines). Software Engineering = (MDD OR model-driven OR MDA OR notation OR tool OR interface OR engineering OR test).*

We have included the term "non-functional" into the "Usability Requirements" group since usability is frequently considered as a NFR.

The **selection criteria** contain:

Inclusion criteria (IC): IC1) Does the work define how to extract usability requirements?; IC2) Is the proposal applied to an environment based on MDD conceptual models?; IC3) Does the work define how to represent the requirements of usability?.

Exclusion criteria (EC): EC1) Publications focused on guidelines, notations, and tools where usability has not been considered or has not been included; EC2) Publications that consider only functional requirements; EC3) Publications written in a language that is not English.

Next, we **select the publications** through a systematic process:

Reading the title and the abstract. A total of 150 publications are returned by the search string, which are divided into three groups (50 publications) to be independently evaluated by three reviewers in order to apply the inclusion and exclusion criteria. The publications whose inclusion is doubtful must be discussed by the three reviewers until they arrive at a consensus. The result of this selection is a total of 65 publications, which are based only on the title and abstract of the publications. This selection is called "potential publications". Reading the whole publication. At this time, the whole publication is read. The inclusion and exclusion criteria are applied again for each potential publication, which are divided into three groups (one group per reviewer). The result of this selection is a total of 27 "initial selected publications", which are considered to be relevant.

Searching in references. In several cases there may be some relevant publications prior to the year 2000, such as Nielsen's work [36]. In order to avoid discarding these interesting older publications, we review all the publications referenced in publications from 2000 to 2011. If a publication was written before 2000 and it has not been referenced in the last 12 years, then that work is not relevant for the community, and it is therefore discarded from our study. The process to review the references of publications from initial selected publications obtains 5 publications. 2 publications support inclusion criteria and are added to initial selected publications. Finally, a total of 29 publications are our "selected publications".

In order to assess the reliability of inclusion, we apply the statistical measure of Fleiss' Kappa [19]. This statistic assesses the reliability of agreement between a fixed number of rates when classifying items. Its value ranges between 0 (poor agreement) and 1 (full agreement). We take a sample of 20 publications of the 65 potential publications, 10 of which are randomly selected and 10 of which are defined by the reviewers from the 29 selected publications. The Fleiss' Kappa value is 0.63, which is considered to be a "Considerable level".

Table 1. Likert-Scale Questionnaire

Subjective Questions	1=Yes	0=Partially	-1=No
1. Is the method to capture the usability requirements clear?			
2. Are the guidelines to capture requirements comprehensible?			
3. Are the guidelines to capture requirements useful in other contexts?			
4. Are the publications tools downloadable?			
5. Is there a clear case study or example illustrating the proposal?			
6. Is the whole proposal empirically validated?			
7. Are the results clearly explained?			
8. Is the notation to capture requirements easy to learn?			
Objective Questions			
9. Has the publication been published in journal or conference proceedings?			
	1=Very important	0=Important	-1=Not important
10. Has the publication been cited by other authors?			
	1= More than 4	0=Between 2 and 4	-1=Less than 2

In order to perform the **quality assessment**, we use the Likert-Scale to be filled out by three reviewers for each selected publication. Table 1 contains closed-questions that are classified into two groups: Subjective Questions and Objective Questions. For question Nº 9, we consider conferences at CORE ranking [38]. The publication is “Very important” if the conference is CORE A or B or if it is a book section, “Important” if the conference is CORE C or if it is a Workshop, “Not important” when the conference is not any CORE. For journals, the Journal Citation Report (JCR) [23] classification is used. The publication is considered to be “Very important” when it appears in JCR, “Important” when it does not appear in JCR but is indexed in other lists, and “Not important” when it is not published in any known list. For question N° 10, we use the H factor, which identifies the number of citations that each publication receives from other authors. The Publish or Perish [1] tool was used. In order to identify the quality of each publication, the three reviewers filled out the quality questionnaire. The aggregation of all the reviewers is performed by means of an arithmetic mean. After calculating the arithmetic mean for each question, we add these values, providing a single number between -10 and 10 which is denominated Quality Score. We consider that the Quality Score publication is “Very good" if it is more than 3, "Good" if it is between -2 and 2.99, and "Bad" if it is less than -2 (See Fig. 2b).

The **data extraction strategy** consists of classifying the possible answers for each research subquestion. The classifications are defined to facilitate the answer for our research question. These are:

- SQ1 Methods to elicit usability requirements. a) Yes b) No
- SQ2 Methods to elicit interaction requirements. a) Yes b) No
- SQ3 Guidelines to elicit usability requirements. a) Existing b) New c) Not exist
- SQ4 Tools to support the usability requirements elicitation a) Interface design (assistant to design) b) Model development c) Not Exist
- SQ5 Notations to elicit usability requirements. a) UML b) Natural Language (workshop sessions, checklists, questionnaires, heuristics, brainstorming, or interviews) c) i* framework d) CTT (Concur Task Trees) [40] e) Formal. (logical operators or grammars) f) QOC (Question Option Criteria) [31] g) BPMN h) Not Exist .
- SQ6 Empirical validation environment. a) Industrial b) Academic c) Not Exist.

1.4 Results

Summary sources from search studies. The selected publications used in our MS are published in different sources. Table 2 shows the 65 potential publications and the 29 selected publications, classified by conference, journal, book, workshop, and other sources. Table 3 shows publications presented in conferences only. They are classified by level of the conference according to the CORE list. Finally, Table 4 shows publications published in journals only. The classification is based on the JCR list.

Table 2. Publications by Source

Source	Potential	Selected
Conference	31	14
Journal	16	9
Book	4	3
Workshop	4	1
Other	10	2
Total	65	29

Table 3. Publications by Conferences

CORE	Potential	Selected
A	12	6
B	10	4
C	9	4
Total	31	14

Table 4. Publication by JCR

JCR	Potential	Selected
Yes	10	8
No	6	1
Total	16	9

Selected publication analysis. Table 5 shows the results of the 29 selected publications according to the data extraction strategy. Note that the answer for research subquestion SQ5 is not exclusive, i.e. more than one choice can be the answer.

Table 5. Mapping of selected publication

SQ1		SQ2		SQ3			SQ4			SQ5								SQ6			Quality	ID
A	B	A	B	A	B	C	A	B	C	A	B	C	D	E	F	G	H	A	B	C	Score	
	X		X		X				X			X							X		5,00	[14]
	X	X		X				X		X									X		3,67	[15]
	X		X		X				X								X		X		7,00	[16]
	X		X		X				X								X			X	1,00	[17]
	X	X				X		X						X					X		-1,00	[18]
	X		X	X				X			X		X							X	1,33	[19]
X			X			X			X		X								X		3,67	[20]
	X		X			X			X			X								X	1,00	[21]
	X		X	X					X	X		X							X		0,00	[22]
	X		X			X			X	X		X							X		-0,33	[23]
	X		X			X			X	X		X							X		-0,67	[24]
X		X				X	X			X			X			X			X		3,00	[25]
X			X	X					X	X	X	X						X			4,67	[26]
	X		X	X			X								X			X			-0,33	[27]
X			X			X		X				X							X		-2,00	[28]

	X		X			X			X	X									X		0,33	[29]
	X		X		X				X											X	0,33	[30]
	X		X	X					X		X									X	0,67	[31]
	X		X		X		X										X		X		4,00	[32]
	X		X		X				X								X			X	-2,67	[33]
	X		X			X			X	X	X								X		2,67	[34]
	X	X		X			X						X						X		5,00	[35]
	X		X			X		X		X									X		2,67	[36]
	X		X			X		X		X										X	0,33	[37]
	X	X				X			X	X			X						X		4,00	[38]
	X		X		X				X								X			X	1,33	[39]
X			X			X	X				X				X			X	X		7,67	[40]
X			X	X				X		X	X	X		X					X	X	6,67	[41]
X			X	X					X		X									X	4,00	[42]

SQ1: A) Yes 24.14% B) No 75.86%; SQ2: A) Yes 17.24% B) No 82.76%; SQ3: A) Existing 31.03% B) New 24.14% C) Not Exist 44.83%; SQ4: A) Interface Design 17.24% B) Model Development 24.14% C) Not Exist 58.62%; SQ5: A) UML 41.38% B) Natural Language 27.59% C) i* 27.59% D) CTT 13.79% E) Formal 6.9% F) QOC 6.9% G) BPMN 3.45% J) Not Exist 17.24% SQ6: A) Industrial 10.34% B) Academic 58.62% C) Not Exist 31.03%.

Next, we summarize the most relevant outcomes for each research subquestion:

SQ1 Methods to elicit usability requirements. There are few methods that propose capturing usability requirements, and usually they are included within NFR methods. In general, the requirements elicitation process uses traditional techniques (e.g. interviews, questionnaires, checklists, workshops) to elicit NFR at the same time the system functionality and architecture are defined [45], [14], [25]. The most common goals of the studied NFR methods are to elicit measurable NFRs such a way they can be evaluated [14], [24]. These methods can be customizable for a different context if some settings are applied to a specific context. Therefore, a holistic quality model that fits every context does not exist, and NFR methods only provide basic requirements management by means of extensions [14]. The major benefits are the enhancement of the communication between the stakeholders and an increase in the flexibility of their applications, although some methods [25] tend to use more resources than others. The results indicate a limited number of approaches that deal with usability requirements at early stages.

SQ2 Methods to elicit interaction requirements. Methods to specify interaction requirements are based on the construction of a model and the definition of structural patterns for different design solutions [38], [37], [6]. These models support the systematic analysis of interaction requirements that can be selected from artefacts like a library of interaction attributes [47], [45]. These methods improve usability by means of applying formal modelling to analyze interactive systems systematically [6]. How-ever, further work is needed to deal with dynamic specifications that depend on system functionality.

SQ3 Guidelines to elicit usability requirements. The publications aim to overcome the obstacle of the usability inclusion in the methods to elicit usability requirements and the different interpretations of the guidelines by the stakeholders. The methods that use existing guidelines, for instance ISO 9241-11 or ISO 9126, provide guidelines to determine usability requirements according to the definition of usability. They are understandable and can be implemented in a specific context [8], [32], [14], [51]; however, their application is not an easy task [25], [21], [47]. The guidelines related to functional usability features are more practical, but they need to specify the usability feature by means of design patterns in the architectural design [37]. On the other hand, the new guidelines show a variety of representations (e.g. catalogues, method-ologies, styles) [10], [22], [30] that are used to elicit usability requirements in different situations. All these representations allow to reuse its knowledge, to add new knowledge, to combine organizational memory or to combine different requirement scenarios. Other representations are based on patterns, templates, or models [27], [26], [38]. These artefacts can be improved or adapted according to which usability requirements are being captured. Nowadays, the guidelines do not provide precise, practical support to address usability requirements elicitation at the early stages.

SQ4 Tools to support usability requirements elicitation. These publications present tools to support: frameworks [45], structured styles [21], scenarios [48], notations [32], and methods [47]. The interface design tools support the requirements specification and validation through task flows and scenarios. Their main goals are focused on relating design options with functional and non-functional requirements within the design process of interactive systems. In order to reach this goal, it is necessary to incorporate a mechanism of transformation, (for example, from task flow diagrams to formal representations [45], [48]) and to solve traceability problems. The tools that are model-based can resolve this inconvenience by means of a global integration approach among notations and tools. However, this is not an easy task [4], since most tools focus specifications on requirements models or requirements metamodels. In order to define an elicitation process, the use of templates that are obtained through interviews [15], [16] or the use of patterns that provide a concise description of the users (detailing every significant characteristic [21]) are common.

SQ5 Notations to elicit usability requirements. The different notations are used in different stages of the software development process, and more than one notation is usually applied to the development method [28], [51]. The user requirements specifications are usually presented to end-users in normal text, even though the analyst works with languages based on models (SysML, UML). These requirements are based on a series of interviews and studies with end-users [46], [25], [14]. Some proposals aim to integrate functional requirements and NFR in the same elicitation process. These works propose a metamodel that combines UML with PLUS [51], [35], [45]. Therefore, UML and Natural Language are the most widely used notations (41.38% and 27.59%). In Formal notation, the specification is structured using hierarchical interfaces components that describe all the actions and visible attributes of the system [6]. In general, the other studied notations are currently supported by patterns, scenarios, and formatted templates in order to visualize and implement usability require-ments [6], [38], [48], [25]. These representations help analysts to elicit requirements, even though they are not always easily understood by the end-user.

SQ6 Empirical validation environment. We observe that case studies, experiments and illustrative examples that have been presented in Industrial or Academic environments do not have explicit metrics to evaluate the usability requirements elicitation. In general, existing validations are focused on quantitative [24], [35], [27] and qualitative usability requirements [25]. The users' usability evaluation is often based on test and usability scenarios [27]. All the studied publications share the same protocol for the empirical validation. First, the publication proposes a method, technique or model to elicit usability requirements. Second, the publication details the results of the validation. Third, there is a discussion where a qualitative analysis is performed in detail and some lessons learned are shown. [47], [27], [48], [15]. Studied publications are focused on evaluating a few usability features; however, the study of a reduced number of features is not enough to consider software as being usable. The patterns [6], [47], [37], [15], scenario management [48], [9], [21], checklists [14], work sessions [25], and templates [6] are the most common artefacts used to evaluate usability and other NFRs.

Graphics of mapping results. We present four graphics of the MS results. Two correspond to comparison between research subquestions and the others correspond to the potential and selected publications and to the Quality Score of the selected publications. The six research subquestions give us an overview of the usability require-ments and how they are related. Apart from reinforcing our conclusions of this study, this information can highlight some gaps that should be researched further.

Fig. 1a shows comparisons between research subquestions SQ1, SQ2, SQ3, and SQ4. The most important outcomes are the following: there is not any new guideline to elicit usability requirements or interaction requirements; there is the same number of publications where the tool is a support for interface design and model development; there are a large number of publications that do not address methods of usability requirements elicitation or methods of interaction requirements elicitation.

Fig. 1b shows comparisons between the research subquestions SQ4 and SQ5. The most important outcomes are the following: UML, Natural Language, and CTT are notations used by model development tools and by design interface tools; BPMN and QOC are notations that are not used by model development tools; i* and Formal are notations that are not used by interface design tools.

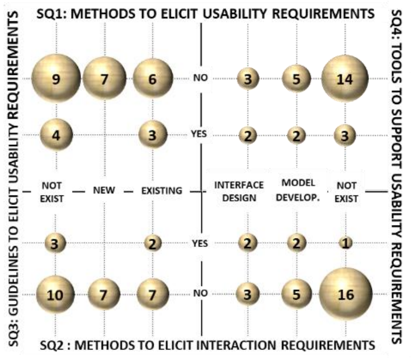


Fig.1a) Mapping results SQ1,SQ2,SQ3,SQ4

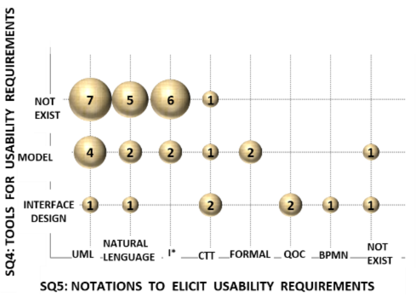


Fig.1b) Mapping results SQ4,SQ5

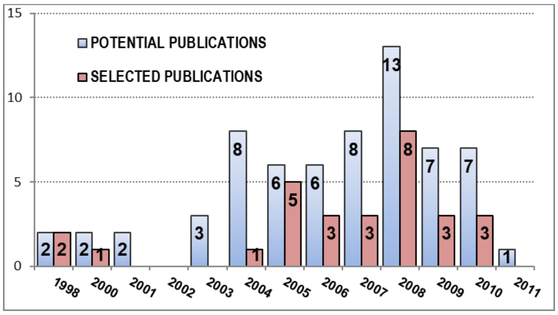


Fig. 2a) Frequency of publications by year

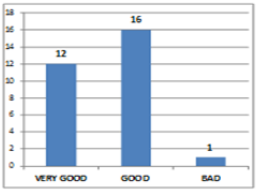


Fig. 2b) Publications by Quality Score

Fig. 2a shows the number of potential publications and selected publications classified by year. It can be observed that there are very few publications published each year. Of the 29 selected publications, 8 of them were published in 2008. This is the year that had the most publications for improving usability requirements elicitation. The year 1998 is included in the graphics because the two publications obtained from the referenced publications were published that year. None of the selected publications were published in 2001, 2002, 2003, and 2011.

Fig. 2b shows a frequency graphic that describes the quality assessment of the selected publications. This graphic is obtained from the Quality Score of selected publications, which can be “Very good”, “Good”, and “Bad”, according to our quality criteria. The graphic shows a high number of publications that are considered to be “Good” publications and “Very good” publications. Both results make up 95% of the total of the selected publications

1.5 Discussion

In the selected publications, the usability requirements elicitation is usually performed at the analysis stage [46], [15], i.e., once all functional requirements have been captured. This late capture involves changes in system architecture since some usability requirements are related to functionality [5], [20]. In general, the methods used to elicit usability requirements deal with usability when the functional requirements have been previously captured by means of traditional techniques (e.g. interviews, questionnaires, focus groups, use cases) [35], [3].
The analysis of the results shows that there are very few publications that clearly address how to perform the capture process of usability requirements at early stages. Moreover, existing approaches do not propose a precise and unambiguous notation to represent these requirements, which makes difficult to apply them in real systems. There are some publications where usability requirements elicitation is performed at the design stage together with interaction requirements elicitation [25], [45], [24].
When the usability topic is dealt with at requirements elicitation, the ISO standards are used as guidelines to be applied in software development systems. For instance, the ISO 9241-11 is considered to be a basic reference for some practitioners, re-searchers, and designers [25], and for any kind of requirements the standard ISO 9126-1 is used [32]. The application of guidelines is necessary, but it is not enough; the main problem is the correct application and complete understanding by the end user. Guidelines are only built up in a general way, but they are not a total support for usability system development. There are some

proposals that aim to help the require-ments engineers to address usability requirements from the early stages by means of GUIDE rules [22] and a catalogue based on the i* framework [10]. Both techniques are context-specific, even though GUIDE uses a case-based repository for taking decisions and i* framework collects a large amount of knowledge to achieve usability goals.

Another aspect that is observed in selected publications is the use of artefacts, such as: patterns, scenarios, and templates, which are frequently used as support for methods to elicit usability requirements and interaction requirements [6], [48], [16]. The methods proposed in the selected publications are inflexible and require considerable effort to be applied in contexts that are different from the contexts where they have been defined [22]. The guidelines, notations, and artefacts used in these methods are closer to elicit interaction characteristics rather than usability characteristics. In general, guidelines for usability requirements elicitation are defined in a very generic way for different abstraction levels [8].

The tools to represent usability requirements which are based on a conceptual model have great possibilities of being useful for building extensions to other models (e.g. finite state machine) [45] or for being used in different contexts with other usability requirements. For large project, these tools are too limited, since the identification of requirements and modularization of the system need more special processes, methods and techniques. Moreover, once these requirements have been structured and gathered in a tool, they could be reused in later projects. Only few approaches include tools to support existing eliciting methods. Most approaches must be applied manually, or they require a tool that is not provided by the authors [17], [42], [38]. This makes difficult the adoption of those approaches in industrial environments. The necessity of a tool is more urgent in those proposals that use several notations and combine the use of different artifacts (e.g. templates, questionnaires, workshops) [30], [14], [47]. Working with all these items manually is a huge effort for the analyst.

Validation methods are another crucial aspect for the evaluation of a proposal. The selected publications present case studies, experiments, and examples that do not show whether or not the inclusion of usability requirements produces a positive im-pact on the final product. In

addition, only a small percentage of proposals have been applied in an industrial context [24].
Many works propose eliciting usability requirements with a graphical notation [10], [9], [6]. This enhances the abstraction for the requirements engineer but some-times can difficult the end-user participation, who usually cannot understand those notations. Other proposals elicit usability requirements textually [25], [8], [48] facilitating the end-user participation. However, these proposals cannot be used for a development method based on models, since models do not exist.
If we focus our analysis on approaches to capture usability requirements in MDD environments, we notice that there are few proposals [38], [17], [46], [4]. Moreover, usability requirements are not usually considered as a main topic in those proposals. Usability requirements are combined with other NFR or with functional requirements, which makes difficult to focus the elicitation process on usability issues. Moreover, transformations among models are not discussed in those publications even though this is a basic pillar in the MDD paradigm (where transformations can be automated or semi-automated). Another problem of the existing proposals within the MDD paradigm is that there are not evaluations or tools to demonstrate that they can work in real systems. Existing approaches are just theoretical proposals that have not been implemented yet.
Note that our mapping study has some limitations. The first one is that we cannot ensure that all existing publication related to usability requirements have been considered. We have focused our research on Scopus, which is a tool that looks for publications in several digital libraries, such as IEEExplore, ACM Digital Library, Springer Link, and Science Direct (among others). In order to minimize the loss of some important publications, we have analyzed references from publications retrieved by Scopus. However, publications that have not been published in those libraries or publications that have not been referenced are out of our search. Second, some found publications were not accessible (our university had no license to read them). This happened with 6 publications from 65. If we compare inaccessible publications with the total amount of publications, we notice that the percentage of unread publications is a minimum portion 9.23%.

Throughout the whole mapping study we have been guided by an expert at mapping studies and systematic reviews. This expert helped us in the application of the protocol and recommended us some tools. For example, the use of Refworks [41] to eliminate duplicities in our search of publications, since the search string can find the same publication more than once.

1.6 Conclusions and Future Works

This MS combines usability aspects from both the Software Engineering (SE) community and the Human-Computer Interaction (HCI) community. We have explored the development methods that consider usability as a requirement from the SE community. We have studied the guidelines and heuristics from the HCI community that are used to develop usable applications. The MS aims to review existing studies related to usability requirements in both communities. Our main target is specially focused on proposals to elicit usability requirements from the early stages of the software development process.

The MS has been performed according to Kitchenham’s methodology, focusing on the last 12 years. A total of 29 publications were selected from an initial set of 150 publications returned by the search string. The quality assessments of the publications were developed in order to contrast the significance of the selected publications, where 97% is composed of good publications and very good publications.

Using the results of the MS, we can conclude that there is a clear research line in the field of usability requirements in MDD environments. Usually, MDD methods have historically been focused on modelling behaviour and persistency, but relegating interaction (and particularly usability) to manual implementation. This manual implementation clearly contradicts the MDD paradigm, which advocates that the analyst must work with holistic conceptual models, where every feature of the system (including usability features) could be represented. We plan to develop a framework to elicit usability requirements in such a way that it could be used in any MDD method.

The main benefit of embedding usability requirements in a MDD method is that the next steps of the software development process can be derived from the requirements elicitation step. We plan to develop transformation rules from the usability requirements to generate analysis and design models. Furthermore, the MS can also be used as a starting point for future systematic reviews based on usability requirements elicitation.

References

1. Publish or Perish, http://www.harzing.com
2. Acerbis, R., Bongio, A., Brambilla, M., Butti, S.: WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications. In 7th International Conference on Web Engineering, Springer-Verlag, Berlin, Heidelberg, 501-505. (2007)
3. Akoumianakis, D., Katsis, A., Vidakis, N.: Non-functional User Interface Requirements Notation (NfRn) for Modeling the Global Execution Context of Tasks. In 5th International Conference on Task Models and Diagrams for Users Interface Design, Springer-Verlag , Hasselt, Belgium, 259-274. (2007)
4. Ameller, D., Franch, X., Cabot, J.: Dealing with Non-Functional Requirements in Model-Driven Development. In 18th IEEE International Conference on Requirements Engineering (RE). Sydney, NSW, 189-198. (2010)
5. Bass, L., John, B.: Linking Usability to Software Architecture Patterns through General Scenarios. Journal of Systems and Software, Vol. 66, No. 3, 187-197. (2003)
6. Campos, J., Harrison, M., Graham, T., Palanque, P.: Systematic Analysis of Control Panel Interfaces Using Formal Tools Interactive Systems. Design, Specification, and Verification. Springer-Verlag, Vol. 5136, Berlin, Heidelberg, 72-85. (2008)
7. Carrizo, D., Dieste, O., Juristo, N.: Study of Elicitation Techniques Adequacy. In 11th Workshop on Requirements Engineering. Spain, Barcelona, 104-114. (2008)
8. Cronholm, S. and Bruno, V.: Do you Need General Principles or Concrete Heuristics?: A Model for Categorizing Usability Criteria. In 20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat, ACM, Cairns, Australia. (2008)
9. Cysneiros, L. M., Leite, J.C.S.P.: Nonfunctional Requirements: from Elicitation to Conceptual Models. IEEE Trans. on Softw. Eng., Vol. 30, No. 5, 328-350. (2004)
10. Cysneiros, L.M., Werneck, V. M. Kushniruk, A.: Reusable Knowledge for Satisficing Usability Requirements. In 13th IEEE

International Conference on Requirement Engineering, IEEE Computer Society, Washington, DC, USA, 463-464. (2005)

11. Chung, L. Leite, J.C.S.P.: On Non-functional Requirements in Software Engineering. LNCS, Springer, Vol. 5600, Berlin, Heidelberg, 363-379. (2009)
12. Daniel, S., Rita de Almeida, P., Isbela, M.: OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW. SIGWEB Newsl., Vol. 8, No. 2, 18-34. (1999)
13. Dieste, O., Lopez, M., Ramos, F.: Updating a Systematic Review about Selection of Software Requirements Elicitation Techniques In 11th Workshop in Requirements Engineering, Barcelona, Spain. (2008)
14. Doerr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T.: Non-functional Requirements in Industry - Three Case Studies Adopting an Experience-based NFR Method. In 13th IEEE International Conference on Requirements Engineering, Washington, DC, USA, 373-384. (2005)
15. Escalona, M.J., Arag, G.: NDT. A Model-Driven Approach for Web Requirements. IEEE Trans. Softw. Eng., Vol. 34, No. 3, 377-390. (2008)
16. Escalona, M.J., Koch, N., Filipe, J., Cordeiro, J., Pedrosa, V.: Metamodeling the Requirements of Web Systems Web Information Systems and Technologies. Springer-Verlag, Berlin, Heidelberg, Vol 1, 267-280. (2007)
17. Fatwanto, A. and Boughton, C.: Analysis, Specification and Modeling of Non-Functional Requirements for Translative Model-Driven Development. In International Conference on Computational Intelligence and Security, Washington, DC,USA, 405-410. (2008)
18. Fernandez, A., Insfran, E., Abrahão, S.: Usability Evaluation Methods for the Web: A Systematic Mapping Study. Information and Software Technology, Vol. 53, No. 8, 789-817. (2011)
19. Fleiss, J.L.: Statistical Methods for Rates and Proportions. John Wiley & Sons, New York, Ed. (1981)
20. Folmer, E., Bosch, J.: Architecting for usability: A Survey, Journal of Systems and Software, Vol. 70, No. 1, 61-78. (2004)
21. Grosse-Wentrup, D., Stier, A., Hoelscher, U., Dössel, O., Schlegel, W.C.: Supporting Tool for Usability Specifications. In World Congress on Medical Physic and Biomedical Engineering. Springer-Verlag, Munich, Germany, 845-847. (2009)

22. Henninger, S.: A Methodology and Tools for Applying Context-specific Usability Guidelines to Interface Design. Journal Interacting with Computers, Vol. 12, No. 3, 225-243. (2000)
23. Journal Citation Reports, http://ip-science.thomsonreuters.com
24. Jokela, T., Koivumaa, J., Pirkola, J., Salminen, P., Kantola, N.: Methods for Quantitative Usability Requirements: A Case Study on the Development of the User Interface of a Mobile Phone. Personal Ubiquitous Comput., Vol. 10, No. 6, 345-355. (2006)
25. Jokela, T., Seffah, A., Gulliksen, J., Desmarais, M.C.: 8 Guiding Designers to the World of Usability: Determining Usability Requirements Through Teamwork. Springer Netherlands, Vol. 8, 127-145. (2005)
26. Juristo, N.: Impact of Usability on Software Requirements and Design. Springer-Verlag, Vol. 55-77. (2009)
27. Juristo, N., Moreno, A. M., Sánchez, M. I.: Guidelines for Eliciting Usability Functionalities, IEEE Trans. Softw. Eng., Vol 33, No. 11, 744-758. (2007)
28. Kitchenham, B.: Procedures for Performing Systematic Reviews, Technical Report TR/SE-0401. (2004)
29. Kitchenham, B. A., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering, EBSE Technica Report. (2007)
30. Lauesen, S. Younessi, H.: Six styles for usability requirements. In REFSQ'98 (1998)
31. MacLean, A., Young, R. M., Bellotti, V. M. E., Moran, T.P.: Questions, Options, and Criteria: Elements of Design Space Analysis. Human-Computer Interaction, Vol. 6, No. 3, 201-250. (1996)
32. Martinie, C., Palanque, P., Winckler, M., Conversy, S., DREAMER: A Design Rationale Environment for Argumentation, Modeling and Engineering Requirements. In 28th International Conference on Design of Communication. Säo Paulo, Brazil. (2010)
33. Mehwish, R., Emilia, M., Ewan, T.: A Systematic Review of Software Maintainability Prediction and Metrics. IEEE Computer Society, Washington, DC, USA, 367-377. (2009)
34. Mellado, D., Blanco, C., Sánchez, L. E., Fernandez, E.: A Systematic Review of Security Requirements Engineering. Comput. Stand. Interfaces, Vol. 32, No. 4,153-165. (2010)
35. Nguyen, Q.L., Non-Functional Requirements Analysis Modeling for Software Product Lines. In ICSE Workshop on Modeling in Software Engineering, Washington, DC, USA, 56-61. (2009)

36. Nielsen, J.: Usability Engineering. Morgan Kaufmann. (1993)
37. Panach, J.I., España, S., Moreno, A. and Pastor, Ó.: Dealing with Usability in Model Transformation Technologies. In ER 2008, Springer LNCS Barcelona, 498-511. (2008)
38. Panach, J.I., España, S., Pederiva, I., Pastor, O.: Capturing Interaction Requirements in a Model Transformation Technology Based on MDA. Journal of Universal Computer Science (JUCS), Vol. 14, No. 9, 1480-1495. (2007)
39. Pastor, O., Molina, J.: Model-Driven Architecture in Practice. Springer, Ed. (2007)
40. Paterno, F.: Model-based Tools for Pervasive Usability. In Interacting with Computers 17 (3), Elsevier, 291-315. (2004)
41. Refworks, http://www.refworks.com/
42. Röder, H.: Using Interaction Requirements to Operationalize Usability. In ACM Symposium on Applied Computing, Sierre, Switzerland. (2010)
43. Sajedi, A., Mahdavi, M., Pourshirmohammadi, A., Nejad, M. M.: Fundamental Usability Guidelines for User Interface Design. In International Conference on Computational Sciences and Its Applications ICCSA.Washington, DC, USA, 106-113. (2008)
44. Shehata, M., Eberlein, A., Fapojuwo, A., O.: A Taxonomy for Identifying Requirement Interactions in Software Systems. Comput. Netw., Vol. 51, No. 2, 398-425. (2007)
45. Sindhgatta, R. and Srinivas, T. Functional and Non-Functional Requirements Specification for Enterprise Applications. Springer-Verlag,Vol. 3547, Berlin, Heidelberg, 189-201. (2005)

2.2 Towards a proposal to capture usability requirements through guidelines

The Model-Driven Development (MDD) paradigm states that analysts can build a conceptual model that represents the system abstractly. This conceptual model is the input for a set of transformation rules that can generate the code that implements the system automatically. Nowadays, there are sound MDD methods that deal with functional requirements, but, in general, usability is not taken into consideration from the early stages of the development. Analysts who work with MDD implement usability features manually once the code has been generated. This manual implementation contradicts the MDD paradigm, and it can affect the system architecture, involving a lot of reworking. This paper proposes a method to capture usability requirements at the early stages of the software development process in such a way that non-experts in usability can use it. The approach consists of organizing several interface design guidelines and usability guidelines in a tree structure. These guidelines are shown to the analyst through questions that she/he must ask the end-users. Answers to these questions mark the path through the tree structure. At the end of the process, if we gather all the end-user's answers, we have the usability requirements. Then, by means of model to model transformations, we could transform usability requirements into a conceptual model of any existing MDD method

Versión del autor del artículo: Ormeño, Y. I., Panach, J. I., Condori-Fern, N., & Pastor, Ó. (2013, May). Towards a proposal to capture usability requirements through guidelines. In IEEE 7th International Conference on Research Challenges in Information Science (RCIS) (pp. 1-12). IEEE, 10.1109/RCIS.2013.6577677

2.1 Introduction

The Software Engineering (SE) community has been working for several years on the Model-Driven Development (MDD) paradigm [1], which states that the analysts' entire effort should be focused on a conceptual model, and the system should be implemented by means of model to code transformations. In MDD, a conceptual model is used to represent a system, independent of the platform and technology. This conceptual model is the input for a model compiler which includes transformation rules to generate the code according to the target platform.

Even though existing MDD methods (e.g. WebML [2] or UWE [3]) are very powerful for building conceptual models, they do not have a process to capture usability requirements. In general, usability features are manually implemented once the code has been generated. This manual implementation contradicts the MDD paradigm, which proposes focusing the analyst's entire effort on building a holistic conceptual model. According to Bass [4] and Folmer [5], these manual changes may involve changes in the system architecture, which can result in a lot of extra effort. Moreover, these manual implementations can produce a source code that contradicts the system's characteristics expressed in the conceptual model.

So, why are usability requirements not captured in the early software development stages together with functional requirements? One reason for this is that usability is strongly related to human behavior (software psychology [6]) and, unfortunately, analysts who capture system requirements are not experts in this field. In order to facilitate the software development process, the Human Computer Interaction (HCI) community has defined usability guidelines for non-experts in usability.

For example, Shneiderman [7], and Nielsen's [8] usability design guidelines are widely accepted and used as tools to measure usability. However, these guidelines are usually described in such an abstract way that they are difficult to apply (directly) in software development. Moreover, the evolution and presence of new technologies and communication devices encourages the development of usability guidelines oriented to different platforms (contexts) such as: the Web, development tools, phones, tablets and media devices [9]. According to Nielsen [10], there are around 2394 guidelines. The Web is the software platform with the most guidelines. It contains 874 user-experience design guidelines, 144 guidelines for commercial businesses, 103 for corporate sites and 614 usability design guidelines on the intranet. This huge number of guidelines hinders the analyst when he/she is searching for the most suitable guideline for a specific system.

Thus, the main contribution of this work is to define an approach to facilitate the usability requirements capture process for analysts who are not experts in usability engineering. This approach can be included in an MDD method in such a way that these requirements generate part of the conceptual model of the MDD method. This is in accordance with the MDD paradigm, which states that models used in the early stages of the software development process can be transformed into models for the next stages. The approach is based on textual questions, and design alternatives for each question that end-users must be asked relevant questions, and design alternatives, are extracted from interface design guidelines and they are represented in a tree structure. End-users must choose which alternative is the most suitable according to their requirements (or constraints). Usability guidelines can help the end-user select an alternative throughout the tree structure. At the end of the process, we have a design for our system based on the end-user's requirements. This design can be embedded in a conceptual model of an existing MDD method through transformation rules.

This paper is divided into the following sections: Section 2 presents the state of art of various approaches made by other authors concerning the use of usability guidelines; section 3 describes the concepts that are involved in the usability requirement capture approach; section 4 explains the proposed scheme to capture usability requirements viewed from both the analyst's and the expert's side; section 5 presents a proof of concept based on an example, and finally, Section 6 describes the conclusions and future work.

2.2 Related Work

The literature presents a lot of usability guidelines to support the design of user interfaces, but they may confuse the analyst if she/he is not an expert in usability. In general, the analyst may face the following problems (among others): it is not easy to understand how to apply the guideline; sometimes it is difficult to determine when a guideline has been broken; and, some guidelines are so ambiguous that they are difficult to apply to specific contexts. All these aspects require a huge effort on the part of the analyst that leads us to determine if the usability guidelines are still usable.

Cronholm's work [11] and Henninger's work [12] describe possible solutions to some of these problems. Cronholm's work proposes meta guidelines as a solution to obtain more systematic and categorized guidelines. These meta guidelines consist of a set of principles whose objective is to improve the usability of the guidelines. Design guidelines defined by Henninger include two types of guidelines: interface principles, or typed rules, and usability examples, also known as cases. These cases are examples of specific interfaces developed for organizations that contain a lot of knowledge about the needs and common practices of clients' work.

Furthermore, Cysneiros's work [13] proposes a reusable catalogue to capture usability requirements. The method is based on i* framework and it uses personal experiences to obtain knowledge to achieve the objectives of usability. His work shows how usability can be modeled through different views with different alternatives. Bevan [14] makes a comparison between three guidelines: HHS for a Web site, JISC for Web services, and ISO 9241-151, which includes principles and specific solutions (conceptual models, task structure, and navigational structures). Bevan highlights differences and similarities between these three guidelines. He states that a perfect set of guidelines does not exist, since the necessities of different audiences are not homogeneous.

The cited works aim to mellow the ambiguity of the usability guidelines, but they increase the complexity of use for non-experts in usability. All these solutions involve a lot of effort to understand all the guidelines and choose the most suitable one for a specific context. For example, understanding the notation, or the information arrangement in a guideline may involve some of the analyst’s effort in order to use the guideline optimally. Furthermore, the comparison of guidelines shows great variability, which leads to creating specific usability guidelines for specific domains.

Usability guidelines for the Web and for WAP mobile phone applications are widely used. Pei [15] states that web design should be focused on the user Web site to improve usability. The design of a usable web is made up of the following three elements: user research, web design, and usability evaluation. On the other hand, the usability of mobile phone applications is increasing, although it is lower than Web Sites accessed by computer [16]. Sabine's work [17] proposes usability guidelines to design applications based on WAP. This author compares two versions of a travel management Web Site, one which includes usability guidelines of design and the other which does not. The results show that user-experience of the Web site which uses usability guidelines is higher than mobile phone or Smartphone applications with standard features.

The literature provides a wide range of usability guidelines for web sites, web applications, desktop applications, mobile phones and others [10]. Some examples of usability guidelines are: development tools (AJAX, RIA), User Interface (Apple Mac OSX, iPad user experience) platform (Window XP, Vista User Experience Interaction) Interface Software Mobile (Android, Nokia top 10, WebOS) among others [9]. Moreover, these existing guidelines are continuously in state of change and development especially for mobile phone Internet services looking to improve usability.

Some examples of methods used to capture usability requirements are: a method for quantitative usability requirements applied in user interfaces to depict the true usability [18]; multimedia user interface designs that design attractive and usable multimedia systems [19]; and, embedded Functionality Usability Features in model transformation technologies [20]. We can state that there are many proposals but none of them clearly and concisely addresses how to perform the extraction process of usability requirements in the early stages.

This paper proposes a method to organize the information stored in different usability guidelines. This way, analysts without a background in usability can work with the guidelines. Based on a review of the literature, we can say that for the MDD paradigm very few papers have been written that address how to perform the extraction process of usability requirements. Generally, this task is done when the usability requirement capture has been done. Moreover, usability requirement capture has not been developed focusing on the MDD method. This paper aims to cover this gap, proposing a process to capture usability requirements such a way they can be transformed later into part of the conceptual model of the MDD method.

2.3 Proposal to Capture Usability Requirement

This section describes our approach to capture usability requirements within the MDD paradigm. Based on the ISO 9241-11 [21] standard, the usability requirements are the effectiveness, efficiency and satisfaction of a user achieving his/her goals in a defined context of use. Our approach is based on existing usability guidelines, and design guidelines, that are stored in a tree structure. The analyst navigates through this structure in order to capture the usability requirements by asking the end-users questions. The tree structure helps the analyst to identify the different design alternatives, and how these decisions will affect the system's usability. Figure 1 shows the elements used in our approach. Next, we describe each element:

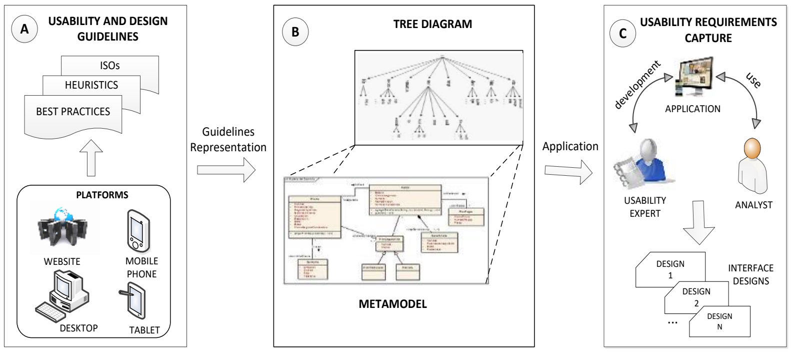


Figure 1. Schema of the proposal to capture usability requirements.

A. *Usability guidelines and interface design guidelines*

Both usability guidelines and interface design guidelines have been created to guide the analyst to develop systems. Usability guidelines recommend how to combine users, tasks, and context to enhance the system usability [21]. Interface design guidelines provide alternatives and recommendations for design systems [22]. These guidelines have been built for different technologies and platforms which are represented by standards, principles, heuristics, styles, patterns, best practices, etc. Both types of guidelines are related to each other since some design guidelines can improve or decrease the usability (depending on the combination of tasks, users and context). Working directly with both kinds of guidelines [23], [24], [21], implies a huge effort as the variability and amplitude of these guidelines is very high. In order to reduce this effort, we propose storing all the relevant guideline information in a tree structure, which is explained in more detail below.

B. *Tree Diagram*

In this context, we propose using these guidelines by means of a tree structure in order to minimize the cognitive effort to work with both types of guidelines. A tree structure is defined as a connected graph with no cycles and a root [25],[26]. Figure 2 shows a general schema of the tree structure used in our approach, which is composed of four elements: question, answer, group of questions, and designs. In the next part, we will present these elements:

1) *Question(Q_i):* The design guidelines present diverse design alternatives for many UI (User Interface) components (e.g. menu). In order to ask the end-user which alternative she/he prefers, we have defined a question when alternatives to design appear. For example, when we are designing dialog elements for mobile, design guidelines [27], [24] specify that dialog elements provide a top-level window for short-term tasks and a brief interaction with the user. We can define a question to decide which is the UI component to represent a selectionable task, *Which UI component is used to show selectable tasks?*. This question could enable the user to complete a specific task. In Figure 2, questions are represented by Q_i.

2) *Answer(A_i):* These are the exclusive options for each question according to interface design guidelines. These options are presented to the analyst in such a way that she/he can choose which one best fits the user's requirements. The analyst's decision is not only based on end-user criteria, but also on usability guidelines. This means that we have related answers with usability guidelines depending on the type of user, type of task, and type of context. When the answers are shown to the analyst, we will show which answers are recommended by usability guidelines. For example, the answers to the question *"Which UI component is used to show selectable tasks?"* can be: radio buttons, text field, checkboxes, slider [24],[27]. Mobile design guidelines [28] advise using a UI component dialogue to show tasks as information that require users to take an action before they can proceed. The usability recommendations are identified when answers have been defined. For example a radio button is constructed for a persistent single-choice list [24], where aspects such as

"simplify navigation" and *"minimize user input"* are usability requirements [28]. In Figure 2, answers are represented as A_i, A_{i+1}, … , A_n.

3) Group of Question (GQ_i). Some branches of the tree structure are not mutually exclusive (the end-user should be asked all of the questions). This type of branch is represented by a group of questions, which gathers several questions grouped by a design characteristic. For example, the question *"Which UI component is used to show selectable tasks?"* can be gathered with other questions that ask about Selection Dialogues, such as *"Where is the action button located?"*, *"Where is the dialogue box located?"*, and *"Where is the positive action on button located?"*. All these questions have also in common that deal with how the selection dialogs are displayed, and all of them are gathered in the same Group of questions. In the tree structure these are represented as GQ_i, in Figure 2.

4) *Designs (D_i)*: These are the interface designs reached through the alternatives that the analyst has been choosing. The analyst navigates through the tree structure asking the questions to, the end-user, who selects the most suitable answer (usability guidelines can recommend some answers). When the analyst reaches a leaf in the tree, a design has been obtained. The final design of the whole system is the set of leaves in the tree that the analyst has reached. For example, a design can be a selection dialog with radio buttons, where each item shows an enumerated data [27],[24]. At the tree structure these are represented as D_i, in Figure 2.

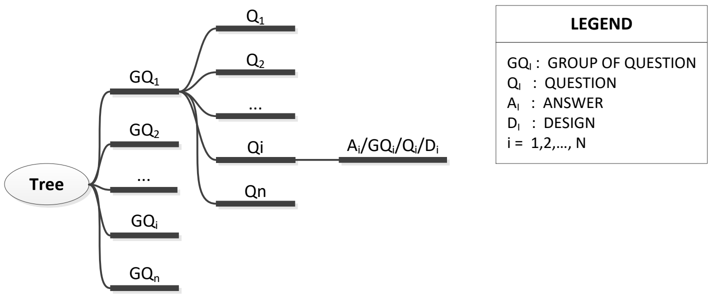


Figure 2. General representation of the tree structure of a figure caption

The navigation starts from the root of the tree while the analyst asks the questions to the end-users. The analyst asks the questions according to their sequence in the tree, from the root to the leaves. Questions are mutually exclusive, in other words, the analyst only navigates through the branch of the answer selected by the end-user. Questions that are gathered in the same group of questions are all asked. When the analyst reaches a branch with a group of questions, the flow continues with the first question in the group. Only when this flow has finished, can the analyst continue with the next question in the group. The possible navigation between two nodes of the tree structure can be: i) From a group of questions to a question, or to another group of questions ($GQ_i \rightarrow Q_i$ / GQ_i); ii) From a question to an answer ($Q_i \rightarrow A_i$); iii) From an answer to a question to a group of questions or to a design ($A_i \rightarrow Q_i$ / GQ_i / D_i).

Note that if we work with several usability guidelines, they can contradict each other when they recommend an answer. This contradiction is not a problem in our approach, since usability guidelines are only recommendations. The choice of the most suitable answer only depends on the analyst and on the user's requirements.

One advantage of our approach is that designs reached throughout the navigation in the tree can be transformed into a conceptual model of a MDD method. For this aim, each design of the tree must have a transformation rule to generate part of the conceptual model of the target MDD method, as Figure 3 shows. In order to facilitate these transformations, we recommend using UsiXML (USer Interface eXtensible Markup Language) [29] as the language to specify the designs. UsiXML is an XML-based markup language for defining user interfaces which is widely used in the academy. The main advantage of using UsiXML is that a framework has already been defined to support interface modeling, and there are also transformations from UsiXML to some MDD methods, which facilitates the transformation work.

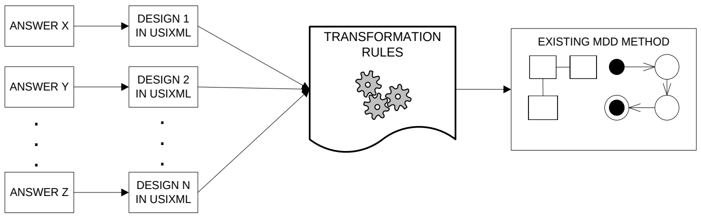


Figure 3. General process to generate a conceptual model from the designs

In order to formalize all the elements that compose the tree structure, we have defined a meta-model (Figure 4). Below, we describe its classes.

Class Design Guideline represents the interface design guidelines used in our tree structure. Questions that the end-user will be asked in order to discover which design alternative is most suitable are derived from these guidelines. Every question can be related to a Group of questions, or to at least two Answers. The class Group of questions represents the set of questions we can define, and the class Answer specifies the exclusive alternatives for the question. Some of these answers can be recommended by one or several usability guidelines, recommendations, standards and best practices, represented as instances of the class Usability Guideline.

According to the usability definition described in ISO-9241 [21], some usability guidelines are specific for a context, task or user [30],[31]. This is represented through the classes Context, Task, and User respectively. The class Context describes the context where the guideline is recommended, the class Task describes the type of task for which the guideline is recommended, and class User describes the type of user for which the task is recommended. Context, Task and User are related to class Description, to describe how they enhance the system's usability. Finally, class Design represents the designs that the analyst can get to at the leaves of the tree. Each instance of this class is a different interface design which we can reach through different answers.

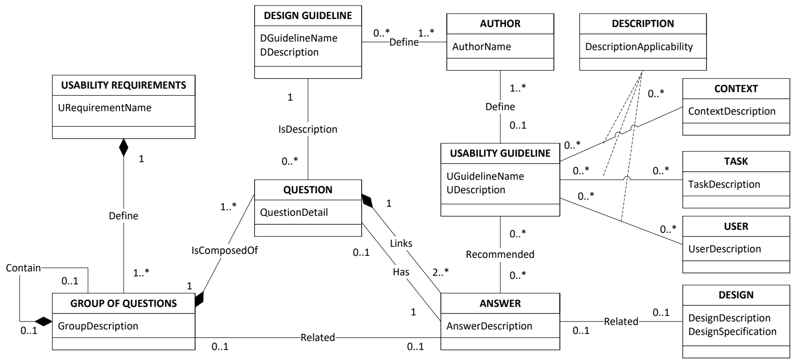


Figure 4. Meta-model of usability requirements capture

C. Usability requirement capture

The usability requirement capture is the process to capture usability requirements using our approach. The next section explains how to build the tree structure, and how to use it in the requirement capture process.

2.4 Process to Capture Usability Requirement in MDD

This section describes the process to build an instance of the meta-model shown in Figure 4. This instance will be used later to capture usability requirements. Three stakeholders participate in this process: an expert in usability, an analyst and the end-user. In the next section we will explain how the stakeholders participate in both activities: the construction of the tree structure and requirement capture.

A. Phase of construction

This phase is performed by the usability expert and the analyst. First, the usability expert builds the tree structure using interface design guidelines and usability guidelines.

Second, the analyst specifies the transformation rules to transform the designs into a conceptual model of a MDD method. Figure 5 summarizes all the steps that make up this phase. Below we detail all of them.

SE1) Analysing the usability guidelines and interface design guidelines: The usability expert looks for existing interface design guidelines and usability guidelines that can be applied to build the tree structure.

In the literature there are many guidelines, the expert must choose on those guidelines focused on the type of systems we aim to build using the tree structure. Then, an analysis of these guidelines is required to identify the relevant aspects for designing usable systems. It is important to point out that this identification of relevant aspects depends on the experience level of the "usability expert" to appropriately construct the tree. The identification of these relevant aspects depends on the experience of the usability expert.

SE2) Defining the question: Using interface design guidelines, the usability expert defines the questions. When there is a set of possible alternatives for a design, the expert must define a question in order to ask the user which is the most suitable alternative.

SE3) Defining the answer: Each alternative to a question is expressed as a possible answer for that question. According to the tree structure, after specifying an answer the usability expert has several possibilities: (1) To define another more specific question (if we need more information to determine the final design); (2) To define a final design (if we have reached a leaf in the tree because there are no more alternatives); (3) To define a group of questions (if the answer leads to more than one related questions).

SE4) Recommending usability guidelines: Usability guidelines may recommend some answers. In this step, the usability expert defines which answers are recommended by which usability guideline. Recommendations can be given with respect to any of the elements: context, task, or user. The relationship between answers and usability guidelines is not mandatory, but the more guides we provide to the end-user to choose the answer the more possibilities to build a usable system we have.

SE5) Defining the group of question: The usability expert defines the groups according to the topic of the questions. Note that the end-user will be asked every question included in a group.

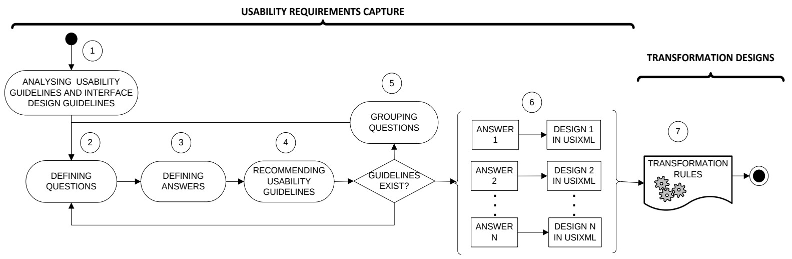


Figure 5. Process to build the tree structure to capture usability requirements

SE6) Obtaining interface designs: When the usability expert identifies that there are no more alternatives to specify a design, she/he can define this design formally. Each design (leaf) of the tree structure must be completely different to other designs, since the path used to reach the design will be exclusive. We propose defining these designs using the UsiXML [29] language. This definition must be performed by the analyst, since the usability expert does not work with conceptual models usually, and this topic is out of the scope of his / her expertise.

SE7) Transformation rules definitions: Once the designs have been defined, the analyst must specify transformation rules to transform these designs into primitives of the conceptual model of a MDD method. The transformations aim to include all the usability requirements in the software development process. Since we propose specifying the designs with UsiXML some of these transformations already exist [32].

B. *Phase of use.*

This phase explains how the analyst uses the tree structure to capture usability requirements. The process starts from the tree root to the leaves. When a question arises in the path, the analyst must ask the end-user the question. Apart from the question, the analyst must tell the end-user the possible answers to the question. If the answers are recommended by some usability guidelines, the analyst must specify which answers are recommended. Note that more than one answer can be recommended, and some usability guidelines can contradict each other.

This is not a problem, since the end-user must choose the answer that best fits the requirements, independent of the recommendations. When the end-user chooses an answer, the flow continues through the branches of that answer, while the branches of the other rejected answers will not be crossed. When a group of questions arises in the path, the analyst must ask the end-user every question in this group to based on the order they were created. Once the analyst asks the first question in the group, the flow continues with the branch of that question. When this branch has been completely gone through, the flow continues with the second question in the group. This process is repeated for every question in the group.

When a design arises in the path, the flow continues with the closest unresolved question. At the end of the process, we have a set of designs we have reached through the navigation. These designs are then transformed into primitives of a conceptual model of a MDD method according to the transformation rules previously defined. Note that rules are defined once, but they can be used indefinitely for the same tree structure and the same MDD method.

2.5 A Laboratory Demonstration

In order to illustrate the usability requirements capture process, we show an example to design a menu for a mobile phone application. Next, we exemplify our proposal for capturing usability requirements:

A. Phase of construction

SE1) Analizing the usability guidelines and interface design guidelines: As there are many interface design guidelines specific for mobile devices, our analysis focus only on Android[24], iOs [23], and Symbian [27] guidelines, since they provide specific descriptions to design menus and are widely used. With respect to usability guidelines, we used Nielsen's heuristics [33] since it is widely known and used by user interface designers to develop usable systems. From the interface design guidelines [24], we identified the most relevant aspects that should be considered in order to capture usability requirements. In our example, we focus on the "display mode" as a relevant aspect, since there are different ways to display menu options.

SE2) Defining the questions: We define the questions to ask concerning how to display the menu options in a system. According to interface design guidelines [24], we have identified the following questions: Q1. *How can the menu options be displayed?*; Q2. *What is the layout type to display nest views?*; Q3. *How is the contextual action item displayed?* Q1 has been extracted from Symbian [27] guidelines, which state that menu options are "an efficient way to allow users to perform actions". Therefore, the definition of the menu display is essential to allow users to trigger actions. Q2 has been extracted from the Android guideline [24], which proposes defining the menu hierarchy as simply as possible using a nest view. Q3 has been extracted from the Android guideline [24], which proposes contextual actions, such as actions that affect a specific item or context frame in the UI. This guideline describes different alternatives to display contextual action items. With these three questions, we began to define a part of a branch in our tree structure (Figure 6).

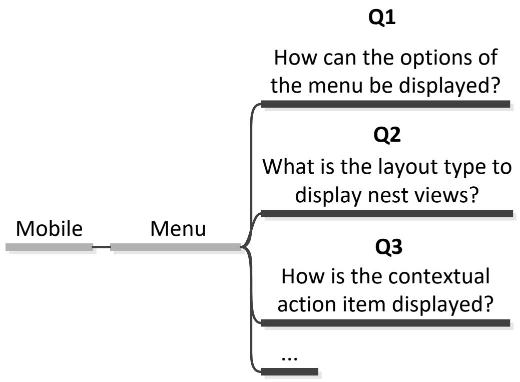
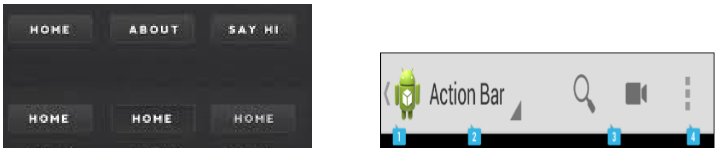

Q1
How can the options of the menu be displayed?
Q2
What is the layout type to display nest views?
Mobile
Menu
Q3
How is the contextual action item displayed?
...

Figure 6. Example of questions

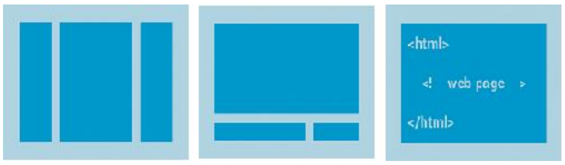
SE3) Defining the answers. For question Q1, we have identified the alternatives *"Button"* and *"Action Bar"*, since both options are the two possible ways to display the options of a menu. This classification is also used in the guidelines of Symbian [27], iOS [23] and Android [24]. Figure 7 shows an example of button and action bar. The different between them is that the button is based on option displayed by pressing the Buttons while the action bar is based on the combination of onscreen action items overflow options.
For question Q2, we have identified the alternatives *"Linear"*, *"Relative"*, and *"Web view"*, which appear in the Android guidelines. These answers gather all the possibilities to display a nest menu. These alternatives are also used in the design guidelines of Symbian and iOS. Figure 8 shows an example of *"linear"*, *"relative"* and *"Web view"*. All of them deal with the arrangement of view hierarchy. *"Linear"* arranges the view in a single column or in a single row. *"Relative"*, arranges the view in sections, and *"Web"* arranges the view as a web view.
For question Q3, we have identified the alternatives *"Floating contextual"* and *"Contextual action mode"*. These answers have been defined using the design Android guidelines [24],[23] Figure 9 shows an example of a floating contextual

menu and a contextual action mode. The difference between both types is that the Floating contextual displays actions using a flying list, while the Contextual action mode displays action item on the screen.



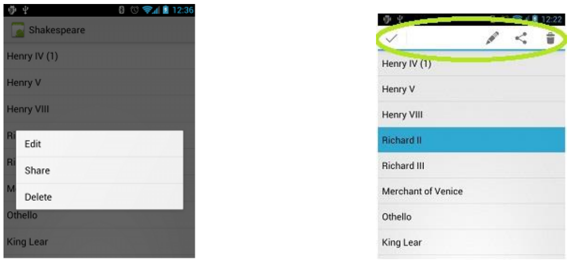
a) Button *b) Action Bar*

Figure 7. Alternatives Design for question Q1



a) Linear *b) Relative* *c) Web view*

Figure 8. Alternatives Design for question Q2



a) Floating Contextual Menu *b) Contextual Action Mode*

Figure 9. Alternatives Design for question Q3

Figure 10 shows how the tree is built using the questions and answers identified in our example. Next, we must continue following this procedure in order to define questions and answers until we do not have any more design alternatives defined by interface design guidelines.

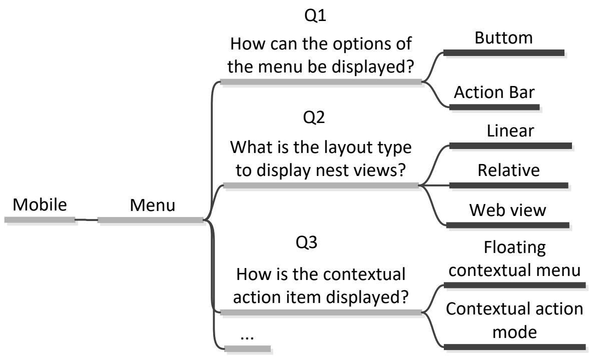

Q1
How can the options of the menu be displayed?
Buttom
Action Bar
Q2
What is the layout type to display nest views?
Linear
Relative
Web view
Mobile
Menu
Q3
How is the contextual action item displayed?
Floating contextual menu
Contextual action mode
...

Figure 10. Example of answers

SE4) Recommending usability guidelines: Following this process, once the answers have been defined, we must define which answers are recommended by usability guidelines.

As shown in Fig 10, for question Q1, two design alternatives (answers) are considered: *Button* and *Action bar*. Their respective recommendations are given with respect to the context of use (type of platform). For example, the alternative *Button* is recommended if we are developing an application for Symbian, Nokia, or Android (lower until version 2.3) platforms. This design alternative fulfils the usability feature which is stated in Nielsen heuristic [33], *"match between system and the real world"*, because the user activities should follow real-world conventions without essential changes. The alternative *Action bar* is recommended when the application is planned to be developed for Android (version 3.0 or higher) [24]. This design alternative fulfills the usability feature "flexibility and efficiency of use" according to Nielsen's heuristics [33]; since it offers flexibility for accessing actions.

For question Q2, three design alternatives are considered: *Linear*, *Relative* and *Web view*. The recommendations are given taking into account all platforms [24], [27], [23] and considering the tasks for which they are used. For example, the alternative *Linear* is recommended when the tasks consist of displaying content that has dynamic layout, or is not predetermined, or the menu structure is not too deep [24]. This design alternative fulfills the usability feature which is stated in

Nielsen heuristic [33*], “give people a logical path to follow”*, because the information should appear in a logical order. The alternative *Relative* is recommended when the task is to locate the main actions easily without high hierarchy. This design alternative fulfills the usability feature, *“minimize the user’s memory load by making the object, action and option visible”* specified by Nielsen’s heuristic [33] since the user does not need to remember information required for her/his activities. The Web view alternative is recommended when the task is to embed a web browser into the action. In this case, the design alternative fulfills the usability feature *“Any such information should be easy to search, focused on the user’s task”*, according to Nielsen’s heuristic [33], because frequency actions are tailored by users.

For question Q3, two design alternatives are considered, the Floating contextual menu and the *Contextual action mode*. These have been selected for use with Android and Symbian platforms, and tasks in which they are used. We recommend using the *Floating contextual menu* alternative when the task consists of displaying the contextual menu on views displayed by list view or grid view, where the user can perform direct actions on each item. This design alternative fulfills the usability feature *“The main tasks should be available quickly”* recommended by the Symbian usability guideline [27] since the actions frequently used should have priority in terms of visibility. The *Contextual action mode* alternative is recommended when the task is to perform an action on multiple items at once. This alternative fulfills the usability *“The help would assist the user in making full use of the functionalities”* according to Nielsen’s heuristic [33], since the user should be informed about what is going on.

The recommendation was continued for each alternative, but the usability guidelines are not always in concordance with the context, task and/ or user; so situations involving contradiction exist. For example, when the task consists of defining the hierarchy of the actions, a recommendation is that the application “Can suffer from poor usability and discoverability” if a drop down is used. This is a piece of advice contemplated in the Symbian platform. When the context is the Android platform, the drop down is called “linear layout”, and the advice is to use it when the task is to reduce the hierarchy of views on applications. Therefore, the recommendations have been made according to context, task or user.

SE5) Defining the group of questions: Questions: Q1, Q2, Q3, are grouped by “Menu”, since the end-user must be asked all of them in order to know the requirements with regard to the menu. We differentiate the group of questions in the tree structure with the character “*”, as Figure 11 shows.

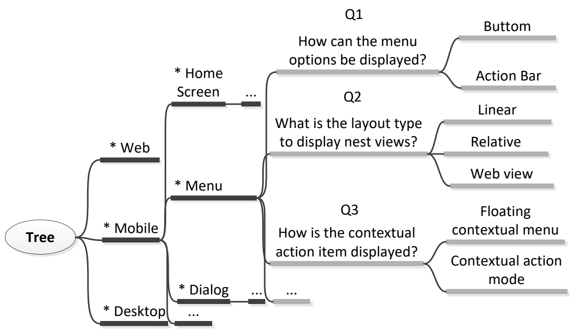
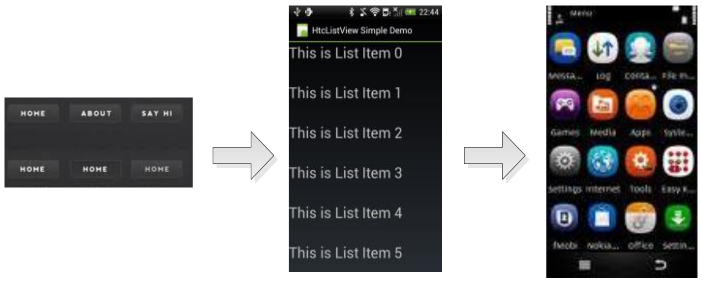


Figure 11. Example of groups of questions

SE6) Obtaining interface designs: At the end of our navigation we arrive at a set of designs depending on the user’s requirements. For example in Figure 18, we arrived at the leaf Grid following the sequence: Mobile → Menu → *How can the menu options be displayed?* →Button → *What type of menu is required?* → *View menu* → *What is the item display mode* → *Grid* → *Grid View*.

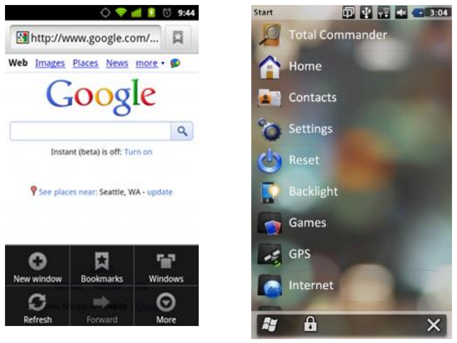
Figure 12 shows the differences between the designs of *Button*, *View Menu* and *Grid*. Depending on the end-user’s answers, the navigation process guides the analyst towards one of these designs.



Button *View Menu* *Grid*

Figure 12. Sequence of alternatives in order to obtain a design

As the same way, we could obtain other alternatives of design. Such designs are depicted in Figure 13. These are obtained following the same trajectory but selecting the alternative Six Button or List as answers for question *What is the item display mode?* (See *Q8* in Figure 18)



a) *Six menu button* b) *List view*

Figure 13. Some possible design alternatives

SE7) Transformation rules definitions: in this stage, we must define transformation rules to transform the designs into primitives of a MDD method. In order to facilitate this transformation, we recommend using UsiXML [29] to specify the designs, since there are existing rules to generate primitives for some MDD methods. The definition of these rules is beyond the scope of this paper, but existing rules can be used with our proposal. For example, there is a set of rules to transform UsiXML interface designs into conceptual models of a MDD method called OO-Method [34].

B. *Phase of use.*

Once we have defined the tree structure, we can use it to capture requirements. Figure 18 shows tree structure of our example completed with more questions and answers. The navigation process in the tree starts from the root to the leaves. Next, we describe a possible navigation process to capture the requirements for a mobile phone. Since we are developing for a mobile platform, we start selecting the alternative Mobile from the root. Inside *Mobile* there are other groups of questions (*Menu, Dialogue, among others*). The end-user must be asked the questions in all these groups of questions. We begin our navigation process with the first group, Menu (GQ1 → GQ2).
Once we begin the flow through the Menu, we follow the next sequence of branches:

- The Navigation process derived from Q1. A possible sequence could be: *Q1→ GQ3→ Q4→ GQ5→ Q8→ A3→ D1*. With this navigation process, we can arrive at the design D1-*Grid View* (See Figure 14). Once we arrive at a leaf, the navigation process continues with the closest unresolved question. In this example, we must continue with Q9, since it was in a group (GQ5) together with Q8. This navigation process brings us to D2 (*Drop Down Menu*) through *Q9→A5→D2* arriving at design D2. Figure 15 shows an example of this design. The flow continues with the other questions in GQ3.

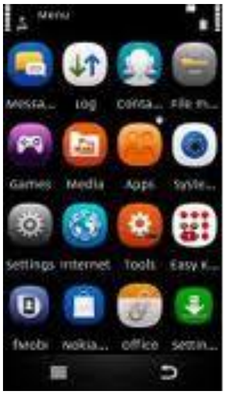


Figure 14. Design D1 - Grid view.

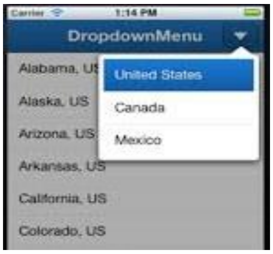


Figure 15. Design D2 – Drop Down menu

- Navigation process derived from Q2: A possible sequence could be: *Q2→A16→D3*. Since A16 was selected, we arrived at design alternative D3. Figure 16 shows a possible design for D3.

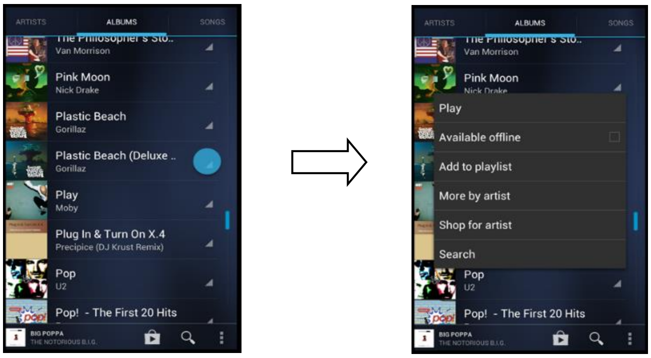


Figure 16 Design D3 - Linear Vertical with nest view.

- Navigation process derived from Q3: A possible sequence could be: *Q3→A19→D4*. This last selection addresses to the Floating Contextual Menu design, represented by D4 in Figure 18. A possible design for D4 is represented in Figure 17.

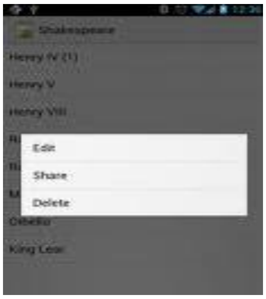


Figure 17. Design D4 - Floating Contextual Menu

At this point we have ended up with a design that is composed of D1, D2, D3 and D4. These designs will be gathered with the other designs arrived at through the whole navigation process. Finally, the designs arrived at can be transformed into conceptual primitives of an existing MDD method according to previously-defined transformation rules. Note that we have not exemplified this process since these transformations are beyond the scope of the current paper.

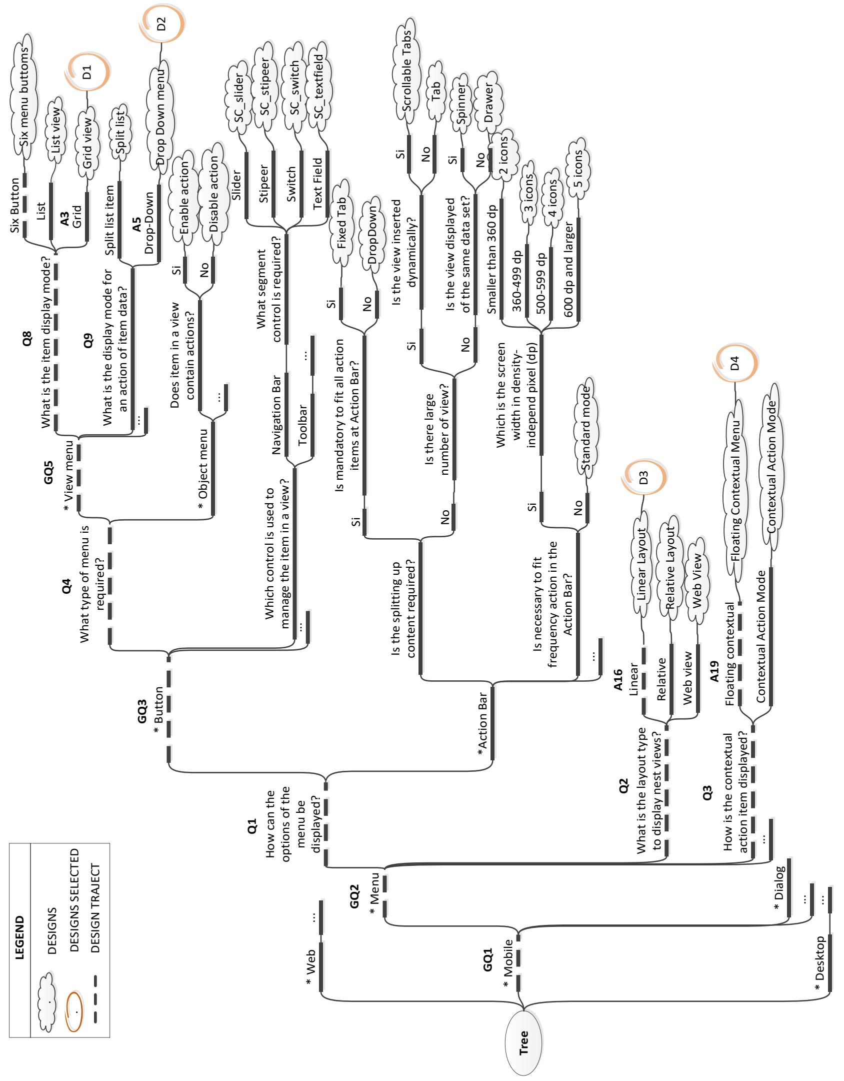


Figure 18. Usability Requirement Capture

2.6 Conclusion

This paper presents an approach to deal with usability requirements in MDD environments. The process consists of building a tree structure using interface design guidelines and usability guidelines that helps the analyst to capture usability requirements. The approach is based on a

question-answer format in such a way that requirements are captured with an interview with the end-user. The output of the interview is a set of designs that the system must satisfy. If we specify these designs formally, we can transform them into conceptual primitives of an existing MDD method.
As a language to specify the designs, we recommend UsiXML, since there are current works that have defined transformations between this language and existing MDD methods. However, our proposal is independent of the language to specify the designs. Note that the approach is also independent of the MDD method we used as the target of the transformations. However, if the chosen MDD method does not have conceptual primitives to express interaction features, we could hardly define transformations from the designs to the conceptual model, and few requirements could be included in the software development process. The tree structure and the transformation between the designs and the MDD method are defined once only, and they can be reused indefinitely to develop any system.
Note that the size of the tree structure will increase with the number of guidelines we consider. Even with few guidelines, the size of the tree is difficult to manage if we do not have a tool. As future work, we plan to develop a tool that helps with the definition of the tree structure and with navigation through the branches. In order to simplify the structure, we recommend focusing only on the more frequently used interface design and usability guidelines.
The main contribution of this work is the definition of the process to capture usability requirements, but there is still a lot of work needed to make this viable. The next step is to enrich the existing transformation rules from UsiXML to a MDD method in order to ensure that we can work with any design. Next, with a tool to support the process and the transformation rules, we plan to empirically evaluate the proposal. For this aim, we will compare a software development using our approach to capture usability requirements with a development which does not take these requirements into consideration.

References

1. S. J. Mellor, A. N. Clark, and T. Futagami, “Guest Editors' Introduction: Model-Driven Development,” IEEE Software, vol. 20, pp. 14-18, 2003.
2. S. Ceri, Fraternali, P., Bongio, A., "Web Modeling Language (WebML): a modeling language for designing Web sites." pp. 137 - 157.
3. N. Koch, A. Knapp, G. Zhang et al., “Uml-based web engineering,” Web Engineering: Modelling and Implementing Web Applications, pp. 157-191, 2008.
4. L. Bass, and B. John, “Linking usability to software architecture patterns through general scenarios,” The journal of systems and software, vol. 66, pp. 187-197, 2003.
5. E. Folmer, and J. Bosch, “Architecting for usability: A Survey,” Journal of Systems and Software, vol. 70, pp. 61-78, 2004.
6. J. Carroll, M., “Human-computer interaction: psychology as a science of design,” Int. J. Hum.-Comput. Stud., vol. 46, pp. 501-522, 1997.
7. B. Shneiderman, Plaisant, C., Diseño de Interfaces de Usuario. Cuarta Edicion. Estrategias para una Interacción Persona-Computadora Efectiva, Madrid: Addison Wesley, 2006.
8. J. Nielsen, Usability Engineering: Morgan Kaufmann, 1993.
9. E. Dynamic. "UI Styles Guides, " http://www.experiencedynamics.com/science-usability/ui-style-guides
10. G. Nielsen Norman. "Reports," http://www.nngroup.com/reports/.
11. S. Cronholm, "The usability of usability guidelines: a proposal for metaguidelines."
12. S. Henninger, “A methodology and tools for applying context-specific usability guidelines to interface design,” Interacting with Computers, vol. 12, pp. 225-243, 2000.
13. L. M. Cysneiros, V. M. Werneck, and A. Kushniruk, "Reusable knowledge for satisficing usability requirements." pp. 463-464.
14. N. Bevan, "Guidelines and standards for web usability." pp. 22-27.
15. Y. Pei, and G. Jiao, "The research of Web usability design." pp. 480- 483.
16. J. Nielsen. "Mobile Usability Update ": http://www.useit.com/alertbox/mobile-usability.html.
17. S. Schneider, F. Ricci, A. Venturini et al., “Usability Guidelines for WAP-based Travel Planning Tools,” Information and Communication Technologies in Tourism 2010, pp. 125-136.

18. T. Jokela, J. Koivumaa, J. Pirkola et al., “Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone,” Personal Ubiquitous Comput., vol. 10, pp. 345-355, 2006.
19. A. G. Sutcliffe, S. Kurniawan, and S. Jae-Eun, “A method and advisor tool for multimedia user interface design,” Int. J. Hum.-Comput. Stud., vol. 64, pp. 375-392, 2006.
20. J. I. Panach, S. España, A. Moreno et al., "Dealing with Usability in Model Transformation Technologies." pp. 498-511.
21. ISO-9241_11, “Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability,” 1998.
22. J. Tidwell, Designing Interfaces: O'Reilly Media, 2005.
23. iOS Human interface Guidelines Apple, 2012.
24. D. Android, “User Interface Guidelines,” 2012.
25. N. L. Biggs, "Discrete Mathematics," Oxford University Press.
26. R. Johnsonbaugh, Discrete Mathematics, Fourth Edition ed., New Jersey: Prentice Hall Intemational, 1997.
27. Nokia. "Symbian Design Guidelines - Dialogs," http://www.developer.nokia.com/Resources/Library/Symbian_Design_G uidelines/.
28. L. Cerejo, A. "User-Centered Approach To Web Design For Mobile Devices," http://mobile.smashingmagazine.com/2011/05/02/a-usercentered-approach-to-mobile-design/.
29. J. Vanderdonckt, Q. Limbourg, B. Michotte et al., "USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces."
30. M. Maguire, “Context of use within usability activities,” International Journal of Human-Computer Studies, vol. 55, pp. 453-483, 2001.
31. J. A. T. Hackos, and J. Redish, User and task analysis for interface design: Wiley New York, 1998.
32. J. I. Panach, Ó. Pastor, and N. Aquino, “A Model for Dealing with Usability in a Holistic MDD Method.”
33. J. Nielsen. "Ten Usability Heuristics"; http://www.useit.com/papers/heuristic/heuristic_list.html.
34. J. I. Panach, Ó. Pastor, and N. Aquino, "A Model for Dealing with Usability in a Holistic MDD Method," User Interface Description Language (UIDL), D. F. Adrien Coyette, Juan González-Caballeros, Jean Vanderdonckt. (ed.), ed., pp. 68-77, Lisbon (Portugal), 2011

2.3 A Proposal to Elicit Usability Requirements within a Model-Driven Development Environment

Nowadays there are sound Model-Driven Development (MDD) methods that deal with functional requirements, but in general, usability is not considered from the early stages of the development. Analysts that work with MDD implement usability features manually once the code has been generated. This manual implementation contradicts the MDD paradigm and it may involve much rework. This paper proposes a method to elicit usability requirements at early stages of the software development process such a way non-experts at usability can use it. The approach consists of organizing several interface design guidelines and usability guidelines in a tree structure. These guidelines are shown to the analyst through questions that she/he must ask to the end-user. Answers to these questions mark the path throughout the tree structure. At the end of the process, we gather all the answers of the end-user to obtain the set of usability requirements. If we represent usability requirements according to the conceptual models that compose the framework of a MDD method, these requirements can be the input for next steps of the software development process. The approach is validated with a laboratory demonstration.

Versión del autor del artículo: Ormeño, Y. I., Panach, J. I., Condori-Fernández, N., & Pastor, Ó. (2014). A proposal to elicit usability requirements within a model-driven development environment. International Journal of Information System Modeling and Design (IJISMD), 5(4), 1-21, http://dx.doi.org/10.4018/ijismd.2014100101

3.1 Introduction

Model-Driven Development (MDD) paradigm (Embley, Liddle, & Pastor, 2011) states that the analysts' entire effort should be focused on a conceptual model, and the system should be implemented by means of model to code transformations performed by a model compiler. A software production process is then seen as a set of conceptual models that are adequately transformed from requirements to code. A plethora of MDD methods and tools have been proposed, such as WebML (Ceri, Fraternali, & Bongio, 2000) or UWE (Koch, Knapp, Zhang, & Baumeister, 2008) among others. There are two main dimensions to consider in MDD (Frankel, 2002): a "vertical" dimension and a "horizontal" dimension. In the vertical dimension there are at least three main layers that must be present in any MDD process:

1. A Requirements Modeling step, to produce a Requirements Model.
2. A Conceptual Model representation, where requirements are represented from the computer-oriented perspective.
3. The final Software Product (the Code).

The horizontal dimension focuses on the different expressiveness that must be present in the different conceptual perspectives of a MDD software process. Summarizing, these perspectives are:

- The data (static, system structure-oriented) perspective.
- The functional (dynamic, system behavior-oriented) perspective.
- The interaction (user interface-oriented) perspective.

While it can be argued that the two first perspectives (data and functionality) are largely explored by the different MDD approaches, it is surprising to realize that the interaction perspective is not at all so intensively explored. One could conclude that a Software Product is just the sum of a conceptual model where data and behavior are precisely specified, what is not exactly true, because the specification of the system interaction is an essential component of any software product. To confirm this situation, it is enough to consider the current modeling approaches that we find in practice. From the Data perspective, the question of what data models can be used to represent data has an immediate answer: ER and UML Class Diagrams are clearly among the most widely used and known. From the Functional perspective, since the appearance of the Data Flow Diagrams till the most modern UML diagrams designed to represent functionality, the offer is large. However, if the question is what models are specially used to represent System Interaction, the answer is not at all so immediate. Extending a previous version presented at (Y. I. Ormeño, Panach, Condori-Fernandez, & Pastor, 2013), the goal of this paper is to explore the need of an interaction modeling, focusing on an essential software quality criteria that is mainly in the interaction scope: usability. Nowadays, in MDD, usability features are manually implemented once the code has been generated. According to Bass (Bass & John, 2003) and Folmer (Folmer & Bosch, 2004), these manual changes may involve changes in the system architecture, which can result in a lot of extra effort. Moreover, these manual implementations can produce a source code that contradicts the system's characteristics expressed in the conceptual model. In the previous work (Y. I. Ormeño et al., 2013) we defined how to elicit usability requirements according to existent usability guidelines. In this paper, we define how to include the usability requirements elicitation process in a MDD method. The main final goal of the paper is to define an approach to facilitate the usability requirements capture process for analysts who are not experts in usability engineering, and that want to include also the specification of usability requirements in a MDD-based approach. The proposal to elicit usability requirements is based on the idea that first, an expert in usability defines a tree structure where design alternatives and usability guidelines are represented textually with questions and answers. Next, the analyst (non-expert in usability) can use this tree structure indefinitely to ask end-users which alternative is the most suitable according to their requirements. Usability guidelines can help the end-user select an alternative throughout the tree structure. At the end of the process, we have a design for our system based on the end-user's requirements. If we represent the designs according to an existing conceptual model of a MDD method, those designs are the input for next development steps in the MDD process. The approach is validated with a laboratory demonstration with the participation of 4 subjects. This paper is divided into the following sections: Section 2 presents the state of art of various approaches concerning both the modeling of interaction and the use of usability guidelines; Section 3 provides a general view of the approach to elicit usability requirements; Section 4 describes how to build the tree structure to represent all the design alternatives in an existent MDD method; Section 5 shows how to use the approach once the tree structure has been built; Section 6 reports an initial empirical validation of our approach. Finally, Section 7 describes the conclusions and future work.

3.2 Related Work

The literature presents a lot of usability guidelines to support the design of user interfaces, but they may confuse the analyst if she/he is not an expert in usability. In general, the analyst may face the following problems (among others): it is not easy to understand how to apply the guideline; sometimes it is difficult to determine when a guideline has been broken; and some guidelines are so ambiguous that they are difficult to apply to specific contexts. All these aspects require a huge effort on the part of the analyst that leads us to determine if the usability guidelines are still usable. Cronholm's work (Cronholm, 2009) and Henninger's work (Henninger, 2000) describe possible solutions to some of these problems. Cronholm's work proposes meta guidelines as a solution to obtain more systematic and categorized guidelines. Design guidelines defined by Henninger include two types of guidelines: interface principles, or typed rules, and usability examples, also known as cases. These cases are examples of specific interfaces developed for organizations that contain a lot of knowledge about the needs and common practices of clients' work. Cysneiros's work (Cysneiros, Werneck, & Kushniruk, 2005) proposes a reusable catalogue to capture usability requirements. The method is based on i* framework and it uses personal experiences to obtain knowledge to achieve the objectives of usability. The cited works aim to mellow the ambiguity of the usability guidelines, but they increase the complexity of use for non-experts in usability. All these solutions involve a lot of effort to understand all the guidelines and to choose the most suitable one for a specific context. For example, understanding the notation or the information arrangement in a guideline may involve some of the analyst's effort in order to use the guideline optimally. Furthermore, the comparison of guidelines shows great variability, which leads to creating specific usability guidelines for specific domains. Some authors aim to reduce developer's effort, such as Ferre (Ferre, Juristo, & Moreno, 2005), who defined a framework for usability practices integration. HCI techniques are characterized according to relevant criteria from a Software Engineering (SE) perspective and integrated into a framework organized according to development activities. Examples of methods to capture usability requirements are: a method for quantitative usability requirements applied in user interfaces to depict the true usability (Jokela, Koivumaa, Pirkola, Salminen, & Kantola, 2006); multimedia user interface designs that design attractive and usable multimedia systems (Sutcliffe, Kurniawan, & Jae-Eun, 2006); and, embedded Functionality Usability Features in model transformation technologies (Panach, España, Moreno, & Pastor, 2008). We can state that there are many proposals but none of them clearly and concisely addresses how to perform the usability requirements capture in early stages. If we focus on approaches to elicit usability requirements according to the MDD paradigm, we realize that there are not previous works; in spite of MDD methods have usually a model to represent the interaction with the end-user. For example, WebRatio (Acerbis, Bongio, Brambilla, & Butti, 2007) includes a Presentation Model to express the layout and graphic appearance of pages, independently of the output device and of the rendition language. UWE (Koch et al., 2008) enables the definition of the front-end interface by means of a Hypertext Model. NDT (Escalona & Arag, 2008) has an abstract interface based on a set of prototypes to represent the interaction with the user. OO-Method (Pastor, 2007) has two models to represent the interaction: the Abstract Interaction Model (independently of platform) and the Concrete Interaction Model (platform-specific). All these MDD methods have some proposals to capture functional requirements but all of them lack of a process to capture usability requirements. This might result in unsatisfied end-users, which involves changes in conceptual models and in the generated code to solve problems related to interaction. This rework involves a lot of effort if analysts are not experts in usability. An early usability requirements elicitation guided by means of usability guidelines aims to prevent these problems from the first steps of the software development process. This paper defines a process to organize the information stored in different usability guidelines based on a user-centred development (Hassenzahl, 2008). This way, analysts without a background in usability can work with the guidelines. Based on a review of the literature (Yeshica Isela Ormeño & Panach, 2013), we can say that very few papers that address how to perform the extraction process of usability requirements have been written (Henninger, 2000), (Cysneiros et al., 2005). Generally, this task is done when the usability requirement capture has finished. Moreover, usability requirement capture has not been developed focusing on the MDD method. This paper aims to cover this gap, proposing a process to capture usability requirements such a way they can be transformed later into part of the conceptual model of the MDD method.

3.3 A Proposal to Elicit Usability Requirements

Based on ISO 9241-11 (ISO-9241_11, 1998) standard, usability requirements are requirements that affect effectiveness, efficiency and satisfaction of a user achieving his/her goals in a defined context of use. Our approach is based on existing usability guidelines and design guidelines, that are stored in a tree structure. The analyst navigates through this structure in order to capture the usability requirements by asking questions to end-users. The tree structure helps the analyst to identify the different design alternatives, and how these decisions will affect the system's usability. Figure 1 shows the elements used in our approach. Next, we describe each element:

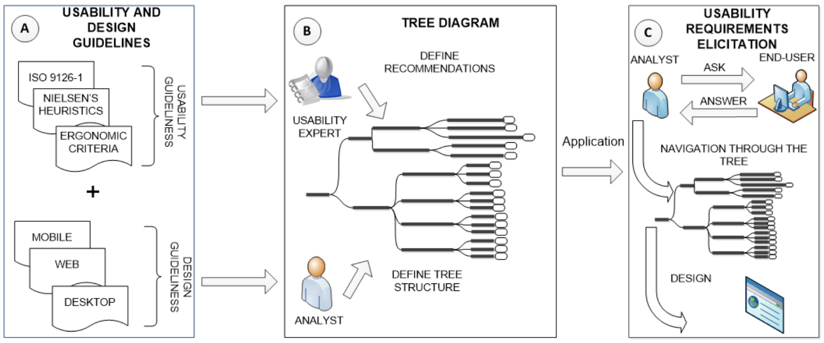


Figure 1. Schema of the proposal to capture usability requirements.

3.3.1 Usability guidelines and interface design guidelines

Both usability guidelines and interface design guidelines have been created to guide the analyst to develop systems (Figure 1a). Usability guidelines recommend how to combine users, tasks and context to enhance the system usability. Interface design guidelines provide alternatives for designing systems. These guidelines have been built for different technologies and platforms that are represented by standards, principles, heuristics, styles, patterns, best practices, etc. Design and usability guidelines are related to each other since some design guidelines can improve or decrease the usability (depending on the combination of tasks, users and context). Working directly with both kinds of guidelines implies a huge effort as the variability and amplitude of these guidelines is very high. In order to reduce this effort, we propose storing all the relevant guidelines information in a tree structure, which is explained in more detail below.

3. 3.2 Tree diagram

We propose using design and usability guidelines through a tree structure in order to minimize the cognitive effort to work with them (Figure 1b). A tree structure is defined as a connected graph with no cycles and a root (Johnsonbaugh, 1997). Figure 2 shows a general schema of the tree structure used in our approach, which is composed of four elements: question, answer, group of questions, and design. Next, we present these elements:

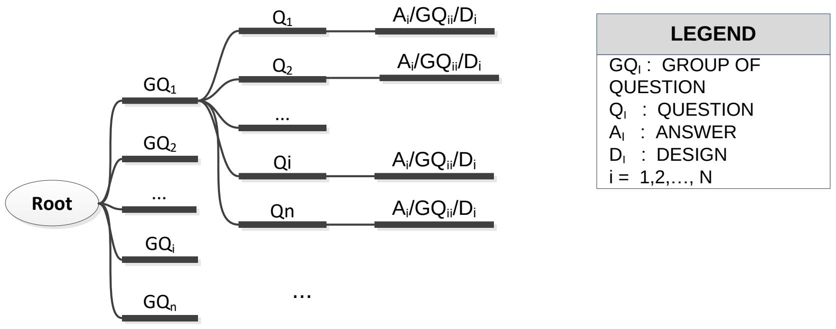


Figure 2. General representation of the tree structure (adapted from (Y. I. Ormeño et al., 2013))

1. Question (Qi): The design guidelines present diverse design alternatives for many UI (User Interface) components (e.g. menu). In order to ask the end-user which alternative she/he prefers, we have defined a question when alternatives to design appear. For example, when we are designing dialog elements for mobile, design guidelines (Nokia), (Android, 2012) specify that dialog elements provide a top-level window for short-term tasks and a brief interaction with the user. We can define a question to decide which is the UI component to represent a selectable, *"Which UI component is used to show selectable tasks?"*. In Figure 2, questions are represented by Qi.
2. Answer (Ai): These are the exclusive options for each question according to interface design guidelines. These options are presented to the analyst in such a way that she/he can choose which one best fits user's requirement. The analyst's decision is not only based on end-user criteria, but also on usability guidelines. This means that we must relate answers to usability guidelines depending on the type of user, task, and context. When answers are shown to the analyst, we will show which answers are recommended by usability guidelines. For example, the answers to the question *"Which UI component is used to show selectable tasks?*" can be: RadioButtons, TextBoxes, CheckBoxes or Slider (Android, 2012), (Nokia). According to usability guidelines, a RadioButton is constructed for a persistent single-choice list (Android, 2012), where aspects such as "simplify navigation" and

"minimize user input" are usability requirements (Cerejo, 2011). In Figure 2, answers are represented as A_i, A_{i+1}, …, A_n.

3. Group of Question (GQi): Some branches of the tree structure are not mutually exclusive (the end-user should be asked all of the questions). This type of branch is represented by a group of questions, which gathers several questions grouped by a design characteristic. For example, the question "Which UI component is used to show selectable tasks?" can be gathered with other questions that ask about Selection Dialogues, such as "Where is the action button located?", *"Where is the dialogue box located?"*, and *"Where is the positive action on button located?"*. All these questions have in common that deal with how selection dialogs are displayed, and all of them are gathered in the same Group of questions. In the tree structure these are represented as GQi, in Figure 2.
4. Designs (Di): These are the interface designs reached through the alternatives that the analyst has been choosing. The analyst navigates through the tree structure asking the questions to the end-user, who selects the most suitable answer (usability guidelines can recommend some answers). When the analyst reaches a leaf in the tree, a design has been obtained. The final design of the whole system is the set of leaves in the tree that the analyst has reached. For example, a design can be a selection dialog with radio buttons, where each item shows an enumerated data (Nokia), (Android, 2012). At the tree structure these are represented as Di, in Figure 2.

The tree structure must be built by an analyst in collaboration with an expert in usability, who knows how to interpret and use usability guidelines. The expert in usability is responsible for defining the recommendations for each answer. In order to identify all the elements that compose the tree structure, we have defined a meta-model (Figure 3). The meta-model allows the replication of the tree structure in any context and the instantiation of as much instances as we need. Each instance can be used for different design and usability guidelines, resulting in different combinations of questions and answers.

Next, we describe the elements of the metamodel (classes). Class Design Guideline represents the interface design guidelines used in our tree structure. Questions that the end-user will be asked in order to discover which design alternative is most suitable are derived from these guidelines. Every question can be related to a group of questions, or to at least two Answers, since there is always more than one choice for each question. The class Group of Questions represents the set of questions we can define, and the class Answer specifies the exclusive alternatives for the question. Some of these answers can be recommended by one or several usability guidelines, recommendations, standards and best practices, represented as instances of the class Usability Guideline. According to the usability definition described in ISO-9241 (ISO- 9241_11, 1998), some usability guidelines are specific for a context, task or user. This is represented through the classes Context, Task, and User respectively. Finally, class Design represents the designs that the analyst can get to at the leaves of the tree. Each instance of this class is a different interface design that we can reach through different answers.

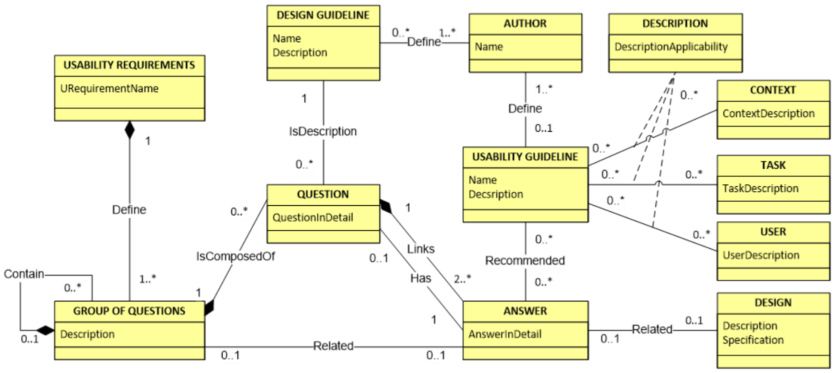


Figure 3. Meta-model of usability requirements capture (adapted from (Y. I. Ormeño et al., 2013))

3.3.3 Usability requirement elicitation

Once the tree structure has been finished, any analyst without explicit knowledge of usability can use it (Figure 1c). The usability requirement elicitation is the process to capture usability requirements using our approach. The navigation starts from the root of the tree while the analyst asks the questions to the end-users. The analyst asks the questions according to their sequence in the tree, from the root to the leaves. Questions are mutually exclusive, in other words, the analyst only navigates through the branch of the answer selected by the end-user. Questions that are gathered in the same group of questions are all asked. When the analyst reaches a branch with a group of questions, the flow continues with the first question in the group. Only when this flow has finished, the analyst can continue with the next question in the group. The possible navigation between two nodes of the tree structure can be: i) From a group of questions to a question, or to another group of questions (GQi→Qi / GQi); ii) From a question to an answer (Qi→Ai); iii) From an answer to a question, to a group of questions or to a design (Ai→Qi / GQi/ Di). Note that if we work with several usability guidelines, they can contradict each other when they recommend an answer. For example, a widget with a ListBox (list of items) is recommended to improve Information Density (amount of information in the interface), since items are hidden inside the list. However, a RadioButton (◎) is recommended to improve Brevity (users' cognitive workload), since the items are displayed directly without the necessity of opening any list. This contradiction is not a problem in our approach, since usability guidelines are only recommendations. In case of contradiction, the analyst must tell the end-user which alternative is proposed by each usability guideline. The choice of the most suitable answer only depends on the user, who must choose according to his preferences. The analyst must explain to the user which usability recommendation satisfies each design alternative.

3.3.4 Including the Approach in a MDD Method

The link between the tree structure and a MDD method is performed through the leaves of the tree (the designs). Our approach consists in specifying the possible designs of the tree structure through a conceptual model of any existing MDD method. Most MDD methods have a specific model to represent end-user interaction (interaction model), that together with other models to represent persistency and behaviour are the input for the code generation process. We propose using those interaction models to represent all the design possibilities expressed in the tree structure. Note that our proposal does not deal with how to work with interaction models or how to transform these interaction models into code. That depends exclusively on the MDD tool used as instantiation of our proposal. We focus on how to elicit usability requirements and how to include them in any of the existing MDD methods without modifying its existing conceptual model.

From all the MDD methods with an interaction model, we focus our illustrative example on OOMethod (Pastor, 2007). This choice is based on two characteristics: (1) OO-Method has an industrial tool named INTEGRANOVA (CARE, 2014) with a model compiler that can generate fully functional systems from a set of conceptual models without writing a single line of code. The generation is performed with ad-hoc transformation rules from models to code. All the models of the OO-Method framework are stored in a XML file that is the input for the code generation process. The XML file is read with a parser implemented in C++ that generates the code in C# or Java. (2) OO-Method has a model expressive enough to represent several design alternatives.

Next, we summarize both models of OO-Method to represent interaction: the **Abstract Interaction Model** (Molina, Meliá, & Pastor, 2002) and the Concrete Interaction Model (Aquino, 2008). The Abstract Interaction Model focuses on representing which are the elements that will be displayed for each interface. From a MDA perspective, this model is PIM since interfaces represented with this model are valid for any platform. These are the possible elements (named interaction patterns):

- Introduction: captures the relevant aspects of data to be entered by the end-user (including masks).
- Defined selection: enables the definition (by enumeration) of a set of valid values for an associated model element.
- Argument grouping: defines which input arguments can be grouped.

- Filter: defines a condition to display a list of elements.
- Order criterion: defines how a list can be ordered.
- Display set: determines the elements that compose a list with several fields.
- Actions: defines the set of available services.
- Navigations: determines the information set that can be accessed through a navigation between two interfaces.

The **Concrete Interaction Model** specifies how the elements that compose the interface will be displayed. From a MDA perspective, this model is PSM since interfaces represented with this model are for a specific platform. For example, in this model, the analyst decides the widget to display a Defined Selection (a list of enumerated values), which can be a ListBox or with a Radiobutton. The Concrete Interaction Model is defined through Transformation Templates, which specify the structure, layout and style of an interface according to preferences and requirements of end-users, and the different hardware and software computing platforms. A Transformation Template is composed of Parameters with associated values which parameterize the different design alternatives of the interfaces (Aquino, 2008). Apart from interaction models, OO-Method is composed of an Object Model (which specifies the system structure in terms of classes of objects and their relations), a Functional Model (which specifies how events change object states) and a Dynamic Model (which represents the valid sequence of events for an object). A detailed description of all these models can be found in (Pastor, 2007).

Next, we apply the three elements of our approach (Figure 1) to OO-Method: (1) Usability and Design Guidelines; (2) Tree Diagram and (3) Usability Requirements Elicitation. This section deals with the two first elements, relegating the Usability Requirements Elicitation to next section. For **the first element** (Figure 1a) we use the design alternatives of the Abstract Interaction Model and the Concrete Interaction Model of OO-Method. As usability guidelines, we use ISO 9126-3 (ISO-9126, 2001) and the ergonomic criteria of Bastien and Scapin (Bastien, 1993).

Both guidelines have been widely used in the software engineering community and in the human-computer interaction community.

The **second element** of our approach (Figure 1b) is the tree structure definition using design and usability guidelines previously chosen. From a MDA perspective, the tree structure is CIM, since it is independent of computation. According to (Y. I. Ormeño et al., 2013), the steps to build a tree structure are the following:

1. Identify design alternatives and define questions to ask the end-user which is the best design.
2. Express each design alternative as a possible answer for the questions defined previously.
3. Gather non-excluding design alternatives in groups of questions.
4. Define specific designs in the leaves of the tree.

After applying all these steps to OO-Method, we have the tree structure displayed in Figure 4 and Figure 5. Each design is identified with the letter "D" and a number. Apart from identifying design alternatives, we have also identified the recommendations for the answers according to the metrics of ISO 9126-3 (ISO-9126, 2001) and the ergonomic criteria (Bastien, 1993). Next, we describe in detail the design alternatives identified in the Abstract Interaction Model of OOMethod and which ones are recommended according to usability guidelines. The tree structure has been performed by an analyst of OO-Method and an expert in usability. Each design alternative is represented in Figure 4 and Figure 5 as an answer:

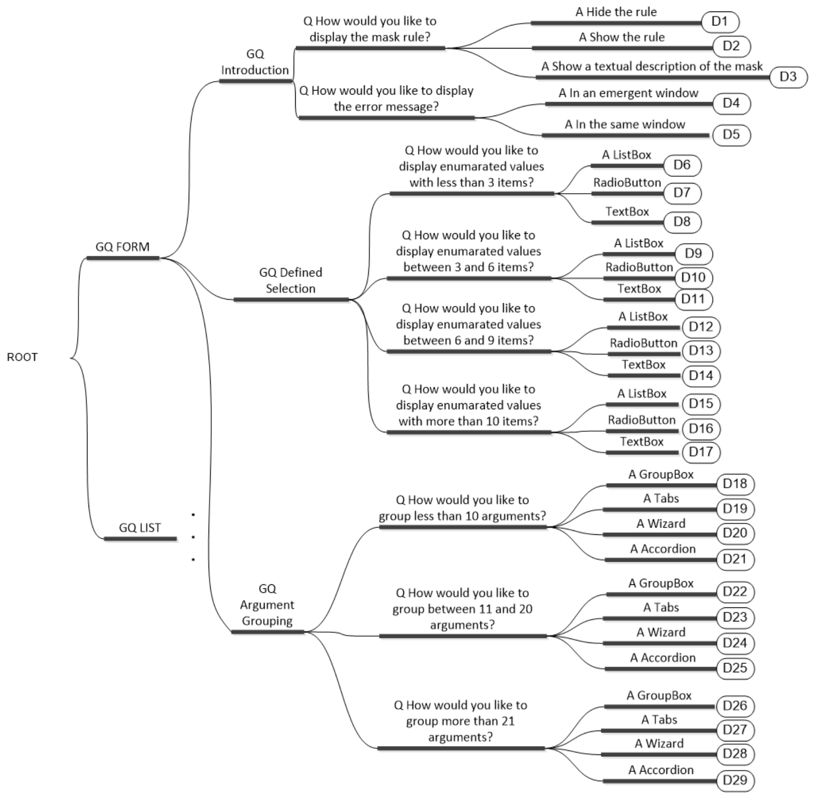

ROOT
GQ FORM
GQ Introduction
Q How would you like to display the mask rule?
A Hide the rule
D1
A Show the rule
D2
A Show a textual description of the mask
D3
Q How would you like to display the error message?
A In an emergent window
D4
A In the same window
D5
GQ Defined Selection
Q How would you like to display enumarated values with less than 3 items?
A ListBox
D6
RadioButton
D7
TextBox
D8
Q How would you like to display enumarated values between 3 and 6 items?
A ListBox
D9
RadioButton
D10
TextBox
D11
Q How would you like to display enumarated values between 6 and 9 items?
A ListBox
D12
RadioButton
D13
TextBox
D14
Q How would you like to display enumarated values with more than 10 items?
A ListBox
D15
RadioButton
D16
TextBox
D17
GQ Argument Grouping
Q How would you like to group less than 10 arguments?
A GroupBox
D18
A Tabs
D19
A Wizard
D20
A Accordion
D21
Q How would you like to group between 11 and 20 arguments?
A GroupBox
D22
A Tabs
D23
A Wizard
D24
A Accordion
D25
Q How would you like to group more than 21 arguments?
A GroupBox
D26
A Tabs
D27
A Wizard
D28
A Accordion
D29
GQ LIST

Figure 4. Tree structure with alternatives of OO-Method (1)

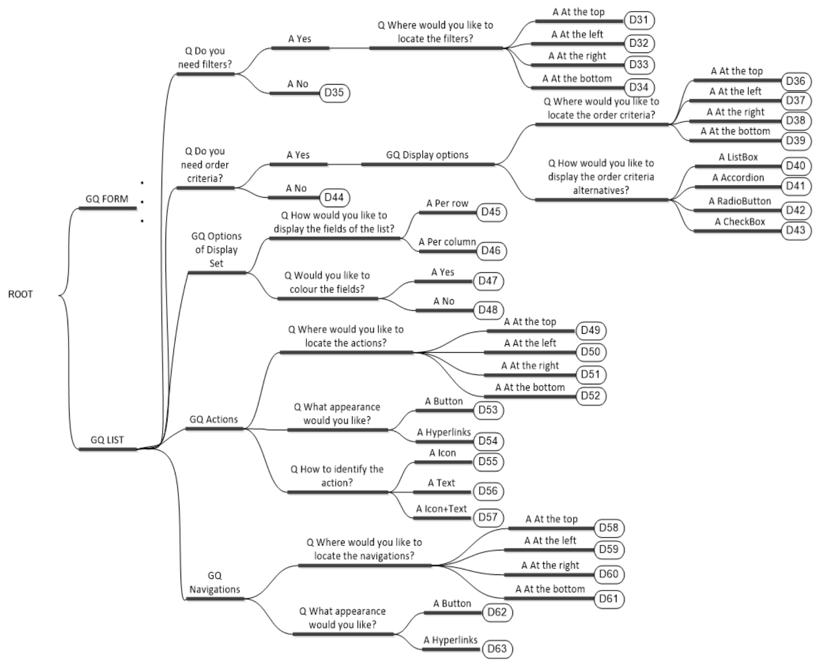


Figure 5. Tree structure with alternatives of OO-Method (2)

- Introduction: the system can show the rule of a mask to prevent end-user from errors or hide it. Moreover, the error message displayed when inserted data does not fulfill the mask rule can be shown in a new emergent window or in the same window of the form. According to the ergonomic criterion Information Density, rules should not be shown, since they can overload the amount of information. However, criterion Error Protection (prevention of data entry errors) and metric Message Clarity (proportion of self-explanatory messages) recommend showing the rules with a textual description to be understandable. Moreover, criterion Minimal Actions (workload regarding the number of actions) recommends showing the error message in the same window; while metric Interface Element Clarity (proportion of self-explanatory interface elements) recommends using a new emergent interface to show the error message.
- Defined Selection: the possible values can be inserted with a ListBox, a RadioButton or a TextBox (free text). According to

criterion Minimal Action, enumerated values with less than 9 items should be displayed with RadioButtons, since all the possible values are shown directly (Panach, Condori-Fernández, Vos, Aquino, & Valverde, 2011). However, according to criterion Information Density, items should be displayed with a ListBox, such a way, the list of possible values is hidden until the end-user opens the list. Enumerated values with more than 9 items should be displayed with a ListBox according to the criteria Information Density and Legibility (lexical characteristics of information that facilitate the reading). In this case, a design with RadioButtons could increase the amount of information in the interface and a design with TextBoxes could not guide the user.

- Argument Grouping: arguments of a form can be grouped by a GroupBox (a group of elements in the same window), Accordion (a group of elements that can be hidden), Tabs (division of a form into different windows without relationship among them) or split into several interfaces through a Wizard (division of a form into different windows with a relationship among them). According to metric Functional Understandability (assessment that new users can understand the system) and criterion Guidance (availability of advising), a Wizard should be used when there are many arguments to perform an action. When there are not so many arguments, criterion Information Density recommends dividing the argument using Tabs or Accordion, since the end-users can show the arguments depending on their needs. When there are a few arguments, the design with a GroupBox is recommended by criterion Minimal Actions, since the arguments do not take up much space and they are shown directly.
- Filter: the first decision is to choose whether or not the system needs filters. Next, we must decide where displaying them. According to criterion Information Density, the use of a filter makes sense when there is a huge amount of information, and the end-user needs some mechanisms to reduce it. However, when the amount of information is little, criterion Minimal Actions recommends not using a filter, such a way, end-users

can list all the information directly. With regard to the position of the filter in the interface, top and left positions will consider the filter more important than the right and bottom positions. This recommendation provides from criterion Compatibility (match between users' characteristics and dialogues), that propose developing the system regarding end-users' perceptions and customs.

- Order Criterion: this pattern shares the same design alternatives as the filter, adding the possibility to choose how to display the different order criteria. According to criterion Legibility and metric Help Facility (proportion of functions described in the user documentation), order criteria should be used when there is much information in interfaces. This mechanism will help end-users identify quickly the required data. However, when the amount of information is little, criterion Minimal Actions than the benefit obtained with the order. With regard to the position of the order criteria, we can apply the same criterion used for Filter (Compatibility). How to display the order criteria alternatives will depend on the size of the screen. For wide screens, criterion Minimal Actions recommends displaying the order criteria with a RadioButton or a CheckBox. However, for narrow screens, criterion Information Density recommends hiding the order criterion until the end-user needs them. In this case, a design with a ListBox or Acordion is the most suitable.
- Display Set: the fields of the list can be displayed per rows or per columns. Moreover, we can colour the fields if we think that this will help to understand displayed data. According to criterion Compatibility, the fields of the Display Set should be compliant with the size of the screen in order to avoid scroll bars. Therefore, wide screens can show the different fields per column and narrow screens should show the fields per row. Moreover, criterion Legibility and metric Help Facility recommend using different colours per field to help end-users understand the information.

- Actions: there are different locations to display the actions; different widgets, such as buttons or hyperlinks; and different representations, such as icons, labels or a combination of icons and labels. According to criterion Compatibility, the recommendation for the position is the same as the recommendation for Filters. With regard to how to display the action in the screen, criterion Compatibility recommends using the widget most commonly used. Therefore, an appearance as Hyperlink is more suitable for Web applications and mobile systems, while an appearance as button is more suitable for desktop systems. Moreover, criterion Prompting (guide to make specific actions) and metric Function Understandability recommend identifying the actions such a way every user can recognize the action. Therefore, a textual label or an icon with a label is more suitable than only an icon. However, systems with a small screen should use icons according to criterion Information Density, since an icon will always take up less space than a textual description.
- Navigations: they share the same alternatives as actions. According to criterion Compatibility, the recommendation for the position is the same as the recommendation for Filters. Moreover, the recommendation for the appearance is the same as the recommendation for Actions according to criterion Compatibility.

The fourth step of our process consists in specifying the designs of the leaves through a conceptual model of the MDD method (Figure 6). This specification is the link between our proposal to elicit usability requirements and an existing MDD method. Each design of the tree structure can be represented in a conceptual model of the MDD method. Note that the process to specify the designs is done once only, when the tree structure is specified. How each design is specified depends exclusively on the used MDD method. As illustrative example, we describe how to specify the design to show a mask rule (D2 in Figure 4) and the design to display its error message in an emergent window (D4 in Figure 4). D2 and D4 must be specified both in Abstract and Concrete Models of OO-Method.

This notation is just an example for the instantiation of our proposal to OO-Method:

- D2 is represented in the Abstract Model through the interaction pattern Mask, which is specified through the XML code:
  <PIntroductionM id=”Mask_XX”>
  <MsgError> “XXXX” </MsgError>
  <PIntroduccionStringM Mask=” XXXX” />
  </PIntroduccionStringM>
  </PIntroduccionM>
  D2 is represented in the Concrete Model through the template:
  .MaskError=Mask_XX.MsgError

- D4 is represented with the same Abstract Model as D2, since both designs share the samein teraction pattern: Mask.
  D4 is represented in the Concrete Model through the next template:
  .DisplayErrorMask= NewWindow

Note that models used to define the designs in the requirements elicitation step are initial interaction models composed of a first draft of Abstract and Concrete Models. By initial, we mean a model where specific details of the interface are not yet represented, just usability requirements. That is the reason why the previous examples of Abstract and Concrete Models do not specify an error message. In next development steps, the analyst must complete the interaction model and together with other models that represent persistency and behaviour, they are the input for the model compiler. Finally, the model compiler interprets the characteristics expressed in the interaction models and generates the code that implements those characteristics.
A detailed description about how to model interfaces with the Abstract Interaction Model (Molina et al., 2002), the Concrete Interaction Model (Aquino, 2008) and model to code transformations are out of scope of this paper since they do not concern the requirements elicitation step. Our contribution in this paper is only the process to elicit usability requirements (in grey background in Figure 6).

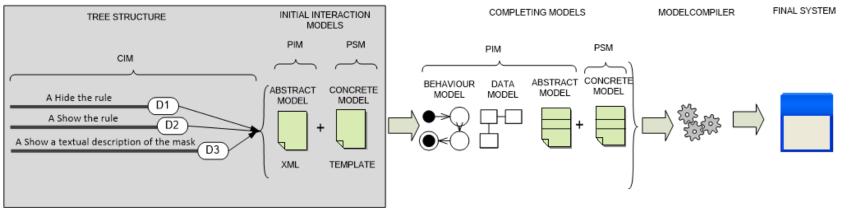


Figure 6. Overview of the process to include usability requirements in an MDD method.

3.4 The Tree Structure in Use

This section describes the third element of our approach (Figure 1c): how to use the tree structure once it has been defined completely. As example, we use a system for car rental that must save information of all the cars that the car rental company has around the world; therefore, the system needs to store much information. The system will follow a client-server architecture, such a way, the same server can connect with several clients in different platforms. In our example, we need to develop for two platforms: Web and mobile. The need of two platforms results in the development of two types of interfaces, in spite of the business logic is the same in both of them. In order to elicit the usability requirements for both systems, we must navigate two times through the tree structure of our approach.

First, we focus the example on eliciting usability requirements for the Web application. The process starts from the tree root to the leaves. When a question arises in the path, the analyst must ask the end-user that question. Apart from the question, the analyst must tell the end-user the possible answers to the question. If the answers are recommended by some usability guidelines, the analyst must specify which answers are recommended and why. Starting from the root (Figure 4 and Figure 5), we have a group of questions with two questions: How would you like to display the mask rule? and How would you like to display the error message? In this case, since the size of the screen is not a key issue, we can guess that the end-user chooses to show a textual description of the mask and to show the error message in a new window (A in Figure 7a). Once all the questions of a group of questions have been answered, the flow continues with the next question or group of questions with a pending answer. When a design (a leaf) arises in the path, the flow continues with the closest unresolved question.

In our example, the flow continues with the group of questions for Defined Selection. We guess that the end-user chooses as answers the recommendations for a Web application: using a RadioButton for items between 2 and 9 elements (B in Figure 7a) and using a ListBox for more than 9 items (C in Figure 7a). The next group of questions in the flow elicits requirements for Argument Grouping. According to the recommendations, the end-user selects a Wizard for more than 20 arguments, Tabs for a set between 11 and 20 arguments (D in figure 7a) and a Group Box for less than 10 arguments. Next, the flow continues with the questions regarding the Filters. Since there is much information to store in the system, the end-user selects to display the filters at the top of the interface (E in Figure 7b). This way, the first task end-users do within the interface is filling filters. Next, the flow continues with the questions regarding Order Criteria. Again, the amount of information recommends using order criteria. Since the size of the screen is not a problem, the end-user selects to display the order alternatives at the top of the interface using RadioButtons (which require less clicks than the use of a ListBox) (F in Figure 7b). Next, the flow continues with the questions regarding Display Sets. Since the screen for a Web application is wide, the recommendations suggest displaying the fields per column using different colours per field (G in Figure 7b). Next, the flow continues with the questions regarding Actions. According to the recommendations, the end-user selects to display the actions on the left with a hyperlink and to use a textual description (the size of the screen is not a problem) (H in Figure 7b). Finally, the flow continues with the questions regarding Navigations. The end-user selects to display the navigations at the bottom, since these actions will not be used very frequently (I in Figure 7b). Moreover, the visual appearance of navigations should be a hyperlink, since it is the most common widget for Web applications.
At the end of the process, we have a set of designs we have reached through the navigation of the tree structure. All these designs compose the set of usability requirements for the Web application. As example,

we show the specification of designs D7, D10 and D13 used to display a RadioButton for lists between 2 and 9 items in INTEGRANOVA (B in Figure 7a). Note that all the designs are specified when the tree structure is defined. D7, D10 and D13 are represented in the Abstract Model through the interaction pattern Defined Selection, which is specified through the XML code:

```
<PDefined_Selection id=”List_2-9”>
<Item1> “XXXX” </Item1>
<ItemN> “XXXX” </ItemN>
</PDefined_Selection>
```

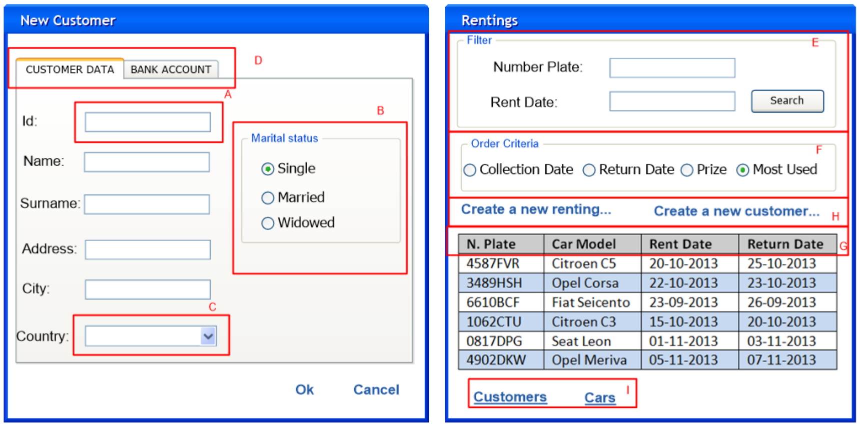
This design is represented in the Concrete Model through the template:

```
:PDefined_Selection_id=”List_2-9”=RadioButton
```

This design is generic for every list of items between 2 and 9 elements. In next steps of the software development process, the analyst must complete this model for each interface that includes the pattern Defined Selection. In our example of Figure 7a, the Abstract Model will be completed with the following XML lines:

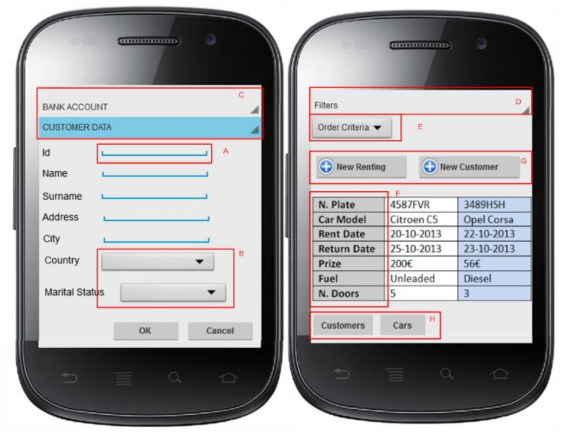
```
<PDefined_Selection id=”List_2-9” name=”Marital_Status”>
<Item1> Single </Item1>
<Item2> Married </Item2>
<Item3> Widowed </Item3>
</PDefined_Selection>
```

The Concrete Model does not need more details to specify how to display the list. The Abstract and Concrete Models are specified together with the other models of the OO-Method framework and finally we can obtain the final system. Figure 7 shows two examples of interfaces compliant with the requirements we have elicited for the Web application.



N. Plate	Car Model	Rent Date	Return Date
4587FVR	Citroen C5	20-10-2013	25-10-2013
3489HSH	Opel Corsa	22-10-2013	23-10-2013
6610BCF	Fiat Seicento	23-09-2013	26-09-2013
1062CTU	Citroen C3	15-10-2013	20-10-2013
0817DPG	Seat Leon	01-11-2013	03-11-2013
4902DKW	Opel Meriva	05-11-2013	07-11-2013

Figure 7.a, b Two examples of interfaces compliant with the requirements for a Web application



N. Plate	4587FVR	3489HSH
Car Model	Citroen C5	Opel Corsa
Rent Date	20-10-2013	22-10-2013
Return Date	25-10-2013	23-10-2013
Prize	200€	56€
Fuel	Unleaded	Diesel
N. Doors	5	3

Figure 8.a, b Two examples of interfaces compliant with the requirements for a mobile application

Second, we use the tree structure again to elicit the usability requirements for the mobile system. In this case, the end-user would accept the recommendations for mobile applications, which claim to reduce as much information as possible in interfaces. In the group of questions Introduction, the end-user chooses to hide mask rules and to show error messages in a new emergent window (A in Figure 8a). Next, in the group of questions Defined Selection, the end-user selects to use ListBoxes in order to reduce the amount of information in interface (B in Figure 8a). Next, in the group of questions Argument Grouping, for a set of arguments between 2 and 20 items, the end-user chooses to use a design with Accordion (C in Figure 8a). Groups with more arguments should be displayed with a Wizard. Next, the end-user selects to display Filters at the top of the interface with an Accordion, since there is much information to display in little space (D in Figure 8b). Next, the end-user also selects Order Criteria at the top of the interface displayed with a ListBox, such a way they do not take up much space (E in Figure 8b). Display Sets are shown per row with colours, since mobile screens are very narrow (F in Figure 8b). Next, the end-user selects to show the Actions on the left of the interface, with a visual appearance of buttons and with a description based on icons (G in Figure 8b). Finally, for Navigations, the end-user selects to display them at the bottom of the interface using buttons, since this is the most frequently used representation for mobile systems (H in Figure 8b). As example of designs specification, we show the specification of D6, D9, D12 and D15, used to display a ListBox for any group of items (B in Figure 8a). The Abstract Model for these designs is the same as the used for D7, D10 and D13. The Concrete Model is:

.PDefined_Selection_id="List2-10"=ListBox

In next steps of the software development process, the analyst must complete the Abstract Model and the Concrete Model for each interface. For the example of list "Marital Status", we can use the same Abstract Model as we defined for Defined Selection in Figure 7a. The Concrete Model does not need more changes. Figure 8 shows the same example of interface represented in Figure 7 but for a mobile system. Filters and Order Criteria have been hidden according to usability requirements.

3.5 Initial Validation of Our Approach

Wieringa (Wieringa, 2010) classifies many different forms of validation that can be conducted with respect to a research proposal. This section describes a laboratory demonstration that we have performed to validate the usability requirements elicitation process. We have used 4 subjects that are members of the PROS research center (http://www.pros.upv.es): 2 subjects play the role of analysts (persons that work usually with INTEGRANOVA) and other 2 subjects play the role of customers (persons without knowledge in INTEGRANOVA). We use two problems: Problem1 is a Web application to manage a car-rental system (like Figure 7) and Problem2 is a mobile application to manage a company of water supply. Table 1 shows the design used in the evaluation.

Treatments	**Without Tree**	**With Tree**
Problems	Problem1	Problem2
Subjects	Analyst1, Customer1	Analyst1, Customer1
	Analyst2, Customer2	Analyst2, Customer2

Table 1. Evaluation design

The experimental process consists in an interview between the analyst and the customer to elicit usability requirements of each problem with the target of developing both problems in INTEGRANOVA. Elicitation of Problem1 is performed without the tree structure and the elicitation of Problem2 is performed with the tree structure of Figure 4 and Figure 5 (design alternatives for INTEGRANOVA). Previously to the elicitation process, we explained how the tree structure works to the analyst. During the interview, the customer can change his requirements if the analyst offers him a better solution. Once the interview is over, we ask the analyst for interface sketches in paper. Next, the customer compares these sketches with his requirements. This way, we can confirm whether elicited usability requirements correspond to expected interfaces by the customer.
The **Factor** used in the experiment is the elicitation technique used for usability requirements.
The factor has two levels: without our proposal and with our proposal. Each level is applied to each problem. **Response variables** are: time spent in the elicitation process (measured as minutes); design alternatives not asked to the customer and design alternatives that the customer changes after talking with the analyst (measured as number of

design alternatives); analyst's satisfaction and customer's satisfaction (measured with a 5 point Likert scale). Table 2 shows the satisfaction questionnaires used.

Analyst's Satisfaction
I have no doubts about customer requirements
I would use the method to elicit requirements frequently
The method to elicit requirements is easy to use
The method to elicit requirements is useful
Customer's Satisfaction
The offered sketches satisfy your expectations
You would change your idea of system for the offered sketches
You think that the analyst has done a good work in the requirements elicitation process

Table 2. Satisfaction questionnaires

Results regarding **spent time** show that time spent using our approach is slightly higher (an average of 5 minutes more). Regarding **design alternatives not asked to the customer** without our approach, Analyst1 forgot asking 68% of design alternatives, and Analyst2 forgot 79%. Both analysts chose the most frequently used design alternatives without contrasting those decisions with the customer. Using our approach, both analysts asked 100% of design alternatives. Regarding **changes in interfaces during the interview**, Customer1 changed 5 features without our proposal and 6 features with our proposal. Customer2 changed 11 features without our proposal and 8 features with our proposal. Regarding **analyst's satisfaction**, both analysts are more self-confident with elicited requirements using our proposal, they would use our approach frequently and they classify our approach as useful and easy to use. Regarding **customer's satisfaction**, there are not differences between using our approach or not for the expected sketch and for the valuation of the analyst's work. Using our approach, both customers prefer the sketches of the analyst rather than their own ideas previous to the interview.

As conclusion, we state that even though this evaluation is a pilot experiment, results show an improvement in the elicitation process of usability requirements in a MDD method such as INTEGRANOVA: more matching between elicited requirements by the analyst and real needs of customers, and more satisfaction for analysts and customers. A disadvantage of our proposal is that it takes more time, since it requires asking the customers all the possible design alternatives.
Note that how to model the interaction and transformations have not been evaluated because they depend on the MDD tool used (INTEGRANOVA in this case).

3.6 Conclusions and Further Work

This paper is a step forward to obtain holistic MDD methods, where all the system features, including usability, can be represented from the early steps till the code (vertical dimension). We propose a process to elicit usability requirements based on existent design alternatives and usability guidelines. The end-user must participate in the process, choosing the design alternative that better fits with her/his requirements. The approach is based on the construction of a tree structure that represents all the design alternatives. How to build the tree structure and how to use it, is explained in detail. Moreover, the approach has been validated with 4 subjects through a laboratory demonstration.
Note that the approach is valid for any MDD method but, as illustrative example, we have used OO-Method. This choice has led the design alternatives and the construction of the tree structure. The use of our approach in other MDD method, with models to represent the interaction different from the Abstract Model and the Concrete Model of OO-Method, involves building another tree structure. The size of the tree structure depends on the number of design alternatives; the more alternatives, the higher is the tree structure. One benefit of our proposal is that its use does not involve changing the existing MDD method. We do not propose any extension of existing interaction models or new transformation rules. We propose using existing interaction models to represent designs of our tree structure, and those models will be the

input for existing transformation rules in next steps of the development process (if the existing MDD method supports these transformations).
In our example, we have used two usability guidelines: ISO 9126-3 and the ergonomic criteria. In Human-Computer Interaction and in Software Engineering communities there are many other guidelines. Our approach accepts as many guidelines as the analyst would like to consider. A contradiction between two guidelines does not mean a problem, since the end-user decides the most suitable design alternative. However, it is important to mention that too many recommendations for the possible designs can confuse end-users.
Our approach focuses on eliciting usability requirements. As outcome of our elicitation process, we get some incomplete conceptual models. In next development steps, the analyst must enhance these models with primitives that represent the functionality and the visual appearance of the system in order to get a fully functional system. How the usability requirements will be expressed in the next steps of the software process will depend exclusively on the MDD method.
As future work, we plan to develop a tool to support the construction and use of any tree structure. Even with a few design alternatives and a few usability guidelines, the size of the tree structure is considerable. Moreover, we also plan to apply our proposal to a real case study in industry with more subjects than the ones used in this paper.

Acknowledgements

This work has been developed with the support of MICINN (PROS-Req TIN2010-19130-C02- 02), UV (UV-INV-PRECOMP13-115032), GVA (ORCA PROMETEO/2009/015), and cofinanced with ERDF. We also acknowledge the support of the Intra European Marie Curie Fellowship Grant 50911302 PIEF-2010. We also thank Sergio España, Francisco Valverde, Marcela Ruiz and María Jose Villanueva for their participation in the experimental validation.

References

1. Acerbis, R., Bongio, A., Brambilla, M., & Butti, S. (2007). WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications. LNCS, 4607, 501-505.
2. Android, D. (2014). User Interface Guidelines, from http://developer.android.com/guide/practices/ui_guidelines/index.html
3. Aquino, N., Vanderdonckt, J., Valverde, F., Pastor, O. (2008). Using Profiles to Support Model Transformations in the Model-Driven Development of User Interfaces. Paper presented at the Proc. of 7th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2008, Albacete, Spain.
4. Bass, L., & John, B. (2003). Linking Usability to Software Architecture Patterns through General Scenarios. Journal of Systems and Software, 66(3), 187-197.
5. Bastien, J. M., Scapin, D. (1993). Ergonomic Criteria for the Evaluation of Human-Computer Interfaces. Rapport technique de l'INRIA, 79.
6. CARE. (2014). CARE Technologies, from https://www.care-t.com.
7. Cerejo, L., A. (2011). User-Centered Approach To Web Design For Mobile Devices. Retrieved 11 october 2012, from http://mobile.smashingmagazine.com/2011/05/02/a-user-centeredapproach-to-mobile-design/
8. Ceri, S., Fraternali, P., & Bongio, A. (2000). Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. Computer Networks, 33(1), 137-157.
9. Cronholm, S. (2009). The Usability of Usability Guidelines: A Proposal for Meta-guidelines. Paper presented at the 2lth Australasian Conference on Computer-Human Interaction, Melbourne, Australia.
10. Cysneiros, L. M., Werneck, V. M., & Kushniruk, A. (2005, Aug 29 - Sept 2, 2005). Reusable Knowledge for Satisficing Usability Requirements. Paper presented at the 13th IEEE International Conference on Requirement Engineering, Washington, DC, USA.
11. Embley, D. W., Liddle, S. W., & Pastor, O. (2011). Conceptual-Model Programming: A Manifesto. In D. W. Embley & B. Thalheim (Eds.), Handbook of Conceptual Modeling (pp. 3-16): Springer Berlin Heidelberg.
12. Escalona, M. J., & Arag, G. (2008). NDT. A Model-Driven Approach for Web Requirements. IEEE Trans. Softw. Eng., 34(3), 377-390.

13. Ferre, X., Juristo, N., & Moreno, A. M. (2005). Framework for integrating usability practices into the software process. Paper presented at the Proceedings of the 6th international conference on Product Focused Software Process Improvement.
14. Folmer, E., & Bosch, J. (2004). Architecting for Usability: A Survey. Journal of Systems and Software, 70, 61-78.
15. Frankel, D. (2002). Model Driven Architecture: Applying MDA to Enterprise Computing: John Wiley & Sons, Inc.
16. Hassenzahl, M. (2008). The interplay of beauty, goodness, and usability in interactive products. Hum.-Comput. Interact., 19(4), 319-349.
17. Henninger, S. (2000). A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design. Interacting with Computers, 12(3), 225-243.
18. ISO-9126. (2001). Software Engineering - Product Quality - Part 1: Quality Model.
19. ISO-9241_11. (1998). Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability.
20. Johnsonbaugh, R. (1997). Discrete Mathematics (Fourth ed.). New Jersey: Prentice Hall Intemational.
21. Jokela, T., Koivumaa, J., Pirkola, J., Salminen, P., & Kantola, N. (2006). Methods forQuantitative Usability Requirements: A Case Study on the Development of the User Interface of a Mobile Phone. Personal Ubiquitous Comput., 10(6), 345-355.
22. Koch, N., Knapp, A., Zhang, G., & Baumeister, H. (2008). UML-Based Web Engineering, an Approach Based on Standards In Web Engineering, Modelling and Implementing Web Applications (pp. 157-191): Springer.
23. Molina, P. J., Meliá, S., & Pastor, Ó. (2002). JUST-UI: A User Interface Specification Model. Paper presented at the Proceedings of Computer Aided Design of User Interfaces, CADUI'2002, Valenciennes, Francia.
24. Nokia. (2014). Symbian Design Guidelines - Dialogs. From http://www.developer.nokia.com/Resources/Library/Symbian_Design_Guidelines/
25. Ormeño, Y. I., & Panach, J. I. (2013). Mapping study about usability requirements elicitation. Paper presented at the Proceedings of the 25th international conference on Advanced Information Systems Engineering.
26. Ormeño, Y. I., Panach, J. I., Condori-Fernandez, N., & Pastor, O. (2013, 29-31 May 2013). Towards a proposal to capture usability

requirements through guidelines. Paper presented at the IEEE Seventh International Conference on Research Challenges in Information Science (RCIS'2013).

27. Panach, J. I., Condori-Fernández, N., Vos, T., Aquino, N., & Valverde, F. (2011). Early Usability Measurement In Model-Driven Development: Definition and Empirical Evaluation. International Journal of Software Engineering & Knowledge Engineering (IJSEKE).
28. Panach, J. I., España, S., Moreno, A., & Pastor, O. (2008). Dealing with Usability in Model Transformation Technologies. Paper presented at the ER 2008, Barcelona.
29. Pastor, O., Molina, J. (2007). Model-Driven Architecture in Practice. Valencia: Springer.
30. Sutcliffe, A. G., Kurniawan, S., & Jae-Eun, S. (2006). A Method and Advisor Tool for Multimedia User Interface Design. Int. J. Hum.-Comput. Stud., 64(4), 375-392.
31. Wieringa, R. (2010). Design science methodology: principles and practice. Paper presented at the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering.

2.4 An Empirical Experiment of a Usability Requirements Elicitation Method based on Interviews

Context: *The usability requirements elicitation process is a difficult task that lacks methods to guide and help analysts, who are usually not experts at usability.* ***Objective****: This paper conducts an experiment with two replications to evaluate a method that elicits usability requirements based on structured interviews named UREM versus an unstructured method. The method consists of guided interviews by the analyst using decision trees. The tree is composed of questions and possible answers. Each question appears when there are different possible design alternatives, and each answer represents one of these alternatives. The tree also recommends the alternative that enhances the usability based on existing usability guidelines.* ***Method****: We have conducted an experiment with two replications with 22 and 26 subjects playing two different roles in a within-subjects design. The analysts used a tree to guide the interview and elicit the requirements while the end users had to explain to the analyst the type of system to develop. During the interview, the analyst must design a paper prototype to be validated by the end user. For the analyst, the experiment measures the effectiveness of usability requirements elicitation, the effectiveness of the use of the usability guidelines, the efficiency of the elicitation process, and the satisfaction with the entire elicitation process. For the end user, the experiment measures the satisfaction with the designed prototype at the end of the interview.* ***Results:*** *UREM yielded significantly better results for the effectiveness in the usability requirements elicitation process and for the effectiveness in the use of usability guidelines when compared to unstructured interviews. The use of UREM did not reduce the analysts' efficiency and both analyst and end user remained the same satisfaction. Conclusions: Eliciting usability requirements is a difficult task if it is done with unstructured interviews and without usability recommendations.*

Versión del autor del artículo., Ormeño, Y. I., Panach, J. I., & Pastor, O. (2023). An Empirical Experiment of a Usability Requirements Elicitation Method to Design GUIs based on Interviews. Information and Software Technology, 107324, https://doi.org/10.1016/j.infsof.2023.107324

1. Introduction

Usability is an important quality characteristic of software and is an essential element to be considered in the development of different software systems in order to determine the development's success or failure [1-2]. The ISO 9241-11 [3] standard defines usability requirements as the effectiveness, efficiency, and satisfaction of a user achieving his/her goals in a defined context of use. Similarly, according to the ISO/IEC 25010 [4] standard, usability is the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. Today we live with new and innovative ways of interacting with computers, and this era requires application software that has high usability levels that decrease potential usability difficulties and risks [5]. However, usability requirements are usually ignored during the software development process, especially in the early stages of requirements elicitation. This increases the cost of solving usability problems and affects the quality of final products.

The software engineering and requirements engineering community knows that the process of eliciting the usability requirements of a system is not an easy task and requires a lot of effort. Therefore, methods that help software engineers or systems analysts in the process of eliciting usability requirements are needed, reducing time and resource costs, and complying with standards or regulations for different domains and platforms. Since usability is a multifaceted concept, there are many usability techniques for performing usability studies. Interviews and prototypes are the most common techniques used to elicit usability requirements, but they must be structured correctly so that they can be defined, measured, and evaluated properly [6]. An analyst that elicits requirements is not usually an expert at usability and needs some guidelines to be able to design usable interfaces.

In order to help analysts design usable systems, in a previous work [7], we proposed the Usability Requirement Elicitation Method (UREM). UREM consists of a decision tree where nodes are questions and answers. The analyst must navigate throughout the tree asking questions to the end user and providing to the end user different answers as possible design alternatives. Questions appear when the analyst has to choose among several design alternatives. Each answer is a design alternative. In order to help in this choice, the tree must also show which alternative optimizes the usability. Each answer of the tree has a description that suggests for which circumstances this design is recommended. Thus, the analyst can recommend a specific option to the end user, but the end user is the one who desires what she/he prefers. The recommendations have been extracted from usability guidelines. The question-answer format of this interview is a way to guide the requirements elicitation process in order to elicit usability requirements. During the interview, the analyst must design a paper prototype with the GUI. The end user must validate this design, proposing any changes that she/he considers optimize usability. Usability requirements is a concept that affects many factors, not only the visible GUI that is the result of the design, but also functionality, learnability, efficiency, etc. [8]. UREM can be used for all the usability requirements whose guidelines can be written in the tree structure as answers or recommendations.

The main contribution of this article is the design and conduction of an empirical experiment to validate UREM with two replications of 22 and 26 subjects respectively. The design includes two treatments: unstructured interviews and UREM. Both treatments are participatory methods to involve the end users and analysts throughout the design process [9]. The experiment is a within-subjects design (repeated measures) where each subject plays the role of analyst or end user in one of both treatments. We defined 24 pairs of subjects from the 48 subjects recruited for both replications. In each of these pairs, roles were swapped during the application of each treatment. The subject that played the role of analyst had to guide the interview in order to elicit the usability requirements and validate these requirements using a paper prototype. The subject that played the role of end user had to explain to the analyst the type of system they needed and the usability requirements that had to be included. We used two different problems in order to avoid the carryover effect between treatments. For the analyst, the response variables were: the effectiveness of the usability requirements successfully elicited; the effectiveness of the usability guidelines properly applied in the prototype; the efficiency in the requirements elicitation process; and the satisfaction during the whole elicitation process. For the end user, the response variable was the satisfaction with the designed GUI.

The results yielded two significant differences between UREM and the unstructured interview: 1) UREM was more effective in the usability requirements elicitation; 2) UREM was more effective in the application of the usability guidelines to improve usability. The lack of significant differences in efficiency using the two elicitation methods means that, even though UREM might be considered more cumbersome at first glance, its use did not increase the time required to design the GUI. The improvement in effectiveness using UREM does not lead to an improvement in the satisfaction of the analyst and the end user. An analysis of these results is discussed in the article.

This article is organized as follows. Section 2 describes the related works. Section 3 explains UREM and the unstructured interview in detail. Section 4 justifies the experimental design. Section 5 presents the statistical results. Section 6 discusses and interprets the results. Finally, Section 7 presents the conclusions and future work.

2. Related Works

In this section, we describe works that are related to usability requirements elicitation and their empirical validations. We conducted a Targeted Literature Review (TLR) [10], which is a non-systematic, in-depth, and informative literature review aimed at keeping only the significant references in order to maximize rigorousness while minimizing selection bias. For this purpose, the semantic question about usability requirements elicitation is translated into the following syntactical queries used as a search string: ("usability requirements" AND ("method" OR "methodology" OR "model") AND ("experiment" OR "case study")). This search string was applied to the title, keyword, and abstract of the Scopus digital library, ACM Digital Library, Web of Science, and IEEExplore in May 2023.

As exclusion criteria, we have: 1) tutorial papers; 2) papers that do not deal strictly with usability requirement elicitation; 3) papers that do not report the results of the experiment; 4) papers without methods or models; and 5) paper without any experimental design carried out. As inclusion criteria, we have: 1) papers that describe the developing methodology in usability requirement elicitation; 2) papers that describe how they evaluated or analyzed developing methodology; and 3) papers that include a case study and/or guidelines for the elicitation process. The search string returned 22 papers from the Scopus digital library and 23 papers from the IEEExplore digital library. After applying the exclusion and inclusion criteria to the title and abstract, and gathering the papers from both outlets and search string, we finally analyzed the content of 15 papers, which we describe below. The references resulting from these searches were classified into four categories, which are discussed further in the following subsections. This classification aims to identify the papers that have proposed requirements elicitation methods for both specific contexts and non-specific contexts, papers that use usability guidelines in their proposals of requirements elicitation, and papers that validate empirically a requirements elicitation method. These four types of papers cover the target of our contribution: an empirical validation of a requirements elicitation method of non-functional requirements based on usability guidelines. Table 1 shows a summary of all of these works, comparing the proposed method, metrics, tools, and techniques.

2.1 Usability Requirements Elicitation for Specific Contexts

This subsection describes the works whose processes have been developed to be carried out for a specific problem domain, to test the method in an existing application, or to understand/complement it. Gunduz and Pathan [11] describe usability problems found in touchscreen mobile flight-booking applications and suggest solutions to eliminate such problems. A qualitative research approach is used for usability analysis. They considered users' actions and reactions towards the application for their specific context and collected their opinions with regard to efficiency, user satisfaction, and adoption of the application. The case study was carried out on a Turkish Airlines' commercial mobile flight-booking application where 20 interviewees from different countries were randomly selected from novice and advanced users. They use questionnaires and interviews during the practical investigation.

Troyer and Janssens [12] present a Feature Modeling method which is a variability modeling technique used in Software Product Lines. It has a twofold approach: one to unlock available information on requirements elicitation and the other to provide a mechanism for guiding the stakeholders (non-computing people) through the requirements elicitation process. The feature model is supported in a tablet app that provides explanations for different usability issues, possible design options and alternatives, and the impact of the choices. Two case studies based on games and e-shop web applications were conducted using evaluation sessions that focused on the usability of the tool, brainstorming sessions, and templates done by requirements engineering experts.

Fahey et al. [13] describe the value of a design approach to elicit user requirements by performing business process modelling (BPM) and the elicitation and modelling of user requirements through the work of the users. It presents a case study of how an outpatient Electronic Patient

Record (EPR) system was successfully implemented in the Epilepsy Unit of Beaumont Hospital, Dublin. The determination of functional (FR) and non-functional user requirements (NFR) was realized through a series of traditional requirements elicitation techniques such as workshops and multi-stage Delphi interviews. Process maps were drawn up and confirmed with end users, and new prototypes were developed on paper and on mock-up screens. They conclude that the more time spent on usability issues in the early stages of system development, the more likely a system will undergo a successful implementation with minimal disruption of the necessary services.

Temper et al. [14] introduce an efficient continuous biometric authentication technique using touchscreen gestures and related posture information that is based on a Vaguely Quantified Nearest Neighbor classifier combined with a scoring model and fuzzy classifier. A bank app prototype implemented on a Google Nexus 4 mobile phone was developed to evaluate the security and usability requirements. The evaluation was conducted with 22 volunteers based on a trust score which was used as an indicator to verify whether or not the person that enters information within the app is a legitimate user. The calculation of the score is based on touchscreen gestures and posture information. The results depicted how the trust score evolves over time. The initial results showed the applicability of behavioral biometrics as an additional security mechanism on mobile phones.

Rocha et al. [15] have defined a method to elicit requirements based on structured interviews using user stories. These user stories are used in a behaviour-driven development context with templates for guiding the writing of such stories. The approach can be helpful to ensure that consistent information about the requirements is provided. User stories written using terms of an ontology describing events, behaviours, and user interface elements can be used to promote consistency of requirements. Moreover, user stories can be used for testing the automation of diverse types of artefacts, such as task models, low-fidelity prototypes or final implementation of the interactive system. The approach was validated in a case study with potential product owners in a research institute, where subjects had to write their own user stories to describe a feature they are used to performing.

The above research works were performed for a specific context. the work of Troyer and Janssens [12] is for Software Product Lines, the work of Fahey et al. [13] is for BPM, the work of Temper et al. [14] is for touchscreen gestures, and the work of Rocha et al. [15] is for behaviour-driven development. Each method seeks to elicit requirements and to find solutions for usability issues in its own way. The techniques that are most widely used to support the methods are unstructured interviews, brainstorming, focus groups, and questionnaires with Likert scale, but there are also proposals such as the work of Rocha et al. that propose a structured method.

2.2 Usability Requirements Elicitation for Non-Specific Contexts

This subsection describes the works to elicit requirements that have been performed from a non-specific context, i.e., the method can be applied in different domains. De Carvalho et al. [1, 16] evaluate the possibility of discovering usability requirements from information in the Functional Resonance Analysis Method (FRAM) in the health field. The methodology follows these steps: 1) identification of the context; 2) identification of problems and difficulties in the execution of a task; 3) definition of solutions; and 4) definition of software requirements. Two experiments were conducted. The first one was a patient selection process with BPMN notation, and the second one was a patient selection process through a FRAM model. The results showed that the FRAM method used for complex systems yields more requirements, especially usability requirements. There was also superiority in the average performance related to the number of requirements per activity/function, the average in functional requirements, and the quality (availability, understanding, clarity, completeness) of the elicited requirements.

Nhavoto [17] presents an integrated mobile phone text-messaging system that is used to follow up on Human Immunodeficiency Virus (HIV) and Tuberculosis (TB) patients. The study focuses on three key activities: eliciting the requirements, design of the GUIs, and implementation of a prototype named SMSaúde to facilitate communication between patients and the healthcare systems. Testing and evaluation of the SMSaúde system were done using seven quality criteria (functionality, completeness, consistency, accuracy, performance, reliability, and usability) and six different requirements (data collection, telecommunication costs, privacy, data security, the content of text messages, connectivity, and system scalability). The artifact was improved interactively and incrementally. During the design and development process, a broad set of usability requirements was identified in two brainstorming design sessions. They plan to perform an evaluation of the system, including a satisfaction survey of the health professionals and patients.

Elias [18] presents a semi-automatic validation system to improve usability in Computer Support Collaborative Learning (CSCL) environments. It uses an ontology to represent usability knowledge and software agents to automate the process. This system uses usability methods and techniques to create SPARQL rules to deal with usability issues. The rules were performed by the interaction among agents, using questionnaires to know the users' opinion about usability. A case study in a real collaborative learning environment based on Moodle at Federal University of Alagoas - Brazil was described to present the advantages of using the proposed system. As a result, the system provides graphical reports and checklists to help the administrator improve the usability of the CSCL environment.

Yuan, X. and X. Zhang [19] present an ontology model to represent the knowledge of common and variable software assets for interactive requirements elicitation. The instances of an abstract model help the interactive software customization system to communicate with software clients via dialogue in natural language. In order to demonstrate how it works and to provide evidence of its usability, they include a case study of an online book shopping system with experienced and non-experienced software clients. The system retrieves product information from the ontology model and presents software requirements in utterances as slots for users to fill in. Learnability, efficiency, reliability, and satisfaction, along with several other measurements, were evaluated. The proposed approach was capable of not only eliciting requirements but also automatically converting client-picked requirements into service descriptions in Web Ontology Language for the production of customized software systems.

Abad et al. [20] study the impact of Loud Paper Prototyping (LPP) on requirements elicitation. They compare this technique with several variations of Silent Paper Prototyping (SPP) such as traditional Woz, sketching, and storyboard. Furthermore, they present a comparison between LPP and elicitation meetings alone as well as paper prototyping versus No Paper Prototyping (NPP). Two research questions were defined: 1) How does paper prototyping help in capturing mobile App requirements?; and 2) Does LPP affect the type of requirements extracted during requirements elicitation? These questions were analyzed in a case study with two mobile application developments teams. The results showed that 1) SPP is more efficient in capturing NFRs than NPP; and 2) LPP is more useful in adding new NFRs and moving/modifying existing ones. Among the techniques reviewed, most teams found LPP to be the most useful approach for managing mobile application requirements.

All of these research works deal with methods, models, and techniques that are oriented to information management in order to elicit requirements during the design and development process. The elicited usability requirements were generally obtained from brainstorming sessions, interview sessions, and questionnaires. Some works show a formal analysis of data to improve the elicitation of usability requirements by algorithms. The selected case studies were adapted to methods or models in order to demonstrate their effectiveness. In most of the previous works, the usability requirements are studied together with functional requirements and other NFRs in the elicitation process. In other words, the methods are not exclusive to the elicitation of usability requirements.

2.3 Using Guidelines

This subsection describes the papers whose elicitation method depend on usability guidelines. Márquez and Taramasco [21] present a methodology that uses dissemination and implementation (D&I) strategies to recommend requirements elicitation guidelines [22] for eliciting requirements in health systems. The D&I framework considers two phases: The first phase aims to identify the goals of the system. The second phase is about the implementation strategies and requirements elicitation guidelines represented in a model and a multidimensional catalog based on a source of knowledge that generates a set of guidelines for the elicitation of requirements to be evaluated by IT professionals. Working sessions were conducted by IT professionals and clinicians to ensure that each strategy/guideline relationship was fully explained. To assess the impact of using the D&I framework, the authors present a real clinical software case study of the main software component of SIGICAM related to clinical priorities that were developed using the D&I framework. The analyzed variables were: impact, perceived usefulness, perceived ease of use, and user control. The results show an acceptable level of usability with approximately 72% approval.

Abdallah et al. [23] introduced an enhancement of an eXtreme Scenario Based Design (XSBD) process named Quatified eXtreme Scenario Based Design (QXSBD) to quantify usability. QXSBD complements XSBD with a set of usability metrics that need to be assessed in an agile process based on usability guidelines. This framework uses the Usability Critical Parameters Workshop (UCPW) to identify usability scenarios from stakeholders (usability engineers, developers, end users, and customers) and Quality in Use Integrated Model (QUIM) procedures to assign required values. The UCPW provides engineering practices defining the usability requirements and design goals. In order to demonstrate the feasibility of the QXSBD, an interactive system, Customer Request Project, was implemented where efficiency, effectiveness, productivity, and learnability were selected as usability critical parameters. After applying the QXSBD process, the usability defect rate was reduced by 30%. The team questionnaire and end user questionnaire show that UCPW provides practical tactics and guidelines to implement usability scenarios on the process cycle, achieving better user satisfaction.

Scope	**Authors**	**Methods**	**Metrics**	**Tools**	**Techniques**
Usability Requirement Elicitation from Specific Context over Existing Systems	Gunduz and Pathan [43]	Qualitative research approach	Easiness, efficiency, user satisfaction, and adoption of the application.		Questionnaire Interview sessions Likert scale questions
	Troyer and Janssens [44]	Feature Modeling	Effectiveness of the Guinea maps tool. Completeness of the template. Relevance of the template. Learnability of the app. Easy of use Good overview	Guidemap tool	Usability questionnaire Interview Templates workshops
	Fahey et al. [45]	Business Process Modelling (BPM)	Usability testing Optimize time management of users Facilitate work practice change		Ethnographic analysis Workshop and multi-stage Delphi interview Iterative prototyping Process maps Screenshots
	Temper et al. [46]	Vaguely Quantified Nearest Neighbor Fuzzy model Rough Set Theory (RST)	Feasibility, trust score, Equal Error Rate	Fuzzy-Weka	Particle Swarm Optimization Fuzzy rules
	Rocha et al. [47]	Behaviour-driven development based on user stories	Adherence to a template to include behaviours		User stories
Usability Requirement Elicitation	De Carvalho et al. [48]	Functional Resonance Analysis Method (FRAM and	Average performance, completeness Likert Scale		Ethnography, Questionnaires,

from Others		MacKnight) and BPMN			
General Methods with Unexisting Systems	Nhavoto [49]	Design science research methodology	Functionality Completeness Consistency Accuracy Performance Reliability and Usability	Web client for the Web-SMS tool	Brainstorming Focus group meetings Algorithm
	Elias [50]	Ontology, software agents, SPARQL rules usability methods	Standardization of Pedagogical Usability Standardization of Technical Usability Moodle graphical report		Questionnaires Usability techniques Checklists
	Yuan, X. and X. Zhang [51]	Ontology model	Learnability Efficiency Reliability Satisfaction		Rules Algorithm
	Abad et al. [52]	LPP (Loud Paper Prototyping) Silent Paper Prototyping (SPP) No Paper Prototyping (NPP)	Learnability Navigation helpful Improvements Understandability		Latent Dirichlet Allocation-LDA NVivo [11] tool

Using Guidelines	Márquez and Taramasco [53]	D&I framework	Perceived usefulness Perceived ease of use and user control Health-ITUES questionnaire		Interviews Requirement elicitation guidelines Working sessions
	Abdallah et al. [54]	eXtreme Scenario Based Design Quality in Use Integrated Model Usability Critical Parameters Workshop	Learnability Efficiency Effectiveness Likert scale		Scenarios Workshops (SUS) questionnaires
Empirical validations	Vitiello et al. [55]	The empowerment-driven (UX) Requirements Engineering method	Index of Self Efficacy (ISE), the Index of Knowledge & Skills (IKS), the Index of Personal Control (IPC), and the Index of Motivation (IMOT). Efficacy and efficiency	Sedato prototype	Interview, Questionnaires
	Tanikawa et al. [56]	Process support method	Validity of the output requirements and the effectiveness		Entry form check item in-house guidelines for usability improvement [Hiramatsu]
	Abad et al. [57]	Wizard-of-Oz (WOz) User Reviews	Efficacy Effective in capturing NFR Clarifying existing FR	Statistical methods Saturate web-based coding tool	Storyboarding Low-fidelity prototyping Meeting Github repository

			Latent Dirichlet Allocation (LDA) algorithm topic models package in R
Peruzzini and Germani [58]	User-Centered Design (UCD) Delphi methodology Design Structure Matrix (DSM) Quality Functional Deployment	Satisfaction Usable solutions Correlation between users' needs and system funcionalities Positive effect on efficiency	Workshops Focus groups Brainstorming Questionnaires

Table 1. Overview of state of the related works

In the previous frameworks, requirement elicitation guidelines are based on a source of knowledge obtained from workshops sessions conducted by usability experts and the IT team. The carrying out of these workshops increases the need to dedicate more time to the process of eliciting, redefining, and updating usability parameters. In addition, the continuous participation of usability specialists is needed to clarify and explain the reasons and effects of the use of these parameters.

2.4 Empirical validations

This subsection describes the empirical evaluations of requirements elicitation methods. There are proposals where the evaluation of the method is unstructured, i.e., formal mechanisms are not used. Vitiello et al. [24] proposed a methodology to extract UX requirements. It is a transformative process that starts from a contextual investigation in order to understand users, their behavior (decision making, self-management, communication, and engagement), and capacities (self-efficacy, knowledge & skills, personal control, and motivation), which are expressed in terms of human needs. The author tested the methodology on a case study of polypharmacy management interviews. The questionnaires give an initial measure of user empowerment perception represented with empowerment perception ratings such as the Index of Self Efficacy (ISE), the Index of Knowledge & Skills (IKS), the Index of Personal Control (IPC), and the Index of Motivation (IMOT). The results showed that an improvement in the described capacity indicators was achieved.

Tanikawa et al. [25] present a method that focuses on clarifying the needs related to the customer's usability (clarification of customer needs) and the matching of these needs with the system design (conformity between needs and design). The approach consists of defining the activities (tasks and procedures) that are needed to support those needs. An entry form is used to specify target tasks of a system, identify representative users, and describe the works they are in charge of in each task. They also developed check items for specifying the characteristics of the users and tasks of the target system based on in-house guidelines for usability improvement [28]. As a result, the needs and requirements generated by the support method were almost equivalent to those extracted with the work of the experts. Positive effects on efficiency and quality improvement of activities were reported, including a reduction of man-hours for preparation of customers interviews and requirements elicitation.

Abad et al. [26] conducted two studies to compare the role of early usability requirements specification and app reviews. The evaluation focuses on how Wizard-of-Oz (Woz) technique can be used to elicit usability requirements. The first study was about the role of Woz in requirement elicitation activity with the use of storyboarding, low-fidelity prototyping, and meetings between the development team and the client. The second study was related to comparing the role of user review analysis and Woz in eliciting and defining mobile app requirements. It was conducted using 40 mobile apps that are available on Google Play. The results showed that while user reviews are a powerful tool for capturing FRs, there were reports of bugs in several app categories. The authors conclude that Woz is effective in capturing usability requirements and clarifying existing FRs.

Peruzzini and Germani [27] propose a new model to design assistive ICT-platforms including smart products and services to support active aging for elderly and frail people by adopting a user-centered approach to define an interoperable architecture that integrates different types of smart objects. The approach aims to deal with three limitations of existing ambient assisted living systems: low system usability, poor acceptance by users, and lack of personalization. As a result, they obtained a highly usable and flexible platform that is designed according to the specific needs of their direct users with high user satisfaction, usable solutions, user-friendly products, and services with high-level functions integrating data from completely different contexts. Techniques such as interviews, questionnaires, focus groups, and brainstorming were used to conduct the process. Positive effects on efficiency and quality improvement of activities were reported, including a reduction of man-hours for preparing customers interviews and for extracting evidence-based requirements.

Most related works are based on interviews and questionnaires, but none include usability recommendations to guide the end user in the different GUI designs. Moreover, the proposed techniques based on interviews are usually unstructured, so, in the end, how the interview is conducted depends on the interviewer's skills. UREM was proposed as an attempt to cover this gap, proposing a structured interview that is specific for usability requirements. The contribution of this article is the validation of UREM based on effectiveness, efficiency, and satisfaction. These three metrics are the most commonly used in the previous works to validate requirements elicitation methods.

3. Usability requirements elicitation process

This section describes the two methods used to elicit usability requirements that we analyze in our experiment. The first method uses unstructured interviews and the second method is UREM [7], which uses structured interviews based on usability guidelines and interface design guidelines by means of a tree structure to minimize the cognitive effort. Note that both methods are participatory methods [9] with the end user. The difference lies in the fact that UREM utilizes a flow for requesting input from the end user and provides usability recommendations. Below, we describe both methods in detail.

3.1 The unstructured requirements elicitation method

The unstructured method [29] consists in eliciting usability requirements in an unstructured way, without any guideline or tool to support the process. These are the steps of the method:

- The process begins with an interview between the analyst and the end user. The analyst must ask to the end user how she/he prefers the GUI. There is no guide for what questions must be asked, what design alternatives are possible, and which design alternative optimizes the usability. The analyst organizes the questions as she/he prefers.
- During the interview, the analyst draws a paper prototype of the GUI described by the end user that best fulfils the elicited requirements.

- During this process, the end user can suggest any changes after seeing the results of the prototype. Thus, the analyst can evolve the prototype during the interview until the end user is completely satisfied with the result and considers that the proposed solution fulfils the GUI requirements.

At the end of the session, we have the paper prototypes of all of the GUI that fulfil the usability requirements from the point of view of the end user.

3.2 The usability requirements elicitation method (UREM)

This section presents a summary of eliciting usability requirements proposed by UREM. UREM is a structured and general purpose method for designing GUIs compliant with usability guidelines, that supports the analyst during usability requirements elicitation. To do this, a tree structure is built by a usability expert based on user interface design guidelines and usability guidelines to be executed in the process of eliciting usability requirements. The tree is composed of four elements: questions, answers, groups of questions, and designs. Figure 1 shows a general schema of the tree structure used by UREM.

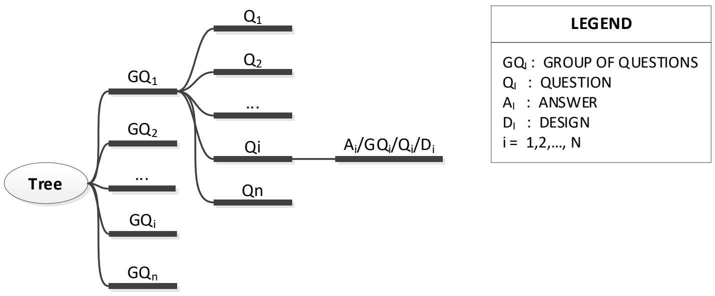


Figure 1. General representation of the tree structure.

We describe each element of the tree as follows.

- Question (Qi) is defined based on UI design guidelines that are represented in different design alternatives for GUI components. The design guidelines present diverse design alternatives for GUI components (e.g. menu). In order to ask the end-user which alternative she/he prefers, we have defined a question when alternatives to design appear. For example, when we are designing a selectable task, we can ask about how to show it. A possible question is “Which UI component is used to show selectable tasks?”

- Answer (Ai) is composed of exclusive alternatives for each question based on GUI design guidelines, where the analyst selects which one best fits the user’s requirement. These options are presented to the analyst in such a way that she/he can choose which one best fits user’s requirements. For each question, some answers are recommended based on usability guidelines. These recommendations aim to help the end user choose the best answers. They are not mandatory; the end user can accept the recommendations or reject them. When answers are shown to the analyst, we will show which answers are recommended by usability guidelines. Possible answers can be yes/no or the choice of one item from a list. For example, the answers to the question “Which UI component is used to show selectable tasks?” can be: RadioButtons, Textfields, CheckBoxes or Slider. According to usability guidelines, a RadioButton is used for a persistent single-choice list.
- Group of Questions (GQi) are created since some branches of the tree structure are not mutually exclusive (the end user should be asked all of the questions). This type of branch is represented by a group of questions that gathers several questions that are grouped by a design characteristic. For example, the question “Which UI component is used to show selectable tasks?” can be gathered with other questions that ask about Selection Dialogues, such as “Where is the action button located?”, “Where is the dialogue box located?”, and “Where is the positive action on a button located?”. All these questions have in common that deal with how selection dialogues are displayed, and all of them are gathered in the same Group of Questions.
- Designs (Di) are the interface designs reached at the end of the tree structure (they are the leaves of the tree). The tree structure

is navigated from the root to the leaves. When the analyst reaches a leaf in the tree, a design has been obtained. The final design of the whole system is the set of leaves in the tree that the analyst has reached. More details can be found in [7]. For example, a design can be a selection dialogue with radio buttons, where each item shows an enumerated data.

The tree structure is built by an expert in interface design and usability. This expert must have enough knowledge to specify design alternatives as questions and answers, as well as to specify the usability guidelines as recommended answers. Once the tree is completed, the analysts can use it an unlimited number of times to elicit usability requirements in several projects. The analysts that use the tree structure do not need knowledge of usability or design since all this information is represented in the tree structure. In order to interview the client to elicit usability requirements, the analyst starts to navigate from the root of the tree, and asks the questions to the end user during the interview. The analyst asks the questions according to their sequence in the tree, from the root to the leaves. The analyst only navigates through the branch of the answer selected by the end user. When the analyst reaches a branch with a group of questions, all of the questions must be answered. Only the analyst can continue with the next question if the flow has reached a leaf and, then continues with the next question in the group of questions. The possible navigation between two nodes of the tree structure can be: 1) from a group of questions to a single question or to another group of questions (Gqi→ Qi / GQi); 2) from a question to an answer (Qi →Ai); 3) from an answer to a question, to a group of questions, or to a design (Ai → Qi / GQi / Di).

The process of eliciting usability requirements is supported by a tool (hci.dsic.upv.es/urem) that supports the creation and navigation of several trees. The analyst uses the tool to perform the elicitation using interview eliciting. The result after navigating the decision tree with UREM can be seen as a design rationale [30-31]; following the flow of the interview we have the report that explains why a system has been designed the way it is. GUI designs must be manually drawn by the analyst.

3.2.1 An illustrative Example of working with UREM

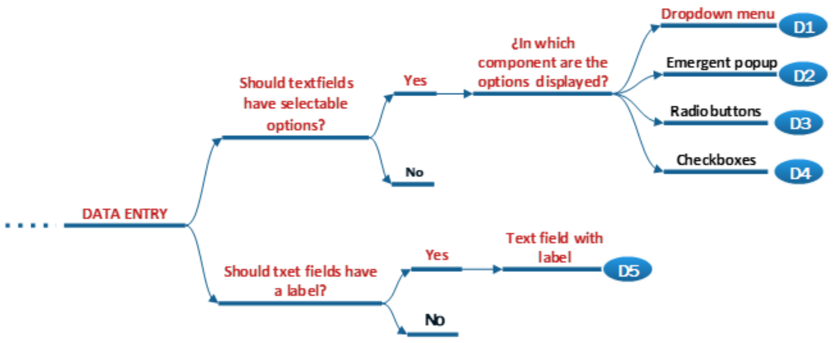

DATA ENTRY
Should textfields have selectable options?
Yes
No
¿In which component are the options displayed?
Dropdown menu
D1
Emergent popup
D2
Radiobuttons
D3
Checkboxes
D4
Should txet fields have a label?
Yes
Text field with label
D5
No

Figure 2. Illustrative example of usability elicitation

This section presents a short and illustrative example of how to deal with UREM to develop a GUI design for a medical system starting from a set of usability requirements and using the usability guidelines represented in the tree structure. The example focuses on the usability requirements that are related to data entry forms (Figure 2). All of the entire process is performed in an interview between the end user and the analyst. The first question that the analyst asks the end user is “Should textfields have selectable options”? This question has two possible answers. “yes” or “no”. The recommended option is “yes”. If the end user opts for “yes”, the next question that the analyst asks is “In which component are the options displayed?” There are four possible answers: Dropdown menu (recommended option); Emergent popup, Radiobuttons; Checkboxes. Each one of these options is a leaf in the tree, so it involves a specific design (Table 2). If the end user opts for the recommendation and chooses the answer “Dropdown menu”, we have reached design D1. Below, the flow continues with the question “Should textfields have a label?”. This question has two possible answers: “yes” or “no”. The answer “yes” is recommended based on usability guidelines. If the end user opts for the recommendation and chooses the answer “yes”, we have reached design D5 (Table 2). Note that D1 refers to the items that compose the textfield, while D5 refers to the label of the textfield.

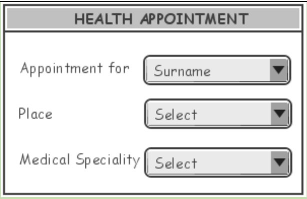
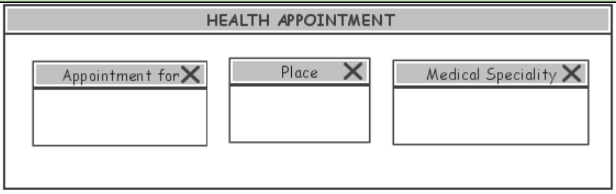
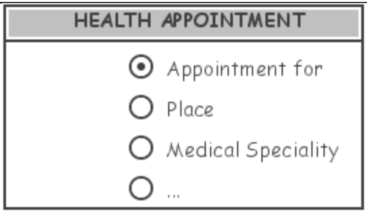
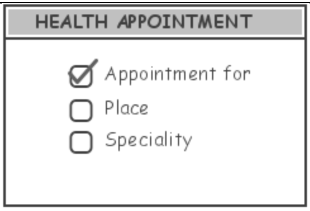
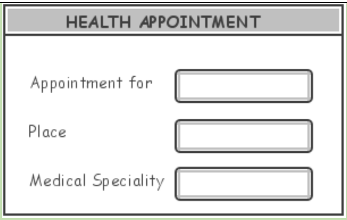
DESIGNS	GUI DESIGNS
D1	 HEALTH APPOINTMENT Appointment for Surname Place Select Medical Speciality Select
D2	 HEALTH APPOINTMENT Appointment for Place Medical Speciality
D3	 HEALTH APPOINTMENT [✓ Appointment for] Place Medical Speciality ...
D4	 HEALTH APPOINTMENT [✓ Appointment for] Place Speciality
D5	 HEALTH APPOINTMENT Appointment for Place Medical Speciality

Table 2. GUI designs for each leaf of the tree

4. Experiment Definition and Planning

In this section, we describe the experiment design according to Juristo and Moreno [32].

4.1 Goal

The main goal of this experiment is to compare the use of a structured method (named UREM) for interviewing the end user in order to elicit usability requirements with the use of unstructured interviews for the purpose of studying the pros and cons of UREM in the GUI design. The experiment is conducted from the perspective of researchers and practitioners who are interested in investigating how useful a structured interview method is compared to an unstructured interview method in eliciting usability requirements.

4.2. Research Questions and Hypothesis Formulation

Our empirical study is based on the concept of quality, which is defined in terms of effectiveness, efficiency, and satisfaction (ISO 25010) [4]. The concept of quality is different depending on the role of the subjects that participate in the validation (as analyst or end user). From the point of view of the analyst, we aim to study whether the requirements elicitation method affects the elicitation process. This means that we need research questions to analyze the effectiveness, efficiency, and satisfaction of the process of usability requirements elicitation. From the point of view of the end user, quality refers to how satisfied the end user is with the designed GUI. Both perspectives of quality are represented in the research questions. Note that the experiment uses a tree structure previously existing. The role of expert in interface design and usability that builds the tree structure of UREM is played by one experimenter. The study of how the tree is built is out of scope of the current analysis. While the construction of the tree structure is done once, its use is unlimited, which leads to focus the experiment on the use of the tree structure instead of its construction. In the experiment, the construction of the tree structure required two hours, including the time to study the design alternatives to be specified as answers, the usability guidelines to be identified as recommendations, and the specification of all this information in the UREM tool. The experimenter who built the tree is an expert in interface design and usability that has been evaluating usability in systems for more than ten years.

The research questions used in our validation are described as follows:

RQ1: Effectiveness is defined in ISO/IEC-25010 as “the degree to which specified users can achieve specified goals with accuracy and completeness in a specified context of use”. *Effectiveness in use* is applied in two contexts: elicited usability requirements (RQ1r) and guidelines recommendations (RQ1g).

RQ1r:

*Is **analyst effectiveness** to elicit usability requirements affected by the elicitation method?*

We operationalize effectiveness as the percentage of usability requirements satisfied by the analyst. The null hypothesis tested to address this research question is: *H01r: The analyst effectiveness using UREM is similar to that of using unstructured interviews.*

RQ1g:

*Is **analyst effectiveness** to apply usability guidelines affected by the elicitation method?*

We operationalize effectiveness as the percentage of usability recommendations that the designed GUI prototype includes. The null hypothesis tested to address this research question is: *H01g: The analyst effectiveness using usability guidelines in UREM is similar to that of using unstructured interviews.*

RQ2: Efficiency is defined in ISO/IEC-25010 as “the degree to which specified users expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use”. *Efficiency* is studied based on usability requirements (RQ2r).

RQ2r:

*Is **analyst efficiency** affected by the usability requirements elicitation method?*

We measure analyst efficiency as the ratio percentage of usability requirements successfully elicited by the time spent to elicit the usability requirements. The null hypothesis tested to address this research question is: *H02r: The analyst efficiency using UREM is similar to that of using unstructured interviews.*

RQ3: Satisfaction is defined in ISO/IEC-25010 as “the degree to which users are satisfied in a specified context of use”. *Satisfaction* is analyzed from two perspectives: analyst satisfaction (RQ3a) and end user satisfaction (RQ3e), since the satisfaction of the analysts who design interfaces may be different from the satisfaction of the end users that will use the interfaces.

RQ3a:

Is ***analyst satisfaction*** *affected by the usability requirements elicitation method?*

We measure analyst satisfaction as the level of contentment of the analysts during the usability requirements elicitation. The null hypothesis tested to address this research question is: *H03a: The analyst satisfaction using UREM is similar to that of using unstructured interviews.*

RQ3e:

Is ***end user satisfaction*** *affected by the usability requirements elicitation method?*

We measure end user satisfaction as the level of contentment of the end-user with the designed prototype as a result of the process of requirements elicitation. The null hypothesis tested to address this research question is: *H03e: The end user satisfaction using UREM is similar to that of using unstructured interviews.*

4.3 Factors and Treatments

We now define factors and their levels to operationalize the reason for our experiment construct. Factors are variables whose effect on the response variables we want to understand [34]. Treatments are the factor alternatives that help us answer the questions of the research hypotheses.

The experiment studies one factor: the usability requirements elicitation method with unstructured interviews (T1) and UREM (T2), where T1 is referred to as the control treatment. Table 3 shows the description of the factor and its two treatments.

Factor	Treatment	Description
Usability Requirements Elicitation Method	T1: unstructured interviews	Experimental subjects elicit usability requirements through unstructured interviews.
	T2: UREM	Experimental subjects elicit usability requirements through UREM

Table 3. Description of the factor and treatments

In the first treatment (T1), the analysts conduct the elicitation process using interviews without any structure. This means that the analysts can ask any question regarding the GUI design. Moreover, even though the subjects playing the role of analysts know usability guidelines, there is no recommendation system to suggest a specific design for enhancing usability (as described in subsection 3.1).

In the second treatment (T2), the analysts use UREM as a method to elicit usability requirements. The analysts must follow a question-answer format based on the different alternatives specified in a decision tree that is defined in advanced. This decision tree also suggests which design alternative optimizes the usability based on usability guidelines. The details of this treatment are described in subsection 3.2.

4.4. Response Variables and Metrics

Response variables are the values that are measured in the experiment in order to study how the factors influence these variables [32]. Below, we define a response variable for each research question (summary in Table 4).

Response Variables	**Metrics**	**Definition**	**Research Questions**
Effectiveness for usability requirements elicitation	Percentage of usability requirements successfully elicited .	Percentage (between 0% and 100%) of the usability requirements included in the GUI prototype after the interview that match the usability requirements of the experimenters' solution.	RQ1r
Effectiveness of usability guidelines	Percentage of usability guidelines used correctly on usability requirement elicitation	The number of usability guidelines used correctly divided by the total number of usability guidelines.	RQ1g
Efficiency for usability requirements elicitation	Percentage of usability requirements successfully elicited /Time spent to complete the usability requirement elicitation process	Time is the amount of minutes that the analyst requires to elicit usability requirements and design the GUI prototype.	RQ2r
Analyst's Satisfaction	Perceived usefulness (PU),	The addition of the questions that ask for PU on a Likert scale	RQ3a
	Perceived ease of use (PEOU)	The addition of the questions that ask for PEOU on a Likert scale	

	Intention to use (ITU)	The addition of the questions that ask for ITU on a Likert scale	
End user's Satisfaction	Computer System Usability Questionnaire (CSUQ)	The addition of the questions of the CSUQ on a Likert scale	RQ3e
	Satisfaction with analyst's recommendations	One extra question in the CSUQ to ask about the usefulness of the recommendations	

Table 4. Response variables

For **RQ1,** Effectiveness is the response variable. This response variable was divided into **RQ1r** to measure the effectiveness of eliciting usability requirements and **RQ1g** to measure the effectiveness of the usability recommendations provided by the guidelines. The metric for RQ1r is calculated as the percentage of usability requirements that are satisfied by the analyst in the GUI prototype built at the end of the interview. For each experimental problem, there is a list of usability requirements that the designed GUI in a prototype must include at the end of the interview. This list is called experimenters' solution *s*ince it is defined by the experimenters (in this case, the authors of the article). Possible values for Effectiveness fluctuate from 0% (no usability requirement of the experimenters' solution appears in the designed GUI) to 100% (all of the usability requirements of the experimenters' solution appear in the designed GUI). The metric for RQ1g is calculated as the percentage of designs reached following the tree structure that fits the recommendations provided by the usability guidelines. Possible values fluctuate from 0% (there is no design that agrees with any usability guidelines) to 100% (all of the designs agree with the usability guidelines).

For **RQ2r**, Efficiency is the response variable. This response variable is measured as the ratio percentage of usability requirements successfully elicited by time spent by the analyst eliciting the usability requirements and drawing the GUI prototype. The time is measured in minutes. The larger efficiency, the better the efficiency.

For **RQ3**, *Satisfaction* is the response variable. This response variable was divided into **RQ3a** to measure the analyst´s satisfaction and **RQ3e** to measure the end user´s satisfaction. RQ3a was measured using the MAM questionnaire developed by Moody [36]. Moody defined a framework (based on the work by Lindland et al..[37]) to measure satisfaction in terms of Perceived Usefulness (PU), Perceived Ease of Use (PEOU), and Intention to Use (ITU). This framework has been previously validated and is widely used [38]. Based on [36], we defined eight questions to measure PU, five questions to measure PEOU, and two questions to measure ITU. The questionnaire is based on a 5-point Likert questionnaire with five possible answers: "Strongly Disagree",

“Disagree”, “Undecided”, “Agree” and “Strongly Agree”. RQ3e is based on the Computer System Usability Questionnaire (CSUQ) [59], which is a 5-point Likert questionnaire that asks about the satisfaction of the end user with the GUI. We have extended this questionnaire with a specific statement to evaluate whether or not the recommendation system was useful: “Are analyst’ recommendations useful to improve the usability of the system?”. Table 5 shows a summary of the research questions, hypotheses, response variables, and metrics used to test these hypotheses.

Research Questions	Hypotheses	Response Variables	Metrics
RQ1r	H_{01r}	Effectiveness of usability requirements elicitation	M1: Completeness
RQ1g	H_{01g}	Effectiveness of usability guidelines	M1: Correctness
RQ2r	H_{02r}	Efficiency for usability requirements elicitation	M2:Completeness/Time
RQ3a	H_{03a}	Analyst Satisfaction	M3A: PU, PEOU, ITU
RQ3e	H_{03e}	End user Satisfaction	M3E: CSUQ

Table 5. Summary of research questions, hypotheses, response variables, and metrics

4.5 Experimental Subjects

The subjects participating in the experiment were undergraduate students in computer science from the Universidad Nacional de San Antonio Abad del Cusco (UNSAAC, Perú). The computer science students have previously taken software engineering courses with enough knowledge about information systems. We selected 48 computer science students. Replication 1 (R1) was conducted with 22 undergraduate students and Replication 2 (R2) was conducted with 26 Master’s students. All of them played the role of analyst and the role of end user. The subjects had previous knowledge of the unstructured requirements elicitation method, but none knew anything about UREM. We spent two hours training the subjects in UREM before conducting the experiment. Apart from a theoretical description, the training activity consisted of doing a brief exercise to navigate throughout the decision tree in order to identify the different alternatives. The subjects filled in demographic questionnaires before running the experiment in order to characterize the population. Table summarize the main characteristics of participants and their background.

None		**1 month**		**1-3 months**		**More than 3-12 months**		**More than 12 months**	
R1	**R2**	**R1**	**R2**	**R1**	**R2**	**R1**	**R2**	**R1**	**R2**
0	0	0	0	10	4	10	4	2	18

Table 6. Job experience at software development companies

		Junior Programmer		**System Analyst/Programmer**		**Lan Technician**		**System Manager**	
Number of students		**R1**	**R2**	**R1**	**R2**	**R1**	**R2**	**R1**	**R2**
		8	4	7	4	5	8	2	6
Duration (months)	Avg.	6	6	12	24	18	24	18	24
	Min	3	3	6	12	8	12	12	12
	Max	12	6	36	36	36	36	24	36

Table 7. Types of jobs performed and the time duration of the job

Experience with	**I have never heard of it**		**I have heard of it**		**I have some knowledge of it**		**I know it**	
	R1	R2	R1	R2	R1	R2	R1	R2
Usability	8	8	7	6	4	7	3	5
User Interfaces design	4	2	11	8	4	11	3	5
Requirements elicitation and requirement analysis	0	0	8	2	7	13	7	11
Requirements elicitation techniques	1	4	5	5	9	9	7	8
Requirements elicitation methods	2	6	4	10	9	5	7	5

Table 8. Experience with software development

Methods	Name of method/technique	Number	
		R1	R2
Unstructured	Interview	20	26
	Focus Group	8	12
	Questionnaires	23	25
	User stories	7	13
	Other	5	12
Structured	Eyetracking	0	0
	Remo	0	0
	Reassure	0	0
	Other	2	0

Table 9. Experience with elicitation methods

Table 7 focuses on development experience measured as the number of months or years that the students have developed software in companies. Most of the participants had work experience even though they were students. Table 6 shows the type of job and the (average, minimum, and maximum) time spent on that job. Table 8 shows their previous experience with usability and requirements elicitation methods. Only 8 persons had not heard of user interface design and only 5 persons had not heard of requirements elicitation techniques. Table 9 shows their previous experience with unstructured interviews and structured methods. Most of the subjects had not worked with any structured method before the experiment, and a few subjects had worked with some method. The item "Other" gathers other options with no agreement among the subjects. Our sample is representative of a population of novice developers. Even though the use of students in experiments limits the generalization of results, it is useful, depending on the target of the experiment, as other works such as Falessi et al. [34] claim. For this experiment, our objective is to compare subjects that have knowledge in unstructured interviews with novice subjects who have experience in structured interviews. At first glance, the structured interview is at a disadvantage due to the absence of experience. Therefore if the results are positive for the structured method, we can conclude that the structured interview is better in spite of this disadvantage. Other benefits of recruiting students are that they often come at a lower cost and are more accessible because they are taking courses at a university. Moreover, for the students, the experiment can be viewed as a learning experience of technology or methods to be evaluated.

4.6. Experiment Design

This section describes the within-subjects design (or repeated measures) where the subjects play two different roles, one for each treatment. We divided the group of subjects into pairs. For each pair, we randomly assigned two roles: analyst and end user. These roles were swapped for each treatment. We used two different problems (one for each treatment) in order to avoid the carryover effect, so this is *paired design blocked by experimental objects* [35]. Table shows the summary of the design that was applied in both replications. In the first session, all of the pairs worked with the unstructured method. Half of the pairs were in a group named G1 and worked with Problem 1 (P1), while the other half were in a group named G2 and worked with Problem 2 (P2). In the second session, the subjects swapped their roles and all of the pairs worked with UREM. G1 worked with P2 and G2 worked with P1.

		P1	**P2**
Session 1	**Unstructured interview**	G1	G2
Session 2	**UREM**	G2	G1

Table 10. Within-subjects design of the experiment

This design has the following advantages: 1) largest sample size possible to analyze the data; 2) we avoid the learning effect; 3) the problem is not confused with the treatments. The expected time required to fulfill the user requirements defined in each treatment was around 30 minutes. This value was defined taking into account two factors: a previous pilot test, and the problem complexity.

The design avoids most of the threats:

- The experiment findings do not depend exclusively on one problem (since we use two problems).
- The pairs cannot share their GUI prototypes with members of other groups since all of the subjects work at the same time with the same treatment.

- All of the subjects are used in both treatments, avoiding variability among subjects.
- The context of the experiment in Session 1 is the same as in Session 2.

4.7 Experimental Object

In order to observe the effects produced by the two treatments (i.e., unstructured interview and UREM), we defined two problems to elicit usability requirements, one for mobile *health center* (P1), and one for *mobile banking* (P2). Both problems are in the context of mobile applications. P1 aims to represent a system where users can login, list the health services, query the schedule for attendance, make a new appointment, and list the previous appointments. P2 aims to implement a bank management application. The end user can log in and access the bank services, such as bank accounts, location of cash dispensers, access news, and language customization. The end user has a personal section where she/he makes bank transfers, list credit cards, and update personal data. Table 5 and Table respectively show the usability requirements that the subjects that play the role of the client must demand in the prototypes designed by the analyst. Even though these lists are not exclusive for each type of problem, using a different list in each problem allows us to validate different branches of the tree structure. These requirements are known by the end user, and the analyst must elicit them with interviews. When clients describe the problem to analysts, they must consider all these requirements shown in Table 5 and Table . The description of the problems in the same way as they were distributed to the clients is shown in Appendix C.

N°	Usability Requirements of List_Req1
1	The widgets must be self-descriptive to facilitate the understanding of the requested data.
2	To avoid errors in data entry, helpful information should be displayed.
3	If the data entry is mandatory, the user should be notified.
4	To facilitate the data entry, the choices must be shown to the user.

Table 5. Mobile Health Center Requirement List

N°	Usability Requirements of List_Req2
1	When inserting data, widgets must avoid errors.

2	Mandatory information must be clearly identified.
3	The system must help fix errors when they arise.
4	The system must offer actions to activate/deactivate pre-established options.

Table 12. Mobile Banking Requirement List

4.8. Instrumentation

All the instruments used for running the experiment can be accessed in a Zenodo repository [36]. Below, we describe all of them:

- **Demographic questionnaires:** The online questionnaires gather information about the subjects', experience using apps or web applications, as well as their level of experience in developing information systems. This questionnaire is shown in Appendix A.
- **Experimental object:** Two problems make up the experimental objects. We have an experimenters' solution with the usability requirements that the GUI must support. This experimenters' solution is shown in Appendix B. The list of requirements shared with the end users to specify the system required is shown in Appendix C
- **Satisfaction questionnaires**: The questionnaires measure the analysts' satisfaction and the end users' satisfaction. Each questionnaire has 15 questions in a 5-Likert scale format. These questionnaires are shown in Appendix D.
- **Spreadsheets**: The spreadsheet is used to evaluate the metrics of the experiment. These calculations were carried out by two experts in usability engineering and measurement.
- **Tool**: This is the tool that supports UREM (http://hci.dsic.upv.es/urem). This tool can guide the end user through the design alternatives, recommending those alternatives that optimize the usability. The tree with of the all the questions, answers, and recommendations is shown in Appendix E.

4.9 Experiment Procedure

This section describes the procedure used to conduct the experiment. This procedure was executed twice, for the two replications R1 and R2). The experimental process consists in interviews within a pair of subjects. The procedure is strictly based on the experiment design configuration shown in Figure 3. The procedure has been labelled with numbers to explain each step. Before the experiment, we explained the goals of the experiment to the experimental subjects as well as the role they played in it. We also randomly created the two groups of subjects (G1, G2). The diagram in Figure 3 summarizes the procedure. Each number inside the circle represents the number of step that is represented in the figure.

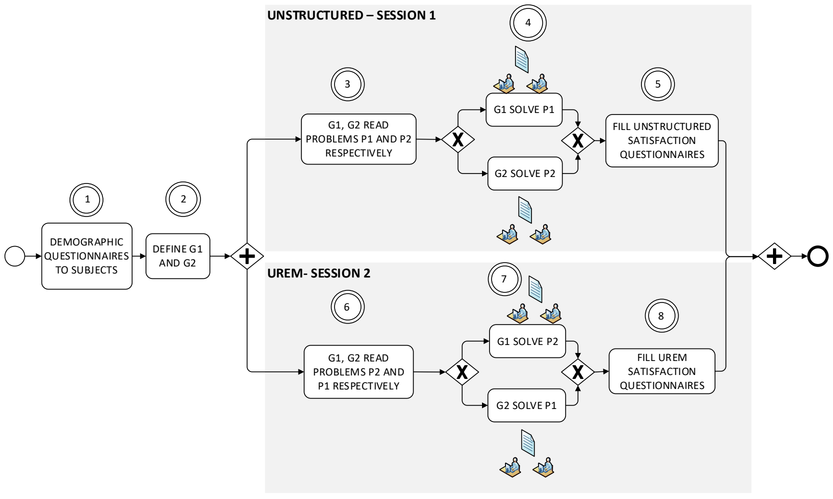

UNSTRUCTURED – SESSION 1
1
DEMOGRAPHIC QUESTIONNAIRES TO SUBJECTS
2
DEFINE G1 AND G2
3
G1, G2 READ PROBLEMS P1 AND P2 RESPECTIVELY
4
G1 SOLVE P1
G2 SOLVE P2
5
FILL UNSTRUCTURED SATISFACTION QUESTIONNAIRES
UREM- SESSION 2
6
G1, G2 READ PROBLEMS P2 AND P1 RESPECTIVELY
7
G1 SOLVE P2
G2 SOLVE P1
8
FILL UREM SATISFACTION QUESTIONNAIRES

Figure 3. Summary of the experimental procedure

Below we describe the steps of Session 1, where unstructured interviews is used.

Step 1. The subjects complete the demographic questionnaire. The questions were the same for all of the experimental subjects independently of their group and role.

Step 2. The experimenter divides all of the subjects into two groups (G1 and G2). The subjects play one role in each of the two sessions.

Step 3. The subjects that play the role of end users read the description of the system (P1 or P2) and the list of the usability requirements that the system must support.

Step 4. The subjects that play the role of analysts must use unstructured interviews to elicit the usability requirements by interviewing the subjects that play the role of end users. Through question-answers, the analysts must draw a prototype of GUI that satisfies the usability requirements for the specific problem.

Step 5. Once the analysts finish the GUI prototype, they complete a satisfaction questionnaire to report their level of satisfaction during the unstructured interview to elicit usability requirements. The end users must complete a satisfaction questionnaire about the result of the prototype. This questionnaire is used to determine whether or not the prototype meets the end users expectations.

Below we describe the steps of Session 2, where UREM is used.

Step 6. The subjects that play the role of end users read the description of the system (a different problem from the one used in Step 3) and the list of the usability requirements that the system must support. The experiment continues in the second session with UREM.

Step 7. The subjects that play the role of analysts must use UREM to elicit the usability requirements by interviewing the subjects that play the role of end users. Following the tree structure, the analysts ask each question following the guide of the tree. The analysts must also recommend the option that best optimizes the usability based on suggestions of the tree. Afterwards, the analysts must draw a prototype of a GUI that satisfies the usability requirements for the specific problem.

Step 8.- Both the analysts and the end users complete the satisfaction questionnaire in the same way as in Step 5, but specifically for UREM.

4.10 Data Analysis

Replications 1 and 2 respectively have 11 and 13 subjects playing the role of analysts. This sample size is not large enough to apply a parametric test. Therefore, when we analyze the replications separately, we opt for a non-parametric test such as Mann-Withney. We consider differences to be significant when the p-value is less than .05. When we analyze Replication 1 and Replication 2 together, we have a large enough sample size (24 subjects playing the role of analysts) to apply the General Linear Model (GLM). There are two requirements for applying a GLM test: homogeneity of the covariance matrices and sphericity. Levene's test is used to check the condition of homogeneity of covariance matrices where the null hypothesis is that the observed covariance matrices of the dependent variables should be equal across groups [37-38]. All of the Levene's test p-values were greater than 0.05. Therefore, we cannot reject the null hypotheses of homogeneity of covariance, which means that the premises of the statistical tests are met in this regard. Mauchly's test is used to check the sphericity condition. In our case, however, there are only two treatments (unstructured interviews and UREM). This precludes a sphericity violation [37], and the test is unnecessary. We regard the differences between treatments as being significant when the GLM p–value is less than .05.

For variables with significant differences according to the GLM, we calculated the degree of these differences using partial eta squared. The partial eta squared results were interpreted as follows: Values of less than 0.3 mean a significant, but weak, effect; values between 0.3 and 0.6 mean a moderate effect, and values greater than 0.6 mean a strong effect. Statistical power is the probability of rejecting a false null hypothesis. Statistical power is inversely related to beta or the probability of making a type II error. In short, power = 1 – β. Power in software engineering experiments tends to be low, e.g., Dyba et al. [39] reports values of 0.39 for medium effect sizes and 0.63 for large effect sizes. Low values of statistical power mean that non-significant results could imply the acceptance of null hypotheses when they are false. Therefore, we calculated the power to find out whether our results were influenced by this widespread problem in software engineering. Note that effect size and power cannot be calculated in non-parametric tests.

5. Results

First, we analyzed the data of each experiment separately using Mann-Whitney as a non-parametric test. Second, we gathered the results using a moderator variable named “Replication” to look for differences between the two experiments. Replication 1 refers to the 22 undergraduate students and Replication 2 refers to the 26 Master’s students (as described in Section 4.5). In the aggregation, apart from analyzing the difference for Method, we looked for differences in the Method*Problem and Method*Replication interactions. This test is based on the GLM. Below, we analyze the results ordered by response variable.

5.1 Effectiveness of Usability Requirements Elicitation

Table 13 shows the statistical results of Replication 1 and Replication 2 separately and both replications together. Replication 1 yielded significant results for the method. The average for effectiveness in the usability requirements elicitation was 78.18 for the unstructured interview and 93.45 for UREM. Therefore, we conclude that UREM yields better effectiveness for Replication 1. Even though Replication 2 did not present statistical differences, the p-value is very close to being less than 0.05 (it is exactly 0.05). When analyzing the averages of Replication 2, the unstructured interview was 71.01 and UREM was 86.61. Thus, there is a clear trend showing that UREM yields better effectiveness in the requirements elicitation process.

Figure shows the box-plot analyzing the two replications together. The first quartile, the median and the third quartile are clearly better for UREM. When analyzing the data with GLM, we obtained a p-value of .000 (Table 13), which means that UREM was statistically better than the unstructured interview. The effect size (.274) yielded a weak effect, and the power (.978) was enough to avoid rejecting the null hypothesis for poor sample size. There are no significant differences in the Method*Problem and Method*Replication interactions, which means that the results do not depend on the problem used or the replication where the experiment was conducted.

In conclusion, **we reject H_{01r}** (the analyst effectiveness using UREM is similar that using unstructured interviews.), since UREM yielded better results than the unstructured interview.

	Rep. 1	**Rep. 2**	**Both rep.**
p-value Method	**.001**	.05	**.000**
p-value Method*Problem	-	-	.195
p-value Method*Replication	-	-	.195
Effect size	-	-	.274
Power	-	-	.978

Table 13. Statistical results of effectiveness for usability requirements elicitation

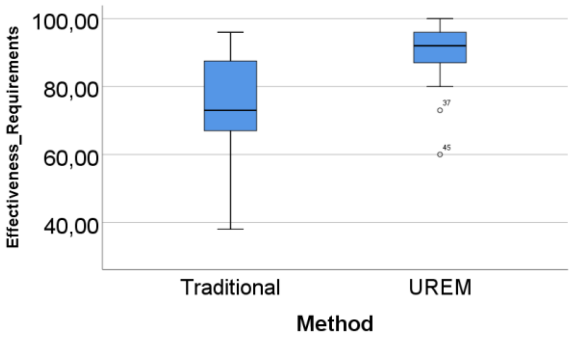


Figure 4. Box plot of effectiveness for usability requirements elicitation with both replications

5.2 Effectiveness of Usability Guidelines

Table 6 shows the statistical results after applying the non-parametric test and GLM to each replication alone and both replications together, respectively. Both Replication 1 and Replication 2 yielded significant results (p-value of .001 and .000[1]). In Replication 1, the average for the effectiveness of the guidelines was 35.36 for the unstructured interview and 62.72 for UREM. Replication 2 also showed a better average for UREM (71.76) than the unstructured interview (33.76). Therefore, we can state that, in both replications, UREM yields a design that better fits the usability guidelines.

Figure 5 shows the box-plot of both replications together. The first quartile, the median and the third quartile are better for UREM. When analyzing the data with the GLM test, we obtained a p-value of .000 (Table 14), which means that UREM is statistically better than the unstructured interview. The effect size of .571 means a moderate effect and the power of 1 is very high, which ensures having enough sample size to avoid rejecting the null hypothesis for a lack of sample. There were no significant differences in the Method*Problem and Method*Replication interactions, which means that results do not depend on the problem used or the replication where the experiment was conducted.

In conclusion, **we reject H_{01g}** (the analyst effectiveness using usability guidelines in UREM is similar to that of using unstructured interviews) since UREM yields better results than the unstructured interview.

	Rep. 1	**Rep. 2**	**Both rep.**
p-value Method	**.001**	**.000**	**.000**
p-value Method*Problem	-	-	.05
p-value Method*Replication	-	-	.05
Effect size	-	-	.571
Power	-	-	1

Table 6. Statistical results of effectiveness for usability guidelines

[1] We use only 3 decimals even though the statistical package works with more.

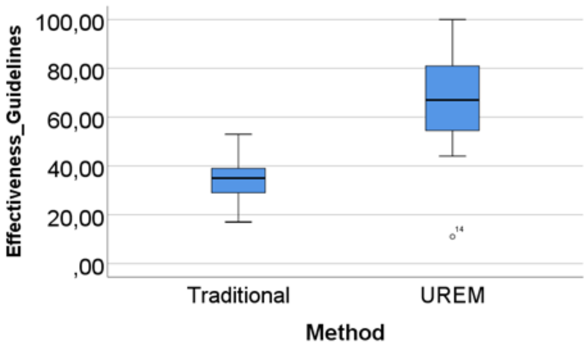


Figure 5. Box plot of effectiveness for usability guidelines with both replications

5.3 Efficiency for Usability Requirements Elicitation

Table 15 shows the statistical results of Replication 1 and Replication 2 separately and both replications together. Replication 1 shows a significant result with a p-value of .018 while Replication 2 shows no significant results with a p-value of .489. In Replication 1 the average was .953 for the unstructured interview and 1.34 for UREM. In Replication 2, the average was 0.998 and .886 respectively. The results are contradictory in both replications, but the differences are so slight that we cannot draw conclusions.

Figure 6 shows the box-plot of efficiency aggregating both replications. The median, the first quartile, and the third quartile are slightly better for UREM. Although these differences are not strong, UREM shows a trend with a better efficiency. The GLM test showed no significant results (p-value .220), with a power of .230, which is low. A larger sample size may produce some significant differences between treatments. Both the Method*Problem and Method*Replication replications yielded significant differences. This means that there is a specific problem and a specific replication that affects the result. To analyze this idea, in Figure 7 we show profile plots of both interactions. Figure 7 a) shows that the Bank Problem (P2) is better in UREM. Figure 7 b) shows that Replication 1 is better for UREM.

In conclusion, **we cannot reject H_{02r}** (the analyst efficiency using UREM is similar to that of using unstructured interviews), so there are no differences between the unstructured interview and UREM.

	Rep. 1	**Rep. 2**	**Both-rep.**
p-value Method	**.018**	.489	.220
p-value Method*Problem	-	-	**.021**
p-value Method*Replication	-	-	**.021**
Effect size	-	-	-
Power	-	-	.230

Table 7. Statistical results of efficiency

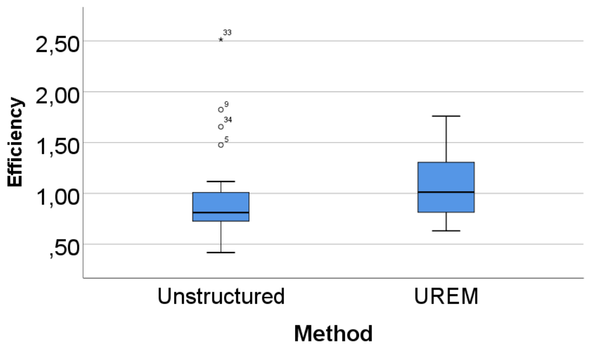
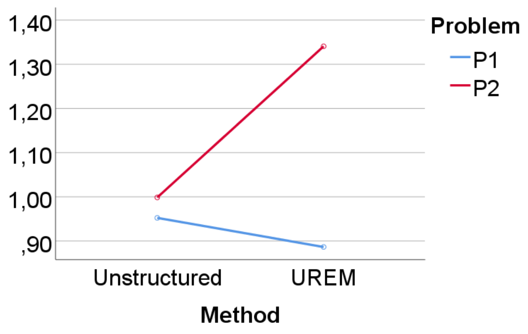


Figure 6. Box plot of efficiency



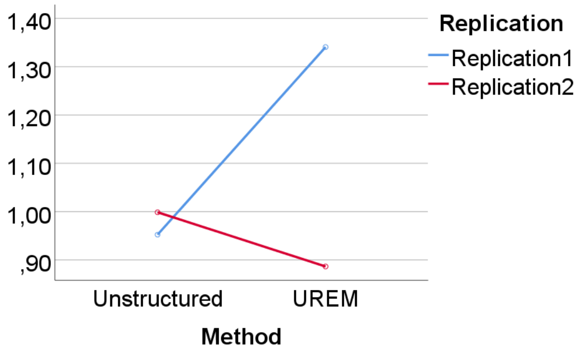


Figure 7. a) profile plot of Method*Problem. b) profile plot of Method*Replication

5.4 Analyst Satisfaction

Analyst satisfaction was measured using three different metrics: Perceived Usefulness (PU), Perceived Ease of Use (PEOU), and Intention to Use (ITU). When analyzing the p-values of each replication separately (Table 8), only PEOU yielded significant results in Replication 1 (p-value was .028). The average in this case was 16 for the unstructured interview and 13.63 for UREM, so the subjects perceived the unstructured interview being as easier to use. The other averages were: PU in Replication 1: 30.18 in the unstructured interview and 25.9 in UREM; ITU in Replication 1: 10.81 in the unstructured interview and 9.81 in UREM; PU in Replication 2: 29.46 in the unstructured interview and 28.76 in UREM; PEOU in Replication 2: 15.07 in the unstructured interview and 14.69 in UREM; ITU in

Replication 2: 10.15 in the unstructured interview and 10.23 in UREM. Note that most of the results yielded slightly better satisfaction for the unstructured interview, but this difference was not significant.

Figure show the box plot of the two replications together for PU, PEOU, and ITU, respectively. PU and ITU yielded the same median for both treatments. In the case of PEOU, the median was slightly better for the unstructured interview. For the three metrics (PU, PEOU, and ITU), the third quartile was very similar for both treatments, but the first quartile was better for the unstructured interview. The statistical test of the GLM did not yield significant differences for any metric (all p-values were higher than .05) and there were no differences for Method*Problem and Method*Replication interactions. The statistical power was low in the three metrics, so significant differences may appear in a larger sample size.

	Rep. 1	**Rep. 2**	**Both rep.**
p-value Method	.065	1	.128
p-value Method*Problem	-	-	.434
p-value Method*Replication	-	-	.434
Effect size	-	-	-
Power	-	-	.330

Table 8. Statistical results of PU

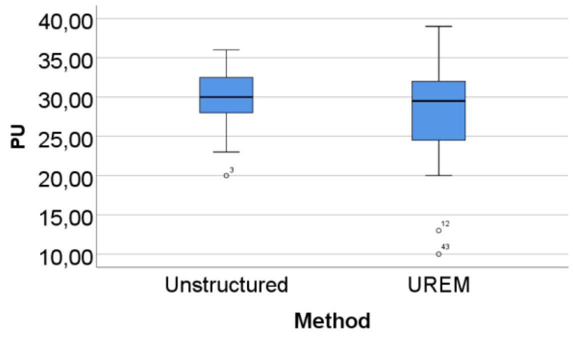


Figure 8. Box plot of PU

	Rep. 1	Rep. 2	Both rep.
p-value Method	**.028**	1	.141
p-value Method*Problem	-	-	.561
p-value Method*Replication	-	-	.561
Effect size	-	-	-
Power	-	-	.311

Table 9. Statistical results of PEOU

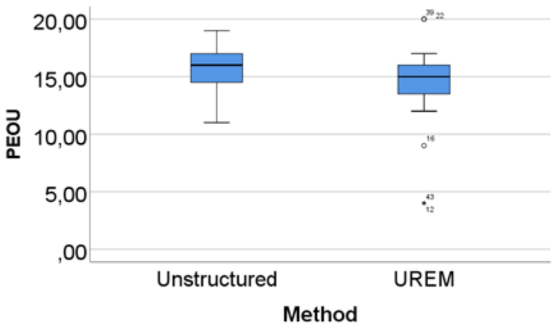


Figure 9. Box plot of efficiency

	Rep. 1	Rep. 2	Both rep.
p-value Method	.193	.579	.429
p-value Method*Problem	-	-	.636
p-value Method*Replication	-	-	.636
Effect size	-	-	-
Power	-	-	.122

Table 10. Statistical results of ITU

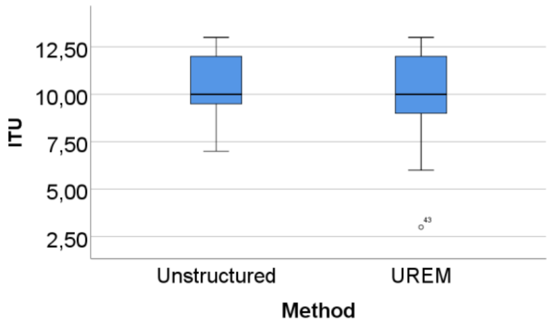


Figure 10. Box plot of efficiency

In conclusion, **we can only reject H_{03a}** (The analyst satisfaction using UREM is similar to that of using unstructured interviews) **for the metric PEOU in Replication 1**, where the unstructured interview yields a better satisfaction level. The other metrics did not present significant differences in each replication separately or together.

5.5 End User Satisfaction

End user satisfaction is measured using two metrics: the CSUQ questionnaire and the satisfaction of the end user with the recommendation offered by the analyst to improve usability. The p-values of each replication individually were higher than .05 (Table 19 and Table 11), so there were no significant differences between treatments in any replication. The average of CSUQ in Replication 1 was 70.72 for the unstructured interview and 75.81 for UREM. In Replication 2 the average was 78.23 for the unstructured interview and 66.46 for UREM. The median of satisfaction with the recommendations to improve the usability in Replication 1 was 4 for both the unstructured interview and UREM. In Replication 2, it was also 4 for both the unstructured interview and UREM. All of this descriptive data does not yield any conclusion in the differences between the two treatments.

Figure 1119 show the box plot of the two replications together for the CSUQ questionnaire and the end user satisfaction with the recommendations to improve usability. The medians in both plots were similar. The first quartile was slightly better for the unstructured interview in both metrics. The third quartile was better for the unstructured interview in the CSUQ metric, while the third quartile does not present differences in the metric of satisfaction with the recommendations. The statistical test did not yield significant differences for any metric (all p-values were higher than .05), and there were no differences for Method*Problem and Method*Replication interactions.

In conclusion, **we cannot reject H_{03e}** (the end user satisfaction using UREM is similar to that of using unstructured interviews), so there were no differences between treatments in terms of satisfaction with the recommendations to improve usability. Table 21 summarizes the results of the statistical tests for all of the hypotheses.

	Rep. 1	**Rep. 2**	**Both rep.**
p-value Method	.151	.153	.426
p-value Method*Problem	-	-	.136
p-value Method*Replication	-	-	.136
Effect size	-	-	-
Power	-	-	.123

Table 19. Statistical results of CSUQ questionnaire

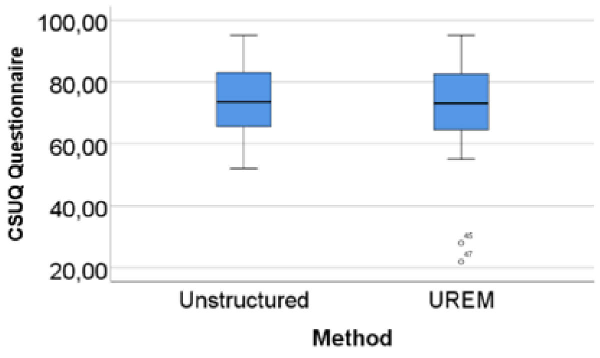


Figure 1119. Box plot of CSUQ questionnaire

	Rep. 1	Rep. 2	Both rep.
p-value Method	.562	.287	.504
p-value Method*Problem	-	-	.396
p-value Method*Replication	-	-	.396
Effect size	-	-	-
Power	-	-	.101

Table 11. Statistical results of end user satisfaction with the recommendations

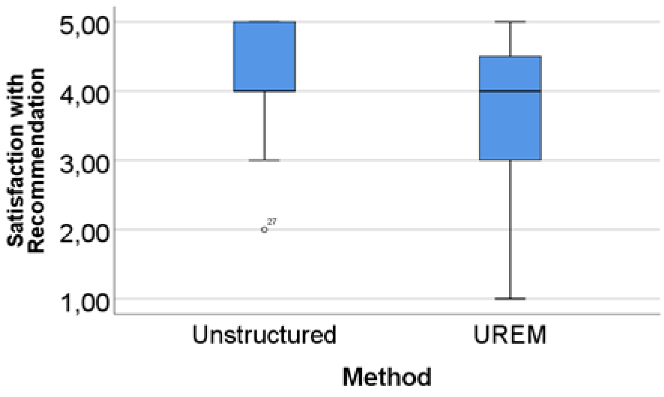


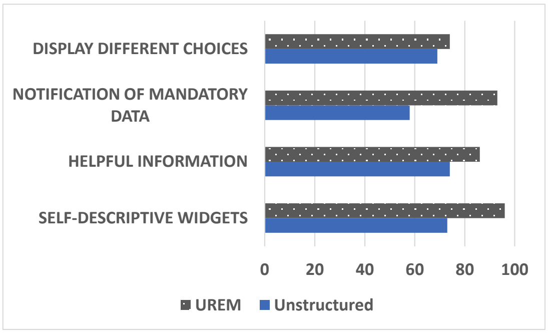
Figure 12. Box plot of end user satisfaction with the recommendations

Hypotheses	Results
H_{01r}	Effectiveness of usability requirements elicitation is **significantly** better for UREM
H_{01g}	Effectiveness of usability guidelines is **significantly** better for UREM
H_{02r}	Efficiency for usability requirements elicitation is **the same** for UREM and the unstructured interview
H_{03a}	Analyst Satisfaction is the **same** for UREM and the unstructured interview
H_{03e}	End user Satisfaction is the same for UREM and the unstructured interview

Table 12. Summary of the results.

5.6 Usability Requirements Problems and Usability Guidelines Compliance

Next, we describe the actual results in terms of usability requirements problems and level of compliance with usability guidelines found during the experimentation. Figure 13.a and b show the percentage of usability requirements used in the experiment that are successfully elicited in P1 and P2 respectively. These requirements were defined in Table 5 and Table and used to measure the response variable Effectiveness for usability requirements elicitation. Both plots show that UREM obtains a better percentage than the Unstructured method. If we focus on UREM for P1, the lowest effectiveness is for “Display different choices” since several prototypes did not show all the menu options by default. “Helpful information” is around 85% since most prototypes included helpful information to describe the options and actions that each interface offers. “Notification of mandatory data” and "Self-descriptive widgets” are close to 100%. Almost all interfaces included self-descriptive widgets and identified the mandatory widgets to fill in. If we focus on UREM for P2, the lowest level is for “Avoid errors”. A few interfaces did not include a list of enumerated options to avoid errors. “Flexibility to activate/deactivate” is around 85%, which means that most interfaces included options to modify the default options; for example, the date of today, or your current position to look for the closest bank to extract money. “Help to fix errors” and “Notification of mandatory data” are close to 100%. Most interfaces included messages to guide the end-user when an error arises, and mandatory data is clearly identified in the interfaces. Note that, even though the requirements are the same for both P1 and P2, UREM yields better effectiveness in the usability requirements elicitation.



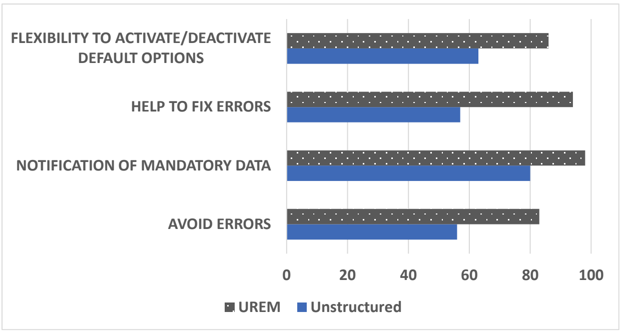


Figure 13. a) Percentage of usability requirements correctly elicited in P1.
b) Percentage of usability requirements correctly elicited in P2

Figure 14 shows the percentage of usability guidelines that are satisfied in P1. These usability guidelines are the ones used to build the tree structure used in the experiment (Appendix B). The percentage of agreement with usability guidelines is used in the experiment to measure the response variable Effectiveness of usability guidelines. Note that there is a large difference between UREM and Unstructured method for “Use a dialogbox to show error message”, “Use asterisk for mandatory fields”, “Use alternative text for textfields”, and “Use dropdown for a menu with several options”. In the Unstructured method, most prototypes did not specify the mechanisms to notify about errors. Moreover, they used the red color or a bold font to highlight the mandatory data (instead of an asterisk). Almost no interface used alternative text for textfields. Menus with several options were designed mainly with a list (instead of a dropdown). The level of agreement with usability guidelines improves when using UREM. All the guidelines are larger than 65% except for "Use dropdown for the menu with several options". Even though the tree structure recommended the use of a dropdown, several clients preferred a design with all the items in the interface without a dropdown.

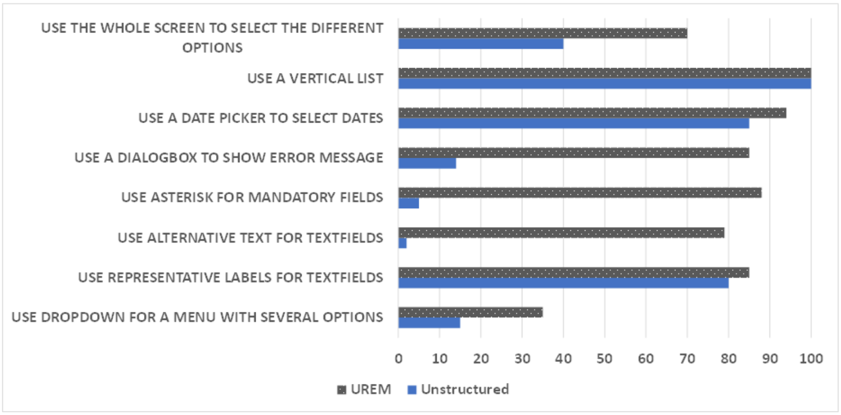


Figure 14. Percentage of usability guidelines satisfied in P1

Figure 15 shows the percentage of usability guidelines satisfied in P2 both with UREM and with the Unstructured method. Note that there are usability guidelines around 0% with the Unstructured method: "Use text and icon for help actions", "Use a dialogbox to show error message", and "Use alternative text for textfields". Even though many subjects used text to describe actions, a few of them complemented the text with an icon. Moreover, as in P1, a few prototypes included dialogboxes to show errors messages and a few prototypes used alternative text for textfields. The guidelines "Use asterisk for mandatory fields" and "Use dropdown for a menu with several options" show a value of around 20%. This is because mandatory fields are represented in red color or bold and menus with several options are displayed with items without dropdown. On the contrary, some guidelines are very similar between

UREM and the Unstructured method: “Use the whole screen to select the different options”, and “Use a vertical list”. Subjects tend to use all the size of the screen to design the interface, and lists are always shown in vertically. If we analyze the results for UREM, all values of agreement with usability guidelines improve. The only guideline that is below 65% is “Use dropdown for a menu with several options”. This shows that even though UREM recommends usability guidelines, the results of the design are not 100% compliant with usability guidelines. The client chooses between applying the usability guidelines or any other alternative she/he prefers.

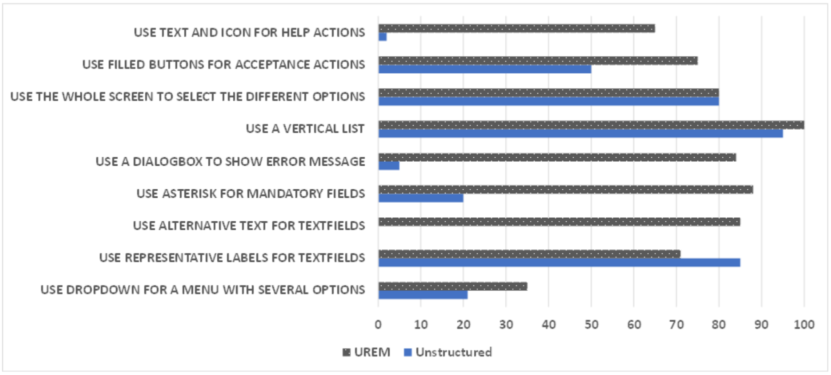


Figure 15. Percentage of usability guidelines satisfied in P2

6. Discussion

This section discusses the results, looking for justifications for the data and comparing the outcomes with previous existing empirical works. We analyze the results for each hypothesis. H01r yields significant differences, where UREM presents better effectiveness in the requirements elicitation process. Since effectiveness is defined as the percentage of usability requirements successfully elicited, this means that working with UREM helps the analyst identify successfully more usability requirements than an unstructured interview does. These differences arise in Replication 1 and when both replications are aggregated, but it does not appear in Replication 2. This may be due to the low sample size if we analyze replications individually. The descriptive data in Replication 2 shows a trend of more effectiveness of UREM than the unstructured interviews. Note that the previous experience of the subjects was mainly in unstructured interviews (Table 7), and only two subjects had experience in structured interviews. Even though the experience in the two treatments is so unbalanced, the effectiveness with UREM (a structured method) is clearly better when a short training is provided before the experiment. This result aligns with previous works in the literature, which state that structured interviews are the most effective elicitation techniques in a wide range of domains and situations [40-41].

H01g also yields significant differences, where UREM shows better effectiveness applying usability guidelines. This means that analysts working with UREM are more compliant with usability guidelines than analysts working with the unstructured interview. Note that the use of UREM does not ensure the support of usability guidelines in the GUI designs. UREM suggests which design alternative is the one that best fits the usability requirements. However, the choice of the final design depends on the agreement between the analyst and the end user, and this choice may be different from the one suggested by UREM. Based on these results, we can state that most analysts agreed to accept the suggestions of the UREM method to improve usability. Median for the effectiveness of usability guidelines (Figure 5) is 70%. This means that even using UREM, some subjects did not follow the usability suggestions. Note that the subjects that were recruited in the experiment had experience in the requirements elicitation process but only half of them had experience with usability (Table 8). Even though their experience in usability is not high, the designed GUI are compliant with the usability guidelines. This means that UREM helps design usable interfaces even when the analyst is not an expert in usability guidelines. There are previous works that have classified the different usability guidelines, reporting advantages and describing how to deal with the guidelines [42]. To our knowledge, there are no previous works that structure the information of the guidelines in a tree structure as a helping guide during the requirements elicitation process. UREM provides a clear contribution to the field of usability guidelines assistance.

H02r does not yield significant differences between UREM and the unstructured interview. Differences only appear in Replication 1. Moreover, if we analyze the descriptive data after aggregating both replications, we see that the averages are very similar between UREM and the unstructured interview. This means that, even though the use of UREM could lead to an increase in the required time, the data shows that this increase in time is not real. The efficiency needed to navigate throughout the tree structure is the same as the efficiency needed to conduct an unstructured interview. This conclusion may be biased by the size of the tree, but, in our experiment, we are not working with a small tree. This may reduce the effort required by the analyst for the navigation. The whole tree is shown in Appendix E. This result contradicts the conclusions of other previous works, which state that structured interviews such as JAD require more effort than unstructured ones such as Brainstorming [43]. The statistical power is low, so to be completely sure that significant differences in terms of efficiency do not arise between the two treatments, we need a larger sample size. In this hypothesis, we identified two interactions as being significant: Method*Problem and Method*Replication. The differences between UREM and the unstructured interview are more evident in P2 (bank) than in P1 (health center). UREM required more time in P1, which reduced the efficiency. The subjects who were recruited for the experiment may have had more experience in interaction with banking systems, so the effort spent for each treatment was low in this problem because the analysts could have had a possible prototype in mind for this type of system. A health center application is usually used with less frequency than a banking application. This may have led to requiring more effort to elicit the requirements, which may highlight the difference in efficiency between the treatments. With regard to the Method*Replication interaction, the difference between treatments is more evident in Replication 1. This could be due to the profile of the subjects of that replication; they are undergraduate students with low experience in software development companies (Table 6). This result together with the significant result for efficiency in Replication 1 leads to thinking that UREM shows a better efficiency in a context with low professional experience.

H03a yields significant differences for the PEOU metric in Replication 1. When analyzing the box plot of the two replications together, there is a trend where the unstructured interview obtains a better satisfaction. The low power may justify that this significant difference is not present when the two replications are aggregated together. Since the significant result focuses only on one replication, general conclusions cannot be drawn. Note that most of the subjects have experience in the area of software development (Table 8), and they have a good background with unstructured interviews (Table 10). Despite this advantage for the unstructured interview compared with UREM, the subjects do not have a clear preference for either method. To the authors knowledge, there are no previous works that have experimentally evaluated how the structured interviews may affect the analysts' satisfaction. This lack of empirical works may be because satisfaction is a broad term with several perspectives. For example, the work of Elrakaiby et al.[44] states that satisfaction depends on motivation, relevance of the realization, and relevance of the statement,. All of these characteristics are difficult to control in an empirical evaluation.

H03e does not yield significant differences between UREM and the unstructured interview. This means that from the point of view of the end user, there is no difference between the two treatments. Even though the usability requirements are elicited with more effectiveness using UREM, the end users are no more satisfied with the designed GUI. Previous works in the literature state that there is a relationship between usability features supported by the system and end user satisfaction [45]. Note that the statistical power is very low in both metrics that analyze the hypothesis; it is possible that some significant differences may arise with a larger sample size. Moreover, the designed GUI are only some parts of the system; the analysts did not design the whole system. An experiment involving more types of interfaces with more complexity might help to find differences between the treatments. We plan to replicate the experiment with a larger sample size and with more complex problems in order to analyze in detail how the use of UREM affects the end user's satisfaction.

As conclusions of our analysis, we can state that UREM helps to improve the effectiveness of the usability requirements elicitation process. Moreover, UREM helps the inclusion of usability guidelines in designs even though the analysts that make the design are not experts in usability. These advantages do not involve a loss of efficiency in the requirements elicitation process and GUI design.

7. Threats to Validity

We have classified the threats to validity of our experiment based on the classification provided by Wohlin [46]. We described each type of threat as: avoided, incurred, and mitigated.

Conclusion validity. This threat is concerned with issues that affect the ability to draw the correct conclusions about relationships between the treatment and the outcome. Threats of this type are: 1) *Low statistical power:* This appears when the sample size is low. After the aggregation of both replications, we obtain enough statistical power for response variables that are related to effectiveness. However, efficiency, analyst satisfaction and end user satisfaction is affected by this threat due to low power. 2) *Violated assumptions of statistical tests*: GLM has some assumptions that must be satisfied in order to conduct the test. We avoided this threat since the aggregation of both replications satisfies all of these assumptions. 3) *Fishing:* This appears when experimenters are looking for a specific result. Even though one experimenter was the designer of UREM, the other two experimenters that participated in the design and interpretation of the results were not the authors of UREM. Therefore, this threat was mitigated. 4) *Reliability of measures*: This appears when measures have errors due to problems with instruments. We mitigated this threat by conducting a pilot study with two subjects before conducting the real experiment. This helped to check all of the experimental artefacts. 5) *Reliability of treatment implementation:* There is a risk that the implementation is not similar between different replications. We mitigated this threat since the experimenter who described the treatments and conducted the experiment was the same in both replications. It is also possible that end users describe the usability requirements wrongly, and this may affect RQ1r and RQ1g. This is mitigated because both treatments suffer this threat, so it should not affect positively or negatively a specific treatment. 6) *Random heterogeneity of subjects:* This appears when the sample size is too heterogeneous, and this variation is larger than the variation produced by the treatment. Subjects of R2 (Master's students) have more job experience than subjects of R1 (undergraduate students). Since we analyze each replication individually, we can analyze whether or not there are differences between both profiles.

Internal validity. This threat is concerned with influences that may affect the dependent variable with respect to a causality which the researchers are unaware of. Threats of this type that may appear are: 1) *History*: This appears when the treatments are applied at different moments. Our experiment was affected since unstructured interviews and UREM are applied in different sessions. Even though we tried to maintain the same context and conditions, we cannot ensure that the different moment of each session did not affect the results. 2) *Maturation*: This appears when the subjects react differently as time pass. We mitigated this threat by conducting each session in a maximum of one hour. This was to avoid boredom and fatigue. 3) *Instrumentation*: This appears when the instruments used in the experiment may affect the results. This threat was mitigated since the satisfaction questionnaires were validated previously. The analyst satisfaction questionnaire is based on the TAM by Davis [60] while the end user satisfaction is based on the CSUQ [59]. 4) *Selection*: How the subjects are recruited may affect the results. In our experiment, the participants participated as part of a course. The participation in the experiment was not mandatory, but it gave the participants extra credit in the course. This may lead to subjects being over motivated, which may result in a threat. 5) *Mortality*: This appears when the subjects abandon the experiment before finishing. We avoided this threat since no subject left the experiment. 6) *Compensatory rivalry:* This appears when the subjects receive different treatments. We avoided this threat since all of the subjects received both treatments and all of the subjects played both roles (analyst and end user). 7) *Differences between roles*: playing the role of the analyst can be easier than playing the role of the end-user. When subjects play the role of the analyst, they act with the role that their course is preparing for. This may lead to more motivated subjects when they play the role of the analyst. We have mitigated this threat by swapping the roles between both treatments.

Construct validity. This threat is concerned with generalizing the results of the experiment to the concept or theory behind the experiment. Threats of this type that our family of experiments may be open to are: 1) *Inadequate preoperational explication of constructs*: This appears when the theory behind the treatment has not been sufficiently defined. We avoided this threat since the UREM method had a proper definition before conducting the experiment. 2) *Mono-operation bias*: This appears when experiments with only one factor may under-represent the construct. We mitigated this threat by analyzing the interaction of the method with the problem and the replication. This was to look for differences due to context or problem complexity. 3) *Mono-method bias*: This appears when a simple type of metrics is used. We mitigated this threat since the analyst satisfaction and end user satisfaction depend on more than one metric. However, the effectiveness of usability requirements elicitation, the effectiveness of usability guidelines, and efficiency were affected by this threat. 4) *Problem homogeneity*: This appears when experimental problems are too homogeneous to generalize the results to other problems. We mitigated this threat by choosing problems from different domains.

External validity. This threat is concerned with conditions that limit the ability to generalize the results of experiments to industrial practice. Threats of this type are: 1) *Interaction of selection and treatment*: This appears when the subjects are not representative of the population that we want to generalize. We mitigated this threat since, even though the subjects were students, they had previous experience in real software development projects. 2) *Interaction of setting and treatment*: This appears when the experimental setting or the material are not representative of our target of study. We mitigated this threat since the usability requirements and the problems were aligned with the context where UREM is used. 3) *Interaction of history and treatment*: this appears when the experiment is conducted at a special time that may affect the results. Our experiment was affected by this threat since each replication was conducted on different days. 4) *Interaction between research questions:* this appears when there is a correlation between research questions. The experiment suffers this threat since RQ2r might be somehow correlated to RQ1r. The fewer usability requirements satisfied by the analyst, the shorter the time required to define them.

8. Conclusions

This article presents an empirical experiment that compares structured interviews with unstructured interviews in order to elicit usability requirements. Structured interviews are operationalized as UREM, which is a method based on a decision tree where the analyst guides the interview by navigating throughout the tree structure. Each branch of the tree includes a question for the end user with possible answers. Moreover, the answer that is more compliant with existing usability guidelines is recommended. In the unstructured interview method, the analyst must elicit usability requirements without any guide. In this work, this control treatment is referred to as unstructured interview. The evaluation is conducted to analyze four response variables: effectiveness in the usability requirements elicitation; effectiveness in the application of usability guidelines; efficiency; the analyst's satisfaction; the end user's satisfaction. As significant results, UREM is more effective in the usability requirements elicitation and also more effective in designing interfaces that are compliant with usability guidelines.

Note that even though the recruited subjects are students, a large percentage of them have experience in real software development companies. Therefore, the results could be generalizable to any person with some type of experience in software development, not just students. The experiment was conducted with two different problems, so the results are not associated to a single problem. This also facilitates the generalization of results.

Some lessons have been learned during the conduction of the experiment: 1) The effort to build the tree in UREM is high. This is something that was not analyzed in the experiment, but the required effort is not null. Note that this effort can be recovered; the same tree structure is useful for any future development; 2) The recommendations during the tree structure navigation may be different depending on the usability guidelines used to build the tree. Even though most usability guidelines agree on the characteristics that optimize usability, there are some guidelines that may present some contradictions. In the end, the expert at usability that builds the tree structure is the one who chooses the most suitable usability guidelines for the recommendations; 3) Most of the end users accepted the usability recommendations. This value may have been different if the subjects had had more experience in usability characteristics. Other experiments can be conducted to determine how the level of experience may affect the results. 4) Due to the structure of questions, UREM may leave no room for discovering designs not included as alternatives in the tree structure.

As future work, we plan to replicate the experiment in order to enhance the sample size. Some response variables such as the analyst' satisfaction and the end user' satisfaction have a low statistical power. With a larger sample size we may be able to identify more significant differences for these response variables. Moreover, we aim to analyze more factors, such as previous experience in usability concepts and the complexity of the problems. In a future validation of UREM, we plan to include other metrics such as creativity when the tree structure is built and when it is used in the interviews; qualitative analysis of how designers perceive the use of UREM; need of training for the method; overall appreciation of the guidance provided; reusability in multiple contexts of use; perception of the time and effort necessary to prepare the tree structure; and flexibility to run the method. We also plan to compare UREM with other structured interview methods.

Acknowledgements

This work was developed with the support of the National University of San Antonio Abad of Cusco under the program Yachayninchis Wiñarinanpaq CONCYTEC and FONDECYT, the support of Generalitat Valenciana with CoMoDID (CIPROM/2021/023) and GENI (CIAICO/2022/229), as well as the support of the Spanish Ministry of Science and Innovation co-financed by FEDER in the project SREC (PID2021-123824OB-I00)

References

1. M. Rajanen and N. Livari, "Usability cost-benefit analysis: How usability became a curse word?," pp. 511-524, 2007.
2. D. Quiñones, C. Rusu, and V. Rusu, "A methodology to develop usability/user experience heuristics," *Computer standards & interfaces,* vol. 59, pp. 109-129, 2018.
3. ISO, *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs): Part 11: Guidance on usability*, 1998.
4. ISO/IEC, "ISO / IEC 25010 : 2011 Systems and software engineering@ Systems and software Quality Requirements and Evaluation (SQuaRE)@ System and software quality models," 2013.
5. H. A. Hutahaean, R. Govindaraju, and I. Sudirman, "Identifying Usability Risks for Mobile Application," in Proceedings of the International Conference on Engineering and Information Technology for Sustainable Industry, Tangerang, Indonesia, pp. 1-6, 2021.
6. E. M. Rey, V. M. Bonillo, and D. A. Ríos, "Session details: Theme: Software design and development: UE - Usability engineering track," in Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 2019.
7. Y. I. Ormeño, J. I. Panach, N. Condori-Fernández, and Ó. Pastor, "Towards a proposal to capture usability requirements through guidelines," in Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science (RCIS), pp. 1-12, 2013.
8. J. Nielsen, *Usability Engineering*: Morgan Kaufmann, 1993.
9. M. J. Muller, "Participatory design: the third space in HCI," in *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, ed: L. Erlbaum Associates Inc., pp. 1051–1068, 2002.
10. K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *EASE*, pp. 68–77, 2008.
11. F. Gunduz and A. S. K. Pathan, "Usability improvements for touch-screen mobile flight booking application: A case study," in Proceedings of the International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2012, pp. 49-54, 2012.

12. O. D. Troyer and E. Janssens, "A feature modeling approach for domain-specific requirement elicitation," in Proceedings of the IEEE 4th International Workshop on Requirements Patterns (RePa), pp. 17-24, 2014.
13. P. Fahey, C. Harney, S. Kesavan, A. McMahon, L. McQuaid, and B. Kane, "Human computer interaction issues in eliciting user requirements for an Electronic Patient Record with multiple users," in Proceedings of the 24th International Symposium on Computer-Based Medical Systems (CBMS), pp. 1-6, 2011.
14. M. Temper, S. Tjoa, and M. Kaiser, "Touch to authenticate—Continuous biometric authentication on mobile devices," in Proceedings of the 1st International Conference on Software Security and Assurance (ICSSA*)*, pp. 30-35, 2015.
15. T. Rocha Silva, M. Winckler, and C. Bach, "Evaluating the usage of predefined interactive behaviors for writing user stories: an empirical study with potential product owners," *Cognition, Technology & Work,* vol. 22, pp. 437-457, 2020.
16. E. A. De Carvalho, A. Jatobá, and P. V. R. De Carvalho, "Usability for complex systems?: An experimental evaluation with functional resonance analysis method," in Proceedings of the *18th Brazilian Symposium on Human Factors in Computing Systems (IHC),* pp. 1-4, 2019.
17. J. A. Nhavoto, Å. Grönlund, and W. P. Chaquilla, "SMSaúde: Design, development, and implementation of a remote/mobile patient management system to improve retention in care for HIV/aids and tuberculosis patients," *JMIR mHealth and uHealth,* vol. 3, 2015.
18. E. Elias, D. Miquilino, I. I. Bittencourt, T. Tenório, R. Ferreira, A. Silva, S. Isotani, and P. Jaques, "Towards an ontology-based system to improve usability in collaborative learning environments," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* vol. 7315 LNCS, ed, 2012, pp. 298-303.
19. X. Yuan and X. Zhang, "An ontology-based requirement modeling for interactive software customization," in Proceedings of the IEEE International Model-Driven Requirements Engineering Workshop (MoDRE), pp. 1-10, 2015.
20. Z. S. H. Abad, S. Moazzam, C. Lo, T. Lan, E. Frroku, and H. Kim, "Loud and Interactive Paper Prototyping in Requirements Elicitation: What is it Good for?," in Proceedings of the IEEE 7th International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 16-23, 2018.

21. G. Márquez and C. Taramasco, "Using Dissemination and Implementation Strategies to Evaluate Requirement Elicitation Guidelines: A Case Study in a Bed Management System," *IEEE Access*, vol. 8, pp. 145787-145802, 2020.
22. S. Tiwari, S. S. Rathore, and A. Gupta, "Selecting requirement elicitation techniques for software projects," pp. 1-10, 2012.
23. A. Abdallah, R. Hassan, and M. A. Azim, "Quantified extreme scenario based design approach," in Proceedings of the ACM Symposium on Applied Computing, pp. 1117-1122, 2013.
24. G. Vitiello, R. Francese, M. Sebillo, G. Tortora, and M. Tucci, "UX-requirements for patient's empowerment - The case of multiple pharmacological treatments: A case study of it support to chronic disease management," in Proceedings of the IEEE 25th International Requirements Engineering Conference Workshops, REW 2017, pp. 139-145, 2017.
25. Y. Tanikawa, R. Okubo, and S. Fukuzumi, "Process support method for improved user experience," *NEC Technical Journal*, vol. 8, pp. 28-32, 2014.
26. Z. S. H. Abad, S. D. V. Sims, A. Cheema, M. B. Nasir, and P. Harisinghani, "Learn More, Pay Less! Lessons Learned from Applying the Wizard-of-Oz Technique for Exploring Mobile App Requirements," in Proceedings of the IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 132-138, 2017.
27. M. Peruzzini and M. Germani, "Designing a user-centred ICT platform for active aging," in Proceedings of the IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA), pp. 1-6, 2014.
28. H. Takeshi and F. Shin'ichi, "Applying human-centered design process to SystemDirector Enterprise development methodology," *NEC Technical Journal*, vol. 3, pp. 12-16, 2008.
29. S. Sharma and S. Pandey, "Revisiting Requirements Elicitation Techniques," *International Journal of Computer Applications*, vol. 75, pp. 35-39, 2013.
30. T. R. Gruber, C. Baudin, J. H. Boose, and J. Webber, "Design Rationale Capture as Knowledge Acquisition," in *ML Workshop*, 1991.
31. C. Martinie, P. Palanque, M. Winckler, and S. Conversy, "DREAMER: a design rationale environment for argumentation, modeling and engineering requirements," in Proceedings of the 28th ACM International Conference on Design of Communication, São Carlos, São Paulo, Brazil, pp. 73–80, 2010.

32. N. Juristo and A. M. Moreno, *Basics of software engineering experimentation*: Springer Science & Business Media, 2013.
33. J. R. Lewis, "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use," *International Journal of Human‐Computer Interaction,* vol. 7, pp. 57-78, 1995.
34. D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, and M. Oivo, "Empirical software engineering experts on the use of students and professionals in experiments," *Empirical Software Engineering,* vol. 23, pp. 452-489, 2018.
35. N. Juristo and A. Moreno, *Basics of Software Engineering Experimentation*: Springer, 2001.
36. Y. Ormeño, J. I. Panach, and Ó. Pastor, "Experimental material of the article "An Empirical Experiment of a Usability Requirements Elicitation Method based on Interviews"," Z. https://doi.org/10.5281/zenodo.7646554, 2023.
37. L. S. Meyers, "Applied multivariate research : design and interpretation," G. Gamst and A. J. Guarino, Eds. Thousand Oaks : Sage Publications, 2006.
38. L. S. Meyers, G. Gamst, and A. J. Guarino, *Applied multivariate research: Design and interpretation*: Sage publications, 2016.
39. T. Dybå, V. B. Kampenes, and D. I. Sjøberg, "A systematic review of statistical power in software engineering experiments," *Information and Software Technology,* vol. 48, pp. 745-755, 2006.
40. A. M. Davis, Ó. D. Tubío, A. M. Hickey, N. J. Juzgado, and A. M. Moreno, "Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review," in Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), pp. 179-188, 2006.
41. N. Bahurmuz, R. Alnajim, R. Al-Mutairi, Z. Al-Shingiti, F. Saleem, and B. Fakieh, "Requirements Elicitation Techniques in Mobile Applications: A Systematic Literature Review," *International Journal of Information Technology Project Management (IJITPM),* vol. 12, pp. 1-18, 2021.
42. M. S. Goundar, B. A. Kumar, and A. B. M. S. Ali, "Development of Usability Guidelines: A Systematic Literature Review," *International Journal of Human–Computer Interaction,* pp. 1-19, 2022
43. .O. Okesola, K. Okokpujie, R. Goddy-Worlu, A. Ogunbanwo, and O. Iheanetu, "Qualitative comparisons of elicitation techniques in requirement engineering," *Journal of Engineering and Applied Sciences,* vol. 14, pp. 565-570, 2019.

44. Y. Elrakaiby, A. Ferrari, P. Spoletini, S. Gnesi, and B. Nuseibeh, "Using Argumentation to Explain Ambiguity in Requirements Elicitation Interviews," in Proceedings of the IEEE 25th International Requirements Engineering Conference (RE), pp. 51-60, 2017.
45. J. M. Ferreira, S. T. Acuña, O. Dieste, S. Vegas, A. Santos, F. Rodríguez, and N. Juristo, "Impact of usability mechanisms: An experiment on efficiency, effectiveness and user satisfaction," *Information and Software Technology,* vol. 117, p. 106195, 2020.
46. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*: Springer Science & Business Media, 2012.
47. F. D. Davis, "User acceptance of information technology: system characteristics, user perceptions and behavioral impacts," *International journal of man-machine studies,* vol. 38, pp. 475-487, 1993.

PARTE III

DISCUSIONES DE LOS RESULTADOS

III

Los temas que cubre esta parte son:

En esta parte de la tesis, se presentan los resultados de este trabajo, conectando las preguntas de investigación planteadas al inicio del trabajo con los resultados plasmados en los artículos de investigación recogidos en las cuatro secciones anteriores de la parte II.

Cada uno de estos artículos intenta investigar y responder a las preguntas y sub preguntas de investigación de la tesis.

En el primer artículo que conforma esta tesis, se ha tratado de responder a la siguiente pregunta de investigación RQ1: ¿Es posible capturar requisitos de usabilidad en etapas iniciales de desarrollo software? y la sub pregunta de investigación SQ1.1: ¿Qué guías de usabilidad, estándares y normas se requieren en el proceso de captura de requisitos de usabilidad que apoyen la labor del analista?

En relación a la RQ1, la elicitación de los requisitos de usabilidad generalmente se realiza en la etapa de análisis [46], [15], después que se hayan capturado todos los requisitos funcionales. Esta captura tardía podría ocasionar cambios en la arquitectura del sistema debido a que algunos requisitos de usabilidad están relacionados con la funcionalidad [5], [20]. Por lo general, los métodos utilizados para elicitar los requisitos de usabilidad tratan la usabilidad mediante técnicas tradicionales (e.g. entrevistas, cuestionarios, grupos focales, casos de uso) [35], [3]. El análisis de resultados de la revisión sistemática muestra que existen muy pocas publicaciones que aborden claramente cómo realizar el proceso de captura de requisitos de usabilidad en etapas tempranas. Además, los enfoques existentes no proponen una notación precisa e inequívoca para representar estos requisitos, lo que dificulta su aplicación en sistemas reales. Hay algunas publicaciones donde la elicitación de requisitos de usabilidad se realiza en la etapa de diseño junto con la elicitación de requisitos de interacción [25], [45], [24].

En relación a la SQ1.1, cuando el tema de la usabilidad se trata en la elicitación de requisitos, las normas ISO se utilizan como directrices para ser aplicadas en los sistemas de desarrollo de software. Por ejemplo, la norma ISO 9241-11 se considera una referencia básica para algunos profesionales, investigadores y diseñadores [25]. Para cualquier tipo de requisitos se utiliza la norma ISO 9126-1 [32]. La aplicación de lineamientos es necesaria, pero no suficiente; el principal problema es la correcta aplicación y completa comprensión por parte del usuario final. Las guías solo se construyen de manera general, pero no son un soporte total para el desarrollo de sistemas usables.

Hay algunas propuestas que tienen como objetivo ayudar a los ingenieros de requisitos a abordar los requisitos de usabilidad desde las primeras etapas por medio de reglas GUIDE [22] y un catálogo basado en el marco i* [10]. Ambas técnicas son específicas del contexto, aunque GUIDE utiliza un repositorio basado en casos para tomar decisiones e i* framework recopila una gran cantidad de conocimiento para lograr los objetivos de usabilidad. Otro aspecto que se observa en las publicaciones seleccionadas es el uso de artefactos, tales como: patrones, escenarios y plantillas, que se utilizan con frecuencia como soporte de métodos para elicitar requisitos de usabilidad y requisitos de interacción [6], [48], [16]. Los métodos propuestos en las publicaciones seleccionadas son rígidos y requieren un esfuerzo considerable para ser aplicados a contextos diferentes de los contextos en que han sido definidos [22]. Las guías, notaciones y artefactos utilizados en estos métodos están más cerca de obtener características de interacción que características de usabilidad. En general, las guías para la elicitación de requisitos de usabilidad se definen de manera muy genérica para diferentes niveles de abstracción [8].

En el segundo artículo que conforma esta tesis se ha tratado de responder a la siguiente sub pregunta de investigación SQ1.2: ¿Es posible desarrollar una estructura de árbol que facilite el proceso de captura de requisitos en un entorno MDD?

En relación a la SQ.1.2, se debe tomar en cuenta que existen guías de diseño de IU y guías de usabilidad que pueden ser gestionadas mediante una estructura de árbol en apoyo a la captura de requisitos de usabilidad durante el desarrollo de software. Se debe tomar en cuenta que el tamaño de la estructura de árbol aumentará con la cantidad de guías que consideremos. Incluso con pocas guías, el tamaño del árbol es difícil de manejar si no es gestionado por una herramienta que ayude con la definición de la estructura de árbol y con la navegación a través de las ramas. Para simplificar la estructura, se recomienda centrarse solo en el diseño de la interfaz y las guías de usabilidad más utilizadas. Como parte de trabajo de la tesis, se ha desarrollado la herramienta que implementa UREM, accesible desde http://hci.dsic.upv.es/urem

La asistencia al analista y la reducción del esfuerzo en el proceso de captura de requisitos de usabilidad son aspectos considerados en la evaluación empírica cuando se compara un desarrollo de software que utiliza el enfoque UREM para capturar los requisitos de usabilidad con el desarrollo que no tiene en cuenta estos requisitos (entrevistas no estructuradas). La validación inicial de UREM se hace en un contexto MDD, donde los desarrolladores expertos deben valorar la herramienta UREM dentro de un proceso de desarrollo MDD.

En el tercer artículo que conforma esta tesis, se ha tratado de responder a la SQ1.3: ¿Es posible representar alternativas de diseño de IU en una estructura de árbol en base a las guías de usabilidad y diseño para la captura de requisitos de usabilidad? Los nodos hoja del árbol a los que llega durante la entrevista con el cliente son los diseños de IU seleccionados por el usuario final. Esta selección puede incluir o no las recomendaciones de usabilidad, dependiendo de las preferencias del usuario. Las alternativas de IU son solo propuestas construidas según los estándares, guías de usabilidad y de diseño para guiar la entrevista de elicitación de requisitos y proponer diseños que optimicen la usabilidad.

En el cuarto artículo que compone esta tesis, se ha tratado de responder a las preguntas de investigación RQ2: ¿Qué impacto produce UREM en la captura de requisitos de usabilidad? y las sub preguntas: SQ2.1 ¿Cuál es el impacto del uso de las guías de usabilidad en el diseño de IU?, SQ2.2 ¿Cuál es el impacto de la aplicación del UREM en un contexto académico? y SQ2.3 ¿Cuál es el impacto de las recomendaciones de usabilidad propuestas por UREM?

En relación a la RQ2, se ha realizado el experimento para validar UREM, que consiste en realizar la captura de requisitos de usabilidad comparando UREM con entrevistas no estructuradas. El experimento se ha realizado en dos réplicas bajo un diseño intra-sujetos Replicación 1 (22 estudiantes de pregrado) y Replicación 2 (26 estudiantes de máster). Se han utilizado dos problemas diferentes Problema 1 (App para un Centro de Salud) y Problema 2 (App para una entidad bancaria) para evitar el efecto “carry over” entre tratamientos. Además de buscar diferencias significativas entre tratamientos, se han buscado diferencias en las interacciones Método*Problema y Método*Replicación b. Todo el análisis estadístico se hizo con el Método Lineal General (GML).

En el experimento, se han refutado las hipótesis nulas de las variables respuesta Efectividad (H_{01r}) referente a la Efectividad en la captura de requisitos de usabilidad y Efectividad (H_{01g}) referente a la Efectividad en el uso de las guías, lo que significa que la efectividad lograda en la obtención de los requisitos y en el uso de las guías con UREM es superior frente a la entrevista no estructurada. Este resultado no se muestra en ambas replicaciones, quizá por el bajo tamaño de la muestra. Por otro lado, no se ha podido refutar la hipótesis nula de la variable respuesta Eficiencia (H_{02r}), referente a la Eficiencia en la captura de requisitos de usabilidad, lo que significa que no se aprecia diferencias significativas. Se aprecia una mejora en la efectividad, pero no en el tiempo, lo que implica que no haya variaciones significativas en la eficiencia. De igual forma no se ha podido refutar la hipótesis nula de la variable respuesta Satisfacción (H_{03e}) referente a la Satisfacción del usuario final y la Satisfacción del analista (H_{03a}), lo que significa que no existe diferencias significativas. Esto puede deberse a que los analistas vienen con una amplia experiencia en entrevista no estructuradas.

En relación a la SQ.2.1, la Efectividad (H_{01g}) referente a la Efectividad en el uso de las guías de usabilidad, arroja diferencias significativas, siendo UREM más efectivo. Es decir, que los analistas que trabajan con UREM cumplen mas con las guías de usabilidad en relación a los analistas que trabajan con entrevistas no estructuradas. El uso de UREM no garantiza la gestión de los requisitos de usabilidad para los diseños de IU, sino que ofrece alternativas que se ajusten a los requisitos de usabilidad. La decisión final sobre optar o no por el diseño de la IU ofrecido, siempre será tomada en acuerdo entre el usuario final y el analista. Por otro lado, se ha observado que los analistas que usan UREM siguen de media el 70% de las recomendaciones de usabilidad que se ofrecen con el método. El otro 30% son otros diseños que ha elegido el usuario, diferentes a los recomendados por las guías de usabilidad.

En relación a la SQ.2.2, la aplicación de UREM a través del experimento, se realizó en el contexto académico con sujetos estudiantes (Replica1, estudiantes de pregrado de último ciclo y la Réplica 2, estudiantes de maestría) de la Universidad Nacional de San Antonio Abad del Cusco – Perú. Todos los sujetos tenían suficiente conocimiento en el campo del desarrollo de software. De los resultados se observa que las variables respuesta como la satisfacción del analista y la satisfacción del usuario tienen un bajo poder estadístico. Esto se debe al tamaño de muestra utilizada en su ejecución. Un aspecto positivo es que la propuesta al ser evaluada dentro del entorno académico conlleva a la identificación de las fortalezas y debilidades del método que serían temas de investigación posterior para la mejora del método y de la herramienta en la elicitación de requisitos de usabilidad.

En relación a la SQ.2.3, el método UREM cuenta con la herramienta que ayuda a garantizar la inclusión de las exigencias de las guías de usabilidad y diseño de IU sen los proyectos de desarrollo software, que contribuyen en la mejora de la calidad. La herramienta está accesible en hci.dsic.upv.es/UREM

PARTE IV

CONCLUSIONES IV

El tema que se cubre en esta parte son las conclusiones a las que se arribó en el trabajo de investigación enmarcados en:

Esta parte presenta las conclusiones finales de la tesis, resumiendo los objetivos, el estudio realizado y los resultados de nuestro trabajo. También se presentan futuras líneas de investigación que pueden contribuir a ampliar estos resultados.

4.1 Contribuciones a partir de los Objetivos

Las contribuciones de la tesis surgen directamente de los objetivos principales de la tesis contenidos en las preguntas de investigación:

1) Objetivo OBJ1 (RQ1): ¿Es posible capturar requisitos de usabilidad en etapas iniciales de desarrollo software? La respuesta a esta pregunta está inmersa en el desarrollo del primer, segundo y tercer artículo como sigue:

   El *primer artículo* presenta un estudio sistemático en relación a la a las propuestas existentes para la captura de requisitos de usabilidad en entornos MDD, la misma que ha sido subdivida en 6 sub preguntas respecto a métodos, guías, notaciones, herramientas y validaciones que contiene las propuestas para capturar requisitos de usabilidad. como resultado de la revisión sistemática. Se seleccionaron un total de 29 publicaciones de un conjunto inicial de 150 publicaciones devueltas por la cadena de búsqueda. Las valoraciones de calidad de las publicaciones se desarrollaron con el fin de contrastar la importancia de las publicaciones seleccionadas, donde el 97% está compuesto por buenas y muy buenas publicaciones. A partir de los resultados del mapeo sistemático, podemos concluir que se evidencia una línea de investigación en el campo de los requisitos de usabilidad.

   La aplicación de los métodos de captura de requisitos de usabilidad facilita un apoyo básico que demandan mucho esfuerzo y tiempo en su gestión y ejecución. Las guías de usabilidad, normas, y estándares son de difícil interpretación por parte del equipo de desarrollo. Se requiere un ingeniero de usabilidad para su correcta interpretación, las notaciones y representaciones utilizadas por las diferentes soluciones son extensiones y adaptaciones de los requisitos funcionales. Las herramientas existentes son limitadas y en general son de soporte para el diseño de las interfaces no tomando en cuenta aspectos de usabilidad.

El *segundo artículo* plantea una primera versión de la estructura en árbol. Se define un metamodelo de la propuesta y un ejemplo ilustrativo.

El *tercer artículo* aborda cómo incorporar la propuesta de UREM en un entorno MDD. Se tiene una primera validación inicial con usuarios expertos en MDD.

2) Objetivo OBJ2 (RQ2): ¿Qué impacto produce UREM en la captura de requisitos de usabilidad? La respuesta a esta pregunta está inmersa en el desarrollo del cuarto artículo, como sigue:

   El *cuarto artículo* es el diseño y ejecución de un experimento para validar UREM comparándolo con entrevistas no estructuradas. El experimento se hace en base a la efectividad. eficiencia, y satisfacción desde el rol usuario o analista según corresponda.

   El impacto de la aplicación del UREM en un contexto académico conlleva a que los resultados podrían ser generalizables a cualquier analista con algún tipo de experiencia en el desarrollo software y no solo estudiantes. Esto se debe a que en el experimento los sujetos que eran estudiantes tenían experiencia en empresas reales de desarrollo de software en un alto porcentaje. Por otro lado, los resultados no han estado asociados a un solo problema, esto también facilita la generalización de los mismos y hace que UREM sea una propuesta que pueda ser utilizada en otros sistemas de igual complejidad.

4.2 Fortalezas y Debilidades de la Tesis

La usabilidad es una de las características esenciales de la calidad software y su proceso de captura debe darse conjuntamente con los requisitos funcionales para garantizar la calidad en proceso y producto del software. Con la presente investigación se logró construir un método al que denominamos UREM que realiza la captura de requisitos de usabilidad.

Los puntos fuertes de UREM son los siguientes:

- Puede ser utilizado por no expertos en usabilidad. La ausencia de expertos en los equipos de desarrollo es muy común debido a la complejidad que presentan las normas ISOs, guías de usabilidad y guías de diseño.
- Presenta una estructura de árbol basado en nodos, ramas y hojas representados en preguntas, respuestas y alternativas. Esta estructura es de fácil comprensión y aprendizaje tanto para el analista como para el usuario final en cuanto a su uso durante el proceso de captura de requisitos de usabilidad.
- La propuesta de UREM está contenida en una herramienta que contiene la estructura de un árbol. El árbol debe ser diseñado por un experto en usabilidad, donde las alternativas de los diseños de IUs contienen aspectos de usabilidad provenientes de las guías de usabilidad y diseño existentes en la literatura.

Dentro de los puntos débiles de UREM identificamos los siguientes:

- Hay que invertir un esfuerzo inicial en la construcción del árbol. Se deben seleccionar las guías de usabilidad y diseño de IUs e introducirlas en la estructura de árbol.
- La aplicación de las recomendaciones de usabilidad propuestas a raíz de las guías de usabilidad depende de las decisiones del usuario durante la entrevista. Esto puede resultar en diseños que no sigan ninguna de las recomendaciones de usabilidad. En estos casos, el diseño sería de la satisfacción del usuario, pero no estaría acorde a las guías de usabilidad.
- Puede haber contradicciones entre guías de usabilidad que impliquen recomendaciones contradictorias en algunos puntos del árbol que deben ser analizados por el analista. Es el usuario final el que debe tomar la decisión de qué diseño elige en estos casos.

4.3 Trabajos Futuros

Durante el desarrollo de la tesis se han identificado varios temas de investigación que podrían abordarse en las próximas investigaciones. El objetivo principal de estos trabajos futuros será superar algunas de las limitaciones del presente trabajo que se ha desarrollado hasta el momento.

- A partir de los diseños alcanzados en los nodos hoja, se pueden utilizar modelos abstractos que representen estos diseños y ser entrada para modelos MDD.
- Implementar otra herramienta colaborativa con varios analistas que apoyen en la construcción y el uso de cualquier estructura de árbol.
- Se pueden realizar otros experimentos en el futuro para aumentar el tamaño de la muestra y poder determinar cómo el nivel de experiencia del analista y la complejidad de los problemas puede afectar a los resultados.
- Comparar UREM con otros métodos de entrevista estructurada.

Referencias

1. Berendes, S., et al., *Evaluating the usability of open source frameworks in energy system modelling.* Renewable and Sustainable Energy Reviews, 2022. **159**: p. 112174.
2. Jeong, J., N. Kim, and H.P. In, *Detecting usability problems in mobile applications on the basis of dissimilarity in user behavior.* International Journal of Human-Computer Studies, 2020. **139**: p. 102364.
3. Calvary, G. and J. Coutaz, *Introduction to model-based user interfaces.* Group Note, 2014. **7**: p. W3C.
4. Silveira, S.A.M., et al., *On the evaluation of usability design guidelines for improving network monitoring tools interfaces.* Journal of Systems and Software, 2022. **187**: p. 111223.
5. Bass, L. and B.E. John, *Linking usability to software architecture patterns through general scenarios.* Journal of Systems and Software, 2003. **66**(3): p. 187-197.
6. Folmer, E. and J. Bosch, *Architecting for usability: a survey.* Journal of systems and software, 2004. **70**(1-2): p. 61-78.
7. Svensson, R.B., et al., *Quality requirements in industrial practice—an extended interview study at eleven companies.* IEEE transactions on software engineering, 2011. **38**(4): p. 923-935.
8. Acerbis, R., et al. *Webratio 5: An eclipse-based case tool for engineering web applications*. Springer.
9. Koch, N., et al., *UML-based web engineering. web engineering: modelling and implementing web applications.* Human-Computer Interaction Series, 2008: p. 157-191.
10. Selic, B., *The pragmatics of model-driven development.* IEEE software, 2003. **20**(5): p. 19-25.
11. Wieringa, R. *Design science methodology: principles and practice*.
12. Urbieta, M., et al., *The impact of using a domain language for an agile requirements management.* Information and Software Technology, 2020. **127**: p. 106375.
13. Laurel, B. and S.J. Mountford, *The art of human-computer interface design*. 1990: Addison-Wesley Longman Publishing Co., Inc.

14. Cysneiros, L.M., V.M. Werneck, and A. Kushniruk. *Reusable Knowledge for Satisficing Usability Requirements*. in *13th IEEE International Conference on Requirement Engineering*. 2005. Washington, DC, USA: IEEE Computer Society.

15. Panach, J.I., et al. *Dealing with Usability in Model Transformation Technologies*. in *ER 2008*. 2008. Barcelona: Springer LNCS 5231.

16. Juristo, N., A.M. Moreno, and M.I. Sánchez, *Guidelines for Eliciting Usability Functionalities.* IEEE Transactions on Software Engineering, 2007. **33**(11): p. 744-758.

17. Juristo, N., *Impact of Usability on Software Requirements and Design*, in *Software Engineering*, L. Andrea and F. Filomena, Editors. 2009, Springer-Verlag. p. 55-77.

18. Campos, J., et al., *Systematic Analysis of Control Panel Interfaces Using Formal Tools Interactive Systems. Design, Specification, and Verification*. 2008, Springer-Verlag: Berlin, Heidelberg. p. 72-85.

19. Grosse, D., et al., *Supporting Tool for Usability Specifications*, in *World Congress on Medical Physics and Biomedical Engineering*, R. Magjarevic, Editor. 2009, Springer-Verlag: Munich, Germany. p. 845-847.

20. Jokela, T., et al., *Methods for Quantitative Usability Requirements: A Case Study on the Development of the User Interface of a Mobile Phone.* Personal Ubiquitous Comput., 2006. **10**(6): p. 345-355.

21. Ameller, D., X. Franch, and J. Cabot. *Dealing with Non-Functional Requirements in Model-Driven Development*. in *18th IEEE International Conference on Requirements Engineering (RE)*. 2010. Sydney, NSW.

22. Yi, L., M. Zhiyi, and S. Weizhong, *Integrating Non-functional Requirement Modeling into Model Driven Development Method*, in *2010 Asia Pacific Software Engineering Conference*. 2010, IEEE Computer Society.

23. Fatwanto, A. and C. Boughton, *Analysis, Specification and Modeling of Non-Functional Requirements for Translative Model-Driven Development*, in *International Conference on Computational Intelligence and Security*. 2008, IEEE Computer Society: Washington, DC, USA. p. 405-410.

24. Nguyen, Q.L. *Non-Functional Requirements Analysis Modeling for Software Product Lines*. in *ICSE Workshop on Modeling in*

Software Engineering. 2009. Washington, DC, USA: IEEE Computer Society.

25. Sindhgatta, R. and T. Srinivas, *Functional and Non-functional Requirements Specification for Enterprise Applications*, in *Product Focused Software Process Improvement*. 2005, Springer-Verlag: Berlin Heidelberg. p. 189-201.

26. Doerr, J., et al. *Non-functional Requirements in Industry - Three Case Studies Adopting an Experience-based NFR Method*. in *13th IEEE International Conference on Requirements Engineering*. 2005. Washington, DC, USA: IEEE Computer Society.

27. Martinie, C., et al. *DREAMER: A Design Rationale Environment for Argumentation, Modeling and Engineering Requirements*. in *28th International Conference on Design of Communication*. 2010. Säo Paulo, Brazil: ACM.

28. Akoumianakis, D., A. Katsis, and N. Vidakis. *Non-Functional User Interface Requirements Notation (NfRn) for Modeling the Global Execution Context of Tasks*. in *5th International Conference on Task Models and Diagrams for Users Interface Design*. 2007. Hasselt, Belgium: Springer-Verlag.

29. Röder, H., *Using Interaction Requirements to Operationalize Usability*, in *ACM Symposium on Applied Computing*. 2010, ACM: Sierre, Switzerland.

30. Shehata, M., A. Eberlein, and A. Fapojuwo, O., *A Taxonomy for Identifying Requirement Interactions in Software Systems.* Comput. Netw., 2007. **51**(2): p. 398-425.

31. Cronholm, S. and V. Bruno. *Do You Need General Principles or Concrete Heuristics?: A Model for Categorizing Usability Criteria*. in *20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat*. 2008. Cairns, Australia: ACM.

32. Henninger, S., *A Methodology and Tools for Applying Context-specific Usability Guidelines to Interface Design.* Journal Interacting with Computers, 2000. **12**(3): p. 225-243.

33. Sajedi, A., et al. *Fundamental Usability Guidelines for User Interface Design*. in *International Conference on Computational Sciences and Its Applications ICCSA*. 2008. Washington, DC, USA: IEEE Computer Society.

34. Soares, M.S. and J.L.M. Vrancken, *Model-driven User Requirements Specification using SysML.* Journal of Software, 2008. **3**(6): p. 57-68.
35. Sutcliffe, A.G., S. Kurniawan, and S. Jae-Eun, *A Method and Advisor Tool for Multimedia User Interface Design.* Int. J. Hum.-Comput. Stud., 2006. **64**(4): p. 375-392.
36. Escalona, M.J. and G. Arag, *NDT. A Model-Driven Approach for Web Requirements.* IEEE Trans. Softw. Eng., 2008. **34**(3): p. 377-390.
37. Escalona, M.J., et al., *Metamodeling the Requirements of Web Systems Web Information Systems and Technologies*, W. Aalst, et al., Editors. 2007, Springer Berlin Heidelberg. p. 267-280.
38. Panach, J.I., España, S., Pederiva, I., Pastor, O., *Capturing Interaction Requirements in a Model Transformation Technology Based on MDA*, in *Journal of Universal Computer Science (JUCS)*. 2007.
39. Lauesen, S. *Usability Requirements in a Tender Process*. in *Computer Human Interaction Conference, 1998*. 1998. Australia.
40. Sutcliffe, A., G. and M. Ryan, *Experience with SCRAM, a SCenario Requirements Analysis Method*, in *3rd International Conference on Requirements Engineering: Putting Requirements Engineering to Practice*. 1998, IEEE Computer Society. p. 164-171.
41. Cysneiros, L.M. and J.C.S.P. Leite, *Nonfunctional Requirements: from Elicitation to Conceptual Models.* IEEE Trans. on Softw. Eng., 2004. **30**(5): p. 328-350.
42. Jokela, T., et al., *8 Guiding Designers to the World of Usability: Determining Usability Requirements through Teamwork*, in *Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle*. 2005, Springer Netherlands. p. 127-145.
43. Gunduz, F. and A.S.K. Pathan. *Usability improvements for touch-screen mobile flight booking application: A case study*. in *Proceedings - 2012 International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2012*. 2012.
44. Troyer, O.D. and E. Janssens. *A feature modeling approach for domain-specific requirement elicitation*. in *2014 IEEE 4th*

International Workshop on Requirements Patterns (RePa). 2014.

45. Fahey, P., et al. *Human computer interaction issues in eliciting user requirements for an Electronic Patient Record with multiple users*. in *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*. 2011.
46. Temper, M., S. Tjoa, and M. Kaiser. *Touch to authenticate—Continuous biometric authentication on mobile devices*. in *1st International Conference on Software Security and Assurance (ICSSA)*. 2015. IEEE.
47. Rocha Silva, T., M. Winckler, and C. Bach, *Evaluating the usage of predefined interactive behaviors for writing user stories: an empirical study with potential product owners.* Cognition, Technology & Work, 2020. **22**(3): p. 437-457.
48. De Carvalho, E.A., A. Jatobá, and P.V.R. De Carvalho. *Usability for complex systems?: An experimental evaluation with functional resonance analysis method*. in *IHC 2019 - Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*. 2019.
49. Nhavoto, J.A., Å. Grönlund, and W.P. Chaquilla, *SMSaúde: Design, development, and implementation of a remote/mobile patient management system to improve retention in care for HIV/aids and tuberculosis patients.* JMIR mHealth and uHealth, 2015. **3**(1).
50. Elias, E., et al., *Towards an ontology-based system to improve usability in collaborative learning environments*, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012. p. 298-303.
51. Yuan, X. and X. Zhang. *An ontology-based requirement modeling for interactive software customization*. in *2015 IEEE International Model-Driven Requirements Engineering Workshop (MoDRE)*. 2015.
52. Abad, Z.S.H., et al. *Loud and Interactive Paper Prototyping in Requirements Elicitation: What is it Good for?* in *2018 IEEE 7th International Workshop on Empirical Requirements Engineering (EmpiRE)*. 2018.
53. Márquez, G. and C. Taramasco, *Using Dissemination and Implementation Strategies to Evaluate Requirement Elicitation*

Guidelines: A Case Study in a Bed Management System. IEEE Access, 2020. **8**: p. 145787-145802.

54. Abdallah, A., R. Hassan, and M.A. Azim. *Quantified extreme scenario based design approach*. in *Proceedings of the ACM Symposium on Applied Computing*. 2013.

55. Vitiello, G., et al. *UX-requirements for patient's empowerment - The case of multiple pharmacological treatments: A case study of it support to chronic disease management*. in *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017*. 2017.

56. Tanikawa, Y., R. Okubo, and S. Fukuzumi, *Process support method for improved user experience.* NEC Technical Journal, 2014. **8**(3): p. 28-32.

57. Abad, Z.S.H., et al. *Learn More, Pay Less! Lessons Learned from Applying the Wizard-of-Oz Technique for Exploring Mobile App Requirements*. in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. 2017.

58. Peruzzini, M. and M. Germani. *Designing a user-centred ICT platform for active aging*. in *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. 2014.

59. Lewis, J.R., *IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use.* International Journal of Human-Computer Interaction, 1995. **7**(1): p. 57-78.

60. Davis, F.D., *User acceptance of information technology: system characteristics, user perceptions and behavioral impacts.* International journal of man-machine studies, 1993. **38**(3): p. 475-487.