UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

MODELO CO-PARTICIPATIVO EN LA CONSTRUCCIÓN DE SOFTWARE PARA LA OBSERVACIÓN DEL DESEMPEÑO DOCENTE EN AULA

PRESENTADO POR:

MANUEL JESÚS IBARRA CABRERA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2018

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

MODELO CO-PARTICIPATIVO EN LA CONSTRUCCIÓN DE SOFTWARE PARA LA OBSERVACIÓN DEL DESEMPEÑO DOCENTE EN AULA

PRESENTADO POR:

MANUEL JESÚS IBARRA CABRERA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

APROBADA POR EL JURADO SIGUIENTE:

PRESIDENTE

.....
Dr. BERNABE CANQUI FLORES

PRIMER MIEMBRO

.....
Dra. JUANA IDELZA ZAVALETA GÓMEZ

SEGUNDO MIEMBRO

.....
Dr. YALMAR TEMISTOCLES PONCE ATENCIO

ASESOR DE TESIS

.....
Dr. VLADIMIRO IBAÑEZ QUISPE

Puno, 14 de diciembre de 2018

ÁREA: Ciencias de la computación
TEMA: Modelo del desarrollo de software
LÍNEA: Informática y sociedad

DEDICATORIA

- Gracias a Dios por darme sabiduría y entendimiento para alcanzar una meta más.
- Este trabajo es dedicado a mi esposa Moraima y a mis hijos: Camila, Daniela, Nicolás y Diego, quienes son la fuerza e inspiración para concluir con este trabajo de investigación.

AGRADECIMIENTOS

- Agradezco de manera especial a mi asesor Dr. Vladimiro Ibáñez Quispe por apoyarme en esta investigación.

- Agradezco también a todas las personas, que con su experiencia en proyectos de software, aportaron esta tesis.

- Finalmente, agradezco los Directores, Especialistas y Profesores a la Dirección Regional de Educación de Apurímac, por haber permitido llevar adelante y así poner en práctica este modelo co-participativo en la creación de proyectos de software.

ÍNDICE GENERAL

Pág.

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ANEXOS

Pág.

RESUMEN

Hoy en día, en la construcción de software existen varias razones por las cuales el software falla: requisitos poco entendidos al inicio, escasa participación de los clientes/usuarios en el proceso de desarrollo de software, requisitos cambiantes en el tiempo, entre otros. Este trabajo propone un modelo co-participativo en el proceso de construcción de software, que a diferencia de otros modelos, el cliente es parte del equipo de desarrollo y se involucra activamente durante todo el proceso: análisis, diseño, programación e implantación; esto permite el desarrollo del software en los tiempos establecidos y conforme a los requisitos solicitados por el cliente. El objetivo del trabajo de investigación es proponer el "modelo co-participativo" para construcción de software y para poner a prueba el concepto, se desarrolló un proyecto de software para optimizar el monitoreo de las sesiones didácticas a los profesores del nivel primaria y secundaria de los colegios de la región Apurímac del 2018. La investigación concluye en que, que para superar las dificultades en la construcción de software el cliente debe participar en forma activa durante todo el proceso de desarrollo de software. Por otro lado un 66.67% de los encuestado manifiesta que utiliza metodología XP o SCRUM en el proceso de desarrollo de software y un 33.33% todavía utiliza modelos tradicionales como Cascada, Iterativo o crean su propio modelo de trabajo; los proyectos que mayormente se desarrollan en Apurímac y Cusco son de escala pequeña o mediana y tienen una corta duración de menos de 7 meses.

Palabras clave: modelo co-participativo, software, cliente, usuario y SIREMAP.

ABSTRACT

Now a days, in software development there are several reasons why software fails: requirements lightly understood at the beginning, short-time participation of customers/users in the software development process, requirements changing over time, among others. This research work proposes a co-participatory model in the software development process, unlike other models, the client is part of the development team and is actively involved throughout the process: analysis, design, programming and implementation; this allows the software development in the established time and according with the requirements requested by the client. The objective of this research work is to propose the "co-participatory model" for software construction and to validate the model concept, a software project was developed to optimize the monitoring of teacher's sessions class at primary and secondary levels of the schools of the Apurímac region in 2018.The research concludes that, in order to overcome the difficulties in software construction, the client must participate actively during the entire software development process. On the other hand, 66.67% of the respondents answered that they use the XP or SCRUM methodology in the software development process and 33.33% still use traditional models such as Waterfall, Iterative or create their own work model; the projects that are mostly developed in Apurimac and Cusco are small or medium scale and have a short duration of less than 7 months.

Keywords: co-participative model, software, customer, user and SIREMAP

INTRODUCCIÓN

Hoy en día el área de Ciencias de la Computación se ha convertido en un área muy importante y transversal a otras áreas, de tal forma que se pueden resolver problemas de la sociedad mediante la construcción de software que permita automatizar procesos y ayudar en la toma de decisiones. En el caso especial del sector educación, el software debe ayudar a optimizar procesos, a mejorar la gestión pública y a obtener información en el momento oportuno, en forma consistente y confiable, para de esta forma tener un objetivo final: mejorar la calidad de aprendizaje del alumno.

El problema fundamental que aborda esta investigación es la escasa participación del cliente/usuario durante el proceso de desarrollo de software; por tanto, en este trabajo se analizan los problemas existentes y luego se propone el modelo co-participativo, entendiendo que "co-participativo" implica que el cliente y/o usuario participa durante todas las fases de construcción del software, de tal manera que el software se realiza en dentro del tiempo establecido, cumpliendo los requisitos solicitados y sobre todo con la satisfacción del cliente.

Este trabajo de investigación está estructurado de la siguiente manera: el Capítulo I describe la revisión de la literatura, analizando el marco teórico relacionado a la problemática de la ingeniería de software, el análisis los modelos de desarrollo de software y los trabajos relacionados más importantes para esta investigación. El capítulo II menciona el planteamiento del problema, la postura epistémica, las preguntas de investigación, la justificación y los objetivos de esta investigación. El capítulo III describe la metodología utilizada en esta investigación, el acceso al campo describiendo los procedimientos formales e informales del estudio, luego se analiza la selección de informantes y situaciones que fueron observadas, luego la estrategia de recojo, registro y análisis de datos. El capítulo IV muestra los resultados obtenidos, analizando, describiendo e interpretando la información obtenida en esta investigación, explicando las coincidencias y diferencias con los resultados de los estudios anteriores.

CAPÍTULO I

REVISIÓN DE LA LITERATURA

1.1 Contexto y marco teórico

1.1.1 Enfoque Co-participativo

Aparentemente los términos “colaborativo” y “co-participativo” son similares; sin embargo, existen diferencias entre ambos términos, especialmente cuando se trata de vincular estas palabras con el “trabajo en equipo” y que esté relacionado con la generación de un producto.

El aprendizaje colaborativo y aprendizaje cooperativo, son dos términos que se a veces se usan indistintamente en el ámbito educativo. La palabra “Colaborativo” es un adjetivo que implica *trabajar en grupo de dos o más personas* para conseguir un objetivo respetando el aporte de cada uno de ellos; la palabra “co-participativo” es un adjetivo que significa *trabajar y aprender en conjunto* para conseguir un objetivo en común, sin poner énfasis en el aporte individual, sino más bien en el aporte colectivo (McInnerney & Roberts, 2004; Schöttle, Haghsheno, & Gehbauer, 2014); en ambos casos implica tener habilidades sociales, habilidades de pensamiento, conocimientos y cumplimiento del trabajo en forma oportuna (Davidson & Major, 2014).

Por otro lado, los colaboradores pueden trabajar en *diferentes áreas* y en forma serial (uno seguido de otro y demora más tiempo); mientras que los co-participativos trabajan *en el mismo espacio geográfico*, realizan ajustes en caso de ser necesarios, comparten su trabajo y lo realizan el mismo trabajo en tiempos menores (Ashkenas, 2015). Asimismo, el trabajo colaborativo consiste en participar en un equipo de trabajo que tiene un *objetivo en común*, en la cual se respeta al aporte individual de

cada uno de los participantes; asimismo, el trabajo cooperativo, consistiría en el trabajo en conjunto para cumplir con un *objetivo compartido* (Davidson & Major, 2014)(Halynska, 2017).

Después de analizar los conceptos teóricos de "colaborativo" y "co-participativo", para el caso de la ingeniería de software podemos estructura algunas características que permitan justificar el uso apropiado de cada uno de los términos, la Tabla 1 muestra las diferencias entre ambos términos:

Tabla 1
Características del trabajo colaborativo y co-participativo de los clientes o usuarios en la ingeniería de software

		Usuario Colaborativo	Usuario Co-participativo
Característica	1.-Apoya durante la construcción del software	Esporádicamente	Permanentemente
	2.-Compromiso con el proyecto	Cumple con su tarea de responder a las preguntas que le hacen	Siente que tiene un objetivo compartido con el equipo de desarrollo de software[1]
	3.-Espacio geográfico de trabajo	Apoya desde su oficina	Comparte el mismo espacio geográfico con el equipo de desarrollo del software
	4.-Análisis del sistema	Apoya en casos concretos	Participa durante todo el proceso de análisis
	5.-Diseño del sistema	Menciona sus preferencias por el diseño	Diseña las interfaces en coordinación con el equipo de desarrollo de software
	6.-Programación	No participa	Participa esporádicamente en validaciones
	7.-Pruebas	No participa	Realiza las pruebas de "caja negra" del sistema

Por otro lado, existen cambios en la forma de aprender y enseñar de los alumnos y los docentes en el ámbito de la educación superior universitaria. Antes el docente asignaba un *"trabajo individual"* a los alumnos; hoy en día se suele utilizar bastante el término "*trabajo en grupo*" (cada uno colabora); y se prevé que en un futuro el término utilizado será *"co-aprendizaje"* (cada uno co-participa en el aprendizaje); estos son los cambios educativos que permitirán adquirir en los alumnos autonomía para el aprendizaje y confianza en resolución de problemas futuros no experimentados con anterioridad (Universidades Chilenas, 2010).

[1] Equipo de desarrollo de software se refiere al conjunto de personas que tienen un rol en el proyecto: analista, diseñador, programador, probador y jefe de proyecto.

1.1.2 Modelo co-participativo.

a) Elementos del modelo co-participativo

El diseño co-participativo explora las condiciones para la participación de los usuarios analistas, diseñadores y jefes de proyecto en el diseño del sistema de información basado en computador o en cualquier otro contexto de trabajo (Carroll & Rosson, 2007; Loup-Escande, Burkhardt, Christmann, & Richir, 2014). El enfoque participativo contribuye a las prácticas y dinámicas de acción que permiten hacer el diseño del producto con la participación de las partes interesadas.

b) Actores del diseño (usuario, diseñador, programador, partes interesadas)

Los usuarios, son las personas interesadas en el desarrollo del producto de software, ellos son los que sugieren para el contexto en el cual funcionará el sistema, indicarán los patrones culturales que se deben respetar, las restricciones del sistema, otros; son los usuarios más interesados en que el producto de software sea usable. Es el propio usuario el que crea sus artefactos de acuerdo a su contexto, es el que crea y modifica sus interfaces de acuerdo a sus necesidades, interactúa con el diseñador y el programador es encargado en analizar si es factible implementar el diseño propuesto.

Los diseñadores y programadores, son parte del equipo de desarrollo de software, que participan en el diseño, son los encargados de entender los intereses y objetivos del usuario y traducirlos en un artefacto, ellos utilizarán sus conocimientos y habilidades para que el producto final sea conforme a lo acordado con el usuario.

Las partes interesadas, son personas que individualmente o como representantes de grupos sociales tienen el interés en el desarrollo del producto o servicio, son personas conocedoras del contexto social, político, económico, otros y que pueden aportar en el diseño del producto para evitar conflictos de interés.

c) Oficinas co-participativas de diseño

La comunicación entre los actores del diseño es un factor muy importante, por tanto la oficina co-participativa es el lugar donde se realizará el diseño que debe

tener condiciones adecuadas, para asegurar la confianza y la predisposición para realizar un diseño creativo, colaborativo y participativo.

d) Artefactos, Sistemas y Métodos

La capa informal

Contiene aspectos generales tales como: la comunidad, las normas sociales, la política, la cultura, otros. Estos aspectos se pueden variar de acuerdo a los patrones de comportamiento de la sociedad e individuos. La interoperabilidad pragmática consiste en alinear los diferentes aspectos para resolver conflictos de cohesión (Liu, Li, & Liu, 2015) .

La capa formal

En esta capa se definen las funciones, los procedimientos, las reglas de negocio, entre otros, que especifican cómo deben realizarse las tareas. La interoperabilidad pragmática consiste en alinear los procedimientos y las reglas para lograr una mayor eficiencia (Liu et al., 2015). Define los objetivos de negocio, modela los procesos de negocio y trae la colaboración de las administraciones que tiene como objetivo intercambiar información y tener diferentes estructuras y procesos internos. Además, esta capa admite la capa superior atendiendo a los requisitos de la comunidad de usuarios, cómo hacer que los servicios sea disponibles, accesibles, identificables y orientados al usuario.

La capa técnica

Son los sistemas informáticos técnicos y sus funciones técnicas. Los sistemas y funciones se pueden programar de acuerdo con las normas y procedimientos. La interoperabilidad pragmática no se refiere directamente a esta capa, pero indudablemente requiere el apoyo continuo de la misma que consiste en alinear las funciones técnicas y los procesos comerciales para lograr una mayor productividad del sistema (Liu et al., 2015). Además, admite el intercambio de datos sin interrupciones, que es el intercambio automatizado de datos entre sistemas de información basados en un modelo de intercambio común. También cubre los problemas técnicos de vincular los sistemas y servicios informáticos.

Se incluyen algunos aspectos clave: servicios de interconexión, integración de datos, interfaz abierta, presentación e intercambio de datos, y la accesibilidad en este nivel.

1.1.3 Modelos de proceso software

Según Sommerville (2005), el modelo de proceso de software es *"Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real"*

Los modelos son abstracciones que sirven para explicar los diferentes enfoques del desarrollo de software (no son descripciones definitivas del proceso de software). Los modelos de proceso de software más importantes son: Codificar y corregir, Modelo en cascada, Desarrollo evolutivo, Desarrollo formal de sistemas, Desarrollo basado en reutilización, Desarrollo incremental, Desarrollo en espiral (Sommerville, 2005).

La construcción del software históricamente, ha tenido problemas asociados al diseño y construcción, entre ellos se destaca (Pressman, 2001):

- Los sistemas no responden a las expectativas de los usuarios.
- Los programas "fallan" con cierta frecuencia.
- Los costes del software son difíciles de prever y normalmente superan las estimaciones.
- La modificación del software es una tarea difícil y costosa.
- El software se suele presentar fuera del plazo establecido y con menos prestaciones de las consideradas inicialmente.
- Normalmente, es difícil cambiar de entorno hardware usando el mismo software.
- El aprovechamiento óptimo de los recursos (personas, tiempo, dinero, herramientas, etc.) no suele cumplirse.

Por otro lado, en el año 1996 según el estudio realizado por The Chaos Report elaborado por Standish Group Internactional y el Centro Experimental de Ingeniería

de Software (CEIS), se determinó que sólo un 16.2% de los proyectos de software que se realizaron en Estados Unidos fueron exitosos, es decir, concluyeron el proyecto propuesto, con los requisitos acordados, en los plazos determinados y con el presupuesto establecido. El 52.7% de los proyectos son completados, pero sobrepasan costos y plazos establecidos y cumplen parcialmente con los requerimientos acordados. Finalmente, el 31.1% de los proyectos son cancelados en alguna parte del ciclo de vida del desarrollo en forma parcial sin concluir. Algunas deficiencias comunes en el desarrollo de software son:

- Escasa o tardía validación con el cliente.
- Inadecuada gestión de los requisitos.
- No existe medición del proceso ni registro de datos históricos.
- Estimaciones imprevistas de plazos y costos.
- Excesiva e irracional presión en los plazos.
- Escaso o deficiente control en el progreso del proceso de desarrollo.
- No se hace gestión de riesgos formalmente.
- No se realiza un proceso formal de pruebas.
- No se realizan revisiones técnicas formales e inspecciones de código.

La detección de problemas en la construcción de software, se realizó en el año 1968, en la conferencia organizada por la Comisión de Ciencias de la OTAN en Garmisch (Alemania), en cuya conclusión, se determinó la "crisis del software" por la cual se pasaba. En esta conferencia, así como en la siguiente conferencia (y en la de Roma en 1969), se pretendía acordar las bases para una ingeniería de construcción de software. Lo que se necesitaba era "*establecer y usar principios de ingeniería orientados a obtener software de manera económica, que sea fiable y funcione eficientemente sobre máquinas reales*" (Naur & Randell, 1969).

La calidad es importante en el desarrollo de software, según un estudio realizado, los problemas más comunes para que el software sean deficientes son: requerimientos insuficientemente comprendidos 50%, diseño no comprendido o incorrectamente

trasladado de los requerimientos 30%, codificación (error de programación o diseño mal comprendido) 20%. Por otro lado, más de un 50% del personal técnico de software y un 70% de los directores de software tienen una formación deficiente en control de calidad (Lomprey & Hernandez, 2008).

El "*IEEE Standard Glossary of Software Engineering Terminology*" (Stad. 610.12-1990), ha desarrollado una definición más completa para ingeniería del software (Pressman, 2001): "Es la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software".

Pressman (2001) caracteriza la Ingeniería de Software como "una tecnología multicapa", ilustrada en la Figura 1.

Figura 1. Capas de la Ingeniería de Software.
Fuente: (Pressman, 2001)

Dichas capas se describen a continuación:

a) Un enfoque de calidad.- Cualquier disciplina de ingeniería (incluida la ingeniería del software) debe descansar sobre un esfuerzo de organización de **calidad**. La gestión total de la calidad y las filosofías similares fomentan una cultura continua de mejoras de procesos que conduce al desarrollo de enfoques cada vez más robustos para la ingeniería del software (Pressman, 2001).

b) Capa de proceso.- La capa de proceso se define como un marco de trabajo para un conjunto de áreas clave, las cuales forman la base del control de gestión de proyectos de software y establecen el contexto en el cual: se aplican los métodos y técnicas, se producen resultados de trabajo, se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente (Pressman, 2001).

c) **Capa de métodos.-** Los métodos de la ingeniería de software indican cómo construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Estos métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas (Pressman, 2001).

d) **Capa de herramientas**.- Las herramientas de la ingeniería del software proporcionan un soporte automático o semi-automático para el proceso y los métodos, a estas herramientas se les llama CASE (Computer-Aided Software Engineering) (Pressman, 2001).

A continuación se abordan los modelos de desarrollo de software más importantes

a) Modelo Cascada.

Uno de los primeros modelos de proceso de desarrollo de software, fue establecido por W. Royce en 1970 y es conocido como el "modelo de cascada" o en inglés "waterfall model", debido a que existe una cascada de una fase a otra (Sommerville, 2005).

El modelo en cascada es el primero en establecer el proceso de desarrollo como la ejecución de un conjunto de actividades. En su concepción básica, cada una de las actividades generan como salidas productos y modelos que son utilizados como entradas para el proceso subsiguiente, lo cual supone que una actividad debe terminarse (por lo menos, en algún grado) para empezar la siguiente.

Fases del modelo en cascada

Las fases del modelo en cascada se pueden visualizar en la Figura 2.

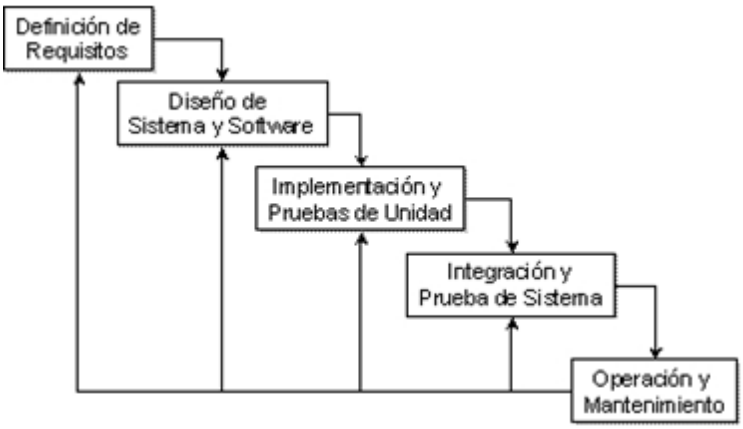


Figura 2. Fases del modelo Cascada

Fuente: (Sommerville, 2005)

Definición de requisitos. En esta fase, se obtienen los requisitos del sistema a implementar, se define las restricciones, las metas, los servicios y la interacción con otros sistemas. El analista se encarga de hacer la entrevista al usuario para poder definir los requisitos, el dominio, las funciones requeridas, el comportamiento, el rendimiento, interconexión y otros.

Diseño de sistema y software. El diseño del software transforma los requisitos obtenidos en la fase anterior en artefactos de hardware y software; es un proceso que se centra en cuatro atributos de programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmo). El proceso del diseño traduce los requisitos funcionales y no funcionales en una representación del software donde se pueda evaluar su calidad antes de comenzar la etapa de codificación.

Implementación y prueba de unidades. El diseño realizado en la fase anterior, se debe transformar en un conjunto de unidades o bloques de programas, esto se llama la codificación o programación en un lenguaje de programación específico. La prueba de unidades implica verificar que cada unidad cumpla con las especificaciones.

Integración y pruebas de sistema. Los programas o unidades de la fase anterior, en esta etapa se integran y prueban como sistema integral para verificar el cumplimiento de cada uno de los requisitos. Una vez que se ha generado el

código, comienzan las pruebas del sistema que consiste en probar los procesos lógicos internos del software, asegurando que todas las entradas produzcan las salidas esperadas. Después de realizar estas pruebas, el sistema se entrega al cliente.

Funcionamiento y mantenimiento. Es la fase en la que el sistema se instala y se inicia en marcha. El software sufrirá cambios después de ser entregado al cliente, por mencionar que el cliente requiere mejoras funcionales o de rendimiento, o requiere agregar nuevos requerimientos; esta etapa corresponde al mantenimiento del sistema.

Adicionalmente, los analistas como seres humanos, tienden a cometer errores en las diferentes etapas, que deben ser revisadas y repetidas en etapas posteriores para agregar mejoras al trabajo inicialmente imperfecto.

En términos generales el modelo en cascada tiene algunos problemas en la implementación que se describe a continuación:

Problemas del modelo cascada

- Los proyectos de software reales son dinámicos, que sufren variaciones hasta un momento en que se estabiliza; entonces el modelo secuencial no es el más adecuado, porque, los cambios pueden causar confusión porque el equipo está acostumbrado a terminar cada fase y no modificarlo en lo posterior.
- En la fase de análisis de requerimientos, es poco probable que el cliente exponga explícitamente todos los requisitos, por tanto el analista se imagina o trata de acercarse más a lo que el cliente quiere, pero no obtiene los requisitos completamente (es difícil leer la mente de los clientes o usuarios…).
- El cliente no puede ver el avance del producto, por lo que debe esperar la etapa final de operación y mantenimiento, lo cual es riesgoso, porque el producto podría requerir afinar requisitos o modificaciones que el cliente lo considere. Un error de gran magnitud puede ser desastroso, si no se detecta a tiempo hasta que el cliente revisa el programa.

- Cada integrante del equipo de desarrollo debe esperar que se culmine la fase anterior, entre tanto, no puede avanzar con su trabajo ni con la siguiente fase, lo cual genera retraso en la entrega del producto final.

b) Proceso Unificado de Rational (RUP)

El Proceso Unificado de Rational (RUP:Rational Unified Process), es un proceso para el desarrollo de software que fue creado por Rational Software, y posteriormente adquirida por IBM, con el cual adquiere otro nombre Irup y ahora es una abreviatura Rational Unified Process. RUP proporciona técnicas que miembros del equipo de desarrollo de software deben seguir rigurosamente, con el fin de aumentar su productividad en el proceso de desarrollo. Utiliza el Lenguaje de Modelado Unificado (UML), el cual define el flujo del proceso y los componentes que se utilizarán para construir el sistema y las interfaces que conectarán los componentes (Pressman, 2001).

Problemas de RUP

- Es un proceso considerado pesado y generalmente es aplicable a grandes equipos de desarrollo y grandes proyectos.
- Es un enfoque que requiere una documentación exhaustiva, en la que define minuciosamente cada fase, actividad y componente.
- Requiere un alto conocimiento de UML

c) Metodologías Ágiles

Un proceso es ágil cuando el desarrollo de software es incremental (entrega pequeña de software, con ciclos rápidos), cooperativo (cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación), sencillo (el método en sí mismo es fácil de aprender y modificar, bien documentado), y adaptable (permite realizar cambios de último momento) (Abrahamsson, Salo, Ronkainen, & Warsta, 2017).

Valores del Manifiesto ágil

Se tiene los siguientes valores (Beck, 2001)

- Individuos e interacciones sobre Procesos y herramienta.
- Software que funciona sobre Documentación exhaustiva.
- Colaboración con el cliente sobre negociación de contratos.
- Responder ante el cambio sobre seguimiento de un plan.

Principios de las metodologías ágiles

Se tiene los siguientes principios

- Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
- Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen las tareas.
- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
- El software que funciona es la principal medida del progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica enaltece la agilidad.

- La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.

1.1.4 Complejidad de un software

La Tabla 2, describe las características que tiene un producto de software poco complejo o muy complejo, está en base a nueve atributos que se seleccionaron para medir la complejidad de software.

Tabla 2
Nivel de complejidad del producto de software

	- Complejidad	+ Complejidad
1. Volumen de datos	Bajo	Elevado
2. Integración con otros sistemas	No requiere integrarse a otros sistemas	Requiere integrarse a otros sistemas
3. Reglas de negocio	Bien definidas	Reglas cambiantes
4. Seguridad	Poca seguridad	Mucha seguridad (bancos)
5. Transacciones	Pocas transacciones diarias	Muchas transacciones diarias
6. Tecnología a usar	Conocida	Desconocida
7. Número de integrantes del equipo de desarrollo	Hasta 20 desarrolladores	Más de 20 desarrolladores
8. Documentación	Escasa documentación	Documentación exhaustiva
9.Tiempo del proyecto	Menos de 3 años	Más de 3 años

Por ejemplo, si consideramos la primera característica denominada “Volumen de datos”, se considera que el producto de software es muy complejo (+complejidad) cuando el volumen de datos que se manejará es bastante elevado, unos cuantos millones de registros en las tablas más importantes en la base de datos; y se considera poco complejo (-complejidad) cuando el volumen de datos está en el orden de cientos o miles de registros para las tablas más importantes. Otro ejemplo, si consideramos la característica “reglas de negocio, se considera que el producto de software es muy complejo (+complejidad) cuando las reglas de negocio son bastante cambiantes o cuando las reglas de negocio no están bien definidas por el usuario, esto ocasiona que constantemente se debe cambiar las reglas al momento de la

programación del software; y se considera poco complejo (-complejidad) cuando las reglas de negocio están bien definidas o sufre cambios muy pequeños que no afectan la estructura principal de la programación.

Análisis de los modelos:

- Para proyectos de software muy complejos, muy grandes y muy costos, es recomendable usar el proceso de desarrollo unificado (RUP), porque, requieren estar bien definidos, bien detallados, bien documentados y con buen análisis y diseño de las reglas de negocio (requisitos bien definidos), de tal forma que el producto tendrá menor probabilidad de ser cambiado o modificado.
- Para el caso de proyectos pequeños, con pocos trabajadores, con reglas de negocio bien definidos, que no requiere una documentación exhaustiva, que manejan poco volumen de información y transacción, entonces es recomendable utilizar metodologías ágiles.
- Para el caso de proyectos medianos es mejor utilizar el modelo en cascada.

1.2 Antecedentes

Se considera a Andersen & Mørch (2016), quienes realizaron una investigación que describe el desarrollo de un proyecto web de gran escala en la que participaron usuarios finales, grupos interesados y desarrolladores de software, la participación fue dentro de una comunidad en forma virtual en la cual diseñaron y construyeron artefactos de software en forma compartida. Recopilaron datos de la interacción de los usuarios durante 6 meses, utilizaron el método de análisis de redes sociales y análisis de interacción.

Los autores propusieron un modelo de colaboración masiva para el desarrollo de software, basado en un modelo de colaboración de cuatro patrones que son: 1)control de acceso, control de excesiva información compartida, 2)construir puentes, difundir información a través de grupos en la red, 3)desarrollo general, permitiendo a los desarrolladores profesionales crear nuevas funcionalidades de software y actualizar el software existente, y 4)colaboración usuario-usuario, facilitando actividades de desarrollo no centralizadas, que van desde solicitudes de características hasta desarrollo local.

Una de las preguntas de investigación planteadas fue: "¿Cuáles son los patrones de interacción entre los usuarios finales y los desarrolladores profesionales en una comunidad de colaboración masiva, visto desde una perspectiva de desarrollo mutuo?" Los datos encontrados muestran que en el desarrollo de software usando una comunidad virtual, se obtiene mejor satisfacción con la participación del cliente y del equipo de desarrolladores.

Por otro lado, Bergadano F., Bosio G. y Spagnolo S. realizaron una investigación que destaca la interacción entre el cliente y los desarrolladores para realizar aplicaciones distribuidas. Los autores destacan que el proceso tradicional de desarrollo de software en una oficina centralizada está perdiendo progresivamente su atractivo y más bien hoy en día se tiene el desarrollo de software distribuido en múltiples sitios geográficos. Esto se debe principalmente a las ventajas bien conocidas de este último, como la mayor productividad y los menores costos. Sin embargo, esta práctica también tiene algunas desventajas que son inherentes a la distancia: las distancias geográficas, temporales y socioculturales entre las partes interesadas pueden afectar las actividades de comunicación, coordinación y control, lo que dificulta la colaboración. Esto no parecería de inmediato como el escenario ideal para aplicar metodologías ágiles, que definitivamente dependen de la colaboración continua entre todas las partes interesadas, incluido (con un papel muy importante) los clientes. Esta investigación analiza los problemas relacionados con la colaboración entre clientes y desarrolladores en un entorno ágil distribuido y propone un marco que define prácticas y herramientas para el manejo de información de proyectos y actividades de comunicación (Bergadano, Bosio, & Spagnolo, 2014).

Así mismo, Saiedian y Dale, (2000) realizaron una investigación relacionada a la ingeniería de requisitos en la cual destacan la interacción entre el cliente y los desarrolladores. La ingeniería de requisitos es uno de los pasos más cruciales en el proceso de desarrollo de software. Sin una especificación de requisitos bien escrita, los desarrolladores no saben qué compilar, los usuarios no saben qué esperar y no hay forma de validar que el sistema creado realmente cumple con las necesidades originales del usuario. Gran parte del énfasis en la atención reciente a una disciplina de ingeniería de software se ha centrado en la formalización de las especificaciones del software y su flujo hacia el diseño y la verificación del sistema. Sin lugar a dudas, la incorporación de dicha trazabilidad sólida, completa e inequívoca es vital para el éxito de cualquier

proyecto. Sin embargo, muchos proyectos fracasan incluso antes de que alcancen la etapa de especificación formal. Esto se debe a que, con demasiada frecuencia, el desarrollador no comprende ni aborda realmente los requisitos reales del usuario y su entorno. El propósito de esta investigación e informe fue investigar los actores clave y sus roles, junto con los métodos y obstáculos existentes en la especificación y priorización de Requisitos. El artículo se concentra en enfatizar las actividades y métodos clave para recopilar esta información, así como la participación del cliente para que colabore con el proceso de recolección de requisitos.

Por otro lado, Hassan y Martín Fernández, realizaron una investigación ponen énfasis en el Diseño Centrado en el Usuario para proyectos de software relacionados a la creación de sitios web. En este trabajo se ha presentado una propuesta de adaptación de la metodología de Diseño Centrado en el Usuario inspirada en el concepto de Diseño Inclusivo, resultado del análisis exhaustivo de la literatura científica existente (Hassan Montero & Martín Fernández, 2013).

Más allá de estrategias generales para el diseño de software, los desarrolladores web necesitan de metodologías específicas mediante las cuales puedan diseñar productos web usables y accesibles. El Diseño Centrado en el Usuario (DCU) viene a ser el conjunto metodológico en el que se asume que todo el proceso de diseño debe estar conducido por el usuario, sus necesidades, objetivos y características; si bien asume la necesidad participativa del usuario en el proceso de diseño, no representa en sí mismo un marco de trabajo con el cual pueda satisfacer las necesidades de usuarios con discapacidades. Las pruebas se realizaron con usuarios discapacitados. Por lo tanto, no se trata de un modelo metodológico que en sí mismo, y de forma aislada, permita guiar al diseñador para el desarrollo de sitios web usables y accesibles.

Así mismo, Perdomo, Cardozo, Perdomo y Serrezuela, realizaron una revisión de los sitios web diseñados con la participación del usuario. Los investigadores hacen una revisión sobre el diseño centrado en el usuario (UCD) o el desarrollo dirigido por el usuario (UDD) es un marco de procesos (no restringido a interfaces o tecnologías) en el que los objetivos de usabilidad, las características del usuario, el entorno, las tareas y el flujo de trabajo de un producto, servicio o proceso recibe una amplia atención en cada etapa del proceso de diseño. (Perdomo, Cardozo, Perdomo, & Serrezuela, 2017)

Del mismo modo, nos centramos en otros dos conceptos, como la usabilidad y la arquitectura de la información, es el diseño estructural de los entornos de información compartida; el arte y la ciencia de organizar y etiquetar sitios web, intranets, comunidades en línea y software para respaldar la facilidad de uso y la búsqueda; y una comunidad de práctica emergente enfocada en llevar los principios de diseño y arquitectura al panorama digital, como una herramienta de referencia para futuros investigadores.

Así mismo, Hussain, Slany y Holzinger, realizaron un trabajo de investigación en la cual buscan conocer el estado del arte del diseño para sistemas desarrollados con metodologías ágiles. Los autores mencionan que los métodos de desarrollo ágil de software son muy populares hoy en día y se están adoptando cada vez más en la industria cada año. Sin embargo, estos métodos aún carecen de conocimiento de usabilidad en su ciclo de vida de desarrollo, y la integración de usabilidad / Diseño Centrado en el Usuario (UCD) en métodos ágiles no se aborda de manera adecuada. Este documento presenta los resultados preliminares de una encuesta en línea realizada recientemente sobre el estado actual de la integración de métodos ágiles y usabilidad / UCD (Hussain, Slany, & Holzinger, 2009).

La encuesta fue respondida por 92 participantes en todo el mundo. Los resultados muestran que la mayoría de los profesionales perciben que la integración de los métodos ágiles con usabilidad / UCD ha agregado valor a sus procesos adoptados y a sus equipos; ha resultado en la mejora de la usabilidad y la calidad del producto desarrollado; y ha aumentado la satisfacción de los usuarios finales del producto desarrollado. Las técnicas HCI más utilizadas son la creación de prototipos de baja fidelidad, diseños conceptuales, estudios de observación de usuarios, evaluaciones de expertos en usabilidad, estudios de campo, personas, pruebas iterativas rápidas y pruebas de usabilidad en laboratorio.

Design thinking es un enfoque que está orientado al proceso cognitivo de los diseñadores, las por otro lado, se menciona que design thinking tiene como interés principal en identificar las estrategias mentales esenciales de los diseñadores mientas trabajan en un proyecto. Hoy en día, design thinking se entiende como *un proceso complejo de pensamiento para concebir nuevas realidades, que expresa la introducción de la cultura del diseño y sus métodos en campos como la innovación empresarial.* No

es solo un motor para la innovación promovido por los diseñadores, sino que ofrece nuevos modelos de procesos y herramientas que ayudan a mejorar, acelerar y visualizar todos los procesos creativos, realizados por diseñadores y por los equipos multidisciplinarios en un tipo de organización. (Tschimmel, 2012).

Por otro lado, design thinking describe la aparición del pensamiento de diseño en los negocios y se centra en los cuatro elementos clave del pensamiento de diseño: *el proceso iterativo, los equipos multidisciplinarios, el espacio creativo y la mentalidad del diseñador*; sin embargo, tiene la limitante de no tener un modelo de negocio formal o un plano para su implementación (Wölbling et al., 2012).

En los últimos diez años, los desarrolladores de software han inclinado su preferencia por el uso las metodologías ágiles. Aitken e Ilango (2013) analizaron las diferencias y semejanzas entre las metodologías tradicionales y las metodologías ágiles. Los autores mencionan es que usar metodologías tradicionales o metodologías ágiles tienen diferencias según el proyecto de software que se desee realizar, mientras que en las metodologías tradicionales hace énfasis en los procesos, la documentación, los contratos y el seguimiento a un plan; en las metodologías ágiles se prioriza la colaboración del cliente, el software funcional, la predisposición a los cambios y la interacción entre los miembros del equipo de desarrollo.

En las metodologías tradicionales se exige tener buena documentación, buen análisis para entender bien el problema a resolver y buen hacer un buen diseño, porque se considera que si todas las fases están bien definidas, entonces el producto total también debería de estar bien. Mientras que en las metodologías ágiles, considera que no es necesario documentar, sino es suficiente con describirlo verbalmente y anotarlos en las historias de usuario; más bien, exige que el cliente participe con mayor dedicación y a su vez se le va mostrando avances significativos y visibles de la construcción del software en tiempos muy cortos (Aitken & Ilango, 2013).

Recientes estudios dan a conocer que tanto metodologías tradicionales y metodologías agiles se pueden combinar y obtener un modelo híbrido. Según Theocharis y su grupo de investigación (Theocharis, Kuhrmann, Münch, & Diebold, 2015) publicaron un artículo científico titulado: *"Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices"*, el artículo describe que por muchos años, se ha reportado las bondades y beneficios del uso de las metodologías ágiles en el desarrollo

de software; sin embargo, el mundo no es blanco o negro; por tanto deviene una pregunta: que pasó como las metodologías tradicionales? han desaparecido? o han sido reemplazadas por las metodologías ágiles?. Después de haber realizado un análisis produndo, el trabajo de investigación concluye que ambos métodos pueden coexistir, a pesar que se asume el dominio de las metodologías ágiles, las empresas de software prefieren crear soluciones híbridas entre metodologías tradicionales y metodologías ágiles de acuerdo al contexto del trabajo.

Por otro lado, Pereira R., Baranauskas M. y Silva C. realizaron una investigación en la cual analizan el desarrollo del software con participación de la sociedad especialmente en software que esté orientado a la web. La posibilidad de desarrollar aplicaciones más interactivas e innovadoras llevó a una explosión en la cantidad de sistemas disponibles en la web en la que los usuarios interactúan entre sí y tienen un papel principal como productores de contenido: el llamado software social. Sin embargo, a pesar de su popularidad, pocos de estos sistemas mantienen una participación efectiva de los usuarios, promoviendo una interacción continua y productiva (Pereira, Baranauskas, & Silva, 2010).

Este documento examina el concepto de software social y analiza el software social Honeycomb, un marco para ayudar a comprender este tipo de sistema. Con base en el análisis de una red social inclusiva y en la revisión de la literatura, revisamos ese marco. Los autores argumentan que los valores deben considerarse en el contexto del software social y que el marco debe extenderse y fundamentarse teóricamente para abordar los diversos desafíos impuestos por el término "social".

Finalmente, el trabajo más cercano encontrado para esta propuesta es la que desarrollaron Rosa y Matos en Brasil, en la cual desarrollan un framework para el desarrollo de software educativo. Los autores manifiestan que Brasil tiene una gran diversidad cultural. Tiene efectos en escuelas fundamentales que se han convertido en entornos heterogéneos y multiculturales. En algunas escuelas brasileñas se utilizan diferentes tipos de tecnologías digitales que pueden apoyar los procesos de enseñanza y aprendizaje. Según la ingeniería y la semiótica, las tecnologías digitales son producto intelectual y cultural del diseñador. Luego, las diferencias culturales entre el diseñador y el usuario pueden influir en la calidad de la interacción (Rosa & Matos, 2016).

Este trabajo supone que la participación activa del usuario en el proceso de interacción puede contribuir a reconocer y considerar las diferencias culturales en el diseño de interacción de las tecnologías educativas. Este documento presenta un modelo co-participativo basado en la participación activa del cliente/usuario en todas las fases de la construcción de software.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1 Identificación del problema

Postura epistémica (aspecto filosófico)

Hoy en día, la sociedad contemporánea se caracteriza porque las personas tienen mayor grado de conocimiento respecto a las Tecnologías de Información y Comunicación (TIC). La cantidad de información que se genera y que está disponible digitalmente crece todos los días y los medios para difundir el conocimiento están disponibles a través de artefactos tecnológicos que la sociedad lo crea. Sin embargo, no todos son capaces de acceder, algunos autores indican que las barreras de acceso al conocimiento no son de naturaleza tecnológica, sino social y económica (Varian, 2005).

Por otro lado, las barreras de acceso al conocimiento en el mundo contemporáneo pasan por tres tipos de analfabetismo: literal, funcional y digital; y están presentes en las regiones socioeconómicamente desfavorecidas, una realidad que en muchos países en vías de desarrollo enfrentan, tal como sucede en el Perú.

Por tanto, los impedimentos para el acceso a las TIC no sólo son de naturaleza social y económica, sino también existe una responsabilidad de crear tecnología (hardware y software) adecuada y de calidad que ayude a revertir la situación. Entonces, es necesario que los sistemas de información sean diseñados de acuerdo al contexto social, respetando los patrones culturales y alineados al nivel económico (Pereira & Baranauskas, 2015), para que los usuarios finales se sientan satisfechos con la tecnología proporcionada, y tengan una sociedad más justa y abierta.

Es necesario que los actores involucrados participen activamente en el diseño de sistemas de información, para entender el contexto social sobre el cual funcionará el sistema, para que el producto se adapte a ellos. Esto está relacionado con el concepto de diseño como un proceso social de construcción conjunta de significados que materializan en un producto de software que mediará el acceso al conocimiento (Baranauskas, 2014).

El Diseño Centrado en el Usuario (DCU) es una filosofía en el que las necesidades y requerimientos del usuario final de la aplicación constituyen el foco fundamental de cada etapa del proceso de diseño del software. Por tanto, es importante involucrar al usuario en cada fase del proceso de desarrollo del producto de software, de tal forma que se garantiza que el producto final responda a sus necesidades y características. El DCU apunta a realizar un diseño pensando en y para el usuario, convirtiéndolo en el punto central del desarrollo. Se le implica hasta el punto de incluirlo y hacer que participe activamente durante el proceso, si cabe, como un miembro más del equipo de diseño y desarrollo del producto de software (Lowdermilk, 2013). El DCU surge como un enfoque y método que consiste en conocer algunas particularidades del usuario con el objetivo de hacer más familiares y efectivas las interfaces gráficas que se diseñan para él. (Galeano, 2017).

Antes, los productos de software eran construidos de forma estática, basadas en resolver problemas de cálculos matemáticos, o creación de funciones matemáticas para resolver problemas de la ingeniería; en este contexto el usuario final no tenía una participación activa; sólo explicaba el problema al inicio del proceso de desarrollo de software y luego el programador le entregaba el producto al final. En los últimos años esto ha cambiado drásticamente, los sistemas de información tienen que estar diseñados con interfaces de usuario que sean fáciles de usar, por lo que ha evolucionado notablemente el área de Interacción Humano Computador (en inglés HCI: Human Computer Interaction), para pensar en el desarrollo de sistemas interactivos a partir de la disciplina de “diseño” (Fallman, 2003; Sengers, Boehner, David, & Kaye, 2005; Winograd & Flores, 1986).

Por el diseño se entiende como un proceso social que involucra dos etapas: el entendimiento del problema y la solución propuesta para los diseñadores que son los responsables del diseño, sino que deben participar como los actores involucrados en el

proceso de desarrollo: analistas, diseñadores, programadores, usuarios finales, cliente, otros, para que el diseño realizado sea visto de diferentes puntos de vista y sean contrastados por cada uno de ellos. Los artefactos informales, formales y técnicos son utilizados como herramienta de comunicación y mediación entre los principales actores durante el proceso de desarrollo del sistema interactivo (Baranauskas, 2014; Baranauskas & Bonacin, 2008).

2.2 Definición del problema

Los modelos tradicionales para la construcción de software fueron creados con un enfoque lineal y secuencial; así por ejemplo, en lo modelo cascada tiene las siguientes fases: análisis, diseño, codificación, pruebas y mantenimiento (Pressman, 2001), cada una de las fases del desarrollo del producto de software debe ser completada antes de pasar a la siguiente, entonces la siguiente fase no puede iniciar mientras la anterior no haya concluido. El proceso de desarrollo de software no es un modelo lineal, sino que implica una serie de iteraciones de las actividades de desarrollo (Sommerville, 2005). La mayor desventaja del modelo de cascada es que el producto de software es visible para el usuario final recién en la última etapa del ciclo; es decir, no existe producto real y visible para el usuario durante el desarrollo, la elaboración del software no está en las manos del cliente; por lo tanto, bajo estas condiciones es poco probable que el diseño final sea exactamente lo que el cliente había pedido.

Por otro lado, se tiene la filosofía de las metodologías ágiles, las cuales dan la oportunidad "participación" al cliente y al desarrollo incremental del software con iteraciones muy cortas (Canós & Letelier, 2012). Este enfoque muestra ser efectivo para el caso de proyectos pequeños, que tienen complejidad baja y con requisitos muy cambiantes y cuando se requiere reducir los tiempos de desarrollo pero manteniendo la calidad del producto de software.

También se tiene el modelo de Proceso Racional Unificado (RUP: del inglés Rational Unified Process, desarrollado por la empresa Rational Software, actualmente propiedad de IBM), es un modelo utilizado para el análisis, diseño, implementación y documentación de sistemas orientados a objetos que usa el Leguaje Unificado para el Modelado (del inglés UML: Unified Modeling Language); además, de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante

cambios posteriores durante la construcción y el mantenimiento (Kruchten, 2004). El problema con este modelo es, que requiere de una documentación exhaustiva y se adecua para el caso de proyectos de gran complejidad y por ende son muy costosos.

El problema es que los modelos mencionados anteriormente, en su mayoría son antiguas y están quedando obsoletas; dichos modelos tienen escasa participación del usuario, por tanto el producto de software desarrollado, generalmente no está acorde con las necesidades que inicialmente requería el usuario. Finalmente, no existe un software que permita hacer seguimiento al desempeño docente en aula de acuerdo a las rúbricas emitidas por el Ministerio de Educación.

Las preguntas de investigación

- ¿Es posible diseñar e implementar un modelo co-participativo para la construcción de software orientado a observación del desempeño docente en aula?
- ¿Cuáles son las dificultades que tienen los desarrolladores en la construcción de proyectos de software actualmente?
- ¿Qué opinan los desarrolladores de software respecto a las metodologías tradicionales de desarrollo?

2.3 La intención de investigación

Ante los problemas anteriormente descritos para la construcción de software, el proyecto plantea un modelo co-participativo de los actores principales en la construcción de software de mediana complejidad y aplicado al monitoreo y la observación del desempeño docente en aula.

2.4 Justificación

Este trabajo de investigación es muy importante porque permite que las empresas que se dedican a la construcción de software puedan crear software de calidad dentro de los tiempos establecidos y con satisfacción del cliente. El aporte más importante de este trabajo es la propuesta de un *modelo co-participativo,* en la cual se ha podido demostrar que la participación del cliente en forma activa y durante todas las fases de desarrollo de software (análisis, diseño, implementación y pruebas) permite que el software desarrollado tenga

mejor calidad, se cumplan con los plazos establecidos y sobre todo que el cliente quede satisfecho con el producto.

2.5 Objetivos

2.5.1 Objetivo General

Proponer un modelo co-participativo para la construcción de software.

2.5.2 Objetivos Específicos

- Construir un software utilizando el modelo co-participativo, para apoyar en el monitoreo y acompañamiento del docente en aula, para contrastar que el modelo funciona apropiadamente.
- Determinar las dificultades que tienen los desarrolladores en la construcción de proyectos de software.
- Conocer los modelos de desarrollo de software con los que trabajan los desarrolladores de la industria de software.

CAPÍTULO III

METODOLOGÍA

3.1 Acceso al campo

Lugar de estudio

El estudio se realizó en el año 2018, con profesionales en proyectos de software en las regiones de Apurímac y Cusco. Ambas regiones cuentan con universidades que han formado profesionales en ingeniería informática y sistemas y que hoy en día se han convertido en empresarios de la industria del software. La industria del software de Perú es uno de los sectores que tiene crecimiento económico considerable en los últimos años. Según el presidente de ApeSoft (Apesoft, 2017), Juan José Miranda Del Solar, informó que la industria de software en el Perú representa actualmente 300 millones de dólares al año de los cuales se exportan 23 millones oficialmente. También menciona que existen otros 23 millones en "zona gris", es decir, no oficialmente registrados como tal. El crecimiento ha sido lento pero constante de un 8 a 10% en los últimos años (Software, 2018).

Métodos

El método a usar será "investigación-acción", debido a que se pretende realizar los siguientes pasos:

- Insatisfacción con el actual estado de cosas.
- Identificación de un área problemática;
- Identificación de un problema específico a ser resuelto mediante la acción;

- Formulación de varios objetivos
- Ejecución de la acción para cumplir con los objetivos
- Evaluación de los efectos de la acción
- Generalizaciones.

3.2 Selección de informantes y situaciones observadas

Población y muestra

Población

La población es de tipo censo, se consideró a 24 personas que trabajan con proyectos de software, son empresarios y profesionales que se dedican a la industria del software.

Muestra

12 personas que trabajan con proyectos de software, son empresarios y profesionales que se dedican a la industria del software.

3.3 Estrategias de recogida y registro de datos

Técnica 1: Grupo Nominal

La técnica de "grupo nominal" es una técnica que facilita la generación de ideas y el análisis de problemas, especialmente cuando el grupo a tratar es de carácter heterogéneo. Esta técnica es útil para los casos en que las opiniones individuales deben ser combinadas para llegar a decisiones las cuales no pueden o no conviene que sean tomadas por una sola persona (Aranda & Araújo, 2009).

Con esta técnica se puede identificar, analizar y jerarquizar los problemas, las causas, las controversias, y también dar soluciones a través de consenso en grupos o equipos de trabajo. La técnica de grupo nominal permite asegurar que cada uno de los participantes tenga la opción de dar sus ideas, sus opiniones y sus divergencias y de que la fase de recolección de datos, generación de ideas y la fase de evaluación esté separadas en el proceso de solución de problemas.

Para este trabajo de investigación, se utilizó la técnica de grupo nominal y se dividió en 3 grupos de a 4 personas (expertos en desarrollo de software). Las reuniones de grupo

nominal se realizaron el 5 de marzo del 2018, el 3 abril del 2018 y el 7 mayo 2018 en la ciudad del Cusco y Abancay. Al inicio se les describió la dinámica de la reunión, luego se procedió a analizar cada una de las 4 preguntas para finalmente llegar a conclusiones de la reunión.

Instrumento 1: Pauta de grupo nominal

Se utilizó el instrumento "Guía o pauta de grupo nominal", en la cual se consideró el número de participantes de la reunión, la fecha y hora, el tema a tratar, las preguntas a realizar y el tiempo establecido para dicha reunión.

Para poder cumplir con los objetivos planteados y recabar la información, se utilizó el instrumento de *pauta de grupo nominal*. El instrumento constaba de 4 preguntas: 1) Qué modelo de proceso de software utiliza para hacer un proyecto? 2) Porqué se retrasa la entrega de un proyecto al cliente/usuario? 3) Nivel de participación del cliente/usuario en el proyecto de software? 4) Qué problemas hay en el desarrollo de software?.

Técnica 2: Encuesta

Es una técnica que está destinada a recopilar información; son técnicas complementarias, que el investigador combinará en función del tipo de estudio que se propone realizar. En una encuesta se pueden registrar las situaciones se pueden observar y en vez de realizar un experimento se cuestiona a la persona participante sobre ello. Por ello, se dice que la encuesta es un método descriptivo con el que se pueden detectar ideas, necesidades, preferencias, hábitos de uso, etc.(Torres, Paz, & Salazar, 2006).

En este trabajo de investigación se aplicó una encuesta a los expertos en proyectos de software para obtener información relacionada a: las características de los proyectos, el modelo de software que usan, rango de los montos de los proyectos que desarrollan, alcance de los proyectos entre otros aspectos. En total se encuestó a 12 personas de que trabajan en empresas y que residen en Abancay y Cusco. En un siguiente trabajo de investigación, se pretende abarcar a otras regiones del país.

Instrumento 2: Cuestionario

El instrumento que se utilizó es el "cuestionario", en la cual es un conjunto de preguntas, por lo general las preguntas son de tipo cerradas y preparadas

cuidadosamente, sobre los hechos y aspectos que interesan en una investigación, para que sea contestado por la población o su muestra.

El cuestionario fue validado por juicio de experto y una prueba piloto antes de su aplicación. El cuestionario contiene 12 preguntas y se realizó utilizando la tecnología *Google Forms*, en vista que muchos de ellos estaban trabajando fuera de Abancay y por tanto era más fácil para ellos llenar un cuestionario por la web, luego se exportaron los datos a una hoja electrónica de cálculo y se procesaron los resultados obtenidos mediante cuadros y gráficos estadísticos.

Técnica 3: Entrevista

La entrevista es un diálogo que se realiza entre dos o más personas: el entrevistador interroga y el entrevistado responde a las preguntas. (Goetz, Goetz, & LeCompte, 1988). Con el objetivo de comprender mejor el contexto sobre el cual se llevará a cabo la investigación, es necesario acceder a la comunidad, seleccionar las personas clave, participar en todas las actividades de la comunidad que sea posible, aclarando todas las opiniones que se vayan realizando mediante las entrevistas (ya sean formales o informales).

Para este proyecto se entrevistó a los Especialistas de la Dirección Regional Educación de Apurímac, Especialistas de la Unidades de Gestión Educativa Local de Apurímac y Directores de Institución Educativa de la Región de Apurímac para que puedan dar su opinión respecto al software SIREMAP. La entrevista se realizó el 5 de octubre del 2018 en la oficina de la Dirección de Gestión Pedagógica de la DRE Apurímac.

Instrumento 3: Pauta de entrevista

Para la observación participante se utilizó el instrumento de “pauta de entrevista”, en el cual se anotó las opiniones de cada participante. En la pauta de entrevista se consideran los aspectos importantes como por ejemplo la fecha y hora de la entrevista, el entrevistador, el entrevistado, las preguntas sobre las que se desarrollará la conversación, entre otros aspectos importantes.

En este trabajo de investigación, la pauta de entrevista ha tenido 3 preguntas para los clientes del sistema y en base a las respuestas manifestadas por cada uno de ellos se procedió a analizar y redactar las conclusiones.

3.4 Análisis de datos y categorías

Unidades de análisis y categorías

Personas con amplia experiencia en el desarrollo o dirección de proyectos de software. En este caso se buscó empresas o personas naturales cuyo rubro de trabajo es la industria del software. Se planteó 2 grupos para la recolección de datos: Se consideró a personas o empresarios que se dedican al desarrollo de proyectos de software de la región Apurímac y la región Cusco.

Se ha tratado de mantener la privacidad de sus datos en la medida de las posibilidades, en vista que la información proporcionada a veces es parte de la empresa y prefieren guardar su privacidad. Hubo un ingeniero que trabajaba para una empresa que al inicio nos brindó información, pero luego no quiso dar mayor detalle porque requería de una autorización de su jefe inmediato y por lo cual se tuvo que descartar su participación.

Categorías de análisis

Para el análisis de datos, se planteó 3 categorías para la problemática actual en el desarrollo de proyectos de software de esta investigación:

Tabla 3
Descripción de las categorías de análisis

Item	Categoría
1	Definir el modelo de proceso para el desarrollo de software
2	Causas para el retraso de la entrega de un proyecto de software
3	Nivel de participación del cliente/usuario en el proyecto de software
4	Problemas principales en el desarrollo de software

3.5 Descripción del modelo co-participativo con el usuario

3.5.1 Determinación de Requisitos con el Usuario (DRU)

Entrevista→Determinación→Priorización

En esta fase se realiza la entrevista a los usuarios para determinar la lista de requisitos, preferentemente se pide autorización para grabar la conversación, debido a que muchas veces el usuario cambia de opinión o no recuerda lo que dijo al inicio, entonces en estos casos se vuelve a escuchar el audio o video con el usuario para aclarar los requisitos.

En esta fase también se prioriza los requisitos con la participación del cliente, porque a veces existen requisitos que son más prioritarios que otros. En caso de haber igualdad de prioridad en requisitos, entonces el equipo de desarrollo define

3.5.2 Diseño Centrado en el Usuario (DCU)

Una vez que se tienen la lista de requisitos, lo que se hace es hacer el diseño o maquetación de las interfaces con apoyo y participación del usuario. Para el diseño se puede utilizar herramientas como: Balsamiq Mockups, AxureXP, FluidUI, Invision, WebFlow, OmniGraffle, Origami, entre otros; También se puede hacer diseños de interfaces con plumón y papel, para esto se puede que tener elementos pre-impresos para acelerar el diseño (formularios, botones, textbox, combobox, entre otros).

El diseño arquitectónico también se debe realizar en esta etapa, para determinar la arquitectura lógica y física del sistema, es decir, la forma en que se comunicarán los datos, el ambiente operacional sobre el cual se desplegará la aplicación, entre otros aspectos.

En esta etapa también se realiza el diseño de la base de datos (incluye la normalización de las tablas y las restricciones de los datos), de tal manera que en los formularios y reportes ya se conocen los campos que se van a utilizar.

3.5.3 Programación y Pruebas con el Usuario (PPU)

En esta parte se realiza la programación en base al diseño planteado anteriormente, es importante mencionar que en esta etapa se definirá el lenguaje de programación, las buenas prácticas de programación, También se verá por la calidad del código desarrollado, los tipos de pruebas que se realizarán (caja negra, caja blanca, pruebas de rendimiento, pruebas de estrés, pruebas alfa, pruebas beta, entre otros).

En esta etapa es importante que el usuario contraste el diseño realizado con las interfaces desarrolladas y luego da su conformidad en caso de cumplimiento. Aquí se realizan reuniones inter-diarias entre el equipo de desarrollo para ver tres aspectos: el avance, las dificultades y las posibles soluciones. Cada fin de semana se realiza la integración y despliegue de una versión de la aplicación para que el usuario lo pruebe.

3.5.4 Implantación y Mantenimiento con el Usuario (IMU)

Una vez que el sistema fue programado y probado de acuerdo a la especificación de requisitos y el diseño centrado en el usuario, se debe planificar la implantación del sistema. El software debe funcionar en un ambiente real, con datos reales y con usuarios reales. Por otro lado, en esta etapa se debe capacitar a los usuarios en el uso manejo del sistema. En caso que el sistema requiera algunos ajustes mínimos, entonces se debe hacer las correcciones necesarias.

Una vez que el sistema está funcionando, probablemente los usuarios encuentren nuevas necesidades o requerimientos; esta fase es conocida como la fase de mantenimiento, en donde el software evoluciona a medida que pasa el tiempo y los usuarios encuentran nuevas reglas de negocio, nuevas necesidades, nuevos reportes, entre otros.

3.5.5 Descripción de las 4 fases del modelo co-participativo

En la *primera fase DRU,* trata de entender el problema a resolver y las necesidades que tiene el usuario, para esto se realizan las entrevistas con los usuarios, se solicita formatos manuales o datos anteriores. La lista de requisitos se describe en el formato de “Requisitos de software”, en la cual se considera: código, título, descripción, prioridad, dificultad, iteración y fuente. En la priorización de requisitos participa el usuario y el arquitecto de software.

La *segunda fase es el DCU* en la que el arquitecto de software y el usuario diseñan conjuntamente las interfaces requeridas para la aplicación; se puede volver a la fase anterior y hacer ajustes en los requisitos (por eso se enmarca con una flecha en doble sentido).

La tercera fase PPU consiste en realizar la programación de todas las interfaces y la lógica de negocio en base a las reglas dadas; los programadores inician la programación en base a los requisitos priorizados. Cada vez que terminan un módulo deben de hacer las pruebas de caja negra, pruebas de caja blanca y las pruebas alfa que son realizadas por otro miembro del equipo de desarrollo. Debe considerar el diseño arquitectónico planteado (cliente servidor, distribuido), el tipo de aplicación (web, móvil o desktop), el framework adecuado, el número de capas, los niveles de seguridad, entre otros aspectos. Cada semana se debe integrar toda la

aplicación y generar una versión funcional (ejecutable) del sistema; de tal manera que el usuario prueba el sistema conjuntamente que el programador, se hacen pruebas de todo el sistema y se usa el método de las pruebas beta. Cada vez que el usuario haya comprobado que el módulo está bien y de acuerdo al diseño realizado, entonces deberá dar su conformidad.

En *la última fase IMU*, que consiste en la Implantación o puesta en funcionamiento de la aplicación, se debe considerar que la programación y las pruebas se han debido de realizar previamente; sin embargo, en caso que todavía existan algunos detalles que afinar, todavía se puede ajustar, por eso es que la flecha va en ambos sentidos. En la parte de mantenimiento, se debe considerar los posibles pequeños cambios que el sistema pueda necesitar.

En caso que el usuario requiera implementar nuevos requisitos o hacer cambios sustanciales como parte del proceso pos-implantación, entonces *el ciclo puede volver a repetirse*. Esto implica hacer una reingeniería de los procesos, optimizar los recursos, generar nuevos algoritmos, entre otros. La línea que une IMU y DRU está con línea punteada, porque no siempre se puede dar el caso, aunque en la mayoría de las veces si se da. La Figura 4 muestra el esquema gráfico del modelo co-participativo propuesto en este trabajo de investigación. Existen 4 fases y en las 4 fases siempre participa el usuario o cliente.

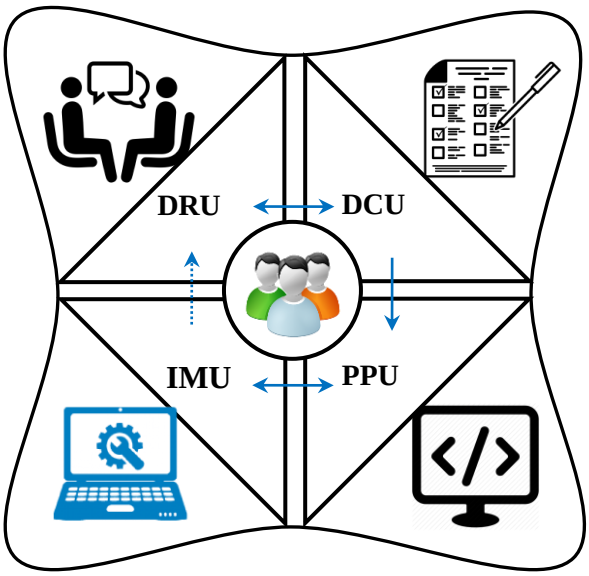


Figura 3. Modelo co-participativo propuesto en esta investigación

3.6 Descripción del desarrollo del proyecto de software SIREMAP utilizando el modelo co-participativo

3.6.1 Determinación de Requisitos con el Usuario (DRU)

En esta fase de reunió a todos los usuarios para determinar los requisitos del sistema, entre los participantes estuvieron: profesores de aula, Directores de Instituciones Educativas, Especialistas de UGEL, Especialistas de DRE, entre otros (Ver Figura 4).

Figura 4. Reunión con los usuarios para determinar los requisitos

Fuente: Dirección Regional de Educación Apurímac.

Luego se tuvo que hacer el diseño de las interfaces de usuario, en la cual participaron también los Especialistas de la DRE y de UGEL quienes diseñaron las interfaces primero en papel y luego se pasó el diseño al software llamado Balsamiq Mockups versión 3, ver Figura 5.

Figura 5. Diseño de interfaces de usuario con los usuarios

Fuente: Dirección Regional de Educación Apurímac.

Tipos y roles de los usuarios del sistema

Para que el sistema tenga niveles de seguridad en el control de acceso, se segmentó el acceso al sistema en 4 niveles: *Usuario de IIEE*, que significa usuario la Institución Educativa, *Usuario EspecialistaUGEL* que significa usuario que pertenece a la UGEL y requiere ver la información de las las Instituciones Educativas de su jurisdicción (por ejemplo UGEL Abancay, UGEL Andahuaylas, UGEL Cotabambas, etc.), *Usuario especialista DRE* que significa que es un usuario que trabaja en la DRE y requiere ver la información de las UGEL, y el *Usuario Administrador* que puede configurar los parámetros más importantes del sistema, dar los permisos a los usuarios, entre otras acciones de seguridad para el sistema. La tabla 4 muestra los tipos de usuario y su descripción detallada:

Tabla 4
Tipos de usuarios del sistema

Tipo de usuario	Descripción
Usuario *IIEE*	Es aquel Director de Institución Educativa (IIEE) que puede hacer un monitoreo en aula a cualquier docente de su IIEE. Requiere autenticarse. Está habilitado para acceder *Inicio, Mis Fichas y Mis datos*. Para nuestro caso: Director de la IIEE Las Mercedes
Usuario *EspecialistaUGEL*	El usuario *EspecialistaDRE* es un Especialista de la UGEL. Este usuario tendrá todas las características del usuario IIEE

	y adicionalmente podrá acceder al menú *Seguimiento docente, Seguimiento IIEE, Reportes*. Para nuestro caso: Especialista de UGEL Abancay Ivan Fuentes
Usuario *EspecialistaDRE*	El usuario *EspecialistaDRE* es un Especialista de la DRE. Este usuario tendrá todas las características del usuario UGEL y adicionalmente podrá acceder al menú *Seguimiento UGEL, Usuarios*. Para nuestro caso: Especialista Juan José Huamán
Usuario *Administrador*	Usuario habilitado para realizar todas las acciones del menú y tiene todos los privilegios para manipular el sistema. Para nuestro caso: Manuel Ibarra

Fuente: Especialistas de la Dirección Regional de Educación Apurímac

Lista de requisitos del Sistema

Una vez que los usuarios brindaron información sobre el proceso de monitoreo de la sesión de aprendizaje del docente de Educación Básica Regular en el Perú, los Especialistas y el equipo de desarrollo definieron en conjunto los requisitos del sistema. Al inicio hubo requisitos ambiguos (se considera requisito ambiguo cuando dos o más usuarios definen de manera distinta las reglas de funcionamiento un requisito) y algunas discrepancias entre los usuarios sobre la funcionalidad de algunos requisitos, pero que en reuniones permanentes se tuvo afinar y finalmente se llegó a la lista de requisitos mostrados en la Tabla 5.

Tabla 5
Requisitos funcionales y no funcionales de SIREMAP

Requisitos funcionales

Código : RS01
Título: Hacer módulo que permita manejar CRUD de usuarios.
Descripción: Funcionalidad que permite al Administrador crear, seleccionar, eliminar y actualizar usuarios, con su respectivo perfil y permisos para acceder al menú. Los datos requeridos para el acceso son: Id y contraseña. Una vez que el usuario haya ingresado estos datos, el sistema verificará que esté registrado en la base de datos, para luego concederle el acceso al sitio de acuerdo a su rol.
Prioridad : Media
Dificultad : Media
Iteración : 1
Usuario responsable: Juan José Huamán

Código : RS02
Título: Hacer el módulo de seguimiento a UGEL.
Descripción: Funcionalidad que permite al *EspecialistaDRE* crear, seleccionar, eliminar y actualizar UGELs y asimismo permite ver las visitas que han hechos

los especialistas a la UGEL.
Prioridad : Media
Dificultad : Baja
Iteración : 2
Usuario responsable: Juan José Huamán

Código : RS03
Título: Hacer el módulo de seguimiento a IIEE.
Descripción: Funcionalidad que permite al *EspecialistaUGEL* crear, seleccionar, eliminar y actualizar IIEE y asimismo permite ver las visitas que han hechos los especialistas a la IIEE.
Prioridad : Media
Dificultad : Media
Iteración : 2
Usuario responsable: Ivan Fuentes

Código : RS04
Título: Hacer el módulo de seguimiento a Docente.
Descripción: Funcionalidad que permite al *EspecialistaUGEL o DirectorIIEE* crear, seleccionar, eliminar y actualizar docentes y asimismo permite ver las visitas que han hechos los especialistas a un docente.
Prioridad : Media
Dificultad : Media
Iteración : 1
Usuario responsable: Juan José Huamán

Código: RS05
Título: Hacer el módulo para el CRUD de Ficha de Monitoreo a Sesión de aprendizaje.
Descripción: Funcionalidad que permite al *EspecialistaDRE, EspecialistaUGEL o DirectorIIEE* crear, seleccionar, eliminar y actualizar Fichas de monitoreo
Prioridad : Alta
Dificultad : Alta
Iteración : 1
Usuario responsable: Juan José Huamán, Edith Montalvo

Código: RS07
Título: Llenar ficha de monitoreo.
Descripción: Funcionalidad que permite llenar la ficha de monitoreo a sesión de aprendizaje con las rúbricas emanadas por el Ministerio de Educación (ver Anexo 3 de fichas de monitoreo a sesión de aprendizaje). Las fichas deben ser fáciles de llenar como si el usuario estuviese haciéndolo con lápiz y papel.
Prioridad : Alta
Dificultad : Alta
Iteración : 1
Usuario responsable: Edith Montalvo

Código: RS08

Título: Buscador de docente monitoreado.
Descripción: Funcionalidad que permite al *EspecialistaDRE, EspecialistaUGEL o DirectorIIEE* buscar a un determinado docente por cualquiera de los siguientes campos: apellido paterno, apellido materno, nombre, DNI, situación. El buscador debe ser utilizando filtros por uno o varios campos.
Prioridad : Media
Dificultad : Media
Iteración : 2
Usuario responsable: David Calapuja

Código: RS09
Título: Exportar ficha de monitoreo a formato PDF
Descripción: Funcionalidad que permite al *EspecialistaDRE, EspecialistaUGEL o DirectorIIEE* exportar la ficha de monitoreo a sesión de aprendizaje a formato PDF, calculando las sumas por ítem.
Prioridad : Media
Dificultad : Media
Iteración : 2
Usuario responsable: Edith Montalvo

Código: RS10
Título: Mostrar reporte de estadísticas por ficha
Descripción: Funcionalidad que permite al *EspecialistaDRE, EspecialistaUGEL o DirectorIIEE* generar reportes con las estadísticas de las fichas llenadas por año, por UGEL, y por periodo.
Prioridad : Media
Dificultad : Alta
Iteración : 1
Usuario responsable: Fanny Choque

Código: RS11
Título: Mostrar reporte de estadísticas por Institución Educativa
Descripción: Funcionalidad que permite al *EspecialistaDRE, EspecialistaUGEL o DirectorIIEE* generar reportes con las estadísticas de las Institución Educativa por año, por UGEL, y por periodo.
Prioridad : Media
Dificultad : Media
Iteración : 2
Usuario responsable: Edith Montalvo

Código: RS12
Título: Mostrar reporte del resumen de porcentaje de docentes evaluados
Descripción: Funcionalidad que permite al *EspecialistaDRE, EspecialistaUGEL o DirectorIIEE* generar reportes que resumen la cantidad de docentes que han sido monitoreados frente a los que faltan monitorear.
Prioridad : Media
Dificultad : Media
Iteración : 2

Usuario responsable: Fanny Choque

Requisitos no funcionales

Código: RS13
Título: El sistema debe funcionar en el servidor de la DRE Apurímac
Descripción: Funcionalidad que restringe el funcionamiento del sistema en el servidor de la DRE Apurímac. Por tanto debe funcionar en lenguaje PHP 5.5, servidor de base de datos Mysql 5.5 y apache web server 2.
Prioridad : Baja
Dificultad : Media
Iteración : 2
Usuario responsable: Edison Montes

Código: RS14
Título: El sistema debe ser fácil de usar
Descripción: Funcionalidad que requiere que el sistema se fácil de usar para un Director de IIEE que tenga conocimientos básicos de ofimática e internet.
Prioridad : Alta
Dificultad : Baja
Iteración : 2
Usuario responsable: Edison Montes

Código: RS15
Título: El sistema debe soportar varios navegadores
Descripción: Funcionalidad que requiere que el sistema funcione en por lo menos dos navegadores: Firefox 3.5 o superior, Chrome 7 o superior, o Internet Explorer 9 o superior (son los navegadores más utilizados).
Prioridad : Alta
Dificultad : Baja
Iteración : 2
Usuario responsable: Edison Montes

Fuente: Usuarios del sistema

3.6.2 Diseño Centrado en el Usuario (DCU)

El diseño centrado en el usuario ser realizó utilizando las siguientes herramientas:

- **Balsamiq Mockups 3**. Es una herramienta para realizar diseño de prototipos de interfaces de usuarios.
- En algunos casos se hizo el diseño utilizando paleógrafos y plumón. El diseño es una aproximación muy cercana a lo que el usuario quiere que haga el sistema, es por eso que se da mucho énfasis en esta fase, ver Figura 6.

Figura 6. Reuniones con los usuarios para definir las interfaces

Fuente: Dirección Regional de Apurímac

- En otros casos, se realizaron diseños utilizando el software Balsamiq Mockups, de tal forma que el cliente o usuario participaba activamente en el diseño, eligiendo los colores, los tamaños, los formatos, etc. tal como aparecerían en el producto final. El hecho que el cliente o usuario participe en esta etapa, ayuda bastante a que el equipo de desarrollo programe de acuerdo a las expectativas del usuario final.

3.6.3 Programación y Pruebas con el Usuario (PPU)

Para la programación del sistema se utilizó las siguientes herramientas:

- **Framework Code Igniter 3**. Es un framework diseñado para el desarrollo de aplicaciones web bajo el esquema de programación Modelo Vista Controlador (MVC).
    - Modelo: El modelo se encarga de la conexión a la base de datos y es encarga de procesar y obtener los datos.
    - Vista: La vista se encarga de presentar los datos en la interfaz del usuario (HTML, CSS, JavaScript). Controlador:
    - El controlador se encarga de controlar los datos, es decir obtiene datos de un modelo, los procesa, y se los pasa a la vista.

- **Base de datos Mysql 5.X.** Es un gestor de base de datos relacional bajo una tipo de licencia dual (pública y comercial) que pertenece a la empresa ORACLE Corporation y es considerada como la base de datos de código abierto más popular.

- **PHP 5.X** Viene del acrónimo Hypertext Preprocesor, un lenguaje de programación orientado a la web y de propósito general.

- **GroceryCRUD.** Es una herramienta para realizar operaciones de Crear, Leer, Actualizar y Eliminar registros de una tabla. Esta herramienta funciona en varios navegadores, es multilenguaje, es adaptable a cualquier tabla, tiene varios temas para aplicarlos y es de código abierto.

Las pruebas que se realizaron fueron:

- **Pruebas de caja negra**, estas pruebas lo realiza el programador introduciendo datos de entrada y verificando los resultados que debe dar los algoritmos programados

- **Pruebas de caja blanca**, en esta fase de probó el verificando el código fuente, especialmente las estructuras condicionales y de control

- **Pruebas de rendimiento y de estrés**, para probar la performance del servidor y sobre todo para evaluar la cantidad de usuarios que pueden acceder simultáneamente.

- **Pruebas de módulo con el cliente**, estas pruebas se realizaron en coordinación con el cliente. El cliente verifica si cada módulo está de acuerdo a lo que se diseñó inicialmente y las pruebas se realizan con datos concretos y reales.

3.6.4 Implantación y Mantenimiento con el Usuario (IMU)

Implantación

El objetivo principal de esta fase es realizar un conjunto de procedimientos o actividades necesarias para poner la aplicación en funcionamiento en el ambiente operacional definido por el cliente/usuario.

El sistema fue implantado y funciona actualmente en el servidor de la Dirección Regional de Educación de Apurímac http://siremap.dreapurimac.gob.pe , tiene almacenados: 3871 usuarios, habilitado para 2744 Instituciones Educativas en Apurímac (nivel primaria y secundaria) y 13000 fichas de monitoreo. Se capacitó al responsable del área de informática de la DRE para que pueda sacar copias de seguridad, hacer un plan de mantenimiento preventivo y sobre todo la creación o modificación de usuarios, Directores, Especialistas, entre otros. La Figura 7 muestra el funcionamiento del sistema

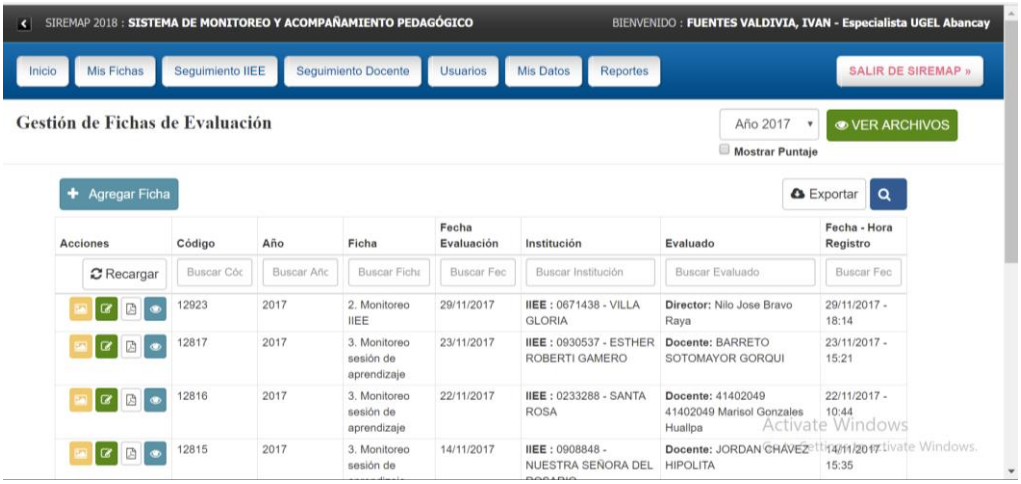


Figura 7. Funcionamiento de SIREMAP

Fuente: Dirección Regional de Educación Apurímac

Mantenimiento

El mantenimiento de software es la modificación parcial de algunos requisitos de un producto de software después de la entrega; generalmente es para corregir errores de forma, para agregar funcionalidad, para mejorar el rendimiento o para realizar actualizaciones en las reglas de negocio.

El mantenimiento en SIREMAP se da porque existen algunos cambios que anualmente se debe dar al sistema, debido a que el Ministerio de Educación es el responsable de aprobar la ficha de monitoreo de sesión de aprendizaje, y la DRE utiliza dicho formato y a veces lo adapta de acuerdo a sus necesidades. El responsable del área de Informática de la DRE es el encargado de hacer dichas modificaciones en el sistema.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

En este capítulo se presentan los resultados más importantes encontrados en el desarrollo de este proyecto de investigación.

4.1 Análisis de resultados

En este capítulo se presentan los resultados más importantes encontrados en el desarrollo de este proyecto de investigación.

4.1.1 Resultados de Grupo Nominal

Los resultados de las reuniones de grupo nominal se dividieron en 3 grupos de 4 personas cada uno. Las transcripciones de cada una de las reuniones se muestran en la tabla 6, 7 y 8:

Tabla 6
Resultados del Grupo Nominal 1

Guía o Pauta de Grupo Nominal – grupo 1	
Este instrumento permite recabar información de la problemática actual en el proceso de desarrollo de software. http://repositorio.uchile.cl/bitstream/handle/2250/133403/ANEXOS.pdf?sequence=2 Adaptado por el investigador	
Lugar: Jirón Huancavelica 707 Motivo: Reunión de trabajo con expertos en proyectos de software Tema: Problemas actuales en el proceso de desarrollo de software. Moderador: Manuel Ibarra Cabrera	Fecha: 7 mayo 2018 Hora de inicio: 17:00 Hora de finalización: 18:40 Número de participantes: 4

Detalle de la reunión de grupo nominal
Preguntas: 1) Qué modelo de proceso de software utiliza para hacer un proyecto? 2) Porque se retrasa la entrega de un proyecto al cliente/usuario? 3) Nivel de participación del cliente/usuario en el proyecto de software? 4) Que problemas hay en el desarrollo de software?. **Participantes:** - Yonatan (Y) - Kevin (K) - Cristhian (C) - Robert (R)
1) Qué modelo de proceso de software utiliza para hacer un proyecto? Respuestas: **R:** Por lo general hacemos una reunión con el cliente para capturar los requisitos y las necesidades. No seguimos una metodología paso a paso, sino más bien algo adaptado y es parecido al SCRUM con los Sprints. En algunos casos hay proyectos en el que ya estaba plasmado el análisis y el diseño, así que empezamos a programar; en los casos que empezábamos desde cero, entonces hacíamos el análisis, el diseño y luego la programación. **C:** Trabajamos más en XP, capturamos los requisitos, y vemos el proceso, nos basamos en que es lo que quien, y luego de determinar los requisitos, hacíamos iteraciones de 2 semanas y luego le mostramos las interfaces al usuario **K:** Para el caso de las empresas, varía de acuerdo a si el trabajo es formal o informal, en algunos casos hacíamos en cascado, y depende mucho de la persona que hace el análisis que cuando es experimentado entonces define de manera casi terminada el análisis. Cuando no se tiene bien conocido, entonces es mejor trabajar con un modelo iterativo, se hace la lista de requerimientos, luego armamos los Sprints y se adecua de acuerdo a las necesidades, por lo tanto hacemos un sprint en cada semana, y se le va mostrando al cliente. **Y:** En mis primeras experiencias, necesariamente tenía que aplicar la metodología un poco rígida y se tenía que cumplir con plazos, y la empresa exigía documentar, entonces tenía que cumplir de acuerdo a los requerimientos y diseño que se ha dado. Hoy en día trabajo con SCRUM o XP y tratamos de recopilar los requisitos y luego empiezo a programar, no pienso tanto en que metodología seguir paso a paso, simplemente empiezo a programar de acuerdo a lo que pide el diseño. Sabemos cómo se usa RUP, SCRUM o XP, pero no es una camisa de fuerza para empezar a programar y como los proyectos son pequeños, entonces no hacemos la documentación. Pero cuando el cliente nos exige entonces podemos agregar la documentación
2) Porque se retrasa la entrega de un proyecto al cliente/usuario?

Respuestas:

R: El cliente siempre quiere algo más, especialmente cuando ve por primera vez, va pidiendo más cosas

C: Los requisitos son cambiantes o quiere agregar requisitos

Y: Parte primero por el tiempo del cliente, el otro aspecto es que los requerimientos van cambiando. Ahora ya existe un nuevo área de ingeniería de requisitos, justamente porque es una debilidad del área y tener el análisis bien definido es un factor clave, porque si el análisis está bien hecho, entonces la programación ya es sencilla, incluso existen herramientas que permiten modelar y hacer la programación, como por ejemplo Bizzagi.

K: Coincido en que a veces los requisitos van cambiando o los clientes van agregando funcionalidad. Existe otros factores, como por ejemplo la poca experiencia del analista funciona, porque a veces hay analistas que no conocen el tiempo que podría demorar porque son más teóricos y no saben programar, entonces cuando hacen el calcular de los tiempos, luego el programador se demora más. Suponiendo que esta bien el análisis, luego el cliente quiere algo más, y pide más requisitos, entonces el jefe de proyecto a veces acepta más requisitos que no estaban programados en el calendario, entonces falta la tenacidad del Jefe de Proyecto, por tanto , se puede hacer pero entraría en un nuevo calendario.

3) Nivel de participación del cliente/usuario en el proyecto de software?

Respuestas:

Y: De acuerdo a un nivel de escala la participación del cliente se podría clasificar en Muy Frecuente, Frecuente, Casi Frecuente y Nada Frecuente. Bajo esa escala de clasificación, podría decir que, en mi experiencia la participación del cliente es casi frecuente pero solo al inicio, luego ya va dejando de participar… piensa que el programador es el erudito y cree que él debe resolverlo todo

R: Yo creo que tiene que ver mucho con cuán grande es el proyecto, en mis proyectos, la verdad es que me reunía la primera vez para capturar los requisitos y luego el cliente se aparecía casi al final para mostrarle el funcionamiento del sistema y para el momento de cobrar se desaparecía… Luego en la empresa actual donde trabajo, se trata de que el cliente venga a algunas reuniones, pero a veces no tiene tiempo.

C: A veces hay clientes que participan y hay otros poco participativo y a veces no tiene mucho tiempo

K: Depende del cliente, algunos participan bastante y otros no. Para que el cliente participe hay que invitarlo, luego te da los requerimientos y posteriormente el cliente cambia de opinión y no se acuerda de lo que te pidió al inicio; y por eso ahora grabo las reuniones para demostrarle los acuerdos que se tenían. Por otro lado, a veces el cliente dice: se supone que tú lo debes hacer bien. En concreto yo creo que el nivel de participación es nada frecuente.

4) Que problemas hay en el desarrollo de software?

Respuestas:

R: Con los clientes yo tuve dos problemas: no se valora tanto el software, con el segundo cliente fue con el pago que no nos pagó lo que acordamos en un inicio.

C: En nuestra región el cliente no lo considera como inversión sino como un gasto, a veces te dicen, pero en Wilson ese software está más barato. Otro problema es que el cliente no sabe exactamente lo que quiere.

K: El cliente valora muy poco el desarrollo de software, los egresado cobran muy barato el producto de software, entonces el cliente piensa que así de barato cuesta el software, a veces por ochocientos soles hacen un software, pero como son egresados no pagan agua, luz, no tienen familia que mantener; y lo desarrollan en unos 3 meses y creen que les sale a cuenta. El otro problema es que, muchas personas no hacen bien el software, entonces el cliente ya no quiere que lo vuelvan hacer porque piensan que no lo van a terminar. El otro problema es que el cliente no piensa que se puede hacer en la región un software de calidad y que piensa que el producto de Lima es mejor que el de la región, por tanto, hay desconfianza en los desarrolladores de la región. A veces lo hacen personas que no tienen experiencia y lo hacen mal.

Y: Muchos requerimientos para poco tiempo, el otro es que el costo es muy barato. A nivel regional hay desconocimiento sobre como es el desarrollo de software, no saben en qué consiste, entonces los clientes piensan que es sentarse y programar. Los clientes no conocen sobre calidad de software, las pruebas, etc. entonces no valoran el trabajo del informático.

Concusiones:

- No hay una metodología preferida por todos, pero la mayoría usa metodologías ágiles SCRUM, XP o una adaptación de una de ellas.
- Los proyectos de software se retrasan porque los requisitos son cambiantes, o porque el cliente agrega requisitos después de ya se inició el proyecto
- El nivel de participación del cliente es escaso, al inicio del proyecto se interesa un poco, pero luego cree que el ingeniero debe resolverlo todo.
- El cliente no valora la calidad de software, el cliente siempre busca lo más barato y sobre todo desconfía que el software le pueda ser útil.

Tabla 7

Resultados del Grupo Nominal 2

Guía o Pauta de Grupo Nominal – grupo 2

Este instrumento permite recabar información de la problemática actual en el proceso de desarrollo de software.

http://repositorio.uchile.cl/bitstream/handle/2250/133403/ANEXOS.pdf?sequence=2

Adaptado por el investigador

Lugar: Urb Marcavalle H-20, Cusco	Fecha: 3 abril del 2018

Motivo: Reunión de trabajo con expertos en proyectos de software. Tema: Problemas actuales en el proceso de desarrollo de software. Moderador: Manuel Ibarra Cabrera	Hora de inicio: 16:00h Hora de finalización: 18:00h Número de participantes:4

Detalle de la reunión de grupo nominal

Preguntas:

1) Qué modelo de proceso de software utiliza para hacer un proyecto?
2) Porque se retrasa la entrega de un proyecto al cliente/usuario?
3) Nivel de participación del cliente/usuario en el proyecto de software?
4) Que problemas hay en el desarrollo de software?.

Participantes:

- Edson (E)
- Cristian V. (C)
- Percy (P)
- Jhon (J)

1) Qué modelo de proceso de software utiliza para hacer un proyecto?

Respuestas:

E: Yo he liderado proyectos usando Kanban teniendo buenos resultados, en otros proyectos en los que que he participado hemos utilizado Scrum, en todos los casos no se cumplen los lineamientos al 100%, siempre se agrega o se quita ciertas reglas según el proyecto.

C: En la empresa en la que trabajo utilizamos SCRUM, comenzamos con los requerimientos, y partimos por un Sprint y lo hacemos cada dos semanas y tiene que ser utilizable y ejecutable porque pasa a producción. Los analistas recaban los requerimientos y a veces a falta de tiempo los Sprints tienen que ser priorizando y se agrupan en un sprint. Nosotros trabajamos con el área específica de “Logística” y nos dedicamos a ese rubro y está bien definido, por tanto nuestro trabajo ya es especializado.

P: Por lo general yo uso metodologías ágiles con SCRUM, pero depende del proyecto a veces uso Cascada

J: Depende del proyecto, pero últimamente utilizamos XP

2) Porqué se retrasa la entrega de un proyecto al cliente/usuario?

Respuestas:

E: Generalmente por 3 razones: primero, por la falta de claridad al momento de

definir los requisitos (fase de análisis incompleta); segundo, por repentinos cambios en los requerimientos por parte del cliente. La tercera razón es por falta de dominio técnico en el equipo.

C: El Project Manager se encarga de hacer la negociación. En proyecto hubo un conjunto de requerimientos y se determina el costo y luego se envía la propuesta de los costos y se definen los Sprints. Y si después salen nuevos requerimientos y se analiza las priorizadas y se define entre mover un requisito de un sprint a otro; de todas estas negociaciones lo hace el Project Manager y siempre se dará esos casos, en mi opinión un software nunca va a terminar de desarrollarse, el software siempre va creciendo.

P: La poca colaboración del cliente en el proyecto

J: Escasa colaboración del cliente, porque a veces no se involucra dentro del desarrollo del proyecto y se le muestra el producto final acabado

3) Nivel de participación del cliente/usuario en el proyecto de software?

Respuestas:

E: Lo ideal es que participe activamente. Las metodologías ágiles requieren de opiniones por parte del cliente por lo menos al culminar un sprint. Evidentemente si no se cumple, las características desarrolladas son asumidas como correctas y en el momento de la entrega del producto se pueden generar discrepancias.

C: Nosotros tenemos otro tipo de clientes y nuestra empresa es bastante establecida, tenemos un área específica que se encarga de establecer la participación del cliente y su participación es muy importante. Tenemos el área de sistemas y el área de negocio. En este caso la participación del cliente es Frecuente, en cada comienzo del Sprint y en la finalización del Sprint, es más, y le preparamos un ambiente de prueba antes de pasar a producción de software para que pueda darnos su conformidad sobre todo lo que se planificó. A la vez se tiene que establecer canales de comunicación por Skype, correo, etc. porque necesitamos la conformidad de inmediato y no generar retrasos

P: El usuario tiene poca participación porque a veces no tiene tiempo o no se interesa mucho en el proyecto, porque piensa que el desarrollador debe resolver el problema entero.

J: El cliente o usuario piensa deja la responsabilidad de que el software salga bien al programador, porque piensa que para eso está formado, por tanto, pocas veces participa, solo al inicio del proyecto en las entrevistas y luego al final para ver cómo está el producto final.

4) Qué problemas hay en el desarrollo de software?

Respuestas:

E: La menor importancia que se le da a un proyecto de software en comparación con proyectos de otros rubros. Los bajos presupuestos destinados. La baja participación de los clientes en el proceso de desarrollo.

C: El costo de mano de obra es bastante barato en la región, a pesar que uno le ofrezca un producto muy bueno, a veces el cliente no quiere pagar por la misma situación económica de la región. A veces hay tantas empresas que compitan por vender un producto que cada vez van reduciendo los precios. Cuando el cliente es extranjero, entonces el Cliente prefiere hacer outsorcing, de tal forma que los precios son mejores, pero hay que hacer todas las tareas desde el análisis hasta la puesta en producción, es un negocio bastante rentable, porque competir a nivel internacional tiene niveles económicos más altos, pero entrar al mercado internacional es bastante complejo, porque hay que trabajar con niveles de calidad bastante altos.

P: Las personas piensan que desarrollar software es una tarea fácil y creen que es cómo manejar un paquete de Word o Excel, mas no piensan en cómo se hizo el Word o Excel, por tanto siempre lo ven como una tarea sencilla que cualquiera lo puede hacer.

J: El hecho que el cliente no colabore con el desarrollo de software a veces hace que se tenga insatisfacción de parte de él porque piensa que le hicieron un software que no se ajusta a sus requerimientos.

Concusiones:

- Generalmente se usa metodologías ágiles XP o SCRUM porque los proyectos son pequeños, pero en proyectos medianos o grandes que requieren documentación se usa cascada
- Un proyecto de software se retrasa porque existe poca claridad en la explicación de los requisitos de parte del cliente y la falta de experiencia de parte de los analistas en recabar bien los requisitos.
- Escasa colaboración del cliente o usuario porque piensan que el software lo deben construir solo los programadores.
- La competencia que hay entre los programadores hace que ofrezcan software más barato pero sin calidad.

Tabla 8

Resultados del Grupo Nominal 3

Guía o Pauta de Grupo Nominal – grupo 3
Este instrumento permite recabar información de la problemática actual en el proceso de desarrollo de software. http://repositorio.uchile.cl/bitstream/handle/2250/133403/ANEXOS.pdf?sequence=2 Adaptado por el investigador

Lugar: Jr. Apurímac 102, Abancay Motivo: Reunión de trabajo con expertos en proyectos de software Tema: Problemas actuales en el proceso de desarrollo de software. Moderador: Manuel Ibarra Cabrera	Fecha: 5 de marzo del 2018 Hora de inicio: 17:00h Hora de finalización: 18:40h Número de participantes:4

Detalle de la reunión de grupo nominal

Preguntas:

1) Qué modelo de proceso de software utiliza para hacer un proyecto?
2) Porque se retrasa la entrega de un proyecto al cliente/usuario?
3) Nivel de participación del cliente/usuario en el proyecto de software?
4) Que problemas hay en el desarrollo de software?.

Participantes:

- Emerson (E)
- Jesús(J)
- Robinson (R)
- Miriam (M)

1) Qué modelo de proceso de software utiliza para hacer un proyecto?

Respuestas:

E: Generalmente usamos metodología ágil SCRUM, primero se define los Sprints, luego se definen los Sprint Back Log, luego se asignan la tareas a cada programador y luego se hacen los tests.

J: Depende del proyecto, cuando el proyecto es pequeño generalmente usamos metodologías ágiles con programación extrema, pero si el proyecto es mediano o grande, tratamos de documentar y analizar bien entonces usamos Cascada con retroalimentación. Pero en la mayoría de las veces los proyectos que desarrollamos son pequeños, por tanto usamos XP con SCRUM.

R: En los proyectos medianos usamos cascada porque tiene que estar bien definido cada fase y también piden documentación detallada, pero en proyectos pequeños usamos las historias de usuarios y luego adaptamos rápidamente algún framework de acuerdo a las necesidades del cliente.

M: Trabajamos con proyectos pequeños en la cual nos exigen rápidamente que el sistema tenga funcionalidad o que esté operativo en pocos días, por tanto usamos metodologías ágiles con SCRUM.

2) Porque se retrasa la entrega de un proyecto al cliente/usuario?

Respuestas:

E: Generalmente se retrasa porque el cliente agrega más funcionalidad en los últimos momento

J: El analista a veces no hace bien su análisis porque no tiene mucha experiencia, entonces el analista hace su primer análisis como un primer borrador, pero no está bien definido los requisitos, los alcances, los límites o las restricciones; y cuando el cliente ve por primera vez las interfaces, dice: "esto no es lo que yo quería…". Entonces hay que volver a hacer el análisis una y otra vez.

Cuando el cliente pide más requisitos, la cosa es saber negociar, de tal forma que nos nuevos requisitos entran a la cola de espera o si el cliente lo prefiere de inmediato, entonces se prioriza en vez de otro requisitos; pero en ambos casos siempre tiene que tener un pago adicional y si es muy prioritario, entonces el costo debe ser mayor. Otra forma de negociar sería dejamos de hacer un requisito y agregamos el nuevo requisito.

R: Generalmente se retrasa porque el usuario cambia constantemente los requisitos o agrega nuevos requisitos que no han estado establecidos inicialmente y piensa que agregar un nuevo requisito no implica cambios sustanciales y a veces hasta se molesta cuando le cobramos por el nuevo requisito.

M: Al inicio el cliente te explica de una forma los requisitos, luego cuando se le muestra la primera versión del sistema funcionando, se da cuenta que faltaban algunos campos o que se olvidó de un requisito y por tanto se modifica el requisito. Estos cambios generan retrasos en la entrega de los proyectos.

3) Nivel de participación del cliente/usuario en el proyecto de software?

Respuestas:

E: Los clientes están ocupados con sus actividades y colaboran muy poco con el sistema desarrollado

J: El cliente tiene poca participación, si bien es cierto que el equipo de desarrollo conoce la parte técnica para el desarrollo del programa, el cliente conoce más sobre el proceso o las reglas de negocio. Lo que puedo percibir es que cuando el cliente se involucra más, la probabilidad de que el software tenga éxito, pero si no participa mucho entonces la probabilidad de que el proyecto pueda fracasar.

R: El cliente cree que el programador debe resolver los problemas y que el cliente sólo debe esperar los resultados, por tanto no quiere involucrarse mucho.

M: Por las actividades cotidianas que debe realizar en su trabajo, pocas veces tiene tiempo para colaborar con la construcción del software.

4) Qué problemas hay en el desarrollo de software?.

Respuestas:

E: El cliente siempre busca un software que sea barato, no le interesa que tecnología se usa,

si el software es de calidad, si el software tiene seguridad, nada por el estilo, solo quiere que el software sea barato. Esto implica que el trabajo que realiza el desarrollador de software nos e valora.

J: En general el cliente piensa que hacer software es fácil, barato y que cualquiera lo puede hacer; por tanto, le dan el trabajo a personas sin conocimiento técnico o recién egresados o a veces a personas que ni siquiera han terminado la universidad; y luego ya cuando falla buscan a alguien con experiencia profesional para que lo resuelva; a la larga eso genera desconfianza de los clientes en que se puede hacer un software en nuestra Región, creen que el de Lima siempre es mejor. El otro problema es el económico, el Cliente siempre busca lo más barato, más no la calidad. El otro problema es que a veces el software se lanza sin hacer mucho testing entonces a futuro el software falla, por tanto, el cliente cree tiene desconfianza.

R: La competencia que hay en la venta de software, hace que cada vez el software sea más barato, sin importar que sea de calidad.

M: El cliente busca software barato, sin importarle que se de calidad, y a veces cuando falla recién te pide ayuda y quiere que le resuelvas el problema inmediatamente, pero no entiende que el software se tiene que construir a medida para el cliente y que el cliente debe colaborar para que cumpla con sus expectativas.

Concusiones:

- Depende del tipo de proyecto para definir el modelo de proceso de software a usar, a veces se usa metodología ágil y a veces cascada.
- Un proyecto se retrasa porque hay muchos cambios en los requisitos. El cliente/usuario no le da mucha importancia al desarrollo del software porque está ocupado con sus actividades cotidianas
- Poca participación del cliente en el desarrollo del software
- El cliente busca software barato y no de calidad

4.1.2 Resultados de la Encuesta

Se aplicó un cuestionario para conocer la opinión de las personas respecto a la problemática del desarrollo de proyectos de software y para conocer el tipo de software que se construye en la región. Los resultados se muestran a continuación:

En la Tabla 9 se puede se puede observar que participaron 7 personas de Cusco y 5 personas de Apurímac. Las personas que participaron en la encuesta, tienen roles distintos, como por ejemplo: Jefes de proyecto, analistas, diseñadores, programadores y probadores que trabajan en Empresas o Instituciones.

Tabla 9
Empresas, instituciones o trabajadores libres que participaron en la investigación por cada región

Empresa/Institución	Región
Codideep	Apurímac
Freelance	Apurímac
Universidad	Cusco
Fastworkx S.R.L.	Apurímac
Minsa	Cusco
Universidad	Cusco
Freelance	Cusco
SVS	Apurímac
Global Corp	Cusco
Freelance	Cusco
Universidad	Cusco
MIQ Logistics	Apurímac

La Tabla 10 muestra el tiempo que requiere un proyecto de software. La frecuencia más alta es 5, lo cual quiere decir que los proyectos se desarrollan entre 3.01 y 7 meses

Tabla 10
Tiempo de ejecución de un proyecto de software

Tiempo de duración	Frecuencia
de 0 a 3 meses	3
entre 3.01 y 7 meses	5
entre 7.01 y 11 meses	2
entre 11.01 a 24 meses	1
24.01 a más meses	1

La Tabla 11 muestra el rango de los costos de software en los cuales trabajan los encuestados, 75% de los proyectos están entre 0 y 2000 dólares, y el 25% entre 2000.01 y 8000 mil dólares, ninguno de los encuestados trabaja en proyectos costosos por más de 8000.01 dólares. Aquí se puede notar que la mayoría de los empresarios encuestados trabaja en proyectos cuyos costos son bajos.

Tabla 11
Costos de los proyectos de software desarrollados

Costo del proyecto de software	frecuencia
de 0 a 500 dólares	2
entre 500.01 y 1000 dólares	3
entre 1000.01 y 2000 dólares	4
entre 2000.01 y 4000 dólares	2
entre 4000.01 y 8000 dólares	1
entre 8000.01 y 16000 dólares	0
entre 16000.01 y 32000 dólares	0
más de 32000 dólares	0

La Tabla 12 muestra la complejidad de los proyectos desarrollados por los encuestados, de los cuales 8.3% trabaja en proyectos nada complejos, 75% de los encuestados considera que los proyectos en los que traban son medianamente complejos, 16.6% trabaja con proyectos complejos. Aquí se explicó previamente a los encuestados la definición y las características que tienen los proyectos complejos.

Tabla 12
Complejidad de los proyectos de software desarrollados

Complejidad del proyecto	frecuencia
Nada complejo (1)	1
Muy poco complejo (2)	2
Medianamente complejo (3)	7
Complejo (4)	1
muy complejo (5)	1

La Tabla 13 muestra el alcance geográfico de los proyectos de software en los que trabajan los encuestados, 25% trabajan con proyectos a nivel internacionales (o que por lo menos han desarrollado un proyecto de alcance internacional), 41.67% son alcance a nivel nacional y el 33.33% trabajan en proyectos de alcance local.

Tabla 13
Alcance de los proyectos desarrollados

Alcance del proyecto	frecuencia
A nivel internacional	3
A nivel nacional	5
A nivel local	4

La Tabla 14 muestra la cantidad de proyectos desarrollados por los encuestados desde que iniciaron como trabajadores de la industria del software hasta la fecha de la

encuesta, los resultados muestran que el 75% trabajaron hasta la fecha hasta un total de 10 proyectos y el 25% trabajó con 11 o más proyectos.

Tabla 14
Cantidad de proyectos desarrollados

Cantidad de proyectos	frecuencia
entre 2 y 4 proyectos	2
entre 5 y 10 proyectos	7
entre 11 y 20 proyectos	2
entre 21 y 50 proyectos	1

La Tabla 15 muestra el nivel de experiencia de trabajo en expresado en años de los encuestados, los resultados muestran que el 91.67% trabaja entre 2 y 4 años en la industria del software y 8.3% tiene menos de 2 años trabajando en proyectos de software.

Tabla 15
Experiencia en años de los desarrolladores

Experiencia en años	Frecuencia
menos de 1 año	0
entre 1.01 y 2 años	1
entre 2.01 y 4 años	7
entre 4.01 y 8 años	3
entre 8.01 y 16 años	1

La Tabla 16 muestra el resumen del modelo de software utilizado por los encuestados, 75% de los encuestados trabajan mayormente con metodologías ágiles y por prototipos y el 25% utiliza los métodos tradicionales como por ejemplo el modelo en Cascada.

Tabla 16
Modelo de proceso de software utilizado por los encuestados

Modelo de software utilizado	Frecuencia
Cascada	1
De acuerdo al tipo proyecto	1
Modelo Iterativo Incremental	1
Por prototipos	1
SCRUM	5
XP	3

La Tabla 17 muestra el resumen del grado de participación del cliente/usuario en el desarrollo del producto de software, los resultados muestran que el 50% de los encuestados manifiesta que el cliente no se involucra en el desarrollo del proyecto de software, el 33.33% manifiesta que el cliente si se involucra y un 17.67% manifiesta que tiene un posición neutral frente a la pregunta que se le planteó.

Tabla 17
Participación del cliente en los proyectos de software

No participa el cliente	frecuencia
Totalmente en Desacuerdo (1)	1
En Desacuerdo (2)	3
Neutral (3)	2
De Acuerdo (4)	3
Totalmente de Acuerdo (5)	3

La Tabla 18 muestra el resumen sobre la pregunta relacionada al software que mejora con el transcurso del tiempo, según la opinión de los encuestados el 100% afirma que el software mejora en el transcurso del tiempo.

Tabla 18
El software mejora con el transcurso del tiempo

Software mejora con el tiempo	frecuencia
No participa el cliente	0
Totalmente en Desacuerdo (1)	0
En Desacuerdo (2)	0
Neutral (3)	0
De Acuerdo (4)	4
Totalmente de Acuerdo (5)	8

La Tabla 19 muestra el resumen de la opinión dada por los encuestados respecto al retraso que tiene los proyectos de software, el 50% de los encuestados manifiesta que está de acuerdo en que los proyectos de software se retrasan en su entrega y el trabajan con metodologías ágiles y por prototipos, y el 25% manifiesta que está en desacuerdo y el 25% opina neutralmente.

Tabla 19
El software y el retraso

El software se retrasa	frecuencia
Totalmente en Desacuerdo (1)	0
En Desacuerdo (2)	3
Neutral (3)	3
De Acuerdo (4)	2
Totalmente de Acuerdo (5)	4

La Tabla 20 muestra el resumen de los problemas más relevantes respecto al desarrollo de proyectos de software, es preciso aclarar que esta pregunta con varias opciones y el encuestado podía marcar varias alternativas (preguntas con opción múltiple). Los resultados muestran que 31% opina que el problema más importante es que "el cliente/usuario cambió los requisitos durante el desarrollo del proyecto", luego el 20.6% de los encuestados manifiesta que los "requisitos no se entendieron bien al inicio" y también el 20.6% manifiesta que "el cliente/usuario no explicó bien los requisitos" y las otras razones son que no se tuvo un plan de pruebas, el nivel técnico de los programadores es bajo, requiere excesiva documentación, existe cambio de prioridades y existe una organización burocrática, que en total suman el 27.8%.

Tabla 20
Problemas en el desarrollo de proyectos de software

Tipo de problema	frecuencia
Los requisitos no se entendieron bien al inicio	6
No se tuvo un plan de pruebas y el software empezó a fallar al final	1
El nivel técnico de los programadores es bajo y demoran al implementar	3
El cliente/usuario no explicó bien los requisitos	6
El cliente/usuario cambió los requisitos durante el desarrollo del proyecto	9
Se requiere excesiva documentación	1
Cambio de prioridades	2
Organización burocrática	1

4.1.3 Resultados de la Entrevista

En las entrevistas participaron 12 personas que actuaron como clientes/usuarios del sistema, los participantes fueron seleccionados entre Especialistas de UGEL,

Especialistas de DRE, Directores de Institución Educativa y profesores, todos ellos participaron en las cuatro fases del modelo que plantea esta investigación: "modelo co-participativo para la construcción de software". Las cuatro fases en las que participaron los clientes/usuarios son: 1) Determinación de Requisitos con el Usuario, 2) Diseño Centrado en el Usuario, 3) Programación y Pruebas con el Usuario y 4) Implantación y Mantenimiento con el Usuario.

En las entrevistas se les hizo preguntas relacionadas a la experiencia que han tenido en participar en esta nueva forma de involucrar al cliente/usuario en el desarrollo del sistema de tal forma que coopera y se involucra permanentemente en el desarrollo del producto de software.

En las entrevistas con los clientes/usuarios se abordó básicamente 3 preguntas, y cuyo resumen se presentan a continuación:

¿Cuál es su opinión respecto a que Ud. como cliente/usuario ha participado activamente en el desarrollo del software SIREMAP?. A diferencia de la construcción de otros sistemas, la experiencia con SIREMAP fue bastante interesante, porque nosotros nos reuníamos con los desarrolladores y ellos nos entrevistaban para pedir información, luego nos pedían participar en el diseño de los formularios y reportes, en las pruebas del sistema; es decir en todo momento estábamos participando en la elaboración del software. Prácticamente se desarrolló de acuerdo a lo que se había pedido al inicio. En cambio en otros sistemas, solo nos reunían en la primera fase del desarrollo y luego en la entrega final.

¿Cuál es la diferencia entre utilizar SIREMAP y hacerlo tradicionalmente el monitoreo al docente en aula?. Los participantes afirmaron que antes era bien complicado hacer los consolidados anualmente, porque cada especialista hacía su trabajo en hojas impresas individuales, otros hacían en cuadernos, y el problema era que para consolidar demoraban mucho y gastaban mucho papel. Incluso había años en que el consolidado salía para diciembre y a veces enero del siguiente año, lo cual no era pertinente, porque durante el año no se sabía que Instituciones Educativas requerían ayuda. Ahora con el sistema, es fácil de obtener los reportes y se puede obtener los resultados en unos cuantos segundos, incluso se puede buscar a cualquier docente y ver cuántas veces lo visitaron y cuáles son sus puntuaciones; esto permite planificar intervenciones

inmediatas y durante cualquier época del año. En concreto SIREMAP permite tomar decisiones en forma oportuna y rápida.

¿Cree UD. que SIREMAP facilita el proceso de monitoreo a la sesión de aprendizaje del docente en aula?. Los participantes afirman que sí, porque el sistema permite realizar el monitoreo a la sesión de aprendizaje de los docentes, sin embargo sería bueno que posteriormente se pueda agregar la función de tomar fotos y grabar un video de la sesión de clase.

4.2 Discusión de los resultados

a) El modelo de desarrollo de software propuesto es "co-participativo" y se caracteriza porque el cliente "coopera" y "participa" activamente y con compromiso durante todo el proceso de construcción del software, tal como lo manifiestan algunos autores (McInnerney & Roberts, 2004; Schöttle et al., 2014), (Davidson & Major, 2014) y (Halynska, 2017).

b) Utilizando el modelo co-participativo, se construyó el software denominado "Sistema Regional de Monitoreo y Acompañamiento Pedagógico - SIREMAP" para apoyar en la observación del desempeño docente en aula. Este modelo está alineado con el concepto de Diseño Centrado en el Usuario (DCU) donde el eje fundamental es el usuario, tal como lo mencionan las investigaciones anteriores a este trabajo (Bergadano et al., 2014), (Saiedian & Dale, 2000), (Hassan Montero & Martín Fernández, 2013) . Por otro lado, este modelo se alinea a los descritos en "*Semio-Participatory Framework for Interaction Design of Educational Software"* (Rosa & Matos, 2016) y "*A discussion on social software: concept, building blocks and challenges"* (Pereira et al., 2010), en los cuales también se prioriza la participación del cliente/usuario durante todo el proceso de desarrollo de software.

c) De acuerdo a los resultados obtenidos se tiene los problemas más resaltantes con el desarrollo de software están relacionados con: los requisitos son cambiantes, escasa participación del cliente o usuario, el cliente busca el menor costo posible de un producto de software, más no le da mucha importancia a la calidad, esto se complementa con lo manifestado por Lomprey (Lomprey & Hernandez, 2008).

d) Según los resultados obtenidos un 66.67% utiliza metodologías ágiles en el desarrollo de software. El modelo co-participativo propuesto, es un modelo que está

basado en las metodologías ágiles XP y SCRUM (Abrahamsson et al., 2017) y utiliza algunos conceptos de las metodologías tradicionales (Cascada), porque hoy en día es posible combinar dos o más modelos, de tal forma que se aprovecha las bondades de cada uno de ellos; esto está alineado con las investigaciones recientes de la comunidad científica, en la cual se manifiesta que se puede tener modelos híbridos para el desarrollo de software (Aitken & Ilango, 2013; Theocharis et al., 2015).

CONCLUSIONES

- Esta investigación propone un "modelo co-participativo" para la construcción de software. La principal característica del modelo es que el cliente/usuario participa en forma activa y comprometida durante todas las fases que son: Determinación de Requisitos con el Usuario (DRU), Diseño Centrado en el Usuario (DCU), Programación y Pruebas con el Usuario (PPU) e Implantación y Mantenimiento con el Usuario (IMU).

- Se construyó un software para apoyar en la observación del desempeño docente en aula. Este modelo permite que los usuarios estén satisfechos con el producto desarrollado y ellos consideran que el hecho de participar durante el proceso de desarrollo de software permitió que el diseño y correcciones se realicen en forma inmediata.

- Las dificultades más resaltantes que tienen los desarrolladores de software son: a) los requisitos son cambiantes durante el proceso de desarrollo del software debido a que los clientes tienen escasa participación o no se describen adecuadamente desde el inicio del proyecto; b) el cliente/usuario no le da mucha importancia a la parte de análisis, diseño y pruebas del sistema porque piensa que el ingeniero informático es el responsable de dar una solución óptima; es decir, deja todo en manos del equipo de desarrollo debido a que está muy ocupado con sus actividades cotidianas; c) generalmente, el cliente busca el menor costo posible de un producto de software, más no le da mucha importancia a la calidad.

- Los proyectos de software que son de escala pequeña o mediana y que tienen corta duración, un 66.67% utiliza metodología XP o SCRUM, un 33.33% utilizan modelos tradicionales como Cascada, Iterativo o crean su propio modelo de trabajo.

RECOMENDACIONES

Las recomendaciones de este trabajo de investigación son:

- En el proceso de desarrollo de software, es altamente recomendable que el cliente o usuario participe en todas las fases de la construcción del software, porque permite que el software tenga menos tasa de errores, el cliente esté satisfecho con el producto y se termine el producto dentro del tiempo establecido.
- El modelo co-participativo en el desarrollo de software se ha probado y ha dado buenos resultados en el área de educación, sin embargo, es recomendable probar en el desarrollo de software para otras áreas.

BIBLIOGRAFÍA

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. *ArXiv Preprint ArXiv:1709.08439*.

Aitken, A., & Ilango, V. (2013). A Comparative Analysis of Traditional Software Engineering and Agile Software Development. In *2013 46th Hawaii International Conference on System Sciences* (pp. 4751–4760). https://doi.org/10.1109/HICSS.2013.31

Andersen, R., & Mørch, A. I. (2016). Mutual development in mass collaboration: Identifying interaction patterns in customer-initiated software product development. *Computers in Human Behavior*, *65*, 77–91.

Apesoft, P. (2017). Asociación Peruana de Software y Tecnologías. Retrieved from http://www.apesoft.org/

Aranda, T., & Araújo, E. G. (2009). Técnicas e instrumentos cualitativos de recogida de datos. *Editorial EOS*, 284.

Ashkenas, R. (2015). There'sa Difference Between Cooperation and Collaboration. *Harvard Business Review, Avril*.

Baranauskas, M. C. C. (2014). Socially aware computing. In *Proceedings of International Conference on Engineering and Computer Education* (Vol. 6).

Baranauskas, M. C. C., & Bonacin, R. (2008). Design—indicating through signs. *Design Issues*, *24*(3), 30–45.

Beck, K. (2001). Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/

Bergadano, F., Bosio, G., & Spagnolo, S. (2014). Supporting collaboration between customers and developers: a framework for distributed, agile software development. *International Journal of Distributed Systems and Technologies (IJDST)*, *5*(2), 1–16.

Canós, J. H., & Letelier, M. P. P. (2012). Metodologías ágiles en el desarrollo de software. In *Metodologías Ágiles en el Desarrollo de Software*.

Carroll, J. M., & Rosson, M. B. (2007). Participatory design in community informatics. *Design Studies*, *28*(3), 243–261.

Davidson, N., & Major, C. H. (2014). Boundary crossings: Cooperative learning, collaborative learning, and problem-based learning. *Journal on Excellence in College Teaching*, *25*.

Fallman, D. (2003). Design-oriented human-computer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 225–232).

Galeano, R. (2017). Diseño centrado en el usuario. *Revista Q*, *2*(4).

Goetz, J. P., Goetz, M. D. P., & LeCompte, M. D. (1988). *Etnografía y diseño cualitativo en investigación educativa*. Ediciones Morata,.

Halynska, Y. (2017). Collaboration or Cooperation: Concept Definitions When Merging Interests of the State and Extractive Companies of Ukraine. *Journal of Research in Business, Economics and Management*, *8*(4), 1470–1476.

Hassan Montero, Y., & Martín Fernández, F. J. (2013). Propuesta de adaptación de la metología de diseño centrado en el usuario para el desarrollo de sitios web accesibles.

Hussain, Z., Slany, W., & Holzinger, A. (2009). Current state of agile user-centered design: A survey. In *Symposium of the Austrian HCI and Usability Engineering Group* (pp. 416–427).

Kruchten, P. (2004). *The rational unified process: an introduction* (Thirth). Addison-Wesley Professional.

Liu, S., Li, W., & Liu, K. (2015). Assessing pragmatic interoperability for process alignment in collaborative working environment. In *International Conference on Informatics and Semiotics in Organisations* (pp. 60–69).

Lomprey, G., & Hernandez, S. (2008). La importancia de la Calidad en el desarrollo de productos de software. *Recuperado El*.

Loup-Escande, E., Burkhardt, J.-M., Christmann, O., & Richir, S. (2014). Needs' elaboration between users, designers and project leaders: Analysis of a design process of a virtual reality-based software. *Information and Software Technology*, *56*(8), 1049–1061.

Lowdermilk, T. (2013). *Design centrado no usuário*. Novatec Editora.

McInnerney, J. M., & Roberts, T. S. (2004). Collaborative or cooperative learning.

Online Collaborative Learning: Theory and Practice, 203–214.

Naur, P., & Randell, B. (1969). Software Engineering: Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany, 7th to 11th October 1968.

Perdomo, E. G., Cardozo, M. A. T., Perdomo, C. A. C., & Serrezuela, R. R. (2017). A Review of the User Based Web Design: Usability and Information Architecture. *International Journal of Applied Engineering Research*, *12*(21), 11685–11690.

Pereira, R., & Baranauskas, M. C. C. (2015). A value-oriented and culturally informed approach to the design of interactive systems. *International Journal of Human-Computer Studies*, *80*, 66–82.

Pereira, R., Baranauskas, M. C. C., & Silva, S. R. P. (2010). A discussion on social software: concept, building blocks and challenges. *International Journal for Infonomics*, *3*(4), 533–542.

Pressman, R. (2001). *Ingeniería de software un enfoque práctico* (Quinta). Mc Graw Hill.

Rosa, J. C. S., & Matos, E. (2016). Semio-Participatory Framework for Interaction Design of Educational Software. In *IHC*.

Saiedian, H., & Dale, R. (2000). Requirements engineering: making the connection between the software developer and customer. *Information and Software Technology*, *42*(6), 419–428.

Schöttle, A., Haghsheno, S., & Gehbauer, F. (2014). Defining cooperation and collaboration in the context of lean construction. In *Proc. 22nd Ann. Conf. of the Int'l Group for Lean Construction* (pp. 1269–1280).

Sengers, P., Boehner, K., David, S., & Kaye, J. (2005). Reflective design. In *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility* (pp. 49–58).

Software, G. (2018). Industria del software en el Perú. Retrieved from https://sg.com.mx/revista/53/industria-software#.We5o01vWzIU

Sommerville, I. (2005). *Ingeniería del software* (Sétima). Pearson Addison Wesley.

Theocharis, G., Kuhrmann, M., Münch, J., & Diebold, P. (2015). Is water-scrum-fall

reality? on the use of agile and traditional development practices. In *International Conference on Product-Focused Software Process Improvement* (pp. 149–166).

Torres, M., Paz, K., & Salazar, F. (2006). Métodos de recolección de datos para una investigación. *Rev. Electrónica Ingeniería Boletín*, *3*, 12–20.

Tschimmel, K. (2012). Design Thinking as an effective Toolkit for Innovation. In *ISPIM Conference Proceedings* (p. 1).

Universidades Chilenas, grupo operativo. (2010). *Diseño curricular basado en competencias y aseguramiento de la calidad en la educación superior*. Retrieved from http://www.edumovil.cl/tportal/portales/tp4964b0e1bk102/uploadImg/File/DisCurricBasCompetAsegurCalidadCINDA2008.pdf

Varian, H. R. (2005). Universal access to information. *Communications of the ACM*, *48*(10), 65–66.

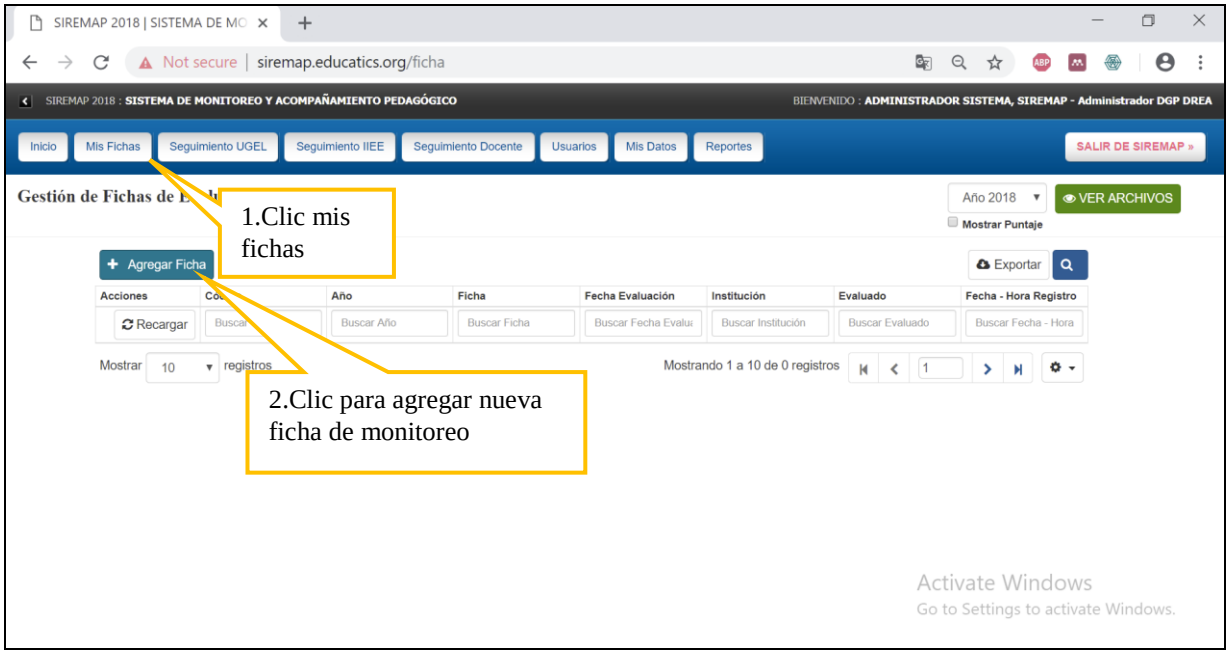
Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Intellect Books.

Wölbling, A., Krämer, K., Buss, C. N., Dribbisch, K., LoBue, P., & Taherivand, A. (2012). Design thinking: An innovative concept for developing user-centered software. In *Software for People* (pp. 121–136). Springer.

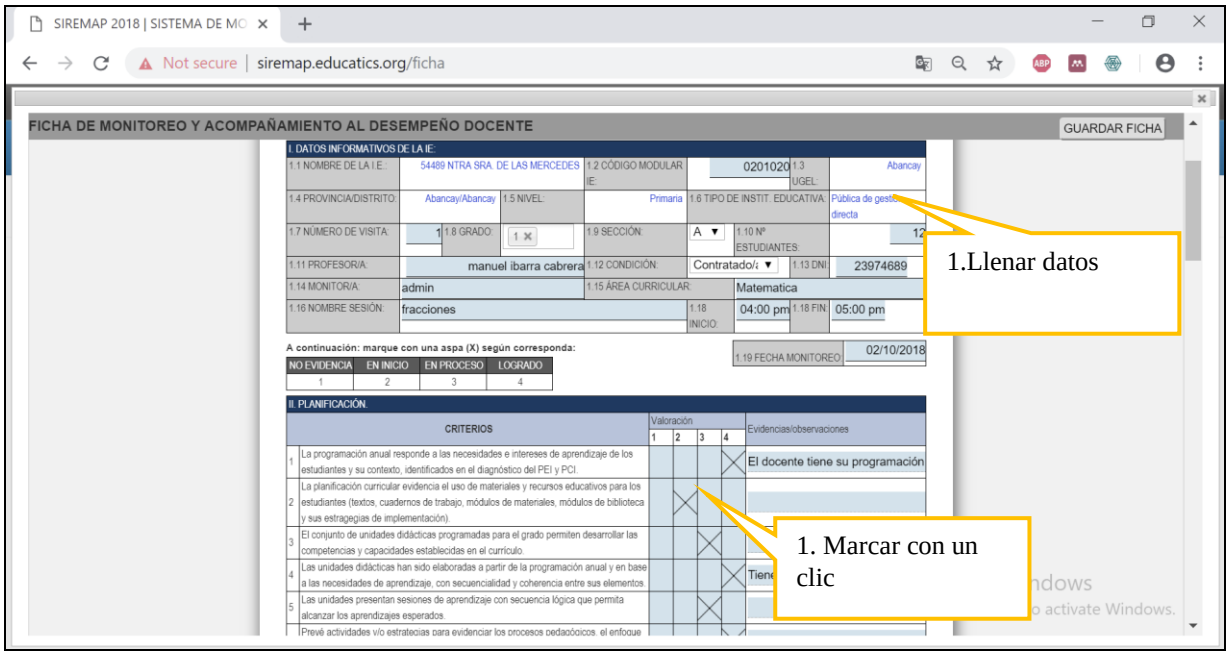
ANEXOS

Anexo 1. Interfaces de usuario de Sistema de Monitoreo y Acompañamiento Pedagógico (SIREMAP)

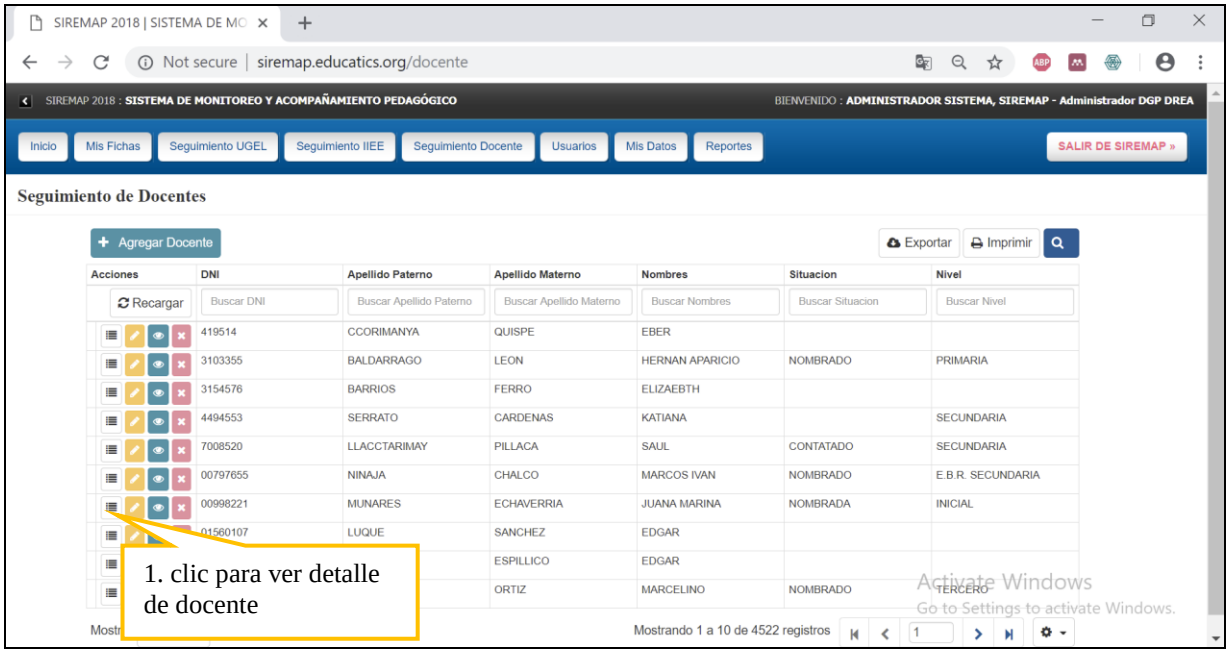
Ingreso al sistema



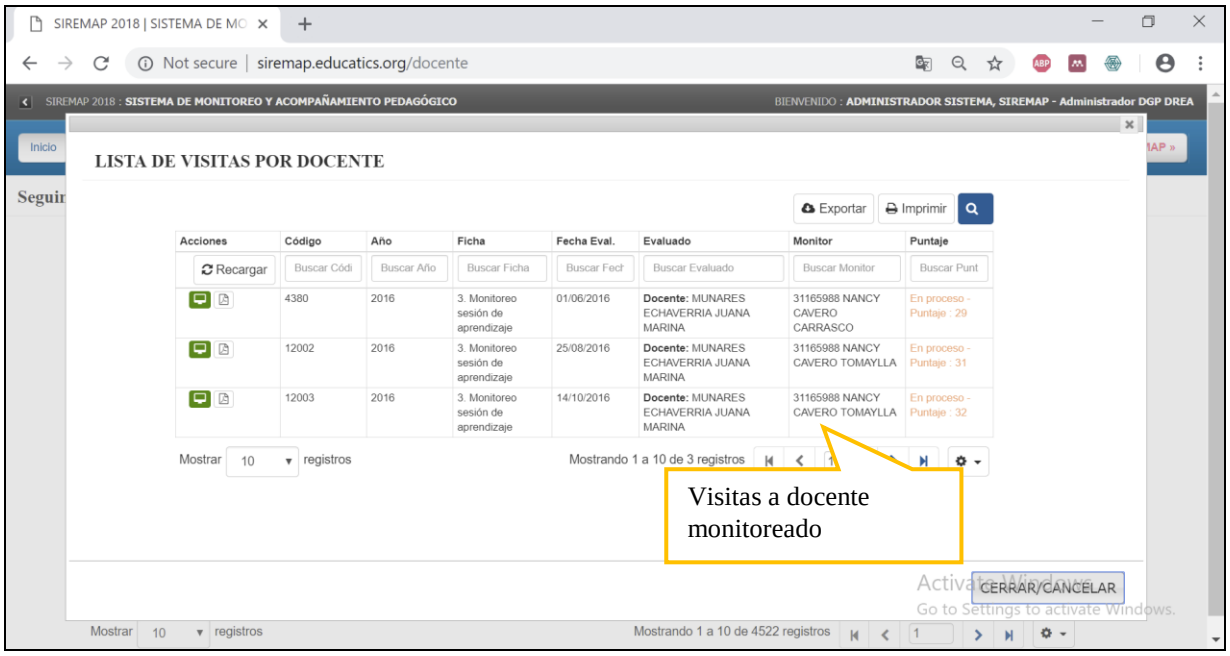
Seleccionar Ficha



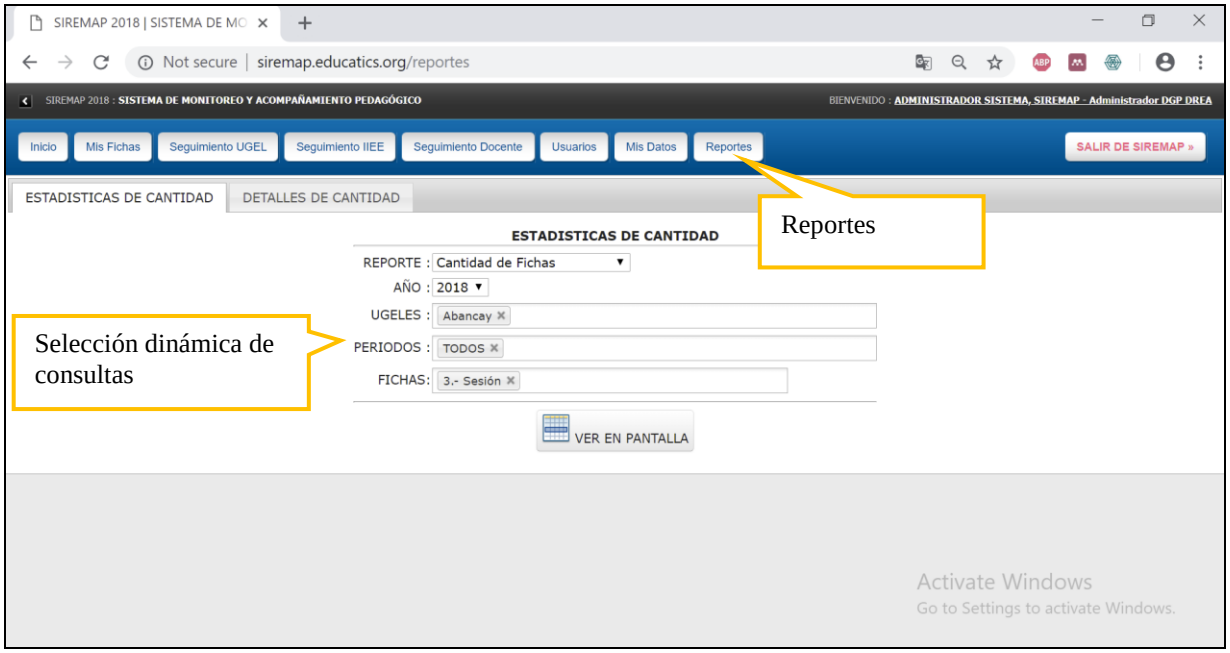
Llenar ficha de monitoreo



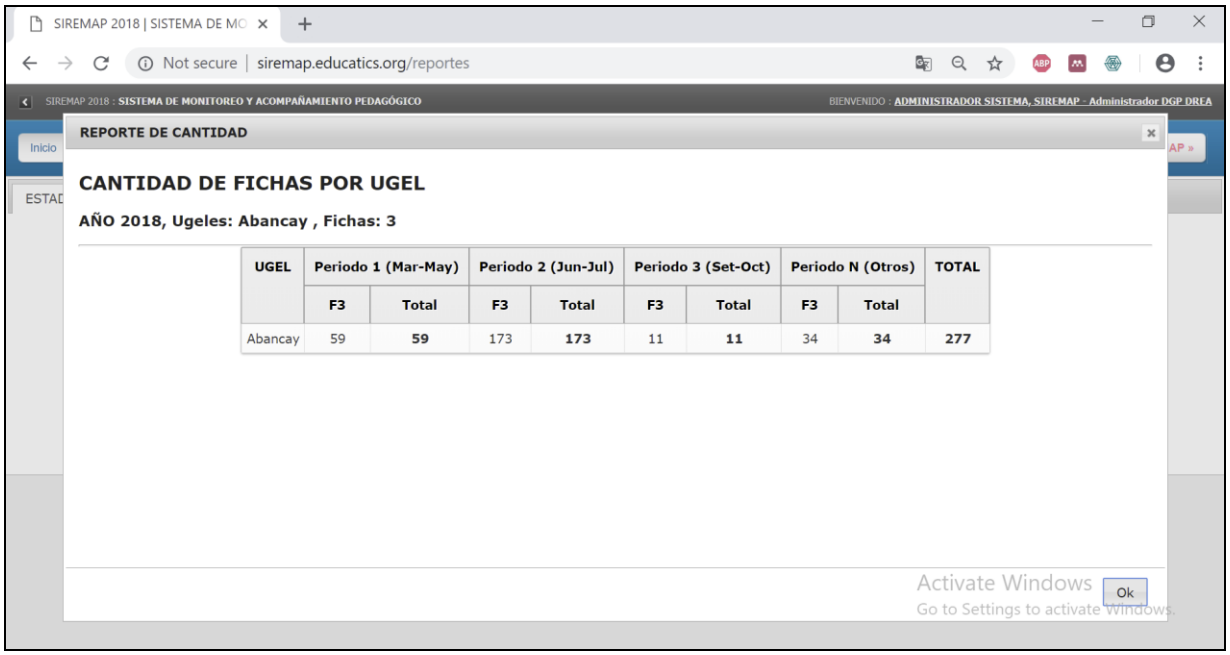
Lista de docentes monitoreados



Reporte de monitoreo realizadas al docente



Reporte de Monitoreo con consultas dinámicas



UGEL	Periodo 1 (Mar-May)		Periodo 2 (Jun-Jul)		Periodo 3 (Set-Oct)		Periodo N (Otros)		TOTAL
	F3	Total	F3	Total	F3	Total	F3	Total	
Abancay	59	**59**	173	**173**	11	**11**	34	**34**	**277**

Cantidad de fichas monitoreadas por periodo.

Anexo 2. Base de datos de SIREMAP

```
CREATE TABLE `ceroficha` (
  `idusuario` int(11) DEFAULT NULL,
  `dni` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `compromiso` (
  `idficha` int(11) NOT NULL,
  `numcompromiso` varchar(2) COLLATE utf8_unicode_ci NOT NULL,
  `bloque` varchar(2) COLLATE utf8_unicode_ci DEFAULT NULL,
  `actividad` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `comentario` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `compromiso` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `estado` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `detalle` (
  `idficha` int(11) NOT NULL,
  `numcompromiso` varchar(2) NOT NULL,
  `numitem` varchar(2) NOT NULL,
  `REF0` varchar(3) DEFAULT NULL,
  `REF1` varchar(3) DEFAULT NULL,
  `REF2` varchar(3) DEFAULT NULL,
  `REF3` varchar(3) DEFAULT NULL,
  `REF4` varchar(3) DEFAULT NULL,
  `REF5` varchar(3) DEFAULT NULL,
  `REF6` varchar(3) DEFAULT NULL,
  `REF7` varchar(3) DEFAULT NULL,
  `REF8` varchar(3) DEFAULT NULL,
  `REF9` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `director` (
  `iddirector` int(11) NOT NULL,
  `codmodular` varchar(7) COLLATE utf8_unicode_ci DEFAULT NULL,
  `fecharegistro` datetime DEFAULT NULL,
  `anio` int(11) DEFAULT NULL,
  `nombre` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `docente` (
  `dni` varchar(8) NOT NULL,
  `apaterno` varchar(30) DEFAULT NULL,
  `amaterno` varchar(30) DEFAULT NULL,
  `nombres` varchar(30) DEFAULT NULL,
  `situacion` varchar(15) DEFAULT NULL,
  `jornada` varchar(4) DEFAULT NULL,
  `nivel` varchar(30) DEFAULT NULL,
  `estado` varchar(100) DEFAULT NULL,
  `fnacimiento` date DEFAULT NULL,
  `finicio` date DEFAULT NULL,
  `ftermino` date DEFAULT NULL,
  `ley` varchar(12) DEFAULT NULL,
  `escala` varchar(4) DEFAULT NULL,
  `especialidad` varchar(30) DEFAULT NULL,
  `segespecialidad` varchar(30) DEFAULT NULL,
  `eib` varchar(2) DEFAULT NULL,
  `lengua` varchar(15) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `ficha` (
  `idficha` int(11) NOT NULL,
```

```
  `numficha` int(11) DEFAULT NULL,
  `anio` varchar(4) DEFAULT NULL,
  `grupo` varchar(4) DEFAULT NULL,
  `ugel` varchar(30) DEFAULT NULL,
  `codmodular` varchar(7) DEFAULT NULL,
  `dnidoc` varchar(8) DEFAULT NULL,
  `dnimon` varchar(8) DEFAULT NULL,
  `fecha` date DEFAULT NULL,
  `numvisita` int(11) DEFAULT NULL,
  `REF0` varchar(100) DEFAULT NULL,
  `REF1` varchar(100) DEFAULT NULL,
  `REF2` varchar(100) DEFAULT NULL,
  `REF3` varchar(100) DEFAULT NULL,
  `REF4` varchar(100) DEFAULT NULL,
  `REF5` varchar(100) DEFAULT NULL,
  `REF6` varchar(100) DEFAULT NULL,
  `REF7` varchar(100) DEFAULT NULL,
  `REF8` varchar(100) DEFAULT NULL,
  `REF9` varchar(100) DEFAULT NULL,
  `REF10` varchar(250) DEFAULT NULL,
  `tiempo` int(11) DEFAULT NULL,
  `idusuario` int(11) DEFAULT NULL,
  `fecharegistro` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `iiee` (
  `codmodular` varchar(7) NOT NULL,
  `codlocal` varchar(6) DEFAULT NULL,
  `iiee` varchar(50) DEFAULT NULL,
  `nivel` varchar(40) DEFAULT NULL,
  `caracteristica` varchar(30) DEFAULT NULL,
  `gestion` varchar(30) DEFAULT NULL,
  `dependencia` varchar(30) DEFAULT NULL,
  `director` varchar(60) DEFAULT NULL,
  `telefono` varchar(12) DEFAULT NULL,
  `correo` varchar(50) DEFAULT NULL,
  `web` varchar(50) DEFAULT NULL,
  `direccion` varchar(80) DEFAULT NULL,
  `localidad` varchar(40) DEFAULT NULL,
  `centro_poblado` varchar(40) DEFAULT NULL,
  `area` varchar(10) DEFAULT NULL,
  `departamento` varchar(10) DEFAULT NULL,
  `provincia` varchar(15) DEFAULT NULL,
  `distrito` varchar(300) DEFAULT NULL,
  `ugel` varchar(20) NOT NULL,
  `lengua` varchar(10) DEFAULT NULL,
  `turno` varchar(40) DEFAULT NULL,
  `alumnos` int(11) DEFAULT NULL,
  `docentes` int(11) DEFAULT NULL,
  `secciones` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `indicador` (
  `idficha` int(11) NOT NULL,
  `indice` int(11) NOT NULL,
  `REF01` varchar(5) COLLATE utf8_unicode_ci DEFAULT NULL,
  `REF02` varchar(5) COLLATE utf8_unicode_ci DEFAULT NULL,
  `REF03` varchar(5) COLLATE utf8_unicode_ci DEFAULT NULL,
  `REF04` varchar(5) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `menu` (
  `idmenu` int(11) NOT NULL,
  `titulo` varchar(45) DEFAULT NULL,
  `url` varchar(50) DEFAULT NULL,
  `orden` int(11) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `permiso` (
  `idpermiso` int(11) NOT NULL,
  `idusuario` int(11) NOT NULL,
  `idmenu` int(11) NOT NULL,
  `estado` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `puntaje` (
  `idficha` int(11) NOT NULL,
  `numcompromiso` int(11) NOT NULL,
  `inicio` int(11) DEFAULT NULL,
  `proceso` int(11) DEFAULT NULL,
  `logrado` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `tmpiiee` (
  `codmodular` varchar(11) CHARACTER SET utf8 NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `ugel` (
  `ugel` varchar(30) NOT NULL,
  `departamento` varchar(45) DEFAULT NULL,
  `provincia` varchar(45) DEFAULT NULL,
  `distrito` varchar(45) DEFAULT NULL,
  `director` varchar(45) DEFAULT NULL,
  `telefono` varchar(45) DEFAULT NULL,
  `espini` int(11) DEFAULT NULL,
  `esppri` int(11) DEFAULT NULL,
  `espsec` int(11) DEFAULT NULL,
  `docini` int(11) DEFAULT NULL,
  `docpri` int(11) DEFAULT NULL,
  `docsec` int(11) DEFAULT NULL,
  `ieini` int(11) DEFAULT NULL,
  `iepri` int(11) DEFAULT NULL,
  `iesec` int(11) DEFAULT NULL,
  `estini` int(11) DEFAULT NULL,
  `estpri` int(11) DEFAULT NULL,
  `estsec` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `usuario` (
  `idusuario` int(11) NOT NULL,
  `dni` varchar(8) NOT NULL,
  `clave` varchar(100) NOT NULL,
  `area` enum('IIEE','DGP DREA','UGEL Abancay','UGEL Andahuaylas','UGEL Antabamba','UGEL
Aymaraes','UGEL Chincheros','UGEL Cotabambas','UGEL Grau','UGEL Huancarama') NOT NULL DEFAULT
'IIEE',
  `cargo` enum('Administrador','Especialista','Monitor','Acompañante','Director') NOT NULL DEFAULT 'Monitor',
  `estado` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `compromiso`
  ADD PRIMARY KEY (`idficha`,`numcompromiso`),
  ADD KEY `fk_compromiso_ficha1_idx` (`idficha`);

ALTER TABLE `detalle`
  ADD PRIMARY KEY (`idficha`,`numcompromiso`,`numitem`);

ALTER TABLE `director`
  ADD PRIMARY KEY (`iddirector`),
  ADD KEY `direccodmod_idx` (`codmodular`);

ALTER TABLE `docente`
  ADD PRIMARY KEY (`dni`);
```

```
ALTER TABLE `ficha`
  ADD PRIMARY KEY (`idficha`),
  ADD KEY `codmodular` (`codmodular`),
  ADD KEY `dnidoc` (`dnidoc`),
  ADD KEY `dnimon` (`dnimon`),
  ADD KEY `idusuario` (`idusuario`),
  ADD KEY `ugel` (`ugel`);

ALTER TABLE `iiee`
  ADD PRIMARY KEY (`codmodular`),
  ADD KEY `ugel` (`ugel`);

ALTER TABLE `indicador`
  ADD PRIMARY KEY (`idficha`,`indice`);

ALTER TABLE `menu`
  ADD PRIMARY KEY (`idmenu`);

ALTER TABLE `permiso`
  ADD PRIMARY KEY (`idpermiso`),
  ADD KEY `idusuario` (`idusuario`),
  ADD KEY `idmenu` (`idmenu`);

ALTER TABLE `puntaje`
  ADD PRIMARY KEY (`idficha`,`numcompromiso`);

ALTER TABLE `tmpiiee`
  ADD PRIMARY KEY (`codmodular`);

ALTER TABLE `ugel`
  ADD PRIMARY KEY (`ugel`);

ALTER TABLE `usuario`
  ADD PRIMARY KEY (`idusuario`),
  ADD UNIQUE KEY `dni` (`dni`,`area`);


ALTER TABLE `director`
  MODIFY `iddirector` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `ficha`
  MODIFY `idficha` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `menu`
  MODIFY `idmenu` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `permiso`
  MODIFY `idpermiso` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `usuario`
  MODIFY `idusuario` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `compromiso`
  ADD CONSTRAINT `fk_compromiso_ficha1` FOREIGN KEY (`idficha`) REFERENCES `ficha` (`idficha`);

ALTER TABLE `detalle`
  ADD CONSTRAINT `detalle_ibfk_1` FOREIGN KEY (`idficha`) REFERENCES `ficha` (`idficha`);

ALTER TABLE `ficha`
  ADD CONSTRAINT `ficha_ibfk_1` FOREIGN KEY (`codmodular`) REFERENCES `iiee` (`codmodular`),
  ADD CONSTRAINT `ficha_ibfk_4` FOREIGN KEY (`idusuario`) REFERENCES `usuario` (`idusuario`),
  ADD CONSTRAINT `ficha_ibfk_5` FOREIGN KEY (`ugel`) REFERENCES `ugel` (`ugel`);

ALTER TABLE `iiee`
  ADD CONSTRAINT `iiee_ibfk_1` FOREIGN KEY (`ugel`) REFERENCES `ugel` (`ugel`);

ALTER TABLE `indicador`
```

```
  ADD CONSTRAINT `fk_indicador` FOREIGN KEY (`idficha`) REFERENCES `ficha` (`idficha`) ON DELETE
NO ACTION ON UPDATE NO ACTION;

ALTER TABLE `permiso`
  ADD CONSTRAINT `permiso_ibfk_1` FOREIGN KEY (`idusuario`) REFERENCES `usuario` (`idusuario`),
  ADD CONSTRAINT `permiso_ibfk_2` FOREIGN KEY (`idmenu`) REFERENCES `menu` (`idmenu`);

ALTER TABLE `puntaje`
  ADD CONSTRAINT `puntaje_ibfk_1` FOREIGN KEY (`idficha`) REFERENCES `ficha` (`idficha`);

ALTER TABLE `usuario`
  ADD CONSTRAINT `usuario_ibfk_1` FOREIGN KEY (`dni`) REFERENCES `docente` (`dni`);
```

Anexo 3. Ficha de monitoreo a sesión de aprendizaje

PERÚ Ministerio de Educación

GOBIERNO REGIONAL DE APURÍMAC

DIRECCIÓN REGIONAL DE EDUCACIÓN DE APURÍMAC - DIRECCIÓN DE GESTIÓN PEDAGÓGICA

FICHA DE MONITOREO Y ACOMPAÑAMIENTO AL DESEMPEÑO DOCENTE

2018
INSTITUCIÓN EDUCATIVA

El llenado de esta ficha es realizada por el coordinador o especialista de la UGEL. La fuente principal es la observación de una sesión de aprendizaje. El aplicador/a debe verificar la información proporcionada en fuentes físicas o hechos visibles.
En la valoración de los ítemes, marcar con un aspa (X).

I. DATOS INFORMATIVOS DE LA DRE:

1.1 NOMBRE DE LA IE:		1.2 CÓDIGO MODULAR IE:			1.3 UGEL:		
1.4 PROVINCIA/DISTRITO:		1.5 NIVEL:		1.6 TIPO DE INSTITUCIÓN EDUCATIVA:			
1.7 NÚMERO DE VISITA:		1.8 GRADO:		1.9 SECCIÓN:		1.10 N° ESTUDIANTES:	
1.11 PROFESOR/A:				1.12 CONDICIÓN:		1.13 DNI:	
1.14 MONITOR/A				1.15 ÁREA CURRICULAR:			
1.16 NOMBRE DE LA SESIÓN:					1.18 INICIO:	1.19. FIN:	
					1.20 FECHA MONITOREO:		

A continuación: marque con una aspa (X) según corresponda:

NO EVIDENCIA	EN INICIO	EN PROCESO	LOGRADO
1	2	3	4

II. PLANIFICACIÓN.

	CRITERIOS	Valoración 1	2	3	4	Evidencias/observaciones
1	La programación anual responde a las necesidades e intereses de aprendizaje de los estudiantes y su contexto, identificados en el diagnóstico del PEI y PCI.					
2	La planificación curricular evidencia el uso de materiales y recursos educativos para los estudiantes (textos, cuadernos de trabajo, módulos de materiales, módulos de biblioteca y sus estragegias de implementación).					
3	El conjunto de unidades didácticas programadas para el grado permiten desarrollar las competencias y capacidades establecidas en el currículo.					
4	Las unidades didácticas han sido elaboradas a partir de la programación anual y en base a las necesidades de aprendizaje, con secuencialidad y coherencia entre sus elementos.					
5	Las unidades presentan sesiones de aprendizaje con secuencia lógica que permita alcanzar los aprendizajes esperados.					
6	Prevé actividades y/o estrategias para evidenciar los procesos pedagógicos, el enfoque del Área y alcanzar los propósitos de aprendizaje.					
7	Los instrumentos de evaluación formativa y/o sumativa seleccionados permitirán recoger evidencias de los desempeños previstos para la unidad.					
8	El tiempo en la sesión de aprendizaje estimado para actividades de inicio, desarrollo y cierre permitirá desarrollar las actividades y alcanzar el propósito de la sesión					
9	Los materiales y recursos educativos seleccionados para la sesión permitirán desarrollar las actividades y alcanzar el logro de los aprendizajes esperados.					
10	Presenta instrumentos de evaluación estructurados por competencias, capacidades y desempeños/indicadores para procesar y sistematizar la información y tomar decisiones.					
11	Gestiona y organiza los espacios del aula: sectores, áreas, equipos de trabajo, responsabilidades, etc.					

III. DESARROLLO DE LA SESIÓN DE APRENDIZAJE

	ASPECTOS	Valoración 1	2	3	4	Evidencias/observaciones
1. INVOLUCRA ACTIVAMENTE	Se promueve el interés de los estudiantes en las actividades de aprendizaje.					
	Los estudiantes se mantienen involucrados en el proceso de aprendizaje con actividades pedagógicas graduales y de alta demanda cognitiva.					
	Las actividades favorecen la comprensión del sentido, importancia y utilidad de lo que se aprende.					

2. PROMUEVE EL PENSAMIENTO	Las actividades e interacciones promueven efectivamente el razonamiento, la creatividad y el pensamiento crítico de los estudiantes.					
3. EVALÚA Y RETROALIMENTA	El docente monitorea el trabajo que realizan los estudiantes y sus avances durante el proceso de aprendizaje.		X			
	El docente brinda retroalimentación y/o la adaptación de las actividades que realiza en la sesión a partir de las necesidades de aprendizaje					
4. RESPETO Y PROXIMIDAD	El docente muestra trato respetuoso y consideración hacia la perspectiva de los estudiantes.					
	El docente muestra cordialidad o calidez a los estudiantes.					
	El docente muestra comprensión y empatía ante las necesidades afectivas o físicas de los estudiantes.					
5. REGULA EL COMPORTAMIENTO	El docente emplea mecanismos para regular el comportamiento y promover el respeto de las normas de convivencia en el aula: formativos, de control externo o de maltrato.					
	El docente implementa eficazmente mecanismos para regular el comportamiento de los estudiantes, lo que se traduce en la mayor o menor continuidad en el desarrollo de la sesión.					

LEYENDA: Planificación				
N° Ítems	No evidencia	En inicio	En proceso	Logrado
11	11--18	19--26	27--35	36-44

RESUMEN: Planificación	
Puntaje	
Valoración	

RESUMEN: Desarrollo de la sesión de aprendizaje			
N°	ASPECTO	VALORACIÓN	TOTAL
1	Involucra activamente		
2	Promueve el pensamiento		
3	Evalúa y retroalimenta		
4	Respeto y proximidad		
5	Regula el comportamiento		
VALORACIÓN FINAL			

** Si desea escribir un salto de línea en una misma celda, pulse "Alt+Enter"*

IV. OBSERVACIONES

V. COMPROMISOS		
COMO DOCENTE DE AULA ME COMPROMETO A:		COMO ACOMPAÑANTE ME COMPROMETO A:

** Si desea escribir un salto de línea en una misma celda, pulse "Alt+Enter"*

ENCARGADO DEL MONITOREO

DOCENTE DE LA IE

Anexo 4. Instrumento 1 - guía o pauta de grupo nominal

Este instrumento permite recabar información de la problemática actual en el proceso de desarrollo de software.
Extraído de:
http://repositorio.uchile.cl/bitstream/handle/2250/133403/ANEXOS.pdf?sequence=2
Descripción la guía o pauta está en:
http://oei.org.ar/ibertic/evaluacion/pdfs/ibertic_guia_grupos_focales.pdf

Adaptado por el investigador

Lugar: **Motivo**: Reunión de trabajo con expertos en proyectos de software **Tema:** Problemas actuales en el proceso de desarrollo de software. **Observador**: _____	**Fecha:** **Hora de inicio**: **Hora de finalización**: **Número de participantes:**
Detalle de la reunión de grupo nominal	
Preguntas: 1) Qué modelo de proceso de software utiliza para hacer un proyecto? 2) Porqué se retrasa la entrega de un proyecto al cliente/usuario? 3) Nivel de participación del cliente/usuario en el proyecto de software? 4) Qué problemas hay en el desarrollo de software?. Participantes: - - - -	
Pregunta: Respuesta:	
Pregunta: Respuesta:	
Pregunta: Respuesta:	
Pregunta: Respuesta:	
Conclusiones:	

Anexo 5. Instrumento 2 - cuestionario sobre proceso de desarrollo de software

Proceso de desarrollo de software

Estimado, estamos recabando información para un trabajo de investigación y en base a su experiencia le pedimos que pueda responder al cuestionario:

*Obligatorio

1. **Ingrese su nombre** *

2. **Ingrese el nombre de la empresa, institución o freelance para la cual trabaja Ud.** *

3. **Ingrese el RUC de la empresa o Institución (opcional)**

4. **Departamento donde trabaja Ud.** *
Marca solo un óvalo.

- Amazonas
- Ancash
- Apurimac
- Arequipa
- Ayacucho
- Cajamarca
- Callao
- Cusco
- Huancavelica
- Huanuco
- Ica
- Junin
- La Libertad
- Lambayeque
- Lima
- Loreto
- Madre De Dios
- Moquegua
- Pasco
- Piura
- Puno
- San Martin
- Tacna
- Tumbes
- Ucayali

Respecto al desarrollo de proyectos de software

5. **En su experiencia, en promedio CUÁNTOS MESES duran los proyectos desarrollados en la empresa/institución** *
Marca solo un óvalo.

- de 0 a 3 meses
- entre 3.01 y 7 meses
- entre 7.01 y 11 meses
- 11.01 a más 24 meses
- 24.01 a más meses

6. **En su experiencia, en promedio cuál es el COSTO de los proyectos desarrollados en la empresa/institución** *
Marca solo un óvalo.

- de 0 a 500 dólares
- entre 500.01 y 1000 dólares
- entre 1000.01 y 2000 dólares
- entre 2000.01 y 4000 dólares
- entre 4000.01 y 8000 dólares
- entre 8000.01 y 16000 dólares
- entre 16000.01 y 32000 dólares
- más de 32000 dólares

7. **En general, LA COMPLEJIDAD (reglas muy variadas, algoritmos complejos, alta de seguridad) de los proyectos desarrollados en su empresa/institución son:** *
Marca solo un óvalo.

	1	2	3	4	5	
Muy fácil	◯	◯	◯	◯	◯	Muy complejo

Sobre los clientes y la relación con los proyectos

8. **El MÁXIMO ALCANCE GEOGRÁFICO de los clientes/usuarios para los que Ud. ha desarrollado un proyecto de software es:** *
Marca solo un óvalo.

- A nivel local
- A nivel nacional
- A nivel internacional

9. **El número de proyectos en los cuales participó Ud. es:** *
Marca solo un óvalo.

- Sólo 1 proyecto
- entre 2 y 4 proyectos
- entre 5 y 10 proyectos
- entre 11 y 20 proyectos
- entre 21 y 50 proyectos
- más de 50 proyectos
- Otro: _____

10. **Cuántos AÑOS DE EXPERIENCIA en el desarrollo de proyectos de software que tiene Ud?** *

Marca solo un óvalo.

- menos de 1 año
- entre 1.01 y 2 años
- entre 2.01 y 4 años
- entre 4.01 y 8 años
- entre 8.01 y 16 años
- más de 16 años
- Otro: _____

Respecto al PROCESO DE DESARROLLO DE SOFTWARE .

11. **El MODELO para el proceso de desarrollo de software que Ud. usa mayormente es:** *

Marca solo un óvalo.

- RUP
- Cascada
- XP
- SCRUM
- Modelo co-participativo
- Modelo iterativo e incremental
- Por prototipos
- Desarrollo rápido de aplicaciones
- Otro: _____

12. **Por lo general el cliente/usuario NO PARTICIPA en el desarrollo del software** *

Marca solo un óvalo.

	1	2	3	4	5	
Totalmente en desacuerdo	◯	◯	◯	◯	◯	Totalmente de acuerdo

13. **El software por lo general VA MEJORANDO (evoluciona) poco a poco en el tiempo** *

Marca solo un óvalo.

	1	2	3	4	5	
Totalmente en desacuerdo	◯	◯	◯	◯	◯	Totalmente de acuerdo

14. **Por lo general los proyectos de software SE RETRASAN en su implementación.** *
Marca solo un óvalo.

	1	2	3	4	5	
Totalmente en desacuerdo	◯	◯	◯	◯	◯	Totalmente de acuerdo

15. **Si los proyectos de software se retrasan, entonces, las razones del retraso son (puede marcar varias opciones):** *
Selecciona todos los que correspondan.

- [ ] Los requisitos no se entendieron bien al inicio
- [ ] No se tuvo un plan de pruebas y el software empezó a fallar al final
- [ ] El nivel técnico de los programadores es bajo y demoran al implementar
- [ ] El cliente/usuario no explicó bien los requisitos
- [ ] El cliente/usuario cambió los requisitos durante el desarrollo del proyecto
- [ ] Se requiere excesiva documentación
- [ ] Otro: _____

Con la tecnología de Google Forms

Anexo 6. Guía o pauta de entrevista

Este instrumento permite recabar información de la opinión de los usuarios del software SIREMAP

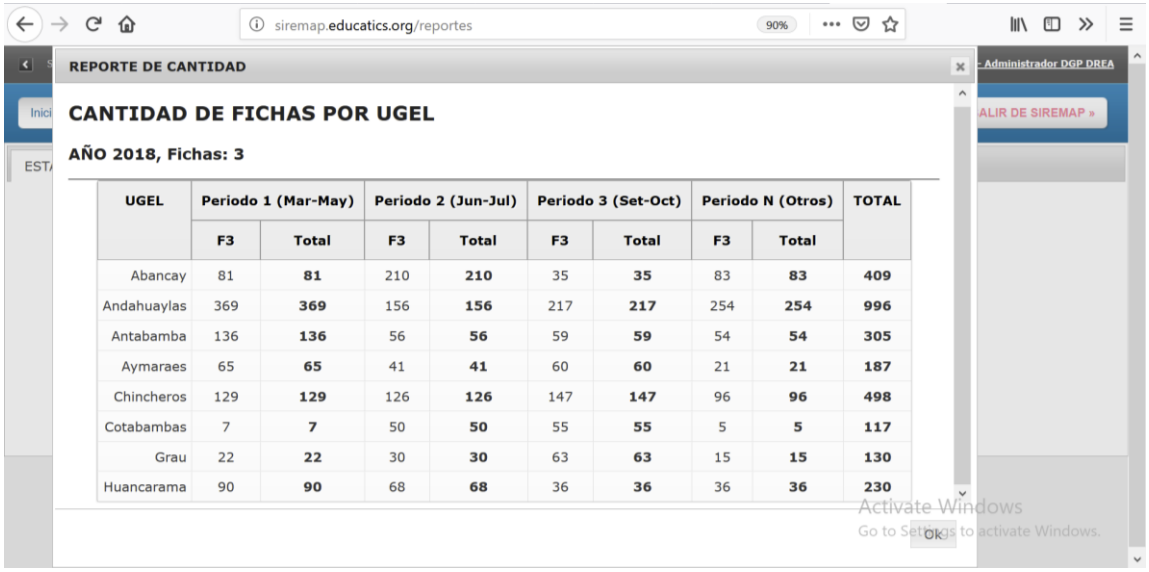
Adaptado por el investigador

Lugar: **Motivo**: Reunión de trabajo con usuarios de software **Tema:** opinión de satisfacción respecto al software SIREMAP. **Observador**: _____	**Fecha:** **Hora de inicio**: **Hora de finalización**: **Número de participantes:**
Detalle de la reunión de grupo nominal	
Preguntas: 1. ¿Cuál es su opinión respecto a que Ud. como cliente/usuario ha participado activamente en el desarrollo del software SIREMAP? 2. ¿Cuál es la diferencia entre utilizar SIREMAP y hacerlo tradicionalmente el monitoreo al docente en aula?. 3. ¿Cree UD. que SIREMAP facilita el proceso de monitoreo a la sesión de aprendizaje del docente en aula?. Participantes: - - - -	
Pregunta: Respuesta:	
Pregunta: Respuesta:	
Pregunta: Respuesta:	
Conclusiones:	

Anexo 7. Artículos científicos publicados como parte del proceso de investigación

- "*mSIREMAP: Cooperative Design for monitoring Teacher's classes in K-12 schools*"
  Paper publicado en CDVE 2017:
  Manuel J. Ibarra, Angel F. Navarro, Vladimiro Ibañez, Wilfredo Soto, Waldo Ibarra. Artículo publicado en: The 14th International Conference on cooperative Design, Visualization and Engineering.
  Conferencia realizada del 17-20 setiembre 2017 - Mallorca, España
  https://link.springer.com/chapter/10.1007/978-3-319-66805-5_15
  Print ISBN: 978-3-319-66804-8
  DOI: https://doi.org/10.1007/978-3-319-66805-5_15

Anexo 8. Resultados de la observación del desempeño docente en aula con el software



UGEL	Periodo 1 (Mar-May)		Periodo 2 (Jun-Jul)		Periodo 3 (Set-Oct)		Periodo N (Otros)		TOTAL
	F3	Total	F3	Total	F3	Total	F3	Total	
Abancay	81	**81**	210	**210**	35	**35**	83	**83**	**409**
Andahuaylas	369	**369**	156	**156**	217	**217**	254	**254**	**996**
Antabamba	136	**136**	56	**56**	59	**59**	54	**54**	**305**
Aymaraes	65	**65**	41	**41**	60	**60**	21	**21**	**187**
Chincheros	129	**129**	126	**126**	147	**147**	96	**96**	**498**
Cotabambas	7	**7**	50	**50**	55	**55**	5	**5**	**117**
Grau	22	**22**	30	**30**	63	**63**	15	**15**	**130**
Huancarama	90	**90**	68	**68**	36	**36**	36	**36**	**230**

Total 2872 fichas aplicadas (14 dic 2018)