Describiendo requisitos verificables.

MSc. Violena Hernández Aguilar[1]
Ing. Yadira Machado Peña[1]

[1]Universidad de las Ciencias informáticas (UCI)

RESUMEN

La cifra de proyectos que fracasan en el mundo debido a problemas con los requisitos es alarmante. Estudios realizados por el Standish Group y que han sido publicados desde 1994 en los Reporte CHAOS, analizaron el desarrollo de más de 40,000 proyectos llevados a cabo por empresas diferentes en los Estados Unidos, concluyeron que en 1994 del total de proyectos estudiados solo el 16% culminó con éxito y diez años después (2004) este porciento se elevó solamente al 34%. El estudio identificó como principales causas de los problemas:

- Requisitos deficientes.
- La planificación de agendas y estimaciones de costes no realizados en base a los requisitos.
- Deficiencias en la aplicación de procesos y desconocimiento del ciclo de vida del proyecto. (CHAOS, 2004)

En este trabajo se definen elementos a tener en cuenta para describir requisitos funcionales y no funcionales verificables durante las pruebas y se proponen listas de chequeo para su evaluación.

Palabras claves: Requisitos funcionales, requisitos no funcionales, requisitos verificables, pruebas.

ABSTRACT

Many software project fail due to requirement's problems. The Researches have been did and have been published by Standing Group in CHAOS reports since 1994, analyzed the development of more than 40,000 software project carried out in several United States´ Enterprise and concluded that: in 1994, 16 % project finished successful and ten year later this percent increase only to 34%. The study identified the major problems in:

- Deficient requirement.
- Planning and estimation of budge don't consider user requirements.
- Deficiency in the application of the process and unknowns about the Project lifecycle. (CHAOS, 2004)

In this investigation are defined some elements to consider when verifiable functional and not functional requirements are described; also check lists are proposed to evaluate verifiable requirement.
Key words: Functional requirement, No functional requirement.

1. INTRODUCCION

El flujo de trabajo de requisitos independientemente de la metodología que se utilice para desarrollar un software (ágil o robusta), es la guía para seguir la traza de los demás flujos que le suceden. Ellos dicen al cliente qué esperar, al programador cuál es el código, al escritor técnico qué es el documento, y al probador qué es la prueba. (Pressman, 2006)

Llegado el momento de llevar a cabo las pruebas al software, el documento de especificación de requisitos en todas sus variantes (ya sea como una lista de requisitos del usuario, requisitos del sistema, casos de uso o historias de usuarios, entre otros) es el documento de consulta obligatoria en este flujo. Si los requisitos están mal redactados, son ambiguos y difíciles de verificar es casi imposible que las pruebas cumplan su objetivo.

El resto de ésta investigación está organizada de la siguiente manera. En la sección 2 se muestra la sección de Trabajos Previos. La sección 3 describe la manera de redactar otras secciones de un paper. La forma de colocar los Experimentos y Resultados se encuentra en la sección 4. La Discusión de los Experimentos se muestra en la sección 5 y finalmente, la manera de redactar las conclusiones está en la sección 6.

2. TRABAJOS PREVIOS

Mucho se ha escrito sobre la importancia de especificar correctamente los requisitos de software y entre los autores más destacados se encuentran: Ian Somerville, Presuman e incluso metodologías de desarrollo como Rational Unified Process (RUP) dedican un flujo de trabajo a su captura. La IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) en su estándar 830 "Guía para la especificación de requisitos" identifica características que estos deben cumplir y la ISO 9126 se centra en los requisitos de calidad de software, sin embargo se abunda mucho en qué hacer para obtener buenos requisitos pero no en cómo lograr una buena descripción que permita su correcta verificación.
Los denominados requerimientos "SMART" de Mannion y Keepence, 1995 tienen las siguientes características: específicos, medible, mccesible, realizable. Para este enfoque es necesario aclarar que cuando un requerimiento no está correctamente especificado, es difícil saber si será accesible y realizable.

3. DESAROLLO

Durante el desarrollo de las pruebas tanto funcionales como no funcionales uno de los artefactos de entrada principales es la especificación de requisitos. Una prueba cumplirá en mayor o menor medida los objetivos con que fue planteada si durante su implementación puede verificar el cumplimiento de la mayor cantidad de los requisitos de software a comprobar.
En dependencia del tipo de prueba que se realice (funcional o no funcional) se comprueba el cumplimiento de uno u otro requisito.

4. REQUISITOS FUNCIONALES

Los requisitos se clasifican en funcionales y no funcionales. Los requisitos funcionales describen lo que el sistema debe hacer mientras que los requisitos no funcionales no se refieren directamente a las funcionalidades que el sistema debe cumplir, sino a las propiedades emergentes de este como: fiabilidad, rendimiento, usabilidad. (Sommerville, 2005)
Específicamente las pruebas funcionales se enfocan en verificar que cada requisito se corresponda directamente con una o varias funcionalidades del sistema. Los objetivos de estas pruebas son verificar la aceptación de los datos, el proceso, la recuperación y la implementación correcta de las reglas del negocio.
Para llevar a cabo este tipo de pruebas se utiliza generalmente el método de caja negra y el artefacto que guía el diseño e implementación de las mismas es la descripción de los requisitos funcionales, los cuales pueden documentarse como una lista de requisitos, como casos de uso o como historias de usuarios, independientemente de la metodología utilizada para documentar el sistema (ágil o robusta), desde el punto de vista de las pruebas los requisitos deben ser verificables.
Un requisito es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina puedan chequear que el software satisface dichos requisitos. (IEEE 830,1998).
Ante requisitos como:
El sistema debe ser capaz de insertar los datos relacionados con matricula docente de un estudiante.
Al probador le es imposible saber el alcance de la prueba a realizar porque no sabe que atributos están asociados a la matrícula, de ahí la importancia de describir requisitos verificables o probables. Para la describir los requisitos, estos se pueden agrupar en dependencia de las características del sistema a implementar.

5. FORMA DE AGRUPAR LOS REQUISITOS FUNCIONALES

Si la aplicación a desarrollar tiene distintas categorías de usuarios (roles), los requisitos del sistema pueden ser especificados por roles. (Dependiente, Gerente, Administrador). (IEEE 830, 1998)

Otra forma de agrupar los requisitos es atendiendo a los objetivos o servicios que se desea que ofrezca el sistema y que requiere una determinada entrada para obtener su resultado. Para cada objetivo o subobjetivo requerido del sistema, se detallarán las funciones que permitan llevarlo a cabo. (IEEE 830, 1998)
Finalmente los requisitos también pueden ser descritos por jerarquía funcional. La funcionalidad del sistema se especifica como una jerarquía de funciones que comparten entradas, salidas o datos del propio sistema. Para cada función y subfunción del mismo se detallará la entrada, el proceso en el que interviene y la salida. Normalmente este tipo de análisis implica que el diseño siga el paradigma de diseño estructurado. Por lo general éste sistema se utiliza cuando ninguno de los anteriores se puede aplicar. (IEEE 830, 1998)
Al describir cada requisito se debe tener en cuenta que:

- Los requisitos deben especificar el procesamiento de información de principio a fin.
- Para cada requisito se deben especificar los mecanismos de control asociados a él (mensajes de confirmación, mensajes de error, entre otros).
- Los requisitos se deben especificar de una función general a una específica (subfunciones). Las subfunciones deben incluir todas las posibles alternativas.
- Al especificarlos se deben tener en cuenta las entidades del mundo real que son reflejadas en el sistema. Por tanto, para cada objeto se detallan sus atributos y sus funciones.
- Los datos asociados a cada requisito deben aparecer en un diccionario de datos donde se especifique el tipo de dato, rango de valores que acepta, etc.

Sommerville propone un formato estándar para especificarlos y asegurar que todos se adhieran a él.

Requisito: Nombre del requisito. Descripción del requisito: Propósito del mismo. Entrada: Elementos de entrada al sistema. Procesamiento: Operaciones a realizar por el sistema. Salida: Salida del sistema. Precondición: Restricción del sistema para poder ejecutar el requisito. Poscondición: Restricción del sistema en el momento de finalizar la ejecución del requisito

Tabla1 Formato para especificar requisitos sugerido por Sommerville.

Otras buenas prácticas:

- Utilizar el lenguaje de forma consistente, existen requisitos deseables y otros sin los cuales el sistema no tiene razón de ser (obligatorios). Sommerville propone que los requisitos obligatorios se escriban en futuro simple y los deseables se escriban en futuro condicional.
- Resaltar el texto del requisito (con negrita, cursiva o color) para distinguir las partes claves del mismo.
- Evitar el uso de jerga informática.
- Si los requisitos son capturados como casos de uso (CU) también se tienen en cuenta reglas para su descripción, independientemente de que cada proyecto usa una plantilla diferente, para documentarlos generalmente tienen en cuenta los puntos que aparecen en la tabla siguiente:

Elementos del Caso de Uso	Descripción
Nombre	Un nombre apropiado para el caso de uso.
Actor que inicia	El rol que inicia el caso de uso.
Breve Descripción	Una breve descripción sobre el papel y el propósito del caso de uso.
Precondición	Una descripción textual que define cualquier restricción del sistema del momento de inicio del caso de uso.
Poscondiciones	Una descripción textual que define cualquier restricción del sistema en el momento de finalización del caso de uso.

Curso Normal de los Eventos	
Flujo Básico	Descripción textual de lo que el sistema hace de forma general con respecto al caso de uso (no cómo los problemas específicos son solucionados por el sistema). La descripción debe ser comprensible al cliente.
Flujo Alterno	A partir del flujo básico, se definen las posibles alternativas que pueden surgir a partir del flujo normal.
Sección	Si el caso de uso usa un patrón de tipo CRUD (create, read, update y delete), el cual se basa en la fusión de casos de uso simples para formar una unidad conceptual, uniendo las operaciones de crear, consultar, actualizar y eliminar elementos de una misma entidad de la base de datos en un solo caso de uso, cada operación será definida como una sección la cual contendrá un flujo básico y flujos alternativos. (Övergaard, 2004)
Curso Normal de los Eventos	
Flujo Básico	

Tabla 2. Aspectos a tener en cuenta al describir un CU.

El Flujo de Eventos es la parte más importante del caso de uso, el cual consta de dos partes, el flujo básico y los flujos alternos. El flujo básico debe cubrir lo que sucede normalmente cuando se realiza el CU. Los flujos alternos cubren el comportamiento de una opción o excepción relativa al comportamiento normal. Se puede pensar en los flujos alternos como desvíos del flujo básico de eventos.
Por tal razón es importante a la hora de describir el flujo de eventos, tener en cuenta algunos aspectos:
Escribir el CU como si se narrara un juego de pelota, es decir la interacción entre dos equipos, en este caso los equipos serían: Actor y Sistema, para que se ubique al lector en cuál es la acción de cada cual, y cómo responde el sistema ante un evento que desencadena el actor. (Cockburn, 2000)
Por lo anteriormente dicho, sería bueno que cada acción comience con el equipo que la realiza, o sea:

1. El actor solicita la modificación de su nombre.	2. El sistema verifica que este actor esté registrado.

Cuando se describe el flujo básico no se debe pensar en que existen flujos alternos, más adelante serán descritos en la sección Flujos alternos.
En la sección flujos alternos se describen todas las excepciones que existan, por muy evidentes que parezcan, pues los CU son una guía para los flujos de trabajos que suceden al suyo, e incluso para los stakeholder que no tienen por qué conocer exactamente todas las posibles alternativas del proceso.
No se deben mezclar elementos de interfaz en la descripción, pues se está describiendo el proceso, no diseñando la interfaz. Es importante ubicar al lector donde comienza y termina cada flujo (básico o alterno).
En el caso de los user story (historia de usuario) son los usuarios los que describen los requisitos que debe realizar el sistema, las historias de usuarios son escritas para ser implementadas en un tiempo de 1 a 3 semana como máximo, por lo que se recomienda que cuando una historia de usuario agrupe varias funcionalidades ésta sea dividida en varias. (Beck, 1999),(Beck, 2000).
Kent Beck en su libro "Planning extreme programming" recomienda que las buenas historias deban:

- Ser entendibles por el cliente.
- Entregar valor al cliente.
- Formarse de una o a lo más dos frases que describan algo importante para el cliente. (ej: "El sistema debe verificar la ortografía de todas las palabras ingresadas en el comentario")
- Mientras más corta, mejor.
- Ser independientes unas de otras. (Beck, 2000)

Para su descripción los proyectos que siguen la metodología XP tienen en cuenta los elementos que aparecen descritos en la tabla 3.

Historia de Usuario	
Número: Número de la historia	Usuario: Rol que lleva a cabo la funcionalidad en el sistema.
Nombre historia: Nombre de la historia, debe estar relacionado con el objetivo de la funcionalidad a implementar.	
Prioridad en negocio: Importancia para el negocio. (Alta / Media / Baja)	Riesgo en desarrollo: Probabilidad de que existan problemas técnicos en su desarrollo que influyan en su terminación en tiempo. (Alto / Medio / Bajo)
Tiempo estimado: Días o semanas que durará su implementación.	Iteración asignada: Número de la iteración en que se implementará la historia de usuario.
Programador responsable: Programador del equipo de desarrollo que la implementará	
Descripción: Descripción de la funcionalidad a realizar por la historia de usuario.	
Observaciones: Reglas asociadas a la historia.	

Tabla 3. Aspectos a tener en cuenta al describir una historia de usuario.

Para comprobar que los requisitos funcionales son verificables se proponen listas de chequeo en dependencia de la metodología de desarrollo utilizada para documentarlos.

6. REQUISITOS NO FUNCIONALES (RNF).

Los requisitos no funcionales son a menudo más difíciles de verificar porque la mayoría de las veces los usuarios declaran estos requisitos como metas generales tales como la facilidad de uso, la capacidad del sistema para recuperarse de fallos o respuestas rápidas al usuario. Estas metas imprecisas causan problemas a los desarrolladores porque dejan abierta diferentes formas de interpretación, lo que provoca discusiones una vez que el sistema se entrega. Un ejemplo clásico de un requisito no funcional de usabilidad imposible de probar es:
El sistema debe poseer una interfaz "amigable". ¿Qué significado tiene una interfaz amigable?
Los requisitos no funcionales tienen una implicación directa sobre la arquitectura y la calidad del sistema y tienen asociados un conjunto de pruebas no funcionales para probar:

- La carga que soporta el sistema.
- Hasta donde es seguro el sistema.
- La configuración de software y hardware soportada por el sistema, etc.

Por todo lo anterior algunos autores recomiendan que siempre que sea posible se redacten los requisitos no funcionales de manera cuantitativa para que se puedan probar de un modo objetivo y se proponen algunas métricas para comprobar su verificabilidad. En la tabla 4 se muestran métricas propuestas por Sommerville.

Requisito no funcional	**Forma de Comprobarlo**
Rapidez	Transacciones procesadas por segundo. Tiempo de respuesta al usuario y a eventos Tiempo de actualización de las pantallas
Tamaño	KBytes Número de chips de RAM

Facilidad de uso	Tiempo de formación Número de páginas de ayuda
Fiabilidad	Tiempo medio entre fallos Probabilidad de no disponibilidad Tasa de ocurrencia de fallos Disponibilidad
Robustez	Tiempo de reinicio entre fallos Porcentajes de eventos que provocan fallos Probabilidad de corrupción de los datos después de fallos.
Portabilidad	Porcentaje de declaraciones dependientes de la plataforma Número de sistemas operativos sobre los que funciona Número de navegadores sobre los que funciona
Eficiencia	Tiempo de respuesta por transacción (promedio, máximo). Rendimiento (ej. transacciones por segundo, cantidad de datos que pueden ser transferidos en un segundo). Número de usuarios conectados en paralelo que el sistema permite. Número de transacciones en paralelo que el sistema puede alojar.

Tabla 4. Métricas para cuantificar requisitos no funcionales. (Sommerville, 2005)

7. ¿CÓMO VALIDAR QUÉ LOS REQUISITOS SON VERIFICABLES?

Para validar la verificabilidad de los requisitos se crearon listas de chequeo que abarcan los elementos antes mencionados. La estructura de la misma aparece en la tabla 5.

Estructura del documento					
Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios
Elementos definidos por la metodología					
Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios
Semántica del documento					
Peso	Indicadores a Evaluar	Eval	(NP)	Cantidad de elementos afectados	Comentarios

Tabla 5. Estructura de la lista de chequeo propuesta.

Las listas están enfocadas a tres áreas fundamentales:
Estructura del Documento: Abarca los aspectos relacionados con el formato del documento definidos con anterioridad por el proyecto.
Elementos definidos por la metodología: Abarca todos los indicadores a evaluar según la metodología seguida y el artefacto a revisar.
Semántica del documento: Contempla todos los indicadores a evaluar respecto a la ortografía y redacción.

Para cada área se elabora un grupo de preguntas que permitan verificar la correctitud y completitud del artefacto a revisar, en este caso la especificación de requisitos en cualquiera de sus variantes, los indicadores que se tuvieron en cuenta fueron:

Peso: Define si el indicador a evaluar es crítico o no.
Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.
Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.
Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.
Las preguntas relacionadas con la verificabilidad de los requisitos aparecerán como indicadores en el área: Elementos definidos por la metodología y serán elaboradas a partir de los criterios vistos anteriormente, ejemplo de estas preguntas pueden ser:
¿Aparece descrito el nombre de cada requisito?
¿Aparece descrito el propósito del requisito?
¿El nombre del requisito es intuitivo teniendo en cuenta su propósito?
Los indicadores dependen de los elementos tenidos en cuenta para especificar los requisitos en cada proyecto de desarrollo pero es una buena práctica elaborar las preguntas en base a las consideraciones antes mencionadas, porque el objetivo de esta investigación no es dar una lista de chequeo estática con los indicadores que los autores del mismo consideran sino dar elementos para que cada proyecto elabore sus propias listas de chequeo que le permitan comprobar que los requisitos especificados son verificables.
Para describir los errores o no conformidades (NC) se proponen los siguientes elementos a tener en cuenta:

- **Elemento:** Elemento de configuración (aplicación o documentación).
- **NC:** Descripción de la No Conformidad. (La aplicación debe ser fácil de usar no es un requisito probable).
- **Ubicación de la NC:** Lugar donde se encuentra la NC (Requisito 4).
- **Etapa de detección:** Etapa de desarrollo en que se detecta la no conformidad (Etapa de pruebas).
- **Significativa:** La NC es significativa si es de:
    - **Redacción.**
    - **Correspondencia con otra documentación.**
    - **Formato.**
    - **Error técnico.**
- **No significativa:** Si la NC no clasifica como significativa entonces es no significativa.
- **Recomendación:** Sugerencias del revisor para mejorar la calidad del documento.
- **Estado de la NC:** Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.
    - **RA:** Resuelta
    - **PD:** Pendiente
    - **NP:** No Procede
- **Respuesta del equipo desarrollo:** Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.

En la tabla 6 aparece la plantilla donde se describe cada elemento.

Elemento	NC	Ubicación de la NC	Etapa de detección	Significativa	No Significativa	Recomendación	Estado de la NC	Respuesta del Equipo Desarrollo

Tabla 6. Registro de defectos y dificultades detectados.

La lista de chequeo tiene asociado además un método para evaluar la calidad del artefacto revisado a partir de los resultados obtenidos en cada indicador y de la cantidad de no conformidades detectadas tal y como se muestra a continuación:
Se aborta la revisión del artefacto revisado si:

- El promedio de las No Conformidades críticas por casos de uso es superior a uno.

Este criterio es empleado únicamente para los casos de uso. En caso de otro artefacto la revisión se aborta si:

- Existen al menos dos indicadores críticos evaluados de mal.
- Existe más de una falta de ortografía por página o pantalla en caso de ser una interfaz.

- Incumple con más del 50 % de los indicadores a evaluar de la sección Estructura del Documento que posee la lista de chequeo.
- Se mantienen las No Conformidades de una revisión a otra.

Se evalúa de regular la calidad del artefacto revisado si el artefacto no cumple los criterios para ser abortado y:

- Existe una No Conformidad crítica.
- La cantidad de elementos afectados de un indicador evaluado de mal es superior a tres.
- Estos criterios se cumplen para todas las secciones que tiene la lista de chequeo.

El artefacto es evaluado de bien si no cumple ninguno de los criterios anteriores y:

- No existe ninguna No Conformidad relacionada con indicadores con peso crítico.
- Si la cantidad de elementos afectados de un indicador que no sea crítico no es mayor que dos.

8. EXPERIMENTOS Y RESULTADOS

Esta lista de chequeo ha sido aplicada durante las pruebas realizadas a 26 proyectos que desarrollan software en la Universidad que utilizan tanto metodologías ágiles como robustas, detectándose gran cantidad de no conformidades. La misma ha servido a los probadores en su mayoría estudiantes, para saber qué verificar y a los analistas de los proyectos de desarrollo de software para evaluar su descripción, permitiéndole subsanar errores.

Luego del uso de esta lista de chequeo el equipo de desarrollo le ha dado más importancia a los llamados requisitos de calidad que se encuentran incluidos dentro de los requisitos no funcionales y al personal de pruebas le ha resultado más factible la realización de las mismas. A partir de la cantidad de no conformidades detectadas se han impartido talleres a los proyectos, relacionados con la necesidad de describir requisitos verificables.

En la tabla 7 aparecen 13 de los proyectos revisados, el total de no conformidades detectadas y las causas que originaron las NC relacionadas con la metodología.

Proyecto	Artefacto revisado	Total de NC	Total de NC de la Metodología	Causa de las NC
1	Requisitos	12	4	Falta la descripción de los datos a introducir y los tipos de datos.
2	CU	27	9	Flujos alternos descritos dentro del básicos
3	CU	42	8	Problemas con la descripción de los flujos y entrada de datos.
4	Requisitos	10	6	Falta la descripción de los datos a introducir y los tipos de datos. y requisitos no verificables.
5	CU	28	10	Mala descripción de flujos incluidos y extendidos.
6	CU	64	48	Faltan los flujos alternos que describan la Validación de datos, secciones que deben ser flujos alternos.
7	Requisitos	24	10	Falta la descripción de los datos a introducir y los tipos de datos.
10	Requisitos	17	6	Historias de usuario poco especificadas.
11	Historias de usuario	15	7	Falta la descripción de los datos a introducir y los tipos de datos y requisitos no verificables o faltan.
12	Requisitos	9	3	Falta la descripción de los datos a introducir y los tipos de datos y requisitos no verificables o faltan.
13	Historias de usuario	8	5	Historias de usuario demasiado extensas.

Tabla 7. Proyectos revisados, principales no conformidades.

9. CONCLUSIONES

Luego de realizada esta investigación se concluye:
Los requisitos constituyen el punto de partida en el desarrollo del software y una mala especificación trae como consecuencia insatisfacciones para el cliente y para el equipo de desarrollo.
Describiendo requisitos verificables se garantiza el punto de partida para un proceso de pruebas exitoso.
En esta investigación se propone un conjunto de buenas prácticas para describir requisitos verificables en proyectos que usan metodologías de desarrollo tales como XP y RUP.

REFERENCIAS

Beck, K. (1999). "Extreme Programming Explained. Embrace Change", Pearson Education. Addison Wesley
Beck, K. (2000). "Extreme Programming Installed", Pearson Education, 2000. Addison Wesley
Chaos Report (2004)
Cockburn, Alistair, (2000). Writing effective use cases, Adisson Wesley.
IEEE (1998) Recommended Practice for Software Requirements Specifications IEEE Std. IEEE Std 830.06-1998
Övergaard Gunnar (2004). Use Cases Patterns and Blueprints. Addison Wesley Professional
Pressman, Roger S. (2005), Ingeniería de Software un enfoque práctico, 2006, 6ta edición, Mac Graw Hill.
Sommerville, Ian (2005). "Software Engineering, Fourth Edition". Addison-Wesley.2005

Autorización y Renuncia

Los autores authorizan a LACCEI para publicar el escrito en los procedimientos de la conferencia. LACCEI o los editors no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.