Revista Peruana de Computación y Sistemas 2018 1(2):25-40
http://dx.doi.org/10.15381/rpcs.v1i2.15380

ISSN Online: 2617-2003
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

Artículo de Contribución

Modelo de elaboración de una lista corta de técnicas candidatas para la priorización de requisitos

Model for the preparation of a short list of candidate techniques for the prioritization of requirements

E. Daniel Bravo Loayza
Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática. Lima, Perú
Email: eliasdanielbl@gmail.com

Resumen

La priorización de requisitos es una etapa importante en el proceso de desarrollo de software. Existen numerosas técnicas de priorización de requisitos que permiten obtener las prioridades relativas que le asignan cada stakeholder, y obtener una lista unificada y priorizada. En distintos proyectos se requieren distintas técnicas. Un paso previo consiste en elaborar una lista de técnicas candidatas a partir de las cuales se seleccionará a la técnica más adecuada. En la bibliografía consultada, los autores definen una lista previa, sin especificar el criterio que se utilizó para su preselección. Se propone un modelo de elaboración de una lista corta de técnicas candidatas para la priorización de requisitos. Previamente se realizará una revisión de la definición de requisito, stakeholder, ingeniería de requisitos, gestión de requisitos y su fase de priorización, además de otros conceptos relacionados. Se propone un modelo cuyos elementos que deberán cumplir las técnicas candidatas, permitiendo implementación en herramientas colaborativas CMS. Al final se incluye proceso de evaluación, que validará el modelo propuesto.

Palabras clave: Requisitos; priorización de requisitos; técnica de priorización; software colaborativo; CMS.

Abstract

Prioritization of requirements is an important stage in the software development process. Technical stocks of prioritization of the requirements that must obtain the priorities that correspond to each participant and obtain a unified and prioritized list. Different techniques are sought in different projects. A preliminary step is to develop a list of candidate techniques from which a more appropriate technique is selected. In the bibliography consulted, the authors define a previous list, without specifying the criterion that was used for their pre-selection. A model for the preparation of a short list of techniques for the prioritization of requirements is proposed. Previously a review of the definition of requirement, stakeholder, requirements engineering, requirements management and its prioritization phase, as well as other related concepts. A model is proposed so that the elements comply with the candidate techniques, the implementation of the system in collaborative CMS tools. At the end, the evaluation process is included, which will validate the proposed model.

Keywords: Requirements; requirements prioritization; prioritization technique; collaborative software; CMS.

Correspondencia:
Dirección: Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática. Calle Germán Amézaga N° 375, Ciudad Lima 1.

Recibido 12/03/2018 - aceptado 01/08/2018

Citar como:
Bravo D. Modelo de Elaboración de una Lista Corta de Técnicas Candidatas para la Priorización de Requisitos. Revista Peruana de Computación y Sistemas 2018 1(2):25-40. http://dx.doi.org/10.15381/rpcs.v1i2.15380

1. Introdución

La priorización de requisitos es una actividad importante en el desarrollo de software [59], y, en aquellos proyectos con limitados recursos y restricciones de tiempo de entrega, es particularmente importante. La priorización de requisitos proporciona múltiples beneficios, antes que se inicie la fase de diseño y elaboración, apoyando en la implementación de un software con las características demandadas por los stakeholders [3]. Existen numerosas técnicas que permiten clasificar y ordenar los requisitos. Algunas de ellas cuentan con el respaldo de numerosas investigaciones e implementaciones; de otras solo se dispone del modelo teórico o de descripciones. La implementación de las técnicas de priorización se realiza a través de modelos. Estos deben ser soportados por herramientas de software para realizar dicha priorización, es decir, que la herramienta pueda sustentar el modelo, instanciando la técnica.

En la bibliografía consultada, los autores que realizan procesos comparativos, evaluación o selección y proponen una lista de técnicas candidatas; no definen o sustentan cómo elaboraron dicha lista. Por ejemplo, [25] define una lista de seis técnicas de priorización sobre la cual realiza su investigación, [33] diseña un experimento para comparar diversas técnicas de priorización, en su investigación [25] revisó la efectividad de tres técnicas, describiendo cada una de ellas, [49] realiza un exhaustivo estudio comparativo sobre dos técnicas; no indican cual fue el procedimiento con el que las identificaron como las más adecuadas; en estas investigaciones no se describe el proceso mediante el cual se elaboró la lista sobre la cual trabajan.

Por ello, se propone un modelo de elaboración de una lista corta de técnicas de priorización candidatas. Esta lista permitirá asegurar que se consideró y evaluó aquellas técnicas válidas para realizar la priorización, excluyendo aquellas que no podrían realizar la tarea y considerando aquellas que sí. La técnica como shortlisting, o lista corta [58] permite obtener una lista corta a partir de una lista larga de alternativas disponibles, basados en un conjunto de consideraciones. La técnica permite ser implementada de diferentes maneras.

El modelo propuesto deberá contar con elementos de carácter teórico, que respalden su validez, y práctico, que aseguren su funcionalidad, bajo la premisa de que estas técnicas deben ser implementables en un entorno real. Las técnicas candidatas serán evaluadas sobre la base de los elementos que forman parte el modelo.

Se debe tener en consideración que la priorización de requisitos es un tema importante no solo para la ingeniería de software, sino también para la gestión de Tecnologías de Información - TI y otros ámbitos, como en la industria de productos y servicios [51].

El presente estudio se organiza de la siguiente manera: En la sección 2 Fundamentación Teórica se presentan los conceptos y definiciones a partir de los cuales se desarrolla la priorización de requisitos. En la sección 3 Priorización de Requisitos se describe el proceso de priorización, y sus aspectos relevantes. En la sección 4 se desarrolla la propuesta del presente estudio, explicando cada uno de los elementos del modelo. En sección 5 se realiza el proceso de evaluación que valida el modelo propuesto, basado en el método de evaluación de [32]. Para evaluar el modelo propuesto se realizará un test de viabilidad, desarrollándose dos casos de prueba, que determinarán la validez desde el enfoque práctico el modelo teórico propuesto. Finalmente se incluyen las conclusiones y recomendaciones.

2. Fundamentación Teórica

Se describirán los conceptos relevantes que componen el marco conceptual alrededor de las técnicas de priorización de requisitos. Asimismo, se realizará una descripción de las principales técnicas, las mismas que nos darán una visión general de cómo será implementado el modelo propuesto, que permitirá abordar el problema formulado en la Introducción.

2.1. Requisito

Los requisitos son el punto de partida de todo proyecto nuevo y también un coadyuvante en proyectos de mejora. Con frecuencia son muy ambiguos y en general resultan difíciles de definir en forma concisa y comprensible. Los requisitos tienden a ser inestables, crecer continuamente durante el proceso de desarrollo de software, incluso durante la fase de pruebas [23].

Un error en la lista de requisitos obtenidos puede tener consecuencias en el software desarrollado y provocar el rechazo del producto por parte de los stakeholders; incluso si se terminó en desarrollar a tiempo. Existe un alto costo si los requisitos implementados no son los esperados por los stakeholders [41]. La estimación de costos de software a nivel de proyecto no puede realizarse a menos que se comprendan bien los requisitos del mismo. Los cambios en los requisitos ocasionan modificaciones del diseño e incrementan los costos de desarrollo [51].

En los distintos textos, los requisitos de software se agrupan o definen como en funcionales y no funcionales, y, además, como requisitos del dominio [50]. Sin embargo, la mayoría de las veces, identificar un requisito como de un tipo de otro no es tan clara como sus definiciones lo establecen. Por ejemplo, un requisito de usuario sobre aspectos de autorización y autenticación podría ser clasificado como 'requisito no funcional'. Si se desagrega el requisito, se pueden obtener otros como funcionales, dependiendo del contexto. Estas diferencias en la concepción de los requisitos se encuentran también presente en los stakeholders, de allí que existan distintas apreciaciones sobre los requisitos para un mismo software.

2.2. Stakeholders

Los stakeholders son aquellos que se benefician o se ven impactados en forma directa o indirecta por el sistema que está en desarrollo [37] y [50].

Un stakeholder es una persona o un elemento de una organización que tiene un interés, responsabilidad o será impactado por un sistema o sus salidas. Los stakeholders pueden ser obreros, instaladores, mantenedores, administradores [30], clientes proveedores y otros.

El principal riesgo en la priorización y por tanto en la gestión de requisitos, es una inadecuada participación de los stakeholders [16].

En español, el término más próximo es 'interesado'. Al igual que [7] en el presente trabajo de investigación se usará el término en inglés stakeholder. Los stakeholders fueron también descritos [36] como participantes.

Es muy importante recoger información relevante de los stakeholders tales como: nombre, cargo / posición, información de contacto, necesidades / expectativas sobre el proyecto, e identificar los grupos de stakeholders de tipo interno o externo.

2.2.1. Diferemtes Puntos de Vista: Los stakeholders tienen sus propias características e intereses entre sí [37], los requisitos de la solución a desarrollar se capturarán desde sus propios enfoques. Por ejemplo, en una empresa, el departamento de ventas estará interesado en funciones y características que le permitan llegar a los mercados. En el departamento de finanzas estarán interesados en aquellas funcionalidades que se puedan desarrollar dentro de un presupuesto. Los usuarios requerirán funcionalidades que puedan utilizar y ser más productivos. El departamento de desarrollo se enfocará en funcionalidades y características tecnológicas y de arquitectura de la plataforma. Por tanto, diferentes stakeholders tienen diferentes perspectivas e intereses, y pueden darle diferente importancia a determinado atributo [8].

2.3. Ingeniería de Requisitos

La ingeniería de requisitos brinda y facilita el proceso de comprender lo que desea el stakeholder, a partir de sus necesidades, se estima la viabilidad del proyecto, a partir de ello se gestiona una solución viable al problema. Permite definir una solución con un nivel aceptable de precisión [37]. La ingeniería de requisitos no consiste solo en colectar necesidades, sino que esta cubre el ciclo de vida sistema, más allá del desarrollo del sistema. Para que un software sea aceptado por los stakeholders, los requisitos deben ser capturados, analizados y priorizados [3].

La ingeniería de requisitos cubre también requisitos de negocio y de Tecnologías de Información - TI, ayudando al desarrollo de sistemas que cumplan con las exigencias de la organización. Así, la ingeniería de requisitos se puede aplicar a otros subdominios de la organización [30], [51] y [55].

En resumen, la ingeniería de requisitos se encarga de capturar, clasificar, filtrar y registrar los requisitos del sistema; realiza la gestión de cambios, entre los usuarios y los responsables de la solución [36].

2.4. Gestión de requisitos

Permite al equipo de proyecto a identificar, gestionar y hacer seguimiento a los requisitos y los cambios a estos a lo largo de la construcción del proyecto. Otras denominaciones son 'ciclo de vida de la ingeniería de requisitos' [12] o 'proceso de ingeniería de requisitos' [53]. Gestionar requisitos involucra actividades como gestión de cambios y la trazabilidad de requisitos [37]. Su propósito es gestionar los requisitos para cada uno de los entregables del proyecto, con el fin de asegurar la alineación entre los requisitos, documentos del proyecto y los resultados planeados. Debe gestionar todos los requisitos aprobados [14].

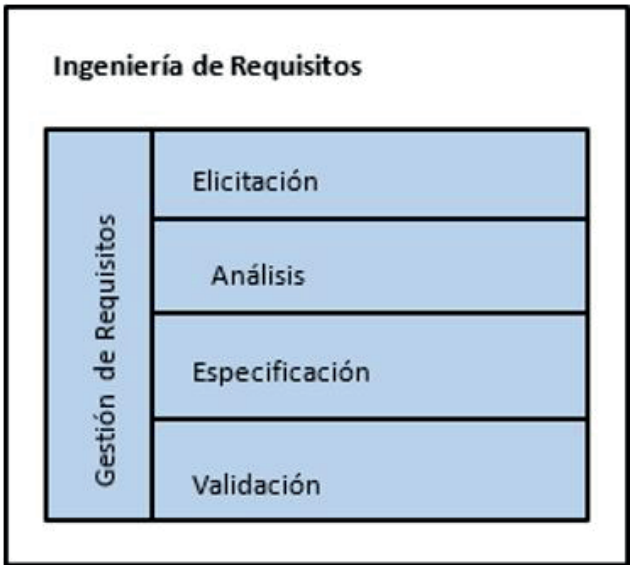

Fig. 1. Fases de la ingeniería de requisitos

En proyectos pequeños, la implementación de los procesos de desarrollo de la ingeniería de requisitos es fácil de incorporar; en proyectos más grandes, la gestión de los requisitos se hace más compleja y difícil [1].

Existen herramientas que apoyan las tareas de gestión de requisitos, incluyendo la priorización. Sin embargo, [37] la mayoría de ellas son de pago y muchas veces no son accesibles por su costo.

2.5. Fases en la gestión de requisitos

Las fases de la gestión de requisitos son la elicitación, el análisis, la especificación y la validación [25], ver Fig. 1.

2.5.1. Elicitación: El propósito es colectar, capturar, descubrir y desarrollar los requisitos de una variedad de fuentes, incluidos los stakeholders. Uno de los pasos en la elicitación es la priorización y esta es un importante procedimiento para la toma de decisiones [28]. Se relaciona directamente con el dominio del problema y el usuario, quien recaba la información del negocio [36]. Permite identificar los riesgos y supuestos del proyecto [6].

2.5.2. Análisis: Tiene como propósito examinar, comprender y modelar los requisitos; luego, se evalúan en términos de calidad, corrección y claridad. Permite determinar qué necesidades son más o menos importantes, y cuáles podrían ser descartadas. También se encarga de la gestión de cambios, el ambiente, los requisitos adicionales que se prevén que surjan [7], [12].

2.5.3. Especificación: Registra y documenta los requisitos. Es usada por los diseñadores, quienes pueden diseñar y construir el sistema. Constituye la entrada a las actividades en las siguientes fases y etapas [53]. Consiste en el registro de los requisitos los stakeholders en un formato que el equipo de software pueda entender. También se denomina documentación.

Una especificación de requisitos de software (ERS) es un formato que define una descripción detallada de las características de la solución informática que se va a construir, en las primeras etapas del proceso. Algunos sugieren que para una especificación debe desarrollarse y utilizarse una 'plantilla predefinida'. Puede incluir elementos gráficos [37], por ejemplo, con UML.

2.5.4 Validación: En esta fase, se valida la calidad de los requisitos listados, a fin de asegurar que estos representen las necesidades de los stakeholders. La validación de requisitos verifica que los entregables cumplan con los estándares establecidos [40]. La validación verifica que los requisitos permiten atender las necesidades de los stakeholders.

2.6. Colaboración e Ingeniería de colaboración

Con la colaboración se unen esfuerzos para alcanzar objetivos comunes. Es un proceso en el que los stakeholders con diferentes perspectivas del problema pueden de manera constructiva explorar dichas diferencias y buscar soluciones más allá de sus visiones. En la colaboración todos los participantes intervienen en cada uno de los pasos, por lo que no es posible distinguir los trabajos individuales en el resultado final [27]. De acuerdo con [46], más del 36% del rendimiento de una organización depende de las capacidades de la misma para colaborar. La colaboración cada vez más ocupa una parte considerable en el desarrollo de productos, por lo que su mejora se está convirtiendo en un factor importante en la productividad [16].

La ingeniería de colaboración tiene como objetivo diseñar e implementar procesos de alto valor para aquellas tareas colaborativas recurrentes [27]. Su propósito es diseñar procesos de manera que los stakeholders los puedan ejecutar con éxito sin la intervención de facilitadores.

3. Priorización de Requisitos

La priorización de requisitos es el proceso de definir las importancias relativas de los requisitos solicitados [20], en los casos en los cuales existen múltiples opciones que deben considerarse [10]. En general, existen más requisitos que los recursos disponibles (limitación de recursos); por ello, es importante seleccionar los requisitos más prioritarios con el fin de garantizar la entrega de un sistema de alto valor [23]. Los requisitos con alta prioridad se pueden implementar al principio, mientras que aquellos con menor prioridad se pueden posponer para entregas posteriores o incluso ser dejados de lado [28].

La priorización de requisitos no solo se utiliza en el ámbito de la ingeniería de software; por ejemplo, se ha aplicado en el despliegue de servicios digitales [60] tales como Apple's iPhone y App Store, destacando la importancia que viene adquiriendo la priorización de requisitos. Por su parte, [32] propusieron la priorización de requisitos para el desarrollo de un diseño conceptual en sistemas de aviones, con la que se proporcionaría una mejor aceptación del producto por parte del usuario final. Además, se aplica en el desarrollo de productos y servicios [56].

Por lo general, la priorización es realizada por múltiples stakeholders, cuyas prioridades individuales se combinan en una única lista de prioridades [44].

3.1. Proceso de priorización

Los distintos métodos permiten obtener una lista de requisitos implementables, a partir de la evaluación que un individuo o un equipo a un conjunto de requisitos mediante el proceso de priorización [8]. Los métodos de priorización ayudan a los responsables de la toma de decisiones a analizar los requisitos y asignarles valores que reflejen su importancia.

El proceso de priorización consta de tres etapas diferenciadas y consecutivas [21]:

1. Etapa de preparación: Un analista estructura los requisitos de acuerdo con el método de priorización a ser empleado: se forma el equipo, se nombra al jefe del equipo para la sesión y se provee toda la información necesaria. Se define el criterio de evaluación.

2. Etapa de ejecución: Es la etapa intermedia en la cual los stakeholders realizan la priorización de los requisitos utilizando la información obtenida en la primera etapa.

3. Etapa de preparación de los resultados: Los resultados se presentan a los stakeholders. Algunos métodos de priorización implican diferentes tipos de cálculos que se deben llevar a cabo antes de presentar los resultados.

3.2. Aspectos de la Priorización de Requisitos

Un aspecto puede definirse como un atributo o propiedad de cada requisito, dentro de un proceso de priorización de requisitos [8]:

1. Importancia: Mediante el cual los stakeholders definen la urgencia de requisito sobre otro.
2. Penalidad: Que consecuencias pueden ocurrir si no se implementa un requisito.
3. Costo: Cuales son los gastos en que se incurrirá.
4. Tiempo: Se valoran de acuerdo con el tiempo necesario para implementarlos en forma efectiva.
5. Riesgo / volatilidad: Cuantifica la posibilidad de un efecto o acción no deseados y que penalizarían el proyecto o requisito.
6. Utilidad: Indica que aprovechamiento brindará.

3.3. Técnicas de priorización de requisitos

El propósito de cualquier técnica priorización es asignar valores a los distintos objetos por priorizar para establecer un orden relativo entre los elementos del conjunto. En la ingeniería de requisitos, los objetos son requisitos [10]. También se denomina 'métodos' [26] a las técnicas de priorización. Los investigadores han desarrollado diferentes técnicas de priorización de requisitos para una correcta selección y ordenación de los requisitos de usuario [23] y [28]. La elección de la técnica dependerá de las características del proyecto [63]. De acuerdo con [23], la mayoría de técnicas propuestas son solo aptas para pocos requisitos y no son adecuadas cuando existe un gran número de requisitos, como en los proyectos grandes.

	SR-1	SR-2	SR-3	SR-4	SR-5
SR-1	1	8	1/5	3	1
SR-2	1/8	1	1/5	1/7	1/7
SR-3	5	5	1	1	2
SR-4	1/3	7	1	1	1/2
SR-5	1	7	1/2	2	1

Fig. 2. Ejemplo de una matriz usando PAJ. Los stakeholders que utilizan la técnica llenan con valores la mitad superior (resaltado) de la matriz

Los problemas con las técnicas de priorización de requisitos pueden ser resumirse en cinco:

1. las técnicas existentes no proporcionan una solución escalable cuando se amplían en forma considerable los requisitos, en proyectos grandes (de orden de varios cientos o miles);
2. la mayoría de las técnicas consumen mucho tiempo y recursos en su implementación;
3. los resultados son propensos a errores o presentar rangos de imprecisión;
4. los resultados no concuerdan cuando se vuelve a repetir la prueba;
5. resuelven problemas de pequeña escala o proyectos de alcance limitado, pero conforme el proyecto es más grande, tienden a producir errores o aparecer un nivel de imprecisión.

Para priorizar los requisitos, los stakeholders tendrán que compararlos para determinar su importancia relativa a través de un sistema de ponderación o puntuación. Estas comparaciones se convierten en un proceso complejo con el aumento en el número de requisitos [4].

3.3.1. Principales técnicas de priorización: Existen numerosas y diferentes técnicas sobre cómo priorizar, y podría resultar difícil determinar el método más adecuado debido al gran número de ellos. En la bibliografía consultada, de las diversas técnicas identificadas solo se dispone de una breve descripción; no hay algoritmos disponibles, o mayor información que posibilite su implementación o evaluación. Las principales técnicas, para las cuales existe información disponible para ser evaluadas serán de diez: 1) Asignación Numérica (AN); 2) Asignación Numérica (AN); 3) Priorización de Valor Orientado (PVO); 4) Voto Acumulativo (VA); 5) Árbol de Búsqueda Binaria (ABB); 6) Planificación del Juego (PJ); 7) Diez Mayores Requisitos; 8) Ordenamiento Burbuja; 9) Grupos Prioritarios; y 10) Despliegue en Función de la Calidad.

Asignación Numérica (AN), a cada requisito le asigna un símbolo que representa la importancia percibida para el stakeholder. Existen diversas variantes de esta técnica, que utiliza categorías como alto, medio o bajo [26]. O una clasificación escala de cinco valores: mandatorio, muy importante, importante, no importante o sin valor [10].

Esta técnica está estandarizada en la IEEE Std. 830-1998 [32].

Proceso Analítico Jerárquico (PAJ), también se la conoce como técnica de comparación por pares, el stakeholder realiza un total de $(n\times(n-1)/2)$ comparaciones, en 10 requisitos se efectúan 45 comparaciones, en 50 serán 1225 [63]. No es la técnica más adecuada para un número grande de requisitos [45]. Asimismo, otros investigadores [52] afirmaron que esta técnica tiene limitaciones cuando los stakeholders se encuentran en entornos distribuidos. Es una de las técnicas más estudiadas de priorización de requisitos.

PAJ Jerárquico (PAJJ), este enfoque busca optimizar la técnica PAJ, permite gestionar un alto número de requisitos, realizando la priorización en la misma jerarquía, reduciendo el número total de comparaciones, aunque, no es considerada una técnica de fácil uso. A diferencia del PAJ, no existen estudios de su implementación [63].

Priorización de Valor Orientado (PVO), utiliza un framework para la priorización y la toma de decisiones sobre los requisitos; permite a los stakeholders visualizarlos de una adecuada toma de decisiones, con un enfoque en los valores centrales, es decir aquellos que son más relevantes para el negocio. Esta técnica proporciona una mayor participación a los stakeholders, minimizando discusiones y argumentaciones para cada requisito individual, y si enfatizando en los valores centrales del negocio [26].

Voto Acumulativo (VA), también conocido como técnica de los cien puntos o prueba de los cien dólares, es un método simple y directo, utilizado en diversos estudios de priorización en ingeniería de software. A los stakeholders se les pide priorizar los requisitos en razón de una escala. El número de unidades asignadas representa su prioridad relativa (por ejemplo, importancia, coste, riesgo) en relación con los otros requisitos [45]. El requisito que obtiene la más alta puntuación es el más importante [21]. Es una de las técnicas más rápidas y precisas. Cuando los participantes han registrado sus puntos, se calcula los puntos que cada requisito tiene, y presenta los resultados. Cuando los puntos se han distribuido, los requisitos con puntaje más alto determinan se consideran los más importantes [10]. Para una fácil referencia al problema, después de dividirlo por la constante la suma de los valores de cada elemento del vector es igual a 100 [41], ver Fig. 3 [28]. Esta técnica es especialmente útil cuando se tiene un alto número de stakeholders, porque permite realizar la priorización de una forma fácil [8]

Árbol de Búsqueda Binaria (ABB), es un tipo especial de árbol de búsqueda binario, en el que cada nodo está etiquetado con un requisito candidato. La creación de un ABB con requisitos que representan los elementos de un conjunto se convierte en un método para priorizar requisitos de software. Al final, se tiene un árbol de los requisitos, el mismo que se compone de nodos que indican el nivel de prioridad de su requisito etiquetado [7]. Su principal ventaja es que utiliza una técnica cuyo algoritmo es muy conocido, además de su bajo número de operaciones, siendo $O(n \log n)$ [19]. Algunos autores, como [21], señalaron que esta técnica es propensa a fallos y provee resultados poco fiables.

Planificación del Juego (PJ), utiliza los principios de las metodologías ágiles, como la Programación Extrema [48]. Es una variante de la técnica de Asignación Numérica, en la que los stakeholders distribuyen los requisitos en tres grupos: 'aquellos sin los cuales el sistema no funcionará', 'aquellos menos esenciales, pero proveen un significativo valor' y 'aquellos que sería bueno tener'. Para cada requisito se decide a qué grupo pertenece, no se comparan entre sí. El tiempo de priorizar *n* requisitos es *n* operaciones. El resultado es una lista de requisitos ordenada [20]. PG es una técnica muy flexible y puede gestionar un gran número de requisitos, sin consumir demasiado tiempo para priorizarlos a todos [20].

Diez Mayores Requisitos (DMR), cada stakeholder selecciona sus diez más importantes requisitos. No se requiere de algún otro procedimiento de ordenación. Tiene como inconveniente que pueden surgir conflictos o insatisfacción por parte de los stakeholders [53]. Es una técnica adecuada cuando los stakeholders pertenecen a una categoría similar [9], por ejemplo, todos son de ventas.

$$(p_1,\ldots,p_k) \text{ where } p_i \geq 0\,,\ \sum_{i=1}^{k} p_i = 100$$

Fig. 3. Análisis de composición de datos

Ordenamiento Burbuja (OB), está muy relacionado con el PAJ, ya que ambas técnicas utilizan la ordenación por pares y requieren $(n\times(n-1)/2)$ comparaciones. Sin embargo, solo tiene que determinarse cuál de los dos requisitos es más prioritario. El resultado es una columna clasificada donde el requisito más importante se ubica en la parte superior, y el menos importante en la parte inferior. Es una de las técnicas más básicas y simples de ordenación. Para una pequeña lista de requisitos, el ordenamiento burbuja puede ser una alternativa válida [59].

Grupos Prioritarios (GP), A pesar de su nombre, esta técnica en realidad no produce grupos de requisitos sino una lista clasificada de requisitos. El principio es asignar a cada requisito una categoría: alta, media y baja prioridad. Los grupos de alta, media y baja prioridad originales se dividen a su vez en subgrupos. La técnica de grupos prioritarios es lenta de realizar y difícil de utilizar, no siendo adecuada para priorizar pequeños números de requisitos. Es una técnica poco confiable debido a que puede producir resultados erróneos [23].

Despliegue en Función de la Calidad (DFC), analiza los requisitos de forma sistemática y los convierte en información sobre la base de las características de los productos, componentes, procesos y calidad. Para aplicarse, se descomponen los requisitos de los stakeholders en procesos de diseño de producto, luego se configuran los componentes, procesos de planificación y control de calidad, convirtiendo los originales requisitos en especificaciones del producto. Esta técnica tiene como deficiencia que soporta un límite de treinta requisitos. Resulta dificultoso hacer cambios a los requisitos, ya que involucraría un nuevo cálculo de los resultados; además, no permite manejar la

dependencia entre requisitos, gestionándolo como independientes [53]. Ayuda a convertir las necesidades del cliente en características del producto o servicio, pero no define como realizar la conversión de necesidades en atributos.

3.3.2. Priorización de requisitos en entornos distribuidos: Un entorno de negocio cada vez más globalizado, obliga una mayor investigación en la ingeniería de requisitos y en la priorización de requisitos con software distribuido como con el que cuentan las organizaciones [4]. Hay enfoques que promueven la participación de los stakeholders en la obtención de requisitos o la priorización; sin embargo, estos enfoques no apoyan en forma adecuada contextos no tradicionales, como dispositivos móviles o cloud computing. Esto se debe al hecho de que, en estos contextos, se necesita involucrar a un gran número de stakeholders heterogéneos, globalmente distribuidos y muy probablemente anónimos. En el caso de stakeholders geográficamente distribuidos, vinculados a un proyecto, que se encuentran en diferentes lugares dentro de una misma región; cada uno tiene su propia percepción y expectativas para dicho proyecto [4].

Algunas de las investigaciones existentes se enfocan en el rol del software colaborativo en estos procesos distribuidos y otros se enfocan en aspectos de gestión de comunicación de procesos [44].

3.3.3. Herramientas de Colaboración para obtener Requisitos: Las herramientas de colaboración permiten la cooperación de todas las partes interesadas en las distintas fases del proceso del proyecto, incluyendo proyectos de desarrollo de software. La elección de una entre la amplia gama de herramientas de colaboración disponibles exige un conjunto de tareas a realizar, como elaborar un inventario del software y hardware existentes, conocer la experiencia y capacidades del equipo [60]. Por lo general, solo un grupo de stakeholders son consultados, obteniéndose con ello una visión incompleta de la importancia de los requisitos, ya que se omiten algunos y se realiza la priorización en forma inadecuada [29]. La intensa comunicación durante la elicitación de requisitos exige un alto nivel de colaboración. Sin embargo, es difícil reunir a las partes interesadas al mismo tiempo y lugar.

3.4. Licencia de software

Una licencia de software es un acuerdo entre el usuario y el propietario del software que le otorga al primero ciertos permisos de uso. Es un acuerdo legalmente vinculante que especifica los términos de uso de un software y define los derechos del desarrollador y del usuario final [57]. Por lo general, un acuerdo de licencia de software se basa en un conjunto de reglas que un usuario debe cumplir mientras esté usando el software [62].

3.4.1. Software con licencia GPL: La Licencia Pública General GNU (General Public License - GNU GPL) está definido en un documento que señala los criterios que autorizan la distribución para publicar programas de software bajo los términos de copyleft. El Proyecto GNU publica la mayoría de sus proyectos de software bajo esta licencia. Una ventaja de la licencia GNU GPL es que ya está redactada, ha sido ampliamente revisada, con múltiples aportes y evaluaciones; además, se encuentra disponible de manera libre, lo cual permite incorporarla sin restricciones en cualquier proyecto.

3.4.2. Software con licencia BSD: A diferencia de la licencia GPL, permite el uso de código fuente con licencia BSD en software no libre (propietario). Además, el desarrollador no está obligado en distribuir el código fuente, incluso si el software que distribuye tiene un valor comercial. Es el tipo de licencia que menos restricciones tiene.

3.5. Software colaborativo

Es aquel que permite a los miembros de un equipo compartir ideas, información y tareas para apoyar a los procesos del negocio de la forma más eficiente posible [13], apoya a los grupos de trabajo [22].

La ingeniería de software implica a personas que colaboran para desarrollar un mejor software [27]. Los proyectos de desarrollo de software grandes requieren la interacción de especialistas de diferentes campos, quienes se comunican para la toma de decisiones y coordinar actividades [18]. Es una tendencia que los proyectos de software estén constituidos por equipos de cuyos integrantes trabajan en diferentes puntos geográficos [44].

Las herramientas colaborativas son tipos de software ampliamente conocidos y difundidos en la actualidad, por ello que su uso es una alternativa válida y viable. Las herramientas de gestión de contenidos son un tipo de software colaborativo. Algunos ejemplos de software colaborativo son foros [17], wikis [44]; [49], CMS [54], entre otros.

3.5.1. Foros: Permiten la interacción entre todos los stakeholders. Se asume que aprovechan de la comunicación asíncrona, lo que les permite pensar en el problema, comprender otros puntos de vista y generar nuevas ideas para contribuir nuevamente. Permite a los administradores del foro la moderación, la edición y la eliminación de comentarios. Una de sus características más importantes es que permiten la discusión entre todos los stakeholders. También incluyen un sistema de votación, por lo general no se aplica a grupos de stakeholders interesados en el sistema de información, y quieren contribuir con ideas. Cada página tiene una sección para que los *stakeholders* puedan comentar y debatir. Un foro orientado al voto está configurado para que comenten, realicen preguntas y voten las ideas propuestas.

Los foros fueron propuestos como software colaborativo [17] para la gestión de requisitos.

3.5.2. Wikis: Una wiki es "Una web donde la gente crea, edita y enlaza una colección de artículos. Cada web puede ser vista como un conjunto de páginas enlazadas, pero las wikis permiten que la comunidad cambie el contenido y la estructura. De esta manera las wikis son un buen camino para coordinar contenido […]" [44]. El mecanismo de interacción colaborativa de las wikis ha sido aplicado en todo el ciclo de vida de desarrollo de software. Las wikis tienen ciertas limitaciones, como el uso de datos no estructurados, difícil reutilización del conocimiento, limitadas funciones de búsqueda (se reducen a la búsqueda del título de la página) [64]. Las wikis son herramientas para el trabajo en un entorno distribuido y colaborativo, y sirven para resolver problemas detrás de la obtención de requisitos en dicho entorno. Un buen ejemplo del éxito de este concepto es Wikipedia.

Las wikis fueron propuestas como software colaborativo [17] para la gestión de requisitos.

3.5.3. CMS: Son un tipo de aplicaciones que básicamente ayudan a los usuarios y autores a crear, editar, mezclar y almacenar el trabajo o contenidos embebidos [52], publicación, gestionar y facilita el descubrimiento de información [31]. Definen procedimientos para gestionar el flujo de trabajo en un entorno colaborativo, pueden ser manuales o mediante un flujo automatizado. Apoyan la gestión de cualquier tipo de contenido [63]. Algunos tipos de aplicaciones CMS son los blogs, comentarios o información de un determinado tópico; y las noticias en línea.

En años recientes, los sistemas de gestión de contenidos incrementaron su presencia en las organizaciones gracias a su eficacia [54].

Existen CMS de código abierto. El término código abierto hace referencia a la posibilidad que se tiene para acceder con libertad a su código y modificarlo, pero con algunas consideraciones. Mientras el software propietario es creado, distribuido, y mantenido por una empresa, en el software de código abierto estas tareas son manejadas por la comunidad de desarrolladores y usuarios.

También existen CMS comerciales, en los cuales el vendedor tiene el control sobre el desarrollo de los módulos. De ese modo, crea un entorno de usuario más amistoso y seguro, debido a que son CMS comerciales la comunidad no tiene acceso al código fuente; además, no está permitido realizar modificaciones. Por lo general, tienen un menor número de actualizaciones de seguridad que los CMS de código abierto. Algunos CMS comerciales incluyen el código fuente en su distribución, bien estructurado y documentado, con una arquitectura extensible que permite a los desarrolladores independientes crear extensiones y módulos.

Los CMS de código abierto son libres de varias maneras. El usuario puede hacer modificaciones en el producto y su código. No hay costo de licencia, ya que cualquiera puede descargarlo e instalarlo en un equipo de prueba o servidor, aunque es probable que tenga que pagar por el servidor o pagar a alguien que instale o mantenga el sistema (Resumen de [39]). Los sistemas de gestión de contenidos adoptan un modelo de tres capas, compuesta por capa de presentación, capa de aplicación y capa de datos [63].

Los CMS no han sido creados para soportar las actividades de un proceso de priorización de requisitos.

4. Propuesta de un Modelo de Elaboración de una Lista Corta de Técnicas Candidatas para la Priorización de Requisitos

Existen numerosas técnicas identificadas en la literatura, por ejemplo [3] en una revisión sistemática de la literatura lista hasta 50 técnicas de priorización. El modelo propuesto permite elaborar una lista corta de técnicas, a las que se denomina técnicas candidatas. Serán consideradas técnicas candidatas aquellas que cumplan con cada uno de los elementos del modelo, con ello se asegura que puedan ser el instrumento que permita al equipo de proyecto realizar de manera adecuada la priorización de los requisitos, obteniendo una única lista unificada y priorizada. A partir de esta lista corta de técnicas candidatas, el equipo de proyecto podrá identificar cual es la más adecuada para realizar la priorización en el proyecto que se ha planificado desarrollar. Ya no se requerirá realizar un exhaustivo análisis a todas las técnicas para identificar cual sería la más adecuada.

Al igual que [7], en el presente trabajo investigación se realizará la integración de diferentes elementos heterogéneos entre sí. Todos ellos son necesarios para que el modelo propuesto sea factible de implementar, en un entorno real; es decir, que sea viable y funcional. Con ello, se obtendrá un modelo que tenga como propósito general solucionar el problema planteado en el presente trabajo de investigación.

Los elementos del modelo propuestos son:

1. Modelo de desarrollo de software: deberá incluir un método que permitan el proceso de desarrollo de software, como [36];
2. Algoritmo de la técnica: utiliza una de las técnicas de priorización existentes, similar a la propuesta de [7];
3. Licencia de software: Para hacer uso de una herramienta colaborativa se requiere de una licencia que otorgue al usuario de permisos de uso, respetando el marco legal y normativo;

4. Herramienta colaborativa: Basado en las propuestas de [17], [44], [45] y [49];
5. Componente para la herramienta colaborativa: el que permitirá capturar las prioridades individuales de los stakeholders, extendiendo las funcionalidades de las herramientas colaborativas.

4.1. Modelo propuesto

A continuación, se describen cada uno de los elementos que constituyen el presente modelo propuesto, ver Fig. 4.

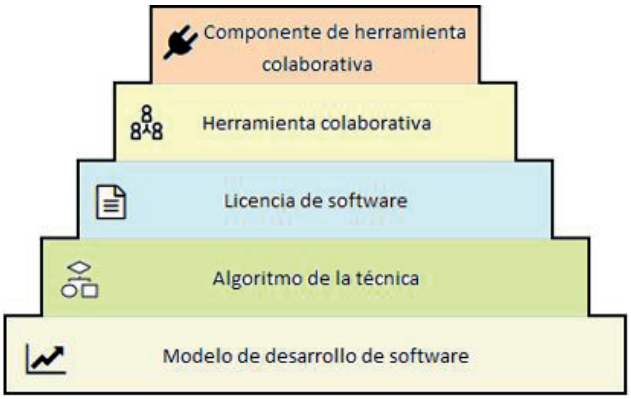

Fig. 4. Elementos del modelo propuesto.

4.1.1. Modelo de desarrollo de software: Tal como se documentó en la Fundamentación teórica y sobre la base del trabajo [36] 'Metodología para la elicitación de requerimientos de software basada en el Marco Lógico', se identificaron numerosos modelos de desarrollo de software que serán soportados en el modelo propuesto, y que incluyen:

1. CMMI: Es un modelo orientado a la mejora de la Organización mediante la mejora de sus procesos, con el fin desarrollar mejores productos y servicios. Es desarrollado y administrado por la Universidad Carnegie Mellon. El CMMI® [14] son colecciones de las mejores prácticas que ayudan a las organizaciones a mejorar sus procesos. Se constituye como una propuesta integral entre la ingeniería de software e ingeniería de sistemas en una única propuesta [33].
2. ISO/IEC 15504: Denominada como 'Software Process Improvement and Capability Determination' (SPICE) o en español, 'Determinación de la Capacidad de Mejora del Proceso de Software', define un modelo de evaluación de procesos y de mejora continua.
3. MoProSoft; Se define como un modelo de gestión procesos para la industria del desarrollo de software. Esta basado en ISO/IEC 9001:2000, ISO/IEC 15504-2:1998 y CMM: 1994. MoProSoft se enfoca en apoyar a las organizaciones en la normalización y mejora de sus prácticas, en la evaluación de su efectividad y la mejora continua; mejorando así su capacidad para ofrecer servicios con calidad y alcanzar altos niveles de competitividad.
4. UP: Definido como el proceso para el modelamiento del negocio y desarrollo de software iterativo e incremental, centrado en la arquitectura y con un enfoque en el riesgo. Elaborado a partir de las mejores prácticas de la industria de la ingeniería de software identificadas al momento de su propuesta [38]. Defne cuatro fases: Incepción, elaboración, construcción y transición.
5. RUP: Fue elaborado para ser un marco de trabajo para desarrollar proyectos de software, es una instancia del UP. En el manual de referencia [47], «IBM Rational Unified Process and Certification Guide: Solution Designer», sus autores lo definieron como un marco de trabajo para el desarrollo iterativo-incremental de software. Es una implementación propietaria de UP.
6. Enterprise Unified Process (EUP): Tiene su origen en RUP [5]. Es una propuesta para incluir los flujos empresariales de la organización, dado que el RUP solo abarca el proceso de desarrollo de software. Incluye dos nuevas fases: producción y retiro.
7. OpenUp: Tiene su origen en RUP. Es un proceso unificado bajo un enfoque ágil que sigue un proceso iterativo e incremental. OpenUP sigue un enfoque pragmático y ágil que se enfoca en un entorno colaborativo para desarrollo de software.
8. Agile Unified Process (AUP): Es una implementación simplificada de RUP. Enfatiza en la técnica de test-driven development (TDD) o desarrollo dirigido por pruebas.
9. Microsoft Solutions Framework (MSF): Es un marco de trabajo orientado al desarrollo de soluciones tecnológicas, en particular software, de manera más rápida, adaptable, con optimización de recursos y minimización de riesgos, y con resultados de más calidad. MSF incluye conceptos, modelos y buenas prácticas de uso que permiten realizar la planificación, realizar el desarrollo y la gestión de proyectos tecnológicos.
10. Programación Extrema (Extreme Programming - XP): Es posiblemente el método ágil más conocido y utilizado. El nombre se debe a que el enfoque fue desarrollado utilizando reconocidas buenas prácticas –como el desarrollo iterativo–, con la participación del cliente

en niveles 'extremos'. Su principal característica es la programación por pares.

11. SCRUM: Es un framework para el desarrollo de software bajo un enfoque ágil, en el cual el equipo de proyecto es auto organizado, autónomo y las fases del proyecto se desarrollan de forma solapada. El marco teórico es bastante conciso.

12. NTP ISO/IEC 12207: Fue elaborada por el Comité Técnico de Normalización de Ingeniería de Software y Sistemas de Información, y revisada y aprobada por la Comisión de Reglamentos Técnicos y Comerciales del INDECOPI, en su calidad de Organismo Nacional de Normalización. Es un marco de trabajo que cubre el ciclo de vida de desarrollo de software desde su formulación hasta su retirada. Define un conjunto de procesos para la adquisición y suministro de productos y servicios de software.

13. Kanban: Es una metodología para la gestión de procesos y mejora continua, la cual también se utiliza en proyectos de desarrollo de software. Su principal elemento es un tablero de trabajo, que permite a los integrantes del grupo de proyecto conocer el nivel de avance de sus tareas asignadas de manera visual. Es considerada una metodología ágil.

Los modelos de desarrollo seleccionados permiten la incorporación de una técnica de priorización de requisitos. En la presente propuesta no se especifica un modelo de desarrollo de software en particular; este deberá ser seleccionado por el equipo de desarrollo en base a las características del proyecto.

4.1.2 Algoritmo de la técnica: No todas las técnicas tienen un algoritmo formal que permita su implementación; de algunas de ellas solo se tienen descripciones breves. La ausencia de un algoritmo puede resultar en interpretaciones personales para su implementación, lo que genera resultados distintos, imprecisos o incluso erróneos. Las técnicas tienen diferentes niveles de facilidad de implementación [46]. Cada técnica tiene sus fortalezas y debilidades, y resulta dificultoso elegir una. Algunos métodos consumen mucho tiempo, pero proporcionan resultados más precisos. Otros se pueden usar con una gran cantidad de requisitos, pero proporcionan resultados menos precisos. Sin embargo, se debe seleccionar un método que sea más adecuado para cada situación. [32].

Propuestas similares la encontramos en [48] 'Effectiveness of Requirement Prioritization Using Analytical Hierarchy Process And Planning Game: A Comparative Study', [28] 'Prioritization of Issues and Requirements by Cumulative Voting: A Compositional Data Analysis Framework'. Frente a [64] en 'Distributed and Collaborative Requirements Elicitation based on Social Intelligence', en la cual los autores no recurren a las técnicas para implementar un modelo, pero, por lo mismo, no tienen una base teórica documentada y probada para su propuesta.

4.1.3. Licencia de software: La licencia de software compatible con el modelo propuesto puede ser GPL o BSD.

Tanto la herramienta colaborativa como el componente para la herramienta seleccionada deberán ser compatibles con las licencias identificadas, lo cual permitirá acceder a las mismas sin infringir las regulaciones, sino, se infringiría la Ley de derechos de autor.

Para la presente propuesta, se siguieron los criterios expuestos en el artículo [42] 'Determinants of the Choice of Open Source' los principales tipos de licencia para software libre y el criterio para su selección, además de recalcar su importancia de incluirlos en los proyectos de software.

4.1.4. Herramienta colaborativa: Diversos autores propusieron este tipo de software para la gestión de requisitos, o alguna de sus fases. Por ejemplo [17] 'Web-Based Focus Groups for Requirements Elicitation', [49] 'Distributed Requirements Elicitation Using A Spatial HypertextWiki', así como [52] 'The ABC-eBook System From Content Management Application to Mash-up Landscape'.

De acuerdo con datos analíticos recopilados en la web [11], los tres mayores CMS viables son: WordPress (1° puesto), Joomla (2°) y Drupal (6°), los cuales representan el 83.87% del total de CMS instaladas, basadas en esta tecnología. Otras distribuciones, como Squarespace (3°) es un CMS publicado como un SaaS (software como servicio); Telerik Sitefinity (4°) son suites de pago; por tanto, estas herramientas no cumplen con el tipo de licencia requerido por el modelo. Blogger (5° puesto) es un CMS gratuito, el cual solo permite agregar contenidos; sin embargo, no permite ampliar funcionalidades a través de componentes, también llamados plugins, ver Tabla 1, recopilada de [11].

Los criterios utilizados para seleccionar la herramienta CMS adecuada entre las candidatas son las siguientes:

1. total de descargas;
2. websites activos;
3. resultados en Google Trends;
4. temas gratuitos;
5. plugins gratuitos;
6. tiempo de instalación;

7. dificultad para programar plugins;
8. tiempo de respuesta;
9. tiempo de respuesta con caché; y
10. nivel de seguridad web.

Estudios [33] identificaron a los mejores CMS disponibles en el mercado, los mismos que son WordPress, Joomla y Drupal.

Tabla 1. *Lista de los veinte CMS más usado*

	Content Management System	Sitios activos	%
1.	WordPress	19,545,516	73.9196
2.	Joomla!	1,982,796	7.4988
3.	Squarespace	1,382,694	5.2292
4.	Telerik Sitefinity	1,204,145	4.5540
5.	Blogger	732,294	2.7695
6.	Drupal	647,479	2.4487
7.	CPanel	522,551	1.9762
8.	Google Search Appliance	180,797	0.6838
9.	BuddyPress	103,254	0.3905
10.	ExpressionEngine	43,885	0.1660
11.	vBulletin	21,863	0.0827
12.	MediaWiki	21,376	0.0808
13.	Kentico	19,911	0.0753
14.	HubSpot COS	16,521	0.0625
15.	Adobe CQ	16,512	0.0624
		26,441,594	83.87

4.1.5. Componente para la herramienta colaborativa (plugin en inglés): La implementación del algoritmo de priorización requiere su despliegue en una aplicación informática.

Por ejemplo, [2] desarrollaron una aplicación completa, elaborando una propuesta teórico y práctica, descrita en su artículo 'ReproTizer: A Fully Implemented Software Requirements Prioritization Tool'. A diferencia de los autores, el presente modelo busca extender las funcionalidades de una herramienta de software ya existente, así, se evita el desarrollar una aplicación completa, y solo se desarrolla las partes faltantes, las mismas que implementan el algoritmo de la técnica.

Es a través de los módulos de software con los que se puede extender la funcionalidad de las herramientas CMS. Estos no forman parte de la herramienta, sino que están diseñados para proveer de funcionalidades adicionales, lo que permite aplicaciones adicionales y minimiza el exceso de código que, de otra forma, saturaría a la herramienta CMS con un gran número de funciones que posiblemente no se utilicen y la harían innecesariamente compleja. Ofrecen características y funciones personalizadas que permiten al usuario manipular y personalizar la herramienta CMS, de acuerdo con sus necesidades específicas. Debido a que la implementación de un algoritmo de priorización de requisitos no es una funcionalidad estándar o muy común, será necesario que la herramienta CMS lo permita implementar a través de un componente. Las técnicas de priorización de requisitos preseleccionada deberán ser factibles de implementarse en un componente.

5. Evaluación

La validez del constructo se refiere a la relación entre la teoría y la observación de los resultados de su implementación [26] y [61].

La validación de los componentes del modelo propuesto se realizará mediante una prueba analítica que determine si cada uno de los elementos que la constituyen.

Con la finalidad de obtener mejores y más generales resultados, afirmaron que se debe ejecutar el experimento en un proyecto grande, de software real, y con un staff de profesionales. Se debe considerar que, al final, todas las validaciones se constituyen en un experimento.

El contexto del experimento puede ser descrito de acuerdo con las siguientes cuatro dimensiones:

1. offline vs. online;
2. estudiante vs. profesional;
3. simulado[1] vs. problema real;
4. especifico vs. general.

La estructura de la evaluación al constructo del presente trabajo de investigación puede ser descrito como:

1. offline: el experimento se relaciona directamente con un producto cuya evaluación deberá ser realizada por el mismo investigador, el cual no requerirá ninguna herramienta del tipo en línea.
2. estudiante o profesional: como participante del experimento el investigador puede ser estudiante o profesional, no afectándose los resultados del mismo.
3. problema real: la selección de una lista candidata se constituye en un problema real, aun cuando sus resultados estén orientados a un trabajo de investigación
4. específico: los resultados del experimento pueden ser aplicados en un contexto similar.

Se realizaron dos pruebas, una descriptiva, orientada a evaluar cómo sería la aplicación del modelo de obtención de una lista de técnicas candidatas; y otra de

1 La mayoría de los estudios [35] para realizar sus pruebas utilizan una lista de requisitos pre elaborada para ser utilizadas por los evaluadores durante su evaluación.

campo, orientada a verificar la validez del modelo propuesto en términos de viabilidad y facilidad de uso.

5.1. Prueba analítica

El objeto de la evaluación será verificar mediante dos casos de prueba que verificarán si el modelo propuesto es viable, es decir si permite identificar una técnica candidata válida para la priorización de requisitos, cumpliendo con las características preestablecidas y sustentadas en secciones anteriores.

La presente propuesta está orientada a obtener los resultados, es decir los elementos de la lista corta, de forma ágil, sin requerir destinar bastantes recursos expresados en tiempo y personal, lo contrario la haría una herramienta poco práctica y funcional.

Se selecciona a las técnicas 'Voto Acumulativo' y 'PAJ Jerárquico', se evalúan frente a cada uno de los elementos del modelo propuesto:

Voto Acumulativo:

1. Modelo de desarrollo de software: La técnica permite usar cualquier modelo de los propuestos, por ejemplo, RUP o Scrum.
2. Algoritmo de la técnica: La técnica cuenta con un algoritmo claro y definido, existiendo numerosos ejemplos en la bibliografía de su implementación, por lo tanto, se pueden obtener resultados precisos de su ejecución.
3. Licencia de software: La técnica no es propietaria, por tanto, puede usarse con licencias compatibles.
4. Herramienta colaborativa: La técnica no es propietaria, por tanto, puede ser desplegada en herramientas colaborativas compatibles con las licencias del modelo.
5. Componente para la herramienta colaborativa: La técnica puede ser desplegada en un componente, a través de la programación del algoritmo.

Por tanto, SI se incluye a la técnica como un ítem en la lista corta de técnicas candidatas para la priorización de requisitos.

PAJ Jerárquico:

1. Modelo de desarrollo de software: La técnica permite usar cualquier modelo de los propuestos, por ejemplo, ISO/IEC 15504.
2. Algoritmo de la técnica: La técnica no cuenta con un algoritmo, solo una descripción de su implementación.
3. Licencia de software: La técnica no es propietaria, por tanto, puede usarse con licencias compatibles.
4. Herramienta colaborativa: La técnica no es propietaria, por tanto, puede ser desplegada en herramientas colaborativas compatibles con las licencias del modelo.
5. Componente para la herramienta colaborativa: La técnica puede ser desplegada en un componente, a través de la programación del algoritmo.

Por tanto, NO se la incluye como un ítem más en la lista corta de técnicas candidatas para la priorización de requisitos que se está elaborando.

5.2. Prueba de campo

La prueba de campo evalúa que la propuesta sí permita realizar la tarea de obtención de técnicas candidatas, es decir, que sea viable. Se realizó un experimento y se tomó como muestra a ocho (08) profesionales de TI recién egresados, en su ambiente de trabajo. Por su formación, los participantes conocen los conceptos de ingeniería de requisitos y los distintos elementos del modelo por haberlo llevado en sus cursos de formación profesional, al igual que lo propuesto por [4]. A cada participante se le dio una lista de técnicas de priorización, extraídas del presente artículo. El experimento se estimó para una sesión, de una duración de cinco a diez minutos. A los participantes se les hizo una explicación verbal del mismo, se les entregó una breve descripción del modelo basado en el presente artículo, y una ficha impresa en la cual realizarían la elaboración de la lista de selección de técnicas candidatas.

Luego de realizada la prueba, se le consultó a los entrevistados su opinión acerca del proceso que han realizado, se les hizo un cuestionario de dos preguntas: 1) Viabilidad; y 2) sobre la facilidad de uso. Para la pregunta uno se tenía las alternativas Sí y No. Para la pregunta dos se aplicó una escala de valores de del 1 al 10 (ascendente positiva).

Para la pregunta uno se obtuvo un 100% de respuesta afirmativa, es decir, calificaron al modelo de viable.

Para la pregunta dos el puntaje promedio asignado por los participantes de la prueba fue de 8.2.

5.3. Resultados de las pruebas

De la observación de los resultados de ambas pruebas se concluye que el resultado de la evaluación es positivo.

6. Conclusiones

Se demostró la importancia de la priorización de requisitos en la realización de proyectos de TI y para la obtención de software de calidad.

Las técnicas para determinar las prioridades de los requisitos son de mayor importancia en el desarrollo de software, ya que permiten orientar el esfuerzo del equi-

po de trabajo, enfocándose primero en aquellos requisitos que tendrán mayor importancia para el éxito del proyecto.

Se realizó una identificación y descripción de las técnicas más importantes para la priorización de requisitos, obteniéndose una visión global del nivel de avance y desarrollo actual de las mismas.

Se determinó que es importante disponer de un modelo válido a nivel teórico, así como contar con un modelo que sea válido a nivel práctico; es decir, que también permita ser implementado en una herramienta de software, teniendo como respaldo una metodología y teniendo en consideración las licencias.

7. Recomendaciones

Con el fin de obtener un modelo o propuesta válida, esta debe ser verificada desde un enfoque teórico y práctico.

Se pueden continuar con investigaciones sobre cómo elaborar un modelo de elementos candidatos en otras áreas del conocimiento, además del desarrollo de software, la gestión de requisitos en general y la priorización de requisitos en particular es importante, por ejemplo, en la gestión de TI y otras áreas.

8. Referencias bibliográficas

[1] Abdullah Awais, M. (31 de Julio de 2016). Requirements Prioritization: Challenges and Techniques for Quality Software Development. *ACSIJ Advances in Computer Science: an International Journal, 5*(20), 14-21. Obtenido de http://www.acsij.org/acsij/article/view/458.

[2] Achimugu, P., Selamat, A., & Ibrahim, R. (2016). ReproTizer: A Fully Implemented Software Requirements Prioritization Tool. En *Transactions on Computational Collective Intelligence XXII* (Vol. 9655, págs. 80-105). Berlin, Alemania: Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-662-49619-0_5.

[3] Achimugu, P., Selamat, A., Ibrahim, R., & Naz'ri Mahrin, M. (11 de Febrero de 2014). A systematic literature review of software requirements prioritization research. *Information and Software Technology, 56*(6), 568–585. doi:http://dx.doi.org/10.1016/j.infsof.2014.02.001.

[4] Ahmad, A.; M. Goransson y A. Shahzad, *Limitations of the Analytic Hierarchy Process Technique with Respect to Geographically Distributed Stakeholders*, Proceedings of World Academy of Science, Engineering and Technology*,* 70(9), 97-103 (2010).

[5] Ambler, S. W. (07 de Setiembre de 2008). *Enterprise Unified Process (EUP).* Recuperado el 02 de Julio de 2009, de http://www.enterpriseunifiedprocess.com/.

[6] Arcus, M., *Business Case: How to work out Requirements?,* Tesis de Magister, Department of Computer Science, Universidad de Erlangen-Núremberg, Baviera, Alemania (2013).

[7] Azzolini, C. M., *Un Enfoque de Priorización de Requerimientos, a partir de la Segmentación de las Preferencias de los Stakeholders*, Tesis de Magister, Facultad de Informática, Universidad Nacional de La Plata, Argentina (2012).

[8] Benestad, H., & Hannay, J. (19 de Setiembre de 2012). Does the Prioritization Technique Affect Stakeholders' Selection of Essential Software Product Features? *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, 261-270. doi:http://dx.doi.org/10.1145/2372251.2372300.

[9] Berander, P., *Evolving Prioritization for Software Product Management,* Tesis de Doctorado, Department of Systems and Software Engineering, Blekinge Institute of Technology, Karlskrona, Suecia (2007).

[10] Berntsson Svensson, R.; T. Gorschek y otros cinco autores, *Prioritization of Quality Requirements: State of Practice in Eleven Companies*, Requirements Engineering Conference (RE), 2011 19th IEEE International, 69-78, Trento, Italia, 29 de Agosto (2011).

[11] BuiltWith® Pty Ltd. (04 de Marzo de 2018). *CMS technologies Web Usage Statistics.* Obtenido de https://trends.builtwith.com/cms.

[12] Carreón Suarez del Real, M. C. (25 de Junio de 2008). Construcción de un catálogo de patrones de requisitos funcionales para ERP. 125. (X. F. Carme Quer, Ed.) Catalunya, España: Universitat Politècnica de Catalunya - Departament de Llenguatges i Sistemes Informàtics. Obtenido de http://www.researchgate.net/publication/39432525_Construccin_de_un_catálogo_de_patrones_de_requisitos_funcionales_para_ERP.

[13] Chaffey, D. (1998). *Groupware, Workflow and Intranets.* Boston, Massachusetts, United States of America: Digital Press.

[14] CMMI Product Team. (2010). *CMMI for Development.* Pittsburgh, USA: Carnegie Mellon University. de http://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf.

[15] de Vreede, G.-J., & Briggs, R. O. (03-06 de Enero de 2005). Collaboration Engineering: Designing Repeatable Processes for High-Value Collaborative Tasks. *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences*(ISBN: 0-7695-2268-8; ISSN: 1530-1605), 1-10. doi:10.1109/HICSS.2005.144.

[16] Diogo, D., Carla, F., Miguel, M., & Alberto, R. (Junio de 2012). Collaborative Requirements Elicitation with Visualization Techniques. (IEEE, Ed.) *IEEE 21st International Workshop*, 343-348. doi:10.1109/WETICE.2012.14.

[17] Farinha, C.; y M. Mira da Silva, *Web-Based Focus Groups for Requirements Elicitation*, ICSEA 2011: The Sixth International Conference on Software Engineering Advances*,* 504-509, Barcelona, España, 23 de octubre (2011).

[18] Favela, J., & Peña-Mora, F. (Marzo/abril de 2001). An Experience in Collaborative Software Engineering Education. *IEEE Software, 18*(2), 47-54. doi:http://dx.doi.org/10.1109/52.914742.

[19] Fitsilis, P.; G. Vassilis; A. Leonidas y I.K. Savvas, *Supporting the Requirements Prioritization Process Using Social Network Analysis Techniques*, Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*,* 110-115 (2010).

[20] Hankhar, Ritu y D.N. Singh Gill, *A Comparison among Various Techniques to Prioritize the Requirements*, International Journal of Computer Science and Management Studies, 12(3), 601-607 (2012).

[21] Imran Babar, M.; M. Rarnzan y S.A.K. Ghayyur, *Challenges and Future Trends in Software Requirements Prioritization*,

doi: https://doi.org/10.1109/ICCNIT.2011.6020888, Computer Networks and Information Technology (ICCNIT), 319-324 (2011).

[22] Johnson-Lenz, P., & Johnson-Lenz, T. (abril de 1990). Rhythms, Boundaries, and Containers: Creative Dynamics of Asynchronous Group Life. *Awakening Technology Research Report*. Recuperado el 20 de marzo de 2016, de http://nexus.awakentech.com:8080/at/awaken1.nsf/UNIDs/CFB-70C1957A686E98825654000699E1B?OpenDocument.

[23] Jones, C. (2008). *Estimación de costos y Administración de proyectos de Software. Dando realismo a la estimación* (Segunda edición ed.). (F. Castellanos Rodríguez, Ed., & V. C. Juan, Trad.) México D.F., México: McGraw-Hill Educación.

[24] Khari, M. y N. Kumar, *Comparison of Six Prioritization Techniques for Software Requirements*, Journal of Global Research in Computer Science, 4(1), 38-43 (2013).

[25] Konaté, J., El Kader Sahraoui, A., & Kolfschoten, G. L. (11 de Mayo de 2013). Collaborative Requirements Elicitation: A Process-Centred Approach. Group Decision and Negotiation, 3(4), 847-877. doi:0.1007/s10726-013-9350-x.

[26] Kyosev, T., *Comparing Requirements Prioritization Methods in Industry: A study of the Effectiveness of the Ranking Method, the Binary Search Tree Method and the Wiegers Matrix,* Tesis de Maestría, Master Program, Universiteit Utrecht, Utrecht, Holanda (2014).

[27] Lanubile, F., Eber, C., Prikladnicki, R., & Vizcaíno, A. (25 de Febrero de 2010). Collaboration Tools for Global Software Engineering. *IEEE Software*, 52 - 55. doi:http://dx.doi.org/10.1109/MS.2010.39.

[28] Lefteris Angelis, P., Rovegård, P., & Claes, W. (1 de Setiembre de 2010). Prioritization of Issues and Requirements by Cumulative Voting: A Compositional Data Analysis Framework. *Software Engineering and Advanced Applications (SEAA)*, 361-370. doi:http://dx.doi.org/10.1109/SEAA.2010.35.

[29] Ling Lim, S., Damian, D., & Finkelstein, A. (21-28 de Mayo de 2011). StakeSource2.0: using social networks of stakeholders to identify and prioritise requirements. *Software Engineering (ICSE), 2011 33rd International Conference*, 1022 - 1024. doi:http://dx.doi.org/10.1145/1985793.1985983.

[30] Lorentz, J., *The Effect of Requirements Prioritization on Avionics System Conceptual Design*, George Washington University, Washington D.C., USA (2009).

[31] Lucena, D.A. y R.G. Caballero, *ECM/CMS: Content Managements*, pp. 46, lulu.com, Alcalá de Henares, España (2011).

[32] Mustafa, B.A. y A. Zainuddin, *An Experimental Design to Compare Software Requirements Prioritization Techniques*, *doi: http://dx.doi.org/10.1109/ICCST.2014.7045010*, Computational Science and Technology (ICCST), 38(1), 1-5, (2014).

[33] Nicolás Ros, J., *Una propuesta de gestión integrada de modelos y requisitos en líneas de productos software*, Tesis de Doctorado, Departamento de Informática y Sistemas, Universidad de Murcia, Murcia, España (2009).

[34] Patel, S. K., Rathod, V. R., & Prajapati, J. B. (1-2 de Marzo de 2013). Comparative Analysis Of Web Security In Open Source Content Management System. *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*, 344-349. doi:http://dx.doi.org/10.1109/ISSP.2013.6526932.

[35] Pergher, M., & Rossi, B. (30 de Setiembre de 2013). Requirements Prioritization in Software Engineering: A Systematic Mapping Study. *2013 3rd International Workshop on Empirical Requirements Engineering (EmpiRE)*, 40-44. doi: 10.1109/EmpiRE.2013.6615215.

[36] Pineda, L.E., *Metodología para la Elicitación de Requerimientos de Software Basada en el Marco Lógico*, Tesis de Magister, Escuela de Postgrado, Universidad Tecnológica Nacional, Buenos Aires, Argentina (2013).

[37] Pressman, R. S. (2010). *Ingeniería de Software. Un enfoque práctico* (Setima Edición ed.). (R. Pablo, Ed., C. Víctor, & E. Javier, Trads.) Santa Fé, México D. F., México: McGraw-Hill/Interamericana Editores, S.A. de C.V.

[38] Prince, R. (9 de Diciembre de 2005). *Using RUP/UP: 10 Easy Steps.* Recuperado el 13 de Junio de 2015, de http://www.x-tier.com/public/RUPUPIn10EasySteps.doc.

[39] Rima Vyomeshbhai, S., *Building a Web Content Management System*, Tesis de Magister, Master of Computer Science Program, San Diego State University: California, USA (2012).

[40] Rinkevics, K. y R. Torkar, *Equality in cumulative voting: A systematic review with an improvement proposal*, Information and Software Technology*,* 55(2), 267-287 (2013).

[41] Rosyid, H.; E. Prasetyo, A. Hidayat Jatmika y D.O. Siahaan, *Comparison Analysis of Requirement Prioritization Methods Between Case Based Ranking and Cumulative Voting*, Konferensi Nasional Sistem dan Informatika*, ,* Bali, Indonesia, 22-27, 12 Noviembre (2011).

[42] Schwalbe, K. (2013). *Information Technology Project Management, Revised* (7th Edition ed.). (E. Joyner, Ed.) Boston, MA, USA: Cengage Learning.

[43] Sen, R., Subramaniam, C., & Nelson, M. (2014). Determinants of the Choice of Open Source Software *License. Journal of Management Information Systems*, 25(3), 207-240. doi:10.2753/MIS0742-1222250306.

[44] Sevilla, G.; S. Zapata; E. Torres y C.A. Collazos, *Using Wikis as Collaborative Strategy to Support Software Requirements Elicitation*, Computing Colombian Conference (9CCC)*,* 54-61, 3 al 5 de mayo (2014).

[45] Seyff, N., Todoran, I., Caluser, K., & Singer, L. (24 de Abril de 2015). Using popular social network sites to support requirements elicitation, prioritization and negotiation. *Journal of Internet Services and Applications*. doi:http://dx.doi.org/10.1186/s13174-015-0021-9

[46] Sher, F.; Jawawi, D.N.; Mohamad, R., & Muhammad, I. (2014). *Requirements Prioritization Techniques and Different Aspects for Prioritization*, doi: http://dx.doi.org/10.1109/MySec.2014.6985985, Software Engineering Conference (MySEC), 2014 8th Malaysian, 31 - 36, Langkawi, Malasia, 23 al 24 de setiembre (2014).

[47] Shuja, A. K., & Krebs, J. (2008). *IBM Rational Unified Process Reference and Certification Guide—Solution Designer* (Primera Edición ed.). (T. Woodman, & E. Uffer, Edits.) Boston, Massachusetts, EE.UU.: IBM Press.

[48] Siddiqui, S.; D. Rizwan Beg y S. Fatima, *Effectiveness of Requirement Prioritization Using Analytical Hierarchy Process (AHP) And Planning Game (PG): A Comparative Study*, International Journal of Computer Science and Information Technologies*,* 4(1), 46-49 (2013).

[49] Solis, C. y N. Ali, «Distributed Requirements Elicitation Using A Spatial HypertextWiki,» *International Conference on Global*

Software Engineering, 237-247, Princeton, USA, 23 al 26 de agosto (2010).

[50] Sommerville, I., & Alfonso Galipienso, M. (2011). *Ingeniería del software* (Novena Edición ed.). (F. Hernandez Carrasco, Ed., M. I. Alfonso Galipienso, A. Batía Martínez, F. Mora Lizán, & J. P. Trigueros Jover, Trads.) Ciudad de México, México: Pearson Educación, S.A.

[51] Song, W. (Febrero de 2017). Requirement management for product-service systems: Status review and future trends. *Computers in Industry, 85*, 11-22. doi:http://dx.doi.org/10.1016/j.compind.2016.11.005

[52] Thomas, H.; W. Nagler y M. Ebner, *The ABC-eBook System From Content Management Application to Mash-up Landscape*, Proceedings of World Conference on Educational Media, Hypermedia and Telecommunications*,* 6015-6022, Graz, Austria, 7 de julio (2008).

[53] Tong, Z.; Q. Zhuang, Q. Guo y P. Ma, *Research on Technologies of Software Requirements Prioritization*, Communications in Computer and Information Science, vol. 426, DOI: 10.1007/978-3-662-43908-1_2, 9-21, (2014).

[54] Trias, F. (16-18 de Mayo de 2012). Building CMS-based Web Applications Using a Model-driven Approach. *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference*, 1 - 6. doi:http://dx.doi.org/10.1109/RCIS.2012.6240465

[55] Tuunanen, T. y G. Hena, *Utilization of Flow Concept for Digital Service Requirements Prioritization*, AIS Electronic Library (AISeL)*,* 11(167), 1-19 (2011).

[56] Ullah, A., & Lai, R. (01 de Enero de 2013). Requirements Engineering and Business/IT Alignment: Lessons Learned. *Journal of Software, 8*(1), 1-10. doi:10.4304/jsw.8.1.1-10

[57] UNC. (2013). Software Licensing. Recuperado el 17 de marzo de 2016, de https://its.uncg.edu/Software/Licensing/

[58] Van der Heijden, H. (2009). *Designing Management Information Systems* (1era ed.). New York, EEUU: Oxford University Press (ISBN-10: 0199546320).

[59] Vestola, M., *A Comparison of Nine Basic Techniques for* Requirements *Prioritization*. Helsinki University of Technology, (2010).

[60] Whitehead, J. (2007). Collaboration in Software Engineering: A Roadmap. En J. Whitehead, *Future of Software Engineering* (págs. 214-225). DC, Washington, USA: FOSE. doi:10.1109/FOSE.2007.4

[61] Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., & Wesslen, A. (2012). *Experimentation in Software Engineering* (Primera Edición ed., Vol. I). Berlin, Heidelberg, Alemania: Springer. doi:10.1007/978-3-642-29044-2

[62] Zahoor, F. y I. Sarwar Bajwa, *Automatic Extraction of Catchphrases from Software License Agreement*, Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 189-194, 26 al 27 de agosto (2014).

[63] Zi-Jing, J. The Analysis and Design of the Content Management System based on J2EE, *2009 International Conference on Signal Processing Systems,* 829 – 833, Singapore, Singapore, 15 al 17 de mayo 2009.

[64] Ziqiang Luo, B.W. y L. Peng, *Distributed and Collaborative Requirements Elicitation based on Social Intelligence*, 2012 Ninth Web Information Systems and Applications Conference*,* 127-130, Haikou, China, 11 de noviembre (2012).